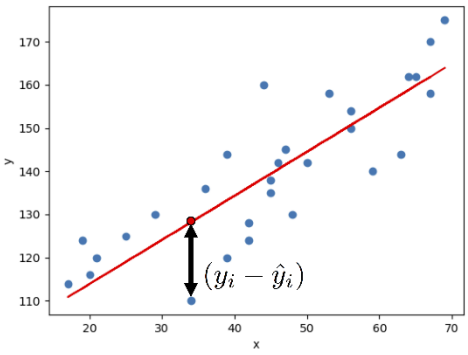
Centro Universitario de Ciencias Exactas e Ingenierías

Universidad de Guadalajara

Práctica 1: Regresión Lineal Simple

Aprendizaje Máquina



Alumno: Samuel David Pérez Brambila

Código: 222966286

Profesora: Karla Ávila Cárdenas

Sección: D01

Fecha de Entrega: 22 de Septiembre de 2024

Introducción

La regresión lineal es “un modelo matemático que describe la relación entre varias variables. Los modelos de regresión lineal son un procedimiento estadístico que ayuda a predecir el futuro. Se utiliza en los campos científicos y en los negocios, y en las últimas décadas se ha utilizado en el aprendizaje automático. La tarea de la regresión en el aprendizaje automático consiste en predecir un parámetro (Y) a partir de un parámetro conocido X.” (EBAC, 2023)

Pero ¿por qué son importantes los modelos de regresión lineal? De acuerdo con EBAC (2023), son importantes debido a que, debido a su capacidad para transformar datos, pueden utilizarse para simular una amplia gama de relaciones, y debido a su forma, que es más simple que la de las redes neuronales, sus parámetros estadísticos se analizan y comparan con facilidad, lo que permite que se les extraiga información valiosa.

La regresión lineal no sólo se utiliza con fines de predicción: también se ha demostrado su eficacia para describir sistemas. Si se quieren modelar los valores de una variable numérica, se tendrá una lista relativamente corta de variables independientes y, como se espera que el modelo sea comprensible, es probable que se elija la regresión lineal como herramienta de modelización.

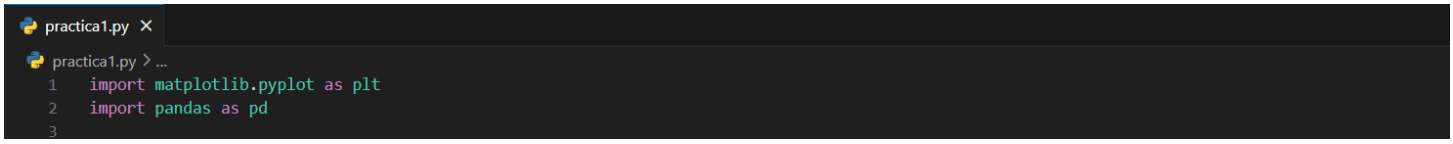
Además, encontramos diversos tipos de regresión lineal, los cuales se enlistan a continuación:

- Simple: En una regresión lineal, se trata de establecer una relación entre una variable independiente y su correspondiente variable dependiente. Esta relación se expresa como una línea recta. No es posible trazar una línea recta que pase por todos los puntos de un gráfico si estos se encuentran ordenados de manera caótica. Por lo tanto, sólo se determina la ubicación óptima de esta línea mediante una regresión lineal. Algunos puntos seguirán distanciados de la recta, pero esta distancia debe ser mínima. El cálculo de la distancia mínima de la recta a cada punto se denomina función de pérdida.
- Múltiple: La regresión lineal múltiple encuentra la relación entre dos o más variables independientes y su correspondiente variable dependiente.

En la presente práctica, se trabajará con una regresión lineal simple utilizando el lenguaje de programación Python y algunas de sus bibliotecas más populares, como Matplotlib y Pandas. Estas herramientas son fundamentales para el análisis de datos y la visualización de resultados, permitiendo gestionar y manipular grandes conjuntos de datos de manera eficiente. Con Pandas, se facilitará la carga y el procesamiento de los datos de origen, mientras que Matplotlib será clave para trazar la gráfica correspondiente que muestre los datos de origen como la recta de regresión.

Contenido de la Actividad

Primero se inicia importando las bibliotecas de matplot (que permite crear gráficas y visualizar los datos) y pandas (que en esta ocasión se utiliza para la manipulación y lectura de los datos del archivo .csv)



Empezamos con el caso donde trabajamos con todos los datos (de las columnas deseadas) del archivo. Donde definimos la variable independiente (x) que hace referencia al salario anual y la variable dependiente (y) que hace referencia al costo del carro, para ello, aplicamos el método **pd.read_csv** colocando como primer argumento el nombre del archivo, luego el argumento **usecols** (el cual selecciona la columna deseada, donde la columna 5 es la que trae todos los registros de salario y en el caso de y la columna 8 que trae todos los datos de costo de carro), después **encoding='latin-1**' que se usa para asegurar que, al leer el archivo CSV, todos los caracteres se interpreten correctamente, especialmente si el archivo contiene caracteres especiales o acentos, por últimos nos encontramos con la aplicación del método **.values.flatten()** que convierte los datos de las columnas seleccionadas en arreglos unidimensionales para facilitar el cálculo (y a la vez para seguir el ejemplo otorgado).

