Act 1. Regresion lineal simple/Multiple

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm

data = pd.read_csv('/home/alanv/Documents/7/mate2/CO2 Emissions_Canada.cs
```

Modelo regresion lineal

In [2]:
```
Y = data['CO2 Emissions(g/km)']
```

Utilizando Fuel Consumption Comb (L/100 km) como variable independiente

Calcular el coeficiente de determinación R^2

In [3]:
```
X_combCO2L = data['Fuel Consumption Comb (L/100 km)']
X_combCO2L = sm.add_constant(X_combCO2L)
print(X_combCO2L)

model = sm.OLS(Y, X_combCO2L)
result = model.fit()
print(result.params)
print('\n','R2:', result.rsquared)
```

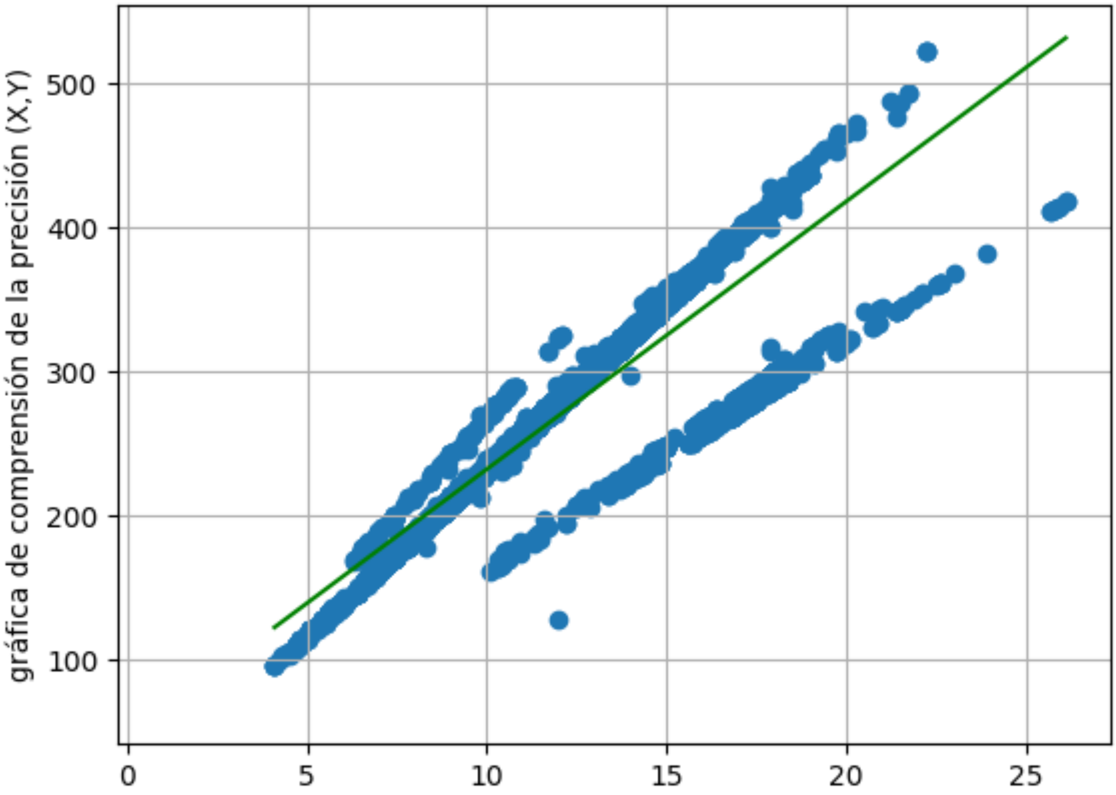
```
      const  Fuel Consumption Comb (L/100 km)
0       1.0                               8.5
1       1.0                               9.6
2       1.0                               5.9
3       1.0                              11.1
4       1.0                              10.6
...     ...                               ...
7380    1.0                               9.4
7381    1.0                               9.9
7382    1.0                              10.3
7383    1.0                               9.9
7384    1.0                              10.7

[7385 rows x 2 columns]
const                               46.763152
Fuel Consumption Comb (L/100 km)    18.571319
dtype: float64

