Simple Linear Regression

October 9, 2024

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

[2]:
```
!curl https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
↪IBMDeveloperSkillsNetwork-ML0101EN-SkillsNetwork/labs/Module%202/data/
↪FuelConsumptionCo2.csv -o FuelConsumptionCo2.csv
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed

  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
  0     0    0     0    0     0      0      0 --:--:--  0:00:01 --:--:--     0
 67 72629   67 48699    0     0  20057      0  0:00:03  0:00:02  0:00:01 20090
100 72629  100 72629    0     0  28893      0  0:00:02  0:00:02 --:--:-- 28935
```

[5]:
```
#Reading data
df = pd.read_csv("FuelConsumptionCo2.csv")

# take a look at the dataset
df.head()
```

[5]:

	MODELYEAR	MAKE	MODEL	VEHICLECLASS	ENGINESIZE	CYLINDERS
0	2014	ACURA	ILX	COMPACT	2.0	4
1	2014	ACURA	ILX	COMPACT	2.4	4
2	2014	ACURA	ILX HYBRID	COMPACT	1.5	4
3	2014	ACURA	MDX 4WD	SUV - SMALL	3.5	6
4	2014	ACURA	RDX AWD	SUV - SMALL	3.5	6

	TRANSMISSION	FUELTYPE	FUELCONSUMPTION_CITY	FUELCONSUMPTION_HWY
0	AS5	Z	9.9	6.7
1	M6	Z	11.2	7.7
2	AV7	Z	6.0	5.8
3	AS6	Z	12.7	9.1
4	AS6	Z	12.1	8.7

	FUELCONSUMPTION_COMB	FUELCONSUMPTION_COMB_MPG	CO2EMISSIONS

0	8.5	33	196
1	9.6	29	221
2	5.9	48	136
3	11.1	25	255
4	10.6	27	244

Data Exploration

[6]:
```
# summarize the data
df.describe()
```

[6]:

	MODELYEAR	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_CITY	\
count	1067.0	1067.000000	1067.000000	1067.000000	
mean	2014.0	3.346298	5.794752	13.296532	
std	0.0	1.415895	1.797447	4.101253	
min	2014.0	1.000000	3.000000	4.600000	
25%	2014.0	2.000000	4.000000	10.250000	
50%	2014.0	3.400000	6.000000	12.600000	
75%	2014.0	4.300000	8.000000	15.550000	
max	2014.0	8.400000	12.000000	30.200000	

	FUELCONSUMPTION_HWY	FUELCONSUMPTION_COMB	FUELCONSUMPTION_COMB_MPG	\
count	1067.000000	1067.000000	1067.000000	
mean	9.474602	11.580881	26.441425	
std	2.794510	3.485595	7.468702	
min	4.900000	4.700000	11.000000	
25%	7.500000	9.000000	21.000000	
50%	8.800000	10.900000	26.000000	
75%	10.850000	13.350000	31.000000	
max	20.500000	25.800000	60.000000	

	CO2EMISSIONS
count	1067.000000
mean	256.228679
std	63.372304
min	108.000000
25%	207.000000
50%	251.000000
75%	294.000000
max	488.000000

