co2-emissions-multiple

October 7, 2023

1 Act 1. Regresion lineal Multiple

CO2 Emission by Vehicles

- **Andrés Alejandro Guzmán González | A01633819**
- **Joel Isaias Solano Ocampo | A01639289**
- **Tania Sayuri Guizado Hernández | A01640092**
- **Ernesto Reynoso Lizárraga | A01639915**

Contenidos

- **5.1** - Temas avanzados sobre análisis regresión
- **5.1.1** - Verificación de supuestos: QQ-plots & análisis de residuales
- **5.2** - Estadística para datos multivariados

¿Qué pasa con el fit del modelo y a que se lo atribuye?

El fit del modelo mejora. En primer lugar el modelo usó las variables regresoras ('Engine Size
'Cylinders', 'Fuel Consumption City (L/100 km)', 'Fuel Consumption Hwy (L/100 km)', 'Fuel Consu
Comb (L/100 km)', 'Fuel Consumption Comb (mpg)') y como variable de respuesta ('CO2 Emissions(g
tuvo un ajuste de 0.904 y tras realizar el análisis de distribución de erroes y al ubicar que l
transformaciones asimétricas positivas aplicando raiz cuadrada y logarítmo base 10 acercaron la
distribución de la variable predictora a una normal se aumentó el ajuste a 0.915 y 0.925 respec

¿Qué sucede con el error y la distribución de este en los datos?

Siguinedo la respuesta de la pregunta anterior el error se reduce considerablemente conforme se
las transformaciones a los datos. Definitivamente el buscar una distribución lo más cercana a l
ayuda al modelo a que las predicciones sean más acertadas.

Describa el impacto de las distintas variables ¿Que sucede si se omiten las variables con nulo impacto?

Finalmente, al identificar las variables que menos impactaban en modelo qe acuerdo con el P-Val
y hacer el proceso de eliminación de variables no hubo ninguna mejora en el ajuste sin embargo
que al eliminar solo una de estas las demás los P-Valores de las demás quedan en 0, denotando q
modelo de regresión de manera importante.

##Llamado a librerias:

[327]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[328]:
```
import sklearn
import numpy as np
import pandas as pd
import seaborn as sns
import statsmodels.api as sm
import matplotlib.pyplot as plt
from scipy.stats import norm, uniform, skewnorm
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

##Importamos los datos a un DataFrame

[329]:
```
df = pd.read_csv('/content/drive/Shareddrives/Reto IA/Actividades/CO2_Emissions/
 ↪CO2 Emissions_Canada.csv')
df.head()
```

[329]:
```
    Make       Model Vehicle Class  Engine Size(L)  Cylinders Transmission  \
0  ACURA         ILX       COMPACT             2.0          4          AS5
1  ACURA         ILX       COMPACT             2.4          4           M6
2  ACURA  ILX HYBRID       COMPACT             1.5          4          AV7
3  ACURA     MDX 4WD   SUV - SMALL             3.5          6          AS6
4  ACURA     RDX AWD   SUV - SMALL             3.5          6          AS6

  Fuel Type  Fuel Consumption City (L/100 km)  \
0         Z                               9.9
1         Z                              11.2
2         Z                               6.0
3         Z                              12.7
4         Z                              12.1

   Fuel Consumption Hwy (L/100 km)  Fuel Consumption Comb (L/100 km)  \
0                              6.7                               8.5
1                              7.7                               9.6
2                              5.8                               5.9
3                              9.1                              11.1
4                              8.7                              10.6

   Fuel Consumption Comb (mpg)  CO2 Emissions(g/km)
0                           33                  196
1                           29                  221
2                           48                  136
```

```
3                            25                   255
4                            27                   244
```

[330]:
```
df.isnull().sum()
```

[330]:
```
Make                                0
Model                               0
Vehicle Class                       0
Engine Size(L)                      0
Cylinders                           0
Transmission                        0
Fuel Type                           0
Fuel Consumption City (L/100 km)    0
Fuel Consumption Hwy (L/100 km)     0
Fuel Consumption Comb (L/100 km)    0
Fuel Consumption Comb (mpg)         0
CO2 Emissions(g/km)                 0
dtype: int64
```

[331]:
```
df.shape
```

[331]:
```
(7385, 12)
```

[332]:
```
df.columns
```

[332]:
```
Index(['Make', 'Model', 'Vehicle Class', 'Engine Size(L)', 'Cylinders',
       'Transmission', 'Fuel Type', 'Fuel Consumption City (L/100 km)',
       'Fuel Consumption Hwy (L/100 km)', 'Fuel Consumption Comb (L/100 km)',
       'Fuel Consumption Comb (mpg)', 'CO2 Emissions(g/km)'],
      dtype='object')
```

[333]:
```
df.describe()
```

[333]:
```
       Engine Size(L)    Cylinders  Fuel Consumption City (L/100 km)  \
count     7385.000000  7385.000000                       7385.000000
mean         3.160068     5.615030                         12.556534
std          1.354170     1.828307                          3.500274
min          0.900000     3.000000                          4.200000
25%          2.000000     4.000000                         10.100000
50%          3.000000     6.000000                         12.100000
75%          3.700000     6.000000                         14.600000
max          8.400000    16.000000                         30.600000

       Fuel Consumption Hwy (L/100 km)  Fuel Consumption Comb (L/100 km)  \
count                      7385.000000                       7385.000000
mean                          9.041706                         10.975071
std                           2.224456                          2.892506
```

```
min                     4.000000                       4.100000
25%                     7.500000                       8.900000
50%                     8.700000                      10.600000
75%                    10.200000                      12.600000
max                    20.600000                      26.100000

