Modelo de regression lineal

July 31, 2024

1 Modelo de Regression Lineal

Elaboración: **Gabriel Armando Landín Alvarado**

1.1 ¿Qué es un modelo de regresión?

Un **Modelo de Regresión** en **Machine Learning** es un algoritmo que se utiliza para predecir valores numéricos continuos a partir de un conjunto de variables de entrada. En esencia, la regresión busca establecer una relación matemática entre la(s) **variable(s) independiente(s) o predictora(s)** [**X**] y la **variable dependiente u objetivo** [**y**], permitiéndonos hacer predicciones sobre el comportamiento futuro de un fenómeno. A continuación, se enumeran algunos puntos clave sobre los tipos de modelos de regresión:

1. **Regresión Lineal**: Este modelo asume una relación lineal entre las variables. Se utiliza para predecir valores numéricos continuos.
2. **Regresión Polinómica**: Permite modelar relaciones no lineales mediante polinomios. Es útil cuando los datos no siguen una tendencia lineal.
3. **Regresión Logística**: Aunque su nombre incluye "regresión", en realidad se utiliza para clasificación binaria. Estima la probabilidad de pertenencia a una clase.

Para evaluar la calidad del modelo, se utilizan métricas como el coeficiente de determinación (R2), el error cuadrático medio (MSE) o el error absoluto medio (MAE), entre otros. Además, se aplican técnicas de validación cruzada para evitar el sobreajuste (1).

1.2 Modelo de Regresión Lineal Simple

En el siguiente ejercicio se implementará un modelo de regresión lineal simple, este modelo solo considera una variable independiente para **X** y por supuesto la variable dependiente **y**. Se hará uso de la biblioteca **Scikit-learn**(sklearn) de Python para la creación e implementación del modelo.

Lo primero será cargar las bibliotecas necesarias para cada proceso, para el primero serán: pandas, numpy y matplotlib.pyplot, así como el método mágico necesario para la visualización de nuestros gráficos.

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline
```

1.2.1 Análisis de datos exploratorio

Los datos a usar para la aplicación del modelo de regresión lineal simple son los publicados por el gobierno de Canada referente a las clasificaciones de consumo de combustible para cada modelo vehícular, entre los datos importantes se incluye las emisiones estimadas de dióxido de carbono (CO2) de cada vehículo nuevo para su venta al menudeo en este país. La variable anterior será la variable dependiente y algunas de las restantes fungirán como variables independientes. Para este ejercicio se hará uso del conjunto de datos o dataset correspondiente al año 2023, los datos mencionados pueden descargarse aquí.

Lo primero después de descargar los datos será realizar un análisis exploratorio de los mismos, a continuación, se muestra algunos procesos de esta tarea.

[2]:
```
# Cargar del archivo csv con la función read_csv() de pandas
data = pd.read_csv('../DATA/fuel_consumption_ratings_2023.csv')
# imprimimos la información general de los datos
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 840 entries, 0 to 839
Data columns (total 15 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   Model_year                    840 non-null    int64
 1   Make                          840 non-null    object
 2   Model                         840 non-null    object
 3   Vehicle_class                 840 non-null    object
 4   Engine_size                   840 non-null    float64
 5   Cylinders                     840 non-null    int64
 6   Transmission                  840 non-null    object
 7   Fuel_type                     840 non-null    object
 8   Consumption_city(L/100km)     840 non-null    float64
 9   Consumption_highway(L/100km)  840 non-null    float64
 10  Consumption_combined(L/100km) 840 non-null    float64
 11  Consumption_combined(mpg)     840 non-null    int64
 12  CO2_emissions(g/km)           840 non-null    int64
 13  CO2_rating                    840 non-null    int64
 14  Smog_rating                   840 non-null    int64
dtypes: float64(4), int64(6), object(5)
memory usage: 98.6+ KB
```

Podemos observar que se tienen 840 resgistros, 15 columnas con diferentes tipos de dato, asimismo, vemos que no existen datos nulos, la información general de las columnas es la siguiente:

- **Model_year:** Año de modelo, en este caso modelos 2023
- **Make:** Fabricante del vehículo
- **Model:** Modelo de vehículo
- **Vehicle_class:** Tipo o clase del vehículo
- **Engine_size:** Tamaño de motor
- **Cylinders:** Número de cilindros

- **Transmission:** Tipo de transmisión
- **Fuel_type:** Tipo de combustible
- **Consumption_city(L/100km):** Consumo en litros por cada 100 km en ciudad
- **Consumption_highway(L/100km):** Consumo en litros por cada 100 km en carretera
- **Consumption_combined(L/100km):** Consumo en litros por cada 100 km combinado
- **Consumption_combined(mpg):** Consumo combinado en millas por galón
- **CO2_emissions(g/km):** Emisiones de dióxido de carbono gramos sobre km
- **CO2_rating:** Clasificación con base en las emisiones de CO2
- **Smog_rating:** Clasificación de acuerdo al smog o niebla tóxica

[3]:
```python
# Visualizamos las primeras 5 filas
data.head()
```

[3]:
```
   Model_year   Make               Model                     Vehicle_class  \
0        2023  Acura             Integra                         Full-size
1        2023  Acura      Integra A-SPEC                         Full-size
2        2023  Acura      Integra A-SPEC                         Full-size
3        2023  Acura          MDX SH-AWD     Sport utility vehicle: Small
4        2023  Acura  MDX SH-AWD Type S  Sport utility vehicle: Standard

   Engine_size  Cylinders Transmission Fuel_type  Consumption_city(L/100km)  \
0          1.5          4          AV7         Z                        7.9
1          1.5          4          AV7         Z                        8.1
2          1.5          4           M6         Z                        8.9
3          3.5          6         AS10         Z                       12.6
4          3.0          6         AS10         Z                       13.8

   Consumption_highway(L/100km)  Consumption_combined(L/100km)  \
0                           6.3                            7.2
1                           6.5                            7.4
2                           6.5                            7.8
3                           9.4                           11.2
4                          11.2                           12.4

   Consumption_combined(mpg)  CO2_emissions(g/km)  CO2_rating  Smog_rating
0                         39                  167           6            7
1                         38                  172           6            7
2                         36                  181           6            6
3                         25                  263           4            5
4                         23                  291           4            5
```

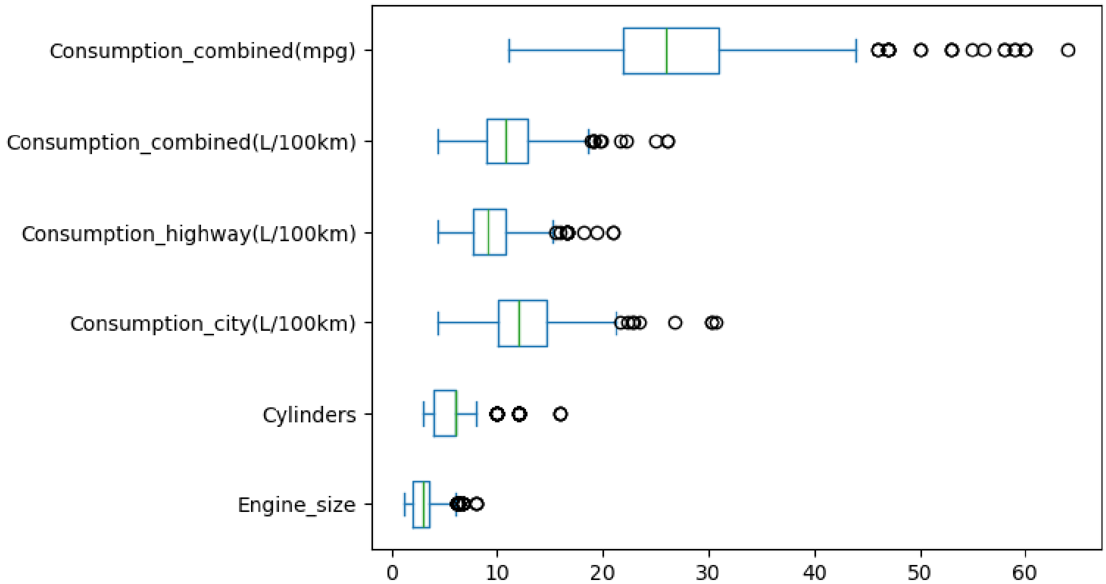
[14]:
```python
# Observamos la forma del dataframe, es decir, número de filas y columnas
data.shape
```

[14]: (840, 15)

[15]:
```python
# Realizamos una exploración de estadística descriptiva de los datos
data.describe()
```

[15]:
```
       Model_year  Engine_size   Cylinders  Consumption_city(L/100km)  \
count       840.0   840.000000  840.000000                 840.000000
mean       2023.0     3.156429    5.638095                  12.472738
std           0.0     1.357584    1.968156                   3.521936
min        2023.0     1.200000    3.000000                   4.400000
25%        2023.0     2.000000    4.000000                  10.100000
50%        2023.0     3.000000    6.000000                  12.100000
75%        2023.0     3.600000    6.000000                  14.700000
max        2023.0     8.000000   16.000000                  30.700000

