| **Experiment No .09** | | | | |
| --- | --- | --- | --- | --- |
| **Date of Performance:** |  | | | |
| **Date of Submission:** |  | | | |
| **Program Execution/ formation/**  **correction/ ethical practices**  **(06)** | **Timely**  **Submission**  **(01)** | **Viva Answer to**  **Sample questions**  **(03)** | **Experiment**  **Total (10)** | **Sign with Date** |
|  |  |  |  |  |

**Experiment No. 09**

**AIM:** Analysis of program based on Supervised machine learning .

**LABORATORY OUTCOME:**

**CO 1:** Define need of Artificial Intelligence and Machine Learning and explain working of Artificial Intelligence and Machine Learning algorithms.

**CO 2:** Make use of machine learning techniques to solve problems in different domains using scientific programming.

**PROBLEM STATEMENT:** To Implement Supervised machine learning using linear regression algorithm using python.

**RELATED THEORY:**

**Supervised Machine Learning**

Supervised learning is the types of machine learning in which machines are trained using well "labelled" training data, and on basis of that data, machines predict the output. The labelled data means some input data is already tagged with the correct output.

In supervised learning, the training data provided to the machines work as the supervisor that teaches the machines to predict the output correctly. It applies the same concept as a student learns in the supervision of the teacher.

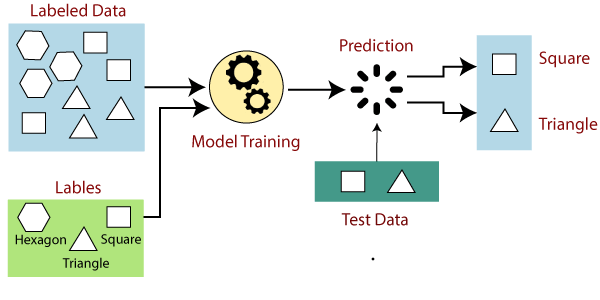
Supervised learning is a process of providing input data as well as correct output data to the machine learning model. The aim of a supervised learning algorithm is to find a mapping function to map the input variable(x) with the output variable(y).

In the real-world, supervised learning can be used for Risk Assessment, Image classification, Fraud Detection, spam filtering, etc.

**How Supervised Learning Works?**

In supervised learning, models are trained using labelled dataset, where the model learns about each type of data. Once the training process is completed, the model is tested on the basis of test data (a subset of the training set), and then it predicts the output.

The working of Supervised learning can be easily understood by the below example and diagram:



Suppose we have a dataset of different types of shapes which includes square, rectangle, triangle, and Polygon. Now the first step is that we need to train the model for each shape.

* If the given shape has four sides, and all the sides are equal, then it will be labelled as a Square.
* If the given shape has three sides, then it will be labelled as a triangle.
* If the given shape has six equal sides then it will be labelled as hexagon.

Now, after training, we test our model using the test set, and the task of the model is to identify the shape.

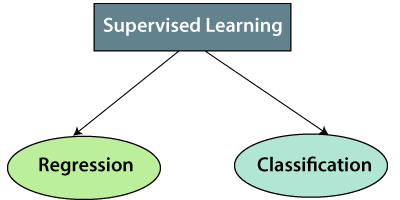
The machine is already trained on all types of shapes, and when it finds a new shape, it classifies the shape on the bases of a number of sides, and predicts the output.

**Steps Involved in Supervised Learning:**

* First Determine the type of training dataset
* Collect/Gather the labelled training data.
* Split the training dataset into training dataset, test dataset, and validation dataset.
* Determine the input features of the training dataset, which should have enough knowledge so that the model can accurately predict the output.
* Determine the suitable algorithm for the model, such as support vector machine, decision tree, etc.
* Execute the algorithm on the training dataset. Sometimes we need validation sets as the control parameters, which are the subset of training datasets.
* Evaluate the accuracy of the model by providing the test set. If the model predicts the correct output, which means our model is accurate.

**Types of supervised Machine learning Algorithms:**

Supervised learning can be further divided into two types of problems:



**1. Regression:**

Regression algorithms are used if there is a relationship between the input variable and the output variable. It is used for the prediction of continuous variables, such as Weather forecasting, Market Trends, etc. Below are some popular Regression algorithms which come under supervised learning:

* Linear Regression
* Regression Trees
* Non-Linear Regression
* Bayesian Linear Regression
* Polynomial Regression

**2. Classification:**

Classification algorithms are used when the output variable is categorical, which means there are two classes such as Yes-No, Male-Female, True-false, etc.

Spam Filtering,

* Random Forest
* Decision Trees
* Logistic Regression
* Support vector Machines

**Advantages of Supervised learning:**

* With the help of supervised learning, the model can predict the output on the basis of prior experiences.
* In supervised learning, we can have an exact idea about the classes of objects.
* Supervised learning model helps us to solve various real-world problems such as **fraud detection, spam filtering**, etc.

**Disadvantages of supervised learning:**

* Supervised learning models are not suitable for handling the complex tasks.
* Supervised learning cannot predict the correct output if the test data is different from the training dataset.
* Training required lots of computation times.
* In supervised learning, we need enough knowledge about the classes of object.

