Unit04 demo correlation regression fuel consumption.ipynb

July 27, 2023

0.1 Importing the required packages

[1]:
```
import matplotlib.pyplot as plt
import pandas as pd
import pylab as pl
import numpy as np
import seaborn as sns
```

0.2 Load the data

[2]:
```
# Reading the data
df=pd.read_csv("FuelConsumption.csv")
```

[3]:
```
# Take a look at the dataset
df.head(10)
```

[3]:
```
   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
5       2014  ACURA         RLX     MID-SIZE         3.5          6
6       2014  ACURA          TL     MID-SIZE         3.5          6
7       2014  ACURA      TL AWD     MID-SIZE         3.7          6
8       2014  ACURA      TL AWD     MID-SIZE         3.7          6
9       2014  ACURA         TSX      COMPACT         2.4          4

  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
5          AS6        Z                  11.9                  7.7
6          AS6        Z                  11.8                  8.1
7          AS6        Z                  12.8                  9.0
8           M6        Z                  13.4                  9.5
```

```
9          AS5        Z                  10.6                 7.5

   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
5                  10.0                        28           230
6                  10.1                        28           232
7                  11.1                        25           255
8                  11.6                        24           267
9                   9.2                        31           212
```

[ ]:

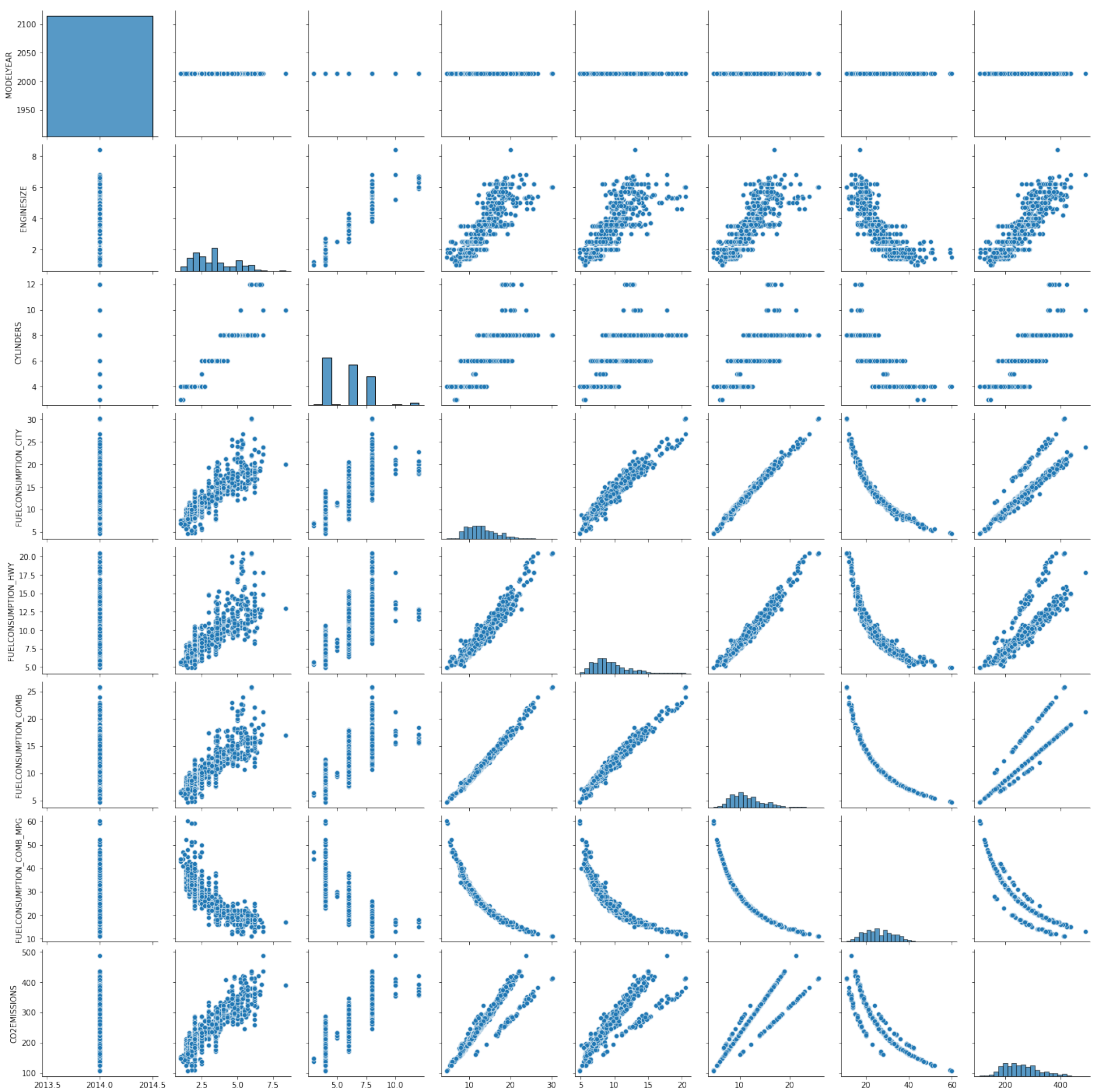
[5]:
```
df.corr()
```

[5]:
```
                          MODELYEAR  ENGINESIZE  CYLINDERS  \
MODELYEAR                       NaN         NaN        NaN
ENGINESIZE                      NaN    1.000000   0.934011
CYLINDERS                       NaN    0.934011   1.000000
