Exercise 4a Date: 22/01/2025

Linear Regression - Bivariate dataset

Aim: To build a Linear Regression model to predict the target variable in a bivariate dataset.

Algorithm:

Linear Regression is a supervised learning algorithm used for predicting a continuous target variable. It works by modelling the relationship between the dependent variable (target) and one or more independent variables (features) using a linear equation of the form:

$y = mx + b$

Where:

- y is the predicted value (target).
- m is the slope (coefficient) of the line, indicating how the target changes with a one-unit change in the feature.
- x is the feature (input variable).
- b is the intercept, indicating the value of y when x = 0.

The algorithm uses the Ordinary Least Squares (OLS) method to minimize the sum of squared errors between the actual and predicted values to find the best-fitting line.

Step 1: Import Libraries

- Import necessary Python libraries such as pandas, numpy, matplotlib, and LinearRegression from scikit-learn for model building.

Step 2: Load the Dataset

- Load the dataset using pd.read_csv() and check the structure and details of the dataset using df.info().

Step 3: Prepare the Data

- Separate the feature (independent variable) and target (dependent variable).
- In this case, X is the feature (column 1), and y is the target (column 2).
- Reshape X to a 2D array as required by scikit-learn's model.

Step 4: Train-Test Split

- Split the dataset into training and testing sets using train_test_split. Typically, 80% of the data is used for training, and 20% is used for testing.

Step 4: Train the Model

- Initialize the Linear Regression model and fit it on the training data (X_train, y_train).
- The model learns the best-fit line by minimizing the sum of squared errors between actual and predicted values.

Step 5: Make Predictions

- After training the model, use it to make predictions on the test data (X_test).

Step 6: Evaluate the Model

- Evaluate the model's performance using key metrics such as R-squared and Mean Squared Error (MSE).
- R-squared indicates how well the model explains the variance in the target variable, with values closer to 1 being better.
- MSE calculates the average squared difference between the actual and predicted values, with lower values indicating better predictions.

Step 7: Visualize the Results

- Create a scatter plot comparing the actual vs. predicted values, and draw the best-fit line on the plot.

Import the libraries

In [2]:

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

Load the Dataset

In [3]:

```
df = pd.read_csv("LR_data.csv",sep=',')
```

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   x_val   1000 non-null   float64
 1   y_val   1000 non-null   float64
dtypes: float64(2)
memory usage: 15.8 KB
```

Split features and target

In [6]:

```
X = df.iloc[:, 0].values.reshape(-1, 1)
y = df.iloc[:, 1].values
```

Split dataset into train and test data

In [7]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Apply Linear Regression

In [8]:

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

Out[8]:

```
▾ LinearRegression
LinearRegression()
```

In [9]:

```
y_pred = model.predict(X_test)
```

Performance metrics

In [11]:

```
r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
```

In [12]:

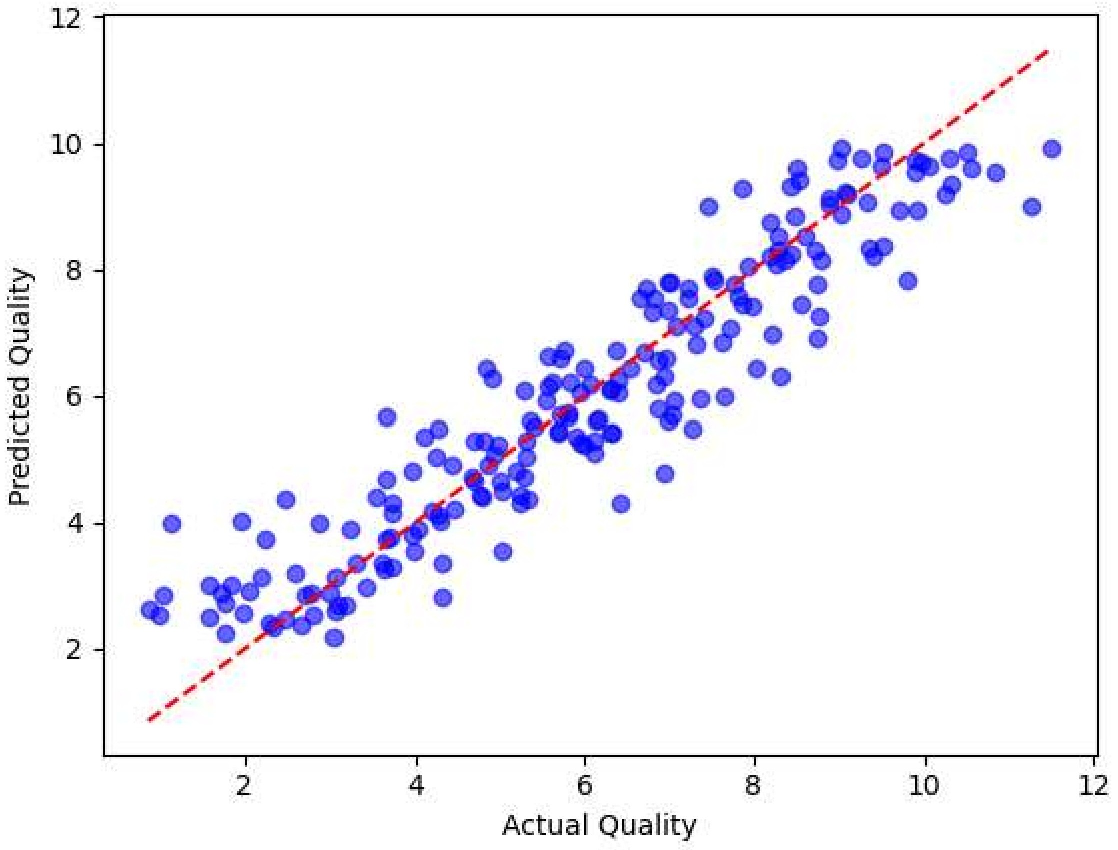
```
print(f'R-squared: {r2:.4f}')
print(f'Mean Squared Error: {mse:.4f}')
```

```
R-squared: 0.8724
Mean Squared Error: 0.7779
```

Plot the graph

In [13]:

```
plt.scatter(y_test, y_pred, color='blue', alpha=0.6)
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red', linestyle='--')
plt.xlabel('Actual Quality')
plt.ylabel('Predicted Quality')
plt.show()
```



Result

A Linear Regression model was built to predict the target based on the given feature, achieving an R-squared value of 0.8724 and a Mean Squared Error (MSE) of 0.7779.