Unit 4 Seminar Activity - Linear Regression with Scikit-Learn

Fuel Consumption data set

```
import matplotlib.pyplot as plt
import pandas as pd
import pylab as pl
import numpy as np
import seaborn as sns
```

importing the required packages

```
# Reading the data
df=pd.read_csv("FuelConsumption.csv")
```

loading the correct data

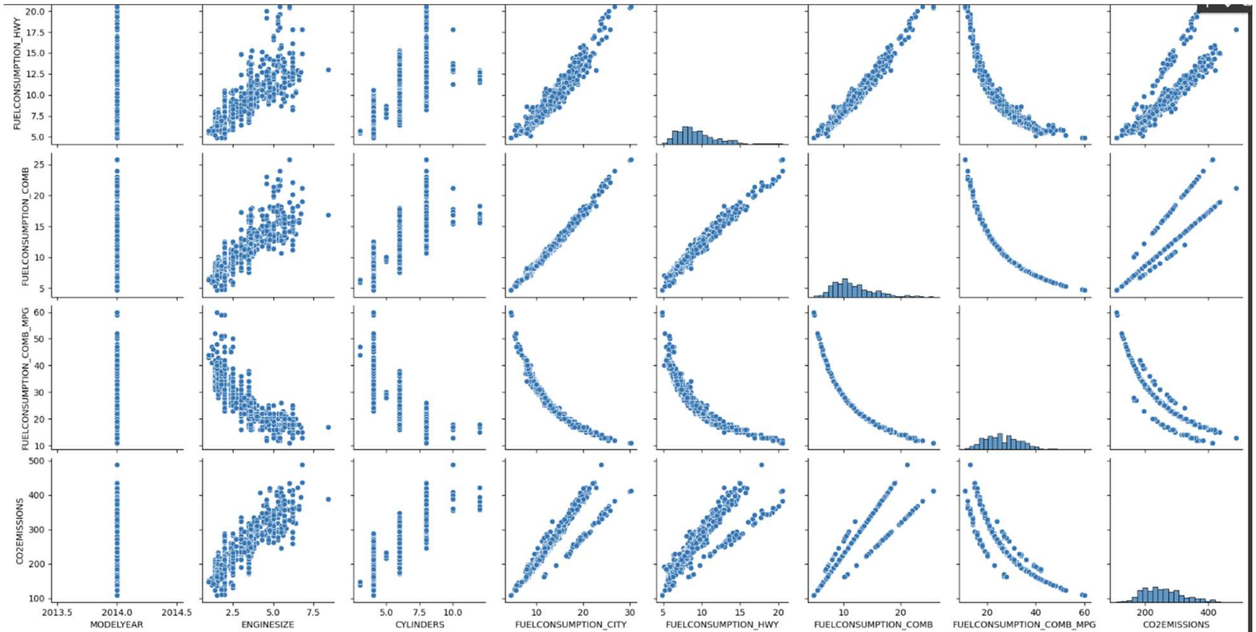
```
# Take a look at the dataset
df.head(10)
```