Pasamos a calcular los promedios o medias de ambas variables, utilizando la suma total de los valores dividida por la cantidad de elementos en cada arreglo.

Para calcular la pendiente (**m**) y el intercepto (**b**), primero se calcula el numerador y denominador que forman la fórmula para la pendiente, **numerator** es el numerador de la fórmula para calcular la pendiente m, para cada punto (xi, yi), se resta la media de la variable independiente (mean_x) y la media de la dependiente (mean_y), y se multiplica cada uno de estos valores. Luego, se suman todos estos productos.

El denominador (**denominator**) de la fórmula para m se calcula restando la media de la variable independiente mean_x de cada valor xi, elevando al cuadrado la diferencia y sumando todos esos valores.

Finalmente, la pendiente **m** se obtiene dividiendo numerator entre denominator. El intercepto **b** se calcula con la fórmula: mean_y - m * mean_x, que prácticamente es el valor de y cuando x = 0.

```
# Calcular medias
mean_x = sum(x) / len(x)
mean_y = sum(y) / len(y)

# Calcular la pendiente (m) y el intercepto (b)
numerator = sum((xi - mean_x) * (yi - mean_y) for xi, yi in zip(x,y))
denominator = sum((xi - mean_x)**2 for xi in x)

m = numerator / denominator
b = mean_y - m * mean_x

```

Habiendo obtenido estos valores, pasamos a realizar las predicciones con la ayuda de un bucle for, donde itera sobre los valores de x (salario anual), y para cada valor, se calcula **y_pred**, que es la predicción del costo del carro usando la fórmula de la recta de regresión lineal (y = mx + b).

Para visualizar los puntos tanto de los datos originales como de las predicciones, utilizamos **plt.scatter,** que imprime en la gráfica cada uno de estos valores en forma de punto, es por ello que como argumentos se le otorgan las respectivas coordenadas y un color para poder distinguir entre uno y otro (azul para el dato original y rojo para la predicción). **plt.xlabel** y **plt.ylabel** etiquetan los ejes *x* y *y* con el salario anual y el costo del carro, respectivamente. Si queremos visualizar una simbología en nuestra gráfica, debemos agregar un argumento que se llama **label,** que en resumen es solamente definir como deseamos etiquetar a ese o esos datos que está imprimiendo, pero si agregamos ese argumento dentro de los scatter del bucle, se mostrarán repetidos (ya que lo imprimirá cada vez que agrega un punto), es entonces, que para poder mostrar la simbología de manera correcta y sin repeticiones se agregan 2 plt.scatter con datos vacíos, donde no altera a la gráfica, solamente al cuadro de simbología. Además, como en el bucle las predicciones se van imprimiendo como puntos, se agrega la variable **y_pred_line** que lo que hace es que obtiene la recta de la regresión lineal, como en este caso queremos visualizar la línea y no puntos u otra figura, utilizamos el método **plt.plot** que funciona igual que scatter con la diferencia que imprime líneas en vez de puntos. **plt.legend()** permite observar el cuadro de simbología junto a las etiquetas que definimos previamente y por último, **plt.show()** muestra el gráfico final.

```
# Realizar predicciones
for xi in x:
    y_pred = m * xi + b
    # Visualizar resultados
    plt.scatter(x,y,color='blue')  # Datos
    plt.scatter(xi,y_pred,color='red') # Predicciones
    plt.xlabel('Variable independiente (salario anual)')
    plt.ylabel('Variable dependiente (costo del carro)')

# Agregamos espacios vacíos, para poder agregar la leyenda, esto permite que no se repita la impresión si lo ponemos en el bucle
plt.scatter([], [], color='blue', label='Datos')  # Espacio vacío para la leyenda de Datos
plt.scatter([], [], color='red', label='Predicciones')  # Espacio vacío para la leyenda de Predicciones

# Graficar la recta de la regresión lineal
y_pred_line = m * x + b
plt.plot(x, y_pred_line, color='black', label='Regresión lineal') # Línea de regresión
plt.legend()
plt.show()
```

