**1. What is Regression?**

**Regression** is a statistical method used to determine the relationship between a dependent variable (target) and one or more independent variables (predictors). The main goal of regression is to predict the value of the dependent variable based on the values of independent variables. It helps in identifying trends and making forecasts based on data.

**2. What is a Dependent Variable?**

A **dependent variable** (also called the target or response variable) is the variable we aim to predict or explain in a regression analysis. It depends on one or more independent variables. For example, in predicting house prices based on various features like area and number of rooms, the house price is the dependent variable.

**3. What are Independent Variables?**

**Independent variables** (also called features, predictors, or explanatory variables) are the variables that influence or predict the dependent variable. In a regression analysis, changes in independent variables are used to predict changes in the dependent variable. For example, in predicting house prices, the area, number of rooms, and location are independent variables.

**4. What is Linear Regression?**

**Linear Regression** is a type of regression analysis where the relationship between the dependent variable YYY and one or more independent variables XXX is modeled as a linear equation. The general form of a simple linear regression model is:

Y=mX+cY = mX + cY=mX+c

Where:

* YYY is the dependent variable.
* XXX is the independent variable.
* mmm is the slope or coefficient.
* ccc is the Y-intercept.

In multiple linear regression, it extends to:

Y=m1X1+m2X2+…+mnXn+cY = m\_1X\_1 + m\_2X\_2 + \ldots + m\_nX\_n + cY=m1​X1​+m2​X2​+…+mn​Xn​+c

**5. Which Model is Used for Regression?**

Several models can be used for regression analysis, including:

* **Linear Regression Model**: For simple or multiple linear relationships.
* **Polynomial Regression**: When the relationship between variables is non-linear but can be approximated using polynomial terms.
* **Ridge Regression**: A type of linear regression that includes a regularization term to prevent overfitting.
* **Lasso Regression**: Similar to Ridge Regression but uses L1 regularization, which can shrink some coefficients to zero, effectively selecting features.
* **Support Vector Regression (SVR)**: Uses Support Vector Machine (SVM) principles for regression.
* **Decision Trees and Random Forest Regressors**: Non-linear models that are used to capture complex relationships.

**6. Which of the Following Metrics Can Be Used for Evaluating Regression Models?**

The most commonly used metrics for evaluating regression models include:

* **Mean Absolute Error (MAE)**: Measures the average absolute difference between actual and predicted values.
* **Mean Squared Error (MSE)**: Measures the average squared difference between actual and predicted values. It gives more weight to larger errors.
* **Root Mean Squared Error (RMSE)**: The square root of MSE, it is in the same unit as the dependent variable and helps interpret the error.
* **R-squared (R²)**: Represents the proportion of the variance in the dependent variable that is predictable from the independent variables. It ranges from 0 to 1, where 1 indicates a perfect fit.

 **: What is the purpose of using train\_test\_split()?**

* **A:** It is used to split the data into training and testing sets to evaluate the model's performance effectively.

 **Q: Why do we reshape the X values using reshape()?**

* **A:** The reshape function is used to transform the X values from a 1D array to a 2D array, as the regression model expects a 2D input.

 **Q: What does the fit() method do in Linear Regression?**

* **A:** It trains the model by finding the best-fit line based on the training data.

 **Q: What is the use of metrics.mean\_absolute\_error()?**

* **A:** It calculates the mean absolute error, which measures the average absolute difference between actual and predicted values.

 **Q: What is the difference between Mean Squared Error and Root Mean Squared Error?**

* **A:** Mean Squared Error (MSE) is the average squared difference between actual and predicted values, while Root Mean Squared Error (RMSE) is the square root of MSE, providing error in the same unit as the target variable.

 **Q: Why do we use sns.regplot() at the end of the code?**

* **A:** It is used to visualize the regression line along with the scatter plot of actual data points.

 **Q: What do the coef\_ and intercept\_ attributes represent?**

* **A:** coef\_ gives the slope (m) of the regression line, and intercept\_ gives the Y-intercept (c) of the regression line.

# Importing necessary libraries

import pandas as pd # Used for data manipulation and analysis

import numpy as np # Used for numerical operations

import matplotlib.pyplot as plt # Used for data visualization

import seaborn as sns # Used for statistical data visualization

# Loading the dataset

df = pd.read\_csv('Temperature.csv') # Reading the CSV file named 'Temperature.csv' into a DataFrame 'df'

print(df) # Printing the entire DataFrame to view its content

# Displaying the columns of the DataFrame

df.columns # Returns the list of column names in the DataFrame

# Checking the data types of each column

df.dtypes # Returns the data type of each column (e.g., int64, float64, object)

# Extracting the required columns for X and Y

x = df.iloc[:, 0].values # Selecting the first column as X (usually 'YEAR')

y = df.iloc[:, 13].values # Selecting the 14th column as Y (usually 'ANNUAL' temperature or any dependent variable)

# Visualizing the scatter plot of X vs Y

plt.scatter(x, y, color='b', marker='o', s=80) # Blue scatter plot with size 80

plt.show() # Display the scatter plot

# Checking the shape of X

x.shape # Returns the shape of the array (e.g., (117,))

# Reshaping X for fitting into the model

x = x.reshape(117, 1) # Reshaping X to be a 2D array with 117 rows and 1 column

# Displaying reshaped X

x # Output the reshaped X values

# Splitting the data into training and testing sets

from sklearn.model\_selection import train\_test\_split

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.25, random\_state=0)

# test\_size=0.25 means 25% of data is for testing, 75% is for training

# Creating and training the Linear Regression model

from sklearn.linear\_model import LinearRegression

regression = LinearRegression() # Creating an instance of LinearRegression

regression.fit(x\_train, y\_train) # Fitting the model with training data

# Getting the coefficient (slope) of the regression line

m = regression.coef\_ # Returns the slope of the regression line

print('Regression Coefficient/slope of regression line:', m)

# Getting the intercept of the regression line

c = regression.intercept\_ # Returns the intercept of the regression line

print('Intercept:', c)

# Predicting the test data using the trained model

y\_pred = regression.predict(x\_test) # Predicting the Y values for the X\_test data

print(y\_pred) # Printing the predicted Y values

# Predicting the value for a specific year, e.g., 2018

regression.predict([[2018]]) # Predicts the temperature for the year 2018

# Creating a DataFrame to compare Actual vs Predicted values

df1 = pd.DataFrame({'Actual value': y\_test.flatten(), 'Predicted Value': y\_pred.flatten()})

df1 # Displaying the DataFrame

# Manually verifying the first prediction using formula y = mx + c

x\_test[0] \* m + c # This should give a similar value as y\_pred[0]

# Visualizing the Actual vs Predicted values

plt.scatter(x\_test, y\_test, color='b') # Blue scatter plot for actual values

plt.plot(x\_test, y\_pred, color='r') # Red line plot for predicted values

plt.show() # Display the plot

# Evaluating the model using error metrics

from sklearn import metrics

print('Mean Absolute Error:', metrics.mean\_absolute\_error(y\_test, y\_pred)) # Mean Absolute Error

print('Mean Squared Error:', metrics.mean\_squared\_error(y\_test, y\_pred)) # Mean Squared Error

print('Root Mean Squared Error:', np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred))) # RMSE

# Plotting regression line using seaborn

sns.regplot(x='YEAR', y='ANNUAL', data=df) # Seaborn plot for regression line

plt.show() # Display the seaborn plot