Analisis_CO2_sin_valores_de_influencia

```
#importar librerias
import pandas as pd
import numpy as np
```

Leer data

```
# pd.read excel BASE CON VALORES SIN INFLUENCIA.xlsx

df = pd.read_excel('BASE CON VALORES SIN INFLUENCIA.xlsx')
```

```
df.columns
```

```
Index(['Emision_CO2', 'Anio_emision', 'Dias_hasta_expiracion', 'coord_y',
       'Clasificacion_Emisiones_ordinal', 'Estado_edificio_ObraTerminada',
       'Estado_edificio_ProyectoNueva', 'Tipo_edificio_Bloquecompleto',
       'Tipo_edificio_Edificiocompleto', 'Tipo_edificio_Local',
       'Tipo_edificio_Unifamiliar', 'Muy_antiguo', 'AJUSTES', 'ID',
       'LogEmisiones', 'RES_1', 'DRE_1', 'ZRE_1', 'SRE_1', 'SDR_1', 'MAH_1',
       'COO_1', 'LEV_1', 'DFF_1', 'DFB0_1', 'DFB1_1', 'DFB2_1', 'DFB3_1',
       'DFB4_1', 'DFB5_1', 'DFB6_1', 'DFB7_1', 'DFB8_1', 'DFB9_1', 'DFB10_1',
       'DFB11_1', 'apalancamiento_alto', 'DFFITS_influyente', 'COVRATIO',
       'COVRATIO_influyente', 'algun_dfbetas_influyente',
       'influencia_conjunta', 'influencia_total_suma', 'RES_2', 'DRE_2',
       'ZRE_2', 'SRE_2', 'SDR_2', 'MAH_2', 'COO_2', 'LEV_2', 'DFF_2', 'DFB0_2',
       'DFB1_2', 'DFB2_2', 'DFB3_2', 'DFB4_2', 'DFB5_2', 'DFB6_2', 'DFB7_2',
       'DFB8_2', 'DFB9_2', 'DFB10_2', 'DFB11_2'],
      dtype='object')
```

```
# Filtrar solo las columnas necesarias después de la selección de variables

df_1 = df[['LogEmisiones','Clasificacion_Emisiones_ordinal', 'Tipo_edificio_Local', 'Tipo_edificio_Unifamiliar',
           'Tipo_edificio_Edificiocompleto', 'coord_y','Tipo_edificio_Bloquecompleto',
           'Estado_edificio_ProyectoNueva','Estado_edificio_ObraTerminada',
           'Dias_hasta_expiracion','Anio_emision','Muy_antiguo']]
```

```
df_1 #Nombre del dataframe a tratar
```

	LogEmisiones	Clasificacion_Emisiones_ordinal	Tipo_edificio_Local	Tipo_edificio_Unifamiliar	Tipo_edificio_Edificiocomple
0	3.404857	5	0	0	
1	3.829945	5	0	0	
2	3.010621	4	0	0	
3	3.668932	5	0	0	
4	4.641984	7	0	0	
...	...	...	...	...	
164906	4.038656	5	0	1	
164907	4.357350	6	0	0	
164908	2.927453	2	0	1	
164909	3.744078	5	0	0	
164910	4.772970	7	0	1	

164911 rows × 12 columns

```
# Cuando se compiló el modelo con las variables seleccionadas, notamos que
# algunas no estaban incluidas, esto era a causa de la dependencia lineal que se
# formó entre algunas columnas, por tanto, verificamos cuáles estaban ocasinando
# el problema:

# Tipo_edificio_Edificiocompleto
# Estado_edificio_ProyectoNueva
# Estado_edificio_Obra_terminada

# Recorre cada columna y muestra los valores únicos
columnas = [
```

```
    'LogEmisiones',
    'Clasificacion_Emisiones_ordinal',
    'Tipo_edificio_Local',
    'Tipo_edificio_Unifamiliar',
    'Tipo_edificio_Edificiocompleto',
    'coord_y',
    'Tipo_edificio_Bloquecompleto',
    'Estado_edificio_ProyectoNueva',
    'Estado_edificio_ObraTerminada',
    'Dias_hasta_expiracion',
    'Anio_emision',
    'Muy_antiguo'
]

for columna in columnas:
    print(f"Columna: {columna}")
    print(df_1[columna].unique())
    print("-"*50)
```

```
Columna: LogEmisiones
[3.40485734 3.82994489 3.01062089 ... 5.21172408 5.01230057 4.77297015]
--------------------------------------------------
Columna: Clasificacion_Emisiones_ordinal
[5 4 7 6 3 2 1]
--------------------------------------------------
Columna: Tipo_edificio_Local
[0 1]
--------------------------------------------------
Columna: Tipo_edificio_Unifamiliar
[0 1]
--------------------------------------------------
Columna: Tipo_edificio_Edificiocompleto
[0]
--------------------------------------------------
Columna: coord_y
[0.59547196 0.59559187 0.59291276 ... 0.34316301 0.35573486 0.69716777]
--------------------------------------------------
Columna: Tipo_edificio_Bloquecompleto
[0 1]
--------------------------------------------------
Columna: Estado_edificio_ProyectoNueva
[0]
--------------------------------------------------
Columna: Estado_edificio_ObraTerminada
[0]
--------------------------------------------------
Columna: Dias_hasta_expiracion
[3652 3653 3651 ... 4544 3003 4105]
--------------------------------------------------
Columna: Anio_emision
[2013 2014 2015 2011 2012 2016 2017 2018 2019 2020 2021 2022 2023]
--------------------------------------------------
Columna: Muy_antiguo
[0 1]
--------------------------------------------------
```

```
#Entonces seleccionamos nuevamente las variables que explican el modelo.
df_2 = df_1[['LogEmisiones','Clasificacion_Emisiones_ordinal', 'Tipo_edificio_Local', 'Tipo_edificio_Unifamiliar',
           'coord_y','Tipo_edificio_Bloquecompleto', 'Dias_hasta_expiracion','Anio_emision','Muy_antiguo']]
```

```
df_2
```

	LogEmisiones	Clasificacion_Emisiones_ordinal	Tipo_edificio_Local	Tipo_edificio_Unifamiliar	coord_y	Tipo_edificio_Bloqu
0	3.404857	5	0	0	0.595472	
1	3.829945	5	0	0	0.595592	
2	3.010621	4	0	0	0.592913	
3	3.668932	5	0	0	0.596922	
4	4.641984	7	0	0	0.595892	
...	...	...	...	...	...	
164906	4.038656	5	0	1	0.355735	
164907	4.357350	6	0	0	0.594765	
164908	2.927453	2	0	1	0.588639	
164909	3.744078	5	0	0	0.597117	
164910	4.772970	7	0	1	0.697168	

