Actividad 1: Regresión Lineal Simple/Múltiple

Mario Alberto Castañeda Martínez - A01640152

Victor Hugo Arreola Elenes - A01635682

Luis Manuel Orozco Yáñez - A01707822

Fernando Ojeda Marín - A01639252

Se importan librerías necesarias:

```
from google.colab import drive
drive.mount('/content/drive')
%cd "/content/drive/MyDrive/CONCENTRACION AI/data"
```

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

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
import  scipy.stats as stats
import seaborn as sns
from scipy.stats import norm, uniform, skewnorm
import statsmodels.formula.api as smf
```

Se importa la base de datos:

```
df = pd.read_csv('CO2 Emissions_Canada.csv')
```

```
df.shape
```

```
(7385, 12)
```

```
df.columns
```

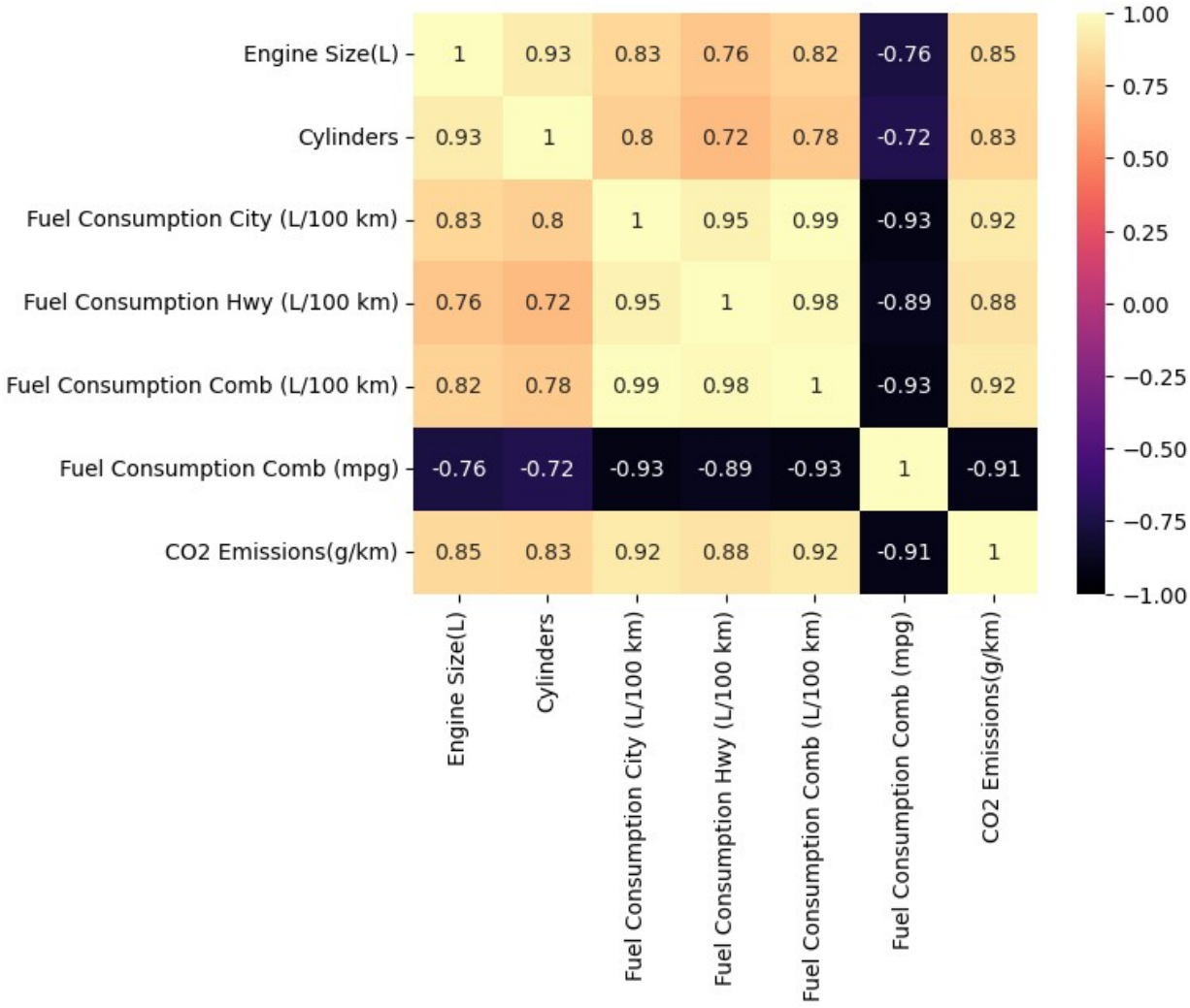
```
Index(['Make', 'Model', 'Vehicle Class', 'Engine Size(L)',
'Cylinders',
       'Transmission', 'Fuel Type', 'Fuel Consumption City (L/100
km)',
       'Fuel Consumption Hwy (L/100 km)', 'Fuel Consumption Comb
(L/100 km)',
       'Fuel Consumption Comb (mpg)', 'CO2 Emissions(g/km)'],
      dtype='object')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

Se hace un análisis de correlación de todas las variables para ver sobretodo cuales son las que más influyen en la emisión de CO2.

```
sns.heatmap(df.corr(),vmin=-1, vmax=1, cmap='magma',annot = True)
```

```
<ipython-input-6-0c57e6232ea7>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it
will default to False. Select only valid columns or specify the value
of numeric_only to silence this warning.
  sns.heatmap(df.corr(),vmin=-1, vmax=1, cmap='magma',annot = True)
```

```
<Axes: >
```



¿Cuáles son las características que más influyen en la emisión de CO2?

De acuerdo con un análisis de correlación de las varibles con la variable de emisión de CO2, se obtiene que las características que más influyen son la del Consumo de combustible en ciudad y la de Consumo de combustible combinado. Esto además se ve reflejado en los resultado de los

modelos de regresión, donde con ambas variables se obtienen las r^2 más altas a comparación de las otras características.

¿Habrá alguna diferencia en las emisiones de CO2 cuando el consumo de combistible es para ciudad y carretera se consideren por separado?

Al estar en carretera, el consumo de combustible se vuelve más eficiente debido a la velocidad constante, en cambio en la ciudad el consumo es mayor por las diferentes variables de la ciudad (tráfico, frenadas, semáforos, etc). Esto se ve reflejado en las gráficas de los datos comparados con el consumo de combustible, donde se observa que en la ciudad hay un mayor consumo a comparación del consumo en Highway o carretera.

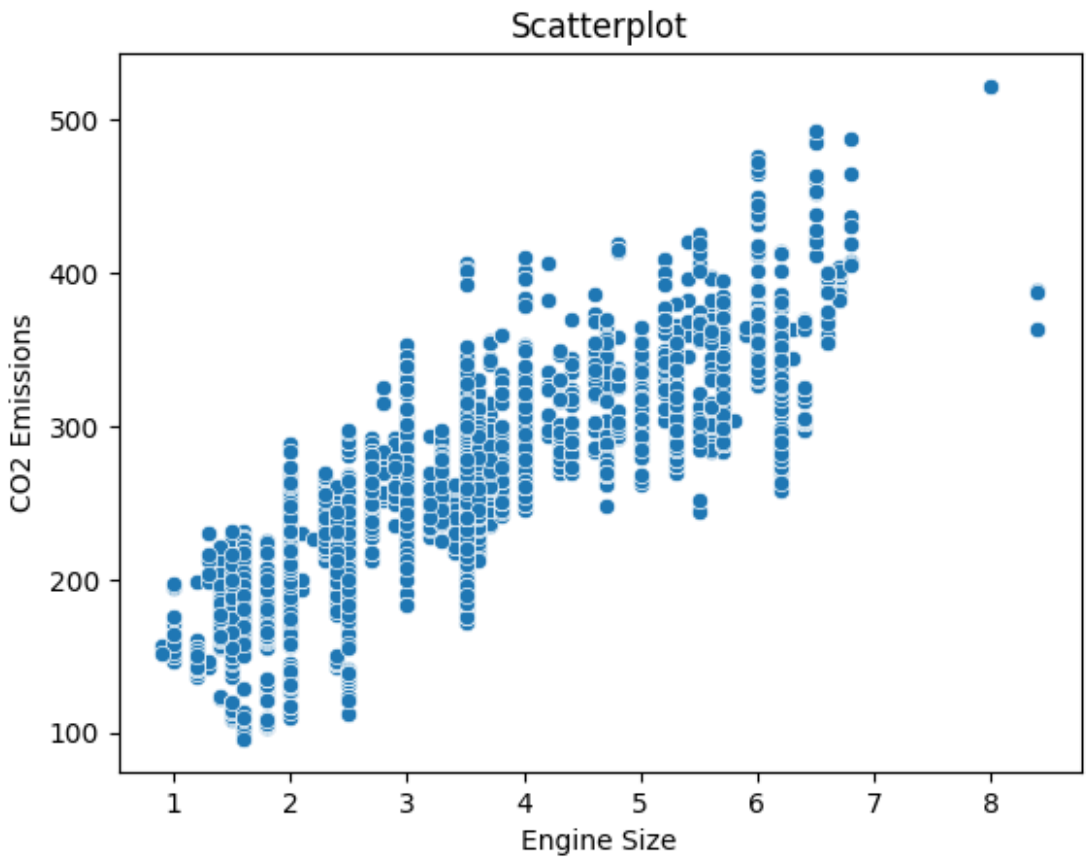
Lo siguiente en realizarse es:

- Gráfica de comprensión de la precisión entre variable X y Y
- Cálculo del coeficiente de determinación
- Gráfica de errores (Predicho vs Residuo)
- Gráfica QQ-Plot

Se realizarán OLS para cada variable vs la variable de respuesta (CO2 Emissions), obteniendo los elementos de la lista anterior. Déspues se procederá con una transformación de la variable de respuesta.

Primero con la variable Engine Size(L)

```
sns.scatterplot(x='Engine Size(L)', y='CO2 Emissions(g/km)', data=df)
plt.xlabel('Engine Size')
plt.ylabel('CO2 Emissions')
plt.title('Scatterplot')
plt.show()
```




```
x=df['Engine Size(L)']
y=df['CO2 Emissions(g/km)']

x = sm.add_constant(x)
model= sm.OLS(y,x)

result = model.fit()
print(result.params)
```

```
const              134.365893
Engine Size(L)      36.777315
dtype: float64
```

```
prediction = result.params['const'] + result.params['Engine Size(L)']
* df['Engine Size(L)']
```

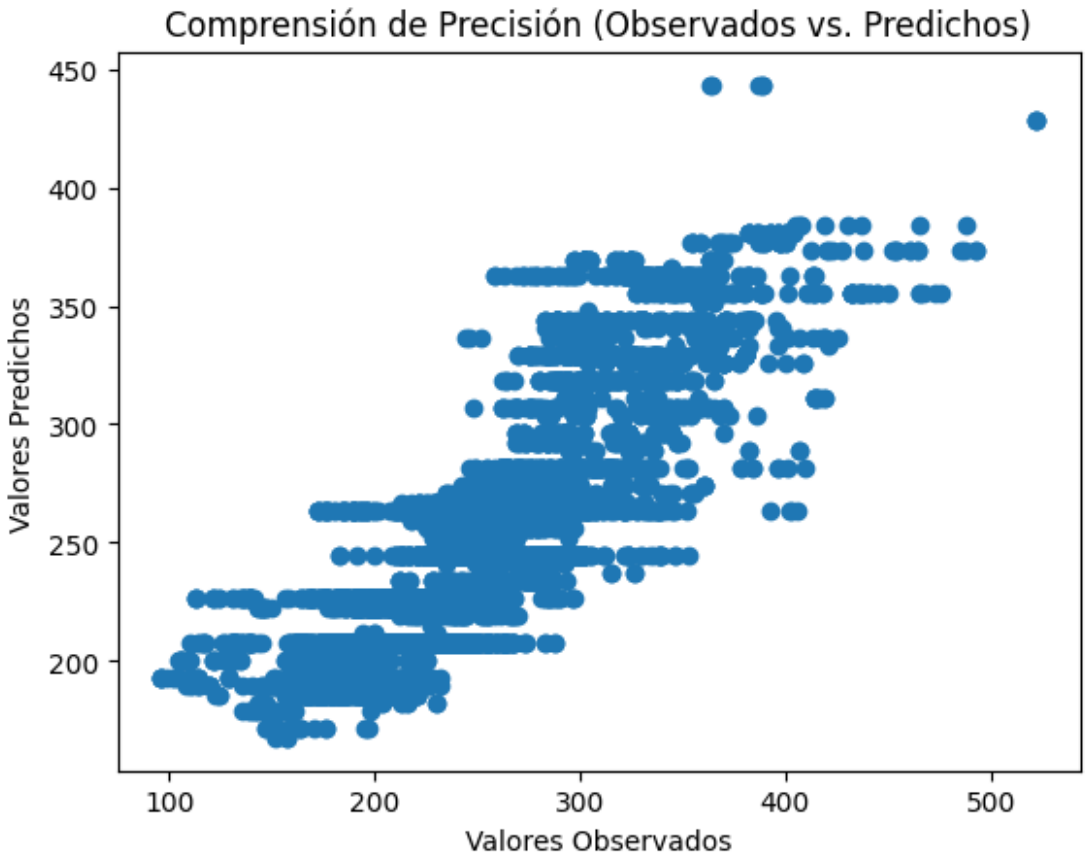
Se obtiene el coeficiente de determinación del modelo:

```
from sklearn.metrics import mean_squared_error, mean_absolute_error,
r2_score, make_scorer
print('Coeficiente de determinación: ', result.rsquared)
```

```
Coeficiente de determinación:  0.7244472046524082
```

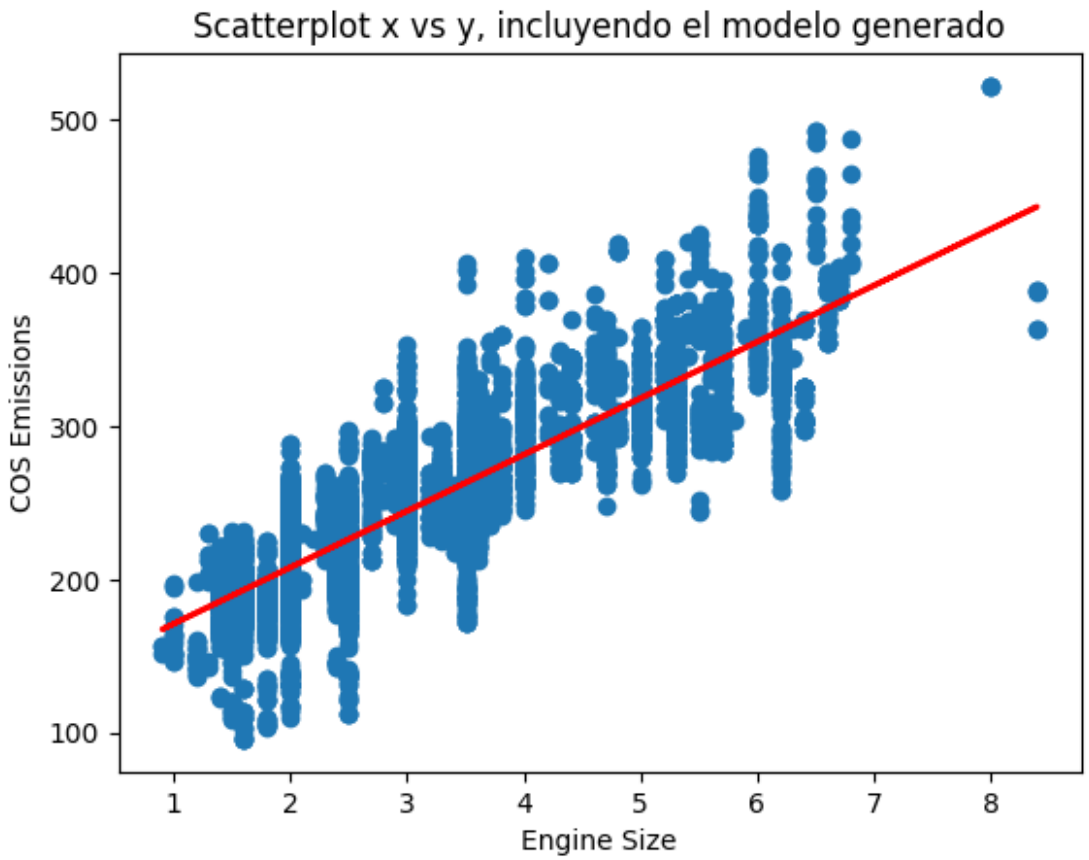
Se obtiene una gráfica para comprender la precisión del modelo, entre los datos observados u originales y los predichos por el modelo.

```
yh = result.predict(x)
plt.scatter(y, yh)
plt.xlabel('Valores Observados')
plt.ylabel('Valores Predichos')
plt.title('Comprensión de Precisión (Observados vs. Predichos)')
plt.show()
```



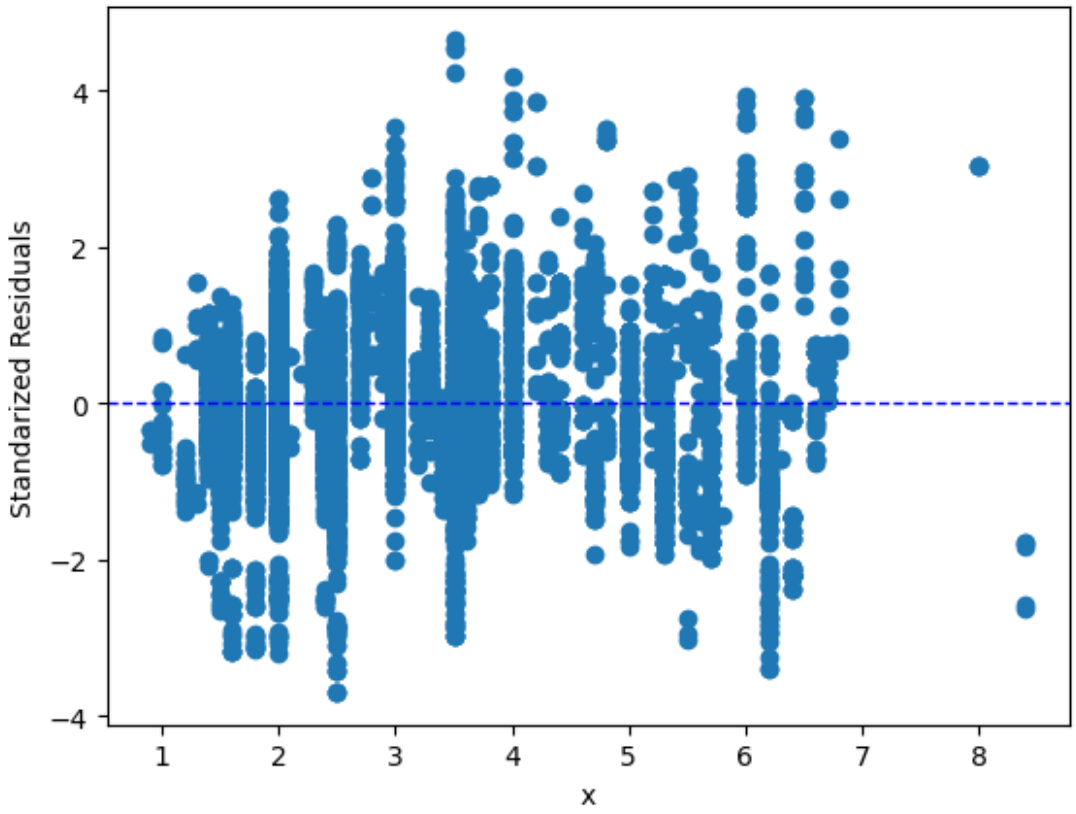
Se obtiene una gráfica entre x y y, mostrando además una línea del modelo generado.

```
plt.scatter(df['Engine Size(L)'], df['CO2 Emissions(g/km)'])
plt.plot(df['Engine Size(L)'], prediction, color='red', linewidth=2)
plt.xlabel('Engine Size')
plt.ylabel('COS Emissions')
plt.title('Scatterplot x vs y, incluyendo el modelo generado')
plt.show()
```