       Consumption_highway(L/100km)  Consumption_combined(L/100km)  \
count                    840.000000                     840.000000
mean                       9.374762                      11.079167
std                        2.321770                       2.922871
min                        4.400000                       4.400000
25%                        7.700000                       9.000000
50%                        9.100000                      10.750000
75%                       10.800000                      12.900000
max                       20.900000                      26.100000

       Consumption_combined(mpg)  CO2_emissions(g/km)  CO2_rating  Smog_rating
count                 840.000000           840.000000  840.000000   840.000000
mean                   27.320238           258.048810    4.511905     5.227381
std                     7.574883            64.662256    1.286160     1.675587
min                    11.000000           104.000000    1.000000     1.000000
25%                    22.000000           211.000000    4.000000     5.000000
50%                    26.000000           255.000000    5.000000     5.000000
75%                    31.000000           299.000000    5.000000     7.000000
max                    64.000000           608.000000    9.000000     8.000000
```

[44]:
```python
# gráficamos mediante un diagrama de caja y bigote algunas variables␣
↪interesantes para ser consideradas como variables independientes en nuestro␣
↪modelo
data[['Engine_size', 'Cylinders', 'Consumption_city(L/100km)',␣
↪'Consumption_highway(L/100km)',  'Consumption_combined(L/100km)',␣
↪'Consumption_combined(mpg)']].plot.box(vert=False)
plt.show()
```



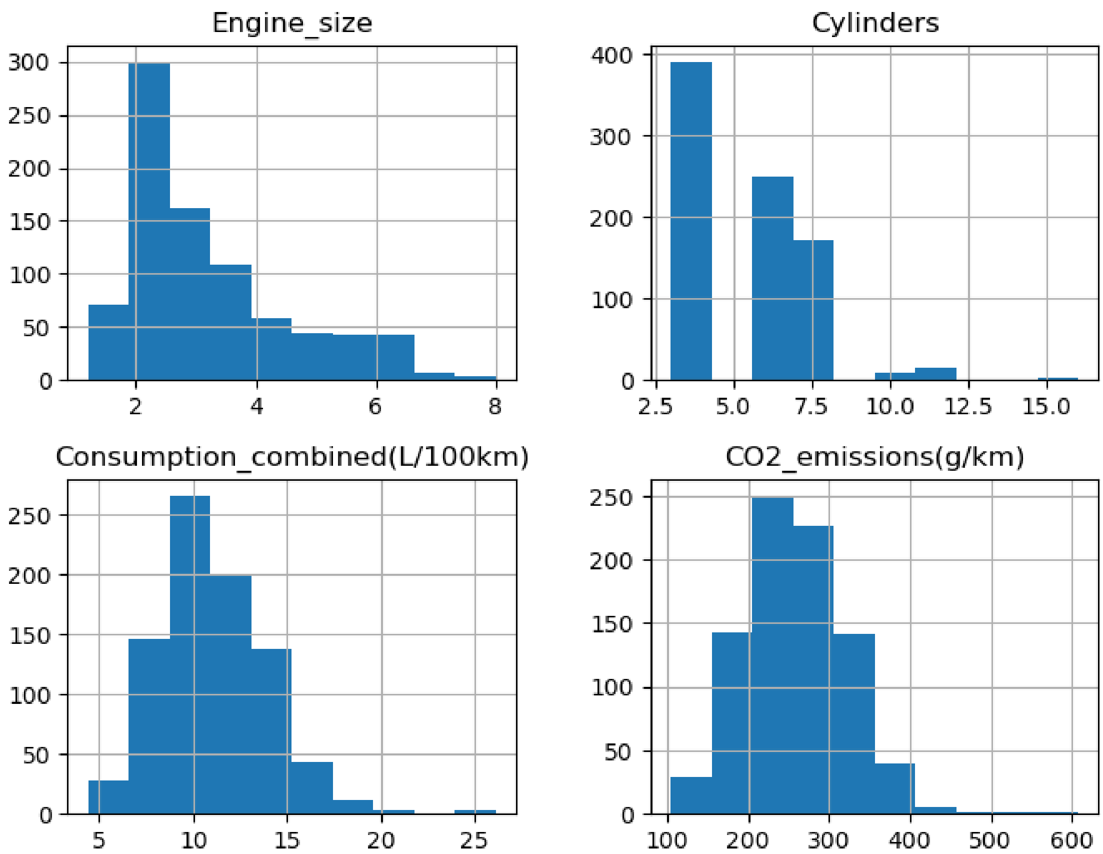
Se seleccionan solo algunas de las características o columnas de interés, en este caso: Engine_size, Cylinders y Consumption_combined(L/100km) como variables independientes, asimismo, la variable dependiente o "target" CO2_emissions(g/km).

[17]:
```
data_1 = data[['Engine_size', 'Cylinders', 'Consumption_combined(L/100km)',
↪'CO2_emissions(g/km)']]
data_1.head()
```

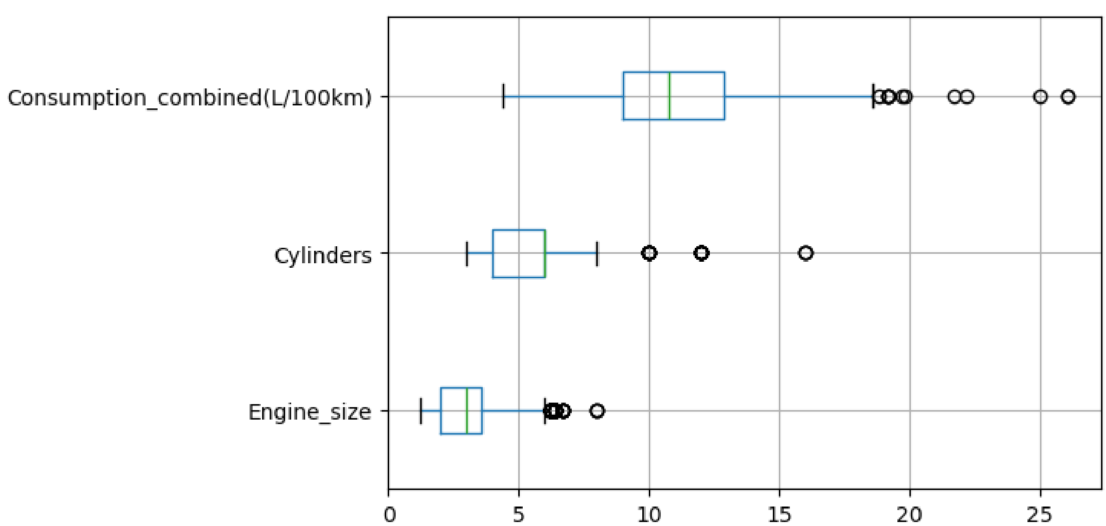
[17]:

	Engine_size	Cylinders	Consumption_combined(L/100km)	CO2_emissions(g/km)
0	1.5	4	7.2	167
1	1.5	4	7.4	172
2	1.5	4	7.8	181
3	3.5	6	11.2	263
4	3.0	6	12.4	291

[18]:
```
# Se visualiza la distribución de cada columna mediante un histograma
data_1.hist(figsize=(8, 6))