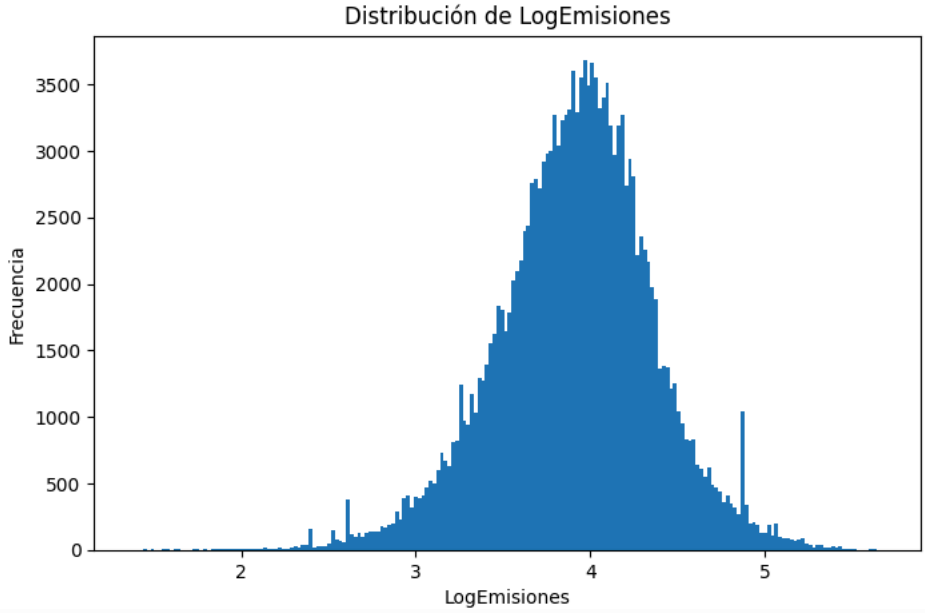
164911 rows × 9 columns

Análisis gráfico

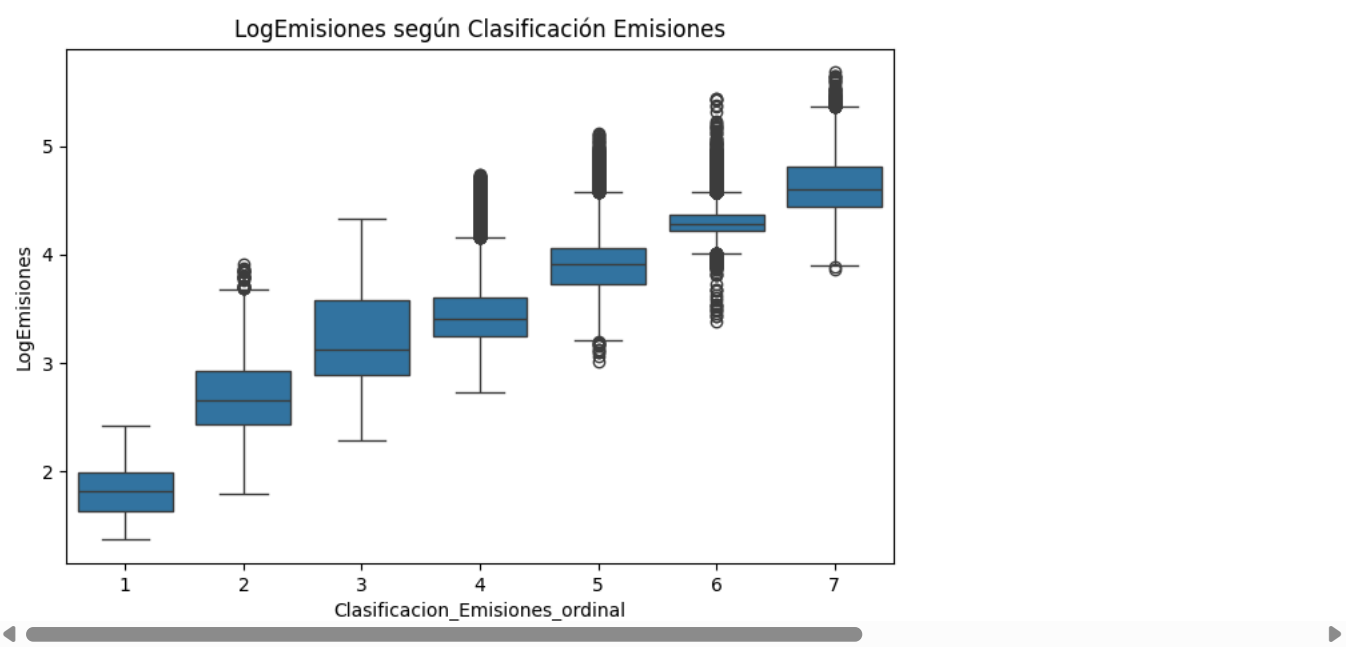
```
df_2.describe()
```

	LogEmisiones	Clasificacion_Emisiones_ordinal	Tipo_edificio_Local	Tipo_edificio_Unifamiliar	coord_y	Tipo_edificio_E
count	164911.000000	164911.000000	164911.000000	164911.000000	164911.000000	
mean	3.914228	5.063489	0.051252	0.136619	0.609406	
std	0.451535	0.964142	0.220512	0.343446	0.124285	
min	1.371181	1.000000	0.000000	0.000000	0.027387	
25%	3.646494	5.000000	0.000000	0.000000	0.591407	
50%	3.935935	5.000000	0.000000	0.000000	0.595856	
75%	4.197803	5.000000	0.000000	0.000000	0.604916	
max	5.683308	7.000000	1.000000	1.000000	0.996095	