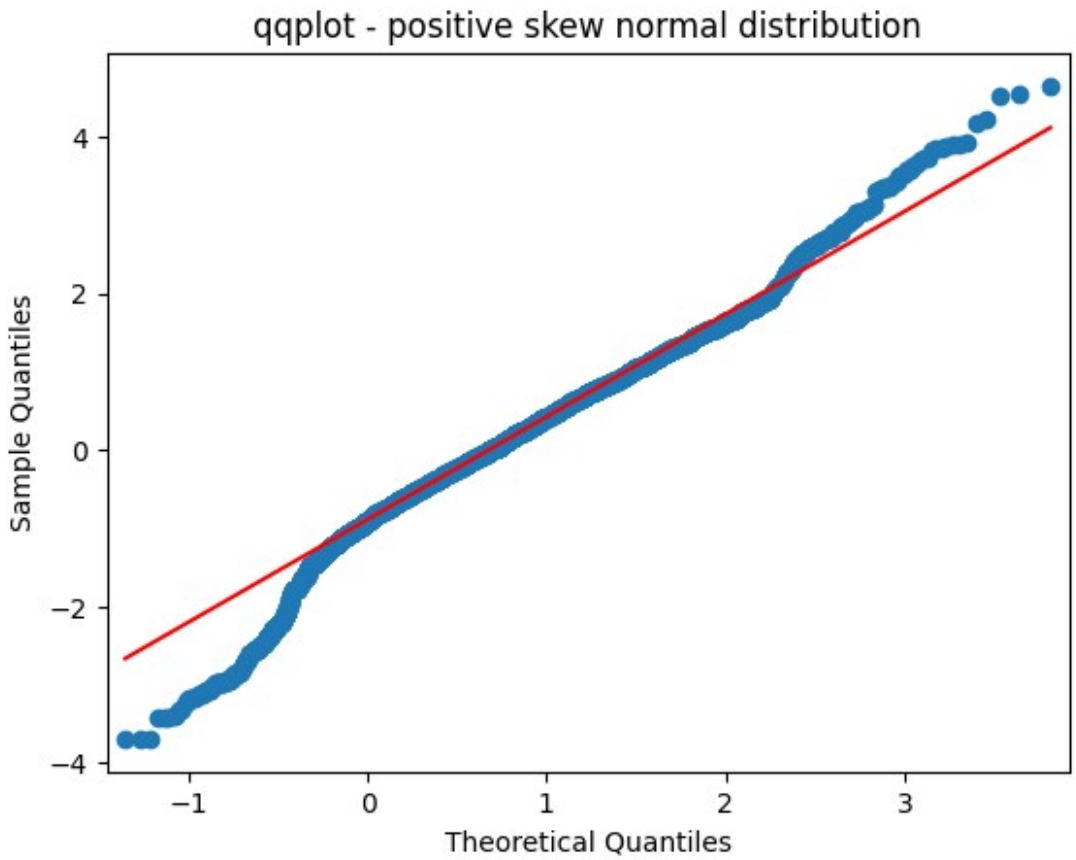



```
influence= result.get_influence()
standarized_residuals = influence.resid_studentized_internal
print(standarized_residuals)
plt.scatter(df['Engine Size(L)'], standarized_residuals)
plt.xlabel('x')
plt.ylabel('Standarized Residuals')
plt.axhline(y=0, color='b', linestyle='--', linewidth=1)
plt.show()
```

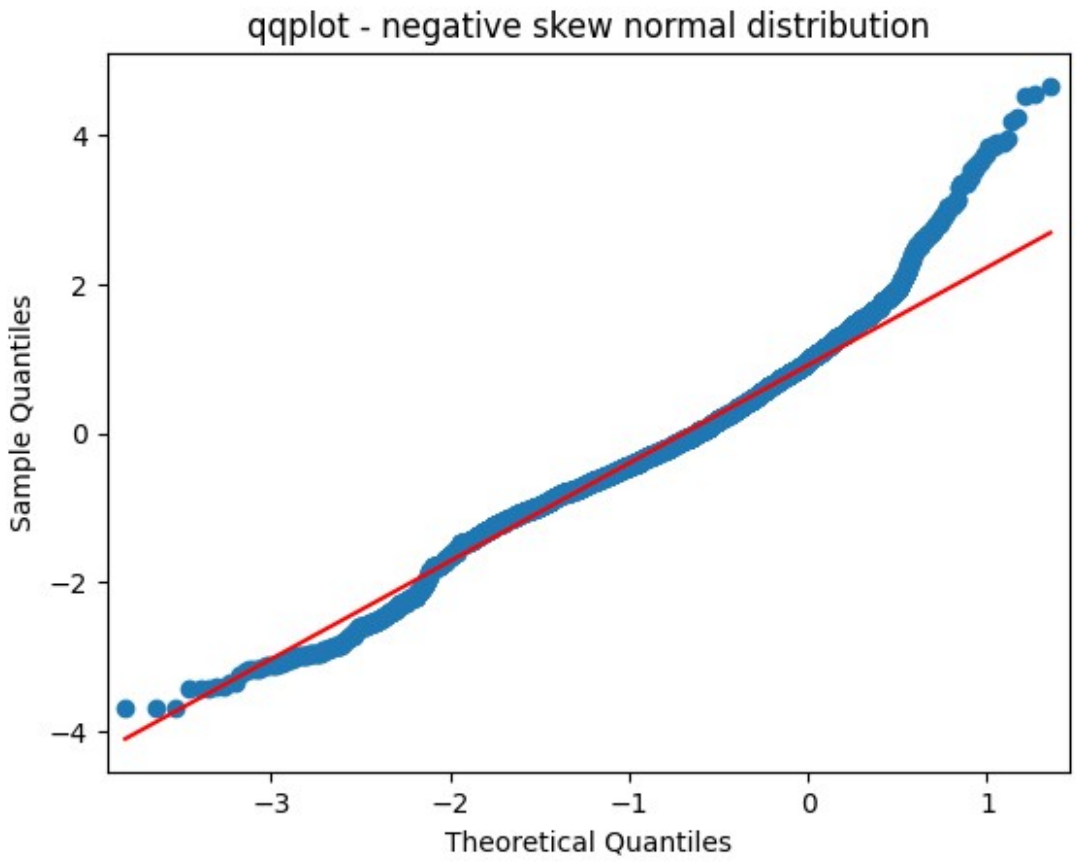
```
[-0.38811864 -0.05311662 -1.74302727 ...  1.04447119  0.78400031
  1.30494206]
```




```
fig = sm.qqplot(standarized_residuals, dist=skewnorm(2), line='q')
plt.title('qqplot - positive skew normal distribution')
plt.show()
```



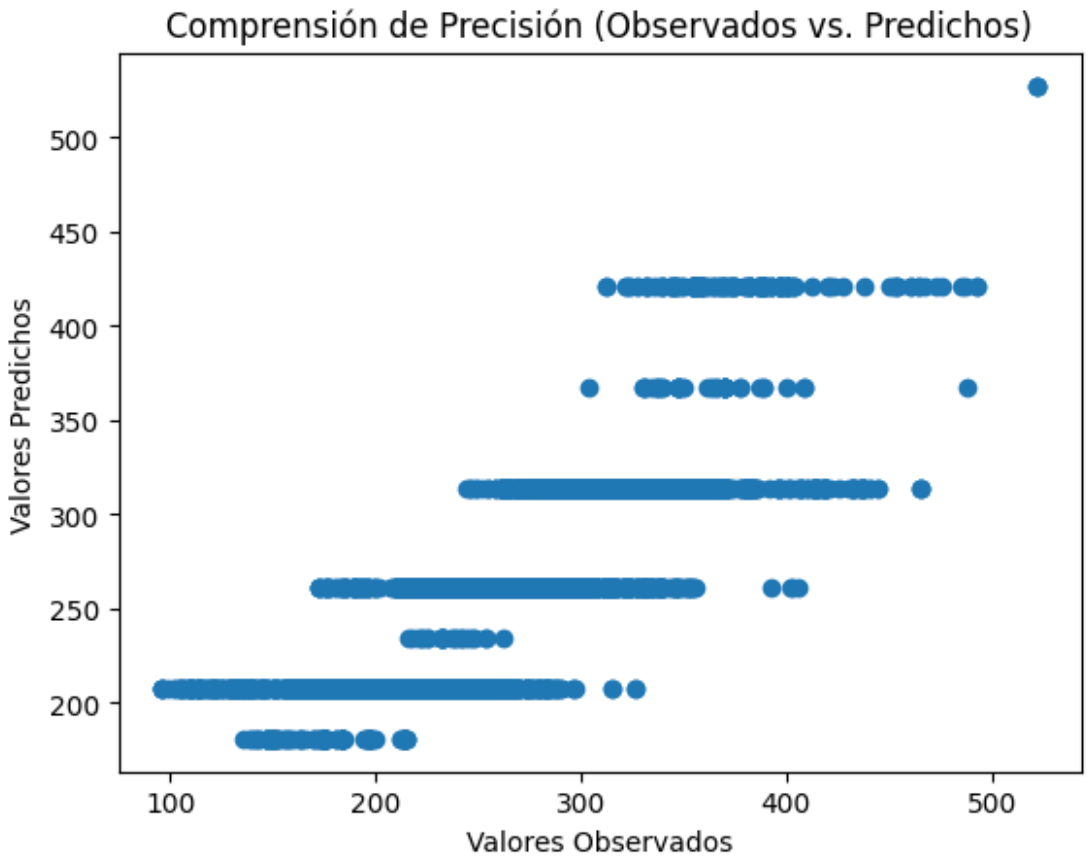

```
fig = sm.qqplot(standarized_residuals, dist=skewnorm(-2), line='q')
plt.title('qqplot - negative skew normal distribution')
plt.show()
```



Con la variable Cylinders:

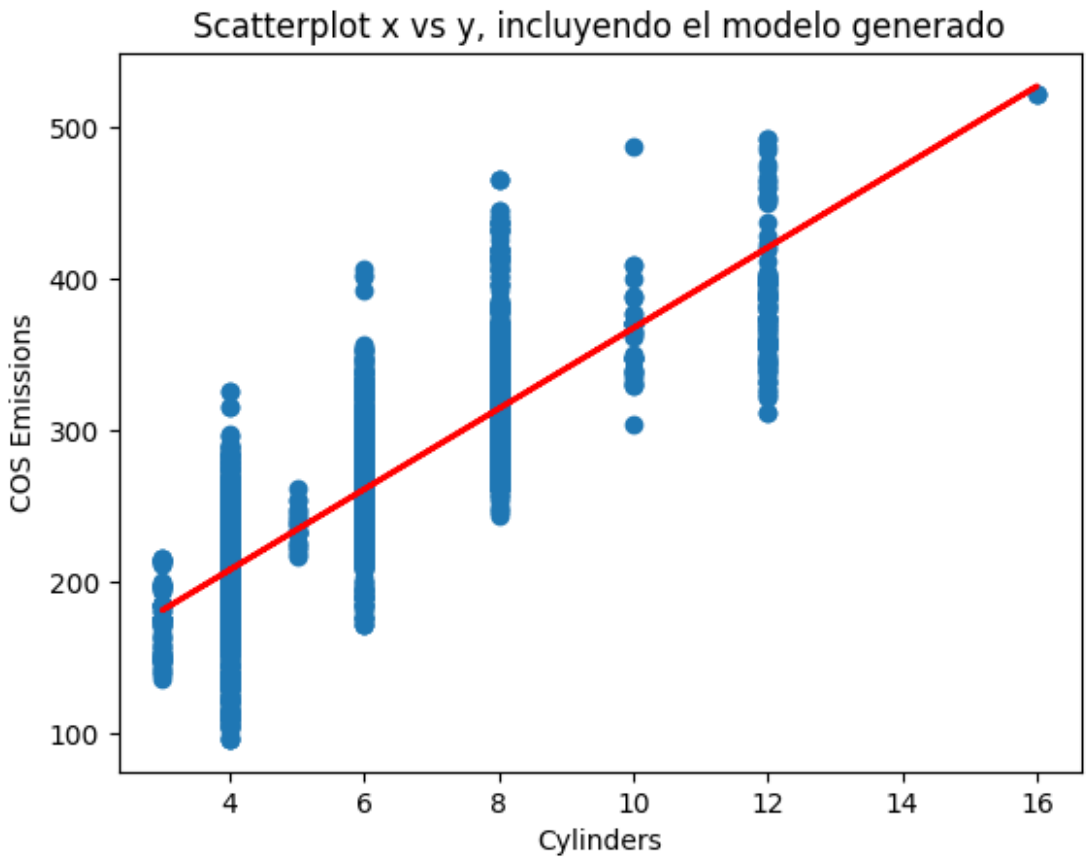
```
x=df['Cylinders']
y=df['CO2 Emissions(g/km)']

x = sm.add_constant(x)
model= sm.OLS(y,x)
result = model.fit()
print('Coeficiente de determinación: ', result.rsquared)
yh = result.predict(x)
plt.scatter(y, yh)
plt.xlabel('Valores Observados')
plt.ylabel('Valores Predichos')
plt.title('Comprensión de Precisión (Observados vs. Predichos)')
plt.show()
```

```
Coeficiente de determinación:  0.6932953649936133
```



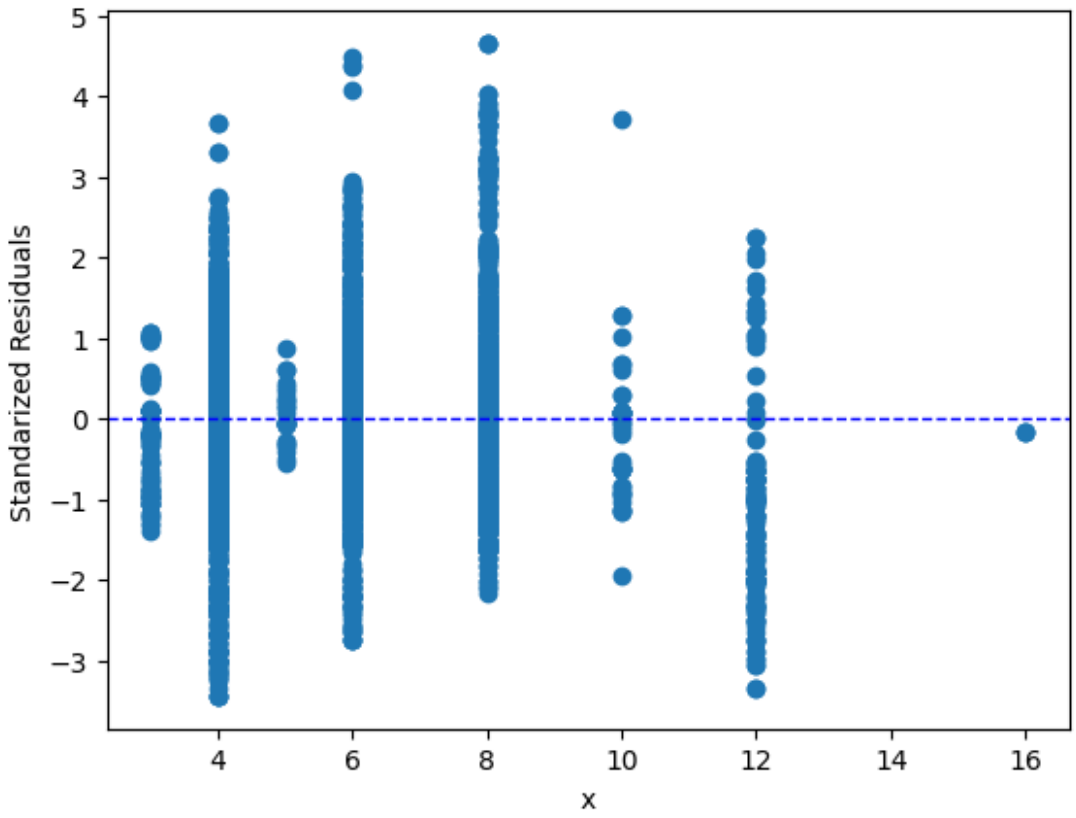
Se obtiene un R^2 del 69%

```
prediction = result.params['const'] + result.params['Cylinders'] *
df['Cylinders']
plt.scatter(df['Cylinders'], df['CO2 Emissions(g/km)'])
plt.plot(df['Cylinders'], prediction, color='red', linewidth=2)
plt.xlabel('Cylinders')
plt.ylabel('COS Emissions')
plt.title('Scatterplot x vs y, incluyendo el modelo generado')
plt.show()
```

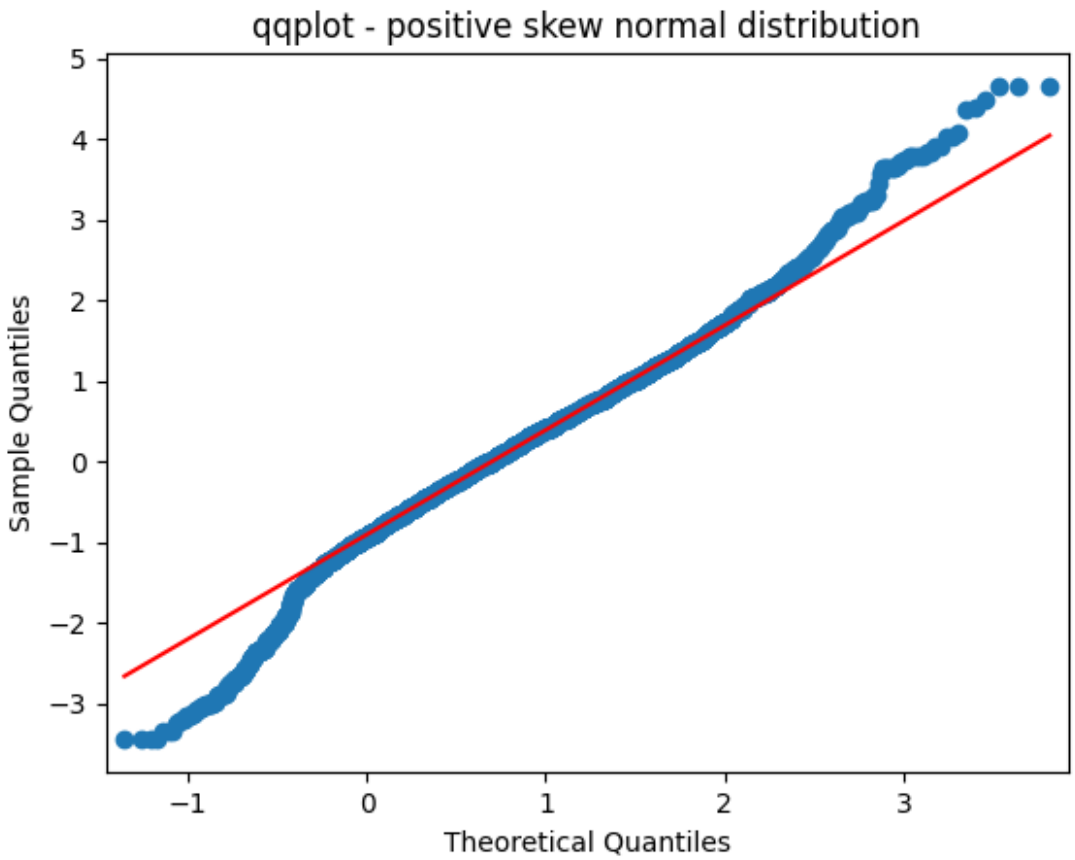



```
influence= result.get_influence()
standarized_residuals = influence.resid_studentized_internal
print(standarized_residuals)
plt.scatter(df['Cylinders'], standarized_residuals)
plt.xlabel('x')
plt.ylabel('Standarized Residuals')
plt.axhline(y=0, color='b', linestyle='--', linewidth=1)
plt.show()
```