taking a look at the dataset

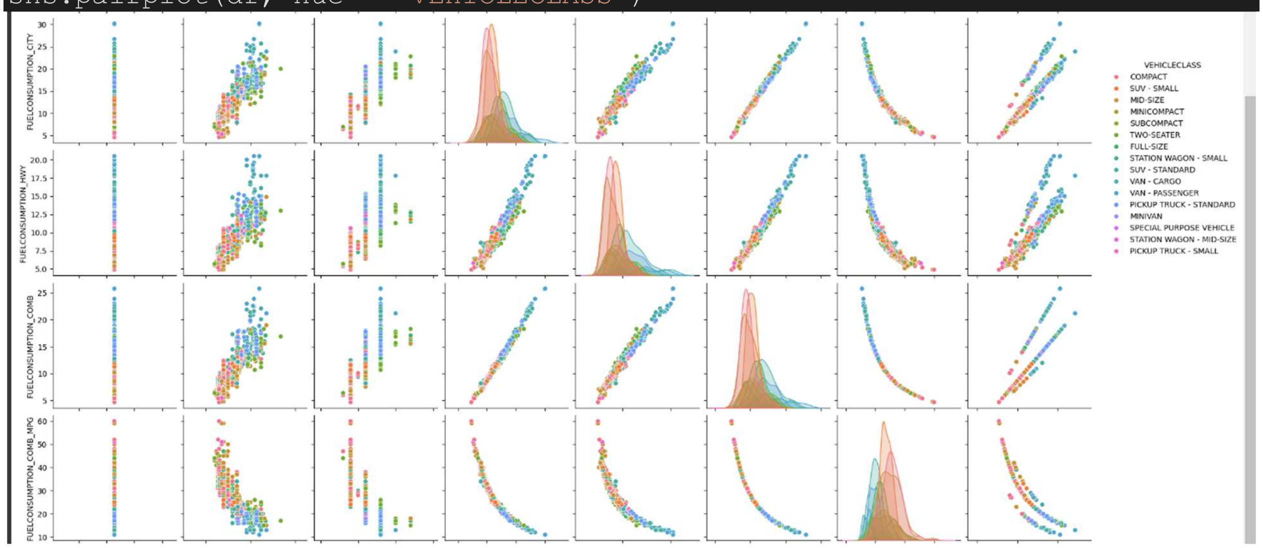
	MODELYEAR	MAKE	MODEL	VEHICLECLASS	ENGINESIZE	CYLINDERS	TRANSMISSION	FUELTYPE	FUELCONSUMPTION_CITY	FUELCONSUMPTION_HWY	FUELCONSUMPTION_COMB	FUELCONSUMPTION_COMB_MPG	CO2EMISSIONS
0	2014	ACURA	ILX	COMPACT	2.0	4	AS5	Z	9.9	6.7	8.5	33	196
1	2014	ACURA	ILX	COMPACT	2.4	4	M6	Z	11.2	7.7	9.6	29	221
2	2014	ACURA	ILX HYBRID	COMPACT	1.5	4	AV7	Z	6.0	5.8	5.9	48	136
3	2014	ACURA	MDX 4WD	SUV - SMALL	3.5	6	AS6	Z	12.7	9.1	11.1	25	255
4	2014	ACURA	RDX AWD	SUV - SMALL	3.5	6	AS6	Z	12.1	8.7	10.6	27	244
5	2014	ACURA	RLX	MID-SIZE	3.5	6	AS6	Z	11.9	7.7	10.0	28	230
6	2014	ACURA	TL	MID-SIZE	3.5	6	AS6	Z	11.8	8.1	10.1	28	232
7	2014	ACURA	TL AWD	MID-SIZE	3.7	6	AS6	Z	12.8	9.0	11.1	25	255
8	2014	ACURA	TL AWD	MID-SIZE	3.7	6	M6	Z	13.4	9.5	11.6	24	267
9	2014	ACURA	TSX	COMPACT	2.4	4	AS5	Z	10.6	7.5	9.2	31	212

```
sns.pairplot(df)
```

this allows us to plot pairwise relationships between variables within a dataset as seen below




```
sns.pairplot(df, hue = "VEHICLECLASS")
```



The variable vehicle class has now been colour coded within the pairwise relationships to help determine and identify the vehicle classes on each plot.

```
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)
```

A correlation heatmap can be used to help determine the strength or correlation between two variables.

```
# 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
```

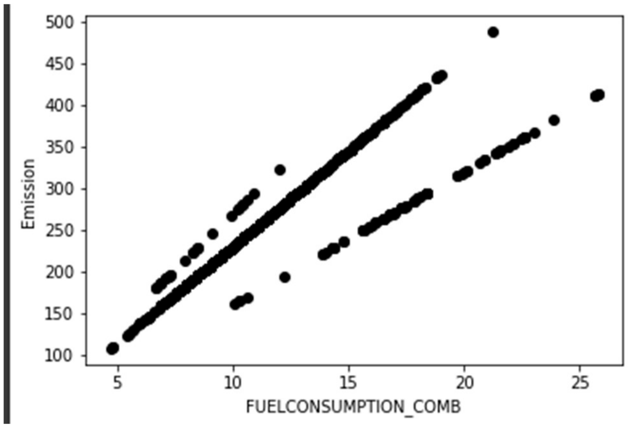
```
         CO2EMISSIONS
count     1067.000000
mean       256.228679
std         63.372304
min        108.000000
25%        207.000000
50%        251.000000
75%        294.000000
max        488.000000
```