plt.show()
```

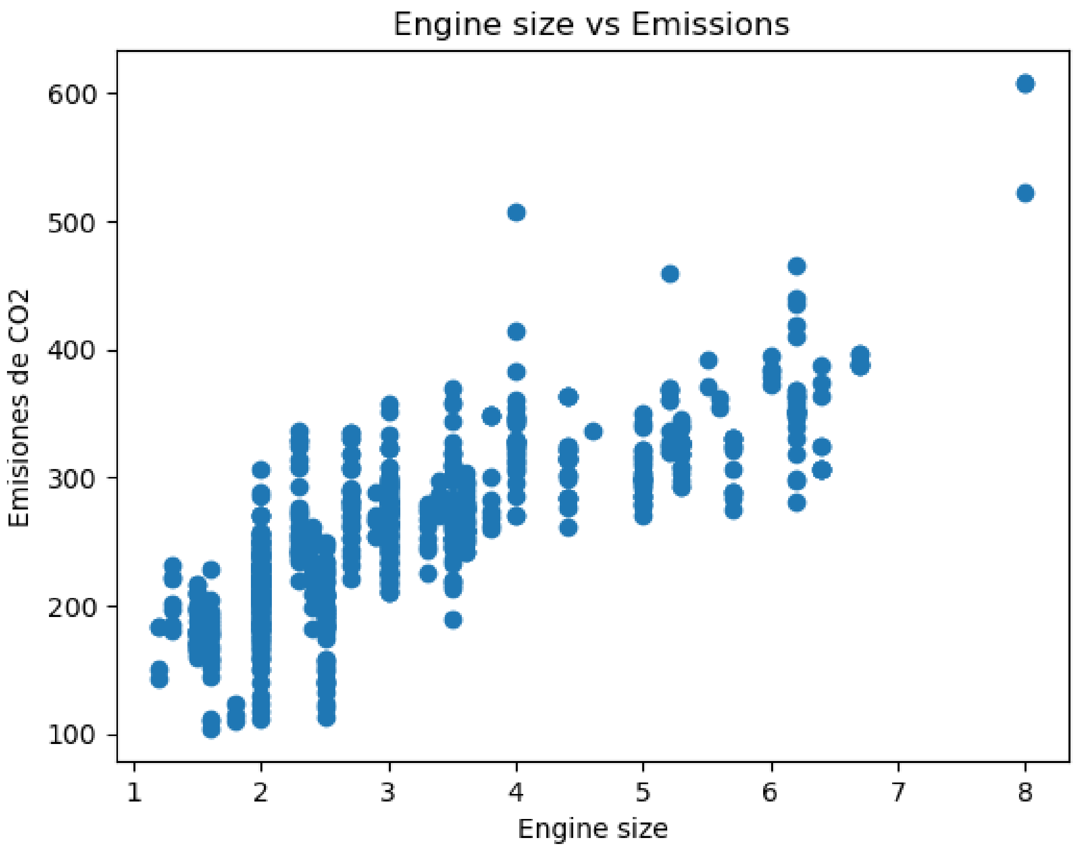


[45]:
```
# graficos de caja y bigote solo de las variables independientes
data_1.drop('CO2_emissions(g/km)', axis=1).boxplot(figsize=(6, 4), vert=False)
plt.show()
```

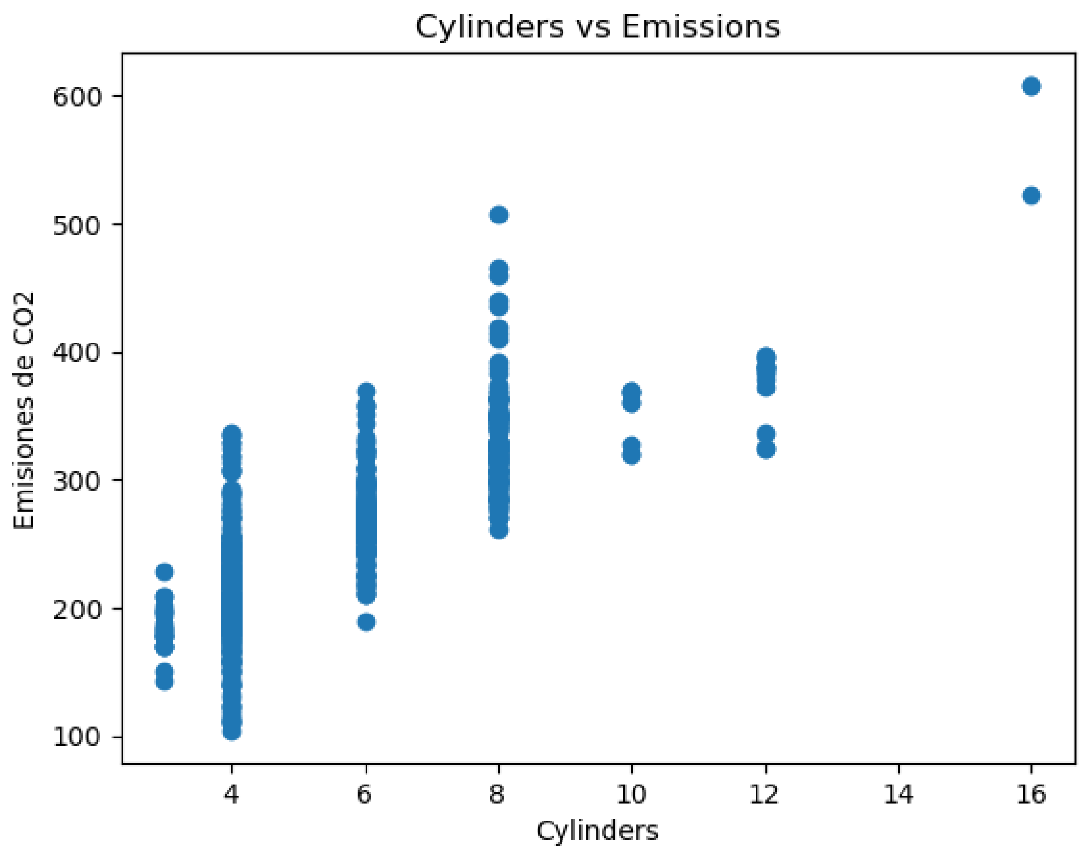


Se gráfica un diagrama de dispersión de cada una de las variables independientes frente a la variable dependiente, para ver su relación:

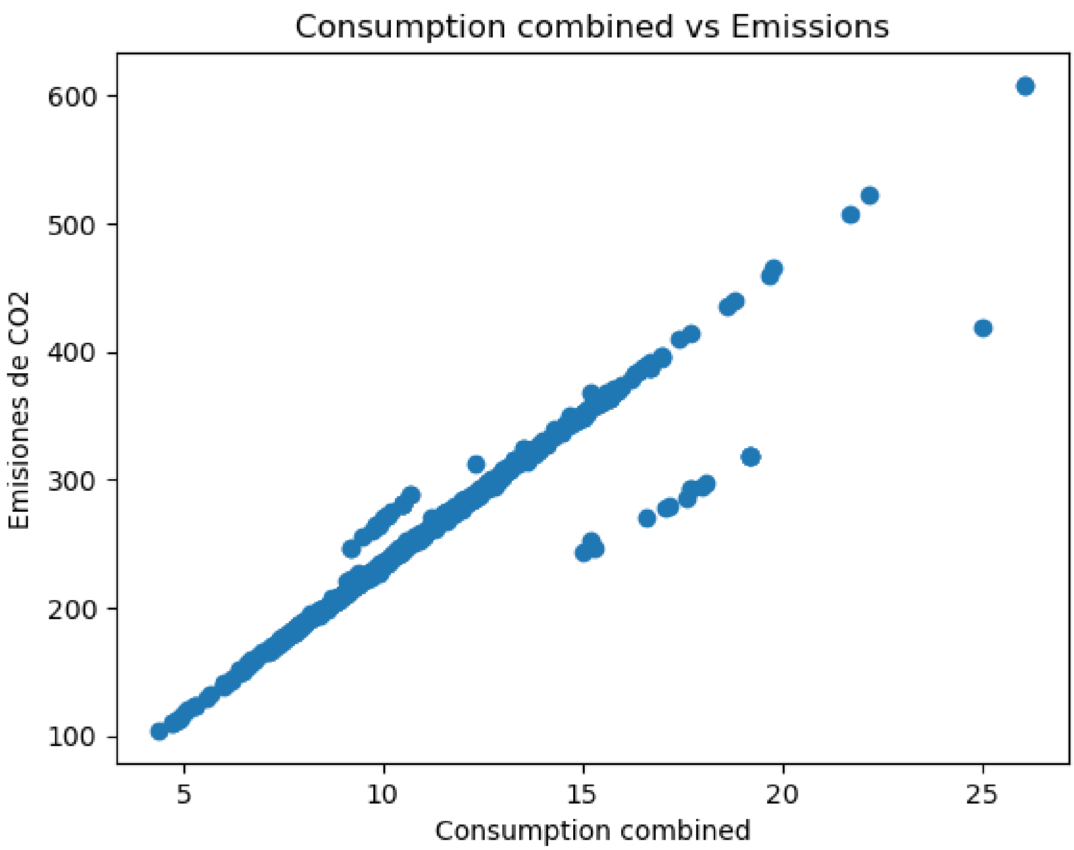
[27]:
```
plt.scatter(data_1['Engine_size'], data_1['CO2_emissions(g/km)'])
plt.xlabel('Engine size')
plt.ylabel('Emisiones de CO2')
plt.title('Engine size vs Emissions')

