**EXPT NO: 1 A python program to implement univariate regression DATE:22/08/2024 bivariate regression and multivariate regression.**

# AIM:

To write a python program to implement univariate regression, bivariate regression and multivariate regression.

# PROCEDURE:

Implementing univariate, bivariate, and multivariate regression using the Iris dataset involve the following steps:

## Step 1: Import Necessary Libraries

### First, import the libraries that are essential for data manipulation, visualization, and model building.

import numpy as np import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

## Step 2: Load the Iris Dataset

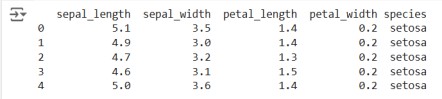
### The Iris dataset can be loaded and display the first few rows of the dataset .

# Load the Iris dataset

iris = sns.load\_dataset('iris')

# Display the first few rows of the dataset print(iris.head())

# OUTPUT :



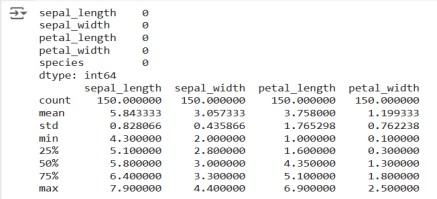
## Step 3: Data Preprocessing

### Ensure the data is clean and ready for modeling. Since the Iris dataset is clean, minimal preprocessing is needed.

# Check for missing values print(iris.isnull().sum())

# Display the basic statistical details print(iris.describe())

# OUTPUT :



## Step 4: Univariate Regression

Univariate regression involves predicting one variable based on a single predictor.

## : Select the Features

### Choose one feature (e.g., sepal\_length) and one target variable (e.g., sepal\_width).

X\_uni = iris[['sepal\_length']] y\_uni = iris['sepal\_width']

## : Split the Data

### Split the data into training and testing sets.

Fit the linear regression model on the training data.

X\_uni\_train, X\_uni\_test, y\_uni\_train, y\_uni\_test = train\_test\_split(X\_uni, y\_uni,

test\_size=0.2, random\_state=42)

## : Train the model

uni\_model = LinearRegression() uni\_model.fit(X\_uni\_train, y\_uni\_train)



## : Make Predictions

### Use the model to make predictions on the test data.

y\_uni\_pred = uni\_model.predict(X\_uni\_test)

## : Evaluate the Model

### Evaluate the model performance using metrics like Mean Squared Error (MSE) and R-squared.

print(f'Univariate MSE: {mean\_squared\_error(y\_uni\_test, y\_uni\_pred)}') print(f'Univariate R-squared: {r2\_score(y\_uni\_test, y\_uni\_pred)}')

# OUTPUT :



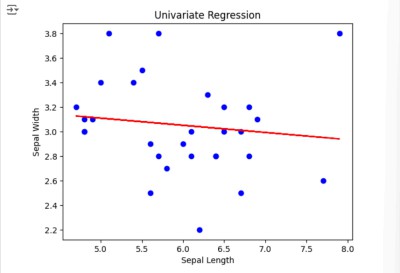
## : Visualize the Results

### Visualize the relationship between the predictor and the target variable.

plt.scatter(X\_uni\_test, y\_uni\_test, color='blue') plt.plot(X\_uni\_test, y\_uni\_pred, color='red') plt.xlabel('Sepal Length')

plt.ylabel('Sepal Width') plt.title('Univariate Regression') plt.show()

# OUTPUT :



## Step 5 : Bivariate Regression

Bivariate regression involves predicting one variable based on two predictors.

## : Select the Features

### Choose two features (e.g., sepal\_length, petal\_length) and one target variable (e.g., sepal\_width).

X\_bi = iris[['sepal\_length', 'petal\_length']] y\_bi = iris['sepal\_width']

## : Split the Data

### Split the data into training and testing sets.

X\_bi\_train, X\_bi\_test, y\_bi\_train, y\_bi\_test = train\_test\_split(X\_bi, y\_bi,

test\_size=0.2, random\_state=42)

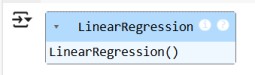
## : Train the Model

### Fit the linear regression model on the training data.

bi\_model = LinearRegression()

bi\_model.fit(X\_bi\_train, y\_bi\_train)

# OUTPUT :



## : Make Predictions

### Use the model to make predictions on the test data.

y\_bi\_pred = bi\_model.predict(X\_bi\_test)

## : Evaluate the Model

### Evaluate the model performance using metrics like MSE and R-squared.

print(f'Bivariate MSE: {mean\_squared\_error(y\_bi\_test, y\_bi\_pred)}') print(f'Bivariate R-squared: {r2\_score(y\_bi\_test, y\_bi\_pred)}')

OUTPUT :



## : Visualize the Results

### Since visualizing in 3D is challenging, we can plot the relationships between the target and each predictor separately.