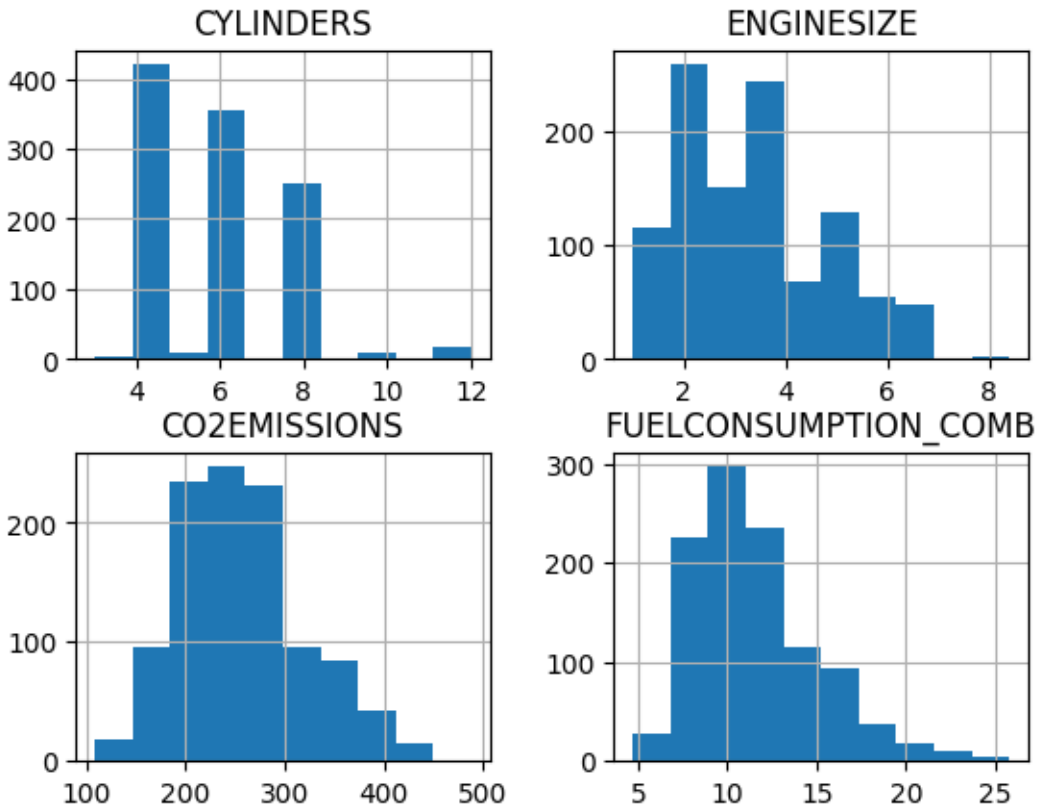
[7]:
```
cdf = df[['ENGINESIZE','CYLINDERS','FUELCONSUMPTION_COMB','CO2EMISSIONS']]
cdf.head(9)
```

[7]:

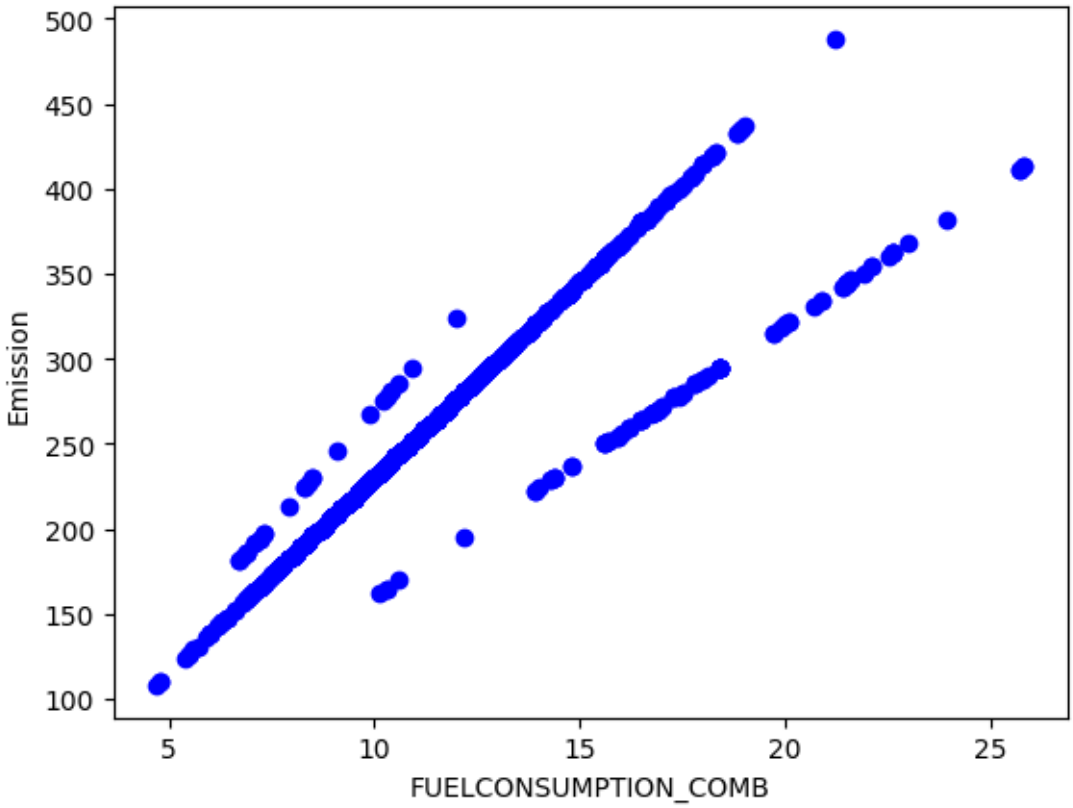
	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSIONS
0	2.0	4	8.5	196
1	2.4	4	9.6	221
2	1.5	4	5.9	136