FUELCONSUMPTION_CITY            NaN    0.832225   0.796473
FUELCONSUMPTION_HWY             NaN    0.778746   0.724594
FUELCONSUMPTION_COMB            NaN    0.819482   0.776788
FUELCONSUMPTION_COMB_MPG        NaN   -0.808554  -0.770430
CO2EMISSIONS                    NaN    0.874154   0.849685

                          FUELCONSUMPTION_CITY  FUELCONSUMPTION_HWY  \
MODELYEAR                                  NaN                  NaN
ENGINESIZE                            0.832225             0.778746
CYLINDERS                             0.796473             0.724594
FUELCONSUMPTION_CITY                  1.000000             0.965718
FUELCONSUMPTION_HWY                   0.965718             1.000000
FUELCONSUMPTION_COMB                  0.995542             0.985804
FUELCONSUMPTION_COMB_MPG             -0.935613            -0.893809
CO2EMISSIONS                          0.898039             0.861748

                          FUELCONSUMPTION_COMB  FUELCONSUMPTION_COMB_MPG  \
MODELYEAR                                  NaN                       NaN
ENGINESIZE                            0.819482                 -0.808554
CYLINDERS                             0.776788                 -0.770430
FUELCONSUMPTION_CITY                  0.995542                 -0.935613
FUELCONSUMPTION_HWY                   0.985804                 -0.893809
FUELCONSUMPTION_COMB                  1.000000                 -0.927965
FUELCONSUMPTION_COMB_MPG             -0.927965                  1.000000
CO2EMISSIONS                          0.892129                 -0.906394
```

```
                          CO2EMISSIONS
MODELYEAR                          NaN
ENGINESIZE                    0.874154
CYLINDERS                     0.849685
FUELCONSUMPTION_CITY          0.898039
FUELCONSUMPTION_HWY           0.861748
FUELCONSUMPTION_COMB          0.892129
FUELCONSUMPTION_COMB_MPG     -0.906394
CO2EMISSIONS                  1.000000
```