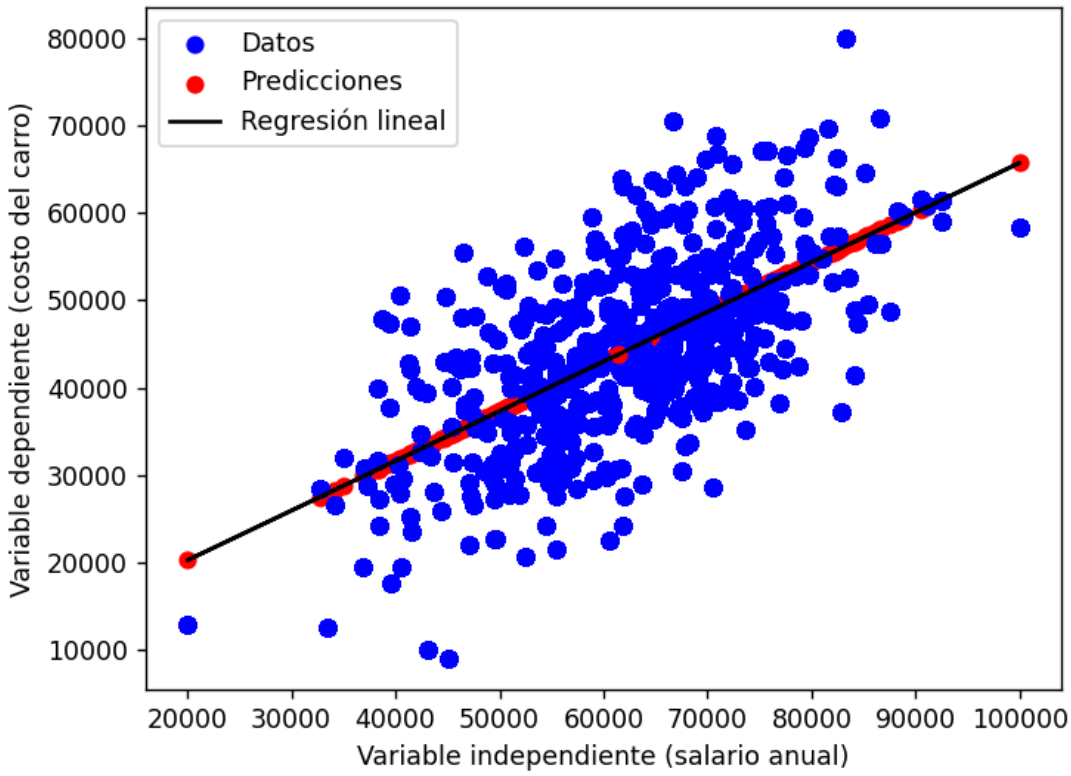
Previo a mostrar el gráfico, se explicarán unos cálculos adicionales que son necesarios para terminar de evaluar la regresión realizada. Primero, **distancias** calcula la distancia absoluta entre cada punto de los datos originales y la línea de regresión esto se hace con **abs(yi - m * xi – b)** donde se calcula la distancia de cada punto (xi, yi) a la línea de regresión. Se usa la ecuación de la recta y = m * x + b, y se resta el valor predicho m * xi + b del valor real yi, el valor absoluto asegura que siempre obtenemos una distancia positiva, pero antes de hacer los cálculos, primero con el **for xi, yi in zip(x, y)** empareja los valores de x (salario anual) con los de y (costo del carro), creando pares (xi, yi) donde xi es un salario y yi es el costo correspondiente y es entonces que hace el cálculo por cada valor dentro de este zip, guardando los resultados en la lista distancias.

```
40  # Calcular la distancia entre cada punto y la línea de regresión
41  distancias = [abs(yi - m * xi - b) for xi, yi in zip(x,y)]
42
43  # Calcular el error cuadrático medio (MSE)
44  mse = sum((yi - m * xi - b)**2 for xi, yi in zip(x,y)) / len(x)
45
46  # Imprimir información
47  print('Coeficiente (pendiente):', m)
48  print('Intercepto:', b)
49  print('Distancias:', distancias)
50  print('Suma de las diferencias cuadradas (SSD):', sum(d**2 for d in distancias))
51  print('Error Cuadrático Medio (MSE):', mse)
```

Después, mse calcula el Error Cuadrático Medio (MSE), que mide qué tan cerca están las predicciones de los valores reales en promedio. Se empieza como en el cálculo creando una lista de tuplas con el **for xi, yi in zip(x,y)**, luego con **(yi - m * xi - b)**2** se calcula la diferencia entre el valor real y el valor predicho. Luego, se eleva al cuadrado para penalizar errores grandes. Por último, se suman todos los resultados con **sum()** y se divide entre el tamaño del arreglo x, para obtener el promedio de estos errores, que en sí es el MSE.

Al final, simplemente se imprimen los valores previamente cálculos y se agrega la impresión de un nuevo cálculo, donde se hace la sumatoria de los cuadrados de las distancias entre cada punto y la línea de regresión (SSD).

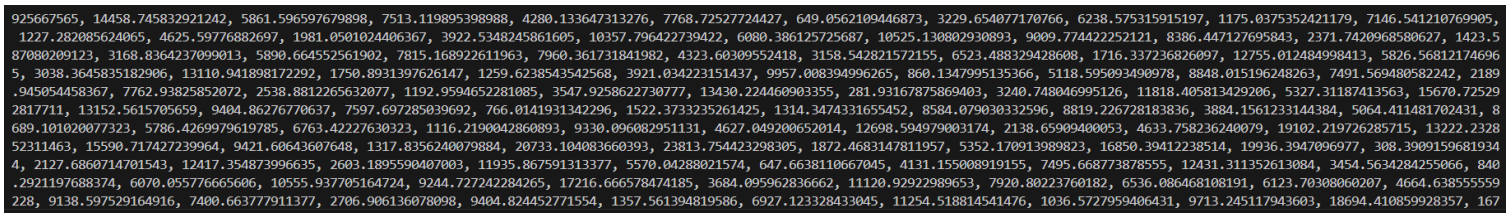
Y es así, que obtenemos los siguientes resultados:

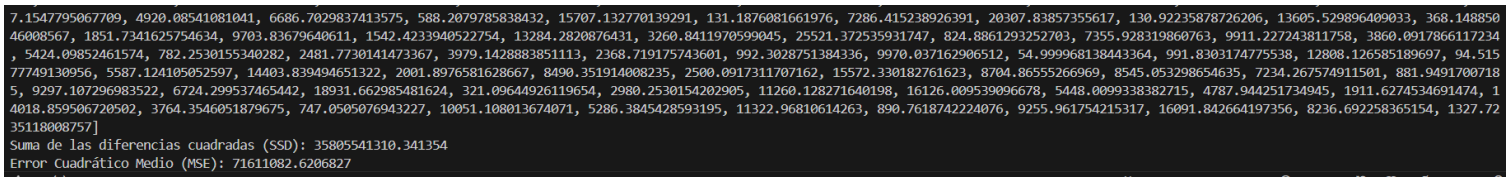



```
Coeficiente (pendiente): 0.5687532549167685
Intercepto: 8874.729472626306
Distancias: [9277.85305019861, 1664.8410909656645, 3452.8786780675327, 13405.666213381155, 13069.583792082558, 8777.754797183014, 2593.662963081595, 9126.001384720468, 6071.991775799339, 1089.1238367917
94, 8453.467511248367, 1892.8329880544115, 1409.8600074249152, 1535.7549046242202, 756.284965086481, 7365.643815029645, 3039.2705312026883, 5063.887585952649, 4760.4222533631, 14461.92945362012, 6176.8
70734919357, 2070.176312942065, 1296.6756723582075, 8169.876178803632, 3388.3861195574354, 5331.637990541869, 18830.136315366384, 5302.260905064482, 7399.736874303162, 650.3229976641232, 13828.21605561
9305, 3388.47040596624, 10647.654108637667, 5993.884741394395, 6210.613166403542, 8843.551676264615, 3110.603901052371, 23313.846985556163, 2671.3538182385128, 97.53522979316767, 844.3213170384988, 555
.1041523963795, 9606.201927205962, 10033.080864380361, 14421.671527128645, 19789.8704329267, 6565.176840037304, 3722.27855134166, 5219.857898282418, 2481.2549106709994, 3531.4708428699705, 18112.709498
557073, 655.41799522558, 3712.1792156032316, 3158.404302709183, 9409.839359982245, 9535.147662501986, 3305.259276968125, 2763.7666244026477, 7536.348447615659, 1918.6506340859632, 2254.2361094594307, 1
477.2133497231698, 502.3592722422072, 6561.493258680315, 4160.830792392328, 3817.127775721223, 1138.3633818147864, 16125.472514846279, 2708.5758471727822, 6630.6441492310777, 7855.719180738182, 3047.057
7112111787, 2548.5487916647835, 6001.687640245909, 2749.8256937481856, 3551.1430116500815, 5159.6161393918155, 10448.929697301668, 2374.8430280302564, 10888.269054851888, 1069.345806138459, 6102.586304
536886, 6022.795278205907, 9374.67974925557, 10325.144292340909, 411.1175544013531, 1798.2519453191671, 9038.405085170212, 7354.079890961677, 4349.673382057001, 13528.895093922682, 8142.121066775566, 8
```

```
594.0862745455552, 4877.646052989567, 725.5747123255605, 1214.4148264654214, 10160.309316348532, 4639.191210396537, 7001.616337405325, 13384.667323673784, 14337.10367100787, 14073.68720566835, 4292.41615
1731719, 15200.489087000205, 6054.813599225003, 3120.2403151263643, 1633.6554141501547, 16506.86021212743, 6315.746113158246, 1522.1721834534546, 6984.443120151482, 1467.9302308445185, 14259.7957187451
25, 9021.162591568012, 15449.09041212625, 5810.31998668782, 15295.636352082314, 7713.809303086193, 5268.568431724369, 10572.472028680255, 616.9226011755491, 13112.307261580128, 5292.6654802633275, 1119
1.392239499117, 3114.598296418124, 17464.25209036652, 8665.788052237076, 3772.0991376104575, 1460.4404315700522, 3287.382167211057, 5083.36476486981, 5354.229096126124, 5376.941178808702, 4987.93509749
5138, 9108.917906366787, 6459.157225443036, 656.7304580248747, 5381.501351316969, 275.4917894217069, 16669.65237573797, 7544.596123240266, 782.1908410187461, 6849.851738758887, 6054.110665310138, 6319.
832365847149, 4107.036045819339, 17005.422827107555, 1130.7185084420998, 9010.929339470247, 10586.701347728489, 8941.85036590687, 8473.900908169322, 9183.695037366979, 453.8632166290481, 8535.930832091
792, 1801.055620706189, 1632.2584458200108, 6933.308695491469, 6380.399338111682, 6937.675265859558, 668.3751986398056, 5555.762001359202, 5942.092781044492, 700.9280028485082, 7320.323797605321, 18068
.35812768289, 504.05590071723054, 132.2592410022262, 14438.88629718257, 5224.020950176855, 3931.7332432720286, 5616.011614603616, 2056.099449467343, 4664.060586110172, 333.8552314509434, 2896.45292128
0943, 7220.340904254015, 6626.505690443944, 12949.581218827567, 2968.6994758556757, 3914.654492382877, 9822.65346222799, 13540.742192753765, 6392.472021051915, 10653.509371741595, 6523.454577425873, 87
26.993648677195, 10911.943225306437, 9660.181072399198, 1246.1875460790543, 3384.6530804979775, 10380.38272258985, 2588.862580077701, 17595.667439566812, 5434.498585318346, 1875.9485107372384, 1048.106
2571371076, 8896.929807420565, 1362.7788350408737, 4022.809502344149, 1820.9186748018983, 9335.178252321253, 14233.868233318408, 1793.3339648560286, 5301.611397469736, 2137.750603295186, 8752.658829140
```

```
302, 840.3542738051001, 265.5037606059486, 11178.302471247946, 2223.574370271017, 8506.63015864039, 5871.1587529925455, 9378.308000967154, 4957.780717612397, 5158.108287407529, 1151.971526028814, 12249
.60700168852, 2524.4669548730308, 8073.16163929164, 11096.498052896102, 3287.5540399187958, 2441.680791216022, 1546.6210383498692, 8101.077945926543, 2239.8015815568288, 16088.566765799682, 1358.964959
4640869, 3086.002062634092, 7562.517336030236, 8483.685023308775, 1506.86973104535, 9068.591971216534, 2671.4784464024815, 876.0483016768558, 14568.834041608898, 6170.050960256514, 2982.4922471019017,
236.57948431358818, 2979.553561878536, 6054.480635977998, 1690.1367559620994, 10445.712779750378, 437.9411209586251, 6833.992544103596, 11675.4729407888, 17579.037607940692, 8530.670505479931, 2742.391
926397482, 14446.454719121386, 19890.57569055081, 2659.2400253302185, 3309.100087439143, 4432.587543242975, 10921.7744973872870, 2949.015190510414, 14159.653546139816, 3130.802701553752, 443.33048412598
873, 18.117052174464334, 5690.669904817405, 7037.08450160981, 12304.59752304028, 3653.9796427290785, 11308.317939621942, 2465.83833589142 52, 8073.8411049620 92, 811.6615615463234, 17681.29793606575, 153
47.22928876164, 312.9718455796319, 1647.9941373425318, 14150.39447004466, 10247.872833708745, 9510.844392262923, 1169.0026446876582, 5579.237282473718, 8831.577087098238, 5111.053869194468, 6607.438848
113692, 8492.2515478899, 5360.197904247678, 13047.699044792666, 12918.1147567972, 16116.51677171051, 4569.614145294006, 2032.4597386721 07, 3067.0137150612063, 12769.35269680703, 20193.66736253404, 5097
.859558897129, 6932.867099919495, 5103.156620209491, 2169.6213624426528, 13112.067818220909, 5225.9316096071925, 5040.189375170456, 10695.609035481 92, 7283.619846941299, 7682.741773387832, 8692.3844141
41932, 6462.618375500271, 15151.944844707716, 13545.807764068632, 7258.092424364528, 4871.087680726574, 10545.070182070609, 11197.600105336518, 9858.014435750993, 11103.99401852251, 795.0327483355759,
9283.930524097414, 5196.713708649728, 4160.834400692307, 23728.894538131375, 9847.675809200875, 369.5985807857651, 9298.119603706451, 11292.104379335542, 1267.0513876995392, 5013.500520532471, 17286.00
```