 R2: 0.8428186895623988
```

Realizar la gráfica de comprensión de la precisión (X,Y)

In [4]:
```
m = 18.571319
b = 46.763152
```

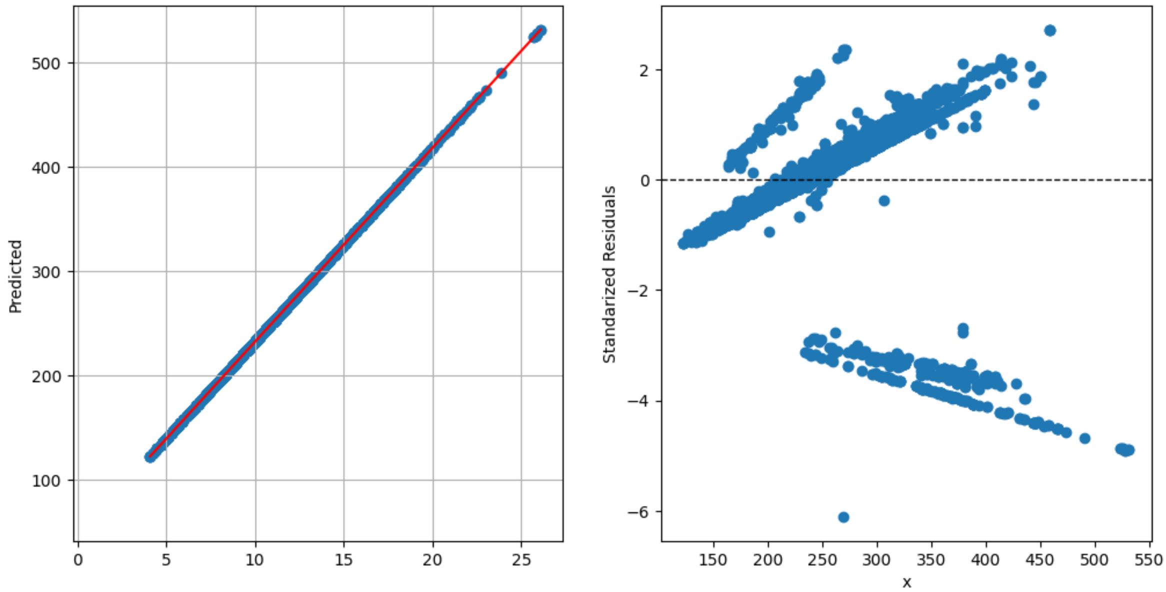
```
X_line = np.linspace(X_combCO2L.min(), X_combCO2L.max(), 100)
Y_line = m * X_line + b
Y_hat = result.predict(X_combCO2L)
plt.plot(X_line, Y_line, color='green')
plt.scatter(data['Fuel Consumption Comb (L/100 km)'], Y)
plt.grid()
plt.ylabel('gráfica de comprensión de la precisión (X,Y)')
```

Out[4]: Text(0, 0.5, 'gráfica de comprensión de la precisión (X,Y)')

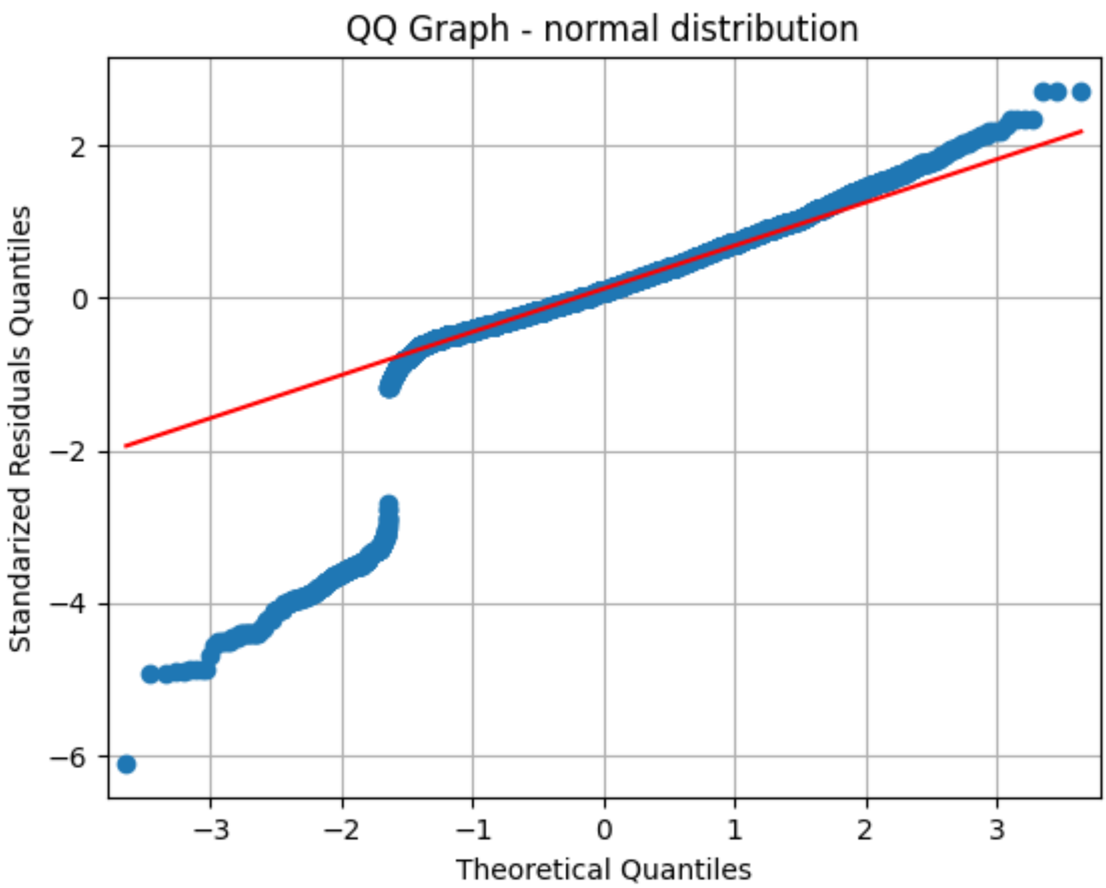


In [18]:
```
# Get the residuals
influence = result.get_influence()

# Calculate standardized residuals
std_residuals = influence.resid_studentized_internal
figure, axis = plt.subplots(1, 2, figsize=(12, 6))
axis[1].scatter(result.fittedvalues, std_residuals)
axis[1].set_xlabel('x')
axis[1].set_ylabel('Standarized Residuals')
axis[1].axhline(y=0, color='black', linestyle='--',linewidth=1)
axis[0].plot(X_line, Y_line, color='red')
axis[0].scatter(data['Fuel Consumption Comb (L/100 km)'], Y_hat)
axis[0].grid()
axis[0].set_ylabel('Predicted')
```

Out[18]: Text(0, 0.5, 'Predicted')



In [21]:
```python
from scipy.stats import norm, uniform, skewnorm

fig = sm.qqplot(std_residuals, line ='q') # dist = skewnorm(10)

plt.title('QQ Graph - normal distribution')
plt.ylabel('Standarized Residuals Quantiles')
plt.grid()
plt.show()
```



Utilizando Transformacion

In [7]:
```python
# aplicando Log10
```

```python
r_root = np.log10(1 + Y + abs(min(Y)))


model = sm.OLS(r_root, X_combCO2L)
result = model.fit()
print('\n','R2:', result.rsquared)
```

```
 R2: 0.8325284987248827
```

In [8]:

```python
# aplicando raiz cuadrada
r_root = np.sqrt(Y + abs(min(Y)))


model = sm.OLS(r_root, X_combCO2L)
result = model.fit()
print('\n','R2:', result.rsquared)
```