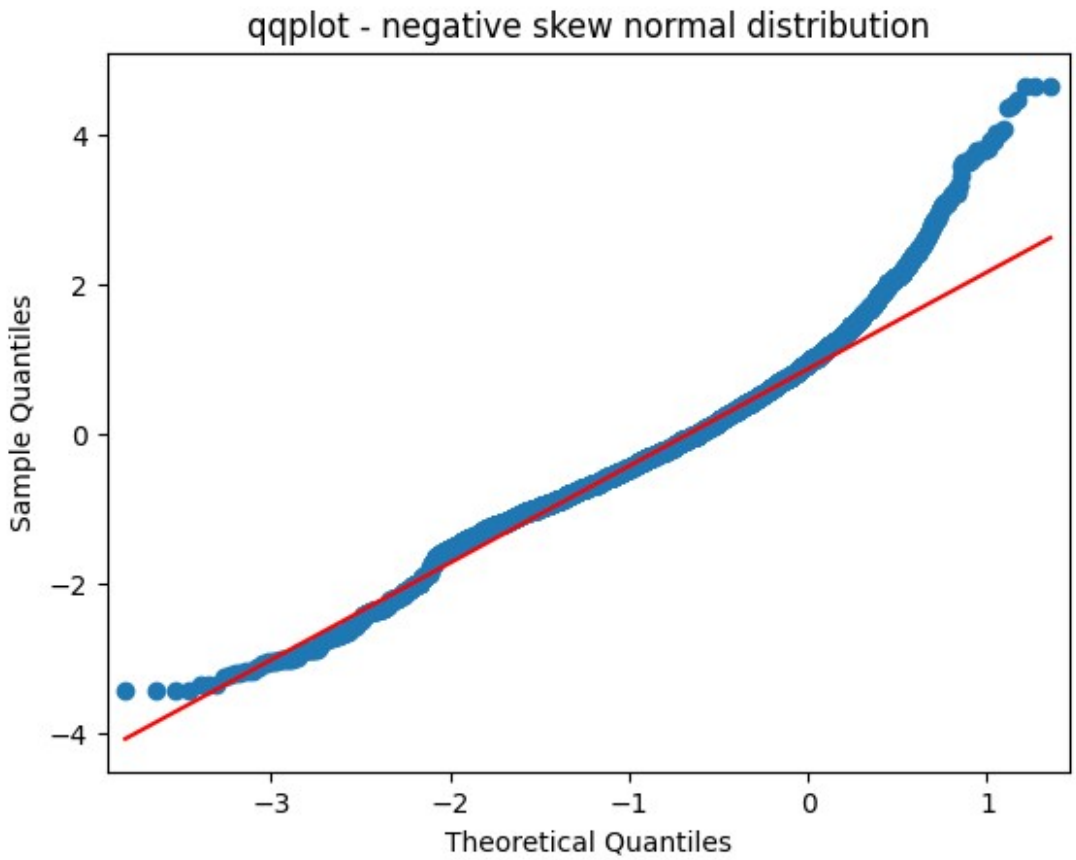
```
[-0.35637932  0.41515065 -2.20805125 ...  1.00151342  0.75462383
  1.24840301]
```




```
fig = sm.qqplot(standarized_residuals, dist=skewnorm(2), line='q')
plt.title('qqplot - positive skew normal distribution')
plt.show()
```

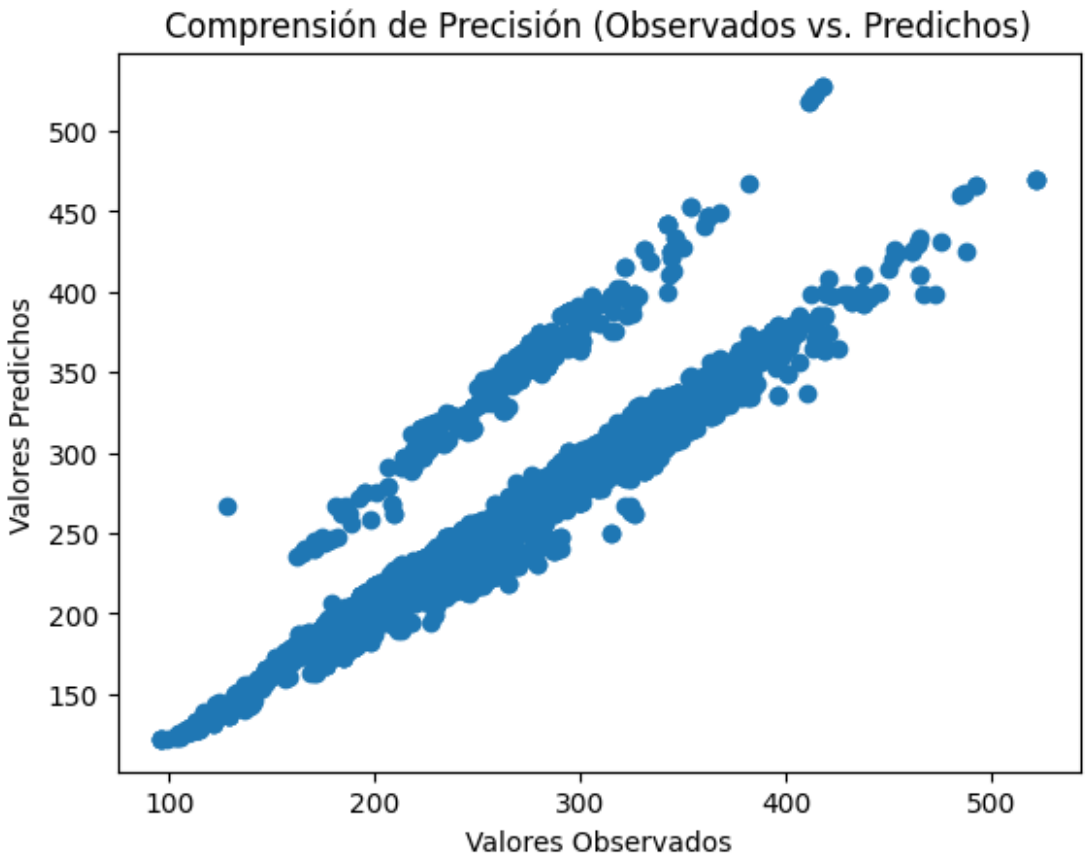



```
fig = sm.qqplot(standarized_residuals, dist=skewnorm(-2), line='q')
plt.title('qqplot - negative skew normal distribution')
plt.show()
```



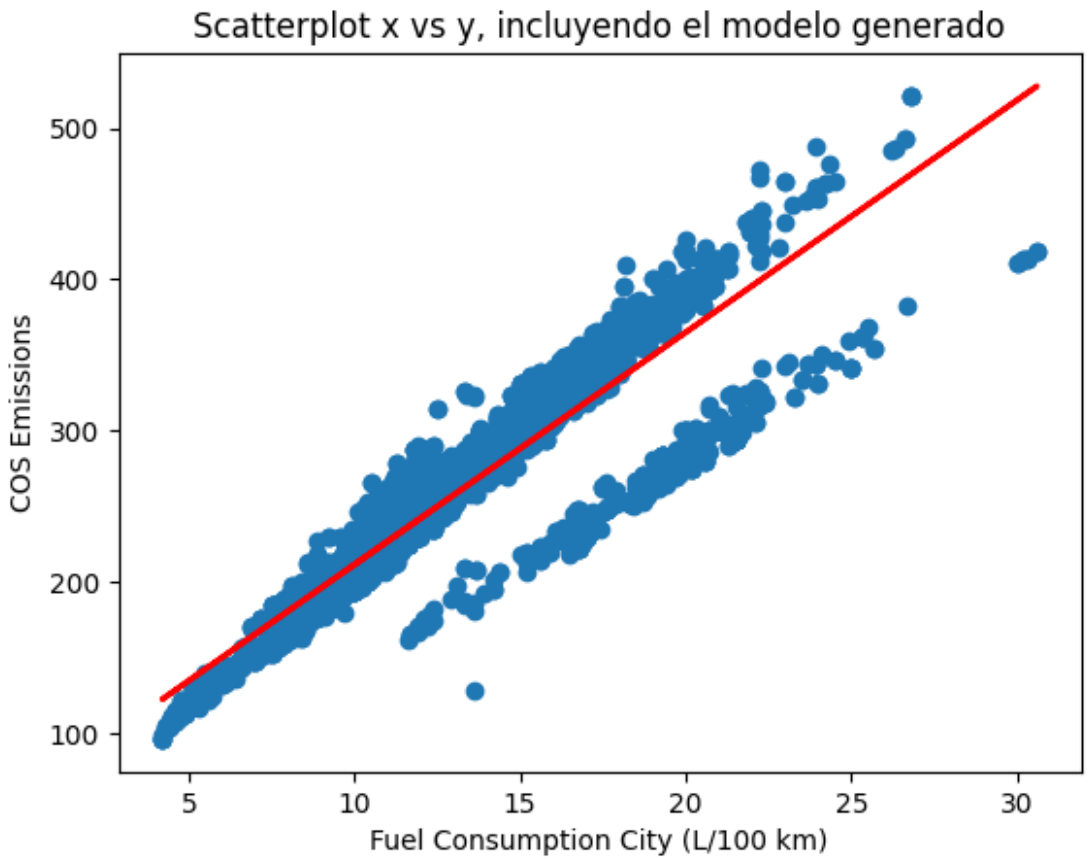
Con la variable Fuel Consumption City (L/100 km):

```
x=df['Fuel Consumption City (L/100 km)']
y=df['CO2 Emissions(g/km)']
x = sm.add_constant(x)
model= sm.OLS(y,x)
result = model.fit()
print('Coeficiente de determinación: ', result.rsquared)
yh = result.predict(x)
plt.scatter(y, yh)
plt.xlabel('Valores Observados')
plt.ylabel('Valores Predichos')
plt.title('Comprensión de Precisión (Observados vs. Predichos)')
plt.show()

Coeficiente de determinación:  0.8456503198972763
```



Se obtiene un R^2 del 84%

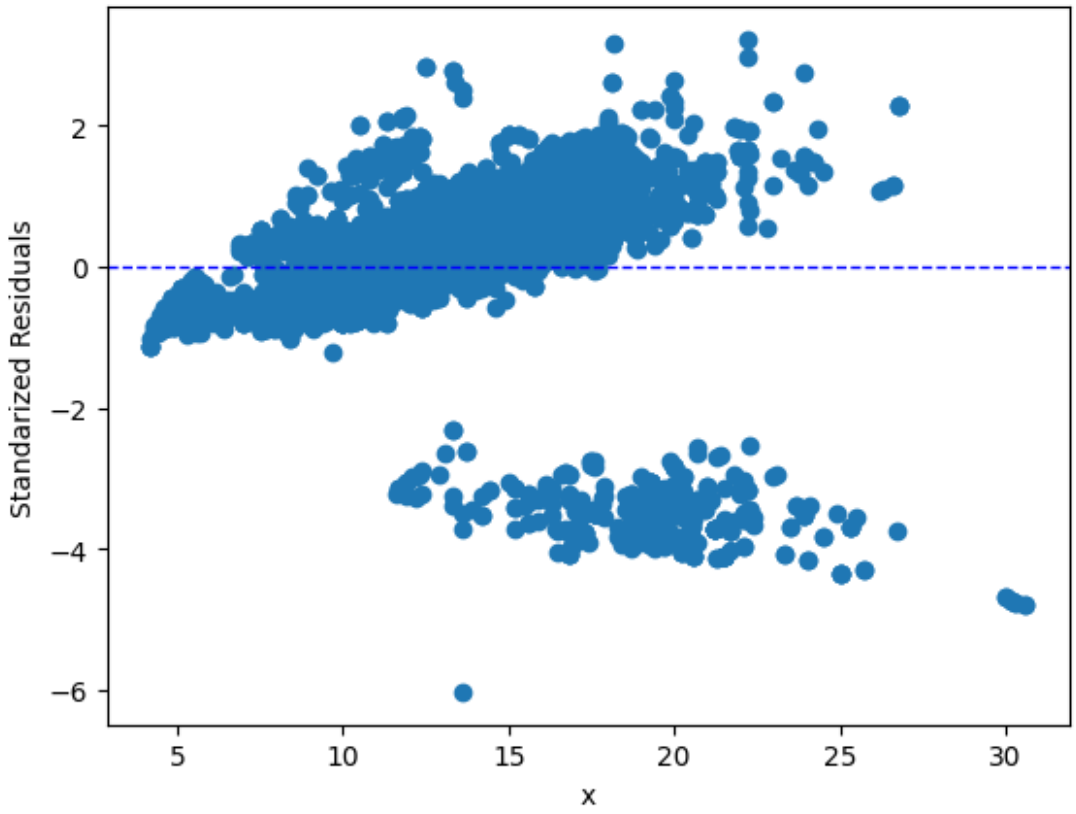
```
prediction = result.params['const'] + result.params['Fuel Consumption
City (L/100 km)'] * df['Fuel Consumption City (L/100 km)']
plt.scatter(df['Fuel Consumption City (L/100 km)'], df['CO2
Emissions(g/km)'])
plt.plot(df['Fuel Consumption City (L/100 km)'], prediction,
color='red', linewidth=2)
plt.xlabel('Fuel Consumption City (L/100 km)')
plt.ylabel('COS Emissions')
plt.title('Scatterplot x vs y, incluyendo el modelo generado')
plt.show()
```



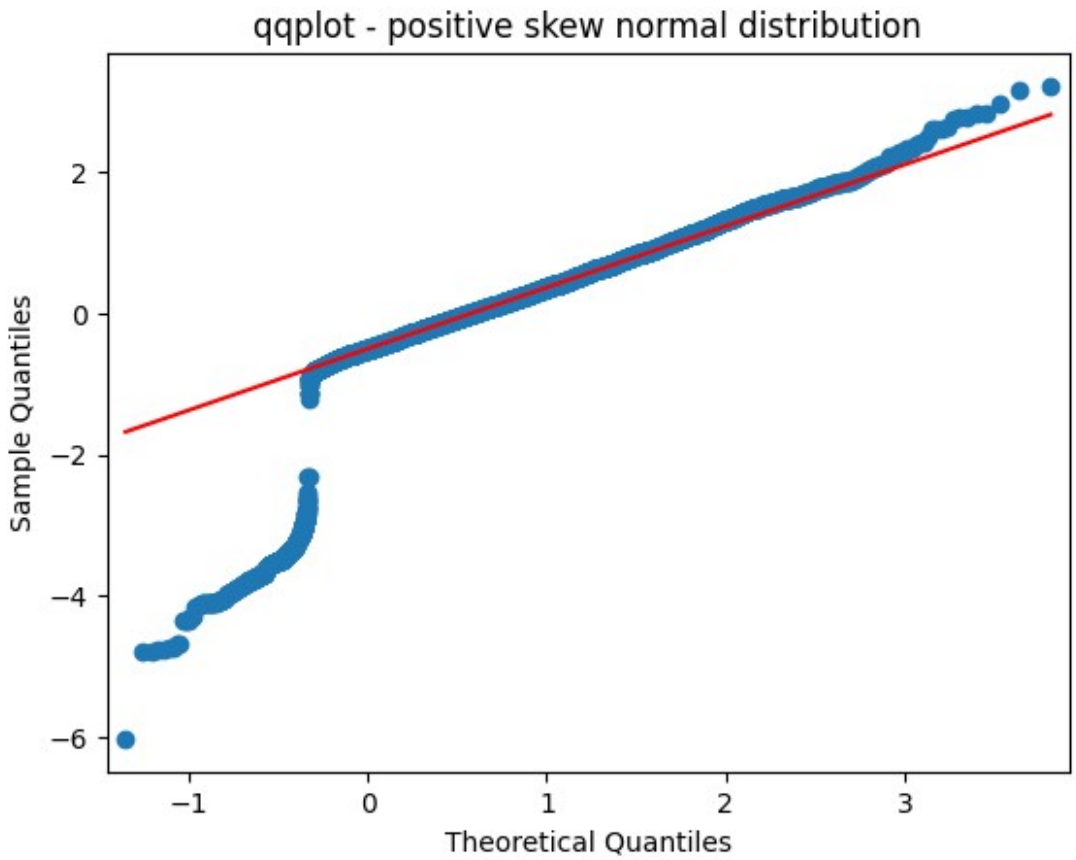
Se obtienen los errores estandarizados del modelo:

```
influence= result.get_influence()
standarized_residuals = influence.resid_studentized_internal
print(standarized_residuals)
plt.scatter(df['Fuel Consumption City (L/100 km)'],
standarized_residuals)
plt.xlabel('x')
plt.ylabel('Standarized Residuals')
plt.axhline(y=0, color='b', linestyle='--', linewidth=1)
plt.show()
```

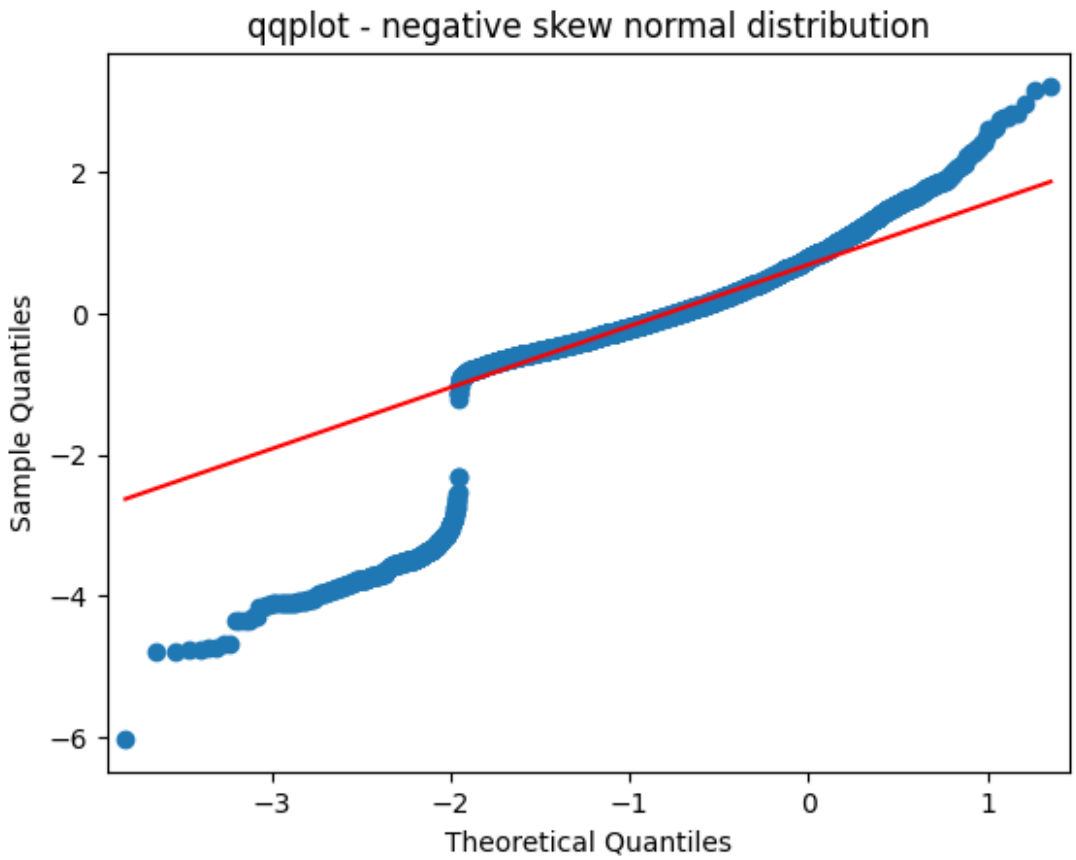
```
[-0.5980394  -0.37982853 -0.60022109 ...  0.11233374  0.09868503
  0.12598264]
```




```
fig = sm.qqplot(standarized_residuals, dist=skewnorm(2), line='q')
plt.title('qqplot - positive skew normal distribution')
plt.show()
```