3	3.5	6	11.1	255
4	3.5	6	10.6	244
5	3.5	6	10.0	230
6	3.5	6	10.1	232
7	3.7	6	11.1	255
8	3.7	6	11.6	267

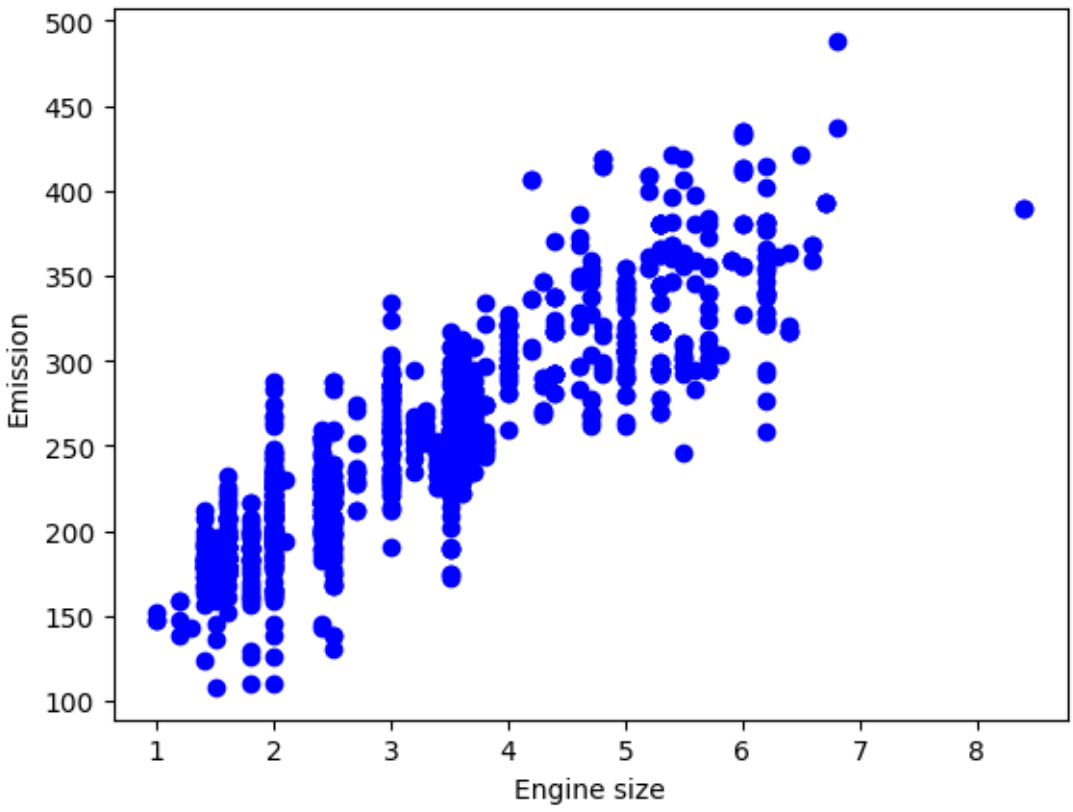
[8]:
```
viz = cdf[['CYLINDERS','ENGINESIZE','CO2EMISSIONS','FUELCONSUMPTION_COMB']]
viz.hist()
plt.show()
```