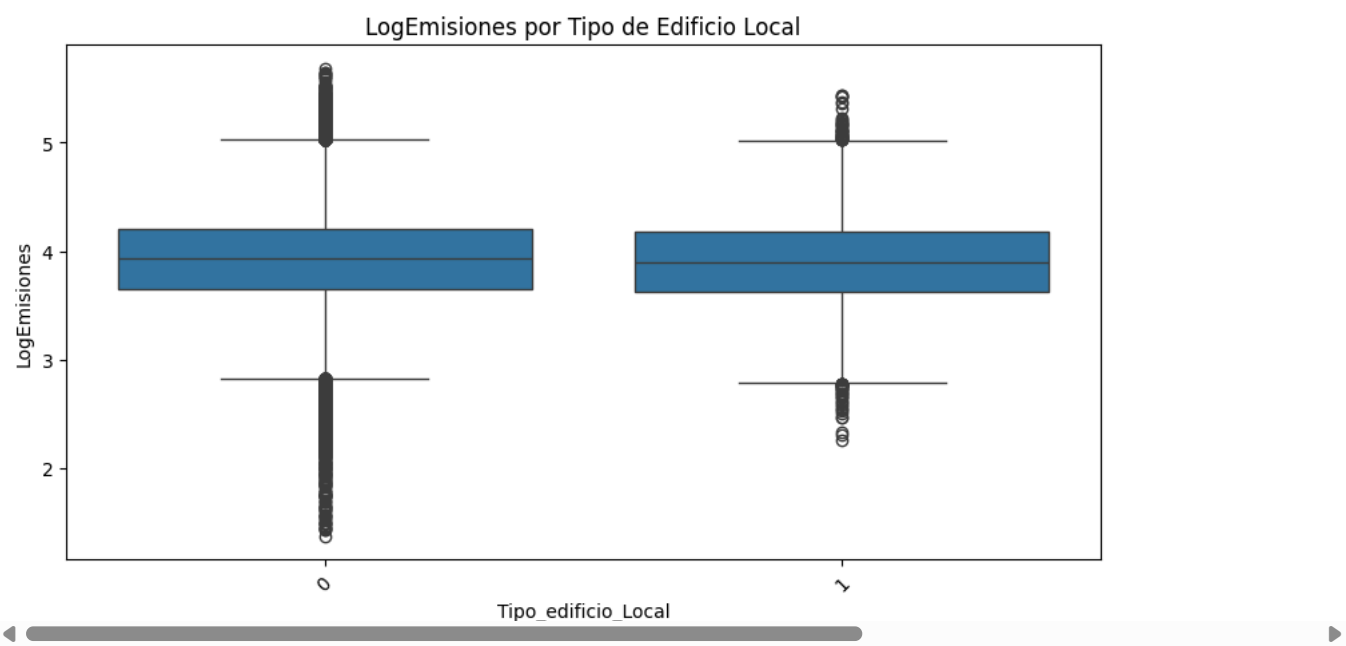
```
import matplotlib.pyplot as plt

plt.figure(figsize=(8,5))
plt.hist(df_2["LogEmisiones"], bins=200)
plt.xlabel("LogEmisiones")
plt.ylabel("Frecuencia")
plt.title("Distribución de LogEmisiones")
plt.show()
```



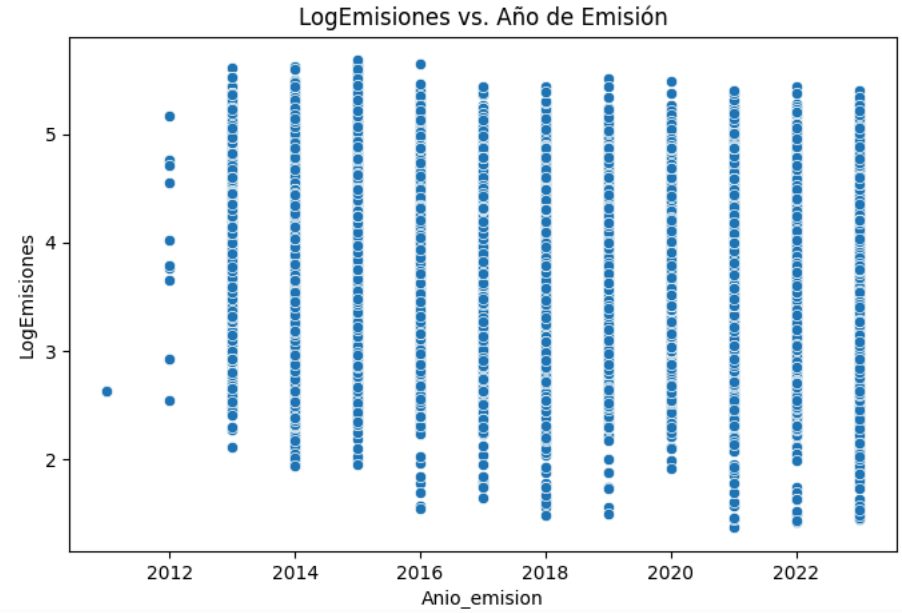
```
import seaborn as sns

plt.figure(figsize=(8,5))
sns.boxplot(x="Clasificacion_Emisiones_ordinal", y="LogEmisiones", data=df_2)
plt.title("LogEmisiones según Clasificación Emisiones")
plt.show()
```



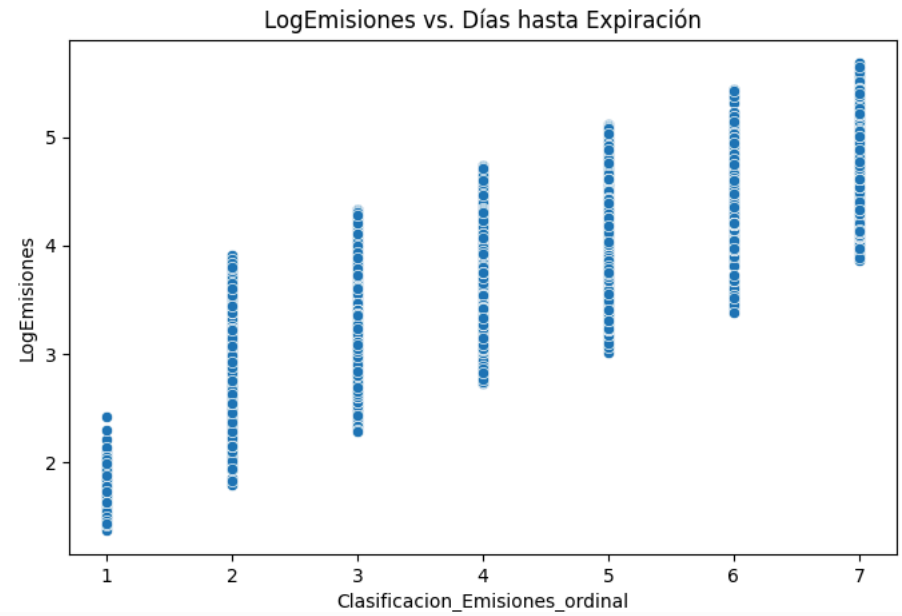

```
plt.figure(figsize=(10,5))
sns.boxplot(x="Tipo_edificio_Local", y="LogEmisiones", data=df_2)
plt.title("LogEmisiones por Tipo de Edificio Local")
plt.xticks(rotation=45)
plt.show()
```




```
plt.figure(figsize=(8,5))
sns.scatterplot(x="Anio_emision", y="LogEmisiones", data=df_2)
plt.title("LogEmisiones vs. Año de Emisión")
plt.show()
```




```
plt.figure(figsize=(8,5))
sns.scatterplot(x="Clasificacion_Emisiones_ordinal", y="LogEmisiones", data=df_2)
plt.title("LogEmisiones vs. Días hasta Expiración")
plt.show()
```



Visualización de la clasificación de emisiones ordinal en la emisión de CO2 (Dummies)

```
print(df_2["Clasificacion_Emisiones_ordinal"].unique())
```

```
[5 4 7 6 3 2 1]
```

```
import pandas as pd

dummies = pd.get_dummies(
    df_2["Clasificacion_Emisiones_ordinal"],
    prefix="Clasif",
    drop_first=True,
    dtype=int   # <-- Esto las hace int en lugar de bool
)

# Combinar dummies con df_2
```

```
df_3 = pd.concat([df_2, dummies], axis=1)
```

```
df_3.columns
```

```
Index(['LogEmisiones', 'Clasificacion_Emisiones_ordinal',
       'Tipo_edificio_Local', 'Tipo_edificio_Unifamiliar', 'coord_y',
       'Tipo_edificio_Bloquecompleto', 'Dias_hasta_expiracion', 'Anio_emision',
       'Muy_antiguo', 'Clasif_2', 'Clasif_3', 'Clasif_4', 'Clasif_5',
       'Clasif_6', 'Clasif_7'],
      dtype='object')
```

```
df_3
```