# Sepal Length vs Sepal Width plt.subplot(1, 2, 1)

plt.scatter(X\_bi\_test['sepal\_length'], y\_bi\_test, color='blue') plt.plot(X\_bi\_test['sepal\_length'], y\_bi\_pred, color='red') plt.xlabel('Sepal Length')

plt.ylabel('Sepal Width')

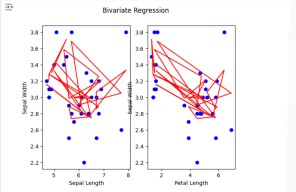
# Petal Length vs Sepal Width plt.subplot(1, 2, 2)

plt.scatter(X\_bi\_test['petal\_length'], y\_bi\_test, color='blue') plt.plot(X\_bi\_test['petal\_length'], y\_bi\_pred, color='red') plt.xlabel('Petal Length')

plt.ylabel('Sepal Width')

plt.suptitle('Bivariate Regression') plt.show()

# OUTPUT :



## Step 6: Multivariate Regression

Multivariate regression involves predicting one variable based on multiple predictors.

## : Select the Features

### Choose multiple features (e.g., sepal\_length, petal\_length, petal\_width) and one target variable (e.g., sepal\_width).

X\_multi = iris[['sepal\_length', 'petal\_length', 'petal\_width']] y\_multi = iris['sepal\_width']

## : Split the Data

### Split the data into training and testing sets.

X\_multi\_train, X\_multi\_test, y\_multi\_train, y\_multi\_test =

train\_test\_split(X\_multi,

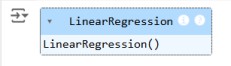
y\_multi, test\_size=0.2, random\_state=42)

## : Train the Model

### Fit the linear regression model on the training data.

multi\_model = LinearRegression() multi\_model.fit(X\_multi\_train, y\_multi\_train)

# OUTPUT :



## : Make Predictions

### Use the model to make predictions on the test data.

y\_multi\_pred = multi\_model.predict(X\_multi\_test)

## : Evaluate the Model

### Evaluate the model performance using metrics like MSE and R-squared.

print(f'Multivariate MSE: {mean\_squared\_error(y\_multi\_test, y\_multi\_pred)}') print(f'Multivariate R-squared: {r2\_score(y\_multi\_test, y\_multi\_pred)}')

# OUTPUT :



## Step 7: Visualize the multivariate regression

plt.figure(figsize=(15,4)) plt.subplot(1, 2, 1)

plt.scatter(X\_multi\_test['sepal\_length'], y\_multi\_test, color='blue') plt.plot(X\_multi\_test['sepal\_length'], y\_multi\_pred, color='red') plt.xlabel('sepal\_length')

plt.ylabel('sepal\_width')

plt.title('Multivariate Regression-1') plt.show()

plt.figure(figsize=(15,4)) plt.subplot(1, 2, 1)

plt.scatter(X\_multi\_test['petal\_length'], y\_multi\_test, color='blue') plt.plot(X\_multi\_test['petal\_length'], y\_multi\_pred, color='red') plt.xlabel('petal\_length')

plt.ylabel('sepal\_width')

plt.title('Multivariate Regression-2') plt.show()

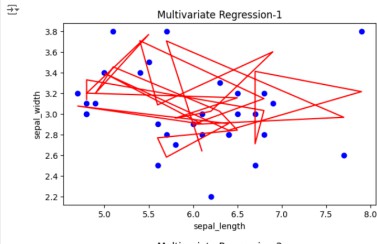
plt.figure(figsize=(15,4)) plt.subplot(1, 2, 2 )

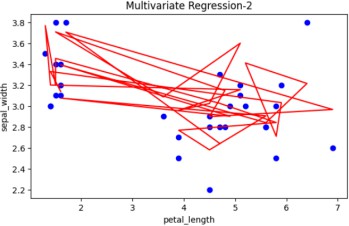
plt.scatter(X\_multi\_test['petal\_length'], y\_multi\_test, color='blue') plt.plot(X\_multi\_test['petal\_length'], y\_multi\_pred, color='red') plt.xlabel('petal\_length')

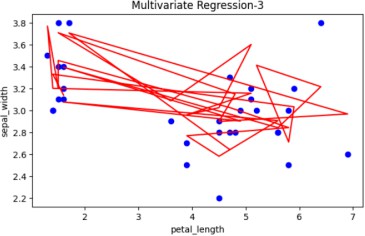
plt.ylabel('sepal\_width')

plt.title('Multivariate Regression-3') plt.show()

# OUTPUT :







## Step 8: Interpret the Results

### After implementing and evaluating the models, interpret the coefficients to understand the influence of each predictor on the target variable.

print('Univariate Coefficients:', uni\_model.coef\_) print('Bivariate Coefficients:', bi\_model.coef\_)

print('Multivariate Coefficients:', multi\_model.coef\_)

**OUTPUT :**



# RESULT:

This step-by-step process will help us to implement univariate, bivariate, and multivariate regression models using the Iris dataset and analyze their performance.