Experiment No.3
Implement R Program for Multiple Linear Regression.
Date of Performance:
Date of Submission:

Experiment No - 3

Aim: Implement R Program for Multiple Linear Regression.

Objective: To understand the use of Multiple linear regression techniques by implementing a predefined dataset of R Studio.

Description-

Multiple linear regression is the extension of linear regression in the relationship between more than two variables. In simple linear regression, we have one predictor and one response variable. But in multiple regressions, we have more than one predictor variable and one response variable.

There is the following general mathematical equation for multiple regression -

$$y=b_0+b_1*x_1+b_2*x_2+b_3*x_3+\cdots b_n*x_n$$

Here,

- y is a response variable.
- **bO, bl, b2...bn** are the coefficients.
- **xl, x2, ...xn** are the predictor variables.

In R, we create the regression model with the help of the **Im()** function. The model will determine the value of the coefficients with the help of the input data. We can predict the value of the response variable for the set of predictor variables using these coefficients.

There is the following syntax of lm() function in multiple regression

1. lm(y x l+x2+x3 ... , data)

Before proceeding further, we first create our data for multiple regression. We will use the "mtcars" dataset present in the R environment. The main task of the model is to create the relationship between the "mpg" as a response variable with "wt", "disp" and "hp" as predictor variables.

For this purpose, we will create a subset of these variables from the "mtcars" dataset.

```
1. data<-mtcars[,c("mpg","wt","disp","hp")]
2. print(head(input)
```

Program:

```
import numpy as np

class MultipleRegression:
    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]
        inner_part = np.transpose (X_bias) @ X_bias
        inverse_part = np.linalg.inv (inner_part)
        outer_part = inverse_part @ np.transpose (X_bias)
        lse = outer_part @ 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 = MultipleRegression ()
    parameters = model.fit (X, y)
    print (f'The parameters are {parameters}')
    y_pred = model.predict ([[5, 3]])
    print (f'The predicted value is {y_pred}')
```

Output:

```
The parameters are [-1.69945355  3.48360656 -0.05464481]
The predicted value is [15.55464481]
```

Conclusion-

1. **Equation for multiple linear regression is** y=b0+b1*x1+b2*x2+b3*x3+⋯bn*xn
2. **When there is only one dependent variable and multiple independent variable then this types of regression is known as** Ordinary least-squares(OLS) Regression
3. **How to check inbuilt dataset in R studio?**

   In R, you can list available datasets using the 'data()' function, and then access individual datasets with their names, like 'data("mtcars")'. In Python, many datasets are available through libraries like scikit-learn or seaborn, and you can typically find information on accessing them in the respective library documentation.