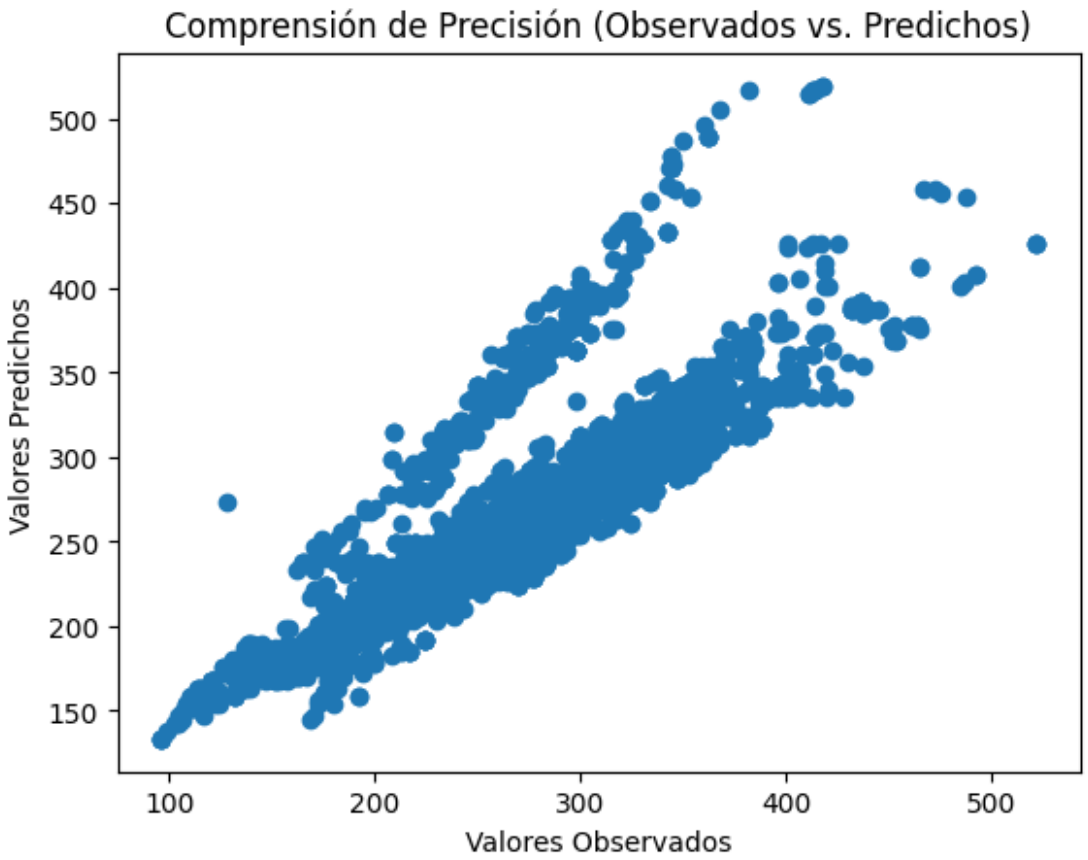

```
fig = sm.qqplot(standarized_residuals, dist=skewnorm(-2), line='q')
plt.title('qqplot - negative skew normal distribution')
plt.show()
```



Con la variable Fuel Consumption Hwy (L/100 km):

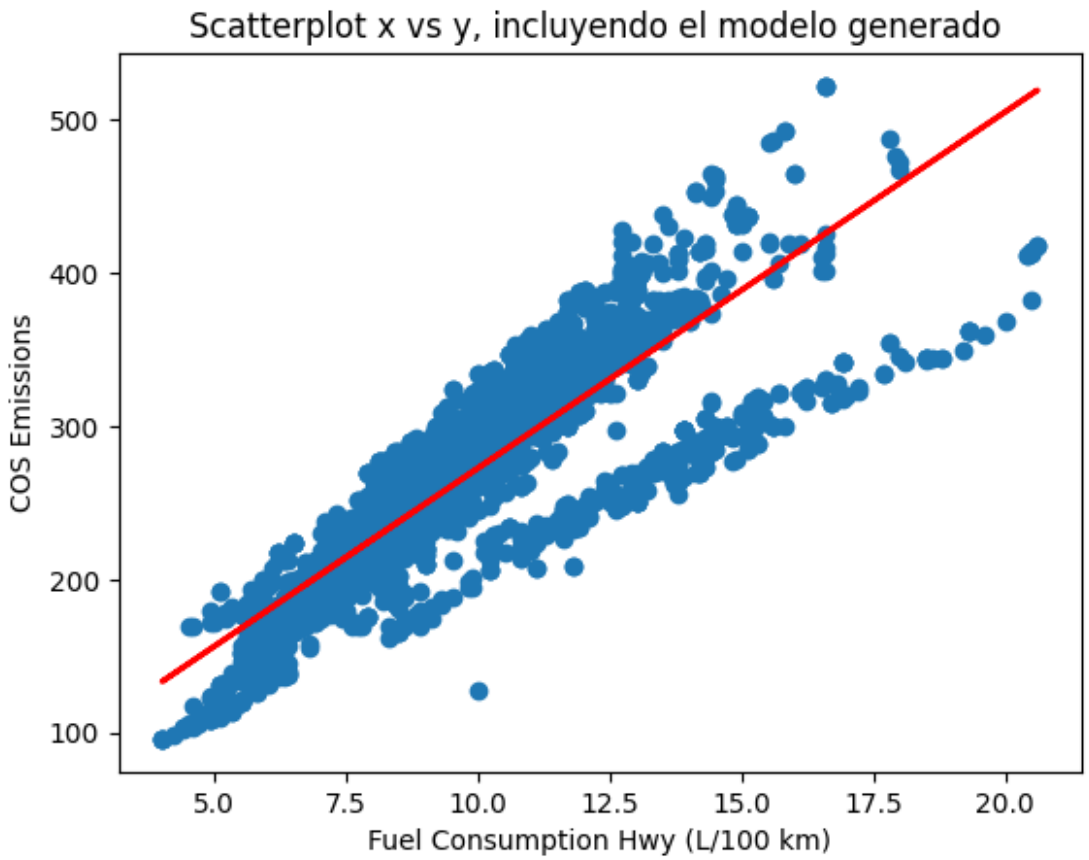
```
x=df['Fuel Consumption Hwy (L/100 km)']
y=df['CO2 Emissions(g/km)']
x = sm.add_constant(x)
model= sm.OLS(y,x)
result = model.fit()
print('Coeficiente de determinación: ', result.rsquared)
yh = result.predict(x)
plt.scatter(y, yh)
plt.xlabel('Valores Observados')
plt.ylabel('Valores Predichos')
plt.title('Comprensión de Precisión (Observados vs. Predichos)')
plt.show()
```

```
Coeficiente de determinación:  0.7806357669286315
```



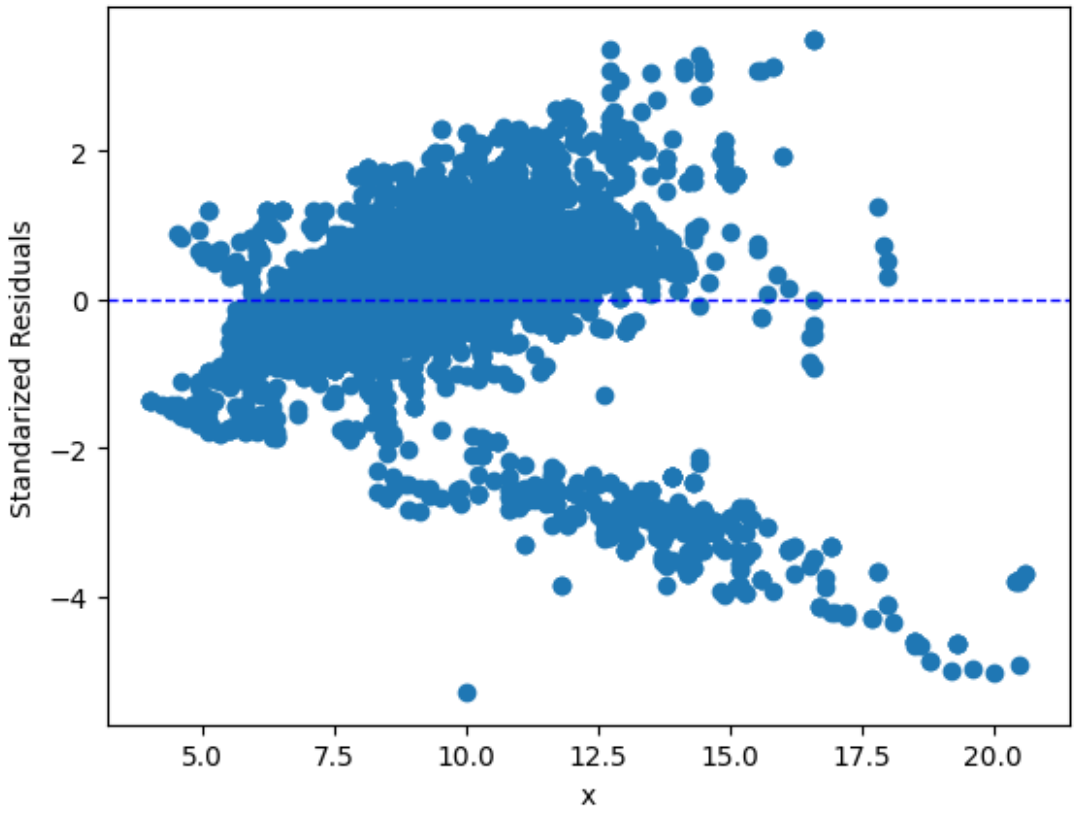
Se obtiene un R^2 del 78%

```
prediction = result.params['const'] + result.params['Fuel Consumption
Hwy (L/100 km)'] * df['Fuel Consumption Hwy (L/100 km)']
plt.scatter(df['Fuel Consumption Hwy (L/100 km)'], df['CO2
Emissions(g/km)'])
plt.plot(df['Fuel Consumption Hwy (L/100 km)'], prediction,
color='red', linewidth=2)
plt.xlabel('Fuel Consumption Hwy (L/100 km)')
plt.ylabel('COS Emissions')
plt.title('Scatterplot x vs y, incluyendo el modelo generado ')
plt.show()
```

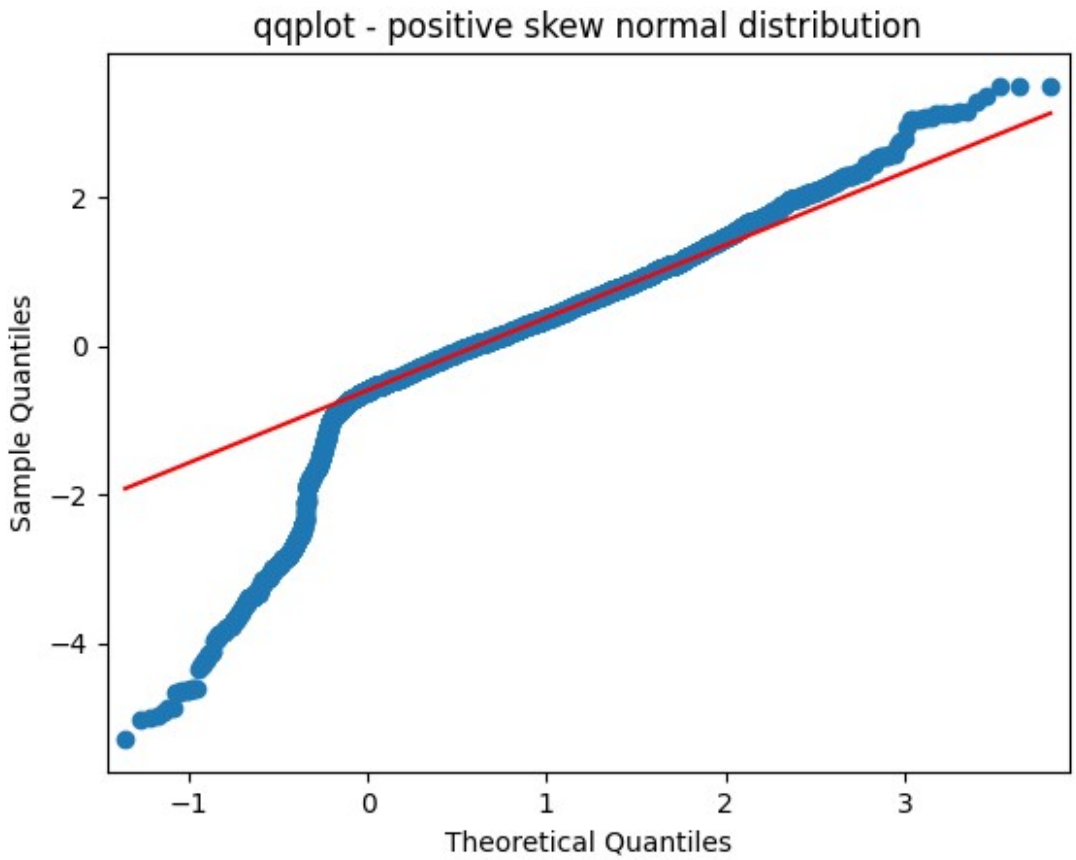



```
influence= result.get_influence()
standarized_residuals = influence.resid_studentized_internal
print(standarized_residuals)
plt.scatter(df['Fuel Consumption Hwy (L/100 km)'],
standarized_residuals)
plt.xlabel('x')
plt.ylabel('Standarized Residuals')
plt.axhline(y=0, color='b', linestyle='--', linewidth=1)
plt.show()
```

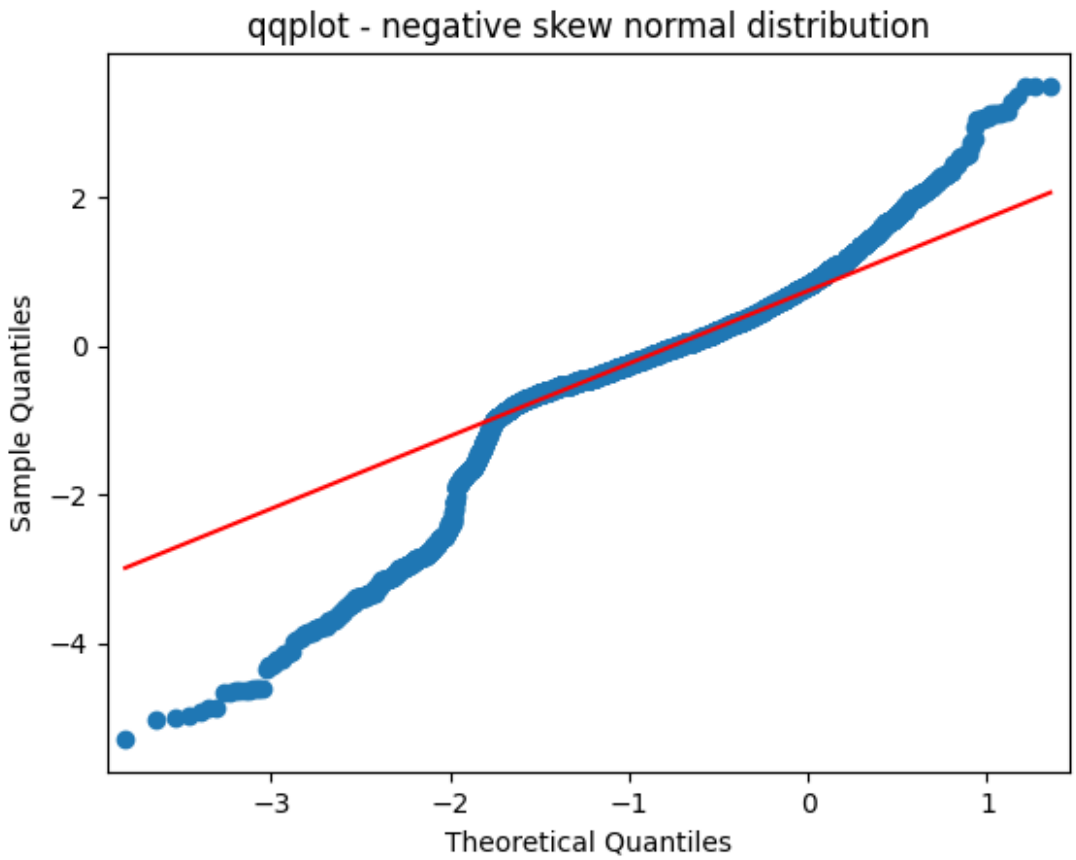
```
[-0.00589966  0.05829591 -1.43223324 ... -0.01164425 -0.04914741
  0.19546734]
```




```
fig = sm.qqplot(standarized_residuals, dist=skewnorm(2), line='q')
plt.title('qqplot - positive skew normal distribution')
plt.show()
```



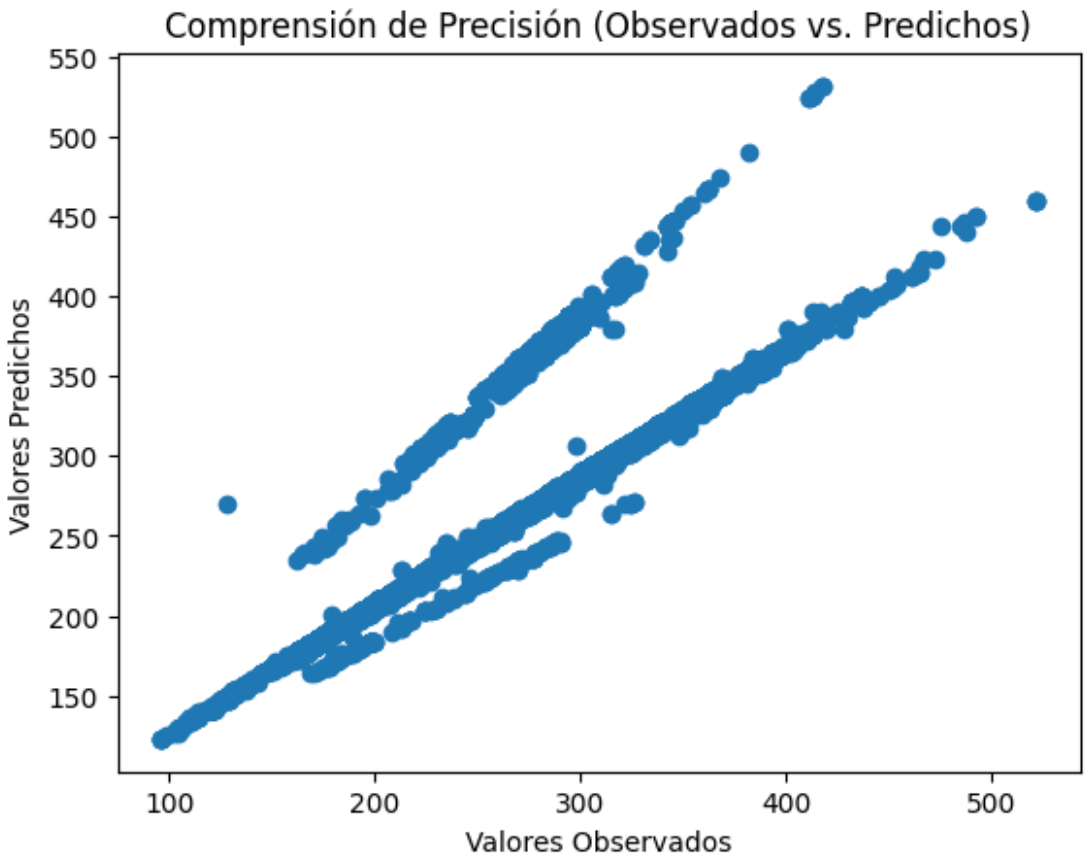

```
fig = sm.qqplot(standarized_residuals, dist=skewnorm(-2), line='q')
plt.title('qqplot - negative skew normal distribution')
plt.show()
```



Con la variable Fuel Consumption Comb (L/100 km):