       Fuel Consumption Comb (mpg)  CO2 Emissions(g/km)
count                  7385.000000          7385.000000
mean                     27.481652           250.584699
std                       7.231879            58.512679
min                      11.000000            96.000000
25%                      22.000000           208.000000
50%                      27.000000           246.000000
75%                      32.000000           288.000000
max                      69.000000           522.000000
```

##Funciones generales

Funcion para graficar en base a un modelo de distribucion QQPlot:

[334]:
```
def QQPlot(x, y, dst):
  X = sm.add_constant(x)
  model = sm.OLS(y, X)
  result = model.fit()
  influence = result.get_influence()
  standardized_residuals = influence.resid_studentized_internal
  fig = sm.qqplot(standardized_residuals, dist = dst, line = 'q')
  plt.title('QQ Graph - Normal Distribution')
  plt.ylabel('Standarized residuals Quantiles')
  plt.grid()
  plt.show()
```

Funcion para obtener los parametros y R^2 de cada iteracion de variable predictora con el modelo OLS:

[335]:
```
def OLS(x, y):
  x = sm.add_constant(x)
  model = sm.OLS(y, x)
  result = model.fit()
  print('Params:', result.params)
  print('R^2:', result.rsquared)
```

Función para graficar las gráficas de asimetría

[336]:
```
def Asimetric_Dist(r, title):
  plt.hist(r, density = True, bins = 'auto', histtype='stepfilled', alpha=0.2)
  plt.title(title)
  plt.grid()
  plt.show()
```

##Modelo de regresión

[337]:
```
x = np.array(df[['Engine Size(L)', 'Cylinders', 'Fuel Consumption City (L/100␣
↪km)', 'Fuel Consumption Hwy (L/100 km)', 'Fuel Consumption Comb (L/100 km)',␣
↪'Fuel Consumption Comb (mpg)']])
y = np.array(df['CO2 Emissions(g/km)'])
```

Obtenemos el valor de R^2 y los resultados de nuestro modelo de regresion OLS:

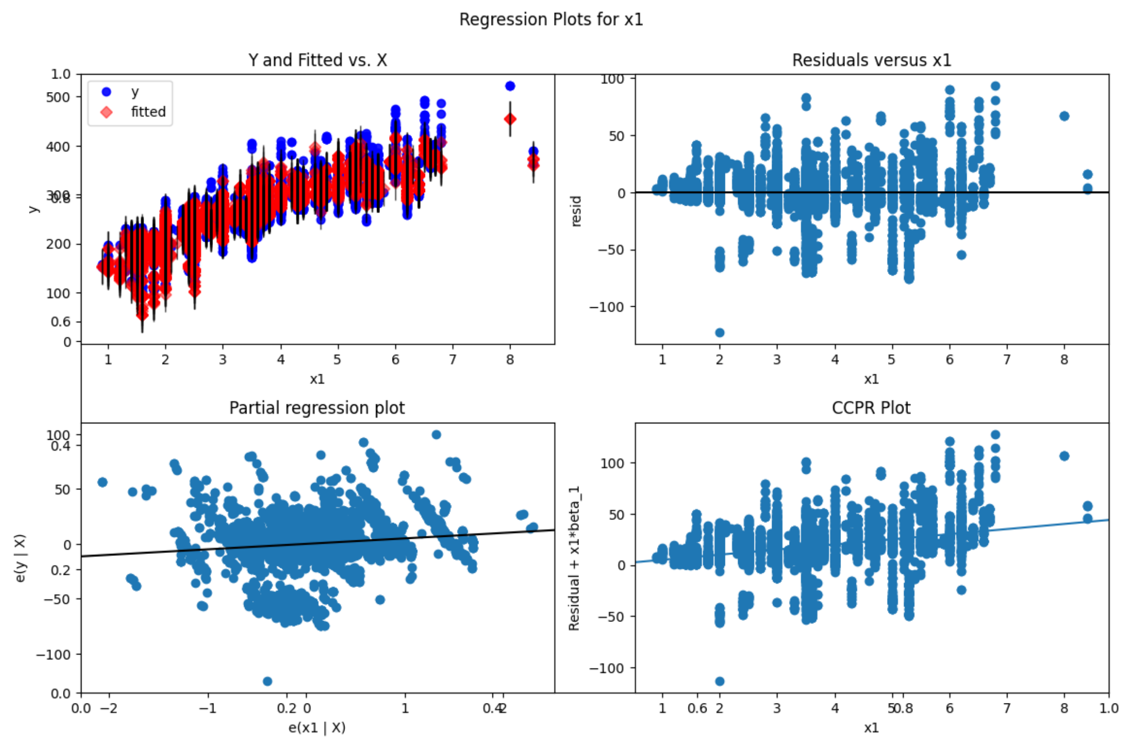
[338]:
```
X = sm.add_constant(x)
model = sm.OLS(y, X)
results = model.fit()

print('\n', results.params)
print('\n', 'R2: ', results.rsquared)
print(results.summary())
```

```
 [ 2.27892751e+02  4.99360380e+00  7.53852995e+00 -2.37835507e-02
  4.49061364e+00  1.67304643e+00 -3.42349241e+00]

 R2:  0.9039065926000305
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.904
Model:                            OLS   Adj. R-squared:                  0.904
Method:                 Least Squares   F-statistic:                 1.157e+04
Date:                Sat, 07 Oct 2023   Prob (F-statistic):               0.00
Time:                        04:12:07   Log-Likelihood:                -31880.
No. Observations:                7385   AIC:                         6.377e+04
Df Residuals:                    7378   BIC:                         6.382e+04
Df Model:                           6
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        227.8928      4.200     54.255      0.000     219.659     236.127
x1             4.9936      0.456     10.962      0.000       4.101       5.887
x2             7.5385      0.319     23.657      0.000       6.914       8.163
x3            -0.0238      2.738     -0.009      0.993      -5.391       5.344
x4             4.4906      2.260      1.987      0.047       0.061       8.920
x5             1.6730      4.969      0.337      0.736      -8.069      11.415
x6            -3.4235      0.079    -43.545      0.000      -3.578      -3.269
==============================================================================
Omnibus:                     1193.702   Durbin-Watson:                   1.618
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             7810.498
Skew:                          -0.609   Prob(JB):                         0.00
Kurtosis:                       7.889   Cond. No.                         987.
```

```
==============================================================================

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

Graficamos los valores predichos vs residuos estandarizados:

[339]:
```
fig, ax = plt.subplots(figsize=(12, 8))
sm.graphics.plot_regress_exog(results, 1, fig=fig)
plt.show()
```



Obtenemos los residuales estandarizados de la influencia de los resultados:

[340]:
```
influence = results.get_influence()
standardized_residuals = influence.resid_studentized_internal
print(standardized_residuals)
```

```
[-0.17261704 -0.00677114 -0.05425761 …  0.48654435  0.53345117
  0.6779954 ]
```

