**G. H. RAISONI COLLEGE OF ENGG., NAGPUR**

**(An Autonomous Institute)**

**Department of Computer Science & Engg.**



**Date: 21-09-2021**

**Practical Subject: Skill Development-2 [BCSP318]**

**Session: 2021-22**

**Student Details:**

| **Roll Number** | 01 |
| --- | --- |
| **Name** | Anand Suralkar |
| **Semester** | 9th |
| **Section** | A |
| **Batch** | CSE |

**Practical Details: Practical Number-8;**

| Practical Aim | Implement Multiple L. Regression in python |
| --- | --- |
| Theory & Syntax | Multiple Linear Regression is an extension of Simple Linear regression as it takes more than one predictor variable to predict the response variable. It is an important regression algorithm that models the linear relationship between a single dependent continuous variable and more than one independent variable. It uses two or more independent variables to predict a dependent variable by fitting a best linear relationship.  It has two or more independent variables (X) and one dependent variable (Y), where Y is the value to be predicted. Thus, it is an approach for predicting a quantitative response using multiple features.  Equation: Y = β0 + β1X1 + β2X2 + β3X3 + … + βnXn + e  Y = Dependent variable / Target variable  β0 = Intercept of the regression line  β1, β2, β3, …. βn = Slope of the regression line which tells whether the line is increasing or decreasing  X1, X2, X3, ….Xn = Independent variable / Predictor variable  e = Error  Example: Predicting sales based on the money spent on TV, Radio, and Newspaper for marketing. In this case, there are three independent variables, i.e., money spent on TV, Radio, and Newspaper for marketing, and one dependent variable, i.e., sales, that is the value to be predicted. |
| Program | import matplotlib.pyplot as plt  import numpy as np  from sklearn import datasets, linear\_model, metrics  # load the boston dataset  boston = datasets.load\_boston(return\_X\_y=False)  # defining feature matrix(X) and response vector(y)  X = boston.data  y = boston.target  # splitting X and y into training and testing sets  from sklearn.model\_selection import train\_test\_split  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.4, random\_state=1)  # create linear regression object  reg = linear\_model.LinearRegression()  # train the model using the training sets  reg.fit(X\_train, y\_train)  # regression coefficients  print('Coefficients: ', reg.coef\_)  # variance score: 1 means perfect prediction  print('Variance score: {}'.format(reg.score(X\_test, y\_test)))  # plot for residual error  ## setting plot style  plt.style.use('fivethirtyeight')  ## plotting residual errors in training data  plt.scatter(reg.predict(X\_train), reg.predict(X\_train) - y\_train, color = "green", s = 10, label =  'Train data')  ## plotting residual errors in test data  plt.scatter(reg.predict(X\_test), reg.predict(X\_test) - y\_test, color = "blue", s = 10, label = 'Test  data')  ## plotting line for zero residual error  plt.hlines(y = 0, xmin = 0, xmax = 50, linewidth = 2)  ## plotting legend  plt.legend(loc = 'upper right')  ## plot title  plt.title("Residual errors")  ## method call for showing the plot  plt.show() |
| Output |  |
| Conclusion | Implemented Multiple L. Regression in python using matplotlib and numpy |