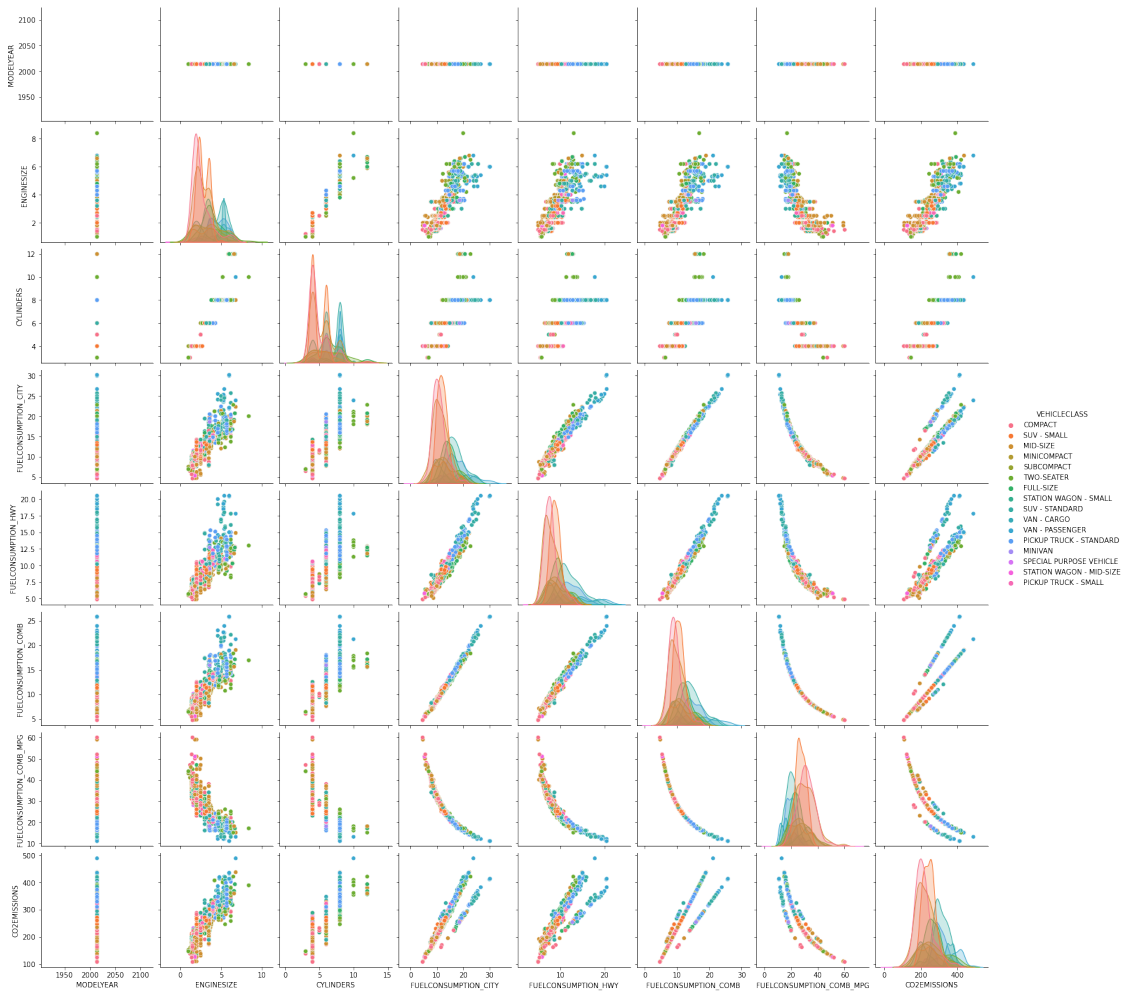
[8]:
```python
sns.pairplot(df)
```

[8]: <seaborn.axisgrid.PairGrid at 0x172deed7c10>



[9]:
```
sns.pairplot(df, hue = "VEHICLECLASS")
```

[9]: <seaborn.axisgrid.PairGrid at 0x172e4900e80>



[14]:
```
plt.figure(figsize=(12, 6))
heatmap = sns.heatmap(df.corr(), vmin=-1, vmax=1, annot=True, cmap='BrBG')
heatmap.set_title('Correlation Heatmap', fontdict={'fontsize':12}, pad=12)
```

[14]: Text(0.5, 1.0, 'Correlation Heatmap')

0.3 Plot to check the linearity

0.4 Data exploration

[5]:
```python
# Summarise the data
print(df.describe())
cdf=df[['ENGINESIZE','CYLINDERS','FUELCONSUMPTION_COMB','CO2EMISSIONS']]
cdf.head(9)
```