plt.show()
```



[28]:
```
plt.scatter(data_1['Cylinders'], data_1['CO2_emissions(g/km)'])
plt.xlabel('Cylinders')
plt.ylabel('Emisiones de CO2')
plt.title('Cylinders vs Emissions')

plt.show()
```




```
[29]: plt.scatter(data_1['Consumption_combined(L/100km)'], data_1['CO2_emissions(g/
      ↪km)'])
      plt.xlabel('Consumption combined')
      plt.ylabel('Emisiones de CO2')
      plt.title('Consumption combined vs Emissions')

      plt.show()
```



1.2.2 Creación de los conjuntos de datos de entrenamiento (train) y prueba (test)

Los conjuntos de datos train y test implican dividir el total de datos en subconjuntos de entrenamiento y prueba los cuales son mutuamente excluyentes. El modelo se entrena con el conjunto train y se realiza la verificación del ajuste con el conjunto test. Esto proporcionará una evaluación de la precisión del modelo, ya que el conjunto de datos test no forma parte del conjunto que se utilizó para entrenar el modelo. Por tanto, nos brinda una comprensión de qué tan bien 'generaliza' o 'aprende' nuestro modelo. Esto significa que conocemos el resultado de la variable objetivo en los datos del conjunto test, es decir, podemos realizar pruebas y verificar el aprendizaje de nuestro modelo al comparar los resultados que arroja el modelo con los existentes.

Dividamos nuestro conjunto de datos en conjuntos de entrenamiento y de prueba. El 80% de todo el conjunto de datos se utilizará para entrenamiento y el 20% para pruebas, estos porcentajes pueden cambiar, aunque lo más usual es usar esta proporción.

De inicio se crea una máscara para seleccionar filas aleatorias usando la función random.rand() de Numpy:

[30]:
```
# crear un cojunto de datos aleatorios de tamaño 80% respecto a la longitud o␣
↪total de registros
mascara = np.random.rand(len(data_1)) < 0.8
```

```
# selecionar el 80% de los dsatos para el conjunto de entrenamiento
train = data_1[mascara]
# seleccionar el restante para el conjunto de prueba
test = data_1[~mascara]
```

[31]:
```
# imprimir el número de filas y columnas para cada conjunto
print(train.shape)
print(test.shape)
```

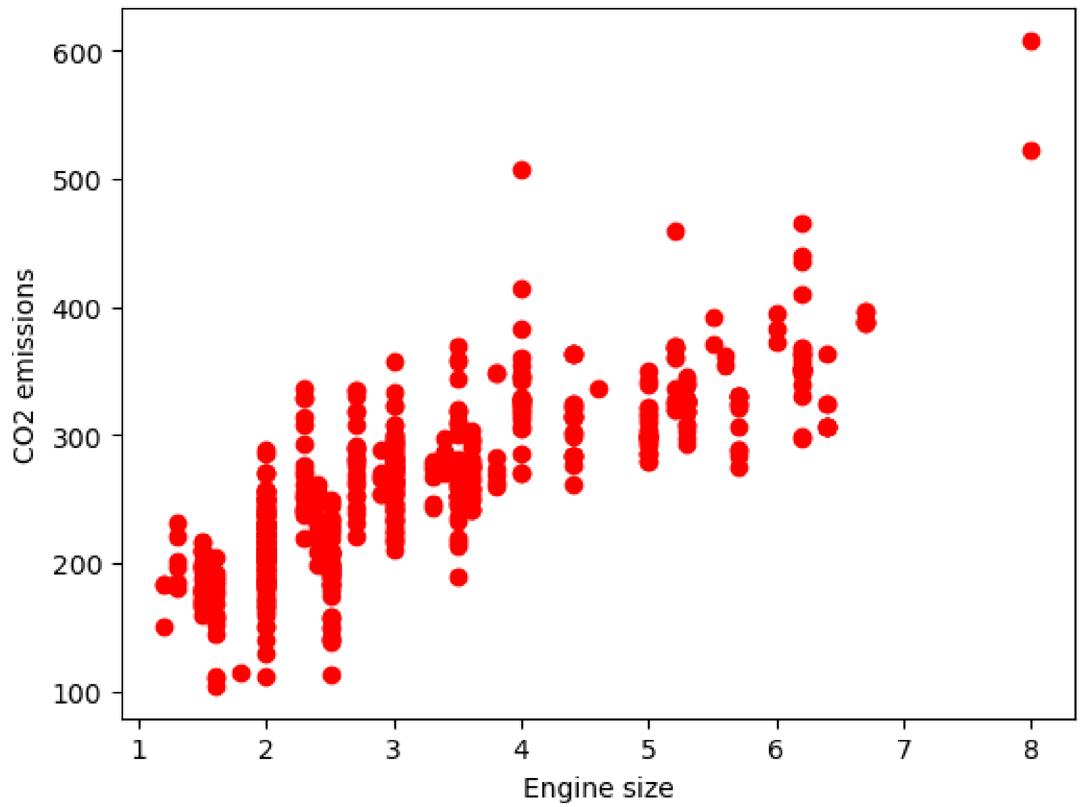
```
(681, 4)
(159, 4)
```

1.2.3 Creación del modelo de regresión lineal simple

Para generalizar, se puede decir que el modelo de regresión lineal se ajusta a un modelo lineal con coeficientes $\beta = (\beta_1 , ..., \beta_n)$ para minimizar la **suma residual de cuadrados** entre el valor real y del conjunto de datos y el valor predicho $\hat{y}$ mediante una aproximación lineal

Veamos un ejemplo gráfico de lo dicho anteriormente para la variable independiente **Engine_size** o tamaño de motor. .

[32]:
```
# Distribución de la relación líneal de los datos de entrenamiento
plt.scatter(train['Engine_size'], train['CO2_emissions(g/km)'], color='red')
plt.xlabel('Engine size')
plt.ylabel('CO2 emissions')

