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.).

In [3]:
```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
df = pd.read_csv('C:/Users/Isaac/Desktop/IHD/EBAC DT/CIENCIA DE DATOS/M52 DS/kc_house_data.csv')
df
```

Out[3]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_above	sqft_basement	yr_bu
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	0	0	...	7	1180	0	195
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	0	0	...	7	2170	400	195
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	0	0	...	6	770	0	193
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	0	0	...	7	1050	910	196
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	0	0		8	1680	0	198

- Construye un modelo de regresión lineal múltiple adecuado para pronosticar el precio de una casa, utiliza un enfoque matricial de pruebas de hipótesis.

In [4]:
```
df[['Intercepto']] = 1
```

In [5]:
```
'bedrooms', 'bathrooms', 'floors', 'waterfront', 'view', 'condition', 'sqft_above', 'sqft_basement', 'yr_built', 'yr_renovated']]
```

Out[5]:

	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 [7]:
```
X = df[['bedrooms', 'bathrooms', 'floors', 'waterfront', 'view', 'condition', 'sqft_above', 'sqft_basement', 'yr_built', 'yr_renc
Y = df[['price']].values
```

In [8]:
```
X = np.array(X)
X
```

Out[8]:
```
array([[3.000e+00, 1.000e+00, 1.000e+00, ..., 0.000e+00, 1.955e+03,
        0.000e+00],
       [3.000e+00, 2.250e+00, 2.000e+00, ..., 4.000e+02, 1.951e+03,
        1.991e+03],
       [2.000e+00, 1.000e+00, 1.000e+00, ..., 0.000e+00, 1.933e+03,
        0.000e+00],
       ...,
       [2.000e+00, 7.500e-01, 2.000e+00, ..., 0.000e+00, 2.009e+03,
        0.000e+00],
       [3.000e+00, 2.500e+00, 2.000e+00, ..., 0.000e+00, 2.004e+03,
        0.000e+00],
       [2.000e+00, 7.500e-01, 2.000e+00, ..., 0.000e+00, 2.008e+03,
        0.000e+00]])
```

In [9]:
```
Y = np.array(Y)
Y
```

Out[9]:
```
array([[221900.],
       [538000.],
       [180000.],
       ...,
       [402101.],
       [400000.],
       [325000.]])
```

In [10]:
```
XT_X = np.matmul(np.matrix.transpose(X), X)
XT_X
```

Out[10]:
```
array([[2.64274000e+05, 1.62054750e+05, 1.10770500e+05, 5.38000000e+02,
        1.82950000e+04, 2.48763000e+05, 1.38241089e+08, 2.39339120e+07,
        1.43686638e+08, 6.30116200e+06],
       [1.62054750e+05, 1.09476812e+05, 7.27991250e+04, 4.36500000e+02,
        1.31037500e+04, 1.54478500e+05, 9.11869682e+07, 1.54141938e+07,
        9.03346532e+07, 4.19694425e+06],
       [1.10770500e+05, 7.27991250e+04, 5.45627500e+04, 2.67500000e+02,
```

In [11]:
```
XT_X_inv = np.linalg.inv(XT_X)
XT_X_inv
```

Out[11]:
```
array([[ 8.44054957e-05, -2.24639559e-05,  9.24885453e-06,
         3.12857788e-05,  8.90399490e-06, -9.04575947e-06,
        -3.72549378e-08, -4.66684954e-08, -7.20228854e-08,
        -6.21861159e-10],
       [-2.24639559e-05,  2.24165100e-04, -9.38790006e-05,
         5.67398603e-06,  1.42978663e-06,  1.22958138e-05,
        -9.95286899e-08, -1.37594871e-07, -4.17672754e-08,
        -4.09013838e-09],
       [ 9.24885453e-06, -9.38790006e-05,  2.87015113e-04,
        -8.59439789e-06, -2.68060443e-06,  2.68241668e-05,
        -3.31869685e-08,  1.16656812e-07, -1.65938852e-07,
        -5.95367060e-10],
```

In [12]:
```
XT_Y = np.matmul(np.matrix.transpose(X), Y)
XT_Y
```

Out[12]:
```
array([[4.16230285e+10],
       [2.78943857e+10],
       [1.85431805e+10],
       [2.70885792e+08],
       [5.15064376e+09],
       [3.99857591e+10],
       [2.48545493e+13],
       [4.53985828e+12],
       [2.30199830e+13],
       [1.38817393e+12]])
```

In [13]:
```
betas = np.matmul(XT_X_inv, XT_Y)
betas
```

Out[13]:
```
array([[-4.59199898e+04],
       [ 1.21116791e+04],
       [ 2.89132544e+04],
       [ 5.34540806e+05],
```

In [14]:
```python
Y_pred = np.matmul(X, betas)
Y_pred
```

Out[14]:
```
array([[234956.17881302],
       [784132.98914506],
       [169002.07681303],
       ...,
       [257893.50943578],
       [394436.77845758],
       [257976.75272351]])
```

In [15]:
```python
resid = Y - Y_pred
resid
```

Out[15]:
```
array([[ -13056.17881302],
       [-246132.98914506],
       [  10997.92318697],
       ...,
       [ 144207.49056422],
       [   5563.22154242],
       [  67023.24727649]])
```

In [16]:
```python
# Calculo de la suma de residuales al cuadrado
RSS = float(np.matmul(np.matrix.transpose(resid), resid))
RSS
```

Out[16]:
```
1247099670851355.5
```

In [19]:
```python
# Calculo de la suma total de cuadrados
TSS = float(np.matmul(np.matrix.transpose(Y), Y) -len(Y) * (Y.mean() **2))
TSS
```

Out[19]:
```
2912916761921300.0
```

In [20]:
```python
# Calculo del coeficiente de Determinacion
R2= float(1 - RSS / TSS)
R2
```

Out[20]:
```
0.571872534377263
```

In [21]:
```python
# Calculo de coeficiente de determinacion ajustado
RSqAj = float(1 - (RSS / (X.shape[0] - X.shape[1])) / (TSS / (X.shape[0] -1)))
RSqAj
```

Out[21]:
```
0.571694172705708
```

In [23]:
```python
# Calculo de la varianza de error de regresión
vari_reg = RSS / (len(Y) - X.shape[1])
vari_reg
```

Out[23]:
```
57728078084.125145
```

In [25]:
```python
# Desviación estandar del error de regresión
import math

s = math.sqrt(vari_reg)
s
```

Out[25]:
```
240266.68117765547
```

In [26]:
```python
# Calculo de las t´s estadisticas para cada coeficiente de regresión
result_t = []
for i in range(0, X.shape[1]):
    t = float(betas[i] / (s * math.sqrt(XT_X_inv[i][i])))
    result_t.append(t)
result_t
```

Out[26]:
```
[-20.802851614755024,
 3.366873564868969,
 7.103150963004842,
 25.82955547781555,
 27.936944548702137,
```

Criterio 1

- En este enfoque, utilizamos las pruebas t para cada coeficiente y determinamos si cada variable independiente es significativa en el modelo.
- Se considera que los coeficientes son significativos si el valor p es menor a un nivel de significancia (por ejemplo, 0.05).

In [27]:
```python
import scipy.stats

grados_libertad = len(Y) - X.shape[1]

# la t_critica se obtendra a un nivel de confianza de 95% (Alfa = 5%)
t_critico = abs(scipy.stats.t.ppf(q = 0.025, df = grados_libertad))
t_critico
```

Out[27]:
```
1.9600738027083116
```

In [28]:
```python
for i in range(0, X.shape[1]):
    if(abs(result_t[i]) > t_critico):
        print('Beta', i, 'es significativa') # Aqui se rechaza H0
    else:
         print('Beta', i, 'No es significativa') # Aqui No se rechaza H0
```

```
Beta 0 es significativa
Beta 1 es significativa
Beta 2 es significativa
Beta 3 es significativa
Beta 4 es significativa
Beta 5 es significativa
Beta 6 es significativa
Beta 7 es significativa
Beta 8 es significativa
Beta 9 es significativa
```

Criterio 2

- Si el valor p es pequeño, indica que el modelo en su conjunto es significativo.

In [45]:
```
# Calculo de valores p
for i in range(0, X.shape[1]):
    print('Valor p de Beta', i, ':', scipy.stats.t.sf(abs(result_t[i]), df = grados_libertad) *2)
```

```
Valor p de Beta 0 : 3.496523034651408e-95
Valor p de Beta 1 : 0.0007615810625966622
Valor p de Beta 2 : 1.257295265983115e-12
Valor p de Beta 3 : 6.524944038434186e-145
Valor p de Beta 4 : 9.441144894464743e-169
Valor p de Beta 5 : 2.9332286885882334e-102
Valor p de Beta 6 : 0.0
Valor p de Beta 7 : 0.0
Valor p de Beta 8 : 1.42306115668164e-41
Valor p de Beta 9 : 6.146929714154527e-51
```

Criterio 3

In [46]:
```
# calculo de intervalos de confianza del 95% para el verdadero valor del coeficiente de cada beta
for i in range(0, X.shape[1]):
    print('Beta', i, 'se encuentra entre', float(betas[i]) - t_critico * s * math.sqrt(XT_X_inv[i][i]),
          'y', float(betas[i]) + t_critico * s * math.sqrt(XT_X_inv[i][i]))
```

```
Beta 0 se encuentra entre -50246.63556623386 y -41593.3441073031
Beta 1 se encuentra entre 5060.691169546324 y 19162.66708940327
Beta 2 se encuentra entre 20934.807514932705 y 36891.70121722156
Beta 3 se encuentra entre 493977.21867036505 y 575104.3934141918
Beta 4 se encuentra entre 64231.24805216617 y 73924.3313995302
Beta 5 se encuentra entre 50667.10495649588 y 60787.57999209191
Beta 6 se encuentra entre 271.2199649197552 y 283.4407383331193
Beta 7 se encuentra entre 258.3672756387775 y 277.4178318190918
Beta 8 se encuentra entre -95.2975861514185 y -71.18898931078934
Beta 9 se encuentra entre 53.76643053653545 y 69.8668334528707
```

In [50]:
```
import statsmodels.api as sm
regressor = sm.OLS(Y, X).fit()
print(regressor.summary())
```

```
                                 OLS Regression Results
=======================================================================================
Dep. Variable:                      y   R-squared (uncentered):                  0.865
Model:                            OLS   Adj. R-squared (uncentered):             0.865
Method:                 Least Squares   F-statistic:                         1.381e+04
Date:                Fri, 18 Oct 2024   Prob (F-statistic):                       0.00
Time:                        14:44:27   Log-Likelihood:                    -2.9844e+05
No. Observations:               21613   AIC:                                 5.969e+05
Df Residuals:                   21603   BIC:                                 5.970e+05
Df Model:                          10
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x1         -4.592e+04   2207.389    -20.803      0.000   -5.02e+04   -4.16e+04
x2          1.211e+04   3597.307      3.367      0.001    5060.691    1.92e+04
x3          2.891e+04   4070.483      7.103      0.000    2.09e+04    3.69e+04
x4          5.345e+05   2.07e+04     25.830      0.000    4.94e+05    5.75e+05
```

Proceso StepWise

In [51]:
```
# Eliminamos variables no significativas
X_new = np.delete(X, 1, axis = 1)

# Repetimos la regresión con las variables seleccionadas
XT_X_new = np.matmul(np.transpose(X_new), X_new)
XT_X_new_inv = np.linalg.inv(XT_X_new)
XT_Y_new = np.matmul(np.transpose(X_new), Y)
betas_new = np.matmul(XT_X_new_inv, XT_Y_new)

print(betas_new)
```

```
[[-4.47062584e+04]
 [ 3.39855530e+04]
 [ 5.34234240e+05]
 [ 6.90005381e+04]
 [ 5.50629975e+04]
 [ 2.82707904e+02]
 [ 2.75326828e+02]
 [-8.09865946e+01]
 [ 6.20376229e+01]]
```

In [54]:
```
# Calcular nuevamente el R^2 y el R^2 ajustado con las variables seleccionadas
Y_pred_new = np.matmul(X_new, betas_new)
resid_new = Y - Y_pred_new
RSS_new = float(np.matmul(np.transpose(resid_new), resid_new))
R_2_new = float(1 - RSS_new / TSS)
RSqAj_new = float(1 - (RSS_new / (X_new.shape[0] - X_new.shape[1])) / (TSS / (X_new.shape[0] - 1)))
R_2_new, RSqAj_new
```

Out[54]:
```
(0.571647881161359, 0.5714892616024482)
```

In [35]:
```
import statsmodels.api as sm
regressor = sm.OLS(Y, X_new).fit()
print(regressor.summary())
```

```
                            OLS Regression Results
=======================================================================================
Dep. Variable:                      y   R-squared (uncentered):                   0.864
Model:                            OLS   Adj. R-squared (uncentered):              0.864
Method:                 Least Squares   F-statistic:                          1.530e+04
Date:                Fri, 18 Oct 2024   Prob (F-statistic):                        0.00
Time:                        13:24:58   Log-Likelihood:                     -2.9846e+05
No. Observations:               21613   AIC:                                  5.969e+05
Df Residuals:                   21604   BIC:                                  5.970e+05
Df Model:                           9
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x1         -4.685e+04   2206.009    -21.238      0.000   -5.12e+04   -4.25e+04
x2          2.157e+04   3345.676      6.447      0.000     1.5e+04    2.81e+04
x3          5.354e+05   2.07e+04     25.842      0.000    4.95e+05    5.76e+05
x4          6.935e+04   2475.168     28.017      0.000    6.45e+04    7.42e+04
```