	LogEmisiones	Clasificacion_Emisiones_ordinal	Tipo_edificio_Local	Tipo_edificio_Unifamiliar	coord_y	Tipo_edificio_Bloqu
0	3.404857	5	0	0	0.595472	
1	3.829945	5	0	0	0.595592	
2	3.010621	4	0	0	0.592913	
3	3.668932	5	0	0	0.596922	
4	4.641984	7	0	0	0.595892	
...	...	...	...	...	...	
164906	4.038656	5	0	1	0.355735	
164907	4.357350	6	0	0	0.594765	
164908	2.927453	2	0	1	0.588639	
164909	3.744078	5	0	0	0.597117	
164910	4.772970	7	0	1	0.697168	

164911 rows × 16 columns

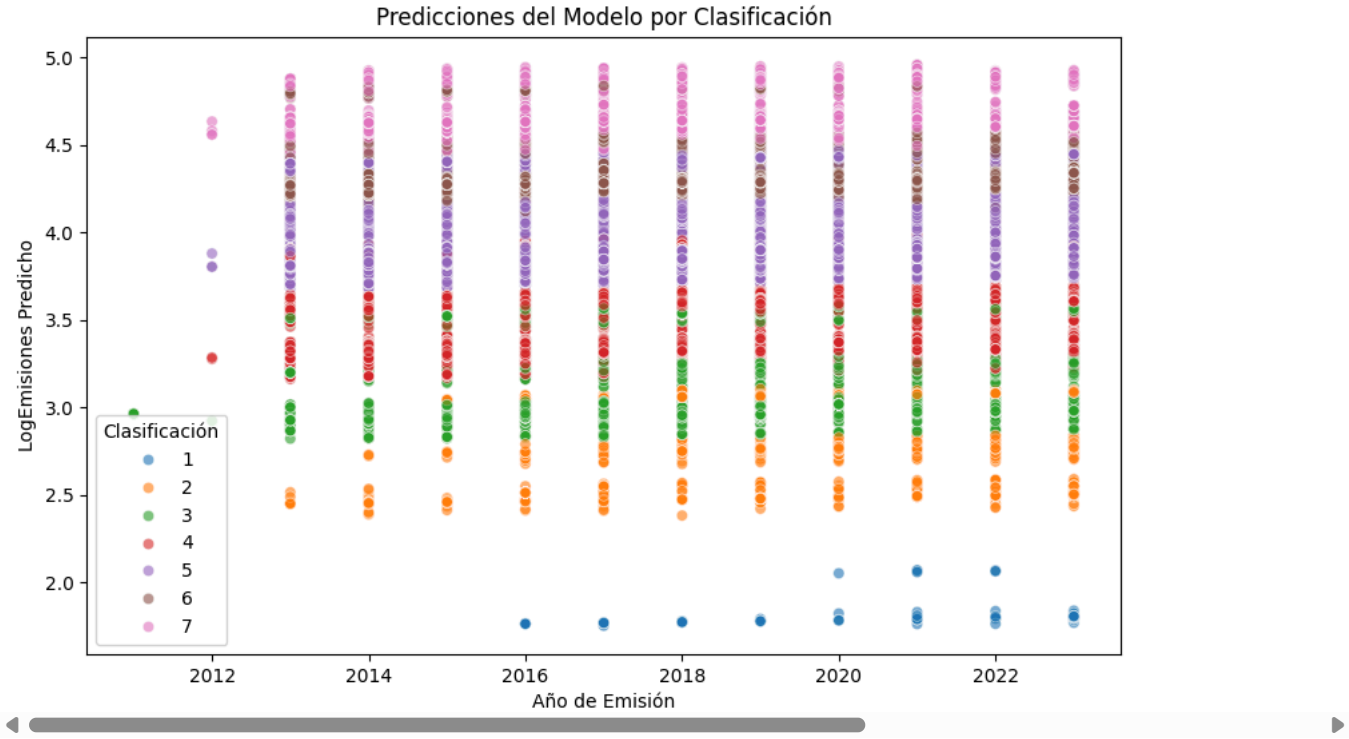
```
# regresión lineal múltiple
import statsmodels.api as sm

# Definir las variables independientes
X = df_3[
    [
        "Anio_emision",
        "Dias_hasta_expiracion",
        "coord_y",
        "Muy_antiguo",
        "Tipo_edificio_Local",
        "Tipo_edificio_Unifamiliar",
        "Tipo_edificio_Bloquecompleto",
        "Clasif_2",
        "Clasif_3",
        "Clasif_4",
        "Clasif_5",
        "Clasif_6",
        "Clasif_7"
    ]
]

# Agregar constante (intercepto)
X = sm.add_constant(X)

# Variable dependiente
y = df_3["LogEmisiones"]

# Ajustar el modelo
modelo = sm.OLS(y, X).fit()

# Mostrar resumen
print(modelo.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           LogEmisiones   R-squared:                       0.776
Model:                            OLS   Adj. R-squared:                  0.776
Method:                 Least Squares   F-statistic:                 4.406e+04
Date:                Sun, 29 Jun 2025   Prob (F-statistic):               0.00
```