plt.show()
```

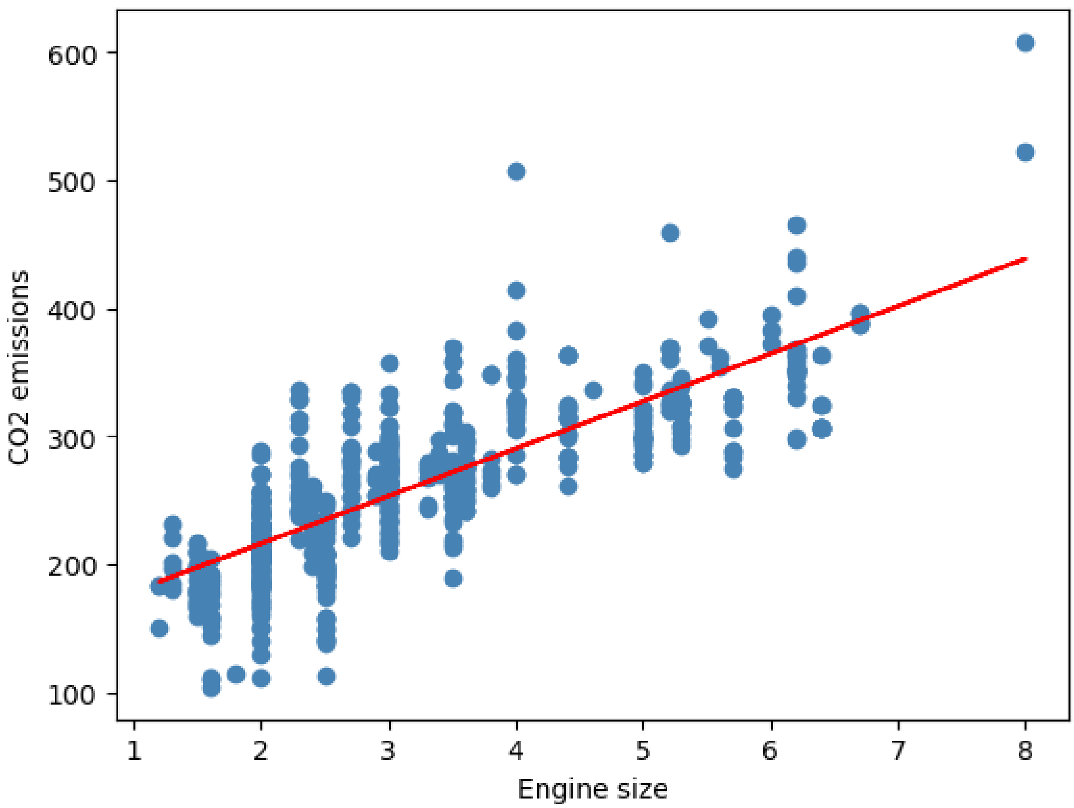


[33]:
```
# importar de la biblioteca sklearn el modelo lineal
from sklearn import linear_model
# instanciar el modelo de regresión lineal en un objeto
regr = linear_model.LinearRegression()
#crear los conjuntos de entrenamiento
X_train = np.asanyarray(train[['Engine_size']])
y_train = np.asanyarray(train[['CO2_emissions(g/km)']])
# se ajusta o entrena el modelo
regr.fit(X_train, y_train)
# Obtenemos los coeficientes
print('Slope o coefficient:', regr.coef_)
print('Intercept:', regr.intercept_)
```

```
Slope o coefficient: [[37.07110708]]
Intercept: [141.92245437]
```

Como se mencionó anteriormente, el coeficiente o pendiente (slope) y la intersección (intercept) son los coeficientes de la línea de ajuste. Dado que es una regresión lineal simple, son solo 2 coeficientes, sabiendo que los parámetros son la intersección y la pendiente en la ecuación, sklearn puede estimar directamente a partir de nuestros datos. Dado lo anterior, podemos gráficar la línea de ajuste usando los coeficientes.

[34]:
```
# gráficar el diagrama de dispersión
plt.scatter(train['Engine_size'], train['CO2_emissions(g/km)'],␣
 ↪color='steelblue')
# para el eje x usamos los datos de entrenamiento y para el eje y la ecuación␣
 ↪con los coeficientes para cada dato de x
plt.plot(X_train, regr.coef_[0][0] * X_train + regr.intercept_[0], '-r')
# agregar etiqueta al eje x
plt.xlabel('Engine size')
# agregar etiqueta al eje y
plt.ylabel('CO2 emissions')
# mostrar el gráfico
plt.show()
```



1.2.4 Evaluación del modelo

Se van a comparar los valores reales y los valores predichos para calcular la precisión del modelo de regresión. Las métricas de evaluación desempeñan un papel clave en el desarrollo de un modelo, ya que proporcionan información sobre los puntos que requieren mejora. Como ya se mencionó, existen diferentes medidas de evaluación del modelo, en este caso usaremos **MSE (Mean Squared Error) o error cuadrático medio** para calcular la precisión. Enseguida se enlista la métrica anterior con su definición general y otras métricas más:

- **Mean Absolute Error (MAE):** El error absoluto medio es la media del valor absoluto de los errores, es la métrica más fácil de entender, ya que es un error promedio.
- **Root Mean Squared Error (RMSE):** La raíz del error cuadrático medio es una medida común para evaluar las diferencias entre los valores predichos por un modelo y los valores observados, es decir, mide la dispersión de los errores de predicción alrededor de la línea de mejor ajuste. Cuanto menor sea el RMSE, mejor se ajusta el modelo a los datos.
- **Mean Squared Error (MSE):** El error cuadrático medio es la media del error cuadrado, esta medida es más popular que el MAE porque su enfoque está más orientado hacia los errores grandes, es decir, el termino cuadrado aumenta exponencialmente los errores más grandes en comparación con los más pequeños.
- **R-squared (R2):** El coeficiente de determinación o R cuadrado no es propiamente dicho un error, sino es una métrica popular para medir el rendimiento en el modelo de regresión. Representa la proporción de la variación en la variable dependiente que se puede predecir a partir de las variables independientes. Su valor oscila entre 0 (ninguna predicción) y 1 (predicción perfecta).

[35]:
```
# importamos los modulos de evaluación
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
# Se crean los conjuntos de prueba
X_test = np.asanyarray(test[['Engine_size']])
y_test = np.asanyarray(test[['CO2_emissions(g/km)']])
# y predicha
y_pred = regr.predict(X_test)
# se imprimen las métricas de evaluación de diferentes formas
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print('1 Mean Absolute Error:', round(np.mean(np.absolute(y_test - y_pred)), 2))
print('2 Mean Absolute Error:', round(mae, 2))
print("-------------------------------------------------------")
print('1 Mean Squared Error:', round(np.mean((y_pred - y_test) **2 ), 2))
print('2 Mean Squared Error:', round(mse, 2))
print("-------------------------------------------------------")
print('R-squared score:', round(r2_score(y_test, y_pred), 2))
```