	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

Df.describe can help give a 5 point summary

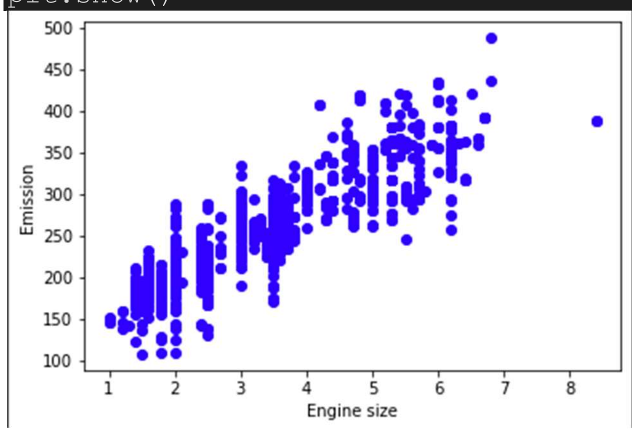
Cdf=df gives a cumulative distribution of the variables that have been selected by the user.

```
plt.scatter(cdf.FUELCONSUMPTION_COMB,cdf.CO2EMISSIONS, color='black')
plt.xlabel("FUELCONSUMPTION_COMB")
plt.ylabel("Emission")
plt.show()
```

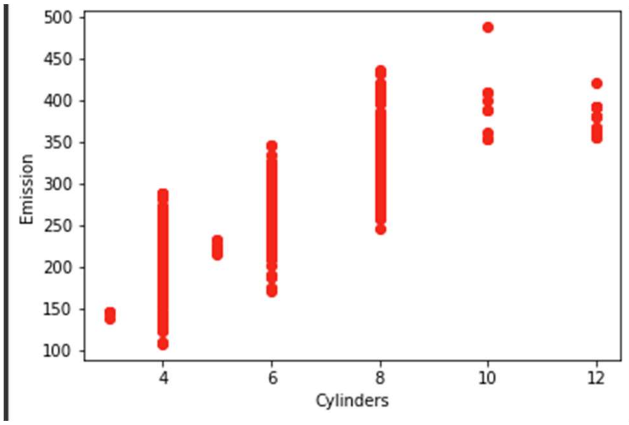


Based on the data gathered a scatter plot can be made to show the correlation between two variables with a strong correlation. In the correlation matrix we know these two variables have a positive correlation of 0.89. A few more variables measured against emissions can be seen below.

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




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



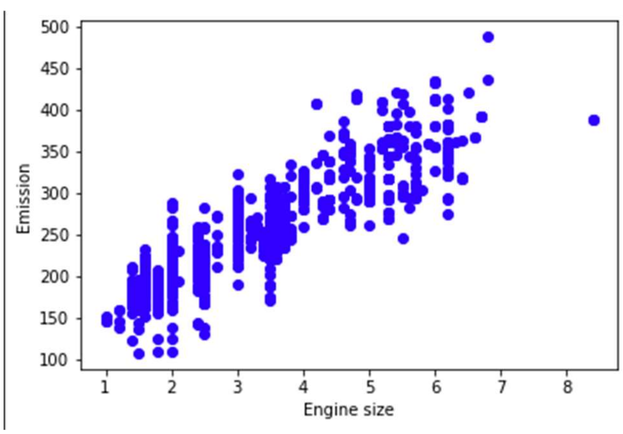
Engine size and fuel consumption look like the best variables to explain the linear relation with CO2.