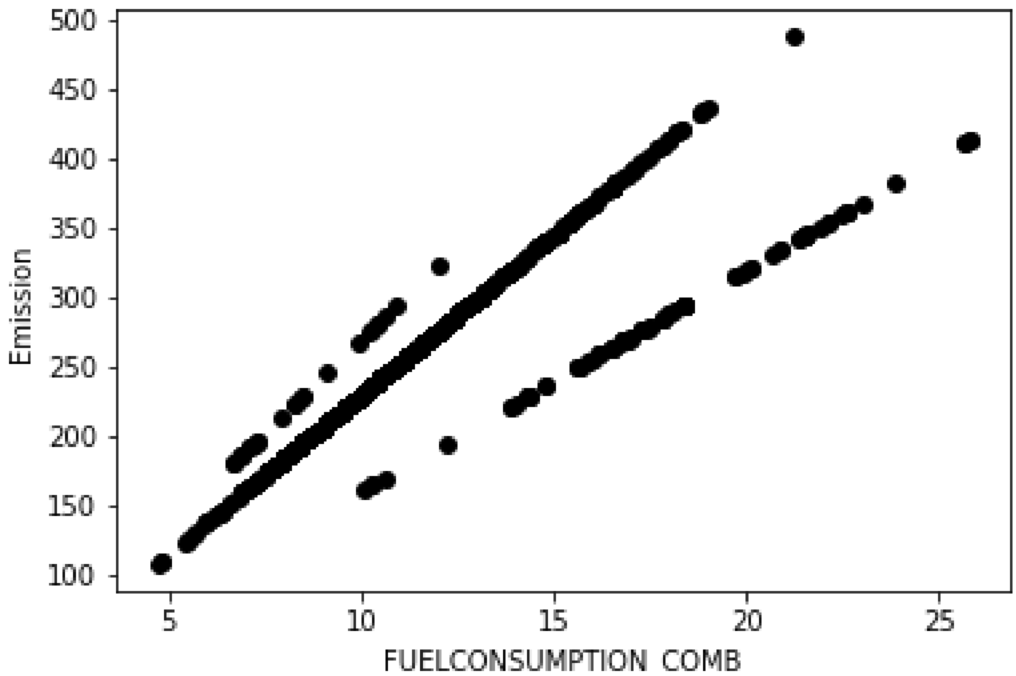
```
       MODELYEAR   ENGINESIZE    CYLINDERS  FUELCONSUMPTION_CITY  \
count     1067.0  1067.000000  1067.000000           1067.000000
mean      2014.0     3.346298     5.794752             13.296532
std          0.0     1.415895     1.797447              4.101253
min       2014.0     1.000000     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
```

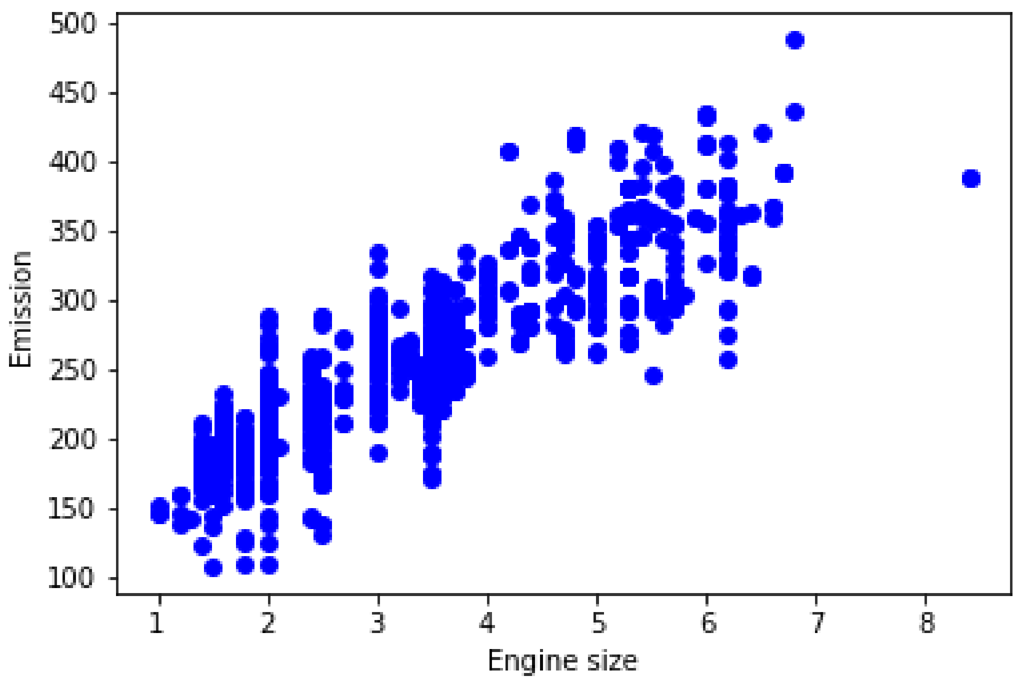
[5]:

	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

[6]:
```python
plt.scatter(cdf.FUELCONSUMPTION_COMB,cdf.CO2EMISSIONS, color='black')
plt.xlabel("FUELCONSUMPTION_COMB")
plt.ylabel("Emission")
plt.show()
```

