import numpy as np  
import sklearn.preprocessing  
Input\_data = np.array([2.1, -1.9, 5.5],  
 [-1.5, 2.4, 3.5],  
 [0.5, -7.9, 5.6],  
 [5.9, 2.3, -5.8])  
data\_binarized = preprocessing.Binarizer(threshold = 0.5).transform(input\_data  
print("\nBinarized data:\n", data\_binarized)

import numpy as np  
import sklearn.preprocessing  
Input\_data = np.array([2.1, -1.9, 5.5],  
 [-1.5, 2.4, 3.5],  
 [0.5, -7.9, 5.6],  
 [5.9, 2.3, -5.8])  
print("Mean = ", input\_data.mean(axis = 0))  
print("Std deviation = ", input\_data.std(axis = 0))  
data\_scaled = preprocessing.scale(input\_data)  
print("Mean =", data\_scaled.mean(axis=0))  
print("Std deviation =", data\_scaled.std(axis = 0))

import numpy as np  
import sklearn.preprocessing  
Input\_data = np.array([2.1, -1.9, 5.5],  
 [-1.5, 2.4, 3.5],  
 [0.5, -7.9, 5.6],  
 [5.9, 2.3, -5.8])  
data\_scaler\_minmax = preprocessing.MinMaxScaler(feature\_range=(0,1))  
data\_scaled\_minmax = data\_scaler\_minmax.fit\_transform(input\_data)  
print ("\nMin max scaled data:\n", data\_scaled\_minmax)

import numpy as np  
import sklearn.preprocessing  
Input\_data = np.array([2.1, -1.9, 5.5],  
 [-1.5, 2.4, 3.5],  
 [0.5, -7.9, 5.6],  
 [5.9, 2.3, -5.8])  
data\_normalized\_l1 = preprocessing.normalize(input\_data, norm = 'l1')  
print("\nL1 normalized data:\n", data\_normalized\_l1)

import numpy as np  
from sklearn import preprocessing  
input\_labels = ['red' ,'black' ,'red' ,'green' ,'black' ,'yellow' ,'white']  
encoder = preprocessing.LabelEncoder()  
encoder.fit(input\_labels)  
test\_labels = ['green','red','black']  
encoded\_values = encoder.transform(test\_labels)  
print("\nLabels =", test\_labels)  
print("Encoded values =", list(encoded\_values))  
encoded\_values = [3,0,4,1]  
decoded\_list = encoder.inverse\_transform(encoded\_values)  
print("\nEncoded values =", encoded\_values)  
print("\nDecoded labels =", list(decoded\_list))

importnumpyasnp  
from sklearn import linear\_model  
import sklearn.metrics as sm  
importmatplotlib.pyplotasplt  
input = 'D:/ProgramData/linear.txt'  
input\_data = np.loadtxt(input, delimiter=',')  
x , y = input\_data[:, :-1], input\_data[:, -1]  
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:]  
reg\_linear = linear\_model.LinearRegression()  
reg\_linear.fit(X\_train, y\_train)  
y\_test\_pred = reg\_linear.predict(X\_test)  
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()  
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 abs error =", round(sm.median\_absolute\_error(y\_test, y\_test\_pred), 2))print("Explain var scr =", round(sm.explained\_variance\_score(y\_test, y\_test\_pred),2))print("R2 score =", round(sm.r2\_score(y\_test, y\_test\_pred), 2))

import numpy as np  
from sklearn import linear\_model  
import sklearn.metrics as sm  
import matplotlib.pyplot as plt  
fromsklearn.preprocessingimportPolynomialFeatures  
input = 'D:/ProgramData/Mul\_linear.txt'  
input\_data = np.loadtxt(input, delimiter=',')  
X, y = input\_data[:, :-1], input\_data[:, -1]  
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:]  
reg\_linear\_mul = linear\_model.LinearRegression()  
reg\_linear\_mul.fit(X\_train, y\_train)  
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 abs error =", round(sm.median\_absolute\_error(y\_test, y\_test\_pred), 2))print("Explain var scr=", round(sm.explained\_variance\_score(y\_test, y\_test\_pred), 2))print("R2 score =", round(sm.r2\_score(y\_test, y\_test\_pred), 2))  
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))