Caso 60% de los datos

Para hacer el caso donde se toma el 60% de los datos, se hacen algunas modificaciones al inicio del código, el resto se mantiene igual, ya que las fórmulas no cambian.

En este caso, primero hacemos la lectura del archivo en su totalidad, esto con el fin de poder obtener el total filas que tiene en cada una de sus columnas. Para ello, se define la variable **full_file**, donde se vuelve a utilizar el método **pd.read_csv** con la diferencia de que no se le agrega el argumento de **usecol** y no se utiliza **.values.flatten().**

Procedemos a calcular el total de filas que tiene el archivo con la variable **total_filas** y con la aplicación de **len(full_file)** donde el resultado se guarda en la variable antes mencionada**.**

Agregamos una impresión (print), solamente para mostrar en la terminal cuántas filas tiene el archivo.

Agregamos las variables **porcentaje_tomar** y **porcentaje_omitir** que hacen referencia al porcentaje de datos que vamos a tomar para esta regresión (60%) y el restante (40%) que no se tomará.

Definimos filas_a_omitir y filas_a_tomar donde se obtiene la cantidad de filas que corresponde a cada uno de los porcentajes, dicho resultado se pasa a entero (int) para evitar el uso de decimales (float). Así como con el total de filas, se agregan unas impresiones solamente para mostrar la cantidad de filas que se tomarán y las que se descartarán.

Por último, pasamos al cambio más relevante, donde definimos a la variable independiente (x) y la variable dependiente (y), a diferencia del caso anterior aquí se le agrega la definición de otro argumento: **skiprows**, con el cual se define cuál es el intervalo de filas que no se deben tomar del archivo original, para lo cual se le asigna el rango (range) de la fila 1 hasta la cantidad de filas que se calcularon en

filas_a_omitir + 1, se le agrega un + 1 ya que en Python range(), el límite superior siempre toma uno menos de lo que se define. Por ejemplo, si definimos range(1,10) solamente tomaría del 1 al 9 y si solamente lo definimos como range(1,filas_a_omitir) en realidad tomaría una menos de la cantidad que se calculó.

Es entonces, que se debe dejar en claro que se están tomando el 60% de las filas desde la siguiente fila a la última fila que se está descartando hasta el último registro.