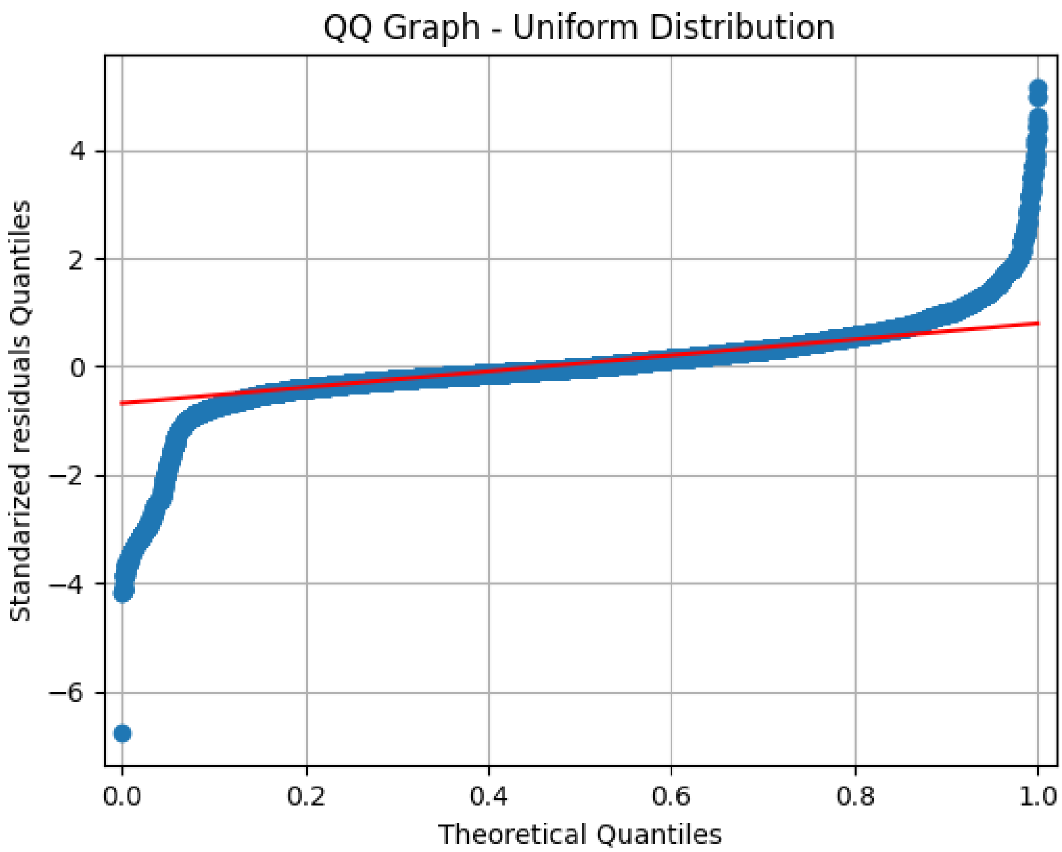
Realizamos una funcion que nos permite graficar una grafica QQPlot con los datos residuales estandarizados:

[341]:
```
def QQPlot_std_residual(std_res, d, l, title):
  fig = sm.qqplot(std_res, dist = d, line = l)
```

```
plt.title(title)
plt.ylabel('Standarized residuals Quantiles')
plt.grid()
plt.show()
```

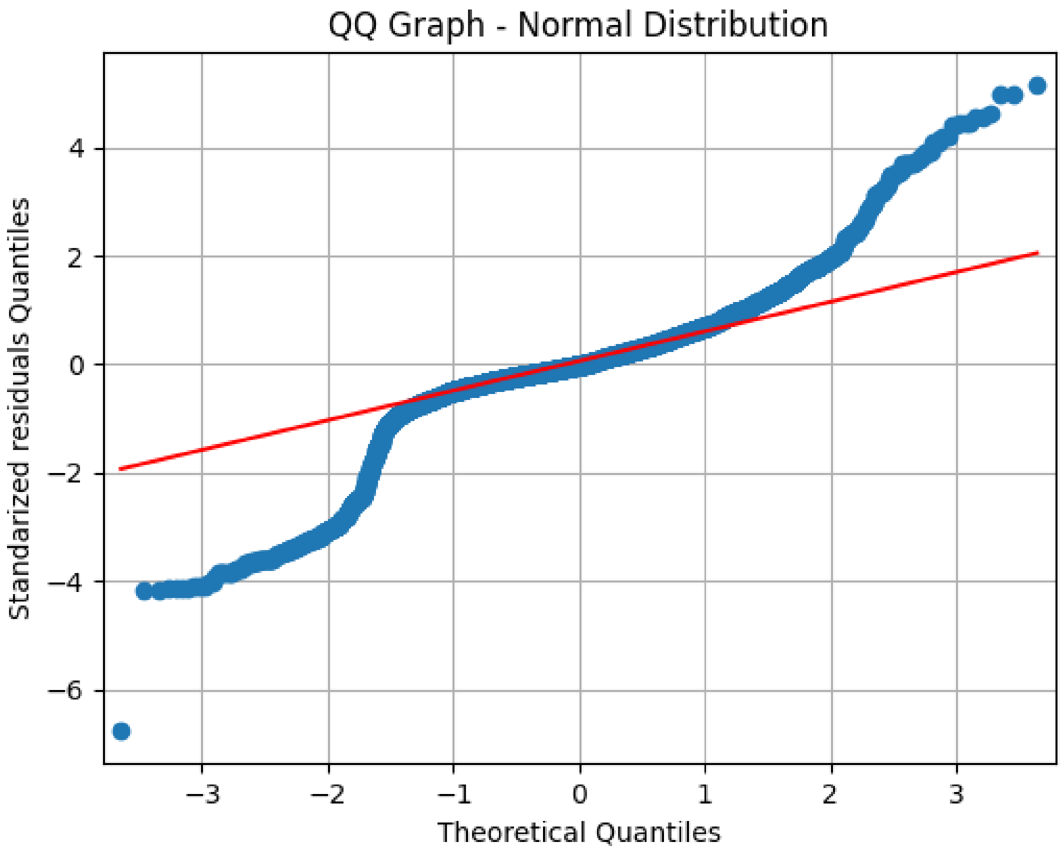
- Distribucion uniforme:

[342]:
```
QQPlot_std_residual(standardized_residuals, uniform, 'q', 'QQ Graph - Uniform␣
↪Distribution')
```



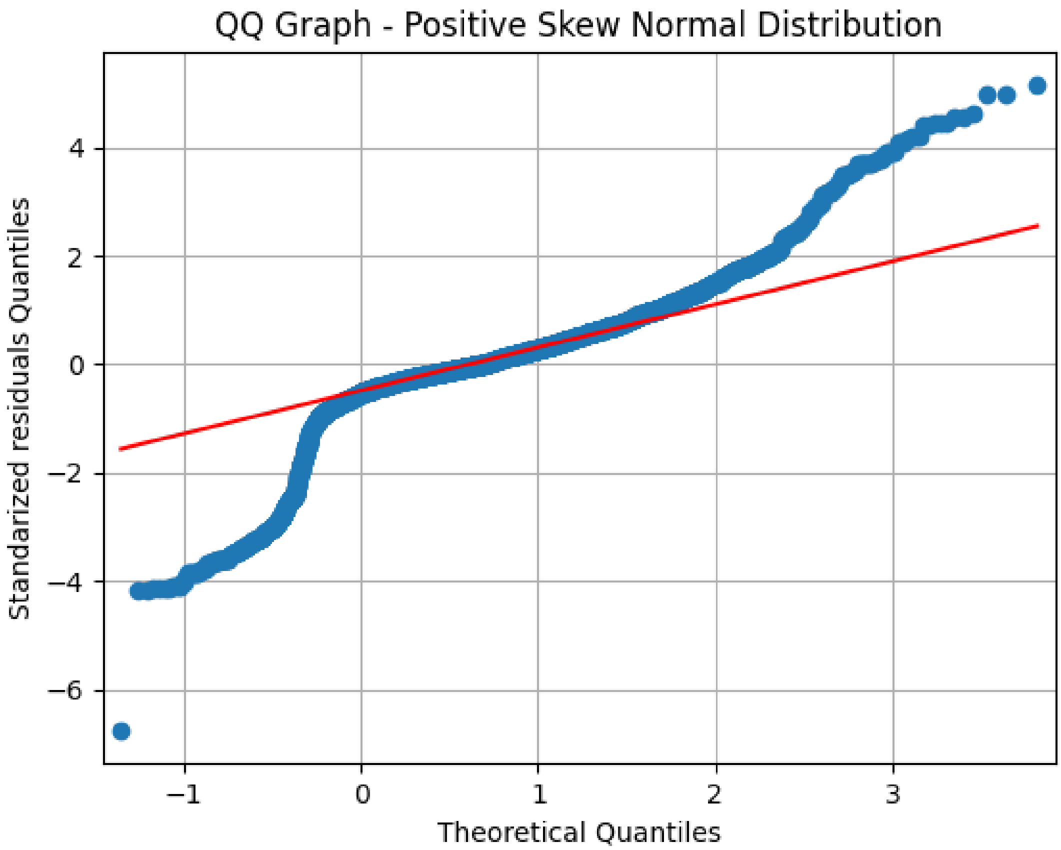
- Distribucion normal:

