**EXPERIMENT-3**

**Student’s Name: Pratham Mittal UID: 20BCS1040**

**Section/Group: 707\_WM\_B Subject Code: 20CSP-317**

**Subject Name: ML Lab Date of performance:25/9/2022**

**Branch: BE CSE Semester:5th**

**Aim:** Data analysis of any data set via graphs using linear regression.

**Objective:** To do Data Analysis on data set.

**Software/Hardware Requirements:** Windows 7 & above version.

**Tools to be used:**

1. Anaconda Jupyter Notebook,
2. numpy, pandas, matplotlib, sklearn.

**Introduction**

**Linear Regression** **– Finding a straight line of best fit through the data .This works well when the true underlying function is linear.**

A linear model makes a "hypothesis" about the true nature of the underlying function that it is linear. We express this hypothesis in the univariate case as

hθ(x)=ax+b

Our simple example above was an example of "univariate regression" - i.e. just one variable (or "feature") - number of hours studied. Below we will have more than one feature ("multivariate regression") which is given by

hθ(x)=aTX

Here **a** is a vector of learned parameters, and X is the "design matrix" with all the data points. In this formulation the intercept term has been added to the design matrix as the first column (of all ones).

**Code:**

--

import matplotlib.pyplot as plt

from sklearn import linear\_model, metrics, model\_selection

import numpy as np

import pandas as pd

--

hrs\_std = np.array([1, 2, 3, 4, 4, 5, 6, 7, 8, 8, 9, 10])

ex\_scr = np.array([17, 25, 33, 39, 48, 60, 71, 67, 75, 90, 86, 98])

plt.scatter(hrs\_std, ex\_scr)

plt.xlabel('Number of hours')

plt.ylabel('Scored in Exam')

plt.show()

--

plt.scatter(hrs\_std, ex\_scr)

x = np.linspace(0, 10)

y = 10\*x + 5.6

plt.plot(x, y, 'r')

plt.xlabel('Number of hours')

plt.ylabel('Scored in Exam')

plt.show()

--

import matplotlib.pyplot as plt

import numpy as np

from sklearn import datasets, linear\_model, metrics

from sklearn.model\_selection import train\_test\_split

# Loading the dataset of boston

bst = datasets.load\_boston()

# Defining the feature matrix(x) & response vector(y)

x = bst.data

y = bst.target

# Splitting the x & y into sets for training & testing

x\_tr, x\_ts, y\_tr, y\_ts = train\_test\_split(x, y, test\_size=0.4, random\_state=1)

# Creating object of Linear Regression

lr = linear\_model.LinearRegression()

# Training the model using training set

lr.fit(x\_tr, y\_tr)

# Regression Coefficients

print(f"Coefficients: {lr.coef\_}")

# Variance score

print(f"Variance score: {lr.score(x\_ts, y\_ts)}")

# Setting the plot style

plt.style.use("fivethirtyeight")

# Plotting Residual Errors in the Training data

plt.scatter(lr.predict(x\_tr), lr.predict(x\_tr)-y\_tr,

color = "red", s=10, label="Train Data")

# Plotting Residual Errors in the Testing data

plt.scatter(lr.predict(x\_ts), lr.predict(x\_ts)-y\_ts,

color = "green", s=10, label="Test Data")