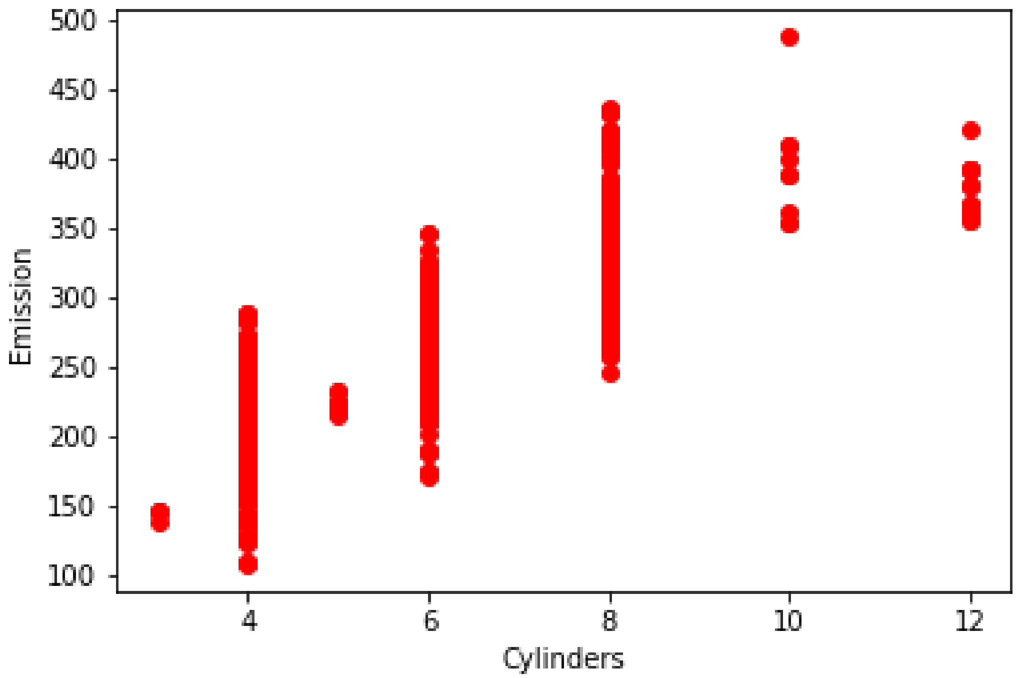


[7]:
```
plt.scatter(cdf.ENGINESIZE,cdf.CO2EMISSIONS, color='blue')
plt.xlabel("Engine size")
plt.ylabel("Emission")
plt.show()
```



[8]:

```
plt.scatter(cdf.CYLINDERS,cdf.CO2EMISSIONS, color='red')
plt.xlabel("Cylinders")
plt.ylabel("Emission")
plt.show()
```



Which of the above variables do you think will work best to explain a linear relation with CO2 emission?

0.5 Demo 2. Regression

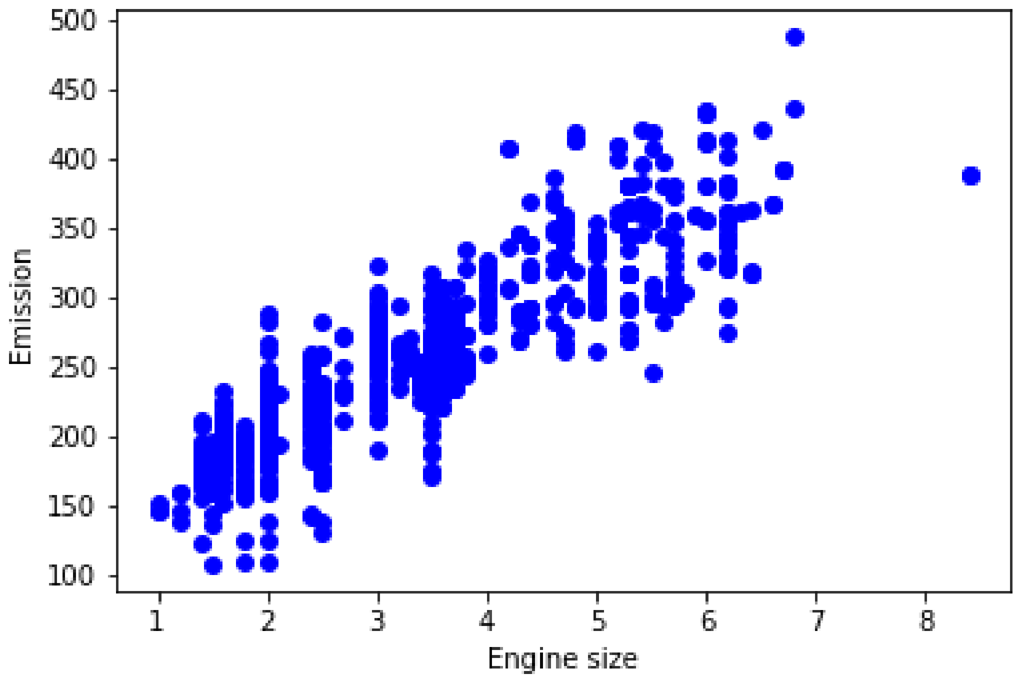
0.6 Train-test data preparation

[9]:

