Name : Rushi Daulatkar
Roll Number : 59

Aim:

To implement and understand linear regression using Python, focusing on data modeling, prediction, and visualizing results.

In [12]:
```
import numpy as np  # For array operations
import matplotlib.pyplot as plt  # For plotting graphs
from sklearn.linear_model import LinearRegression  # Linear Regression model
from sklearn.model_selection import train_test_split  # To split data into training and testing
from sklearn.metrics import mean_squared_error, r2_score  # For model evaluation
```

In [13]:
```
# Step 1: Creating sample data (X: independent, y: dependent)
X = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9], dtype=float).reshape(-1, 1)  # Use native float
y = np.array([1, 2, 3, 4, 5, 6, 7, 8, 10], dtype=float)  # Use native float
```

In [14]:
```
# Step 2: Splitting the data into training and test sets
# Use native Python float and int instead of Sage-specific types
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=float(0.2), random_state=int(0))
```

In [15]:
```
# Step 3: Creating and training the Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)  # Fitting the model
```

Out[15]:

LinearRegression

LinearRegression()

In [16]:
```
# Displaying model parameters
print("Intercept (b0):", model.intercept_)
print("Coefficient (b1):", model.coef_[0])
```

```
Intercept (b0): -0.286585365853659
Coefficient (b1): 1.0884146341463417
```

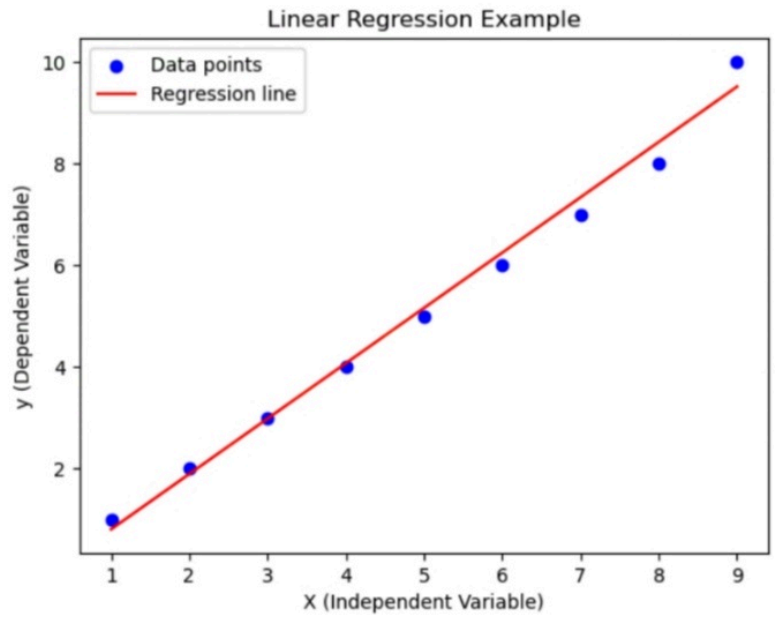
In [17]:
```
# Step 4: Making predictions on the test set
y_pred = model.predict(X_test)
```

In [18]:
```
# Step 5: Evaluating the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print("Mean Squared Error (MSE):", mse)
print("R² Score:", r2)
```

```
Mean Squared Error (MSE): 0.08873531380130918
R² Score: 0.9858023497917905
```

In [19]:
```
# Step 6: Visualizing the regression line
plt.scatter(X, y, color="blue", label="Data points")  # Scatter plot of original data
plt.plot(X, model.predict(X), color="red", label="Regression line")  # Regression line
plt.title("Linear Regression Example")
plt.xlabel("X (Independent Variable)")
plt.ylabel("y (Dependent Variable)")
plt.legend()
plt.show()
```

Out[19]:



Conclusion

In this practical, we implemented linear regression using Python. We split the data into training and test sets, trained a model, made predictions, and evaluated the model using metrics like MSE and R^2. Finally, we visualized the regression line along with the data points.

Code By : Rushi Daulatkar