Los demás argumentos quedan iguales que como el primer caso, donde los valores de las filas se agregan con ayuda de values.flatten() a un arreglo unidimensional, lo que facilita el cálculo de los diferentes parámetros necesarios para la regresión lineal.

practica1.1.py > ...

```python
import matplotlib.pyplot as plt
import pandas as pd

# Regresión lineal con el 60% de los datos

# Leer el archivo completo para obtener el total de filas
full_file = pd.read_csv('car_purchasing.csv', encoding='latin-1')

# Calcular el total de filas
total_filas = len(full_file)
print(f"Cantidad de filas del archivo: {total_filas}")

# Definir que se omite el 40% y se va a tomar el 60%
porcentaje_omitir = 0.4
porcentaje_tomar = 0.6

# Calcular el número de filas a omitir y a tomar
filas_a_omitir = int(total_filas * porcentaje_omitir)
filas_a_tomar = int(total_filas * porcentaje_tomar)
print(f"Cantidad de filas a omitir: {filas_a_omitir}")
print(f"Cantidad de filas a tomar: {filas_a_tomar}")

# Leer las columnas del CSV omitiendo las primeras filas_a_omitir y tomando filas_a_tomar
x = pd.read_csv('car_purchasing.csv', usecols=[5], skiprows=range(1, filas_a_omitir + 1), encoding='latin-1').values.flatten()  # Variable independiente
y = pd.read_csv('car_purchasing.csv', usecols=[8], skiprows=range(1, filas_a_omitir + 1), encoding='latin-1').values.flatten()  # Variable dependiente (
```

El resto del código queda igual, el cual ya se explicó previamente:

```python
# Calcular medias
mean_x = sum(x) / len(x)
mean_y = sum(y) / len(y)

# Calcular la pendiente (m) y el intercepto (b)
numerator = sum((xi - mean_x) * (yi - mean_y) for xi, yi in zip(x,y))
denominator = sum((xi - mean_x)**2 for xi in x)

m = numerator / denominator
b = mean_y - m * mean_x

# Realizar predicciones
for xi in x:
    y_pred = m * xi + b
    # Visualizar resultados
    plt.scatter(x,y,color='blue')  # Datos
    plt.scatter(xi,y_pred,color='red') # Predicciones
    plt.xlabel('Variable independiente (salario anual)')
    plt.ylabel('Variable dependiente (costo del carro)')

# Agregamos espacios vacíos, para poder agregar la leyenda, esto permite que no se repita la impresión si lo ponemos en el bucle
plt.scatter([], [], color='blue', label='Datos')  # Espacio vacío para la leyenda de Datos
plt.scatter([], [], color='red', label='Predicciones')  # Espacio vacío para la leyenda de Predicciones

```

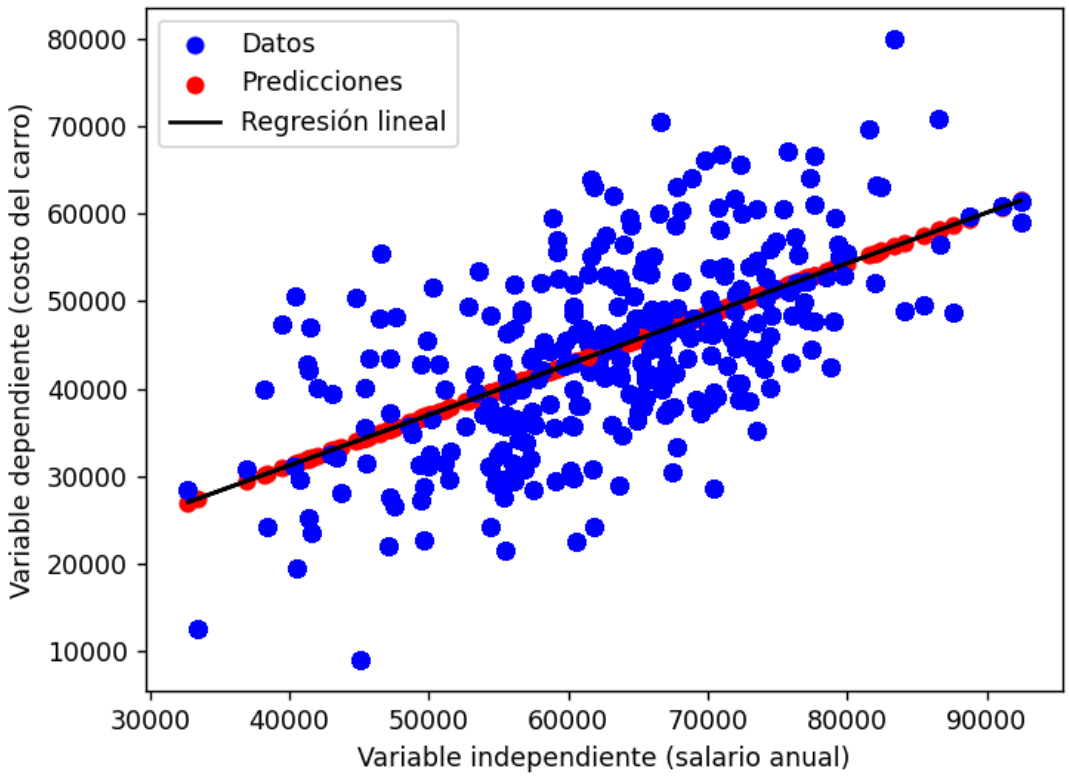
```python
# Graficar la recta de la regresión lineal
y_pred_line = m * x + b
plt.plot(x, y_pred_line, color='black', label='Regresión lineal') # Línea de regresión
plt.legend()
plt.show()

# Calcular la distancia entre cada punto y la línea de regresión
distancias = [abs(yi - m * xi - b) for xi, yi in zip(x,y)]

# Calcular el error cuadrático medio (MSE)
mse = sum((yi - m * xi - b)**2 for xi, yi in zip(x,y)) / len(x)

# Imprimir información
print('Coeficiente (pendiente):', m)
print('Intercepto:', b)
print('Distancias:', distancias)
print('Suma de las diferencias cuadradas (SSD):', sum(d**2 for d in distancias))
print('Error Cuadrático Medio (MSE):', mse)
```