**Linear Regression:**

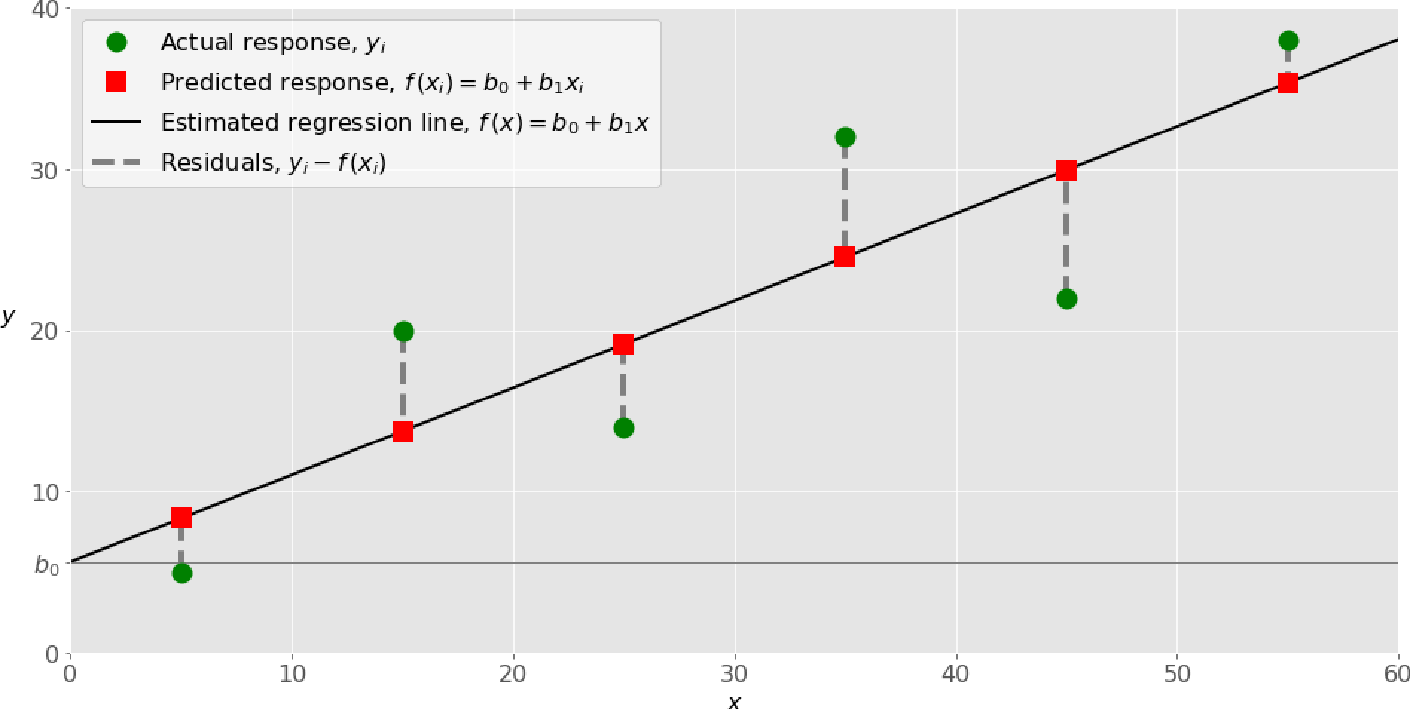
Linear regression is probably one of the most important and widely used regression techniques. It’s among the simplest regression methods. One of its main advantages is the ease of interpreting results. When implementing linear regression of some dependent variable

𝑦 on the set of independent variables x= (x1…xrᵣ), where r is the number of predictors, you assume a linear relationship between y and x: y= β0 + β1x1 + ……+ βrxr+ ϵ. This equation is the **regression equation**. β0,𝛽1, βr are the **regression coefficients**, and 𝜀is the **random error**.

Linear regression calculates the **estimators** of the regression coefficients or simply the **predicted weights**, denoted with b1…br. They define the **estimated regression function f**(x) = b0 +b1x1 + ….+ brxr This function should capture the dependencies between the inputs and output sufficiently well.

**Simple Linear Regression :**

The following figure illustrates simple linear regression:



When implementing simple linear regression, you typically start with a given set of input-output (x-y) pairs (green circles). These pairs are your observations. For example, the leftmost observation (green circle) has the input x= 5 and the actual output (response) y= 5.

The next one has x= 15 and y= 20, and so on.

The estimated regression function (blackline) has the equation f(x)=b0+b1x.Your goal is to calculate the optimal values of the predicted weights b0 and b1 that minimize SSR and determine the estimated regression function. The value of b0,also called the **intercept** ,shows the point where the estimated regression line crosses the y axis. It is the value of the estimated response f(x) for x=0. The value of b1 determines the **slope** of the estimated regression line.

The predicted responses (red squares) are the points on the regression line that correspond to the input values. For example, for the input x= 5, the predicted response is f (5) = 8.33 (represented with the leftmost red square).

