Regresión Lineal Multiple

In [ ]:
```
#Librerías
import statsmodels.api as sm
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
```

In [ ]:
```
#Carga de archivos
inmu = pd.read_csv("Clusters.csv")
inmu.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 657 entries, 0 to 656
Data columns (total 27 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Unnamed: 0          657 non-null    int64  
 1   Alcaldia            657 non-null    object 
 2   Colonia             657 non-null    object 
 3   X1                  657 non-null    float64
 4   X2                  657 non-null    float64
 5   X3                  657 non-null    float64
 6   X4                  657 non-null    float64
 7   X5                  657 non-null    float64
 8   X6                  657 non-null    float64
 9   X7                  657 non-null    float64
 10  X8                  657 non-null    float64
 11  X9                  657 non-null    float64
 12  X10                 657 non-null    float64
 13  Cocina_equip        657 non-null    int64  
 14  Gimnasio            657 non-null    int64  
 15  Amueblado           657 non-null    int64  
 16  Alberca             657 non-null    int64  
 17  Terraza             657 non-null    int64  
 18  Elevador            657 non-null    int64  
 19  m2_construido       657 non-null    float64
 20  Baños               657 non-null    float64
 21  Recamaras           657 non-null    int64  
 22  Lugares_estac       657 non-null    int64  
 23  Precio_m2           657 non-null    float64
 24  Cluster Labels      657 non-null    int64  
 25  Conglomerados       657 non-null    object 
 26  NivelSocioEconomico 657 non-null    object 
dtypes: float64(13), int64(10), object(4)
memory usage: 138.7+ KB
```

Modelo 1 (Todas las variables)

Este es el mejor de todos

In [ ]:
```
model = LinearRegression()
type(model)

x = inmu[["X1", "X2", "X3", "X4", "X5", "X6", "X7", "X8", "X9", "X10", "Cocina_equi
y = inmu["Precio_m2"]

model.fit(X = x, y = y)
model.__dict__
#Coeficiente de determinación
determinacion = model.score(x, y)
correlacion = np.sqrt(determinacion)
print("Determinacion:", determinacion)
print("Correlación: ", correlacion)

# Agrega una constante al conjunto de datos (intercepto)
x_with_intercept = sm.add_constant(x)

# Ajusta el modelo
model = sm.OLS(y, x_with_intercept).fit()

# Imprime un resumen del modelo que incluye valores p
print(model.summary())
```

```
Determinacion: 0.7480588775333463
Correlación:  0.8649039701223173
                            OLS Regression Results
==============================================================================
Dep. Variable:              Precio_m2   R-squared:                       0.748
Model:                            OLS   Adj. R-squared:                  0.741
Method:                 Least Squares   F-statistic:                     99.55
Date:                Mon, 27 Nov 2023   Prob (F-statistic):          3.17e-176
Time:                        19:57:38   Log-Likelihood:                -6075.7
No. Observations:                 657   AIC:                         1.219e+04
Df Residuals:                     637   BIC:                         1.228e+04
Df Model:                          19
Covariance Type:            nonrobust
=================================================================================
                    coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------
const         -1437.4161   2183.934     -0.658      0.511   -5725.997    2851.164
X1             2867.3888   2593.414      1.106      0.269   -2225.286    7960.064
X2            -2781.2602    525.398     -5.294      0.000   -3812.982   -1749.538
X3             1093.0076    139.017      7.862      0.000     820.021    1365.994
X4               52.3560    234.869      0.223      0.824    -408.855     513.567
X5             -477.5905     70.134     -6.810      0.000    -615.312    -339.869
X6            -1534.3042    832.801     -1.842      0.066   -3169.672     101.063
X7              -24.1865    101.479     -0.238      0.812    -223.460     175.087
X8              2.27e+04   5124.160      4.430      0.000    1.26e+04    3.28e+04
X9              151.5963    459.414      0.330      0.742    -750.552    1053.745
X10            -549.5713    145.992     -3.764      0.000    -836.255    -262.888
Cocina_equip      4.0252    468.688      0.009      0.993    -916.334     924.385
Gimnasio       -387.9065    355.493     -1.091      0.276   -1085.987     310.174
Amueblado       935.2883    704.469      1.328      0.185    -448.075    2318.651
Alberca        1371.2545    384.326      3.568      0.000     616.555    2125.954
Terraza         -30.3337    338.209     -0.090      0.929    -694.472     633.805
Elevador        233.7522    297.517      0.786      0.432    -350.481     817.985
Baños          1729.5269    210.258      8.226      0.000    1316.645    2142.409
Recamaras       573.7772    227.433      2.523      0.012     127.168    1020.387
Lugares_estac  1297.8213    200.525      6.472      0.000     904.051    1691.591
==============================================================================
Omnibus:                      514.300   Durbin-Watson:                   1.987
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            19519.111
Skew:                           3.102   Prob(JB):                         0.00
Kurtosis:                      28.972   Cond. No.                     3.63e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spe
cified.
[2] The condition number is large, 3.63e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Modelo 2 (Con significantes)

In [ ]:
```python
#Modelo 2 (Con significantes)
model = LinearRegression()
type(model)
```