```
1 Mean Absolute Error: 33.74
2 Mean Absolute Error: 33.74
-------------------------------------------------------
1 Mean Squared Error: 1950.44
2 Mean Squared Error: 1950.44
-------------------------------------------------------
R-squared score: 0.6
```

El MAE mide la magnitud promedio de los errores absolutos entre las predicciones y los valores reales. En este caso, el MAE es 30.83. Esto significa que, en promedio, las predicciones difieren en aproximadamente 30.83 unidades del valor real.

El MSE mide la dispersión de los errores al cuadrado entre las predicciones y los valores reales. En este caso, el MSE es 1760.6, cuanto mayor sea este valor, mayor será la dispersión de los errores, la opción es intentar con otra variable independiente y comparar los resultados, analizar y tomar decisiones.

El R-squared es una medida de cuánta variabilidad en los datos se explica por el modelo. Un valor de 0.57 indica que aproximadamente el 57% de la variabilidad en los datos se puede explicar mediante el modelo. Cuanto más cercano a 1, mejor es el ajuste..

1.3 Modelo de Regresión Lineal Múltiple

La regresión lineal múltiple es cuando se usa más de una variable independiente (x) para predecir a la variable dependiente (y), en el siguiente ejemplo se va a usar no solo la variable **Engine_size**(tamaño de motor) como variable predictora de **CO2_emissions(g/km)** (Emisiones de CO2), vamos a integrar como variables independientes las siguientes: **Cylinders** (número de cilindros), **Consumption_city(L/100km)** (Consumo en litros por cada 100 km en ciudad), **Consumption_highway(L/100km)** (Consumo en litros por cada 100 km en carretera) y **Consumption_combined(L/100km)** (Consumo en litros por cada 100 km combinado).

Puede decirse que la regresión lineal múltiple es una extensión del modelo de regresión lineal simple. El seleccionar las variables independientes adecuadas para un modelo de regresión lineal múltiple es fundamental para obtener resultados precisos. Aquí se muestran algunos métodos o consideraciones más comunes para hacer dicha selección:

Matriz de correlación: Calcula la correlación entre las variables independientes y la variable dependiente. Selecciona las variables con una correlación significativa.

Selección hacia adelante (Forward selection): Comienza con un modelo sin variables y agrega una a la vez. Evalúa el impacto de cada variable en el ajuste del modelo. Hay que detenerse cuando no se observen mejoras significativas.

Selección hacia atrás (Backward elimination): Comienza con todas las variables y se elimina una a la vez. Evalúa la reducción en el ajuste del modelo. Hay que detenerse cuando todas las variables sean significativas.

Selección bidireccional (Stepwise selection): Combina los métodos hacia adelante y hacia atrás. Hay que agregar o eliminar variables según su impacto en el modelo.

Validación cruzada: Dividir los datos en conjuntos de entrenamiento y prueba. Hay que evaluar el rendimiento del modelo con diferentes combinaciones de variables.

Es importante hacer mención que no existe un método único. La elección de las variables depende del contexto y los objetivos del análisis. Asimsimo, es importante hacer mención los conceptos heterocedasticidad, multicolinealidad y error de especificación, estos son indicadores en la solución de los problemas de residuos en el modelo. Si bien, el tratar a profundidad los conceptos anteriores rebasa el objetivo del presente trabajo, es importante ofrecer una descripción general de cada concepto:

- **Heterocedasticidad:** Si no es muy acusada no es importante. Si es muy acusada será necesario utilizar estimadores robustos. Los estimadores robustos no cambian ni el ajuste ni los parámetros β . Solo aumenta el error estándar estimado y, por tanto aumenta el p-valor por lo que corremos el riesgo de que una variable que fuese significativa deje de serlo. Los problemas de heterocedasticidad tambien pueden ser debidos a la omisión de alguna variable relevante por lo que el uso de estimadores robustos solo se recomienda cuando el tests Reset de Ramsey (estat ovtest) no sea significativo, es decir no indique que hay variables omitidas (2).

- **Multicolinealidad:** La colinealidad no sólo es normal sino que es esperable y deseable. Es imposible que unas variables que explican y son explicadas por un fenómeno sean tan completamente independientes que no estén correlacionadas en algún grado. El problema surge cuando hay, como mínimo, dos variables muy, muy, muy correlacionadas, entonces sucede que una de ellas le "roba" la correlación al resto haciendo que las demás aparezcan como no significativas o incluso significativas con un signo distinto al esperado. Esto es normal, por ejemplo en el caso de la renta, la edad y el nivel educativo. Lo que hay que hacer en estos casos es sacrificar una de ellas y quedarnos con la variable que tenga más sentido interpretativo (2).
- **Error de especificación:** El error de especificación se refiere a que falta por incluir alguna interacción o alguna variable en forma polinómica. El test consiste en regresar a la variable dependiente con potencias de ella misma por lo que las variables omitidas deben ser potencias o interacciones de las variables dependientes. Lamentablemente el test no nos ofrece pistas sobre las variables díscolas por lo que se impone utilizar la lógica y como último recurso, claro está, la prueba y error (2).

La ecuación del modelo de regresión lineal múltiple es la siguiente:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_k X_k + \varepsilon$$

Donde:

- β_0 **: es la intersección (ordenada al origen).**
- $\beta_1, \beta_2, ..., \beta_k$ **: son los coeficientes de las variables explicativas o predictoras.**
- ε **: es el término de error.**

Recordemos que para la estimación utilizamos técnicas como el **método de mínimos cuadrados** para estimar los coeficientes.

Inicio del ejercicio:

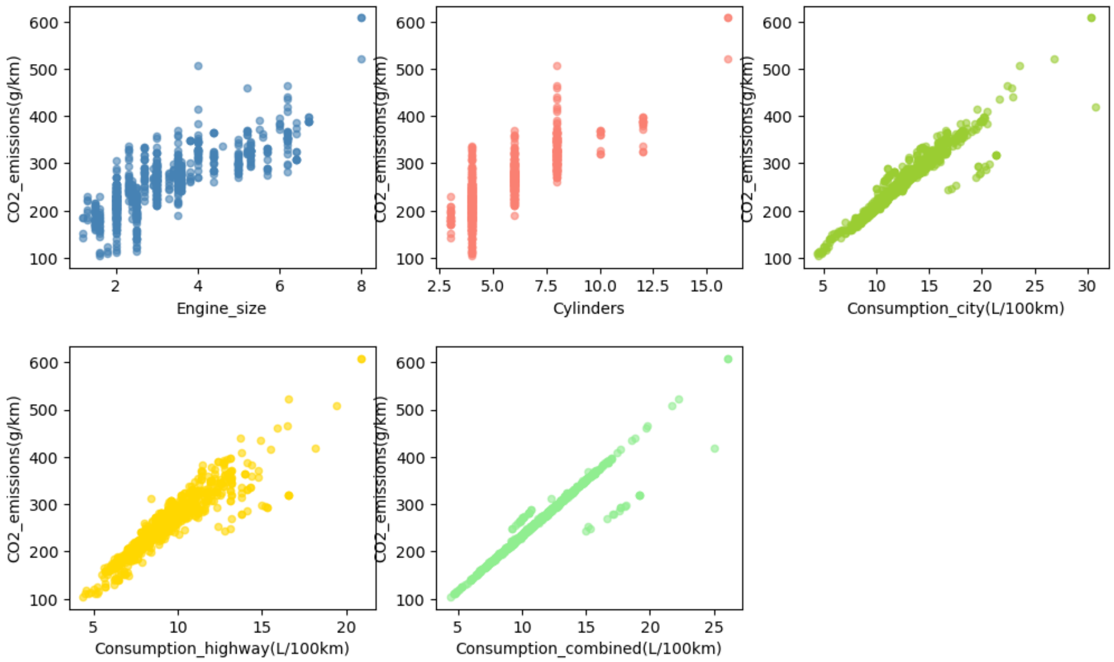
[36]:
```
# selección de los nuevas variables independientes, así como la variable␣
↪dependiente
data_2 = data[['Engine_size', 'Cylinders', 'Consumption_city(L/100km)',␣
↪'Consumption_highway(L/100km)', 'Consumption_combined(L/100km)',␣
↪'CO2_emissions(g/km)']]
# mostrar primeros 5 registros
data_2.head()
```

[36]:
```
   Engine_size  Cylinders  Consumption_city(L/100km)  \
0          1.5          4                        7.9
1          1.5          4                        8.1
2          1.5          4                        8.9
3          3.5          6                       12.6
4          3.0          6                       13.8
```

```
   Consumption_highway(L/100km)  Consumption_combined(L/100km)  \
0                           6.3                            7.2
1                           6.5                            7.4
2                           6.5                            7.8
3                           9.4                           11.2
4                          11.2                           12.4