[343]:
```
QQPlot_std_residual(standardized_residuals, norm, 'q', 'QQ Graph - Normal␣
↪Distribution')
```



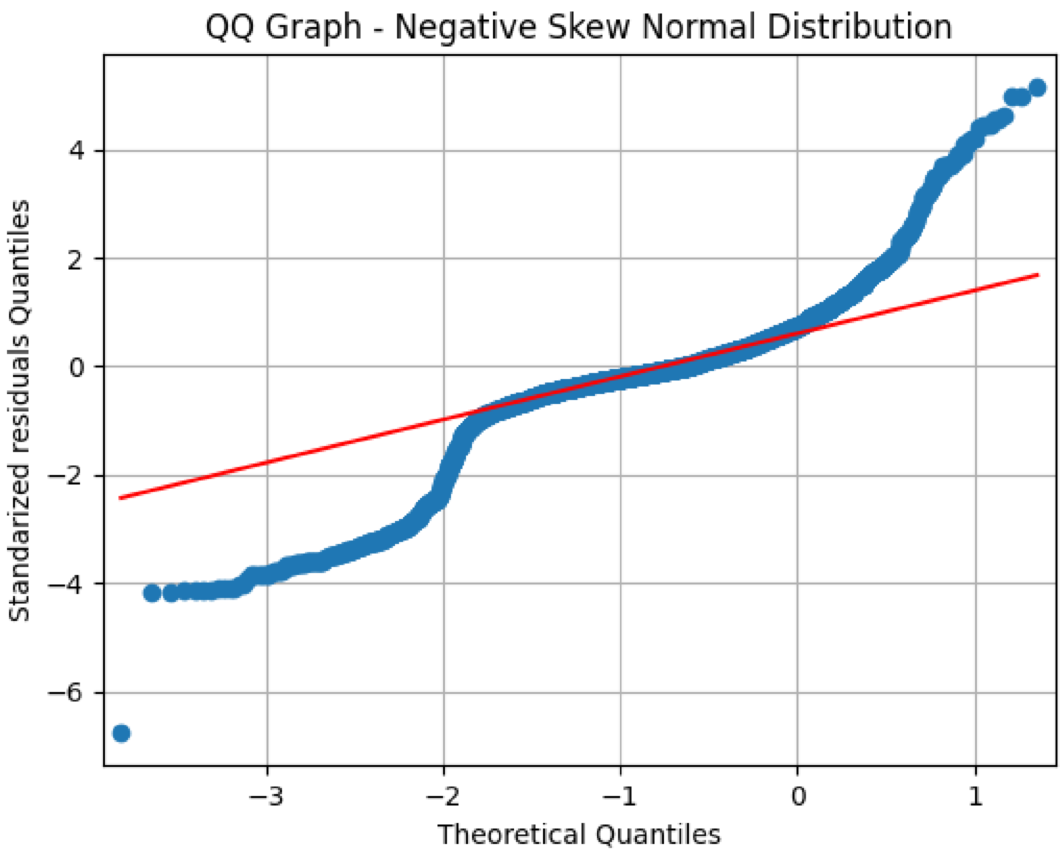
- Distribucion normal de inclinacion positiva:

[344]:
```
QQPlot_std_residual(standardized_residuals, skewnorm(2), 'q', 'QQ Graph -␣
↪Positive Skew Normal Distribution')
```



- Distribucion normal de distribucion negativa:

[345]:
```
QQPlot_std_residual(standardized_residuals, skewnorm(-2), 'q', 'QQ Graph -␣
↪Negative Skew Normal Distribution')
```