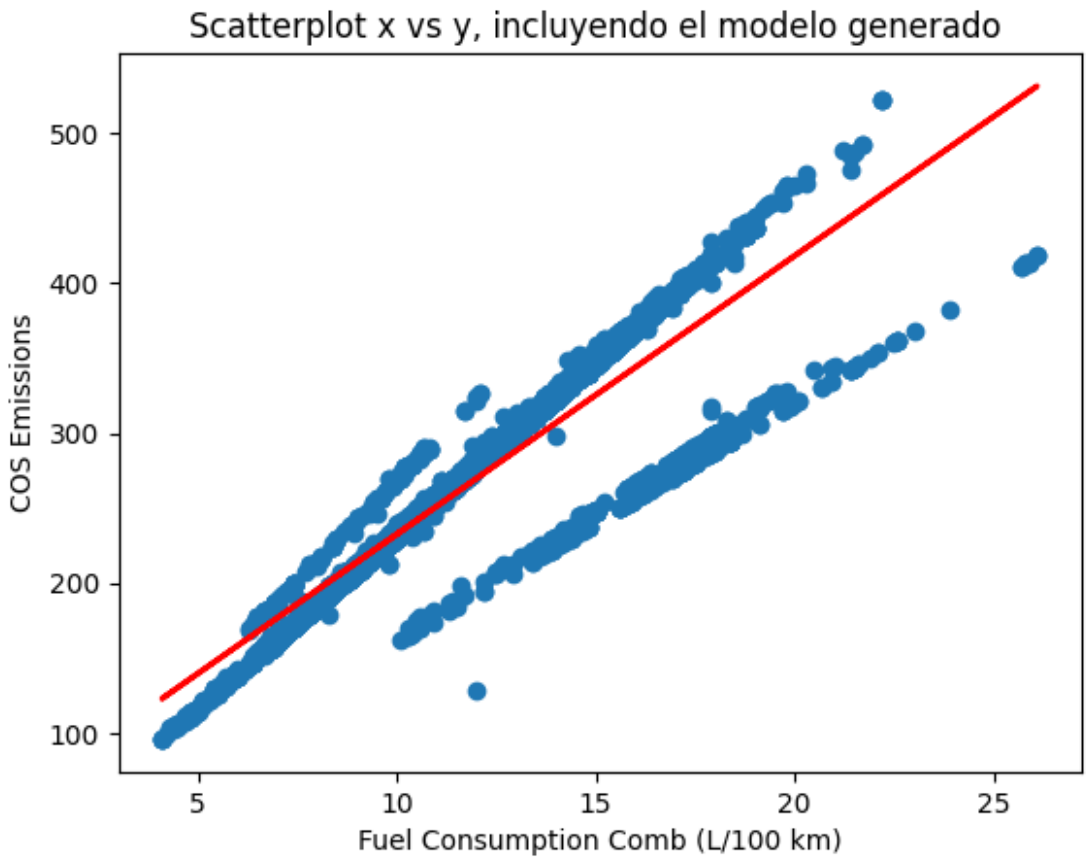
```
x=df['Fuel Consumption Comb (L/100 km)']
y=df['CO2 Emissions(g/km)']
x = sm.add_constant(x)
model= sm.OLS(y,x)
result = model.fit()
print('Coeficiente de determinación: ', result.rsquared)
yh = result.predict(x)
plt.scatter(y, yh)
plt.xlabel('Valores Observados')
plt.ylabel('Valores Predichos')
plt.title('Comprensión de Precisión (Observados vs. Predichos)')
plt.show()
```

```
Coeficiente de determinación:  0.8428186895623988
```



Se obtiene un R^2 del 84%

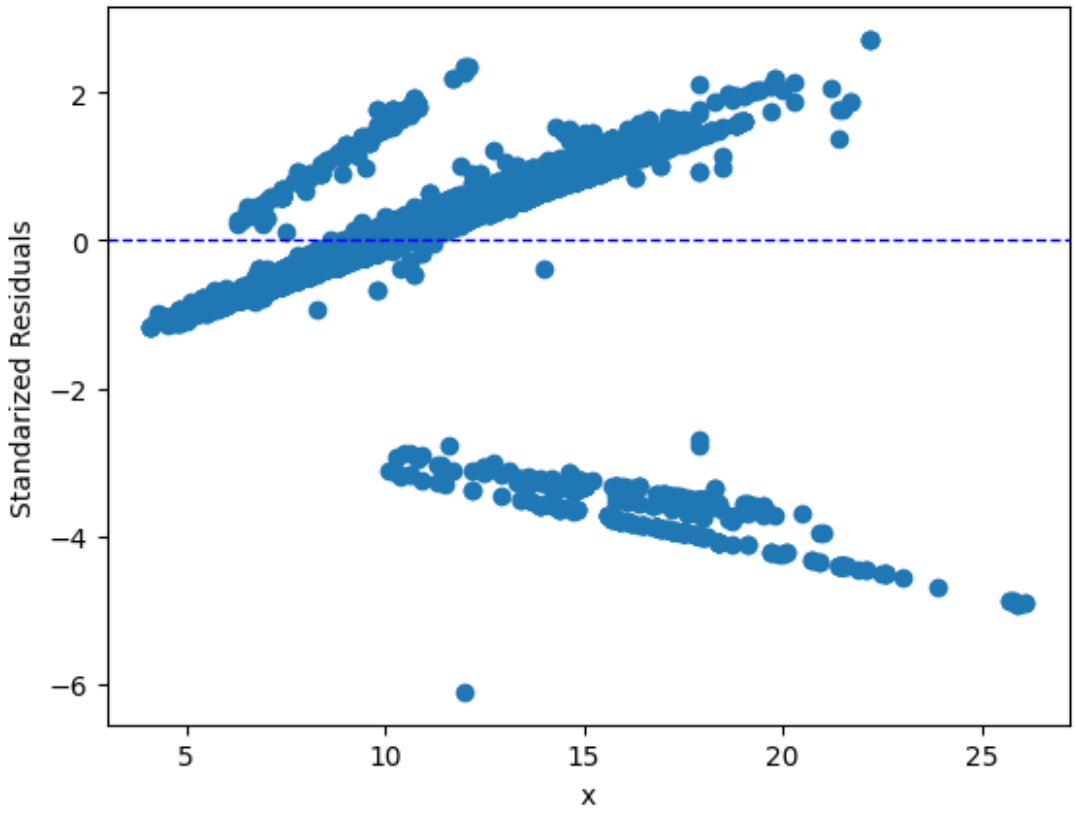
```
prediction = result.params['const'] + result.params['Fuel Consumption
Comb (L/100 km)'] * df['Fuel Consumption Comb (L/100 km)']
plt.scatter(df['Fuel Consumption Comb (L/100 km)'], df['CO2
Emissions(g/km)'])
plt.plot(df['Fuel Consumption Comb (L/100 km)'], prediction,
color='red', linewidth=2)
plt.xlabel('Fuel Consumption Comb (L/100 km)')
plt.ylabel('COS Emissions')
plt.title('Scatterplot x vs y, incluyendo el modelo generado ')
plt.show()
```



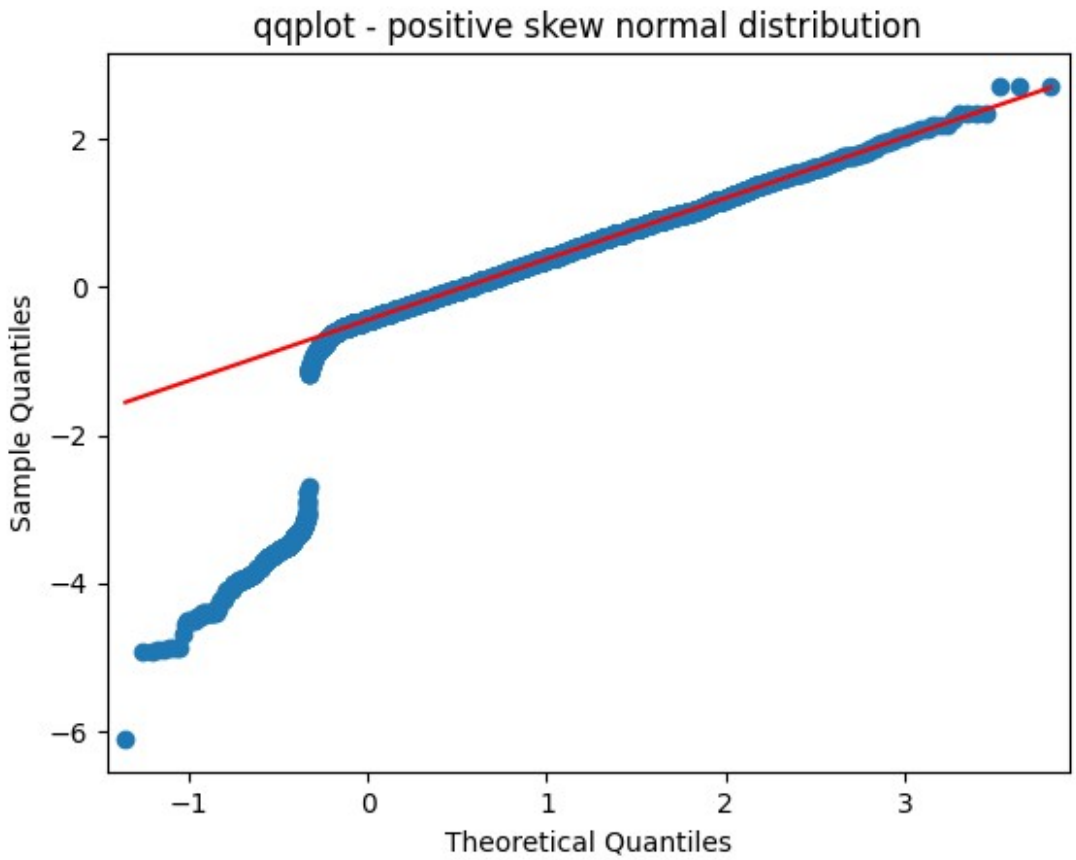
Se obtienen los errores estandarizados del modelo:

```
influence= result.get_influence()
standarized_residuals = influence.resid_studentized_internal
print(standarized_residuals)
plt.scatter(df['Fuel Consumption Comb (L/100 km)'],
standarized_residuals)
plt.xlabel('x')
plt.ylabel('Standarized Residuals')
plt.axhline(y=0, color='b', linestyle='--', linewidth=1)
plt.show()
```

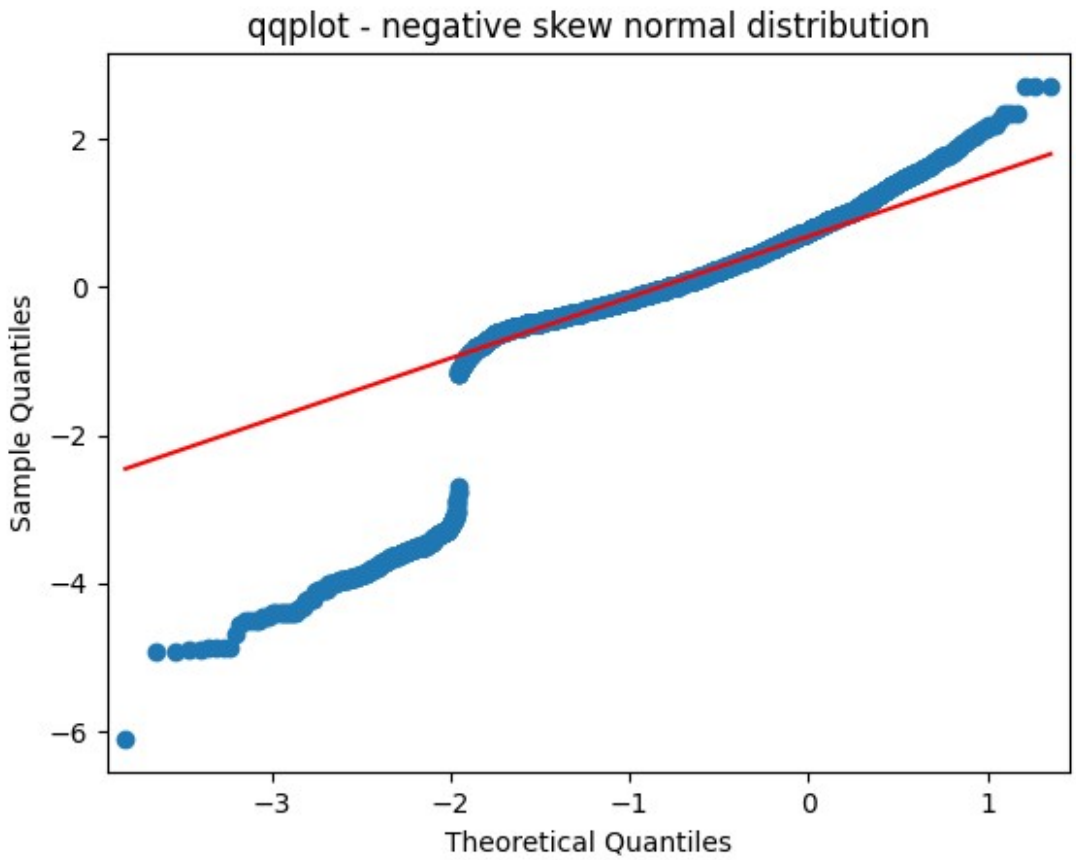
```
[-0.37157485 -0.17449253 -0.87672126 ...  0.0841568   0.05952249
  0.10879112]
```




```
fig = sm.qqplot(standarized_residuals, dist=skewnorm(2), line='q')
plt.title('qqplot - positive skew normal distribution')
plt.show()
```

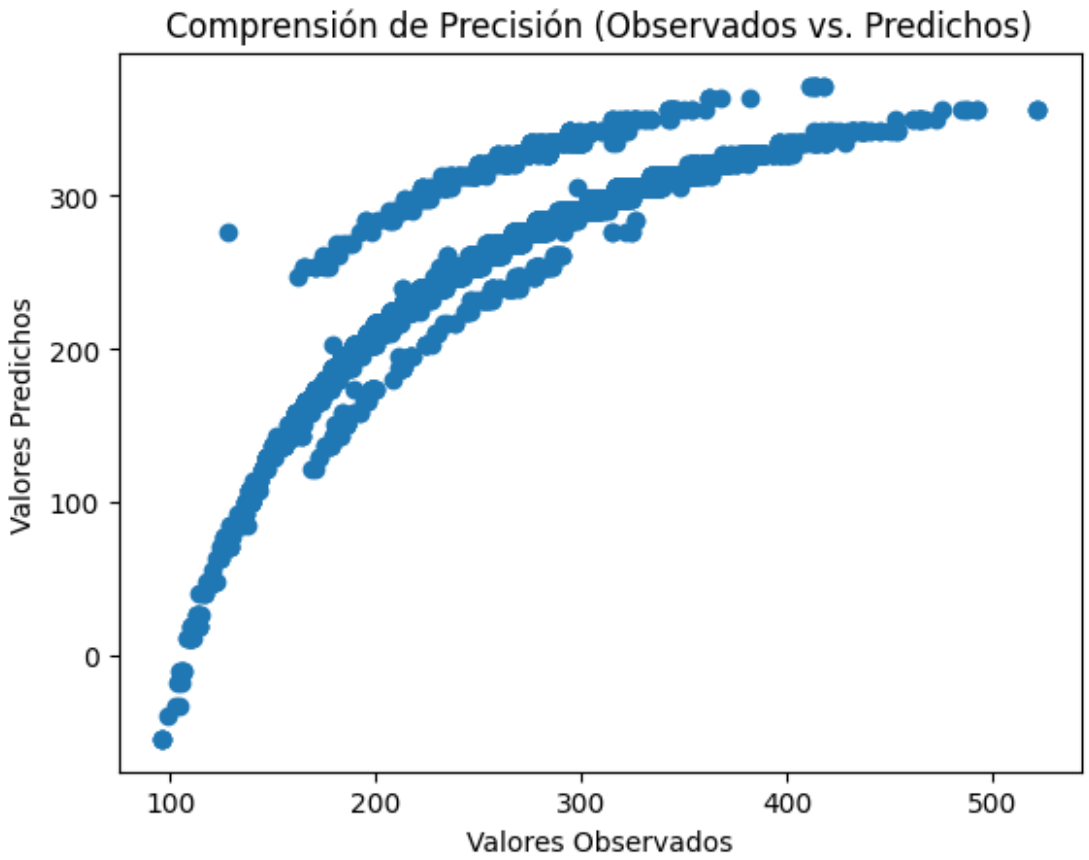



```
fig = sm.qqplot(standarized_residuals, dist=skewnorm(-2), line='q')
plt.title('qqplot - negative skew normal distribution')
plt.show()
```



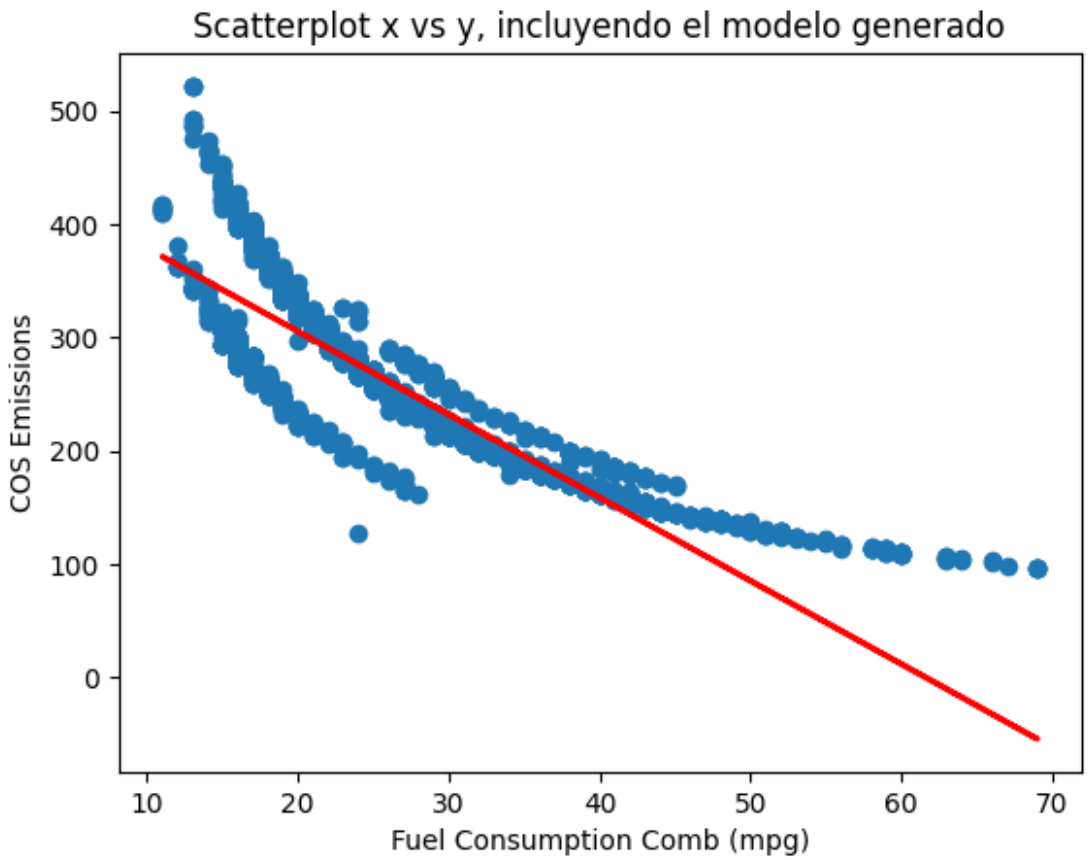
Con la variable Fuel Consumption Comb (mpg)

```
x=df['Fuel Consumption Comb (mpg)']
y=df['CO2 Emissions(g/km)']
x = sm.add_constant(x)
model= sm.OLS(y,x)
result = model.fit()
print('Coeficiente de determinación: ', result.rsquared)
yh = result.predict(x)
plt.scatter(y, yh)
plt.xlabel('Valores Observados')
plt.ylabel('Valores Predichos')
plt.title('Comprensión de Precisión (Observados vs. Predichos)')
plt.show()

Coeficiente de determinación:  0.8234224657110062
```