   CO2_emissions(g/km)
0                  167
1                  172
2                  181
3                  263
4                  291
```

Lo siguente es realizar gráficos de dispersión de las variables independientes vs la variable dependiente, para esto empleamos el siguiente código.

[37]:
```python
fig = plt.figure(figsize=(12, 7))
plt.subplots_adjust(hspace=0.3) # tamaño del espacio hori¿zontal entre gráficos
# asignación de los subplots a las variables
ax0 = fig.add_subplot(2, 3, 1) # 2 filas, 3 columnas, subplot número 1
ax1 = fig.add_subplot(2, 3, 2) # 2 filas, 3 columnas, subplot número 2
ax2 = fig.add_subplot(2, 3, 3) # 2 filas, 3 columnas, subplot número 3
ax3 = fig.add_subplot(2, 3, 4) # 2 filas, 3 columnas, subplot número 4
ax4 = fig.add_subplot(2, 3, 5) # 2 filas, 3 columnas, subplot número 5
# gráficar cada diagrama de dispersión asignado a cada subplot
data_2.plot(kind='scatter', x='Engine_size', y='CO2_emissions(g/km)', ax=ax0,␣
 ↪color='steelblue', alpha=0.6)
data_2.plot(kind='scatter', x='Cylinders', y='CO2_emissions(g/km)', ax=ax1,␣
 ↪color='salmon', alpha=0.6)
data_2.plot(kind='scatter', x='Consumption_city(L/100km)', y='CO2_emissions(g/
 ↪km)', ax=ax2, color='yellowgreen', alpha=0.6)
data_2.plot(kind='scatter', x='Consumption_highway(L/100km)',␣
 ↪y='CO2_emissions(g/km)', ax=ax3, color='gold', alpha=0.6)
data_2.plot(kind='scatter', x='Consumption_combined(L/100km)',␣
 ↪y='CO2_emissions(g/km)', ax=ax4, color='lightgreen', alpha=0.6)

plt.show()
```



Lo siguiente será crear el conjunto de entrenamiento y prueba, a diferencia del ejemplo anterior, en esta ocasión nos apoyaremos del método **train_test_split()** de **sklearn** proveniente del modulo **model_selection**. Lo primero será segmentar el conjunto de datos en la variable **X** que corresponde a las variables independientes y en la variable **y** los datos de la variable dependiente.

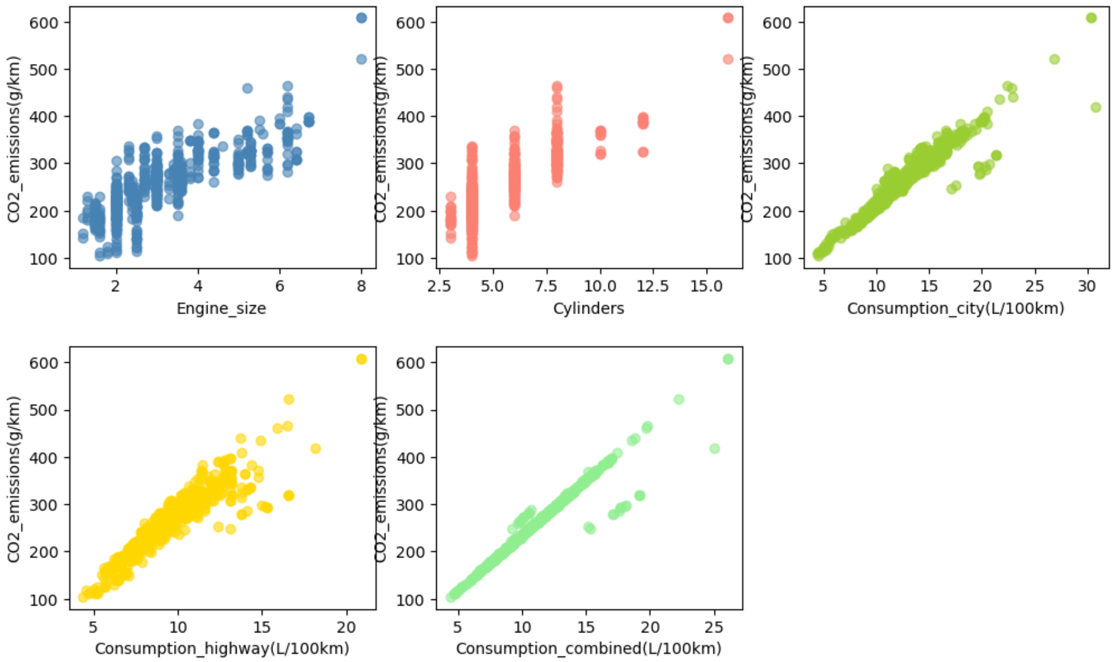
Lo primero será importar el método **train_test_split**, entre los parámetros que se pasan destacan **test_size=** el cuál define la proporción del conjunto de prueba, en este caso será del 20%, el siguiente es **random_state=** este se utiliza como semilla para generar números aleatorios a fin de asegurar que las divisiones sean reproducibles.

[38]:
```python
# importar el método
from sklearn.model_selection import train_test_split

# dividir el conjunto de datos en X e y
X = data_2[['Engine_size', 'Cylinders', 'Consumption_city(L/100km)', 'Consumption_highway(L/100km)', 'Consumption_combined(L/100km)']]
y = data_2['CO2_emissions(g/km)']

# crear los conjuntos de datos de entrenamiento y test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Volvamos a gráficar las variables independientes vs la variable dependiente de los conjuntos de entrenamiento para observar si sigue presentando la misma distribución.

[39]:

```
fig = plt.figure(figsize=(12, 7))
plt.subplots_adjust(hspace=0.3) # tamaño del espacio hori¿zontal entre gráficos
# asignación de los subplots a las variables
ax0 = fig.add_subplot(2, 3, 1) # 2 filas, 3 columnas, subplot número 1
ax1 = fig.add_subplot(2, 3, 2) # 2 filas, 3 columnas, subplot número 2
ax2 = fig.add_subplot(2, 3, 3) # 2 filas, 3 columnas, subplot número 3
ax3 = fig.add_subplot(2, 3, 4) # 2 filas, 3 columnas, subplot número 4
ax4 = fig.add_subplot(2, 3, 5) # 2 filas, 3 columnas, subplot número 5
# gráficos de distribución para cada subplot
ax0.scatter(X_train.Engine_size, y_train.values, color='steelblue', alpha=0.6)
ax0.set_ylabel('CO2_emissions(g/km)', size=10)
ax0.set_xlabel('Engine_size', size=10)

ax1.scatter(X_train.Cylinders, y_train.values, color='salmon', alpha=0.6)
ax1.set_ylabel('CO2_emissions(g/km)', size=10)
ax1.set_xlabel('Cylinders', size=10)

ax2.scatter(X_train['Consumption_city(L/100km)'], y_train.values,␣
↪color='yellowgreen', alpha=0.6)
ax2.set_ylabel('CO2_emissions(g/km)', size=10)
ax2.set_xlabel('Consumption_city(L/100km)', size=10)

ax3.scatter(X_train['Consumption_highway(L/100km)'], y_train.values,␣
↪color='gold', alpha=0.6)
ax3.set_ylabel('CO2_emissions(g/km)', size=10)
ax3.set_xlabel('Consumption_highway(L/100km)', size=10)

ax4.scatter(X_train['Consumption_combined(L/100km)'], y_train.values,␣
↪color='lightgreen', alpha=0.6)
ax4.set_ylabel('CO2_emissions(g/km)', size=10)
ax4.set_xlabel('Consumption_combined(L/100km)', size=10)

plt.show()
```



Podemos observar que la distribución se mantiene símil; lo siguiente será crear o instanciar el modelo de regresión lineal múltiple, ajustar el modelo e imprimir los coeficientes.