```
 R2: 0.8407523790886398
```

Al transformar la variable dependiente Y, no se observa ninguna mejora, de hecho empeora el resultado al aplicar raiz cuadrada y Logaritmo en base 10 a la variable Y

Prueba de hipótesis utilizando Fuel Consumption City (L/100 km)

Calcular el coeficiente de determinación R^2

In [9]:

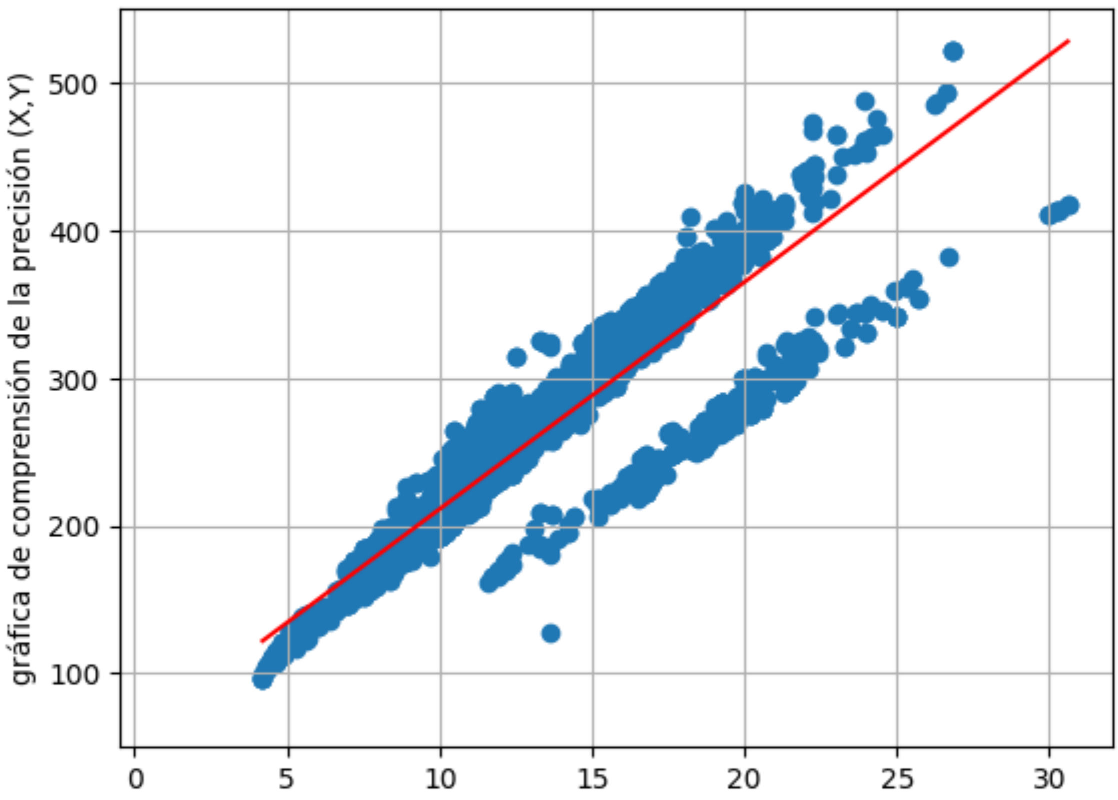
```python
#Fuel Consumption City (L/100 km)
X_combCity = data['Fuel Consumption City (L/100 km)']
X_combCity = sm.add_constant(X_combCity)
print(X_combCity)

model = sm.OLS(Y, X_combCity)
result = model.fit()
print(result.params)
print('\n','R2:', result.rsquared)
```

```
      const  Fuel Consumption City (L/100 km)
0       1.0                               9.9
1       1.0                              11.2
2       1.0                               6.0
3       1.0                              12.7
4       1.0                              12.1
...     ...                               ...
7380    1.0                              10.7
7381    1.0                              11.2
7382    1.0                              11.7
7383    1.0                              11.2
7384    1.0                              12.2

[7385 rows x 2 columns]
const                               57.559903
Fuel Consumption City (L/100 km)    15.372459
dtype: float64

 R2: 0.8456503198972763
```

Realizar la gráfica de comprensión de la precisión (X,Y)

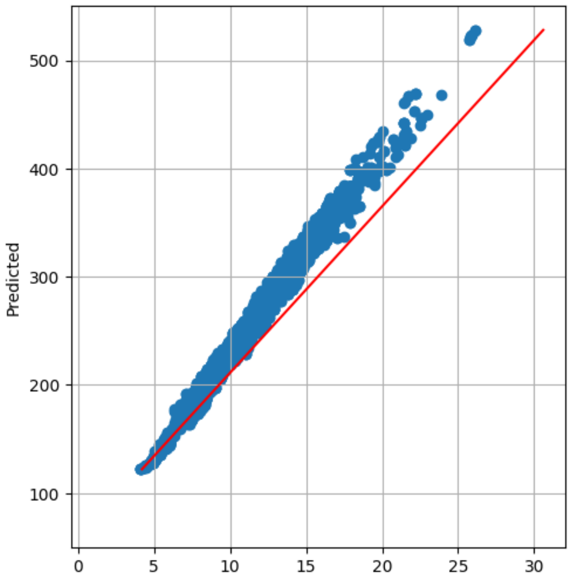
In [10]:
```
m = 15.372459
b = 57.559903
X_line = np.linspace(X_combCity.min(), X_combCity.max(), 100)
Y_line = m * X_line + b
Y_hat = result.predict(X_combCity)