Regression

CDF is the summarised data

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

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




```
from sklearn import linear_model
regr=linear_model.LinearRegression()
train_x=np.asanyarray(train[['ENGINESIZE']])
```

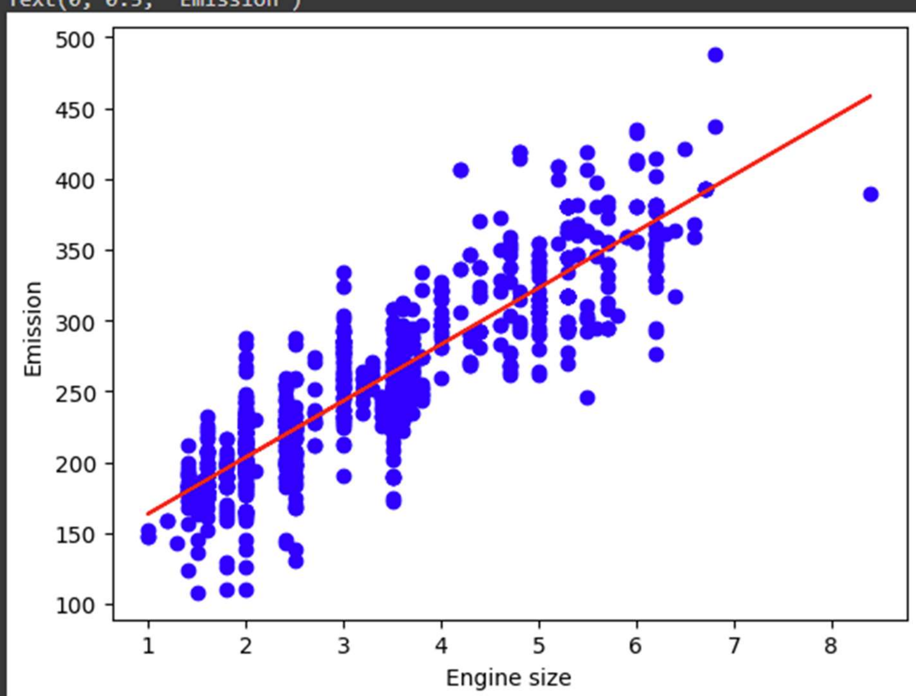
```
train_y=np.asanyarray(train[['CO2EMISSIONS']])

regr.fit(train_x, train_y)
# The coefficients
print('Coefficients:', regr.coef_)
print('Intercept:', regr.intercept_)
```

```
Coefficients: [[39.81891431]]
Intercept: [123.75721334]
```

```
# 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")
```

```
Text(0, 0.5, 'Emission')
```




```
from sklearn.metrics import r2_score
test_x=np.asanyarray(test[['ENGINESIZE']])
```

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

```
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))
```

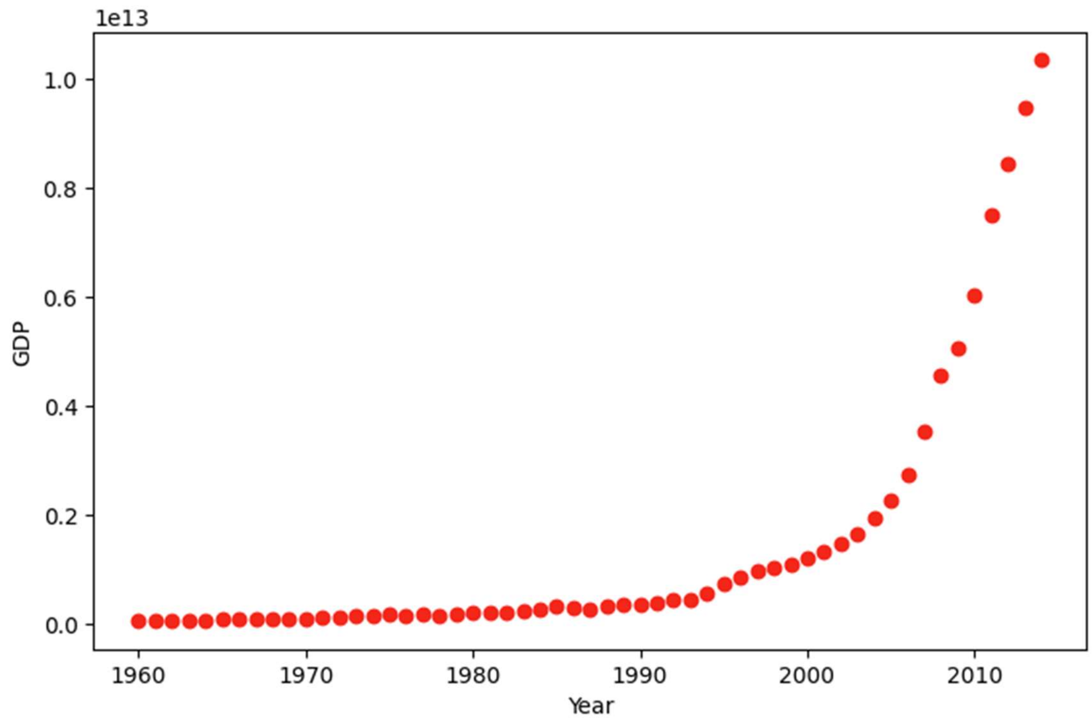
```
Mean absolute error: 10078.83
Residual sum of squares (MSE): 107629588.24
R2-score: -15.92
```