[40]:
```
# instanciar el modelo de regresión
regr_mult = linear_model.LinearRegression()

# ajustar el modelo
regr_mult.fit(X_train, y_train)

#imprimir los coeficientes
print('Intercept:', regr_mult.intercept_)
print('Coefficients:', regr_mult.coef_)
```

```
Intercept: 20.831492887169247
Coefficients: [-1.01776422  4.62860573  4.30162187  4.97019535 10.2684636 ]
```

Ahora toca realizar la predicción con los datos de prueba y obtener las diferentes métricas de evaluación.

[41]:
```
# Realiza predicciones en el conjunto de prueba
y_pred = regr_mult.predict(X_test)

# se obtienen e imprimen las métricas de evaluación
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

```
#print('1 Mean Absolute Error:', round(np.mean(np.absolute(y_test - y_pred)),␣
 ↪2))
print('Mean Absolute Error(MAE):', round(mae, 2))
print("---------------------------------------------------")
#print('1 Mean Squared Error:', round(np.mean((y_pred - y_test) **2 ), 2))
print('Mean Squared Error(MSE):', round(mse, 2))
print("---------------------------------------------------")
print('R-squared score:', round(r2_score(y_test, y_pred), 2))
```

```
Mean Absolute Error(MAE): 8.09
------------------------------------------------------
Mean Squared Error(MSE): 262.28
------------------------------------------------------
R-squared score: 0.93
```

Recordemos las definiciones de las métricas de evaluación:

- **Mean Absolute Error (MAE):** El error absoluto medio es la media del valor absoluto de los errores, es la métrica más fácil de entender, ya que es un error promedio.
- **Root Mean Squared Error (RMSE):** La raíz del error cuadrático medio es una medida común para evaluar las diferencias entre los valores predichos por un modelo y los valores observados, es decir, mide la dispersión de los errores de predicción alrededor de la línea de mejor ajuste. Cuanto menor sea el RMSE, mejor se ajusta el modelo a los datos.
- **Mean Squared Error (MSE):** El error cuadrático medio es la media del error cuadrado, esta medida es más popular que el MAE porque su enfoque está más orientado hacia los errores grandes, es decir, el termino cuadrado aumenta exponencialmente los errores más grandes en comparación con los más pequeños.
- **R-squared (R2):** El coeficiente de determinación o R cuadrado no es propiamente dicho un error, sino es una métrica popular para medir el rendimiento en el modelo de regresión. Representa la proporción de la variación en la variable dependiente que se puede predecir a partir de las variables independientes. Su valor oscila entre 0 (ninguna predicción) y 1 (predicción perfecta).

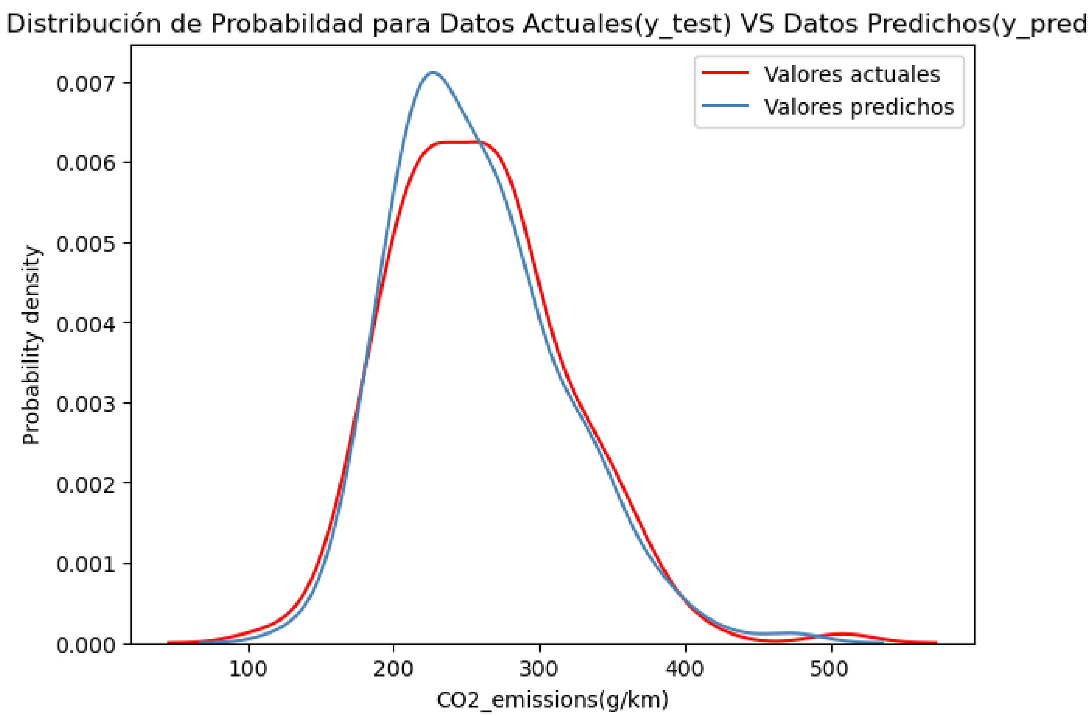
De acuerdo con lo anterior, parece que nuestro modelo es bastante aceptable, pues presenta un valor bajo en el MSE respecto al primer modelo, asimismo, el R2 es muy cercano a 1, pero de igual manera podemos estar probando con diferentes variables indepenedientes para constatar si existe alguna mejora en nuestro modelo a través de éstas métricas.

Una forma visual de hacer una comparación es imprimir las distribuciones de probabilidad de los datos de y de prueba y los datos predichos $\hat{y}$, la siguiente función nos sirve para esto:

[42]:
```
# importar aeaborn
import seaborn as sns

# construir la función
def DistribucionProb(redProb, blueProb, redName, blueName, title):
    plt.figure(figsize=(7, 5))
    # graficos de la estimación de densidad (KDE) y sus parámetros
    ax0 = sns.kdeplot(redProb, color='red', label=redName)
```

```
    ax1 = sns.kdeplot(blueProb, color='steelblue', label=blueName)
    plt.title(title)
    plt.xlabel("CO2_emissions(g/km)")
    plt.ylabel('Probability density')
    plt.legend()

    plt.show()
```

[43]:
```
title = 'Distribución de Probabildad para Datos Actuales(y_test) VS Datos␣
 ↪Predichos(y_pred)'
DistribucionProb(y_test, y_pred, "Valores actuales", "Valores predichos", title)
```



Podemos observar que las distribuciones sugieren que el modelo predictivo utilizado es bastante preciso, ya que las predicciones se alinean bien con los valores observados. Sin embargo, podemos observar un "pico" en el rango aproximado de 170 a 270, esto puede deberse a varias razones, algunas de estas se enlistan a continuación:

Características de los datos: Puede haber una concentración de observaciones en ese rango específico debido a características particulares de los datos, como estamos prediciendo las emisiones de CO2 de vehículos, podría haber una categoría de automóviles que tiende a tener valores en ese intervalo.

Modelo no linealidad: El modelo de regresión lineal múltiple asume una relación lineal entre las variables independientes y la variable dependiente. Si hay una relación no lineal (por ejemplo, una curva), el modelo puede no capturar adecuadamente esa variación en ciertos rangos.

Interacciones entre variables: Las interacciones entre las variables independientes pueden afectar la predicción. Si hay una interacción significativa entre dos o más variables, podría generar un pico en la distribución.

Valores atípicos: Valores atípicos o extremos pueden influir en la predicción en un rango específico. Si hay observaciones inusuales en ese intervalo, podrían afectar la forma de la distribución.

Sin duda, parece que el modelo puede mejorarse, pero para un primer acercamiento en la aplicación de este algoritmo de Machine Learning no está mal.

Elaborado por: **Gabriel Armando Landín Alvarado**

Para descargar el código y los datos puedes visitar el siguiente repositorio de GitHub: /LandinGabriel13