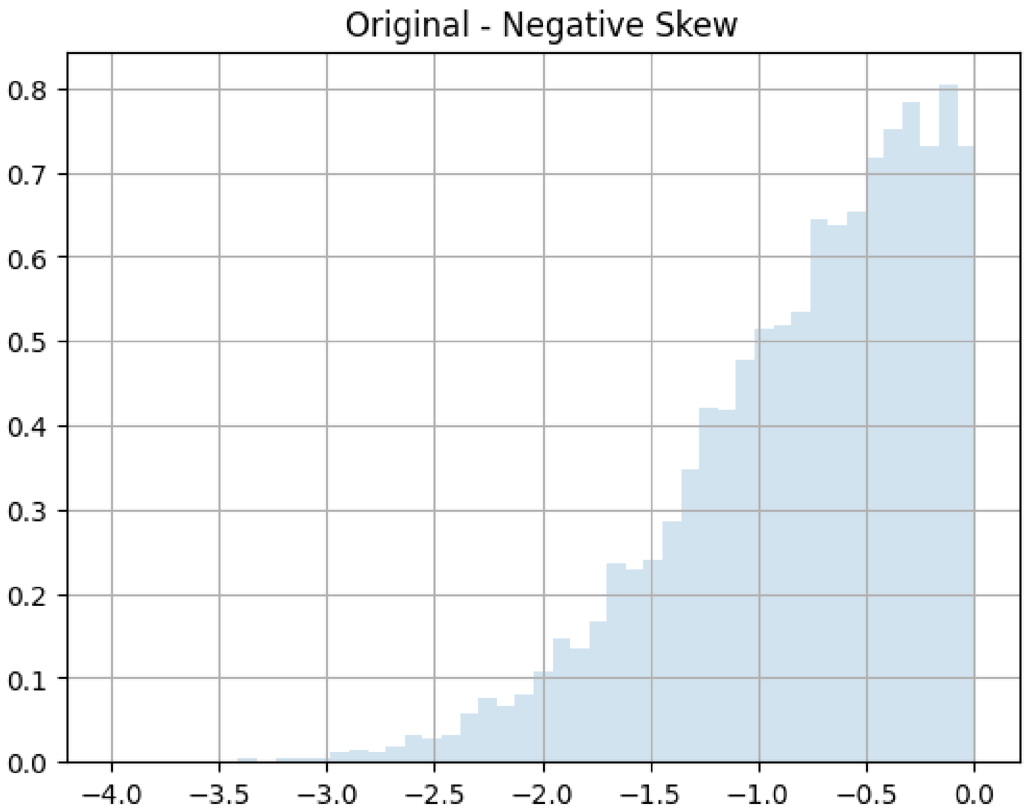


##Búsqueda de la transformacion adecuada

1.1 ### Distribucion Asimetrica Negativa

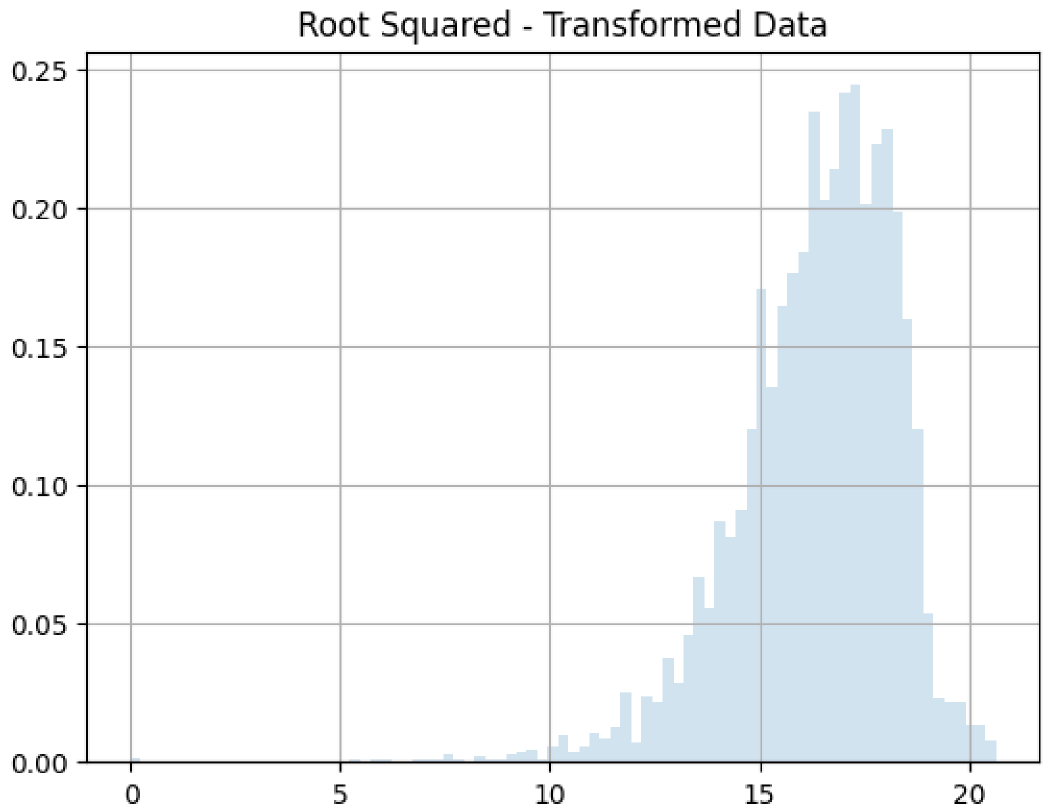
Distribucion Asimetrica Negative (1/3):

[346]:
```
y_skew_neg = -skewnorm.rvs(y)
Asimetric_Dist(y_skew_neg, 'Original - Negative Skew')
```



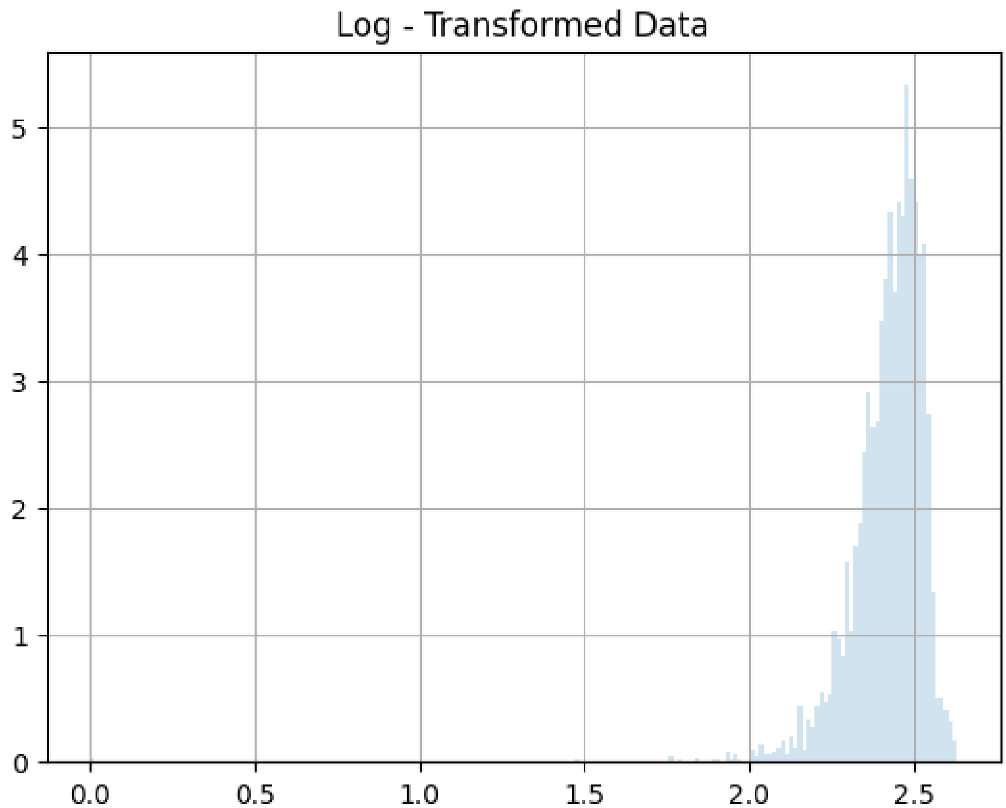
Distribucion Asimetrica Negative (2/3):

[347]:
```
y_pos = y + abs(min(y))
y_root_neg = np.sqrt(max(y_pos) - y_pos)
Asimetric_Dist(y_root_neg, 'Root Squared - Transformed Data')
```



Distribucion Asimetrica Negative (3/3):