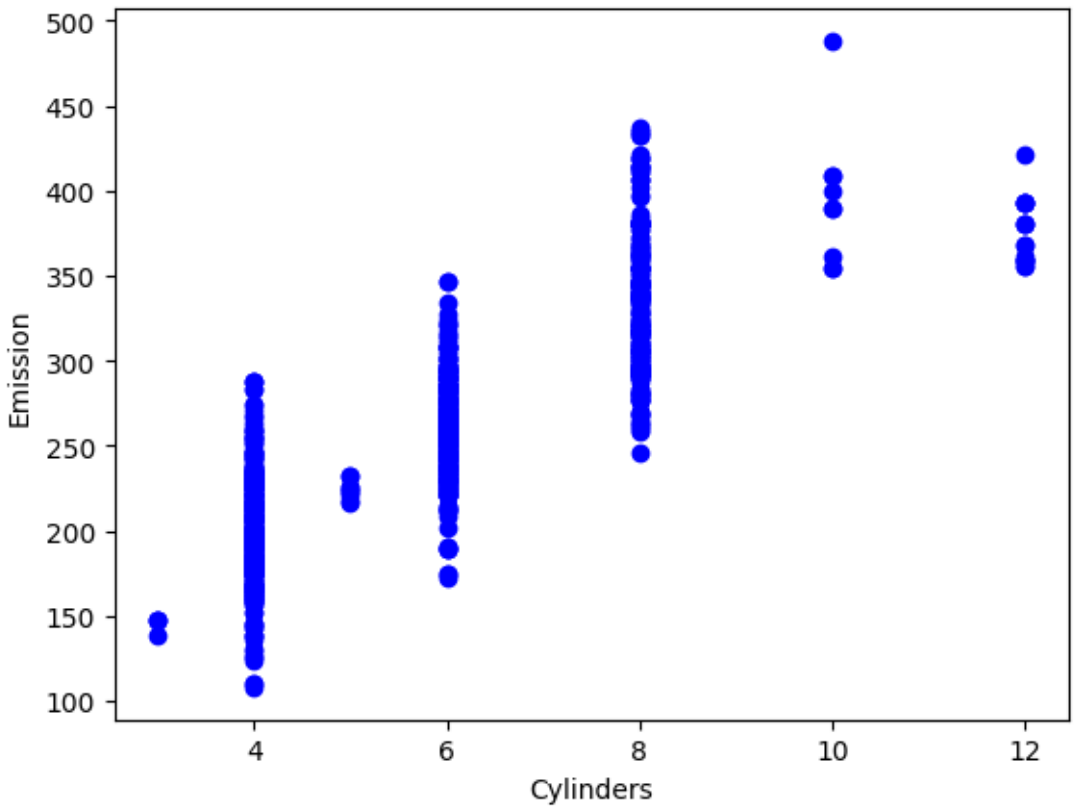
[9]:
```
plt.scatter(cdf.FUELCONSUMPTION_COMB, cdf.CO2EMISSIONS,  color='blue')
plt.xlabel("FUELCONSUMPTION_COMB")
plt.ylabel("Emission")
plt.show()
```



[10]:
```
plt.scatter(cdf.ENGINESIZE, cdf.CO2EMISSIONS,  color='blue')
plt.xlabel("Engine size")
plt.ylabel("Emission")
plt.show()
```



[12]:
```
plt.scatter(cdf.CYLINDERS, cdf.CO2EMISSIONS,  color='blue')
plt.xlabel("Cylinders")
plt.ylabel("Emission")
plt.show()
```