plt.plot(X_line, Y_line, color='red', label='Línea recta de ajuste')
plt.scatter(data['Fuel Consumption City (L/100 km)'], Y)
plt.grid()
plt.ylabel('gráfica de comprensión de la precisión (X,Y)')
```

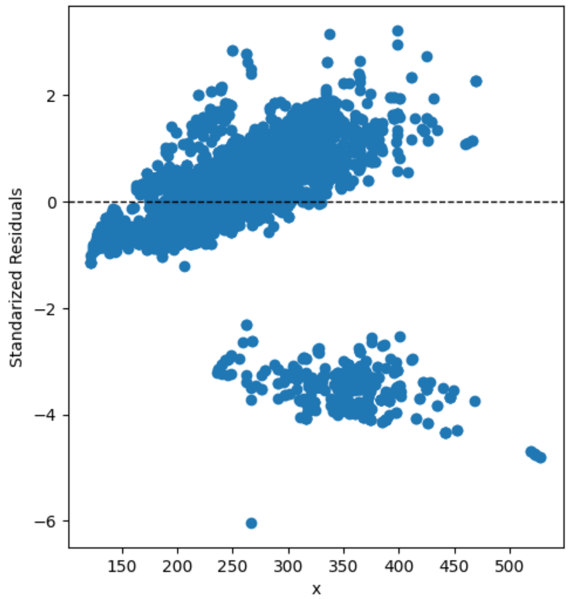
Out[10]:
```
Text(0, 0.5, 'gráfica de comprensión de la precisión (X,Y)')
```



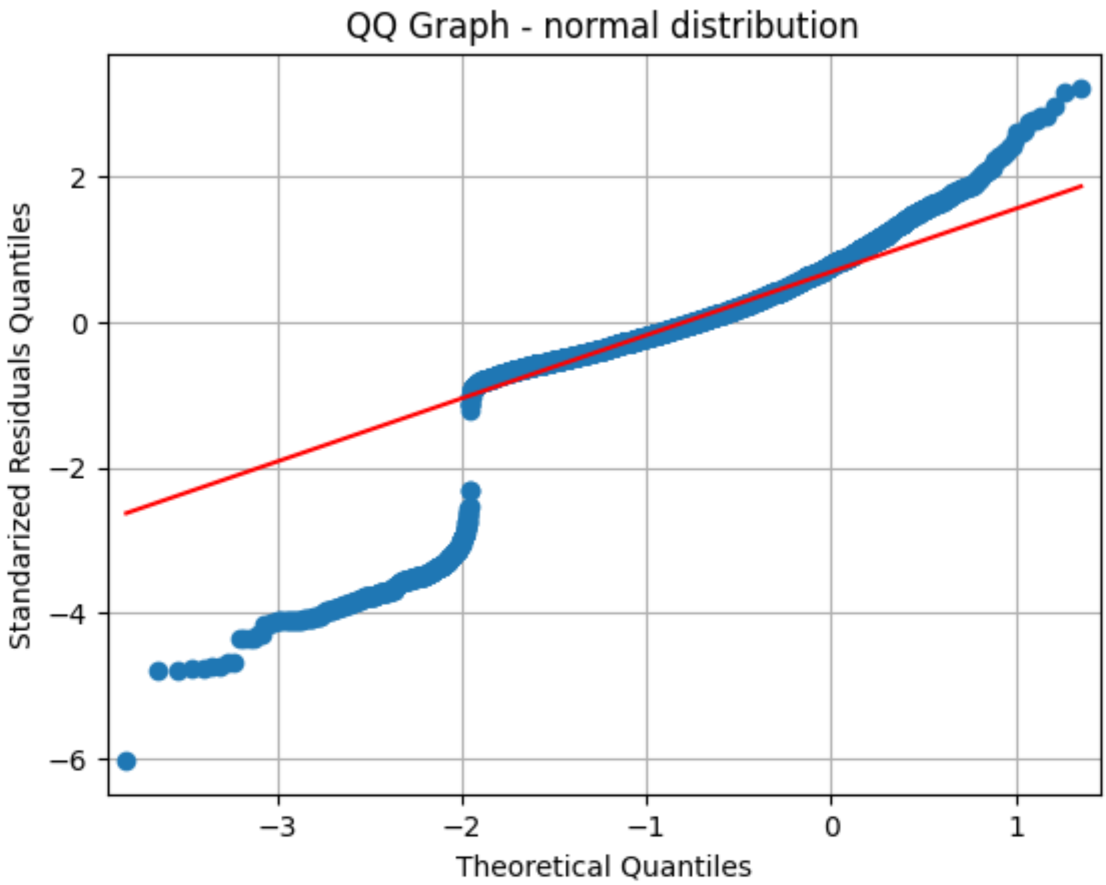
In [11]:
```
# Get the residuals
influence = result.get_influence()

# Calculate standardized residuals
std_residuals = influence.resid_studentized_internal
figure, axis = plt.subplots(1, 2, figsize=(12, 6))
axis[1].scatter(result.fittedvalues, std_residuals)
axis[1].set_xlabel('x')
axis[1].set_ylabel('Standarized Residuals')
axis[1].axhline(y=0, color='black', linestyle='--',linewidth=1)
axis[0].plot(X_line, Y_line, color='red')
axis[0].scatter(data['Fuel Consumption Comb (L/100 km)'], Y_hat)
axis[0].grid()
axis[0].set_ylabel('Predicted')
```

Out[11]:
```
Text(0, 0.5, 'Predicted')
```





In [16]:
```
from scipy.stats import norm, uniform, skewnorm

fig = sm.qqplot(std_residuals, dist = skewnorm(-2), line ='q')

plt.title('QQ Graph - normal distribution')
plt.ylabel('Standarized Residuals Quantiles')
plt.grid()
plt.show()
```



Utilizando Transformacion

In [12]:
```
# aplicando Log10
```

```
r_root = np.log10(1 + Y + abs(min(Y)))


model = sm.OLS(r_root, X_combCity)
result = model.fit()
print('\n','R2:', result.rsquared)
```

```
R2: 0.836968070475829
```

In [13]:
```
# aplicando raiz cuadrada
r_root = np.sqrt(Y + abs(min(Y)))


model = sm.OLS(r_root, X_combCity)
result = model.fit()
print('\n','R2:', result.rsquared)
```

```
R2: 0.844322646434444
```

Prueba de hipótesis utilizando Fuel Consumption Hwy (L/100 km)

Calcular el coeficiente de determinación R^2

In [16]:
```
#Fuel Consumption Hwy (L/100 km)
X_combHwy = data['Fuel Consumption Hwy (L/100 km)']
X_combHwy = sm.add_constant(X_combHwy)
print(X_combHwy)

model = sm.OLS(Y, X_combHwy)
result = model.fit()
print(result.params)
print('\n','R2:', result.rsquared)
```

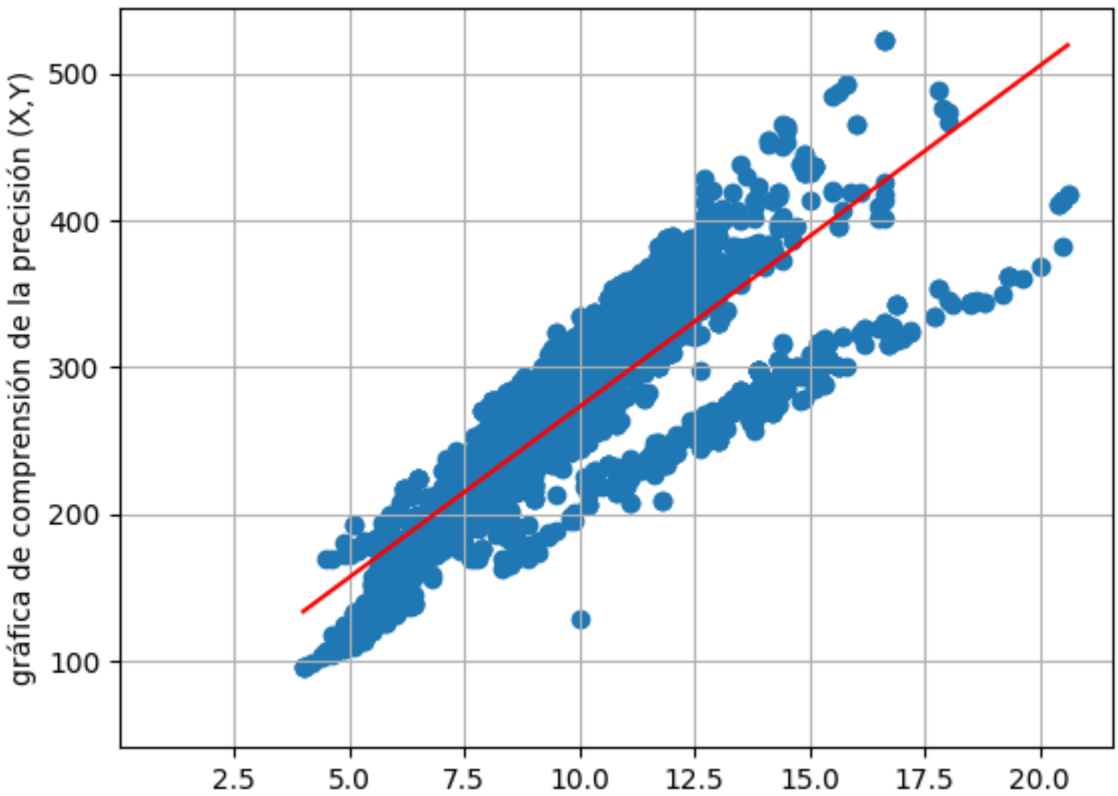
```
      const  Fuel Consumption Hwy (L/100 km)
0       1.0                              6.7
1       1.0                              7.7
2       1.0                              5.8
3       1.0                              9.1
4       1.0                              8.7
...     ...                              ...
7380    1.0                              7.7
7381    1.0                              8.3
7382    1.0                              8.6
7383    1.0                              8.3
7384    1.0                              8.7

