Práctica

- Análisis de regresión lineal múltiple mediante matrices que permitan construir un modelo predictivo a una base de datos real, de manera que puedas generar pronósticos adecuados.

Paso a paso:

- Considera la base de datos del archivo de Kaggle "kc_house_data.csv" donde el objetivo es el de pronosticar el precio de una casa a partir de diversas variables que la definen (como número de habitaciones, baños, etc.).
- Selecciona variables independientes adecuadas a partir del archivo arriba mencionado.
- Construye un modelo de regresión lineal múltiple mediante matrices para pronosticar el precio de una casa.
- Asegúrate de eliminar aquellas variables que no sean relevantes y comenta sobre la bondad del ajuste y la ecuación final resultante, comparando contra el reporte automatizado disponible.

In [45]:
```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
import os
os.chdir('C:/Users/Isaac/Desktop/IHD/EBAC DT/CIENCIA DE DATOS/M51 DS')
df = pd.read_csv('kc_house_data.csv')
```

In [46]:
```
df
```

Out[46]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_above	sqft_basement	yr_l
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	0	0	...	7	1180	0	1
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	0	0	...	7	2170	400	1
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	0	0	...	6	770	0	1
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	0	0	...	7	1050	910	1
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	0	0	...	8	1680	0	1

In [48]:
```
# agregamos el el intercepto con valor de 1

df['Intercepto'] = 1
```

In [49]:
```
# revisamos las columnas y dejamos solo las que nos interesan para el analisis.

df = df[['price','bedrooms','bathrooms','floors','waterfront','view','condition','sqft_above','sqft_basement','yr_built','yr_renc
print(df.shape)
df.head()
```

```
(21613, 11)
```

Out[49]:

	price	bedrooms	bathrooms	floors	waterfront	view	condition	sqft_above	sqft_basement	yr_built	yr_renovated
0	221900.0	3	1.00	1.0	0	0	3	1180	0	1955	0
1	538000.0	3	2.25	2.0	0	0	3	2170	400	1951	1991
2	180000.0	2	1.00	1.0	0	0	3	770	0	1933	0
3	604000.0	4	3.00	1.0	0	0	5	1050	910	1965	0
4	510000.0	3	2.00	1.0	0	0	3	1680	0	1987	0

In [50]:
```
# asignamos columnas para X y Y

Xdata = df[['bedrooms','bathrooms','floors','waterfront','view','sqft_above','sqft_basement','yr_built','yr_renovated']].values
Ydata = df[['price']].values
```

In [51]:
```
# dividimos las base en Trian y Test

from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(Xdata, Ydata, test_size = 0.3, random_state= 1)
```

In [52]:
```
X = Xdata
Y = Ydata
```

In [53]:
```
# Opcion de formato numerico para no usar Notacion Cientifica para las matrices
np.set_printoptions(formatter = {'float_kind':'{:f}'.format})
```

- Vamos a calcular el Vector b 'Minimos Cuadrados'

In [54]:
```
# primero calculamos (X^t X)
XT_X = np.matmul(np.matrix.transpose(X), X)
XT_X
```

Out[54]:
```
array([[264274.000000, 162054.750000, 110770.500000, 538.000000,
        18295.000000, 138241089.000000, 23933912.000000,
        143686638.000000, 6301162.000000],
```

In [55]:
```
# ahora obtenemos la inversa (X^t X)^-1
XT_X_inv = np.linalg.inv(XT_X)
XT_X_inv
```

Out[55]:
```
array([[0.000084, -0.000022, 0.000011, 0.000030, 0.000009, -0.000000,
         -0.000000, -0.000000, 0.000000],
        [-0.000022, 0.000223, -0.000097, 0.000007, 0.000002, -0.000000,
         -0.000000, -0.000000, -0.000000],
        [0.000011, -0.000097, 0.000281, -0.000006, -0.000002, -0.000000,
         0.000000, -0.000000, -0.000000],
        [0.000030, 0.000007, -0.000006, 0.007418, -0.000336, -0.000000,
         0.000000, -0.000000, -0.000000],
        [0.000009, 0.000002, -0.000002, -0.000336, 0.000106, -0.000000,
```

In [56]:
```
# ahora calculamos X^t Y
XT_Y = np.matmul(np.matrix.transpose(X), Y)
XT_Y
```

Out[56]:
```
array([[41623028485.000000],
       [27894385715.750000],
       [18543180450.500000],
       [270885792.000000],
       [5150643758.000000],
       [24854549271975.000000],
       [4539858275073.000000],
       [23019983014570.000000],
       [1388173928583.000000]])
```

In [57]:
```
# por ultimo calculamos 'b'   b = (X^t X)^-1 X^t Y
betas = np.matmul(XT_X_inv, XT_Y)
betas
```

Out[57]:
```
array([[-41553.790986],
       [6176.747052],
       [15965.790244],
       [539684.939906],
       [71215.557914],
       [275.042410],
       [278.536953],
       [22.211812],
       [57.399629]])
```