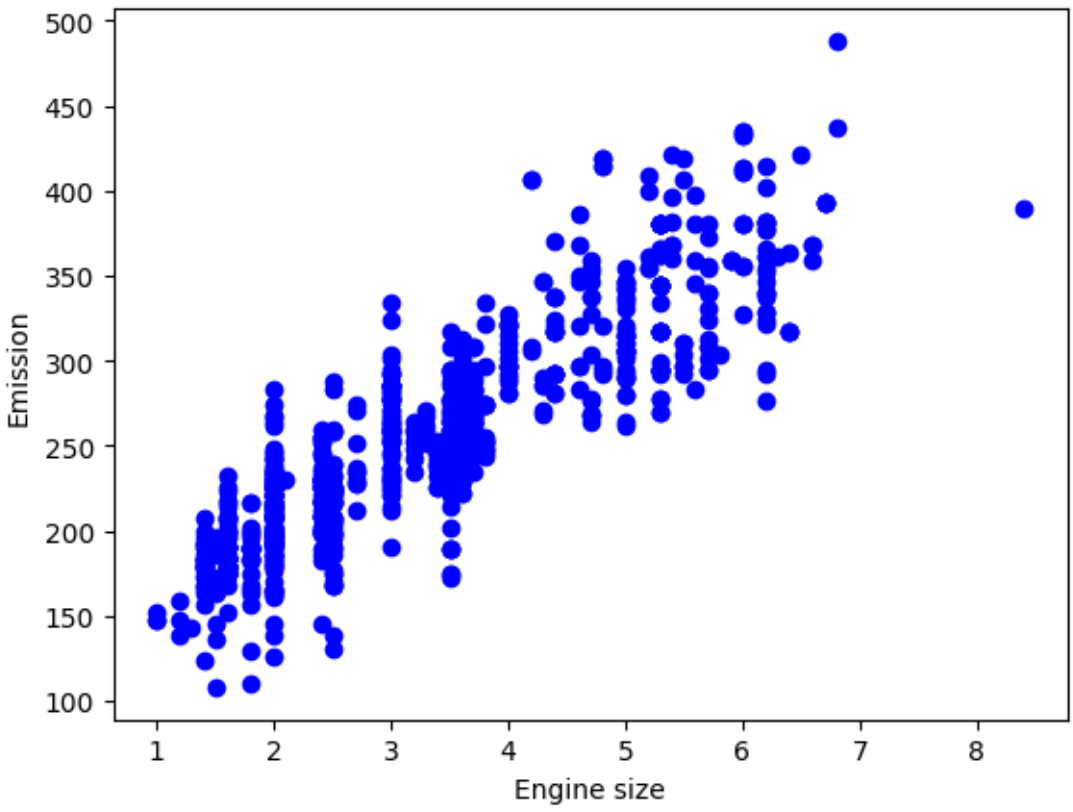
ceating Train and Test dataset

[13]:
```
#Let's split our dataset into train and test sets. 80% of the entire dataset␣
 ↪will be used for training and 20% for testing.
#We create a mask to select random rows using __np.random.rand()__ function:

msk = np.random.rand(len(df)) < 0.8
train = cdf[msk]
test = cdf[~msk]
```

Train data distribution

[14]:
```
plt.scatter(train.ENGINESIZE, train.CO2EMISSIONS,  color='blue')
plt.xlabel("Engine size")
plt.ylabel("Emission")
plt.show()
```