# Plotting the line for Zero Residual Error

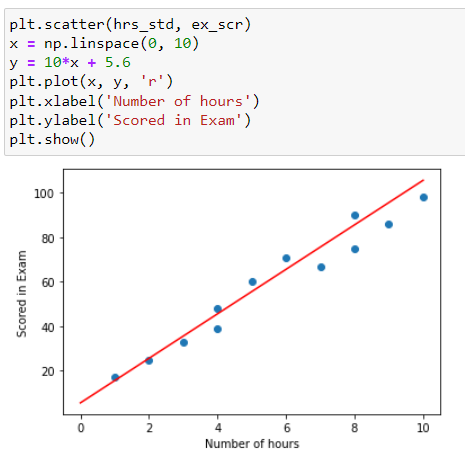
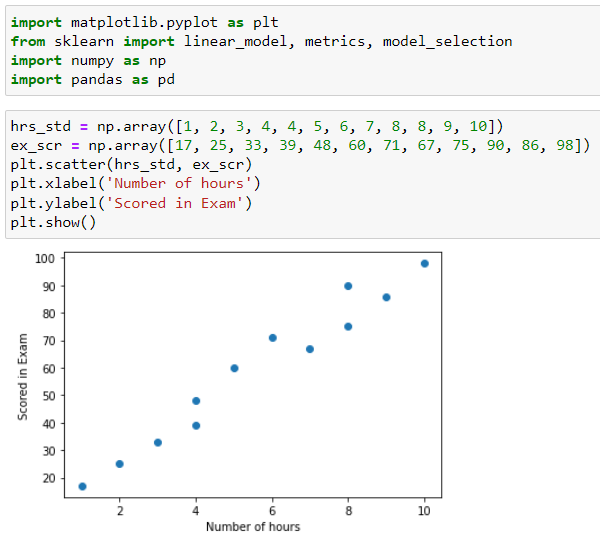
plt.hlines(y=0, xmin=0, xmax=50, linewidth=2)

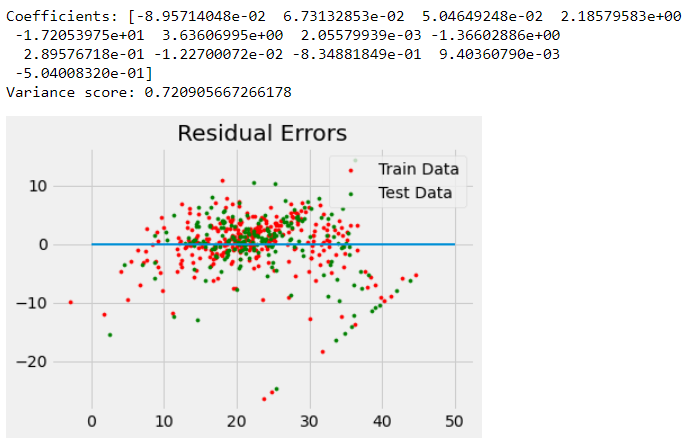
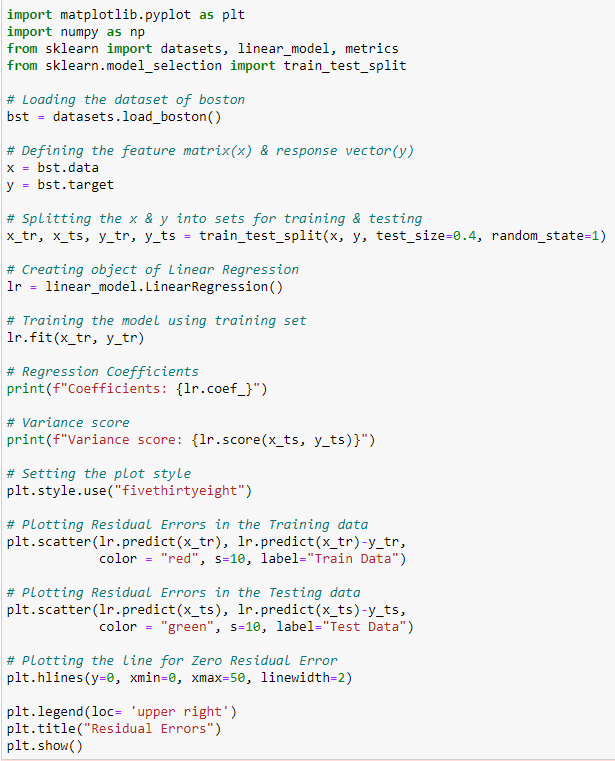
plt.legend(loc= 'upper right')

plt.title("Residual Errors")

plt.show()

**Output:**



--

**Learning Outcomes:**

1. We learned about data analysis and data handling in python.
2. We learned about various basic functions and libraries required for data analysis using python.
3. We learned graphically analyze data functions of matplotlib library in python.
4. We learned about linear regression and its implementation.