Se obtiene un R^2 del 82%

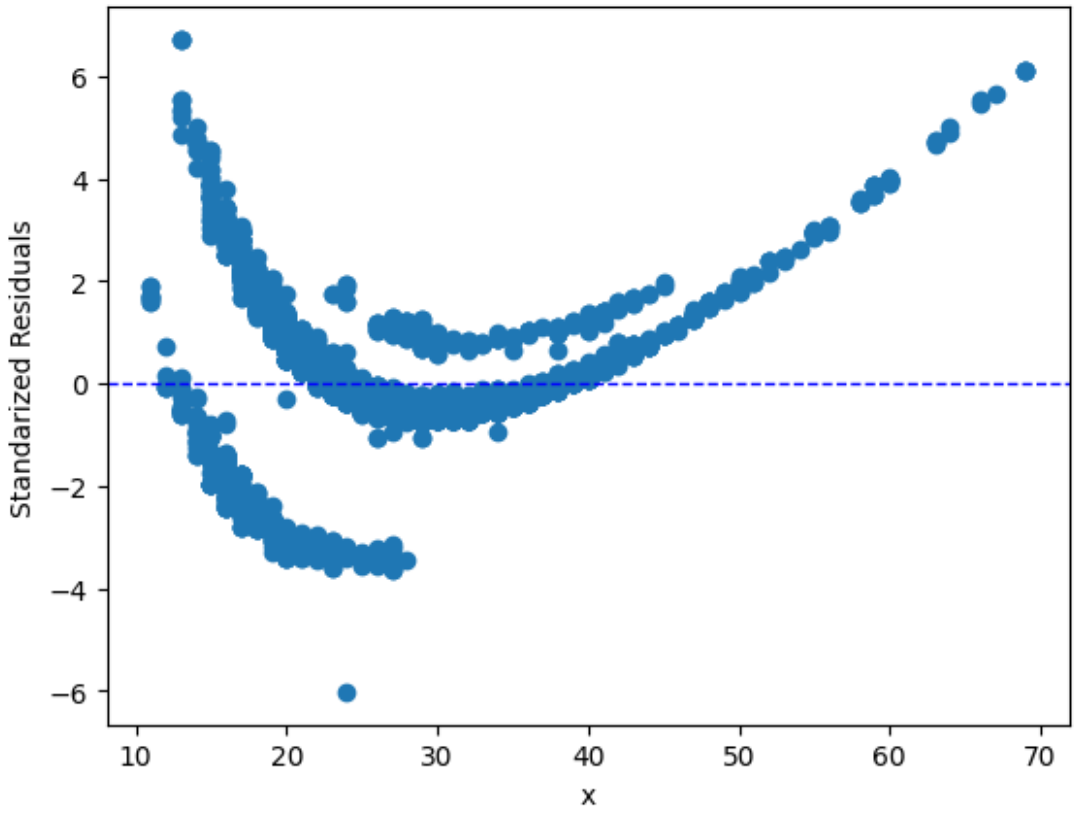
```
prediction = result.params['const'] + result.params['Fuel Consumption
Comb (mpg)'] * df['Fuel Consumption Comb (mpg)']
plt.scatter(df['Fuel Consumption Comb (mpg)'], df['CO2
Emissions(g/km)'])
plt.plot(df['Fuel Consumption Comb (mpg)'], prediction, color='red',
linewidth=2)
plt.xlabel('Fuel Consumption Comb (mpg)')
plt.ylabel('COS Emissions')
plt.title('Scatterplot x vs y, incluyendo el modelo generado')
plt.show()
```



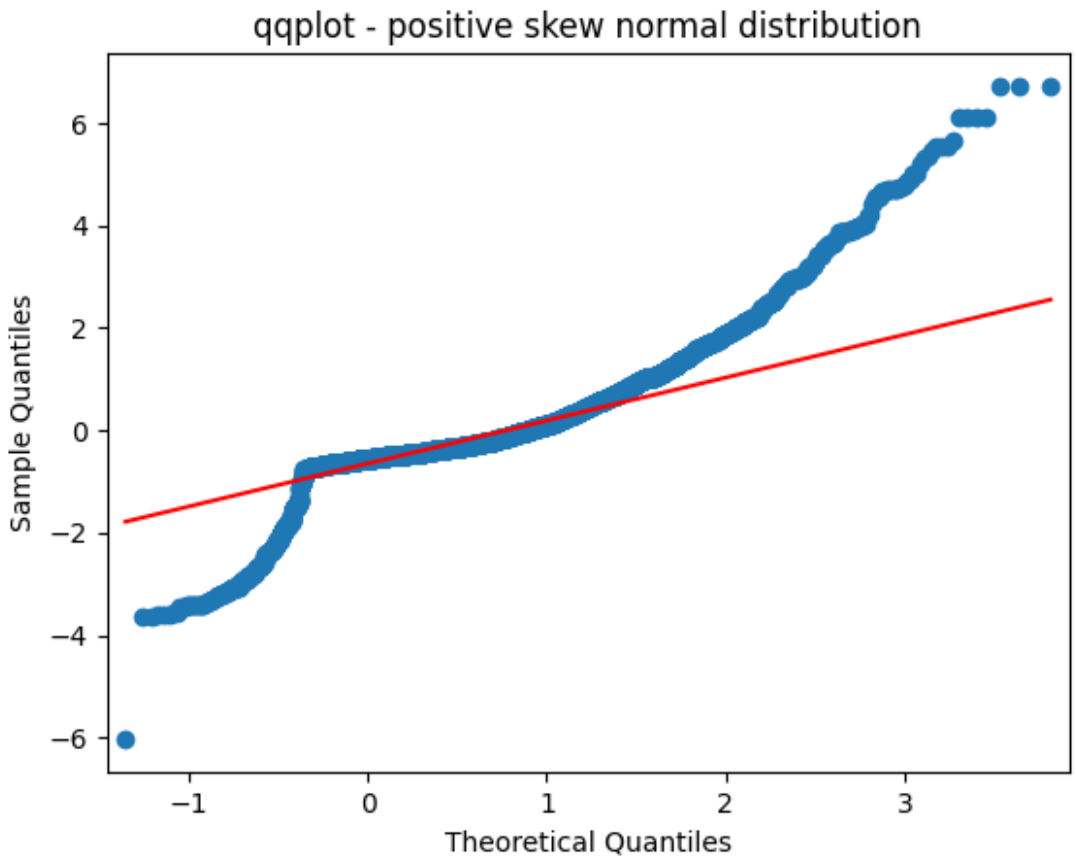
Se obtienen los errores estandarizados del modelo:

```
influence= result.get_influence()
standarized_residuals = influence.resid_studentized_internal
print(standarized_residuals)
plt.scatter(df['Fuel Consumption Comb (mpg)'], standarized_residuals)
plt.xlabel('x')
plt.ylabel('Standarized Residuals')
plt.axhline(y=0, color='b', linestyle='--', linewidth=1)
plt.show()
```

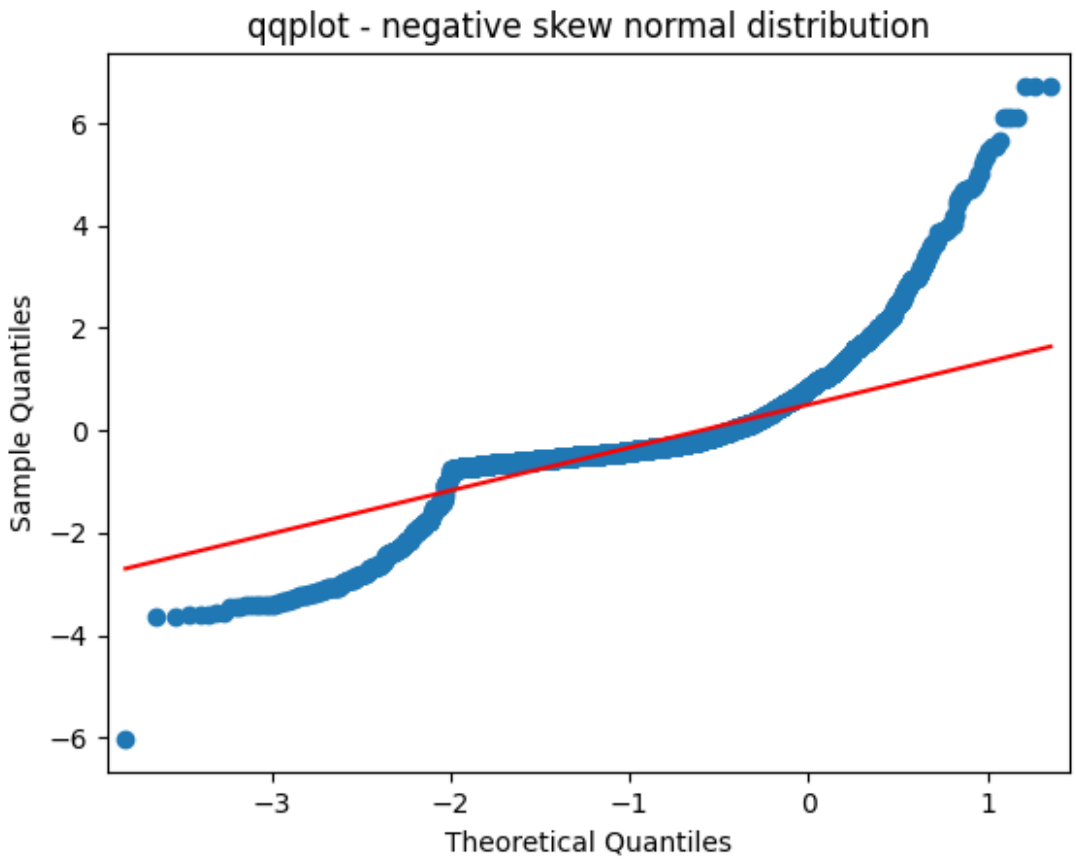
```
[-0.57223492 -0.74985292  1.46736968 ... -0.57431008 -0.30247323
 -0.54754722]
```




```
fig = sm.qqplot(standarized_residuals, dist=skewnorm(2), line='q')
plt.title('qqplot - positive skew normal distribution')
plt.show()
```




```
fig = sm.qqplot(standarized_residuals, dist=skewnorm(-2), line='q')
plt.title('qqplot - negative skew normal distribution')
plt.show()
```

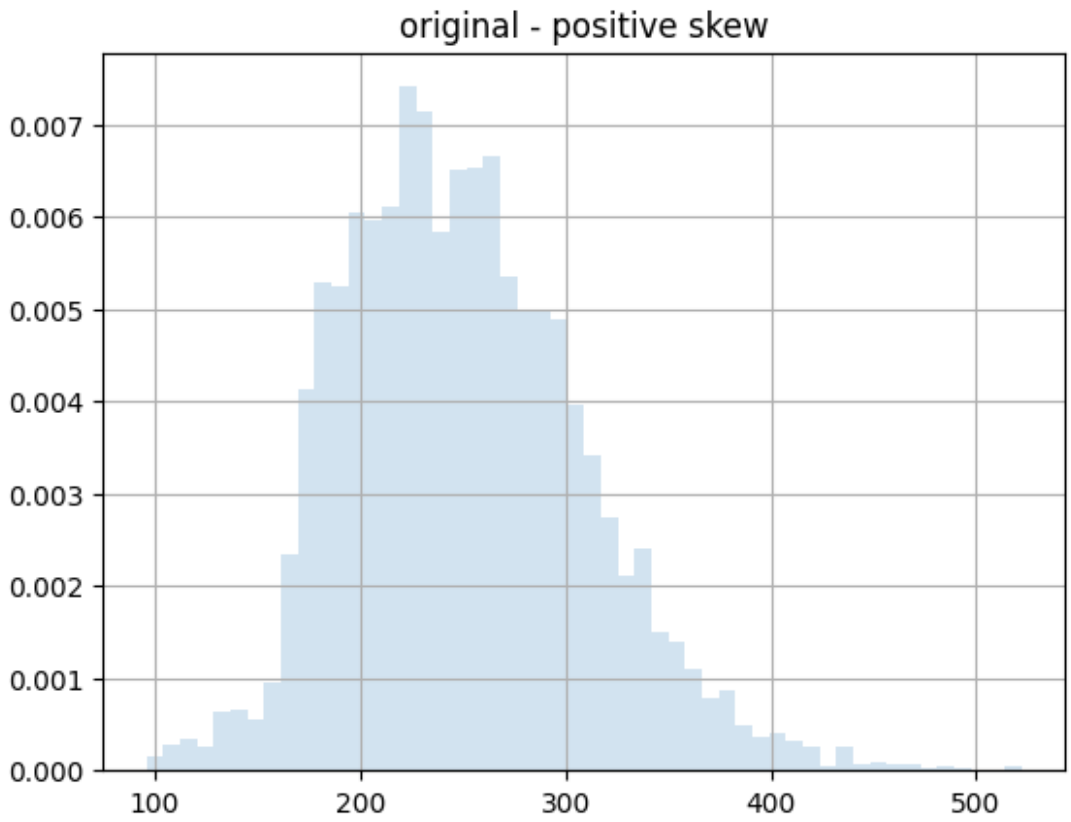


Transformacion de la variable de respuesta

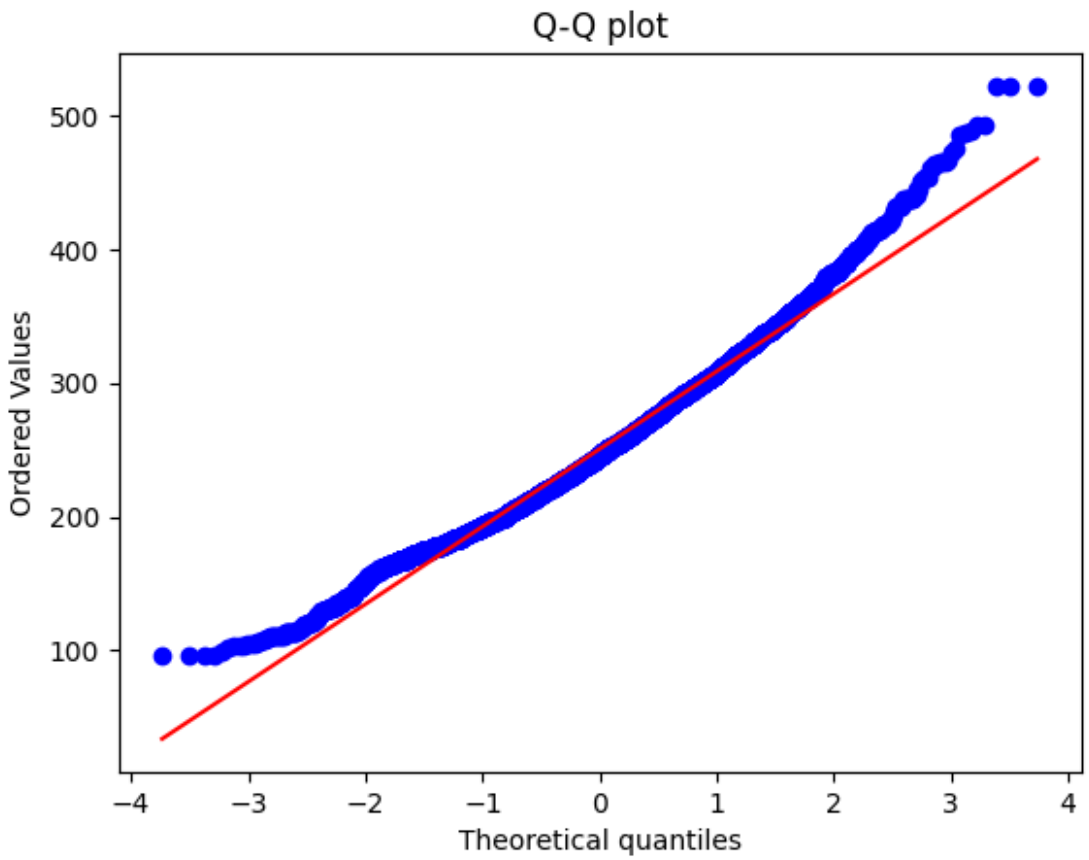
Ahora se procederá con una transformación de la variable de respuesta, utilizando algunas técnicas. A la transformación que mejor simetría demuestre tanto en el histograma como en el QQ-Plt, se utilizará para verificar si se mejoran las métricas

Primero se muestra la variable sin transformar:

```
r = df['CO2 Emissions(g/km)']
#r = skewnorm.rvs(4, size=10000)
plt.hist(r, density=True, bins='auto', histtype='stepfilled',
alpha=0.2)
plt.title('original - positive skew')
plt.grid()
plt.show()
```

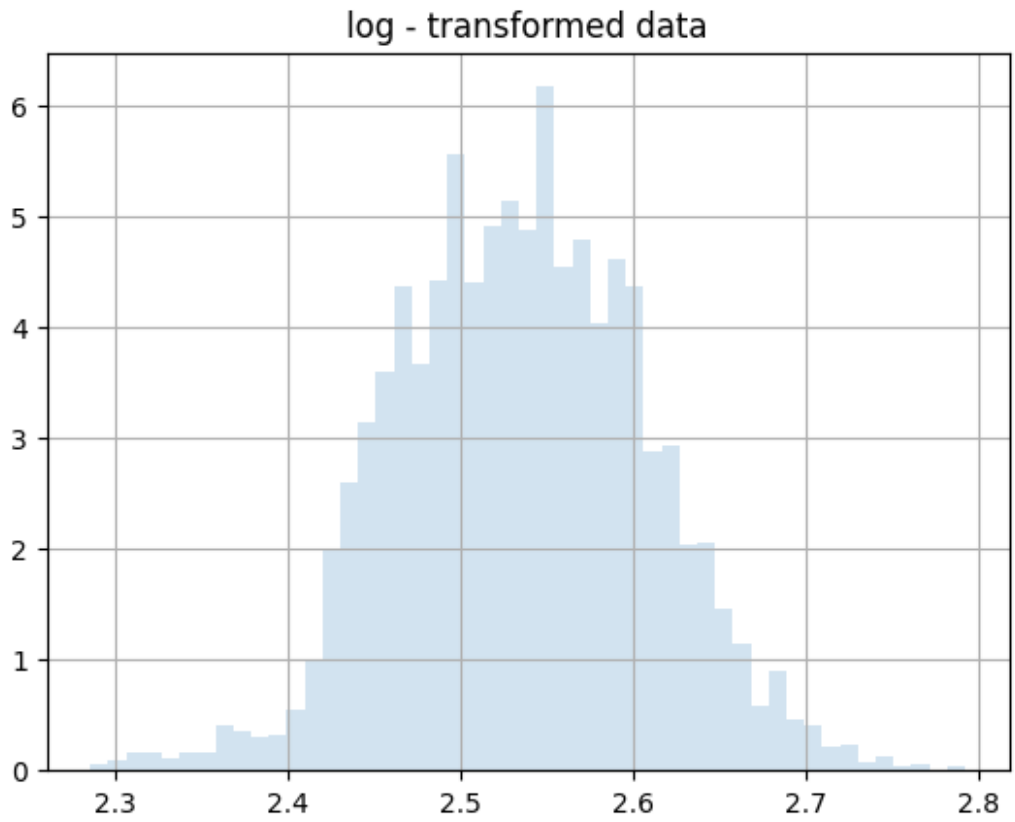



```
stats.probplot(r, plot=plt)
plt.title("Q-Q plot")
plt.show()
```