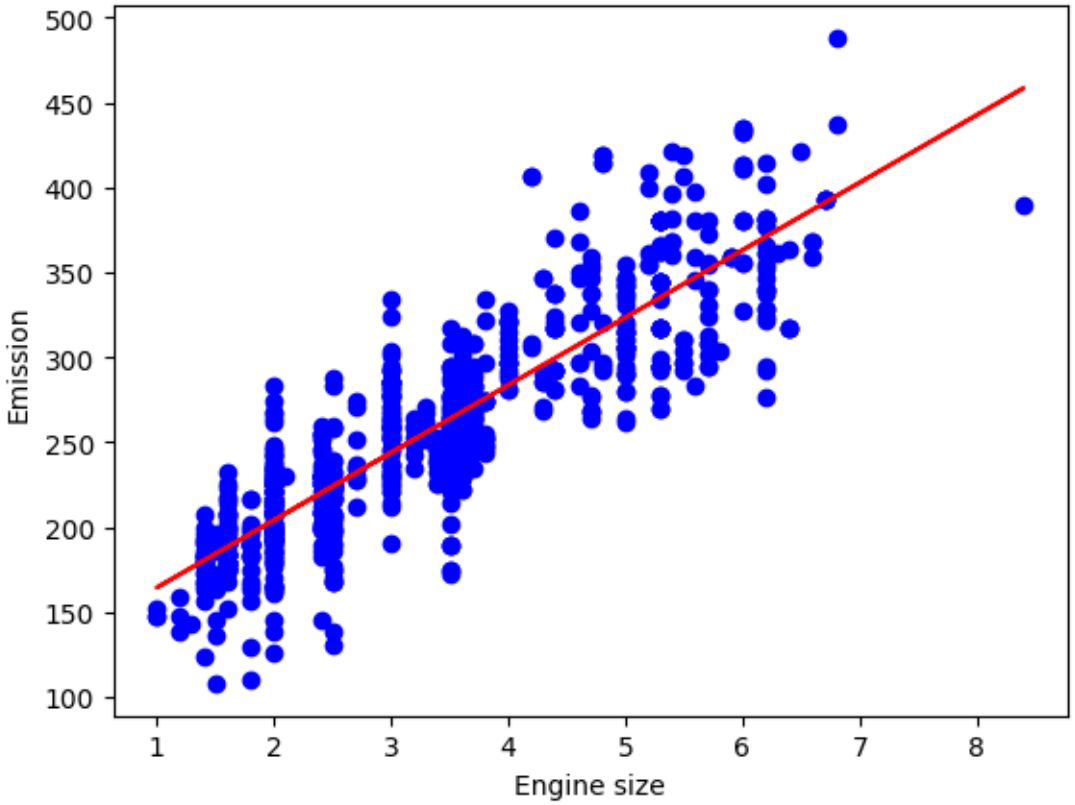
Modeling

[20]:
```
from sklearn import linear_model
regr = linear_model.LinearRegression()
train_x = np.asanyarray(train[['ENGINESIZE']])
train_y = np.asanyarray(train[['CO2EMISSIONS']])
regr.fit(train_x, train_y)
# The coefficients
print ('Coefficients: ', regr.coef_) #This returns the slope of the linear␣
 ↪regression line (i.e., how much CO2EMISSIONS is expected to change with a␣
 ↪unit change in ENGINESIZE
print ('Intercept: ',regr.intercept_) #This returns the intercept of the line␣
 ↪(i.e., the value of CO2EMISSIONS when ENGINESIZE is 0)
```

```
Coefficients:  [[39.74821026]]
Intercept:  [124.52962794]
```

[21]:
```
plt.scatter(train.ENGINESIZE, train.CO2EMISSIONS,  color='blue')
plt.plot(train_x, regr.coef_[0][0]*train_x + regr.intercept_[0], '-r')
plt.xlabel("Engine size")
plt.ylabel("Emission")
```

[21]: Text(0, 0.5, 'Emission')



[24]:
```
from sklearn.metrics import r2_score

test_x = np.asanyarray(test[['ENGINESIZE']])
test_y = np.asanyarray(test[['CO2EMISSIONS']])
test_y_ = regr.predict(test_x)

print("Mean Absolute Error(MAE): %.2f" % np.mean(np.absolute(test_y_ - test_y)))
print("Mean Squared Error(MSE): %.2f" % np.mean((test_y_ - test_y) ** 2))
print("R2-score: %.2f" % r2_score(test_y , test_y_) )
```

```
Mean Absolute Error(MAE): 23.66
Mean Squared Error(MSE): 971.26
R2-score: 0.72
```

