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

**(An Autonomous Institute)**

**Department of Computer Science & Engg.**



**Date: 14-09-2021**

**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-7;**

| Practical Aim | Implement regression for predictive  modelling and linear regression |
| --- | --- |
| Theory & Syntax | Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002.  One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.  Installation :python -mpip install -U matplotlib  Windows, Linux and macOS distributions have matplotlib and most of its dependencies as wheel packages. Run the following command to install matplotlib package :  Plot a histogram.  Compute and draw the histogram of x. The return value is a tuple (n, bins, patches) or ([n0, n1, ...], bins, [patches0, patches1, ...]) if the input contains multiple data. See the documentation of the weights parameter to draw a histogram of already-binned data.  Multiple data can be provided via x as a list of datasets of potentially different length ([x0, x1, ...]), or as a 2D ndarray in which each column is a dataset. Note that the ndarray form is transposed relative to the list form.  Masked arrays are not supported.  The bins, range, weights, and density parameters behave as in numpy.histogram. |
| Program | import numpy as np  import matplotlib.pyplot as plt    def estimate\_coef(x, y):  # number of observations/points  n = np.size(x)    # mean of x and y vector  m\_x, m\_y = np.mean(x), np.mean(y)    # calculating cross-deviation and deviation about x  SS\_xy = np.sum(y\*x) - n\*m\_y\*m\_x  SS\_xx = np.sum(x\*x) - n\*m\_x\*m\_x    # calculating regression coefficients  b\_1 = SS\_xy / SS\_xx  b\_0 = m\_y - b\_1\*m\_x    return(b\_0, b\_1)    def plot\_regression\_line(x, y, b):  # plotting the actual points as scatter plot  plt.scatter(x, y, color = "m",  marker = "o", s = 30)    # predicted response vector  y\_pred = b[0] + b[1]\*x    # plotting the regression line  plt.plot(x, y\_pred, color = "g")    # putting labels  plt.xlabel('x')  plt.ylabel('y')    # function to show plot  plt.show()    def main():  # observations  x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24])  y = np.array([2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97])    # estimating coefficients  b = estimate\_coef(x, y)  print("Estimated coefficients:\nb\_0 = {}\nb\_1 = {}".format(b[0], b[1]))    # plotting regression line  plot\_regression\_line(x, y, b)    if \_\_name\_\_ == "\_\_main\_\_":  main() |
| Output |  |
| Conclusion | Performed the Implementation of regression for predictive  modelling and linear regression |