Important libraries:

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

Load Your Dataset  
data = pd.read\_csv('your\_dataset.csv')

Splitting the data into training and testing

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

Creating model:

model = LinearRegression()

model.fit(X\_train, y\_train)

testing:

y\_pred = model.predict(X\_test)

mse = mean\_squared\_error(y\_test, y\_pred)

print("Mean Squared Error:", mse)

Code for simple linear regression:

import matplotlib.pyplot as plt

from scipy import stats

x = [5,7,8,7,2,17,2,9,4,11,12,9,6]

y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

slope, intercept, r, p, std\_err = stats.linregress(x, y)

def myfunc(x):

return slope \* x + intercept

mymodel = list(map(myfunc, x))

plt.scatter(x, y)

plt.plot(x, mymodel)

plt.show()

For plotting function we use :

By default, the plot() function draws a line from point to point.

The function takes parameters for specifying points in the diagram.

Parameter 1 is an array containing the points on the x-axis.

Parameter 2 is an array containing the points on the y-axis.

If we need to plot a line from (1, 3) to (8, 10), we have to pass two arrays [1, 8] and [3, 10] to the plot function.

import matplotlib.pyplot as plt

import numpy as np

xpoints = np.array([1, 8])

ypoints = np.array([3, 10])

plt.plot(xpoints, ypoints)

plt.show()

Draw a line in a diagram from position (1, 3) to (2, 8) then to (6, 1) and finally to position (8, 10):

import matplotlib.pyplot as plt

import numpy as np

xpoints = np.array([1, 2, 6, 8])

ypoints = np.array([3, 8, 1, 10])

plt.plot(xpoints, ypoints)

plt.show()

Code for prediction:

from scipy import stats

x = [5,7,8,7,2,17,2,9,4,11,12,9,6]

y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

slope, intercept, r, p, std\_err = stats.linregress(x, y)

def myfunc(x):

return slope \* x + intercept

speed = myfunc(10)

print(speed)

Polynomial regression:

import numpy

import matplotlib.pyplot as plt

x = [1,2,3,5,6,7,8,9,10,12,13,14,15,16,18,19,21,22]

y = [100,90,80,60,60,55,60,65,70,70,75,76,78,79,90,99,99,100]

mymodel = numpy.poly1d(numpy.polyfit(x, y, 3))

myline = numpy.linspace(1, 22, 100)

plt.scatter(x, y)

plt.plot(myline, mymodel(myline))

plt.show()

Relation squared:

import numpy

from sklearn.metrics import r2\_score

x = [1,2,3,5,6,7,8,9,10,12,13,14,15,16,18,19,21,22]

y = [100,90,80,60,60,55,60,65,70,70,75,76,78,79,90,99,99,100]

mymodel = numpy.poly1d(numpy.polyfit(x, y, 3))

print(r2\_score(y, mymodel(x)))

Predict future values:

import numpy

from sklearn.metrics import r2\_score

x = [1,2,3,5,6,7,8,9,10,12,13,14,15,16,18,19,21,22]

y = [100,90,80,60,60,55,60,65,70,70,75,76,78,79,90,99,99,100]

mymodel = numpy.poly1d(numpy.polyfit(x, y, 3))

speed = mymodel(17)

print(speed)

Multiple regression:

import pandas

from sklearn import linear\_model

df = pandas.read\_csv("data.csv")

X = df[['Weight', 'Volume']]

y = df['CO2']

regr = linear\_model.LinearRegression()

regr.fit(X, y)

#predict the CO2 emission of a car where the weight is 2300kg, and the volume is 1300cm3:

predictedCO2 = regr.predict([[2300, 1300]])

print(predictedCO2)