- Obtenemos la determinación de la Bondad de Ajuste

In [58]:
```
# Calculamos TSS (Suma Total de Cuadrados)  Y^t Y - nY^2
TSS = np.matmul(np.matrix.transpose(Y), Y) - len(Y) * (Y.mean() **2)
TSS
```

Out[58]:
```
array([[2912916761921300.000000]])
```

In [60]:
```
# Calculo de RSS (Residuales al cuadrado)
RSS = TSS - ESS
RSS
```

Out[60]:
```
array([[1273998093209969.000000]])
```

- Calculo del coeficiente de Determinación de R Cuadrada

In [61]:
```
RSq = 1 -RSS / TSS
RSq
```

Out[61]:
```
array([[0.562638]])
```

- Calculo de Coeficiente de Determinacion de R Cuadrada Ajustada

In [62]:
```
RSq_Aj = 1 -(RSS / (X.shape[0] - X.shape[1])) / (TSS / X.shape[0] -1)
RSq_Aj
```

Out[62]:
```
array([[0.562456]])
```

- Reporte Automatizado de regresión en Python

In [63]:
```
import statsmodels.api as sm
regressor = sm.OLS(Y, X).fit()
print(regressor.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared (uncentered):                   0.862
Model:                            OLS   Adj. R-squared (uncentered):              0.862
Method:                 Least Squares   F-statistic:                          1.497e+04
Date:                Mon, 14 Oct 2024   Prob (F-statistic):                        0.00
Time:                        21:32:15   Log-Likelihood:                     -2.9867e+05
No. Observations:               21613   AIC:                                  5.974e+05
Df Residuals:                   21604   BIC:                                  5.974e+05
Df Model:                           9
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x1         -4.155e+04   2221.630    -18.704      0.000   -4.59e+04   -3.72e+04
x2          6176.7471   3625.176      1.704      0.088    -928.865    1.33e+04
x3          1.597e+04   4069.140      3.924      0.000    7989.976    2.39e+04
x4          5.397e+05   2.09e+04     25.804      0.000    4.99e+05    5.81e+05
x5          7.122e+04   2497.093     28.519      0.000    6.63e+04    7.61e+04
```

In [64]:
```
# Aplicacion del modelo en Base de Prueba(Test)
Y_pred = np.matmul(X_test, betas)
Y_pred
```

Out[64]:
```
array([[694494.418933],
       [398752.138576],
       [654474.313832],
       ...,
       [335206.882288],
       [2158408.658573],
       [945335.347565]])
```

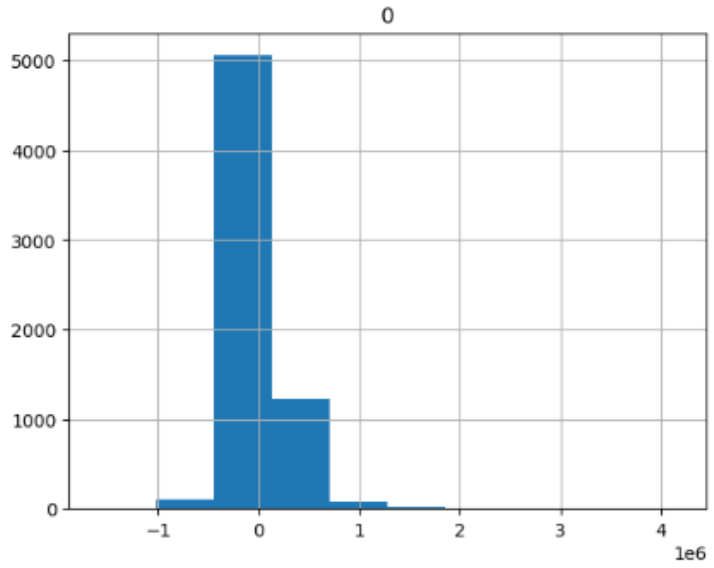
In [65]:
```
# calculo de residuales

Resid = Y_test - Y_pred
Resid
```

Out[65]:
```
array([[-235494.418933],
       [46247.861424],
       [402525.686168],
       ...,
       [-105206.882288],
       [-478408.658573],
       [-652335.347565]])
```

In [66]:
```
# Grafico del Histograma de Residuales para la Base de prueba
df = pd.DataFrame(Resid)
df.hist()
```

Out[66]: array([[<Axes: title={'center': '0'}>]], dtype=object)



In [67]:
```
# Calculamos R2 Score
from sklearn.metrics import r2_score
from sklearn import metrics
print('Coeficiente R Cuadrada', r2_score(Y_test, Y_pred))
```

```
Coeficiente R Cuadrada 0.5731599968271357
```

NOTA

** R^2 = 0.5731, significa que aproximadamente el 57.31% de la variación en el precio de las casas puede ser explicada por el modelo de regresión