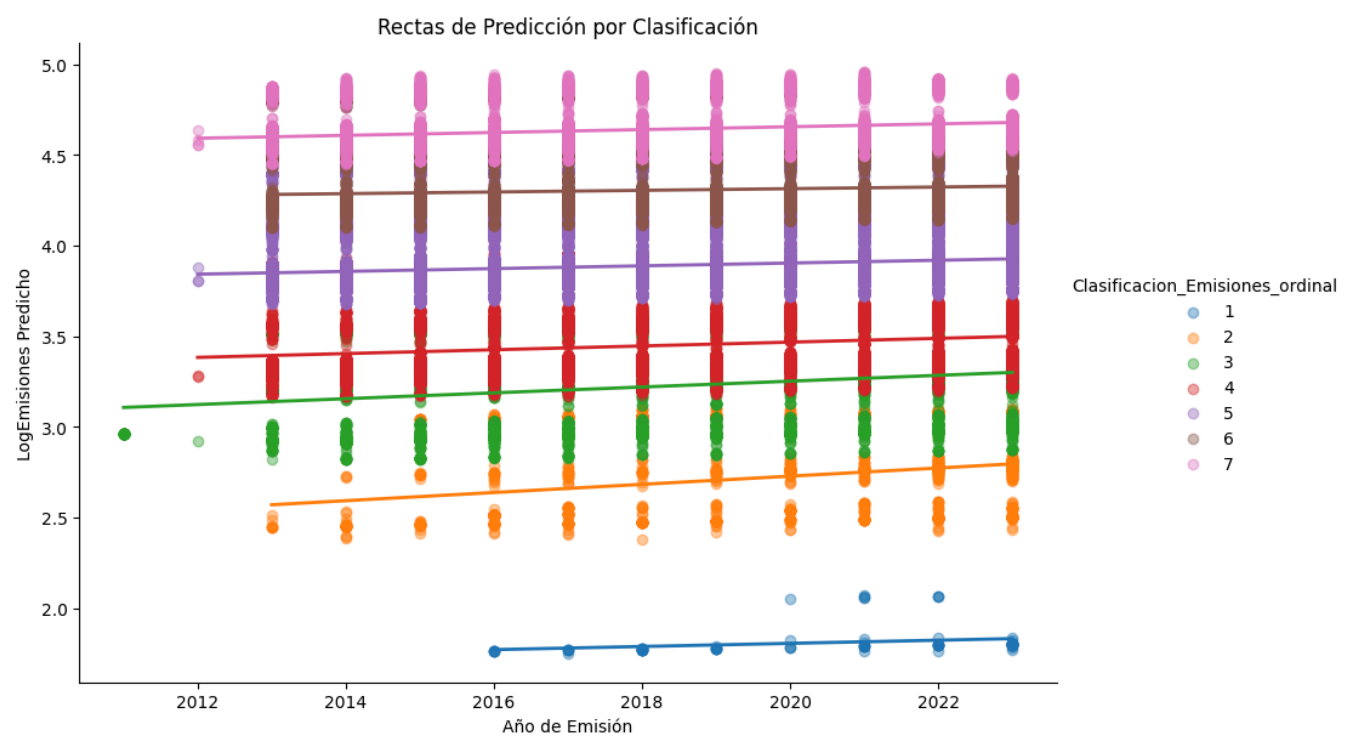
```
Time:                        18:27:38   Log-Likelihood:                 20662.
No. Observations:              164911   AIC:                         -4.130e+04
Df Residuals:                  164897   BIC:                         -4.116e+04
Df Model:                          13
Covariance Type:            nonrobust
==================================================================================================
                                 coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------------------
const                         -9.1896      0.359    -25.569      0.000      -9.894      -8.485
Anio_emision                   0.0054      0.000     30.267      0.000       0.005       0.006
Dias_hasta_expiracion        6.275e-06   1.41e-06      4.450      0.000    3.51e-06    9.04e-06
coord_y                        0.2373      0.004     55.953      0.000       0.229       0.246
Muy_antiguo                    0.0216      0.002     10.794      0.000       0.018       0.025
Tipo_edificio_Local            0.5842      0.003    223.409      0.000       0.579       0.589
Tipo_edificio_Unifamiliar      0.2716      0.002    165.751      0.000       0.268       0.275
Tipo_edificio_Bloquecompleto   0.0477      0.002     30.649      0.000       0.045       0.051
Clasif_2                       0.7024      0.022     31.512      0.000       0.659       0.746
Clasif_3                       1.1807      0.022     54.328      0.000       1.138       1.223
Clasif_4                       1.5343      0.022     70.956      0.000       1.492       1.577
Clasif_5                       2.0639      0.022     95.625      0.000       2.022       2.106
Clasif_6                       2.4640      0.022    113.882      0.000       2.422       2.506
Clasif_7                       2.8184      0.022    130.130      0.000       2.776       2.861
==============================================================================
Omnibus:                     1757.778   Durbin-Watson:                   1.746
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3048.548
Skew:                           0.011   Prob(JB):                         0.00
Kurtosis:                       3.666   Cond. No.                     2.83e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.83e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
```

```
df_3["Predicciones"] = modelo.fittedvalues
```

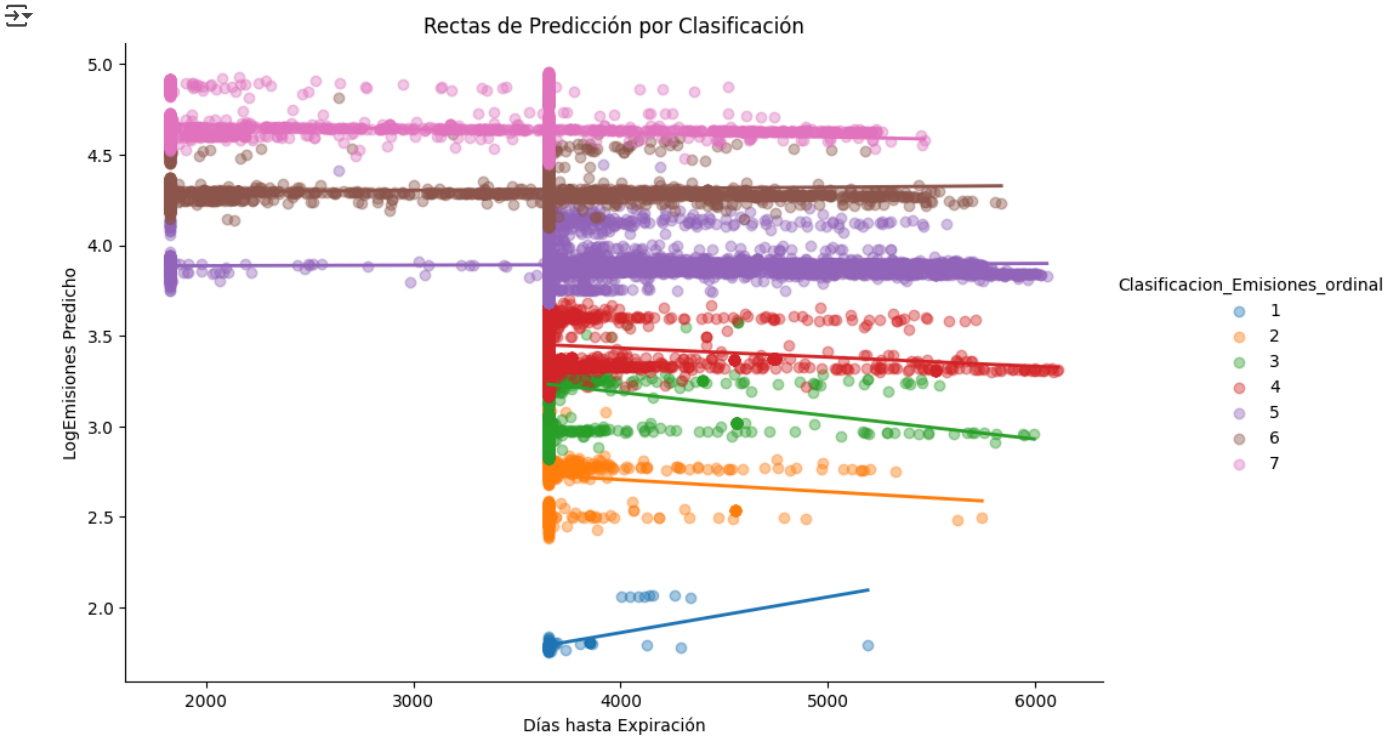
```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10,6))
sns.scatterplot(
    data=df_3,
    x="Anio_emision",
    y="Predicciones",
    hue="Clasificacion_Emisiones_ordinal",
    palette="tab10",
    alpha=0.6
)
plt.title("Predicciones del Modelo por Clasificación")
plt.xlabel("Año de Emisión")
plt.ylabel("LogEmisiones Predicho")
plt.legend(title="Clasificación")
plt.show()
```



