

| **Title: Simple linear regression and multivariate linear regression** |
| --- |

**Aim:** To understand & implement Simple linear regression and multivariate linear regression

**Expected Outcome of Experiment:**

CO1: Describe the basics of machine learning..

**Books/ Journals/ Websites referred:**

**Explanation of simple linear regression:**

Simple linear regression aims to find a linear relationship to describe the correlation between an independent and possibly dependent variable.

Simple linear regression is a parametric test, meaning that it makes certain assumptions about the data. These assumptions are:

**Homogeneity of variance (homoscedasticity)**: the size of the error in our prediction doesn’t change significantly across the values of the independent variable.

**Independence of observations**: the observations in the dataset were collected using statistically valid sampling methods, and there are no hidden relationships among observations.

**Program:**

**import numpy as np**

**import matplotlib.pyplot as plt**

**X = [17,21,24,28,14,16,19,22,15,18] #dependant**

**Y = [140,189,210,240,130,100,135,166,130,170] #independant**

**X = np.array(X)**

**Y = np.array(Y)**

**#Y=mX+C**

**n = len(X)**

**def MSE(A,B):**

**return np.sum((A-B)\*\*2)/n**

**m = 0**

**c = 0**

**loss=[]**

**for i in range(200):**

**print(f"epoch: {i}")**

**# gradient descent**

**lr = 0.001**

**predicted = m\*X+c**

**Error = MSE(predicted,Y)**

**print(f"Error = {Error}")**

**print(f"Values Predicted = {predicted}")**

**loss.append(np.log(Error))**

**m -= (2/n)\*lr\*np.sum((predicted-Y)\*X)**

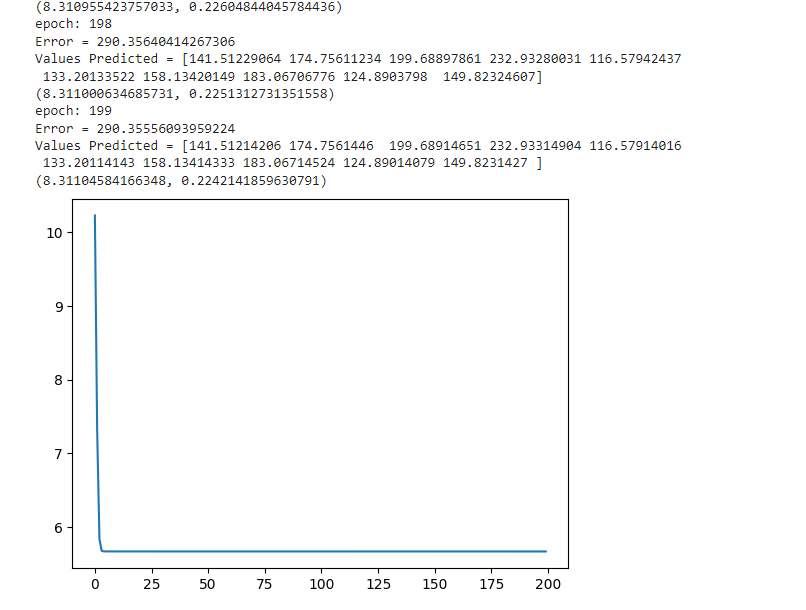
**c -= (2/n)\*lr\*np.sum(predicted-Y)**

**print((m,c))**

**plt.plot(loss)**

**plt.show()**

**Output:**



**Explanation of multivariate linear regression:**

Multivariate Regression is a method used to measure the degree at which more than one independent variable (predictors) and more than one dependent variable (responses), are linearly related. Multivariate analysis is based in observation and analysis of more than one statistical outcome variable at a time. In design and analysis, the technique is used to perform trade studies across multiple dimensions while taking into account the effects of all variables on the responses of interest.

**Program :**

**import numpy as np**

**import matplotlib.pyplot as plt**

**Data = data.to\_numpy()**

**X = Data[:,:-1].T**

**Y = Data[:,-1]**

**#Y=mX+C**

**n = len(X[0])**

**def MSE(A,B):**

**return np.sum((A-B)\*\*2)/n**

**X = np.vstack((X,np.ones\_like(X[0])))**

**loss = []**

**print(X)**

**theta = np.zeros(len(X))**

**print(np.shape(theta))**

**for i in range(2000):**

**print(f"epoch: {i}")**

**# gradient descent**

**lr = 0.00001**

**X**

**predicted = theta.T @ X**

**print(n)**

**Error = MSE(predicted,Y)**

**print(f"Error = {Error}")**

**loss.append(Error)**

**print(f"Values Predicted = {predicted}")**

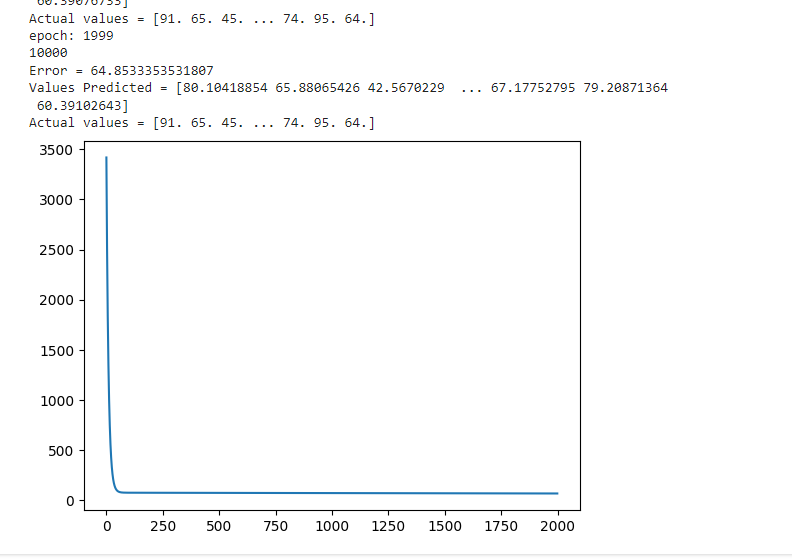
**print(f"Actual values = {Y}")**

**theta -=lr\*1/n\*np.sum((predicted-Y)\*X,axis=1)**

**plt.plot(loss)**

**plt.show()**

**Output:**

****

**Conclusion:-**

Thus we have performed linear regression on a simple and multivariate dataset. We have understood how linear regression works and have implemented it on a sample dataset. We have implemented linear regression without using any external library.