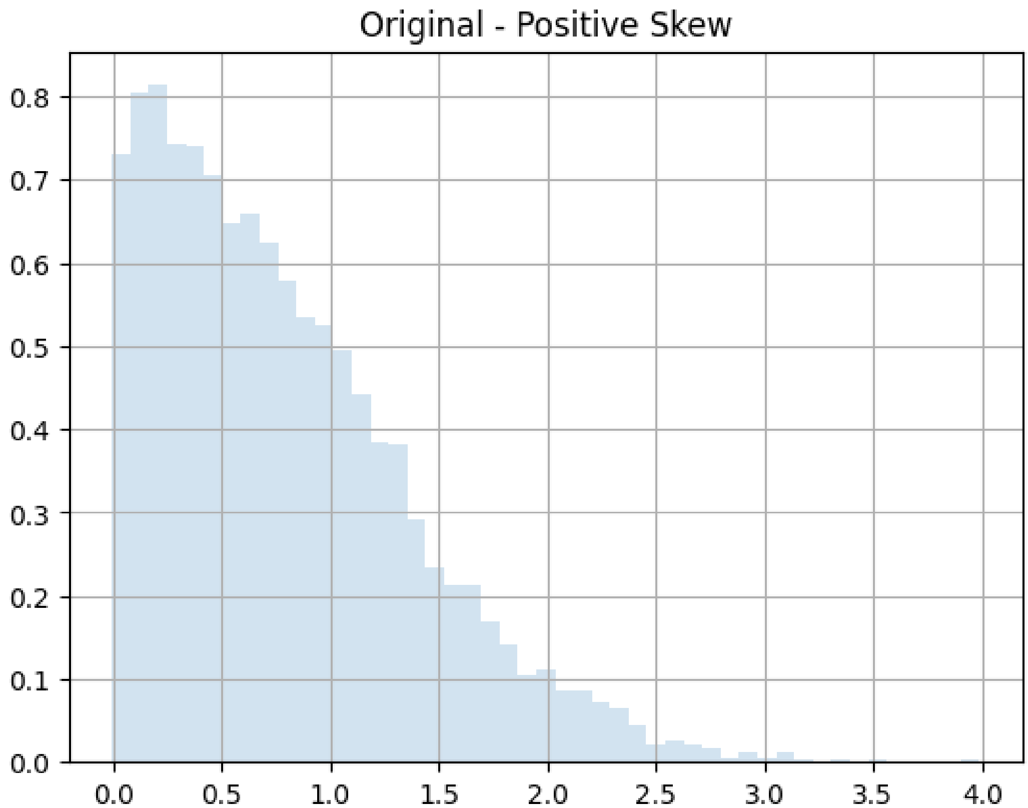
[348]:
```
y_positive = y + abs(min(y))
y_log_neg = np.log10(1 + max(y_positive) - y_positive)
Asimetric_Dist(y_log_neg, 'Log - Transformed Data')
```



1.2 ### Distribucion Asimetrica Positiva

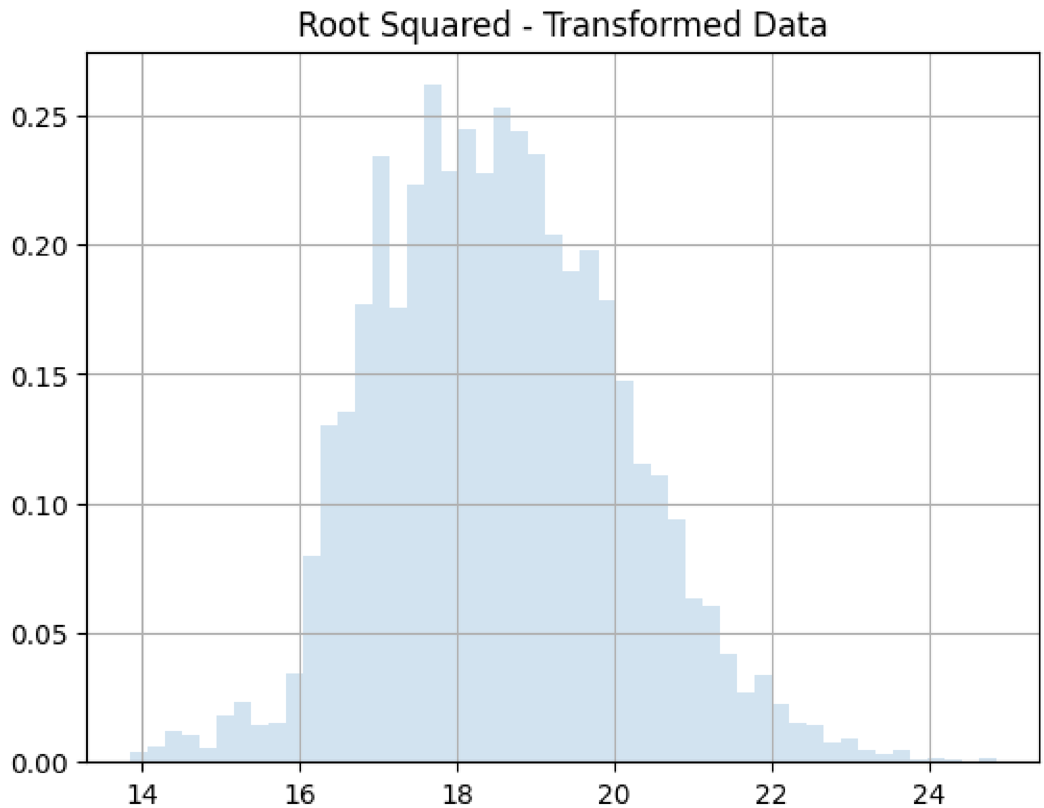
Distribucion Asimetrica Positiva:

[349]:
```
y_skew = skewnorm.rvs(y)
Asimetric_Dist(y_skew, 'Original - Positive Skew')
```



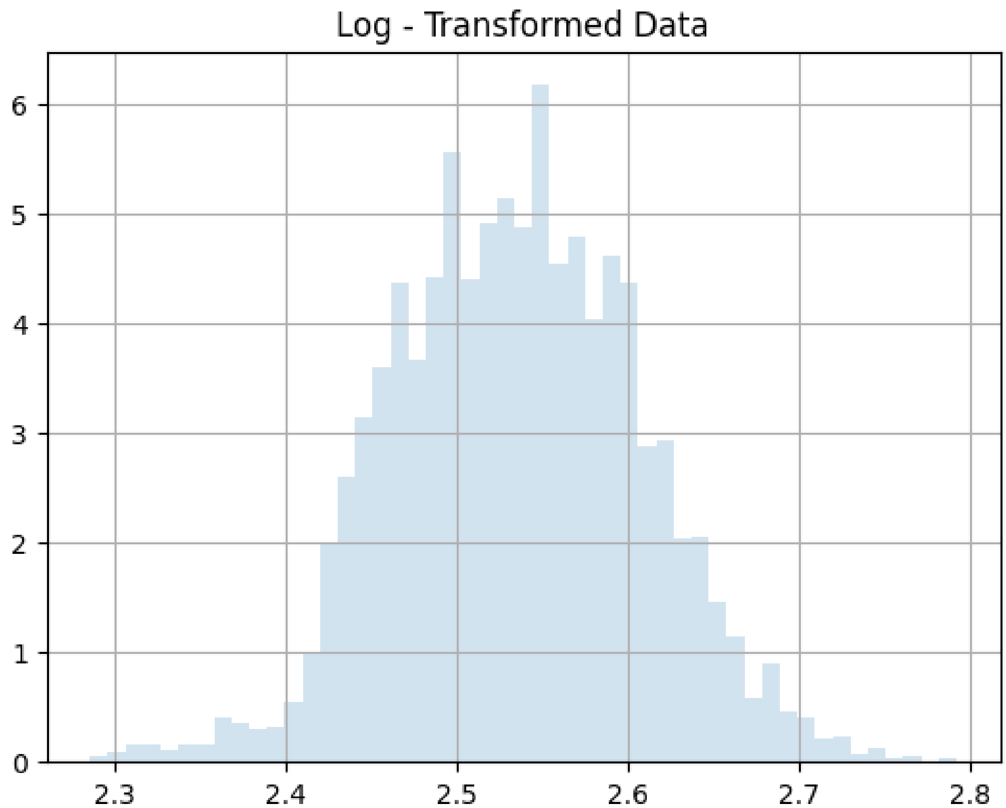
Distribucion Asimetrica Positiva:

[350]:
```
y_root = np.sqrt(y + abs(min(y)))
Asimetric_Dist(y_root, 'Root Squared - Transformed Data')
```



Distribucion Asimetrica Positiva:

[351]:
```
y_log = np.log10(1 + y + abs(min(y)))
Asimetric_Dist(y_log, 'Log - Transformed Data')
```



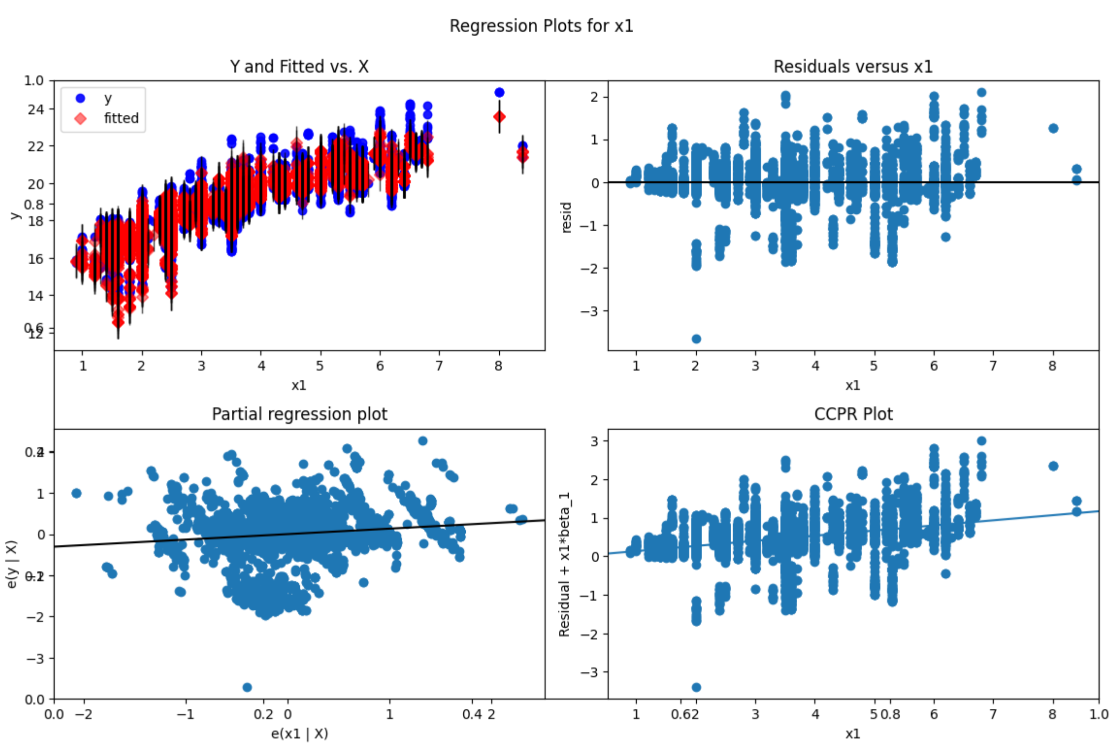
[352]:

```
model = sm.OLS(y_root, X)
results = model.fit()
print('\n', results.params)
print('\n', 'R2: ', results.rsquared)
print(results.summary())
```

```
 [ 1.94367810e+01  1.33053149e-01  1.85578182e-01 -5.95956234e-03
  1.18164557e-01 -6.19298157e-03 -1.19108371e-01]

 R2:  0.9152323690604602
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.915
Model:                            OLS   Adj. R-squared:                  0.915
Method:                 Least Squares   F-statistic:                 1.328e+04
Date:                Sat, 07 Oct 2023   Prob (F-statistic):               0.00
Time:                        04:12:13   Log-Likelihood:                -4635.1
No. Observations:                7385   AIC:                             9284.
Df Residuals:                    7378   BIC:                             9333.
Df Model:                           6
```

```
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         19.4368      0.105    185.164      0.000      19.231      19.643
x1             0.1331      0.011     11.687      0.000       0.111       0.155
x2             0.1856      0.008     23.303      0.000       0.170       0.201
x3            -0.0060      0.068     -0.087      0.931      -0.140       0.128
x4             0.1182      0.056      2.093      0.036       0.007       0.229
x5            -0.0062      0.124     -0.050      0.960      -0.250       0.237
x6            -0.1191      0.002    -60.623      0.000      -0.123      -0.115
==============================================================================
Omnibus:                     1399.064   Durbin-Watson:                   1.617
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             7822.151
Skew:                          -0.794   Prob(JB):                         0.00
Kurtosis:                       7.786   Cond. No.                         987.
==============================================================================

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

[353]:
```python
fig, ax = plt.subplots(figsize=(12, 8))
sm.graphics.plot_regress_exog(results, 1, fig=fig)
plt.show()
```

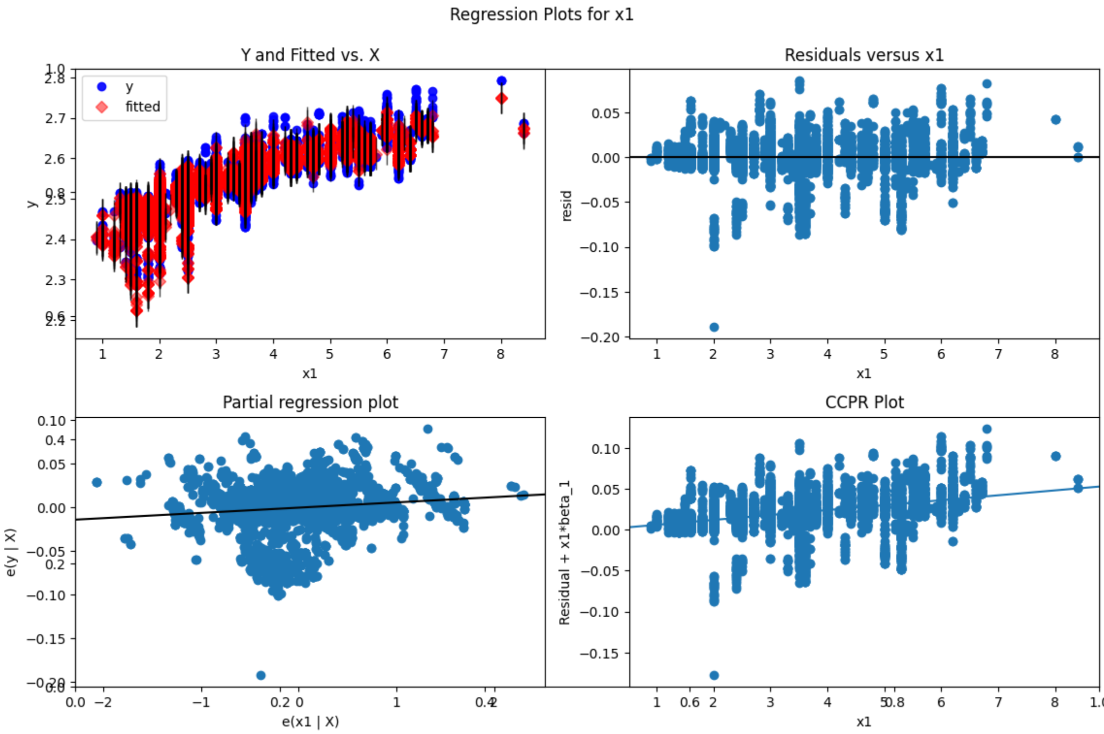


##Evaluación del impacto de cada variable independiente

[354]:
```python
model = sm.OLS(y_log, X)
results = model.fit()
print('\n', results.params)
print('\n', 'R2: ', results.rsquared)
print(results.summary())
```