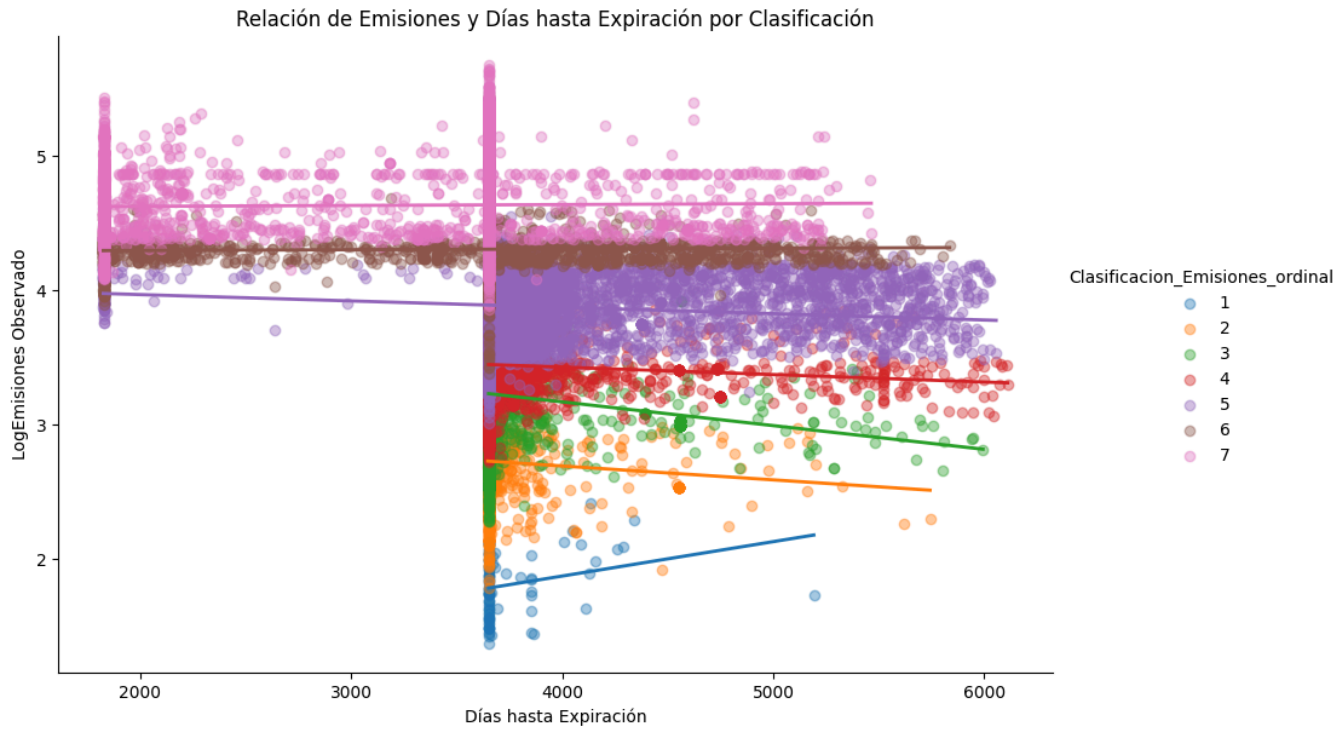

```
sns.lmplot(
    data=df_3,
    x="Anio_emision",
    y="Predicciones",
    hue="Clasificacion_Emisiones_ordinal",
    height=6,
    aspect=1.5,
    ci=None,
    scatter_kws={"alpha":0.4},
    line_kws={"linewidth":2}
)
plt.title("Rectas de Predicción por Clasificación")
plt.xlabel("Año de Emisión")
plt.ylabel("LogEmisiones Predicho")
plt.show()
```



```
sns.lmplot(
    data=df_3,
    x="Dias_hasta_expiracion",
    y="Predicciones",
    hue="Clasificacion_Emisiones_ordinal",
    ci=None,
    height=6,
    aspect=1.5,
    scatter_kws={"alpha":0.4},
    line_kws={"linewidth":2}
)
plt.title("Rectas de Predicción por Clasificación")
plt.xlabel("Días hasta Expiración")
plt.ylabel("LogEmisiones Predicho")
plt.show()
```




```
sns.lmplot(
    data=df_3,
    x="Dias_hasta_expiracion",
    y="LogEmisiones",
    hue="Clasificacion_Emisiones_ordinal",
    ci=None,
    height=6,
    aspect=1.5,
    scatter_kws={"alpha":0.4},
    line_kws={"linewidth":2}
)
plt.title("Relación de Emisiones y Días hasta Expiración por Clasificación")
plt.xlabel("Días hasta Expiración")
plt.ylabel("LogEmisiones Observado")
plt.show()
```



Scater plots de las dummies

```
import statsmodels.formula.api as smf

# Ajustar el modelo completo con TODAS las variables
modelo = smf.ols(
    "LogEmisiones ~ Clasificacion_Emisiones_ordinal + Anio_emision + Dias_hasta_expiracion + coord_y + Tipo_edificio_Local + Tipo_edifi
    data=df_2
).fit()

# Crear un dataframe con las predicciones
df_2["Predicciones"] = modelo.fittedvalues
```

```
C:\Users\JHOSSEP\AppData\Local\Temp\ipykernel_7912\2021751556.py:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  df_2["Predicciones"] = modelo.fittedvalues
```

```
df_2
```

	LogEmisiones	Clasificacion_Emisiones_ordinal	Tipo_edificio_Local	Tipo_edificio_Unifamiliar	coord_y	Tipo_edificio_Bloqu
0	3.404857	5	0	0	0.595472	
1	3.829945	5	0	0	0.595592	
2	3.010621	4	0	0	0.592913	
3	3.668932	5	0	0	0.596922	
4	4.641984	7	0	0	0.595892	
...	...	...	...	...	...	
164906	4.038656	5	0	1	0.355735	
164907	4.357350	6	0	0	0.594765	
164908	2.927453	2	0	1	0.588639	
164909	3.744078	5	0	0	0.597117	
164910	4.772970	7	0	1	0.697168	

