Experiment 1

Title: To implement Linear Regression

Lab Objective: To acquire fundamental knowledge of developing machine learning models.

Theory:
Linear regression is a simple supervised learning method that models the relationship between a dependent variable and one or more independent variables using a straight line: $Y = \beta_0 + \beta_1 X$.
It works by finding the line that minimizes the squared differences between actual and predicted values, using the Least Squares method. It's commonly used for prediction, trend analysis, and forecasting.

- With one independent variable, it's Simple Linear Regression.
- With multiple variables, it's Multiple Linear Regression:

  $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta\square X\square$

Where:

- Y = predicted value
- β_0 = intercept
- β_i = coefficient for X_i (independent variable)
- X_i = i-th input variable

The model works by finding the best-fitting line through the data points using the Least Squares method, which minimizes the sum of squared residuals (difference between actual and predicted values).

Applications of Linear Regression:

- ●Estimating Salaries: Used to predict income based on factors like work experience or educational background.
- ● Forecasting Sales and Demand: Supports inventory planning and financial forecasting.
- ● Analyzing Economics and Healthcare: Applied in trend analysis, evaluating treatments, and building financial models.
- ● Assessing Risks and Allocating Resources: Useful in insurance, budgeting projects, and estimating costs

Advantages of Linear Regression:

- ● **Straightforward and Easy to Use**: Simple to grasp and interpret, making it great for beginners and basic analysis.
- ● **Fast and Lightweight**: Handles large datasets quickly thanks to its simple

mathematical foundation.

- **Meaningful Interpretation**: Coefficients clearly show the impact of each variable on the result, aiding informed decisions.
- **Reliable for Forecasting**: Commonly applied to predict trends in areas such as business, finance, and healthcare.

Limitations of Linear Regression:

- Limited to Linear Patterns: Struggles to capture complex, non-linear relationships.
- Outlier Sensitivity: Extreme values can significantly skew the results.
- Strong Assumptions: Requires constant variance and independence; violations can lead to biased predictions.
- Not Suitable for Classification: Designed for continuous outputs, not for categorical predictions.

Problem:

To implement a Linear Regression model that predicts the salary of an employee based on their years of experience.

Steps:

Step 1: Importing Libraries and Dataset
We begin by importing the required libraries such as NumPy, Pandas, and Matplotlib. The dataset (Salary_Data.csv) is then loaded using pandas.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# importing the dataset
dataset = pd.read_csv('Salary_Data.csv')
print(dataset.head())
```

Step 2: Data Preprocessing
The dataset is divided into independent variables (X) and dependent variables (y). The independent variable (years of experience) must be stored in an array, while the dependent variable (salary) must be stored in a vector.

```
X = dataset.iloc[:, :-1].values # independent variable array
Y = dataset.iloc[:, 1].values # dependent variable vector
```

Step 3: Splitting the Dataset

The dataset is split into training set and test set, commonly using an 80-20 or 70-30 ratio. The training set is used to train the model, while the test set is used to evaluate its accuracy. If predictions on the test set match closely with actual values, the model is considered reliable.

```
from sklearn.model_selection import train_test_split
X_train, X_test,y_train, y_test= train_test_split(X, Y, test_size=1/3)
```

Step 4: Training the Model

We use the LinearRegression class from scikit-learn to create a regression model. The model is trained (fitted) on the training data so it can learn the relationship between years of experience and salary.

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

Step 5: Making Predictions

After training, the model predicts salaries for the test data using the predict() method. The predicted salaries (y_pred) are then compared with the actual salaries (y_test).

```
y_pred = regressor.predict(X_test)
print("Predicted Salaries:", y_pred)
print("Actual Salaries:", y_test)
```

Step 6: Visualizing the Results

Finally, results are visualized using Matplotlib. A scatter plot is created with actual data points (red) and the regression line (blue). The X-axis represents years of experience and the Y-axis represents salaries, showing how well the regression line fits the data.

```
# Visualizing Training set results: plotting training
points plt.scatter(X_train, y_train, color='red')
plt.plot(X_train, regressor.predict(X_train), color='blue')
plt.title("Salary vs Experience (Training set)by Pratham
Singh") plt.xlabel("Years of Experience")
plt.ylabel("Salary")
plt.show()

# Visualizing Test set results: plotting test points
plt.scatter(X_test, y_test, color='red')
plt.plot(X_train, regressor.predict(X_train),
color='blue') plt.title("Salary vs Experience (Test set)by
Pratham Singh") plt.xlabel("Years of Experience")
plt.ylabel("Salary")
plt.show()
```

Step 7: Evaluating Model Accuracy

To check how well the linear regression model predicts salaries, we calculate the accuracy/error metrics. Common metrics include:

R^2 Score → proportion of variance in the dependent variable explained by the model.

Mean Absolute Error (MAE) → average absolute difference between predicted and actual salaries.

Mean Squared Error (MSE) → average of squared differences (penalizes large errors more). Root Mean Squared Error (RMSE) → square root of MSE, in the same unit as salary.

```
from sklearn.metrics import r2_score, mean_absolute_error,
mean_squared_error
# R² Score, MAE, MSE, RMSE
r2 = r2_score(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
print("R² Score:", r2)
print("Mean Absolute Error:", mae)
print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
```

Prerequisite Software and Command:

- python 3 and above
- pip install numpy
- pip install pandas
- pip install matplotlib
- pip install -U scikit-learn

Program Code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score,
mean_absolute_error,  mean_squared_error

# importing the dataset
dataset = pd.read_csv('Salary_Data.csv')
print("Dataset head:")
print(dataset.head())
```

```
# data preprocessing
X = dataset.iloc[:, :-1].values # independent variable array
y = dataset.iloc[:, 1].values # dependent variable vector

# splitting the dataset into training and testing sets X_train,
X_test, y_train, y_test = train_test_split(X, y,  test_size=0.10,
random_state=0)
# fitting the regression model to training data
regressor = LinearRegression()
regressor.fit(X_train, y_train)

# predicting the test result
y_pred = regressor.predict(X_test)
print("Predicted Salaries:", y_pred)
print("Actual Salaries:", y_test)

# Visualizing Training set results: plotting training
points plt.scatter(X_train, y_train, color='red')
plt.plot(X_train, regressor.predict(X_train), color='blue')
plt.title("Salary vs Experience (Training set)by Pratham
Singh") plt.xlabel("Years of Experience")
plt.ylabel("Salary")
plt.show()

# Visualizing Test set results: plotting test points
plt.scatter(X_test, y_test, color='red')
plt.plot(X_train,              regressor.predict(X_train),
color='blue')    plt.title("Salary  vs  Experience  (Test
set)by Pratham Singh") plt.xlabel("Years of Experience")
plt.ylabel("Salary")
plt.show()

# R² Score
r2 = r2_score(y_test, y_pred)
# MAE, MSE, RMSE
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
print("R² Score:", r2)
print("Mean Absolute Error:", mae)
print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
```

Program Output:

```
Dataset head:
   YearsExperience  Salary
0              1.1   39343
1              1.5   43205
2              2.0   40731
3              2.2   43525
4              3.0   60150
```

```
Predicted Salaries: [44475.75300851 62415.09452299 75633.55669154]
Actual Salaries: [43525. 63218. 83088.]
```


```
R² Score: 0.960704326863785
Mean Absolute Error: 5907.968215279128
Mean Squared Error: 45108882.58470911
Root Mean Squared Error: 6716.314657958567
```

Conclusion: Linear Regression model implemented with experiential experimental model on given data set of Salary Data csv file for prediction of salary and experience.