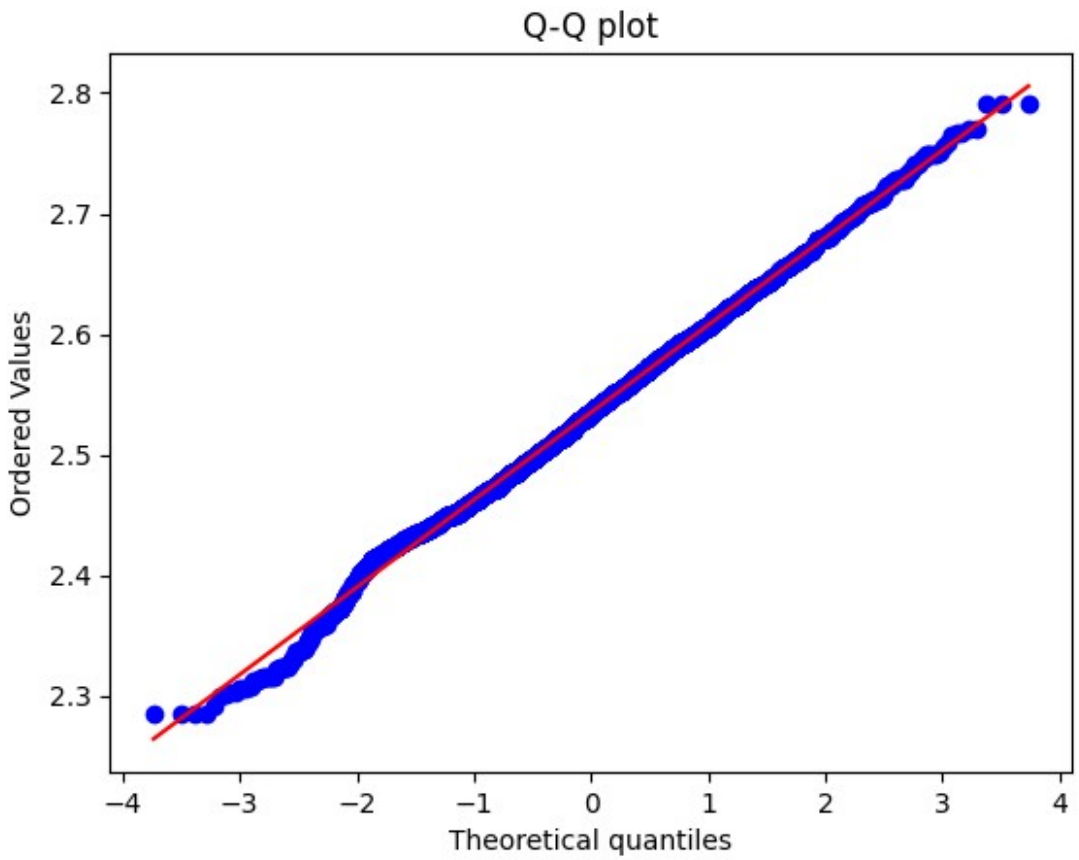


Se hace transformación con Logaritmo

```
r = df['CO2 Emissions(g/km)']
r_log = np.log10(1 + r + abs(min(r)))
plt.hist(r_log, density=True, bins='auto', histtype='stepfilled',
alpha=0.2)
plt.title('log - transformed data')
plt.grid()
plt.show()
```

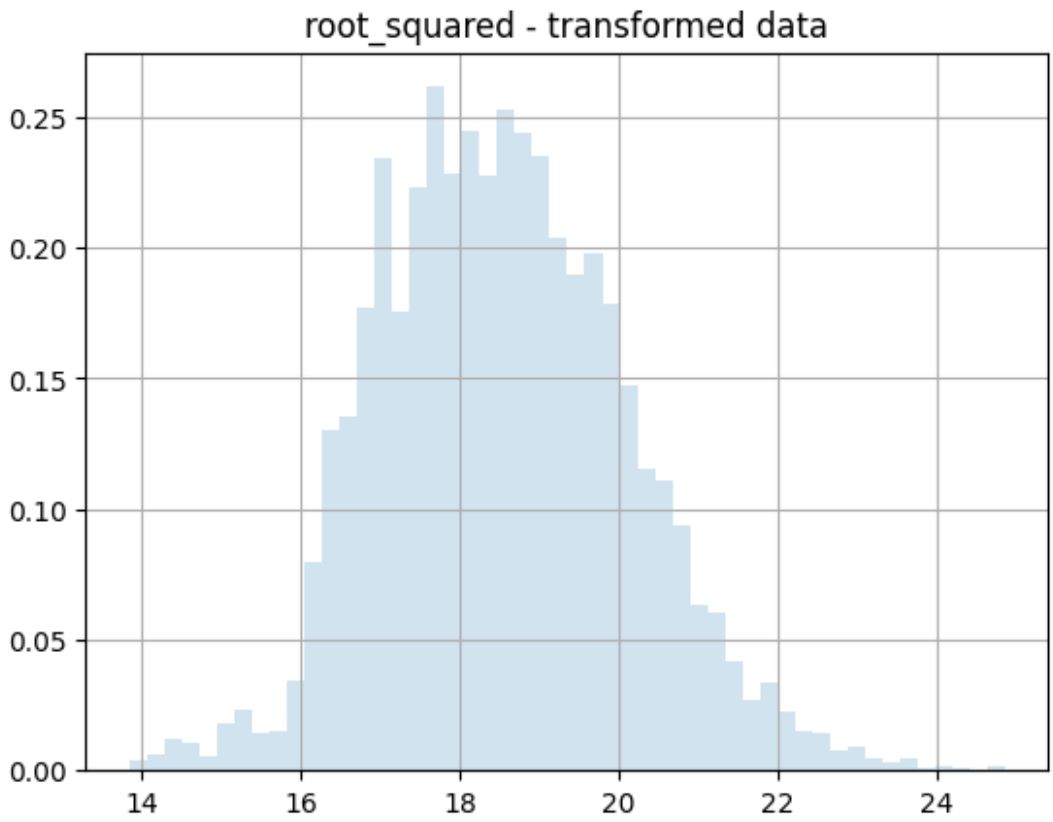



```
stats.probplot(r_log, plot=plt)
plt.title("Q-Q plot")
plt.show()
```

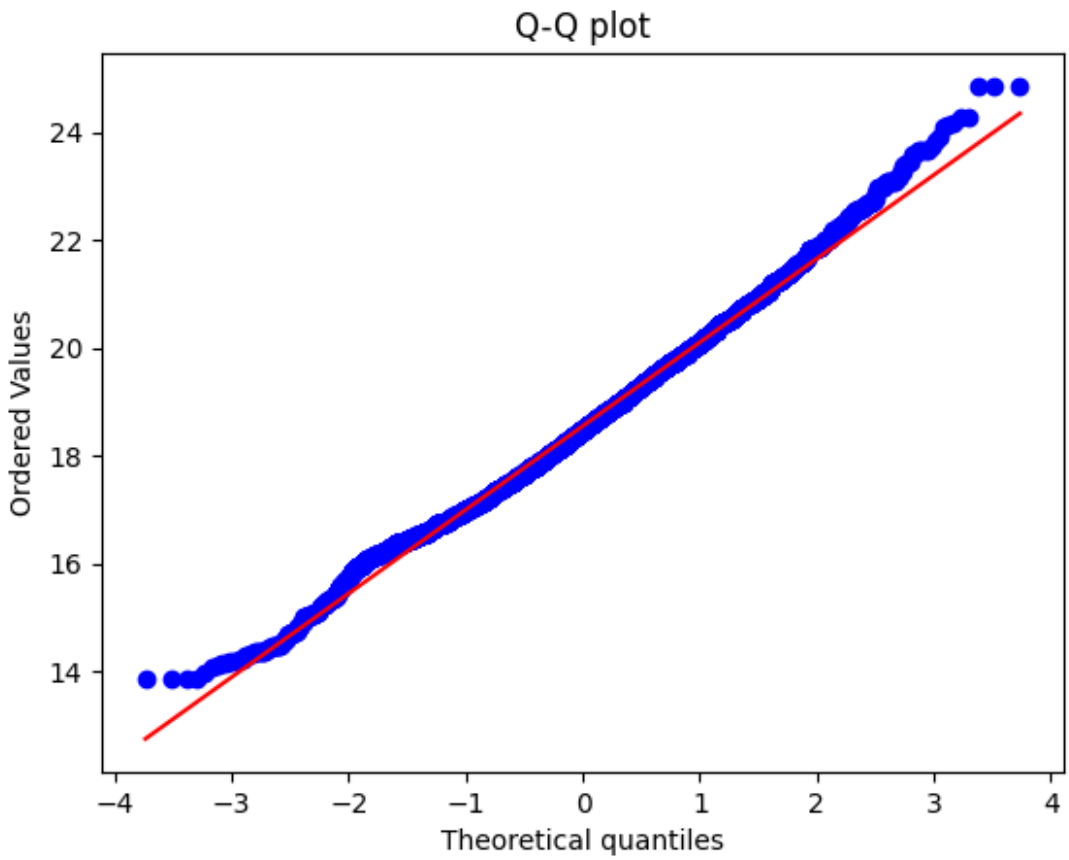


Se hace transformación con Raíz Cuadrada:

```
r_root =np.sqrt(r + abs(min(r)))
plt.hist(r_root, density=True, bins='auto', histtype='stepfilled',
alpha=0.2)
plt.title('root_squared - transformed data')
plt.grid()
plt.show()
```

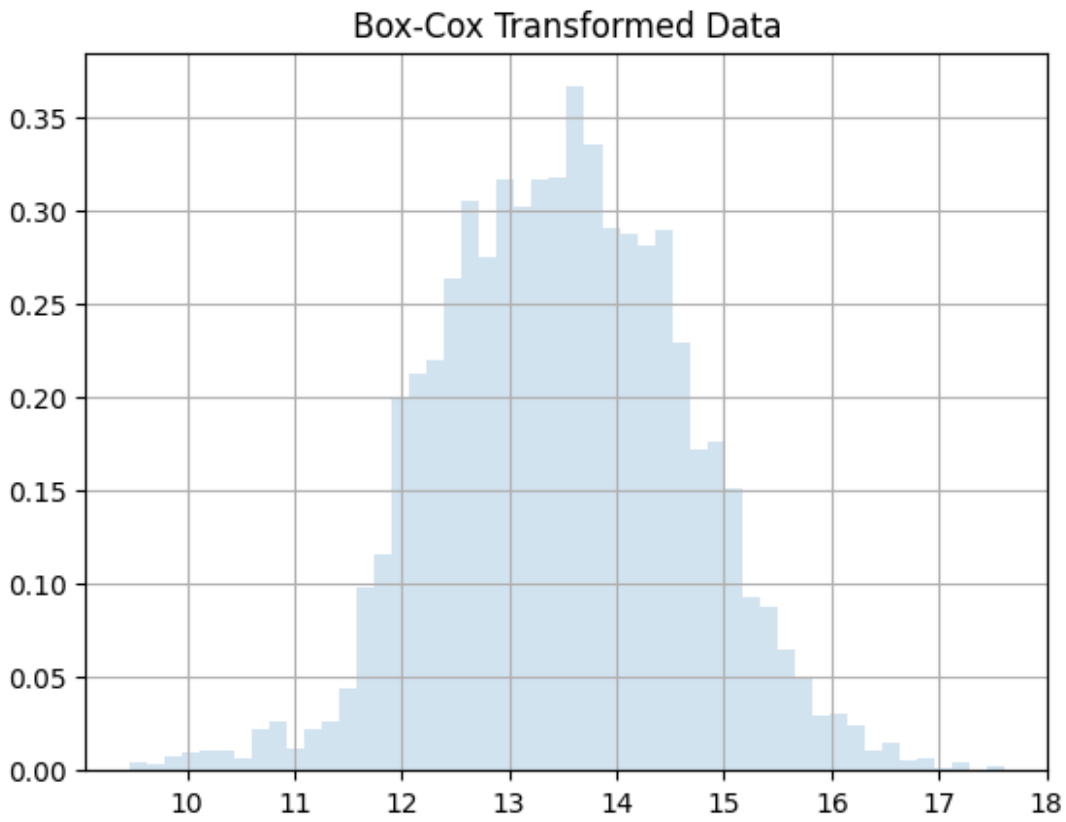



```
stats.probplot(r_root, plot=plt)
plt.title("Q-Q plot")
plt.show()
```

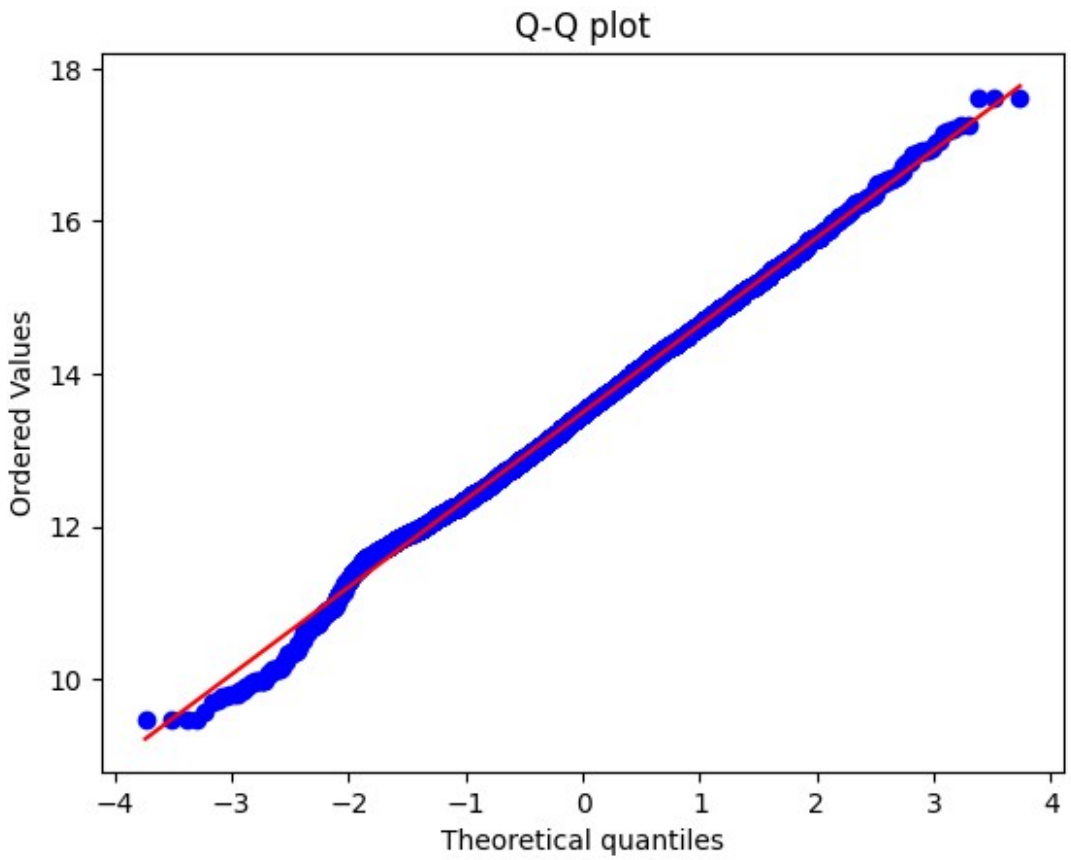


Se hace transformación con Boxcox:

```
r = df['CO2 Emissions(g/km)']
r_boxcox, _ = stats.boxcox(r)
plt.hist(r_boxcox, density=True, bins='auto', histtype='stepfilled',
alpha=0.2)
plt.title('Box-Cox Transformed Data')
plt.grid()
plt.show()
```




```
stats.probplot(r_boxcox, plot=plt)
plt.title("Q-Q plot")
plt.show()
```



Al realizar 3 transformaciones para la variable de respuesta, se obtiene que la más simétrica es la de Logaritmo, a pesar de que con la transformación Boxcox se obtiene también una gran simetría, se va a utilizar la de Logaritmo porque al probar la transformación Boxcox, se obtienen los mismos resultados que con la transformación Logaritmo.

Se realiza un modelo de regresión lineal simple con todas las variables, utilizando la variable de respuesta transformada por medio de Logaritmo, en este caso, se tiene en cuenta un prueba de hipótesis con una hipótesis nula $H0=\beta_i=0$, en caso de que el p-value sea menor a 0.05 (alpha), se rechaza la hipótesis nula.

Con la variable Engine Size:

```
y = r_log
x = df['Engine Size(L)']
x = sm.add_constant(x)
model = sm.OLS(y,x)
results = model.fit()
print('\n', results.params)
print(results.summary())
```

```
 const             2.393627
Engine Size(L)    0.044738
dtype: float64
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:     CO2 Emissions(g/km)   R-squared:                       0.698
Model:                             OLS   Adj. R-squared:                  0.698
Method:                  Least Squares   F-statistic:                 1.705e+04
Date:                 Wed, 04 Oct 2023   Prob (F-statistic):               0.00
Time:                         02:05:32   Log-Likelihood:                 13318.
No. Observations:                7385   AIC:                            -2.663e+04
Df Residuals:                    7383   BIC:                            -2.662e+04
Df Model:                           1
Covariance Type:             nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          2.3936      0.001   2032.055      0.000       2.391       2.396
Engine Size(L)     0.0447      0.000    130.574      0.000       0.044       0.045
==============================================================================
Omnibus:                      702.943   Durbin-Watson:                   0.931
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1547.625
Skew:                          -0.597   Prob(JB):                         0.00
Kurtosis:                       4.899   Cond. No.                         9.36
==============================================================================

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

Se obtiene un R^2 del 69%

Con la variable Cylinders:

```
y = r_log
x = df['Cylinders']
x = sm.add_constant(x)
model = sm.OLS(y,x)
results = model.fit()
print('\n', results.params)
print(results.summary())
```

```
 const        2.354077
Cylinders    0.032222
dtype: float64
                                OLS Regression Results
==============================================================================
Dep. Variable:     CO2 Emissions(g/km)   R-squared:                       0.660
Model:                             OLS   Adj. R-squared:                  0.660
Method:                  Least Squares   F-statistic:                 1.432e+04
Date:                 Wed, 04 Oct 2023   Prob (F-statistic):               0.00
Time:                         02:05:32   Log-Likelihood:                 12881.
No. Observations:                7385   AIC:                            -2.576e+04
Df Residuals:                    7383   BIC:                            -2.574e+04
Df Model:                           1
Covariance Type:             nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          2.3541      0.002   1480.592      0.000       2.351       2.357
Cylinders      0.0322      0.000    119.673      0.000       0.032       0.033
==============================================================================
Omnibus:                      522.677   Durbin-Watson:                   0.962
```

```
Prob(Omnibus):                    0.000   Jarque-Bera (JB):
1118.714
Skew:                            -0.468   Prob(JB):
1.19e-243
Kurtosis:                         4.662   Cond. No.
19.6
================================================================================
========

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

Se obtiene un R^2 del 66%

Con la variable Fuel Consumption City:

```
y = r_log
x = df['Fuel Consumption City (L/100 km)']
x = sm.add_constant(x)
model = sm.OLS(y,x)
results = model.fit()
print('\n', results.params)
print(results.summary())


 const                                 2.296990
Fuel Consumption City (L/100 km)    0.018955
dtype: float64
                            OLS Regression Results

================================================================================
=========
Dep. Variable:     CO2 Emissions(g/km)   R-squared:
0.837
Model:                             OLS   Adj. R-squared:
0.837
Method:                  Least Squares   F-statistic:
3.790e+04
Date:                 Wed, 04 Oct 2023   Prob (F-statistic):
0.00
Time:                         02:05:32   Log-Likelihood:
15596.
No. Observations:                7385   AIC:                            -
3.119e+04
Df Residuals:                    7383   BIC:                            -
3.117e+04
Df Model:                           1

Covariance Type:             nonrobust
```

```
==============================================================================
                                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const                          2.2970      0.001   1809.859      0.000       2.295       2.299
Fuel Consumption City (L/100 km)     0.0190   9.74e-05    194.686      0.000       0.019       0.019
==============================================================================
Omnibus:                     3003.067   Durbin-Watson:                   1.803
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            13379.800
Skew:                          -1.975   Prob(JB):                         0.00
Kurtosis:                       8.279   Cond. No.                         48.8
==============================================================================

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

Se obtiene un R^2 del 83%

Con la variable Fuel Consumption Hwy:

```
y = r_log
x = df['Fuel Consumption Hwy (L/100 km)']
x = sm.add_constant(x)
model = sm.OLS(y,x)
results = model.fit()
print('\n', results.params)
print(results.summary())
```

```
 const                          2.276667
Fuel Consumption Hwy (L/100 km)    0.028572
dtype: float64
                            OLS Regression Results

==============================================================================
Dep. Variable:     CO2 Emissions(g/km)   R-squared:                       0.768
```

```
Model:                            OLS   Adj. R-squared:
0.768
Method:                 Least Squares   F-statistic:
2.444e+04
Date:                Wed, 04 Oct 2023   Prob (F-statistic):
0.00
Time:                        02:05:32   Log-Likelihood:
14294.
No. Observations:                7385   AIC:                            -
2.858e+04
Df Residuals:                    7383   BIC:                            -
2.857e+04
Df Model:                           1

Covariance Type:            nonrobust

==============================================================================
                                 coef    std err          t
P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const                          2.2767      0.002   1337.843
0.000       2.273       2.280
Fuel Consumption Hwy (L/100 km)     0.0286      0.000    156.333
0.000       0.028       0.029
==============================================================================
Omnibus:                     2155.931   Durbin-Watson:
1.628
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
6560.497
Skew:                          -1.507   Prob(JB):
0.00
Kurtosis:                       6.498   Cond. No.
39.4
==============================================================================

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

Se obtiene un R^2 del 76%

Con la variable Fuel Consumption Comb:

```
y = r_log
x = df['Fuel Consumption Comb (L/100 km)']
x = sm.add_constant(x)
```

```
model = sm.OLS(y,x)
results = model.fit()
print('\n', results.params)
print(results.summary())
```

```
 const                                 2.283923
Fuel Consumption Comb (L/100 km)    0.022877
dtype: float64
                                OLS Regression Results

==============================================================================
Dep. Variable:     CO2 Emissions(g/km)   R-squared:
0.833
Model:                             OLS   Adj. R-squared:
0.833
Method:                  Least Squares   F-statistic:
3.670e+04
Date:                 Wed, 04 Oct 2023   Prob (F-statistic):
0.00
Time:                         02:05:33   Log-Likelihood:
15497.
No. Observations:                7385   AIC:                            -
3.099e+04
Df Residuals:                    7383   BIC:                            -
3.098e+04
Df Model:                           1

Covariance Type:              nonrobust

================================================================================
==============================
                                   coef    std err          t
P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
------------------------------
const                              2.2839      0.001   1685.139
0.000       2.281       2.287
Fuel Consumption Comb (L/100 km)     0.0229      0.000    191.578
0.000       0.023       0.023
==============================================================================
========
Omnibus:                     3499.957   Durbin-Watson:
1.865
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
18517.033
Skew:                          -2.292   Prob(JB):
0.00
Kurtosis:                       9.259   Cond. No.
```

```
44.9
==============================================================================
========

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

Se obtiene un R^2 del 83%

Con la variable Fuel Consumption Comb (mpg):

```
y = r_log
x = df['Fuel Consumption Comb (mpg)']
x = sm.add_constant(x)
model = sm.OLS(y,x)
results = model.fit()
print('\n', results.params)
print(results.summary())


 const                          2.793990
Fuel Consumption Comb (mpg)   -0.009424
dtype: float64
                            OLS Regression Results

==============================================================================
==========
Dep. Variable:     CO2 Emissions(g/km)   R-squared:
0.883
Model:                             OLS   Adj. R-squared:
0.883
Method:                  Least Squares   F-statistic:
5.578e+04
Date:                 Wed, 04 Oct 2023   Prob (F-statistic):
0.00
Time:                         02:05:33   Log-Likelihood:
16825.
No. Observations:                7385   AIC:                            -
3.365e+04
Df Residuals:                    7383   BIC:                            -
3.363e+04
Df Model:                           1

Covariance Type:             nonrobust

==============================================================================
=========================
                                 coef    std err          t      P>|
t|      [0.025      0.975]
```

```
------------------------------------------------------------------------------
-------------------------
const                             2.7940      0.001   2464.076
0.000       2.792       2.796
Fuel Consumption Comb (mpg)    -0.0094   3.99e-05   -236.181
0.000      -0.010      -0.009
==============================================================================
========
Omnibus:                      955.846   Durbin-Watson:
1.517
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
10240.837
Skew:                          -0.201   Prob(JB):
0.00
Kurtosis:                       8.755   Cond. No.
112.
==============================================================================
========

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

Se obtiene un R^2 del 88%

Al realizar modelos de regresión lineal simple con cada variable predictora y la variable de respuesta transformada, se puede observar que los coefientes de determinación son muy similares cuando se hacen estas regresiones con la variable de respuesta sin transformar. También, de acuerdo con la prueba de hipótesis, no hubo necesidad de eliminar algun Beta, debido a que ninguno fue mayor a alpha (0.05)

Regresion Lineal Múltiple con variable de respuesta transformada:

```
x = df.drop(['CO2 Emissions(g/km)', 'Make', 'Model', 'Vehicle Class',
'Transmission', 'Fuel Type'], axis=1)
#y = x['r_log']

x['r_log'] = r_log
y=x['r_log']

x.columns

Index(['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)', 'r_log'],
      dtype='object')

nuevos = {'Fuel Consumption City (L/100 km)': 'Fuel_Consumption_City',
'Fuel Consumption Hwy (L/100 km)': 'Fuel_Consumption_Hwy', 'Fuel
```

```
Consumption Comb (L/100 km)': 'Fuel_Consumption_Comb', 'Fuel
Consumption Comb (mpg)': 'Fuel_Consumption_Comb_m', 'CO2
Emissions(g/km)' : 'CO2_Emissions'}

x = x.rename(columns=nuevos)
```

Se hace un conjunto de entrenamiento del 80% y uno de prueba del 20% de los datos:

```
from sklearn.model_selection import train_test_split
entrenamiento, prueba =
train_test_split(x,test_size=0.20,random_state=42)
```

Para el modelo de regresión lineal múltiple, se tiene una hipótesis nula $H0=\beta_i=0$, en caso de que el p-value sea menor a 0.05 (alpha), se rechaza la hipótesis nula.

Para el modelo de regresión, se irán eliminando variables predictoras que tengan un p-value mayor a alpha, para al final, generar el mejor modelo posible.

```
modelo = smf.ols(formula = 'r_log ~
Cylinders+Fuel_Consumption_City+Fuel_Consumption_Hwy+Fuel_Consumption_
Comb+Fuel_Consumption_Comb_m', data=entrenamiento)
modelo = modelo.fit()
print(modelo.summary())

                            OLS Regression Results

==============================================================================
========
Dep. Variable:                   r_log   R-squared:
0.924
Model:                             OLS   Adj. R-squared:
0.924
Method:                  Least Squares   F-statistic:
1.438e+04
Date:                 Wed, 04 Oct 2023   Prob (F-statistic):
0.00
Time:                         02:10:34   Log-Likelihood:
14750.
No. Observations:                5908   AIC:                             -
2.949e+04
Df Residuals:                    5902   BIC:                             -
2.945e+04
Df Model:                           5

Covariance Type:            nonrobust

==============================================================================
=====================
                               coef    std err          t      P>|t|
```

```
[0.025      0.975]
------------------------------------------------------------------------------
---------------------
Intercept                     2.6440      0.005    511.846      0.000
2.634       2.654
Cylinders                     0.0111      0.000     45.204      0.000
0.011       0.012
Fuel_Consumption_City         0.0006      0.003      0.191      0.849
-0.006       0.007
Fuel_Consumption_Hwy          0.0057      0.003      2.052      0.040
0.000       0.011
Fuel_Consumption_Comb        -0.0036      0.006     -0.598      0.550
-0.016       0.008
Fuel_Consumption_Comb_m      -0.0069   9.72e-05    -71.295      0.000
-0.007      -0.007
==============================================================================
========
Omnibus:                     1430.766   Durbin-Watson:
1.982
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
9315.473
Skew:                          -0.997   Prob(JB):
0.00
Kurtosis:                       8.820   Cond. No.
978.
==============================================================================
========

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

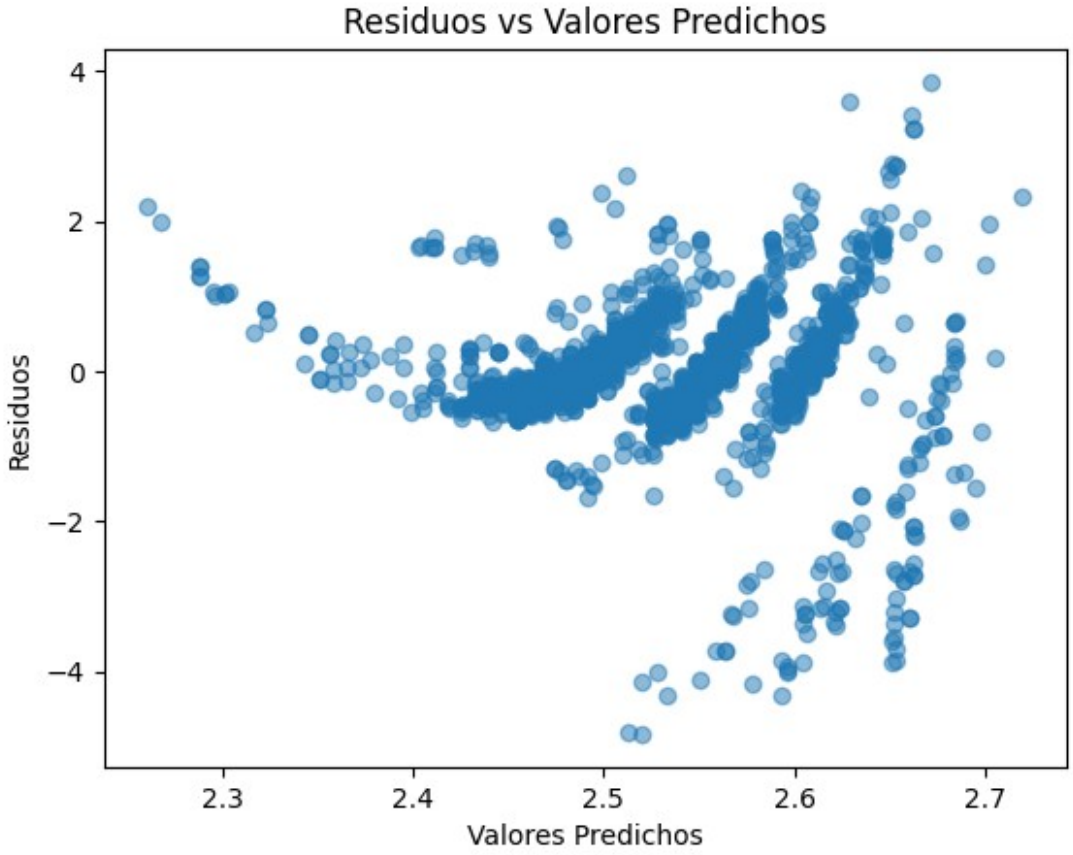
Se eliminan variables no relevantes de acuerdo con la prueba de hipótesis:

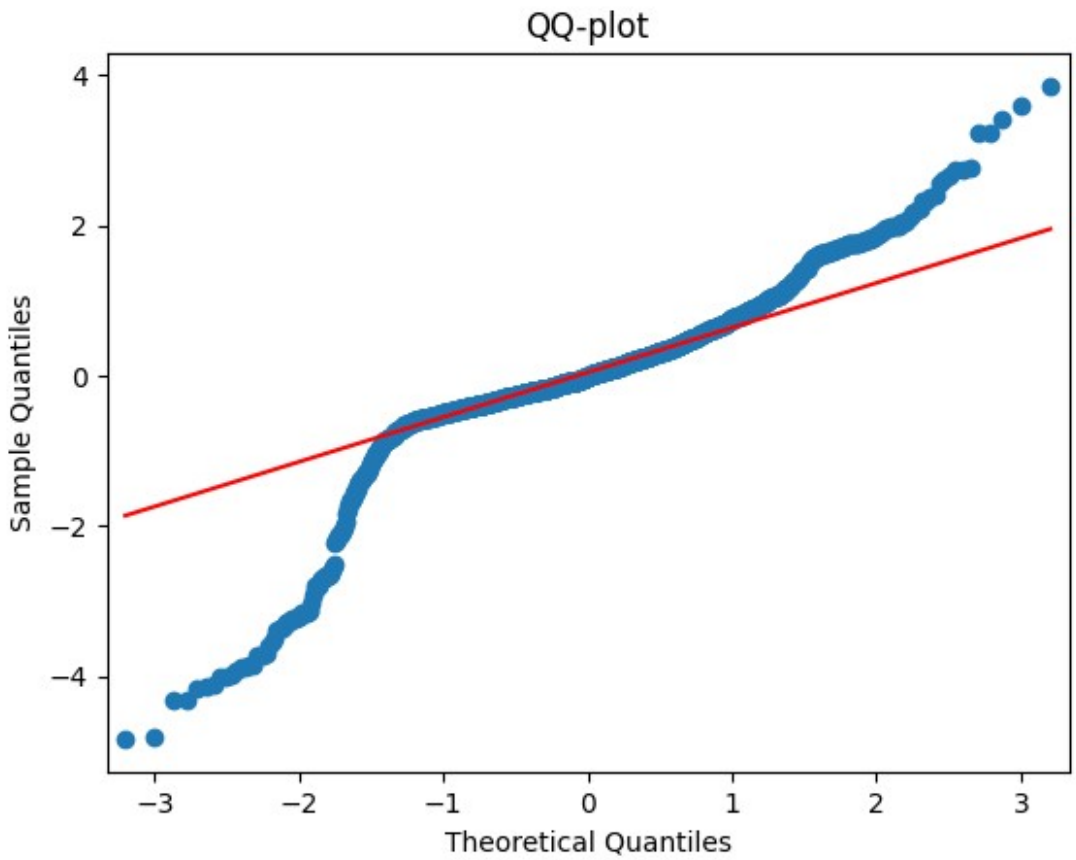
```
modelo = smf.ols(formula = 'r_log ~
Cylinders+Fuel_Consumption_Hwy+Fuel_Consumption_Comb_m',
data=entrenamiento)
modelo = modelo.fit()
print(modelo.summary())

                            OLS Regression Results

==============================================================================
========
Dep. Variable:                    r_log   R-squared:
0.924
Model:                            OLS   Adj. R-squared:
0.924
Method:                 Least Squares   F-statistic:
```

```
2.391e+04
Date:                Wed, 04 Oct 2023   Prob (F-statistic):
0.00
Time:                        02:05:33   Log-Likelihood:
14742.
No. Observations:                5908   AIC:                            -
2.948e+04
Df Residuals:                    5904   BIC:                            -
2.945e+04
Df Model:                           3

Covariance Type:            nonrobust

==============================================================================
                                coef    std err          t      P>|t|
[0.025      0.975]
------------------------------------------------------------------------------
Intercept                    2.6340      0.005    574.810      0.000
2.625       2.643
Cylinders                    0.0106      0.000     50.207      0.000
0.010       0.011
Fuel_Consumption_Hwy         0.0029      0.000     10.975      0.000
0.002       0.003
Fuel_Consumption_Comb_m     -0.0067   8.27e-05    -81.287      0.000
-0.007      -0.007
==============================================================================
Omnibus:                     1571.725   Durbin-Watson:
1.982
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
10070.390
Skew:                          -1.115   Prob(JB):
0.00
Kurtosis:                       8.995   Cond. No.
531.
==============================================================================

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

```
# se genera la prediccion del modelo
y_aprox = 2.6340 + 0.0106*prueba['Cylinders'] +
0.0029*prueba['Fuel_Consumption_Hwy'] -
0.0067*prueba['Fuel_Consumption_Comb_m']
```

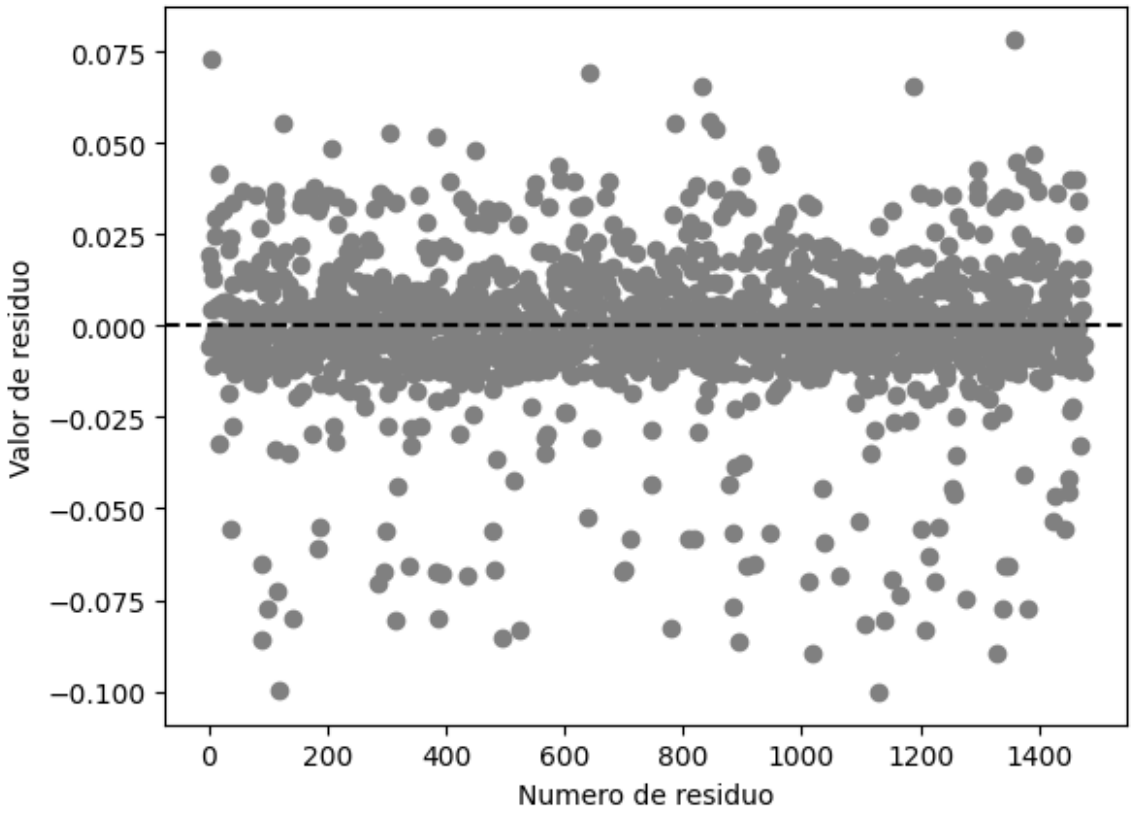
```
tabla=
pd.DataFrame({'Real':prueba['r_log'],'Prediccion':y_aprox,'Errores':pr
ueba['r_log']-y_aprox})

# Gráfica de residuos std vs valores predichos
media= tabla['Errores'].mean()
std=tabla['Errores'].std()
errores=(tabla['Errores']-media)/std

plt.scatter(y_aprox, errores, alpha=0.5)
plt.title("Residuos vs Valores Predichos")
plt.xlabel("Valores Predichos")
plt.ylabel("Residuos")
plt.show()

#QQplot de los errores
sm.qqplot(errores, line='q')
plt.title('QQ-plot')

plt.show()
```






```
residuos=len(tabla['Errores'])
plt.scatter(range(residuos),tabla['Errores'],color='gray')
plt.axhline(y=0,linestyle='--',color='black')
plt.xlabel('Numero de residuo')
plt.ylabel('Valor de residuo')
```

```
Text(0, 0.5, 'Valor de residuo')
```



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

Al obtener el modelo de regresión lineal múltiple con la variable de respuesta transformada, se obtiene un R^2 muy bueno, del 92%, esto puede deberse a que primero, la variable de respuesta está transformada y a que al combinar todas las variables del dataset se tiene un modelo más preciso y completo. Cuando se hacen modelos de regresión lineal simple, se obtienen R^2 inferiores, por lo que se puede concluir que hacer regresión lineal múltiple es mejor para predecir la variable de respuesta.

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

Se observa que los errores estandarizados vs los valores predichos siguen diferentes patrones al graficar los valores predichos y sus respectivos errores, en el QQ-plot se puede observar que incluso podría haber asimetría en los datos. Finalmente se hizo una gráfica de los errores, los cuales son pequeños, no se observan tendencias y solo se observan unos pocos datos que podrían considerarse como atípicos.

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

Al generar el modelo de regresión lineal múltiple final, se tienen 3 variables relevantes y un beta0. En caso contrario , se obtuvieron 2 variables (Fuel Consumption City y Fuel Consumption Comb) con nulo impacto o poca relevancia de acuerdo a la prueba de hipótesis formulada, las cuales se eliminaron del modelo y de hecho, el R^2 seguía siendo el mismo (92%). Esto confirma

la utilidad de la prueba de hipótesis al ayudar para eliminar variables no relevantes para el modelo de regresión.