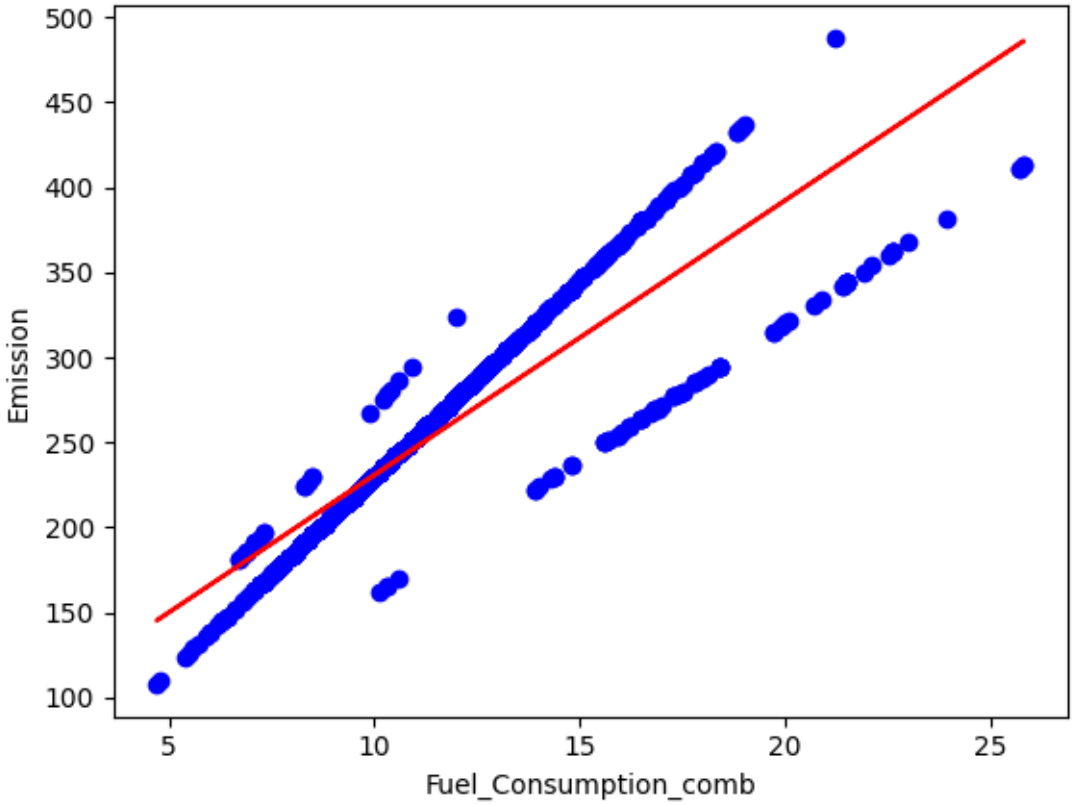
[25]:
```
train_x = np.asanyarray(train[['FUELCONSUMPTION_COMB']])
train_y = np.asanyarray(train[['CO2EMISSIONS']])
regr.fit(train_x, train_y)
# The coefficients
```

```
print ('Coefficients: ', regr.coef_) #This returns the slope of the linear␣
 ↪regression line (i.e., how much CO2EMISSIONS is expected to change with a␣
 ↪unit change in ENGINESIZE
print ('Intercept: ',regr.intercept_) #This returns the intercept of the line␣
 ↪(i.e., the value of CO2EMISSIONS when ENGINESIZE is 0)
```

```
Coefficients:  [[16.15010364]]
Intercept:  [69.3437022]
```

[26]:
```
plt.scatter(train.FUELCONSUMPTION_COMB, train.CO2EMISSIONS,  color='blue')
plt.plot(train_x, regr.coef_[0][0]*train_x + regr.intercept_[0], '-r')
plt.xlabel("Fuel_Consumption_comb")
plt.ylabel("Emission")
```

[26]:
```
Text(0, 0.5, 'Emission')
```



[27]:
```
from sklearn.metrics import r2_score

test_x = np.asanyarray(test[['FUELCONSUMPTION_COMB']])
test_y = np.asanyarray(test[['CO2EMISSIONS']])
test_y_ = regr.predict(test_x)
```

```
print("Mean Absolute Error(MAE): %.2f" % np.mean(np.absolute(test_y_ - test_y)))
print("Mean Squared Error(MSE): %.2f" % np.mean((test_y_ - test_y) ** 2))
print("R2-score: %.2f" % r2_score(test_y , test_y_) )
```

```
Mean Absolute Error(MAE): 18.28
Mean Squared Error(MSE): 643.39
R2-score: 0.81
```

[ ]: