Exp6:- Implementation of Prediction Algorithm (Linear Regression)

Aim:

To implement a **prediction algorithm** using **Linear Regression** in Python for understanding how to model the relationship between independent and dependent variables.

Introduction:

Linear Regression is one of the simplest and most commonly used prediction algorithms in machine learning. It attempts to model the relationship between two variables by fitting a linear equation to observed data. The goal is to predict the dependent variable (output) based on the value of one or more independent variables (input).

The equation for **Simple Linear Regression** is:

y=mx+cy = mx + cy=mx+c

Procedure

1. Import necessary libraries.
2. Load or create the dataset.
3. Split the dataset into training and testing sets.
4. Fit a linear regression model using the training data.
5. Predict the values using the testing data.
6. Evaluate the model using metrics like **Mean Squared Error (MSE)** and **R^2 Score**.
7. Visualize the regression line.

Program Code:-

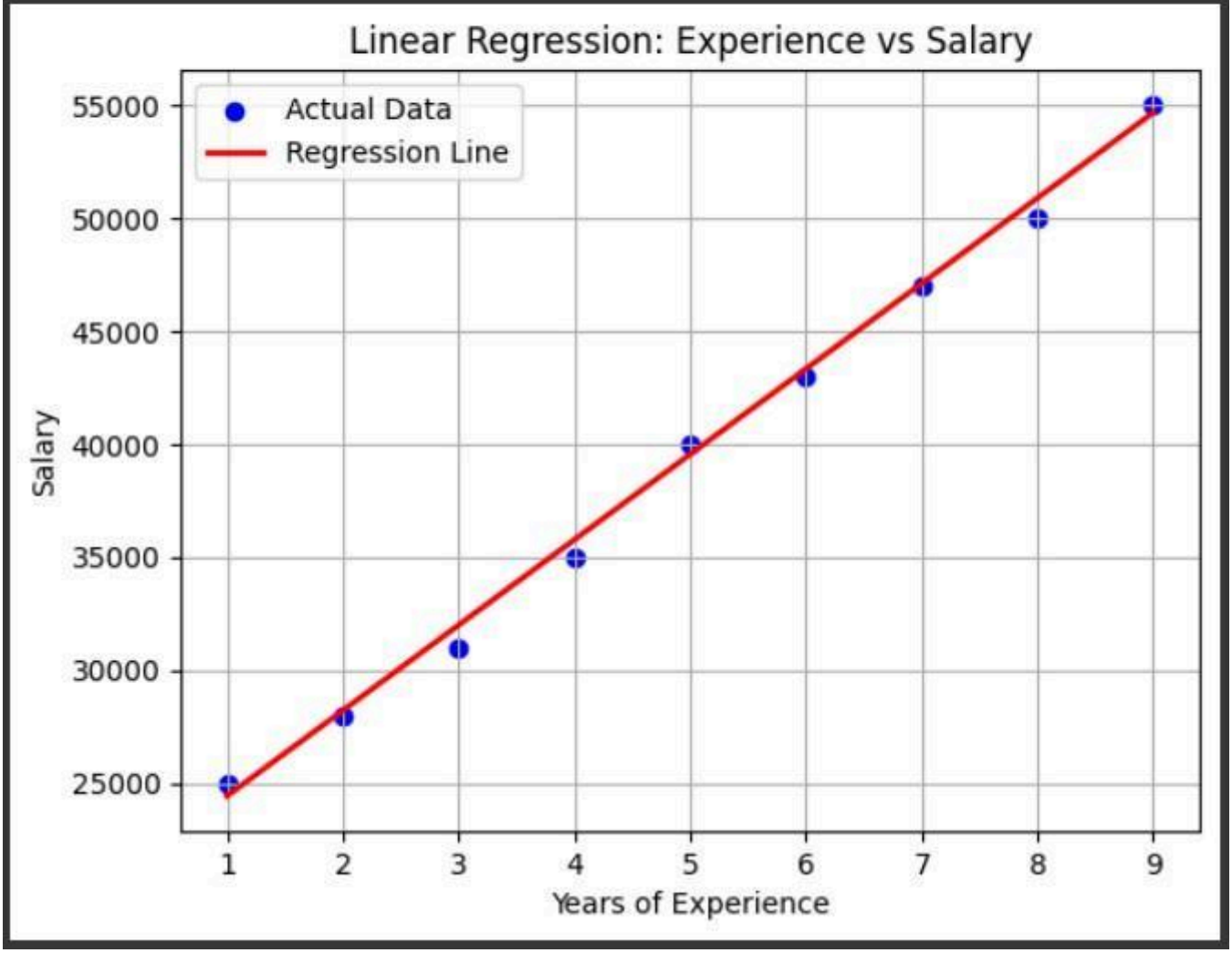
```
import numpy as np

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


X = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9]).reshape(-1, 1) # Years of Experience

y = np.array([25000, 28000, 31000, 35000, 40000, 43000, 47000, 50000, 55000]) # Salary


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

```
model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

from IPython.display import display, Markdown
display(Markdown("**Implementation/Output snap shot:**"))

print("Actual values:", y_test)
print("Predicted values:", y_pred.astype(int))

# Evaluating the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error (MSE):", round(mse, 2))
print("R² Score:", round(r2, 2))

# Plotting the regression line
plt.scatter(X, y, color='blue', label='Actual Data')
plt.plot(X, model.predict(X), color='red', linewidth=2, label='Regression Line')
plt.title("Linear Regression: Experience vs Salary")
plt.xlabel("Years of Experience")
plt.ylabel("Salary")
plt.legend()
plt.grid(True)
plt.show()
```

Output

Implementation/Output snap shot:

```
Actual values: [50000 31000]
Predicted values: [50871 31984]
Mean Squared Error (MSE): 865021.75
R² Score: 0.99
```



Review Questions & Answers

Q1. What are the key steps involved in implementing a simple linear regression model using Python and scikit-learn?

Ans:

1. Import necessary libraries.
2. Prepare and split the dataset.
3. Create and train the LinearRegression model.
4. Make predictions.
5. Evaluate model performance using metrics.

Q2. How can you evaluate the performance of a linear regression model in Python? List and explain at least two metrics.

Ans:

1. Mean Squared Error (MSE): Measures the average squared difference between actual and predicted values. Lower values indicate better performance.
2. R^2 Score: Represents how well the regression model fits the data. A value close to 1 indicates a strong correlation between input and output.

Q3. What is the role of the train_test_split() function in building a linear regression model, and why is it important?

Ans:

The train_test_split() function divides the dataset into training and testing sets. This is important to evaluate the model's performance on unseen data and prevent overfitting.

Conclusion

In this experiment, we successfully implemented the Linear Regression algorithm using Python and scikit-learn. We learned how to train the model, make predictions, and evaluate its performance using metrics like Mean Squared Error and R^2 Score. Visualization of the regression line helped in understanding the fitting of the model to the dataset.

GIthub link:-https://github.com/Pralix20/DWMexp