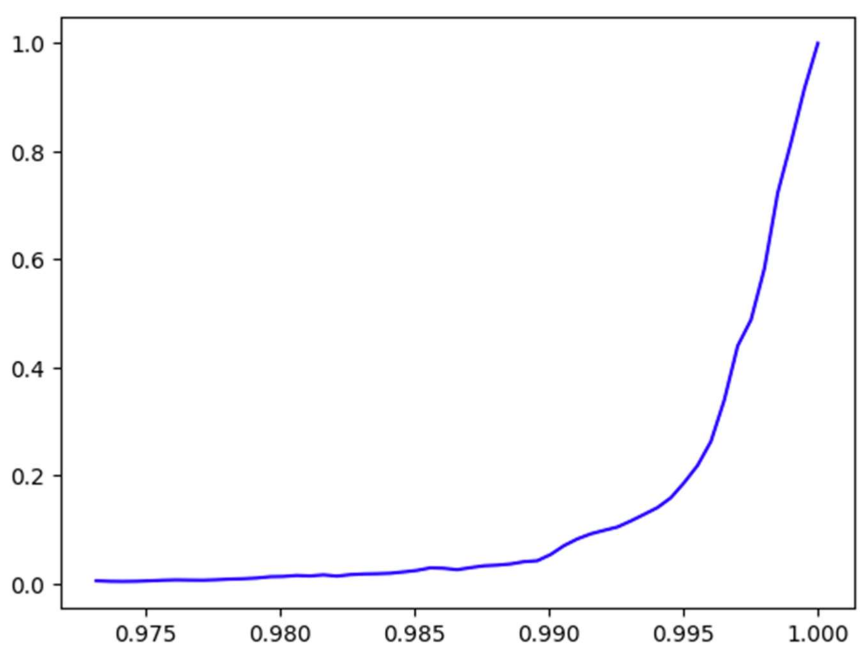
Non-linear regression

```
df=pd.read_csv("china_gdp.csv")
df.head(10)
```

	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

After checking the data the dataset will then be plotted as seen below





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

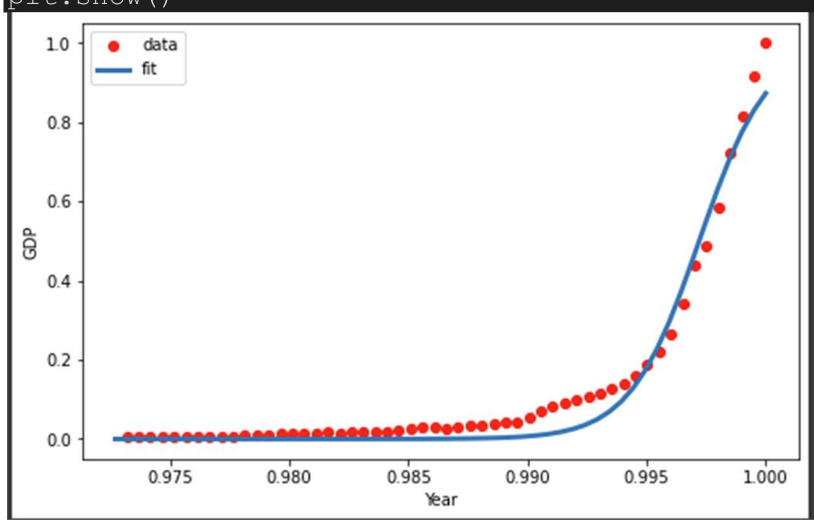
Implementing the logistic function

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

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

```
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()
```



The blue line would be seen as the line of best fit