```
 [ 2.64703621e+00  6.01517403e-03  7.91678618e-03 -3.12833476e-04
  5.42435642e-03 -2.91406589e-03 -6.86385862e-03]

 R2:  0.9250969146244696
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.925
Model:                            OLS   Adj. R-squared:                  0.925
Method:                 Least Squares   F-statistic:                 1.519e+04
Date:                Sat, 07 Oct 2023   Prob (F-statistic):               0.00
Time:                        04:12:14   Log-Likelihood:                 18468.
No. Observations:                7385   AIC:                        -3.692e+04
Df Residuals:                    7378   BIC:                        -3.687e+04
Df Model:                           6
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          2.6470      0.005    575.888      0.000       2.638       2.656
x1             0.0060      0.000     12.066      0.000       0.005       0.007
x2             0.0079      0.000     22.703      0.000       0.007       0.009
x3            -0.0003      0.003     -0.104      0.917      -0.006       0.006
x4             0.0054      0.002      2.194      0.028       0.001       0.010
x5            -0.0029      0.005     -0.536      0.592      -0.014       0.008
x6            -0.0069    8.6e-05    -79.783      0.000      -0.007      -0.007
==============================================================================
Omnibus:                     1698.750   Durbin-Watson:                   1.615
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            10030.801
Skew:                          -0.971   Prob(JB):                         0.00
Kurtosis:                       8.369   Cond. No.                         987.
==============================================================================

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

[355]:
```python
fig, ax = plt.subplots(figsize=(12, 8))
sm.graphics.plot_regress_exog(results, 1, fig=fig)
plt.show()
```



[356]:
```python
x = np.array(df[['Engine Size(L)', 'Cylinders', 'Fuel Consumption Hwy (L/100␣
↪km)', 'Fuel Consumption Comb (L/100 km)', 'Fuel Consumption Comb (mpg)']])
X = sm.add_constant(x)

model = sm.OLS(y_log, X)
results = model.fit()
print('\n', results.params)
print('\n', 'R2: ', results.rsquared)
print(results.summary())
```

```
 [ 2.64703287  0.00601547  0.00791563  0.00567639 -0.00347904 -0.00686376]

 R2:  0.9250968039653008
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.925
Model:                            OLS   Adj. R-squared:                  0.925
Method:                 Least Squares   F-statistic:                 1.823e+04
Date:                Sat, 07 Oct 2023   Prob (F-statistic):               0.00
```

```
Time:                        04:12:16   Log-Likelihood:                 18468.
No. Observations:                7385   AIC:                        -3.692e+04
Df Residuals:                    7379   BIC:                        -3.688e+04
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          2.6470      0.005    575.940      0.000       2.638       2.656
x1             0.0060      0.000     12.068      0.000       0.005       0.007
x2             0.0079      0.000     22.713      0.000       0.007       0.009
x3             0.0057      0.001     10.604      0.000       0.005       0.007
x4            -0.0035      0.001     -6.477      0.000      -0.005      -0.002
x5            -0.0069    8.6e-05    -79.791      0.000      -0.007      -0.007
==============================================================================
Omnibus:                     1698.926   Durbin-Watson:                   1.615
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            10031.533
Skew:                          -0.971   Prob(JB):                         0.00
Kurtosis:                       8.370   Cond. No.                         637.
==============================================================================

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

[357]:

```
x = np.array(df[['Engine Size(L)', 'Cylinders', 'Fuel Consumption City (L/100␣
↪km)', 'Fuel Consumption Hwy (L/100 km)', 'Fuel Consumption Comb (mpg)']])
X = sm.add_constant(x)

model = sm.OLS(y_log, X)
results = model.fit()
print('\n', results.params)
print('\n', 'R2: ', results.rsquared)
print(results.summary())
```

```
 [ 2.64693737e+00  6.01076984e-03  7.91847138e-03 -1.91065345e-03
  4.11171738e-03 -6.86193779e-03]

 R2:  0.9250939993765741
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.925
Model:                            OLS   Adj. R-squared:                  0.925
Method:                 Least Squares   F-statistic:                 1.823e+04
Date:                Sat, 07 Oct 2023   Prob (F-statistic):               0.00
Time:                        04:12:16   Log-Likelihood:                 18468.
No. Observations:                7385   AIC:                        -3.692e+04
```

```
Df Residuals:                    7379   BIC:                        -3.688e+04
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          2.6469      0.005    576.358      0.000       2.638       2.656
x1             0.0060      0.000     12.060      0.000       0.005       0.007
x2             0.0079      0.000     22.710      0.000       0.007       0.009
x3            -0.0019      0.000     -6.455      0.000      -0.002      -0.001
x4             0.0041      0.000     12.187      0.000       0.003       0.005
x5            -0.0069    8.6e-05    -79.833      0.000      -0.007      -0.007
==============================================================================
Omnibus:                     1699.764   Durbin-Watson:                   1.615
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            10035.420
Skew:                          -0.971   Prob(JB):                         0.00
Kurtosis:                       8.370   Cond. No.                         646.
==============================================================================

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