Es así, que obtenemos los siguientes resultados:



```
Cantidad de filas del archivo: 500
Cantidad de filas a omitir: 200
Cantidad de filas a tomar: 300
Coeficiente (pendiente): 0.5780256694682518
Intercepto: 8103.132604183469
Distancias: [3760.4386892075854, 2061.5511621174737, 9233.97649325717, 14539.605180187904, 1632.182510979459, 5513.319672093567, 2003.0867470930389, 8917.40716407582, 467.2704169
1051636, 497.9740470106335, 11378.103979466381, 2393.8149159236345, 8853.947710921508, 5777.34302076749, 9230.272288736654, 4901.697256229629, 5057.05704522349, 1014.897766410285
8, 12443.429367612101, 2802.501120296816, 8124.643220526406, 10969.284831863093, 3492.9461547503815, 2477.3361549308975, 1746.1707233667476, 7767.247235959396, 2116.3256543126627
, 15907.399541654522, 1460.1474121556603, 2904.9420848119553, 7743.663956406635, 8430.101891505401, 1696.887729778573, 9183.304703868009, 2930.4407314477103, 980.0122657465035, 1
```

```
4956.232932671326, 6381.553340080711, 2815.819031323059, 48.58030692455213, 3189.1641911965926, 6305.724472651807, 1722.5169543941156, 10834.948977595112, 381.67650450549263, 658
5.705754887742, 11791.474400440333, 17703.322926661887, 8262.374698075047, 2622.965030406689, 14669.421293746229, 19692.844999761117, 2818.728167058165, 3051.510622405127, 4296.1
48238575923, 10657.84127275549, 2869.050087464806, 14016.47293637351, 2939.32126646908, 562.5108528547935, 59.80218754988164, 5376.633360159958, 6946.324734126625, 12634.04428232
5904, 3749.2481520159927, 11206.911037565602, 2194.2224249220963, 7907.853147230431, 1067.5328383351953, 17795.345236205038, 14885.544302479475, 589.5847217632436, 1731.824562855
4505, 14425.109272955375, 10481.561783821897, 9380.638545610047, 1321.3884692209176, 5616.708562437772, 8644.746170241204, 4947.081701561954, 6788.082547375234, 8682.565016749017
, 5066.500636891858, 12823.731783122428, 13108.203445173618, 16473.194774184933, 4405.635851562649, 2225.211999526029, 2976.9236193454126, 12738.281549302352, 20533.641091927308,
 5219.353165471548, 6780.182284802708, 4999.411685407409, 2053.9300066416545, 12873.224583213781, 5016.7770942553325, 5341.726500370314, 10437.423237174371, 7290.931948673118, 76
93.128048380146, 8607.785496474433, 6376.584357145672, 15221.441624369581, 13210.516410879467, 7640.21991446647, 4952.64483676591, 10827.21154840594, 11156.715849303873, 9707.043
192422323, 11355.722980136827, 473.9970770523323, 9701.192377300438, 4948.496132815948, 4011.2367954799847, 23727.785769092567, 9937.730508123175, 317.65642853248573, 9179.128020
```

```
874297, 11436.443288186172, 1235.225381444936, 5185.493074449616, 17512.036067751025, 14147.127808998637, 5711.549746300669, 7679.732756011552, 4029.4085771053797, 8102.413680683
574, 251.1422357119336, 2879.9961023645046, 5930.76034389095, 1017.5223526614718, 6758.469493557379, 981.314407864822, 4765.607813390969, 2278.948196688554, 3817.8234592106237, 1
0112.937767950018, 6453.085134206929, 10345.811361078384, 8623.793096071844, 8075.336605831781, 2457.5763074630886, 1544.8461983609814, 2885.568645289375, 6146.547624789324, 7552
.742965869718, 8206.27963796891, 4505.835603083433, 2969.1509903844635, 6108.451595070372, 1936.3527429386086, 12894.697727358493, 6140.168280241964, 2936.8128125950607, 13284.51
9996682677, 1483.3697903240136, 1004.1078528179278, 4010.9935264528685, 9840.243572689884, 740.1632343813253, 5037.052266762919, 8580.791813963886, 7279.881295200881, 2342.017932
5896897, 7771.345757026385, 2270.1887911622616, 1367.1662262125974, 3659.7885907237287, 13217.98791744318, 208.96235862538742, 3029.0216895866106, 11723.214136945579, 5405.011575
907745, 15814.093976544682, 13375.176879793871, 9234.804833171387, 7782.296072179299, 682.2098741972004, 1856.025496303726, 1474.1415234968954, 8289.350048485081, 8891.8590133068
06, 3620.1534311436844, 5129.030845761721, 8900.553570594791, 5548.926740812516, 6653.615487877629, 1151.1780504667695, 8999.582801827728, 4738.508309335135, 13039.832927292151,
2378.3155631587288, 4472.528467735654, 19300.550679627326, 13022.95728406973, 15691.822325456225, 9276.349803996964, 1122.757523169843, 20523.275102424654, 23967.29465917791, 169
1.6993275814748, 5144.920803671863, 16704.760555341763, 20136.442134775076, 494.22193597500154, 2355.4218977764103, 12021.835591525982, 2544.36326810436, 12040.325310623382, 5425
.538000244465, 498.44366229884326, 3891.576651322932, 7252.0883537958, 12217.673291945903, 3443.279802440862, 1269.1734071859937, 5895.717670816302, 10434.116669604082, 9140.9011
```

```
53362822, 17402.375475774243, 3618.0078280985545, 11039.6240039697, 7941.770956022607, 6238.419082939385, 6085.22925975878, 4407.799719744053, 9070.793785118316, 7646.62657133353
4, 2924.002756745598, 9164.57248161179, 988.5065890722253, 6702.12540051758, 11002.978764279575, 1066.2297072791844, 10100.87784706143, 19091.514184768363, 1357.5576759591495, 47
99.691653357055, 6517.4550814999675, 352.34484828431596, 15440.39247314239, 86.73195876481623, 7123.13900281745, 20189.64367593321, 66.9857729606083, 13657.053652943214, 157.4208
7692191853, 1995.3106484377422, 9390.88195772162, 1703.4273424508865, 13439.213491324219, 3492.2218869450226, 25167.894252800026, 1004.4785337227586, 7097.036400851459, 9951.8754
84428696, 3753.821342524905, 5775.023046806455, 1133.2284042437677, 2087.8174075583665, 3751.211689058917, 2215.4094511625008, 924.4109227049339, 10191.803805491145, 140.69093801
98631, 1460.237945435696, 12550.39539686271, 38.013278932681715, 5221.102447787518, 14419.12297174435, 2167.4118345478128, 8321.639623150098, 2731.63369059869, 15482.80168186732,
 8863.666868570595, 8811.843929841241, 7019.959485859865, 575.3957463024708, 9499.091077702687, 6885.3402721288585, 18674.069546383453, 508.7203400435028, 3042.293821320338, 1143
8.697094751697, 16531.716897069982, 5296.954901262747, 4480.442305762677, 1710.9883216822054, 14193.034019638042, 3583.6145520015125, 703.507081077536, 10151.152102323533, 5234.9
313449491165, 11377.384356233728, 786.2447893909266, 9003.98634183643, 16224.674224615475, 8546.411700144756, 1530.266015674737]
```

```
Suma de las diferencias cuadradas (SSD): 22846887668.282864
Error Cuadrático Medio (MSE): 76156292.22760954
```