The residuals(vertical dashed gray lines)can be calculated as,

y1ᵢ - f(x1ᵢ) = y1 – b0 – b1x1 for i=1,…,n .

They are the distances between the green circles and red squares. When you implement linear regression, you are actually trying to minimize these distances and make the red squares as close to the predefined green circles as possible.

**Implementing Linear Regression in Python**

It’s time to start implementing linear regression in Python. Basically, all you should do is apply the proper packages and their functions and classes.

**Python Packages for Linear Regression**

The package **NumPy** is a fundamental Python scientific package that allows many high-performing operations on single-and multi-dimensional arrays.It Also Offers Many Mathematical routines. Of course, it’s open source.

The package **scikit-learn** is a widely used Python library for machine learning, built on top of NumPy and some other packages. It provides the means for pre-processing data, reducing dimensionality,implementing regression,classification,clustering,andmore.LikeNumPy,scikit- learn is also open source.

If you want to implement linear regression and need the functionality beyond the scope of scikit- learn, you should consider **stats models**. It’s a powerful Python package for the estimation of statistical models, performing tests, and more. It’s open source as well.

**Simple Linear Regression With scikit-learn:**

Let’s start with the simplest case, which is simple linear regression. There are five basic steps when you’re implementing linear regression:

1. Import the packages and classes you need.

2. Provide data to work with and eventually do appropriate transformations.

3. Create a regression model and fit it with existing data.

4. Check the results of model fitting to know whether the model is satisfactory.

5. Apply the model for predictions.

**SOURCE CODE:**

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)

defplot\_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])

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 = {} \

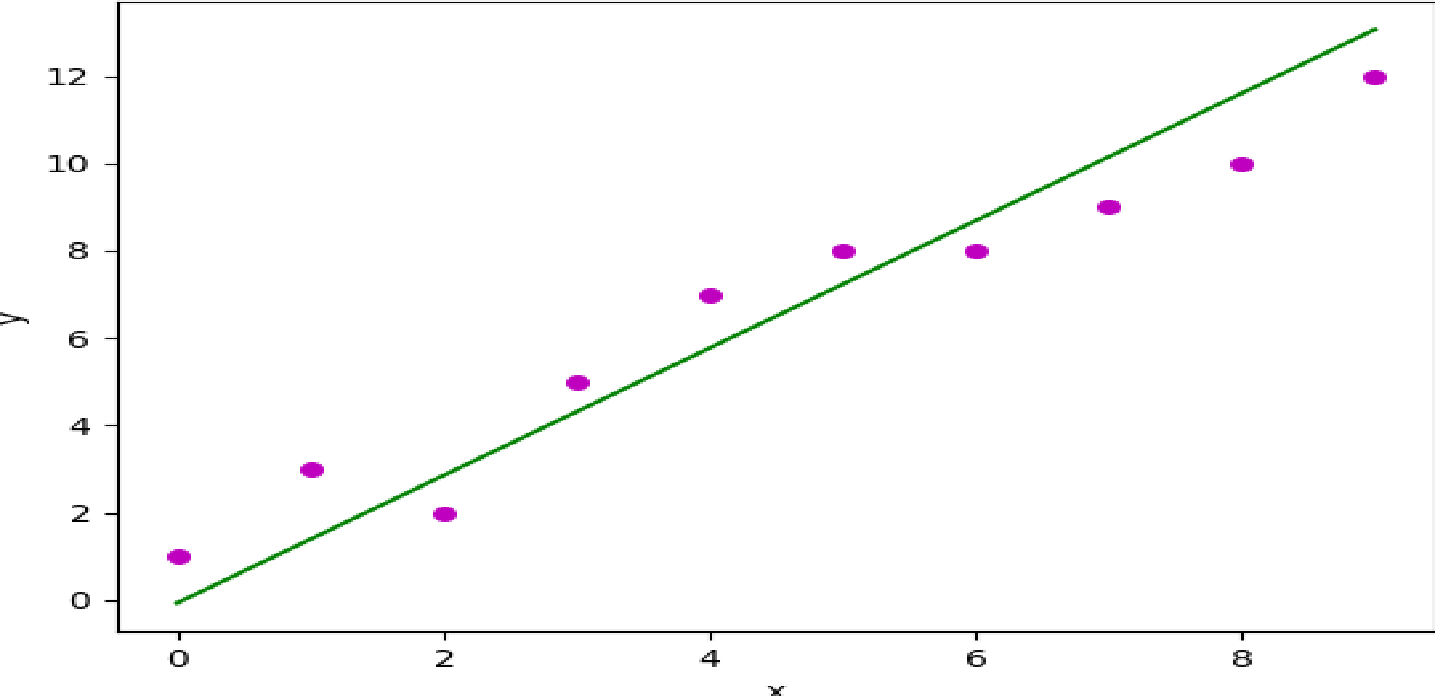
\nb\_1 = {}".format(b[0], b[1]))

# plotting regression line plot\_regression\_line(x, y, b)

ifname == "main": main()

**OUTPUT:**

Estimated coefficients: b\_0 = -0.05862068965



**RESULT/OUTPUT:**

**CONCLUSION:**

**QUESTIONS:**