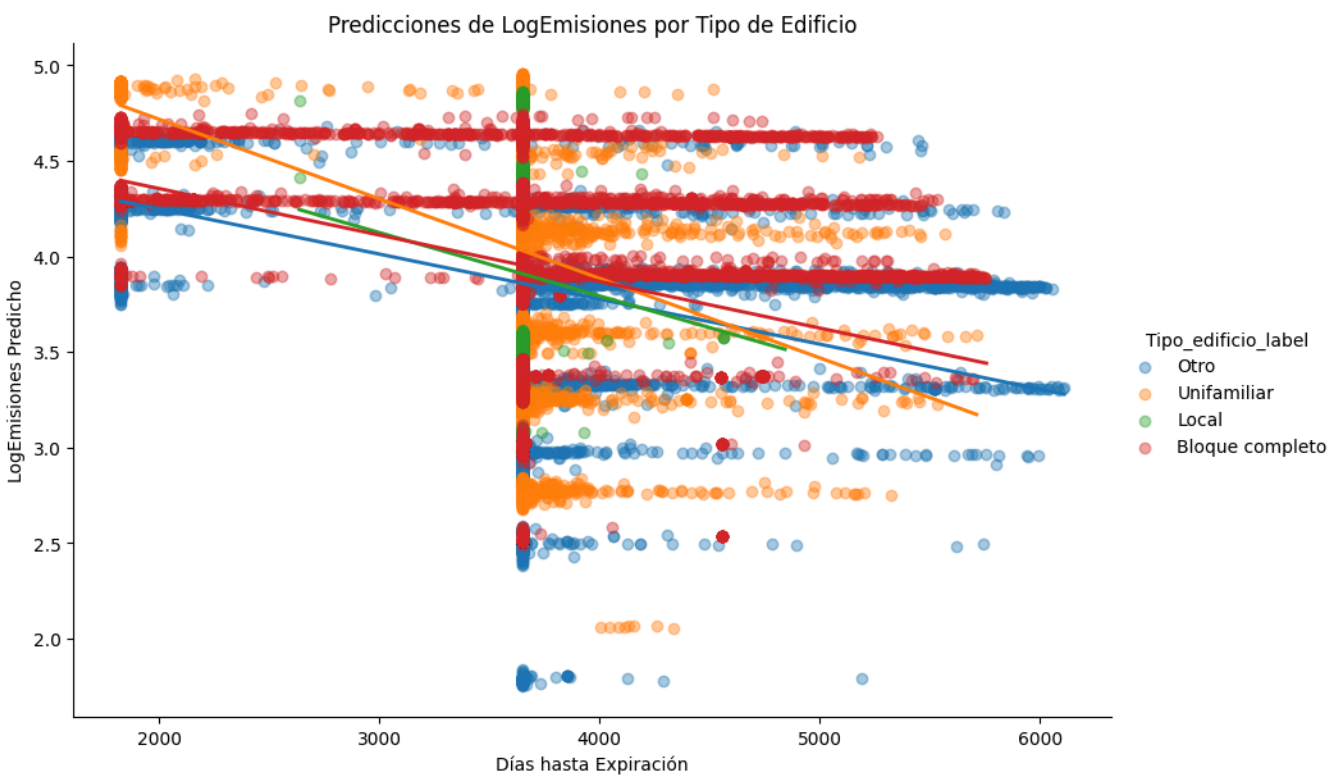
164911 rows × 9 columns

```
df_3
```

	LogEmisiones	Clasificacion_Emisiones_ordinal	Tipo_edificio_Local	Tipo_edificio_Unifamiliar	coord_y	Tipo_edificio_Bloqu
0	3.404857	5	0	0	0.595472	
1	3.829945	5	0	0	0.595592	
2	3.010621	4	0	0	0.592913	
3	3.668932	5	0	0	0.596922	
4	4.641984	7	0	0	0.595892	
...	...	...	...	...	...	
164906	4.038656	5	0	1	0.355735	
164907	4.357350	6	0	0	0.594765	
164908	2.927453	2	0	1	0.588639	
164909	3.744078	5	0	0	0.597117	
164910	4.772970	7	0	1	0.697168	

164911 rows × 16 columns

```
import seaborn as sns
import matplotlib.pyplot as plt

# Graficar las predicciones coloreadas por tipo de edificio, con Días en X
sns.lmplot(
    data=df_3,
    x="Dias_hasta_expiracion",
    y="Predicciones",
    hue="Tipo_edificio_label",
    ci=None,
    height=6,
    aspect=1.5,
    scatter_kws={"alpha":0.4},
    line_kws={"linewidth":2}
)

plt.title("Predicciones de LogEmisiones por Tipo de Edificio")
plt.xlabel("Días hasta Expiración")
plt.ylabel("LogEmisiones Predicho")
plt.show()
```




```
df_3.columns
```

```
Index(['LogEmisiones', 'Clasificacion_Emisiones_ordinal',
       'Tipo_edificio_Local', 'Tipo_edificio_Unifamiliar', 'coord_y',
       'Tipo_edificio_Bloquecompleto', 'Dias_hasta_expiracion', 'Anio_emision',
       'Muy_antiguo', 'Clasif_2', 'Clasif_3', 'Clasif_4', 'Clasif_5',
       'Clasif_6', 'Clasif_7', 'Predicciones', 'Tipo_edificio_label'],
      dtype='object')
```

Empieza a programar o a crear código con IA.