[7385 rows x 2 columns]
const                              40.448581
Fuel Consumption Hwy (L/100 km)    23.240759
dtype: float64

 R2: 0.7806357669286315
```

Realizar la gráfica de comprensión de la precisión (X,Y)

In [17]:
```
m = 23.240759
b = 40.448581
X_line = np.linspace(X_combHwy.min(), X_combHwy.max(), 100)
```

```
Y_line = m * X_line + b
Y_hat = result.predict(X_combHwy)
plt.plot(X_line, Y_line, color='red', label='Línea recta de ajuste')
plt.scatter(data['Fuel Consumption Hwy (L/100 km)'], Y)
plt.grid()
plt.ylabel('gráfica de comprensión de la precisión (X,Y)')
```

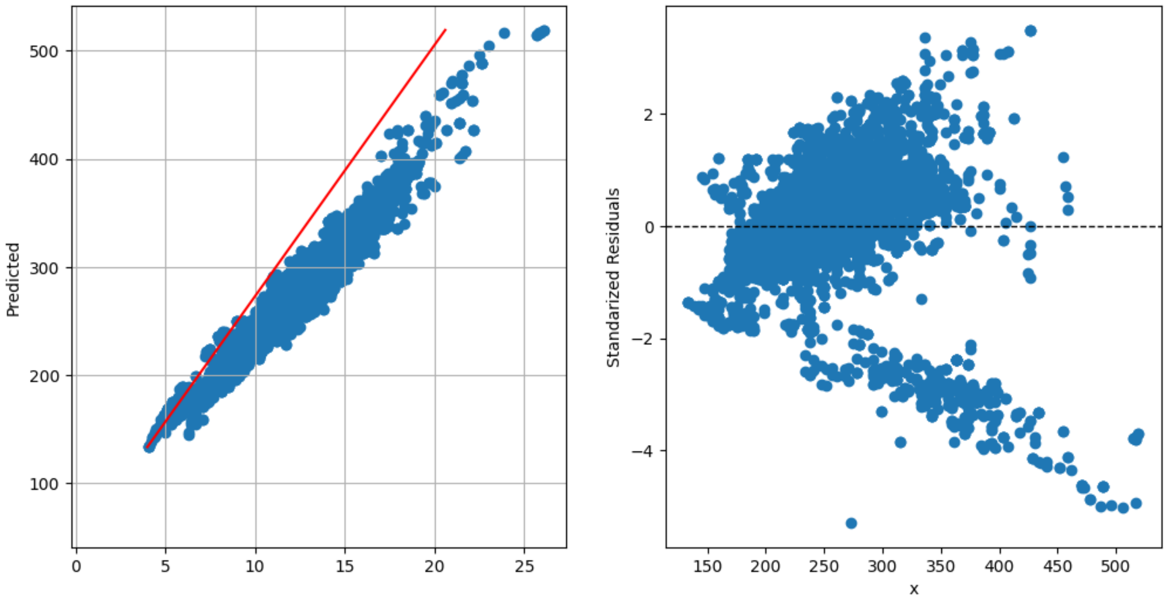
Out[17]: Text(0, 0.5, 'gráfica de comprensión de la precisión (X,Y)')



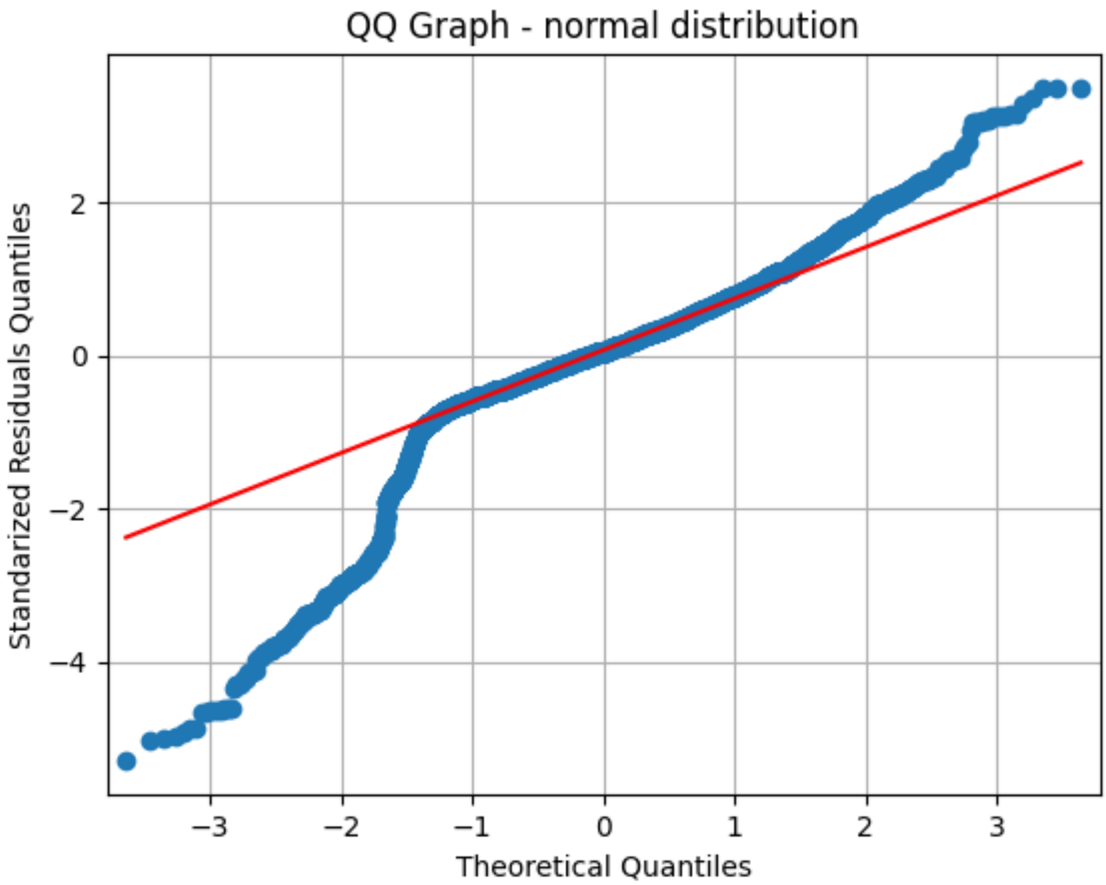
In [30]:

```
# Get the residuals
influence = result.get_influence()

# Calculate standardized residuals
std_residuals = influence.resid_studentized_internal
figure, axis = plt.subplots(1, 2, figsize=(12, 6))
axis[1].scatter(result.fittedvalues, std_residuals)
axis[1].set_xlabel('x')
axis[1].set_ylabel('Standarized Residuals')
axis[1].axhline(y=0, color='black', linestyle='--',linewidth=1)
axis[0].plot(X_line, Y_line, color='red')
axis[0].scatter(data['Fuel Consumption Comb (L/100 km)'], Y_hat)
axis[0].grid()
axis[0].set_ylabel('Predicted')
```

Out[30]: Text(0, 0.5, 'Predicted')



In [20]:
```
from scipy.stats import norm, uniform, skewnorm

fig = sm.qqplot(std_residuals, dist = norm, line ='q')

plt.title('QQ Graph - normal distribution')
plt.ylabel('Standarized Residuals Quantiles')
plt.grid()
plt.show()
```



Utilizando Transformacion

In [18]:
```
# aplicando Log10
```

```
r_root = np.log10(1 + Y + abs(min(Y)))


model = sm.OLS(r_root, X_combHwy)
result = model.fit()
print('\n','R2:', result.rsquared)
```

```
R2: 0.7679970522479547
```

In [19]:
```
# aplicando raiz cuadrada
r_root = np.sqrt(Y + abs(min(Y)))


model = sm.OLS(r_root, X_combHwy)
result = model.fit()
print('\n','R2:', result.rsquared)
```

```
R2: 0.7773010877923558
```

Transformacion adecuada

Preguntas

¿Cuáles son las características que más influyen en la emisión de CO2? Aparentemente, es la combustión en la ciudad y la combustión combinada (carretera y ciudad) en litros por kilómetro, ya que ambos me dieron un valor de R cuadrada de 0.84, el mayor de entre todas las variables.

¿Habrá alguna diferencia en las emisiones de CO2 cuando el consumo de combustible para ciudad y carretera se consideren por separado? Al parecer, sí hay una diferencia, principalmente porque normalmente en carretera el consumo de combustible sería menor(L/Km) debido a las velocidades, ya que en la ciudad existen límites de velocidad más bajos además del tráfico. Si lo comparamos con los modelos, el valor de R cuadrada utilizando la emisión de CO2 en carretera es inferior al de la ciudad.

Modelo de regresion multiple

In [25]:
```
data.head()
```

Out[25]:

	Make	Model	Vehicle Class	Engine Size(L)	Cylinders	Transmission	Fuel Type	Fuel Consumption City (L/100 km)
0	ACURA	ILX	COMPACT	2.0	4	AS5	Z	9.9
1	ACURA	ILX	COMPACT	2.4	4	M6	Z	11.2
2	ACURA	ILX HYBRID	COMPACT	1.5	4	AV7	Z	6.0
3	ACURA	MDX 4WD	SUV - SMALL	3.5	6	AS6	Z	12.7
4	ACURA	RDX AWD	SUV - SMALL	3.5	6	AS6	Z	12.1

In [20]:

```
#Usando todas las columnas

X = data[['Engine Size(L)','Cylinders', 'Fuel Consumption City (L/100 km)
Y = data['CO2 Emissions(g/km)']
Y = np.sqrt(Y + abs(min(Y)))

#X = sm.add_constant(X)
print(X)

model = sm.OLS(Y, sm.add_constant(X)).fit()
#result = model.fit()
print(model.params)
print('\n','R2:', model.rsquared)
```

```
      Engine Size(L)  Cylinders  Fuel Consumption City (L/100 km)  \
0                2.0          4                               9.9
1                2.4          4                              11.2
2                1.5          4                               6.0
3                3.5          6                              12.7
4                3.5          6                              12.1
...              ...        ...                               ...
7380             2.0          4                              10.7
7381             2.0          4                              11.2
7382             2.0          4                              11.7
7383             2.0          4                              11.2
7384             2.0          4                              12.2

      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
...                               ...                               ...
7380                              7.7                               9.4
7381                              8.3                               9.9
7382                              8.6                              10.3
7383                              8.3                               9.9
7384                              8.7                              10.7

      Fuel Consumption Comb (mpg)
0                              33
1                              29
2                              48
3                              25
4                              27
...                           ...
7380                           30
7381                           29
7382                           27
7383                           29
7384                           26

[7385 rows x 6 columns]
const                               19.436781
Engine Size(L)                       0.133053
Cylinders                            0.185578
Fuel Consumption City (L/100 km)    -0.005960
Fuel Consumption Hwy (L/100 km)      0.118165
Fuel Consumption Comb (L/100 km)    -0.006193
Fuel Consumption Comb (mpg)         -0.119108
dtype: float64