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

[10]:

```
# Train data distribution
plt.scatter(train.ENGINESIZE,train.CO2EMISSIONS, color='blue')
plt.xlabel("Engine size")
plt.ylabel("Emission")
plt.show()
```



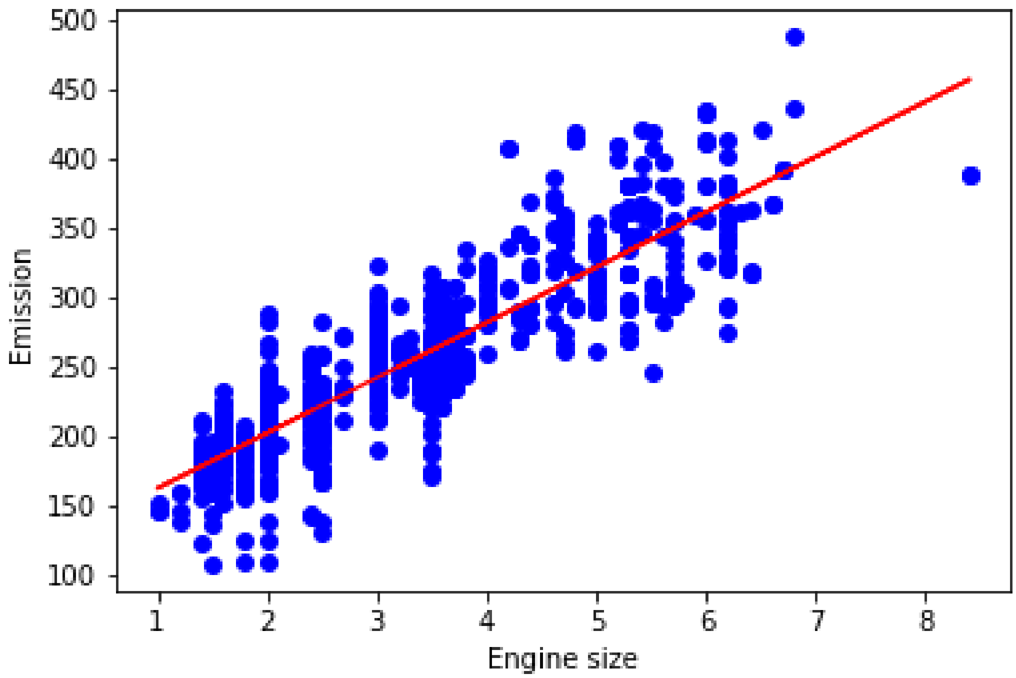
0.7 Using sklearn package for data modelling

[12]:
```
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_)
print('Intercept:', regr.intercept_)
```

```
Coefficients: [[39.79714822]]
Intercept: [122.88558355]
```

[13]:
```
# Plot outputs
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")
```

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



0.8 Model evaluation

[18]:
```
from sklearn.metrics import r2_score
test_x=np.asanyarray(test[['ENGINESIZE']])
test_y=np.asanyarray(test[['CO2EMISSIONS']])
test_y_ = regr.predict(test_y)
```

[19]:
```
print("Mean absolute error: %.2f" % np.mean(np.absolute(test_y_-test_y)))
print("Residual sum of squares (MSE): %.2f" % np.mean((test_y_-test_y)**2))
print("R2-score: %.2f" % r2_score(test_y_,test_y))
```

```
Mean absolute error: 9933.38
Residual sum of squares (MSE): 103641940.32
R2-score: -18.82
```