```
x = inmu[["X2", "X3", "X5","X8", "X10", "Alberca", "Baños", "Recamaras", "Lugares_e
y = inmu["Precio_m2"]

model.fit(X = x, y = y)
model.__dict__
#Coeficiente de determinación
determinacion = model.score(x, y)
correlacion = np.sqrt(determinacion)
print("Determinacion:", determinacion)
print("Correlación: ", correlacion)

# Agrega una constante al conjunto de datos (intercepto)
x_with_intercept = sm.add_constant(x)

# Ajusta el modelo
model = sm.OLS(y, x_with_intercept).fit()

# Imprime un resumen del modelo que incluye valores p
print(model.summary())
```

```
Determinacion: 0.7367105913396992
Correlación:  0.858318467318337
                            OLS Regression Results
==============================================================================
Dep. Variable:              Precio_m2   R-squared:                       0.737
Model:                            OLS   Adj. R-squared:                  0.733
Method:                 Least Squares   F-statistic:                     201.2
Date:                Mon, 27 Nov 2023   Prob (F-statistic):          5.98e-181
Time:                        19:57:38   Log-Likelihood:                -6090.2
No. Observations:                 657   AIC:                         1.220e+04
Df Residuals:                     647   BIC:                         1.225e+04
Df Model:                           9
Covariance Type:            nonrobust
=================================================================================
                    coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------
const         -1522.5496   1732.780     -0.879      0.380   -4925.101    1880.002
X2            -1034.2976    268.884     -3.847      0.000   -1562.288    -506.307
X3              729.6193     79.917      9.130      0.000     572.692     886.547
X5             -465.8834     56.872     -8.192      0.000    -577.560    -354.207
X8             1.941e+04   3930.629      4.939      0.000    1.17e+04    2.71e+04
X10            -390.5308     40.148     -9.727      0.000    -469.368    -311.694
Alberca        1251.9295    321.454      3.895      0.000     620.711    1883.148
Baños          1662.8778    204.103      8.147      0.000    1262.093    2063.662
Recamaras       781.3415    224.378      3.482      0.001     340.745    1221.938
Lugares_estac  1346.4962    195.842      6.875      0.000     961.934    1731.058
==============================================================================
Omnibus:                      516.252   Durbin-Watson:                   2.028
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            21356.515
Skew:                           3.086   Prob(JB):                         0.00
Kurtosis:                      30.240   Cond. No.                     2.63e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spe
cified.
[2] The condition number is large, 2.63e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Modelo 3 (Con correlaciones más altas)

