Experiment No. 2
Implementation of Linear Regression Algorithm
Date of Performance:
Date of Submission:
Marks:
Sign:

Aim: Implementation of Linear Regression Algorithm.

Objective: To implement Linear Regression in order to build a model that studies the relationship between an independent and dependent variable. The model will be evaluated by using least square regression method where RMSE and R-squared will be the model evaluation parameters..

Theory:

The least-squares method is a crucial statistical method that is practiced to find a regression line or a best-fit line for the given pattern. This method is described by an equation with specific parameters. The method of least squares is generously used in evaluation and regression. In regression analysis, this method is said to be a standard approach for the approximation of sets of equations having more equations than the number of unknowns. The method of least squares actually defines the solution for the minimization of the sum of squares of deviations or the errors in the result of each equation. Find the formula for sum of squares of errors, which help to find the variation in observed data. The least-squares method is often applied in data fitting.

Least Squares Regression Example

Tom who is the owner of a retail shop, found the price of different T-shirts vs the number of T-shirts sold at his shop over a period of one week.

Price of T-shirts in dollars (x)	# of T-shirts sold (y)
2	4
3	5
5	7
7	10
9	15

Let us use the concept of least squares regression to find the line of best fit for the above data.

Step 1: Calculate the slope 'm' by using the following formula:

$$m = \frac{n\sum xy - (\Sigma x)(\Sigma y)}{n\Sigma x^2 - (\Sigma x)^2}$$

After you substitute the respective values, m = 1.518 approximately.

Step 2: Compute the y-intercept value

$$c = y - mx$$

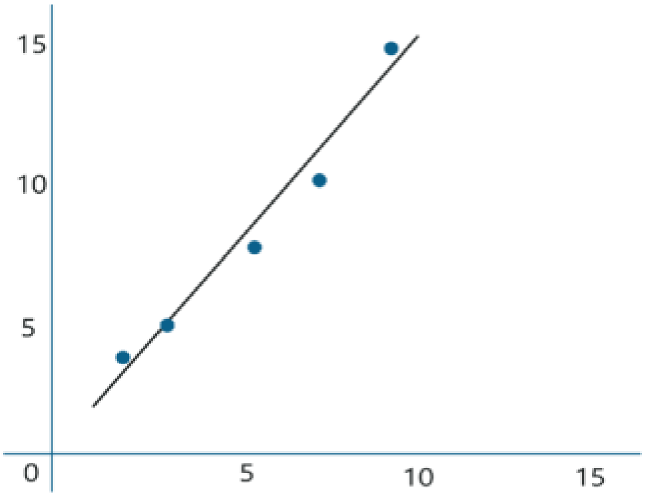
After you substitute the respective values, c = 0.305 approximately.

Step 3: Substitute the values in the final equation

$$y = mx + c$$

Price of T-shirts in dollars (x)	# of T-shirts sold (y)	Y=mx+c	error
2	4	3.3	-0.67
3	5	4.9	-0.14
5	7	7.9	0.89
7	10	10.9	0.93
9	15	13.9	-1.03

Let's construct a graph that represents the **y=mx + c** line of best fit:



Now Tom can use the above equation to estimate how many T-shirts of price $8 can he sell at the retail shop.

y = 1.518 x 8 + 0.305 = 12.45 T-shirts

This comes down to 13 T-shirts!

Dataset:

The data set contains the following variables:

- **Gender:** Male or female represented as binary variables
- **Age:** Age of an individual
- **Head size in cm^3:** An individuals head size in cm^3
- **Brain weight in grams:** The weight of an individual's brain measured in grams

These variables need to be analyzed in order to build a model that studies the relationship between the head size and brain weight of an individual.

Step 1: Import the required libraries

Step 2: Import the data set

Step 3: Assigning 'X' as independent variable and 'Y' as dependent variable

Step 4: Calculate the values of the slope and y-intercept

Step 5: Plotting the line of best fit

Step 6: Model Evaluation

Implementation:

1. **Simple Linear Regression**

```
import numpy as np


class LinearRegression:
    def __init__ (self):
        self.b_0 = 0
        self.b_1 = 0


    def fit (self, X, y):
        X_mean = np.mean (X)
        y_mean = np.mean (y)
        ssxy, ssx = 0, 0
        for _ in range (len (X)):
```

```
            ssxy += (X[_]-X_mean)*(y[_]-y_mean)
            ssx += (X[_]-X_mean)**2

        self.b_1 = ssxy / ssx
        self.b_0 = y_mean - (self.b_1*X_mean)
        return self.b_0, self.b_1

    def predict (self, X):
        y_hat = self.b_0 + (X * self.b_1)
        return y_hat

if __name__ == '__main__':
    X = np.array ([173, 182, 165, 154, 170], ndmin=2)
    X = X.reshape(5, 1)
    y = np.array ([68, 79, 65, 57, 64])
    model = LinearRegression ()
    model.fit (X, y)
    y_pred = model.predict ([161])
    print (y_pred)
```

```
[60.85051546]
```

2. **Multiple Linear Regression**

```
import numpy as np

class LinearRegression:
    def __init__ (self):
        self.params = np.zeros(int(np.random.random()), float)[:,np.newaxis]

    def fit (self, X, y):
        bias = np.ones (len (X))
        X_bias = np.c_[bias, X]
```

```
        lse = (np.linalg.inv (np.transpose(X_bias) @ X_bias) @ np.transpose
(X_bias)) @ y
        self.params = lse
        return self.params

    def predict (self, X):
        bias_testing = np.ones (len (X))
        X_test = np.c_[bias_testing, X]
        y_hat = X_test @ self.params
        return y_hat

if __name__ == '__main__':
    X = np.array ([
        [1, 4],
        [2, 5],
        [3, 8],
        [4, 2]
    ])

    y = np.array ([1, 6, 8, 12])

    model = LinearRegression ()
    parameters = model.fit (X, y)
    print (f'The parameters for the model are : {parameters}')

    y_pred = model.predict ([[5, 3]])
    print (f'The predicted outcome is : {y_pred}')
```

```
The parameters for the model are : [-1.69945355  3.48360656 -0.05464481]
The predicted outcome is : [15.55464481]
```

Conclusion:

The Least Squares Method is a fundamental technique in regression analysis, aiming to find the best-fitting line through a set of data points by minimizing the sum of the squared differences between observed and predicted values. It operates on the principle of minimizing the residual sum of squares, which effectively means finding the line that minimizes the overall distance between each data point and the regression line. This method is widely used due to its simplicity, ease of implementation, and mathematical elegance, making it a cornerstone in statistical modeling and predictive analytics. However, it's essential to note its sensitivity to outliers, as extreme values can disproportionately influence the resulting regression line, potentially leading to skewed interpretations of the relationship between variables.