https://www.tutorialspoint.com/artificial\_intelligence\_with\_python/index.htm

import numpy as np

from sklearn import linear\_model

import sklearn.metrics as sm

import matplotlib.pyplot as plt

input = 'D:/ProgramData/linear.txt'

We need to load this data by using the **np.loadtxt** function.

input\_data = np.loadtxt(input, delimiter=',')

X, y = input\_data[:, :-1], input\_data[:, -1]

Train the model.

training\_samples = int(0.6 \* len(X))

testing\_samples = len(X) - num\_training

X\_train, y\_train = X[:training\_samples], y[:training\_samples]

X\_test, y\_test = X[training\_samples:], y[training\_samples:]

Create a linear regressor object.

reg\_linear = linear\_model.LinearRegression()

Train the object with the training samples.

reg\_linear.fit(X\_train, y\_train)

We need to do the prediction with the testing data.

y\_test\_pred = reg\_linear.predict(X\_test)

Now plot and visualize the data.

plt.scatter(X\_test, y\_test, color = 'red')

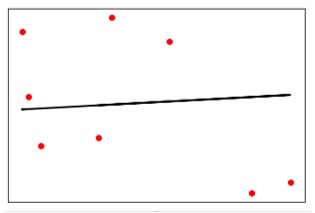
plt.plot(X\_test, y\_test\_pred, color = 'black', linewidth = 2)

plt.xticks(())

plt.yticks(())

plt.show()

Output



Compute the performance of our linear regression as follows −

print("Performance of Linear regressor:")

print("Mean absolute error =", round(sm.mean\_absolute\_error(y\_test, y\_test\_pred), 2))

print("Mean squared error =", round(sm.mean\_squared\_error(y\_test, y\_test\_pred), 2))

print("Median absolute error =", round(sm.median\_absolute\_error(y\_test, y\_test\_pred), 2))

print("Explain variance score =", round(sm.explained\_variance\_score(y\_test, y\_test\_pred),

2))

print("R2 score =", round(sm.r2\_score(y\_test, y\_test\_pred), 2))

Output

Performance of Linear Regressor −

Mean absolute error = 1.78

Mean squared error = 3.89

Median absolute error = 2.01

Explain variance score = -0.09

R2 score = -0.09

In the above code, we have used this small data. If you want some big dataset then you can use sklearn.dataset to import bigger dataset.

2,4.82.9,4.72.5,53.2,5.56,57.6,43.2,0.92.9,1.92.4,

3.50.5,3.41,40.9,5.91.2,2.583.2,5.65.1,1.54.5,

1.22.3,6.32.1,2.8

Multivariable Regressor

import numpy as np

from sklearn import linear\_model

import sklearn.metrics as sm

import matplotlib.pyplot as plt

from sklearn.preprocessing import PolynomialFeatures

input = 'D:/ProgramData/Mul\_linear.txt'

input\_data = np.loadtxt(input, delimiter=',')

X, y = input\_data[:, :-1], input\_data[:, -1]

Train the model and testing samples.

training\_samples = int(0.6 \* len(X))

testing\_samples = len(X) - num\_training

X\_train, y\_train = X[:training\_samples], y[:training\_samples]

X\_test, y\_test = X[training\_samples:], y[training\_samples:]

Create a linear regressor object.

reg\_linear\_mul = linear\_model.LinearRegression()

Train the object with the training samples.

reg\_linear\_mul.fit(X\_train, y\_train)

The prediction with the testing data.

y\_test\_pred = reg\_linear\_mul.predict(X\_test)

print("Performance of Linear regressor:")

print("Mean absolute error =", round(sm.mean\_absolute\_error(y\_test, y\_test\_pred), 2))

print("Mean squared error =", round(sm.mean\_squared\_error(y\_test, y\_test\_pred), 2))

print("Median absolute error =", round(sm.median\_absolute\_error(y\_test, y\_test\_pred), 2))

print("Explain variance score =", round(sm.explained\_variance\_score(y\_test, y\_test\_pred), 2))

print("R2 score =", round(sm.r2\_score(y\_test, y\_test\_pred), 2))

Output

Performance of Linear Regressor −

Mean absolute error = 0.6

Mean squared error = 0.65

Median absolute error = 0.41

Explain variance score = 0.34

R2 score = 0.33

We will create a polynomial of degree 10 and train the regressor.

We will provide the sample data point.

polynomial = PolynomialFeatures(degree = 10)

X\_train\_transformed = polynomial.fit\_transform(X\_train)

datapoint = [[2.23, 1.35, 1.12]]

poly\_datapoint = polynomial.fit\_transform(datapoint)

poly\_linear\_model = linear\_model.LinearRegression()

poly\_linear\_model.fit(X\_train\_transformed, y\_train)

print("\nLinear regression:\n", reg\_linear\_mul.predict(datapoint))

print("\nPolynomial regression:\n", poly\_linear\_model.predict(poly\_datapoint))

Output

Linear regression −

[2.40170462]

Polynomial regression −

[1.8697225]

In the above code, we have used this small data. If you want a big dataset then, you can use sklearn.dataset to import a bigger dataset.

2,4.8,1.2,3.22.9,4.7,1.5,3.62.5,5,2.8,23.2,5.5,3.5,2.16,5,

2,3.27.6,4,1.2,3.23.2,0.9,2.3,1.42.9,1.9,2.3,1.22.4,3.5,

2.8,3.60.5,3.4,1.8,2.91,4,3,2.50.9,5.9,5.6,0.81.2,2.58,

3.45,1.233.2,5.6,2,3.25.1,1.5,1.2,1.34.5,1.2,4.1,2.32.3,

6.3,2.5,3.22.1,2.8,1.2,3.6