MODELO

```
import statsmodels.formula.api as smf
import matplotlib.pyplot as plt
import seaborn as sns

# Ajustar el modelo usando fórmula
formula = """
LogEmisiones ~ Anio_emision + Dias_hasta_expiracion + coord_y + Muy_antiguo +
               Tipo_edificio_Local + Tipo_edificio_Unifamiliar + Tipo_edificio_Bloquecompleto +
               Clasif_2 + Clasif_3 + Clasif_4 + Clasif_5 + Clasif_6 + Clasif_7
"""

modelo_f = smf.ols(formula=formula, data=df_3).fit()

# Mostrar resumen
print(modelo_f.summary())

# R-cuadrado
print("R-squared:", modelo_f.rsquared)

# Tabla ANOVA
from statsmodels.stats.anova import anova_lm
anova_table = anova_lm(modelo_f, typ=2)
print("\nANOVA Table:\n", anova_table)

# Gráfico residuos vs predicciones
plt.figure(figsize=(8,5))
sns.residplot(x=modelo_f.fittedvalues, y=modelo_f.resid, lowess=True, line_kws={"color":"red"})
plt.xlabel("Valores Predichos")
plt.ylabel("Residuos")
plt.title("Residuos vs Predicciones")
plt.axhline(0, color="gray", linestyle="--")
plt.show()
```

```
Df Residuals:                164897   BIC:                         -4.116e+04
Df Model:                       13
Covariance Type:            nonrobust
==============================================================================================
                                 coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------------------
Intercept                    -9.1896      0.359    -25.569      0.000      -9.894      -8.485
Anio_emision                  0.0054      0.000     30.267      0.000       0.005       0.006
Dias_hasta_expiracion       6.275e-06   1.41e-06      4.450      0.000    3.51e-06    9.04e-06
coord_y                       0.2373      0.004     55.953      0.000       0.229       0.246
Muy_antiguo                   0.0216      0.002     10.794      0.000       0.018       0.025
Tipo_edificio_Local           0.5842      0.003    223.409      0.000       0.579       0.589
Tipo_edificio_Unifamiliar     0.2716      0.002    165.751      0.000       0.268       0.275
Tipo_edificio_Bloquecompleto  0.0477      0.002     30.649      0.000       0.045       0.051
Clasif_2                      0.7024      0.022     31.512      0.000       0.659       0.746
Clasif_3                      1.1807      0.022     54.328      0.000       1.138       1.223
Clasif_4                      1.5343      0.022     70.956      0.000       1.492       1.577
Clasif_5                      2.0639      0.022     95.625      0.000       2.022       2.106
Clasif_6                      2.4640      0.022    113.882      0.000       2.422       2.506
Clasif_7                      2.8184      0.022    130.130      0.000       2.776       2.861
==============================================================================
Omnibus:                     1757.778   Durbin-Watson:                   1.746
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3048.548
Skew:                           0.011   Prob(JB):                         0.00
Kurtosis:                       3.666   Cond. No.                     2.83e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.83e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
R-squared: 0.7764793657590837

ANOVA Table:
                                 sum_sq        df             F  \
Anio_emision                  41.752705       1.0    916.115158
Dias_hasta_expiracion          0.902648       1.0     19.805409
coord_y                      142.687101       1.0   3130.762773
Muy_antiguo                    5.309648       1.0    116.501402
Tipo_edificio_Local         2274.765294       1.0  49911.663153
Tipo_edificio_Unifamiliar   1252.126865       1.0  27473.486807
Tipo_edificio_Bloquecompleto   42.812372       1.0    939.365790
Clasif_2                       45.258244       1.0    993.031770
Clasif_3                      134.516675       1.0   2951.491730
Clasif_4                      229.465684       1.0   5034.811272
Clasif_5                      416.752978       1.0   9144.167239
Clasif_6                      591.072737       1.0  12968.996581
Clasif_7                      771.778290       1.0  16933.939579
Residual                     7515.317042  164897.0           NaN

                                   PR(>F)
Anio_emision                1.097435e-200
Dias_hasta_expiracion        8.579576e-06
coord_y                      0.000000e+00
Muy_antiguo                  3.769213e-27
Tipo_edificio_Local          0.000000e+00
Tipo_edificio_Unifamiliar    0.000000e+00
Tipo_edificio_Bloquecompleto 1.033747e-205
Clasif_2                     2.611621e-217
Clasif_3                     0.000000e+00
Clasif_4                     0.000000e+00
Clasif_5                     0.000000e+00
Clasif_6                     0.000000e+00
Clasif_7                     0.000000e+00
Residual                              NaN
------------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[83], line 27
     25 # Gráfico residuos vs predicciones
     26 plt.figure(figsize=(8,5))
---> 27 sns.residplot(x=modelo_f.fittedvalues, y=modelo_f.resid, lowess=True, line_kws={"color":"red"})
     28 plt.xlabel("Valores Predichos")
     29 plt.ylabel("Residuos")

File c:\Users\JHOSSEP\anaconda3\envs\mi_entorno310\lib\site-packages\seaborn\regression.py:939, in residplot(data, x, y,
x_partial, y_partial, lowess, order, robust, dropna, label, color, scatter_kws, line_kws, ax)
    937 scatter_kws = {} if scatter_kws is None else scatter_kws.copy()
    938 line_kws = {} if line_kws is None else line_kws.copy()
--> 939 plotter.plot(ax, scatter_kws, line_kws)
    940 return ax

File c:\Users\JHOSSEP\anaconda3\envs\mi_entorno310\lib\site-packages\seaborn\regression.py:384, in _RegressionPlotter.plot(self,
```

```
ax, scatter_kws, line_kws)
    381     self.scatterplot(ax, scatter_kws)
    383 if self.fit_reg:
--> 384     self.lineplot(ax, line_kws)
    386 # Label the axes
    387 if hasattr(self.x, "name"):

File c:\Users\JHOSSEP\anaconda3\envs\mi_entorno310\lib\site-packages\seaborn\regression.py:429, in
_RegressionPlotter.lineplot(self, ax, kws)
    427 """Draw the model."""
    428 # Fit the regression model
--> 429 grid, yhat, err_bands = self.fit_regression(ax)
    430 edges = grid[0], grid[-1]
    432 # Get set default aesthetics

File c:\Users\JHOSSEP\anaconda3\envs\mi_entorno310\lib\site-packages\seaborn\regression.py:222, in
_RegressionPlotter.fit_regression(self, ax, x_range, grid)
    220 elif self.lowess:
    221     ci = None
--> 222     grid, yhat = self.fit_lowess()
    223 elif self.robust:
    224     from statsmodels.robust.robust_linear_model import RLM

File c:\Users\JHOSSEP\anaconda3\envs\mi_entorno310\lib\site-packages\seaborn\regression.py:309, in
_RegressionPlotter.fit_lowess(self)
    307 """Fit a locally-weighted regression, which returns its own grid."""
    308 from statsmodels.nonparametric.smoothers_lowess import lowess
--> 309 grid, yhat = lowess(self.y, self.x).T
    310 return grid, yhat

File c:\Users\JHOSSEP\anaconda3\envs\mi_entorno310\lib\site-packages\statsmodels\nonparametric\smoothers_lowess.py:226, in
lowess(endog, exog, frac, it, delta, xvals, is_sorted, missing, return_sorted)
    223 x = np.ascontiguousarray(x)
    224 if not given_xvals:
    225     # Run LOWESS on the data points
--> 226     res, _ = _lowess(y, x, x, np.ones_like(x),
    227                      frac=frac, it=it, delta=delta, given_xvals=False)
    228 else:
    229     # First run LOWESS on the data points to get the weights of the data points
    230     # using it-1 iterations, last iter done next
    231     if it > 0:

File statsmodels/nonparametric/_smoothers_lowess.pyx:201, in statsmodels.nonparametric._smoothers_lowess.lowess()

File statsmodels/nonparametric/_smoothers_lowess.pyx:346, in statsmodels.nonparametric._smoothers_lowess.calculate_weights()

File c:\Users\JHOSSEP\anaconda3\envs\mi_entorno310\lib\site-packages\numpy\_core\fromnumeric.py:2333, in _sum_dispatcher(a,
axis, dtype, out, keepdims, initial, where)
   2327         raise ValueError("Passing `min` or `max` keyword argument when "
   2328                          "`a_min` and `a_max` are provided is forbidden.")
   2330     return _wrapfunc(a, 'clip', a_min, a_max, out=out, **kwargs)
-> 2333 def _sum_dispatcher(a, axis=None, dtype=None, out=None, keepdims=None,
   2334                     initial=None, where=None):
   2335     return (a, out)
   2338 @array_function_dispatch(_sum_dispatcher)
   2339 def sum(a, axis=None, dtype=None, out=None, keepdims=np._NoValue,
   2340         initial=np._NoValue, where=np._NoValue):

KeyboardInterrupt:
```