En resumen, se puede observar que si hay modificaciones en los valores de los cálculos realizados. Por ejemplo, cuando se toma el 60% de los datos (las últimas 300 filas), el error cuadrático medio aumenta, la suma de las diferencias cuadradas disminuye (esto por ser menos datos), la pendiente cambia al igual que el intercepto. Por lo tanto, esto indica que el conjunto de datos utilizado para el modelo de regresión tiene un impacto significativo en los resultados de la regresión lineal. Es fundamental considerar el tamaño y la representatividad de la muestra al realizar este tipo de análisis para obtener conclusiones precisas y generalizables.

Conclusiones

En conclusión, la implementación de la regresión lineal simple en Python con el uso de bibliotecas como Pandas y Matplotlib permitió analizar de manera eficiente la relación entre la variable independiente y la variable dependiente de la base de datos disponible en el archivo .csv. Los resultados obtenidos en ambos casos nos dejan en claro que la cantidad de datos, así como cuáles, alteran el resultado final de la regresión lineal. En este caso en particular, pudimos ver que al disminuir la cantidad de datos y tomar el último 60% de los mismos, el error cuadrático medio aumentaba, lo cual sugiere que la muestra seleccionada podría no ser representativa del conjunto de datos completo. Por lo tanto, es crucial considerar el tamaño y la representatividad de la muestra para obtener conclusiones precisas y evitar interpretaciones sesgadas en el análisis de regresión.

Además, es necesario mencionar que la regresión lineal simple presenta limitaciones, especialmente cuando se trata de modelar relaciones no lineales o datos con alta variabilidad. A pesar de estas limitaciones, esta técnica sigue siendo útil para problemas con una relación lineal clara entre variables. Para futuros análisis, sería recomendable explorar modelos más complejos o incluir más variables para obtener predicciones más precisas. En resumen, la práctica proporcionó una valiosa introducción al uso de herramientas de análisis de datos en Python.

Referencias

- Bobadilla, J. (2020). *Machine Learning y Deep Learning Usando Python, Scikit y Keras* (1ª ed.). Ra-Ma Editorial; Ediciones de la U. https://elibro-net.wdg.biblio.udg.mx:8443/es/lc/udg/titulos/222698
- EBAC (2023, 3 de mayo). *Regresión Lineal: teoría y ejemplos*. Escuela Británica de Artes Creativas y Tecnología. Recuperado el 12 de Septiembre de 2024 de: https://ebac.mx/blog/regreson-lineal