**Experiment No.: 10**

**Aim**

 Linear Regression

**Program**

**1)Simple Linear Regression**

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 = np.mean(x)

m\_y = 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 / data

x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

y = np.array([1, 3, 2, 5, 7, 8, 8, 9, 10, 12])

# estimating coefficients

b = estimate\_coef(x, y)

print("Estimated coefficients:\nb\_0 = {} \

\nb\_1 = {}".format(b[0], b[1]))

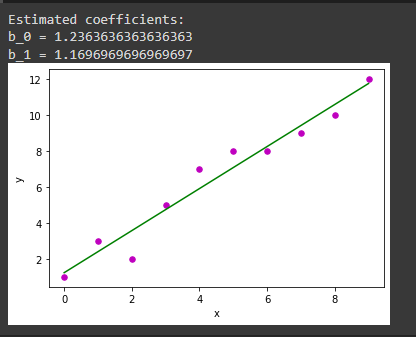
# plotting regression line

plot\_regression\_line(x, y, b)

if \_\_name\_\_ == "\_\_main\_\_":

main()

**output:**



**2)Multiple Linear Regression**

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')

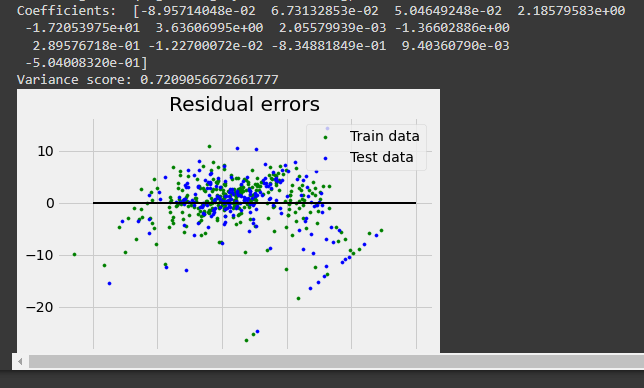
## plot title

plt.title("Residual errors")

## method call for showing the plot

plt.show()

**Output:**

****