[ ]:
```

```

0.9 Nonlinear regression

Importing reqired dataset

[20]:
```
df=pd.read_csv("china_gdp.csv")
df.head(10)
```

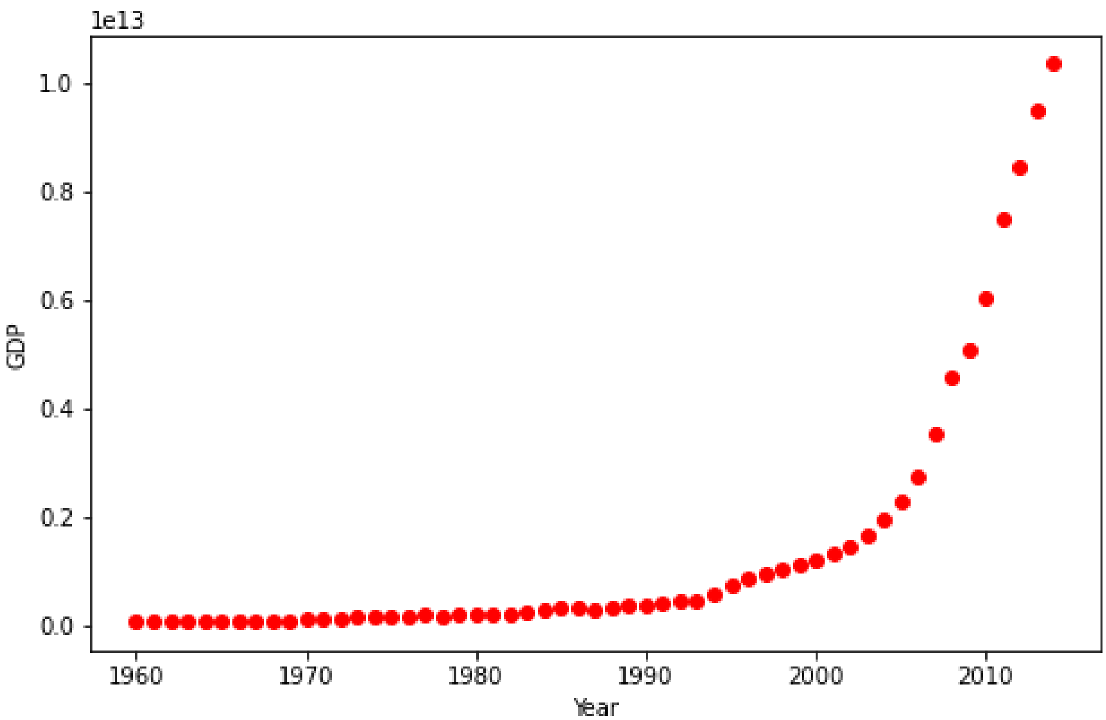
[20]:

	Year	Value
0	1960	5.918412e+10

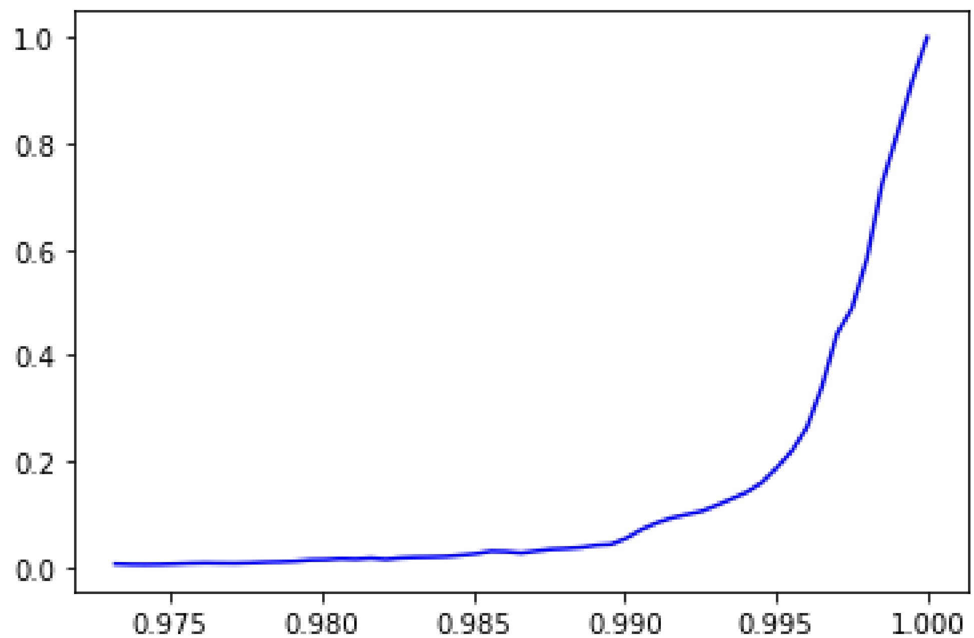
```
1  1961  4.955705e+10
2  1962  4.668518e+10
3  1963  5.009730e+10
4  1964  5.906225e+10
5  1965  6.970915e+10
6  1966  7.587943e+10
7  1967  7.205703e+10
8  1968  6.999350e+10
9  1969  7.871882e+10
```