In [ ]:

```python
#Modelo 3 (Con correlaciones más altas)
# data_selected = inmu[["Precio_m2", "X1", "X2", "X3", "X4", "X5", "X6", "X7", "X8"

# # Calcula la matriz de correlación
# correlation_matrix = data_selected.corr()

# # Crea el heatmap con seaborn
# plt.figure(figsize=(10, 8))
# sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f", linewidth
# plt.title("Heatmap de Correlación entre Variables Seleccionadas")
# plt.show()

model = LinearRegression()
type(model)
```

```python
x = inmu[["Gimnasio", "Alberca", "Terraza", "Elevador", "Baños", "Recamaras", "Luga
y = inmu["Precio_m2"]

model.fit(X = x, y = y)
model.__dict__
#Coeficiente de determinación
determinacion = model.score(x, y)
correlacion = np.sqrt(determinacion)
print("Determinacion:", determinacion)
print("Correlación: ", correlacion)

# Agrega una constante al conjunto de datos (intercepto)
x_with_intercept = sm.add_constant(x)

# Ajusta el modelo
model = sm.OLS(y, x_with_intercept).fit()

# Imprime un resumen del modelo que incluye valores p
print(model.summary())
```

```
Determinacion: 0.6483537987961022
Correlación:  0.8052041969563387
                            OLS Regression Results
==============================================================================
Dep. Variable:              Precio_m2   R-squared:                       0.648
Model:                            OLS   Adj. R-squared:                  0.645
Method:                 Least Squares   F-statistic:                     170.9
Date:                Mon, 27 Nov 2023   Prob (F-statistic):          1.01e-142
Time:                        19:57:38   Log-Likelihood:                -6185.2
No. Observations:                 657   AIC:                         1.239e+04
Df Residuals:                     649   BIC:                         1.242e+04
Df Model:                           7
Covariance Type:            nonrobust
=================================================================================
                    coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------
const         -3983.4077    493.184     -8.077      0.000   -4951.836   -3014.979
Gimnasio       -788.7465    384.514     -2.051      0.041   -1543.788     -33.705
Alberca        1938.8306    412.111      4.705      0.000    1129.599    2748.062
Terraza        1226.8637    336.896      3.642      0.000     565.326    1888.402
Elevador        147.8290    298.111      0.496      0.620    -437.550     733.208
Baños          2375.8266    234.258     10.142      0.000    1915.831    2835.822
Recamaras       246.4787    252.463      0.976      0.329    -249.264     742.222
Lugares_estac  1935.9446    222.752      8.691      0.000    1498.543    2373.346
==============================================================================
Omnibus:                      454.628   Durbin-Watson:                   2.062
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            13589.184
Skew:                           2.642   Prob(JB):                         0.00
Kurtosis:                      24.645   Cond. No.                         16.9
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spe
cified.
```

Modelo 4 (Con significantes de la anterior)

In [ ]:

```
#Modelo 4 (Con significantes de la anterior)
model = LinearRegression()
type(model)

x = inmu[["Gimnasio", "Alberca", "Terraza", "Baños", "Lugares_estac"]]
y = inmu["Precio_m2"]

model.fit(X = x, y = y)
model.__dict__
#Coeficiente de determinación
determinacion = model.score(x, y)
correlacion = np.sqrt(determinacion)
print("Determinacion:", determinacion)
print("Correlación: ", correlacion)

# Agrega una constante al conjunto de datos (intercepto)
x_with_intercept = sm.add_constant(x)

# Ajusta el modelo
model = sm.OLS(y, x_with_intercept).fit()

# Imprime un resumen del modelo que incluye valores p
print(model.summary())
```

```
Determinacion: 0.6477140502689188
Correlación:  0.8048068403467498
                            OLS Regression Results
==============================================================================
Dep. Variable:              Precio_m2   R-squared:                       0.648
Model:                            OLS   Adj. R-squared:                  0.645
Method:                 Least Squares   F-statistic:                     239.4
Date:                Mon, 27 Nov 2023   Prob (F-statistic):          7.59e-145
Time:                        19:57:38   Log-Likelihood:                -6185.8
No. Observations:                 657   AIC:                         1.238e+04
Df Residuals:                     651   BIC:                         1.241e+04
Df Model:                           5
Covariance Type:            nonrobust
=================================================================================
                    coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------
const         -3565.6455    276.410    -12.900      0.000   -4108.408   -3022.883
Gimnasio       -773.0224    383.135     -2.018      0.044   -1525.352     -20.693
Alberca        1930.1871    408.522      4.725      0.000    1128.007    2732.367
Terraza        1233.2721    311.408      3.960      0.000     621.786    1844.758
Baños          2474.0428    215.768     11.466      0.000    2050.357    2897.729
Lugares_estac  1980.0743    217.340      9.111      0.000    1553.303    2406.845
==============================================================================
Omnibus:                      454.458   Durbin-Watson:                   2.065
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            13630.454
Skew:                           2.639   Prob(JB):                         0.00
Kurtosis:                      24.681   Cond. No.                         11.1
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spe
cified.
```