 R2: 0.9152323690604602
```

In [21]:

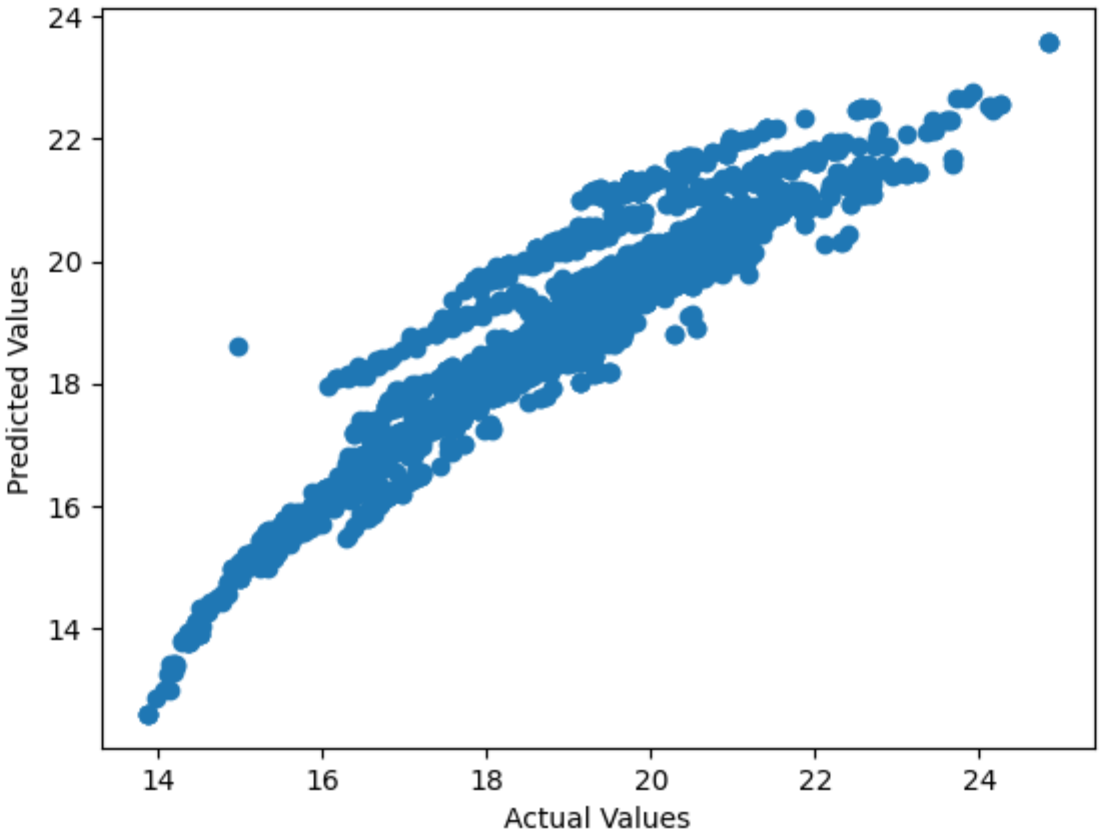
```python
model.summary()

X.shape
```

Out[21]:

```
(7385, 6)
```

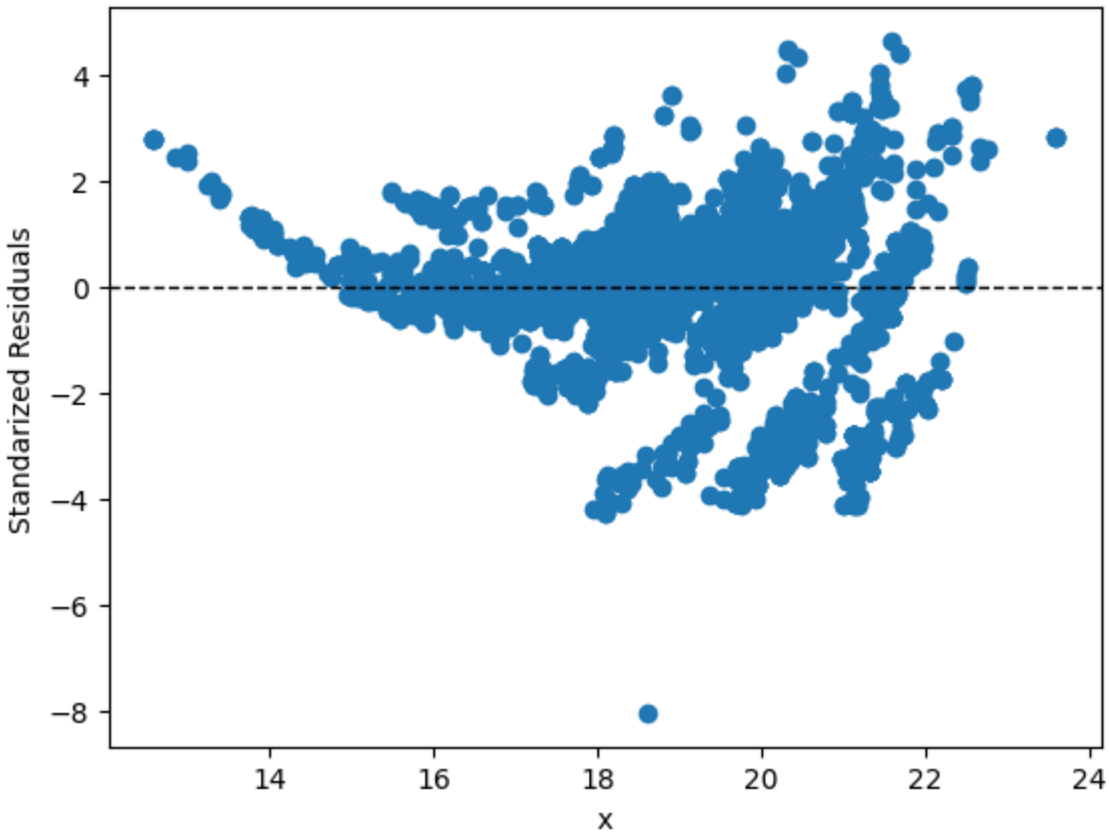
Realizar la gráfica de comprensión de la precisión (X,Y)

In [22]:
```
# Get predicted values
predicted_values = model.predict(sm.add_constant(X))