Plotting the dataset

[21]:
```
plt.figure(figsize=(8,5))
x_data,y_data=(df["Year"].values,df["Value"].values)
plt.plot(x_data,y_data,'ro')
plt.ylabel('GDP')
plt.xlabel('Year')
plt.show()
# Normalisation
xdata=x_data/max(x_data)
ydata=y_data/max(y_data)

plt.plot(xdata,ydata,'b')
```



[21]: [<matplotlib.lines.Line2D at 0x199cce19a60>]



Roughly looking look at the data visualisation, it appears that the logistic function could be a good representation for this very dataset. The logistic function has the property of starting with a slow growth, increasing growth in the middle, and then decreasing again at the end

Implement the logistic function

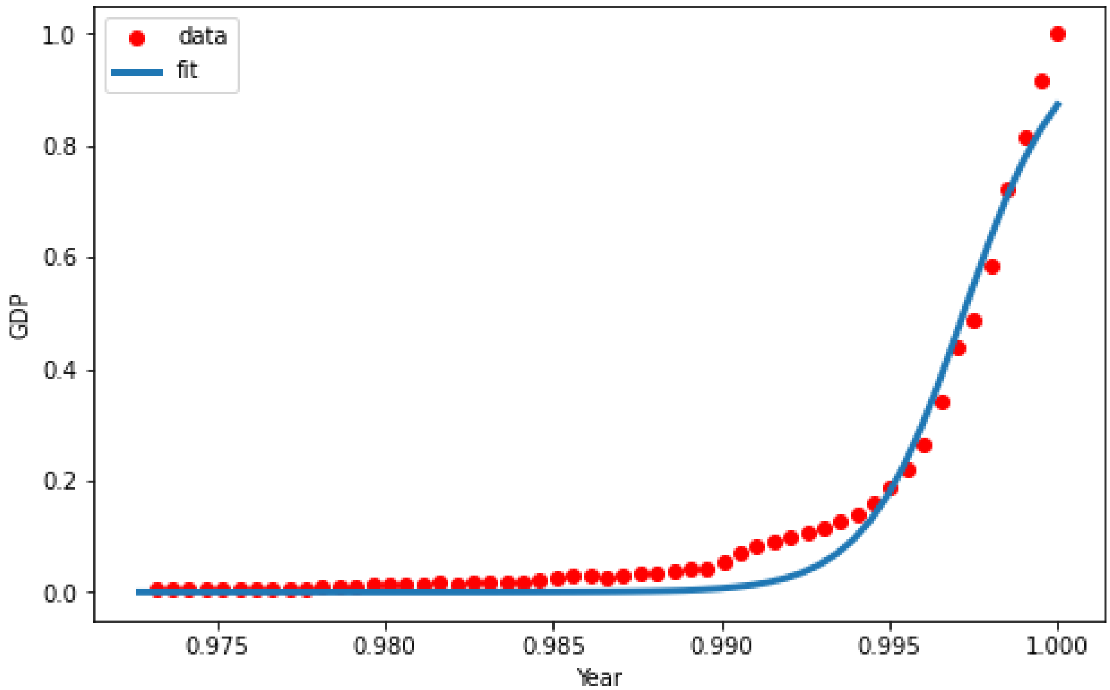
[22]:
```
def sigmoid(x,Beta_1,Beta_2):
    y=1/(1+np.exp(-Beta_1*(x-Beta_2)))
    return y
```

Fit the logistic function on this dataset and estimate the relevant parameters

[23]:
```
from scipy.optimize import curve_fit
popt,pcov=curve_fit(sigmoid,xdata,ydata)
print("beta_1=%f,beta_2=%f"%(popt[0],popt[1]))

##
x=np.linspace(1960,2015,55)
x=x/max(x)
plt.figure(figsize=(8,5))
y=sigmoid(x,*popt)
plt.plot(xdata,ydata,'ro',label='data')
plt.plot(x,y,linewidth=3.0,label='fit')
plt.legend(loc='best')
plt.ylabel('GDP')
```

```
plt.xlabel('Year')
plt.show()
```

```
beta_1=690.451712,beta_2=0.997207
```



[ ]:

```
```

[ ]:

```
```