# Create a scatter plot
plt.scatter(Y, predicted_values)
plt.xlabel("Actual Values")
plt.ylabel("Predicted Values")
plt.show()
```

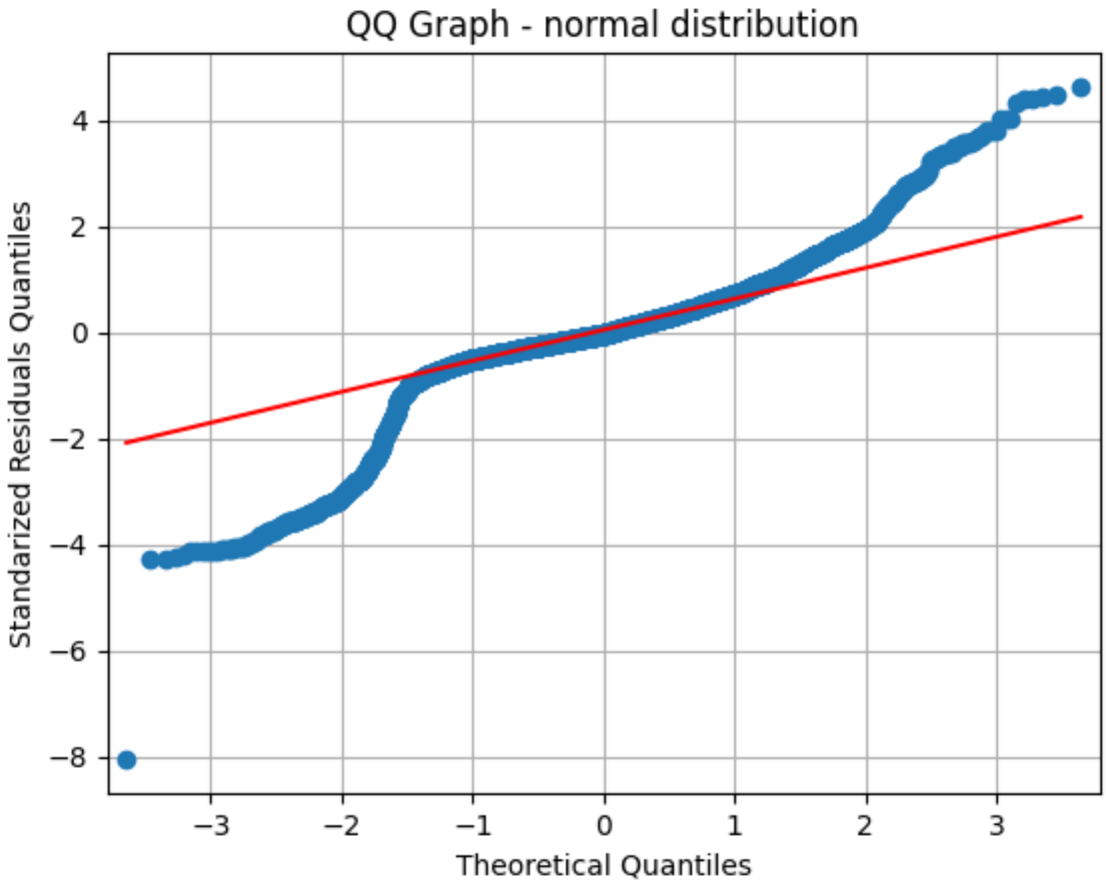


In [23]:
```
# Get the residuals
influence = model.get_influence()

# Calculate standardized residuals
std_residuals = influence.resid_studentized_internal

plt.scatter(model.fittedvalues, std_residuals)
plt.xlabel('x')
plt.ylabel('Standarized Residuals')
plt.axhline(y=0, color='black', linestyle='--',linewidth=1)
```

Out[23]: <matplotlib.lines.Line2D at 0x7f1b9c356dd0>



In [24]:
```
from scipy.stats import norm, uniform, skewnorm

fig = sm.qqplot(std_residuals, dist = norm, line ='q')

plt.title('QQ Graph - normal distribution')
plt.ylabel('Standarized Residuals Quantiles')
plt.grid()
plt.show()
```



In [25]:
```
model.summary()
```

Out[25]: OLS Regression Results

Dep. Variable:	CO2 Emissions(g/km)	**R-squared:**	0.915
Model:	OLS	**Adj. R-squared:**	0.915
Method:	Least Squares	**F-statistic:**	1.328e+04
Date:	Fri, 06 Oct 2023	**Prob (F-statistic):**	0.00
Time:	17:32:44	**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
Engine Size(L)	0.1331	0.011	11.687	0.000	0.111	0.155
Cylinders	0.1856	0.008	23.303	0.000	0.170	0.201
Fuel Consumption City (L/100 km)	-0.0060	0.068	-0.087	0.931	-0.140	0.128
Fuel Consumption Hwy (L/100 km)	0.1182	0.056	2.093	0.036	0.007	0.229
Fuel Consumption Comb (L/100 km)	-0.0062	0.124	-0.050	0.960	-0.250	0.237
Fuel Consumption Comb (mpg)	-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.

Prueba de hipotesis

In [26]:
```
X = data[['Engine Size(L)','Cylinders', 'Fuel Consumption Hwy (L/100 km)'
model = sm.OLS(Y, sm.add_constant(X)).fit()
print('\n','R2:', model.rsquared)
```

```
R2: 0.9152103349606883
```

In [27]:
```
model.summary()
```

Out[27]:

OLS Regression Results

Dep. Variable:	CO2 Emissions(g/km)	**R-squared:**	0.915
Model:	OLS	**Adj. R-squared:**	0.915
Method:	Least Squares	**F-statistic:**	1.991e+04
Date:	Fri, 06 Oct 2023	**Prob (F-statistic):**	0.00
Time:	17:33:14	**Log-Likelihood:**	-4636.1
No. Observations:	7385	**AIC:**	9282.
Df Residuals:	7380	**BIC:**	9317.
Df Model:	4		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	19.3676	0.092	209.826	0.000	19.187	19.549
Engine Size(L)	0.1313	0.011	11.606	0.000	0.109	0.154
Cylinders	0.1830	0.008	23.624	0.000	0.168	0.198
Fuel Consumption Hwy (L/100 km)	0.1078	0.005	19.768	0.000	0.097	0.119
Fuel Consumption Comb (mpg)	-0.1177	0.002	-70.478	0.000	-0.121	-0.114

Omnibus:	1452.486	**Durbin-Watson:**	1.623
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	8021.114
Skew:	-0.832	**Prob(JB):**	0.00
Kurtosis:	7.827	**Cond. No.**	530.

Notes:

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

Preguntas 2

¿Qué sucede con el error y la distribución de este en los datos? Cubren un área más amplia además, se pueden diferenciar algunas clases fácilmente, ya que en el gráfico se puede observar una separación entre los datos.

¿Qué pasa con el fit del modelo y a qué se lo atribuye? Es cuando el modelo busca los mejores coeficientes (betas) para cada variable independiente, con la finalidad de que la variable de respuesta tenga el mínimo error posible, estos coeficientes se pueden encontrar en el summary del modelo.

Describa el impacto de las distintas variables. ¿Qué sucede si se omiten las variables con nulo impacto? Hay algunas variables donde la hipótesis nula se podría aceptar, como lo son Fuel Consumption City (L/100 km) y Fuel Consumption Comb (L/100 km), ya que su p-valor es mayor a 0.05, por lo que se puede suponer que estas variables no son muy importantes. Si se eliminan estas variables, el modelo reduciría su complejidad y la precisión podría ser mejor o la misma, debido a que la hipótesis nula nos dice que los coeficientes para esas variables son 0 y no afectan al modelo, como se puede observar en la prueba de hipótesis.