Regression & Its Evaluation Answer

Question 1: What is Simple Linear Regression?
Ans :- **Simple Linear Regression** is a method used to model the relationship between two variables by fitting a straight line to the data. It predicts the value of one variable (**dependent variable**, Y) based on the value of another (**independent variable**, X).

- **One independent variable (X)** – also called the predictor or explanatory variable.
- **One dependent variable (Y)** – also called the response or outcome variable.

Equation of Simple Linear Regression:

Y=β0+β1X+εY = \beta_0 + \beta_1 X + \varepsilonY=β0+β1X+ε

- YYY: Dependent variable (what you want to predict)
- XXX: Independent variable (the input)
- β0\beta_0β0: Intercept (value of Y when X = 0)
- β1\beta_1β1: Slope (how much Y changes with a one-unit increase in X)
- ε\varepsilonε: Error term (difference between the actual and predicted Y)

Question 2: What are the key assumptions of Simple Linear Regression?
Ans :- **Key Assumptions of Simple Linear Regression:**

Simple Linear Regression relies on several important assumptions to ensure the model is valid and accurate:

Linearity

- The relationship between the independent variable (X) and the dependent variable (Y) is linear.

Independence of Errors

- The residuals (errors) are independent of each other (no autocorrelation).

Homoscedasticity

- The residuals have constant variance at all levels of X (i.e., spread of errors is even across the range of data).

Normality of Errors

- The residuals (differences between observed and predicted values) are normally distributed.

No Multicollinearity

- (Only applies in multiple regression, but worth noting) In simple linear regression, this is not a concern as there's only one independent variable.

No Significant Outliers

- Extreme outliers can distort the regression line and affect results.

Question 3: What is heteroscedasticity, and why is it important to address in regression models?

Ans :- **Heteroscedasticity** occurs when the **variance of the residuals (errors) is not constant** across all levels of the independent variable in a regression model.

In a well-behaved regression model, we expect **homoscedasticity**—i.e., residuals should spread out evenly. When this doesn't happen, and residuals **fan out or cluster**, it indicates heteroscedasticity.

it Important to Address :

1. **Violates Regression Assumptions**
    - One key assumption of linear regression is that residuals have **constant variance** (homoscedasticity). Heteroscedasticity breaks this assumption.
2. **Leads to Inefficient Estimates**
    - Coefficients may still be unbiased, but their **standard errors become unreliable**, which affects **confidence intervals and hypothesis tests**.
3. **Invalid Statistical Inference**
    - p-values and t-tests may become misleading, leading to incorrect conclusions about variable significance.

Question 4: What is Multiple Linear Regression?

Ans :- **Multiple Linear Regression (MLR)** is a statistical method used to model the relationship between **one dependent variable** and **two or more independent variables**.

Question 5: What is polynomial regression, and how does it differ from linear regression?

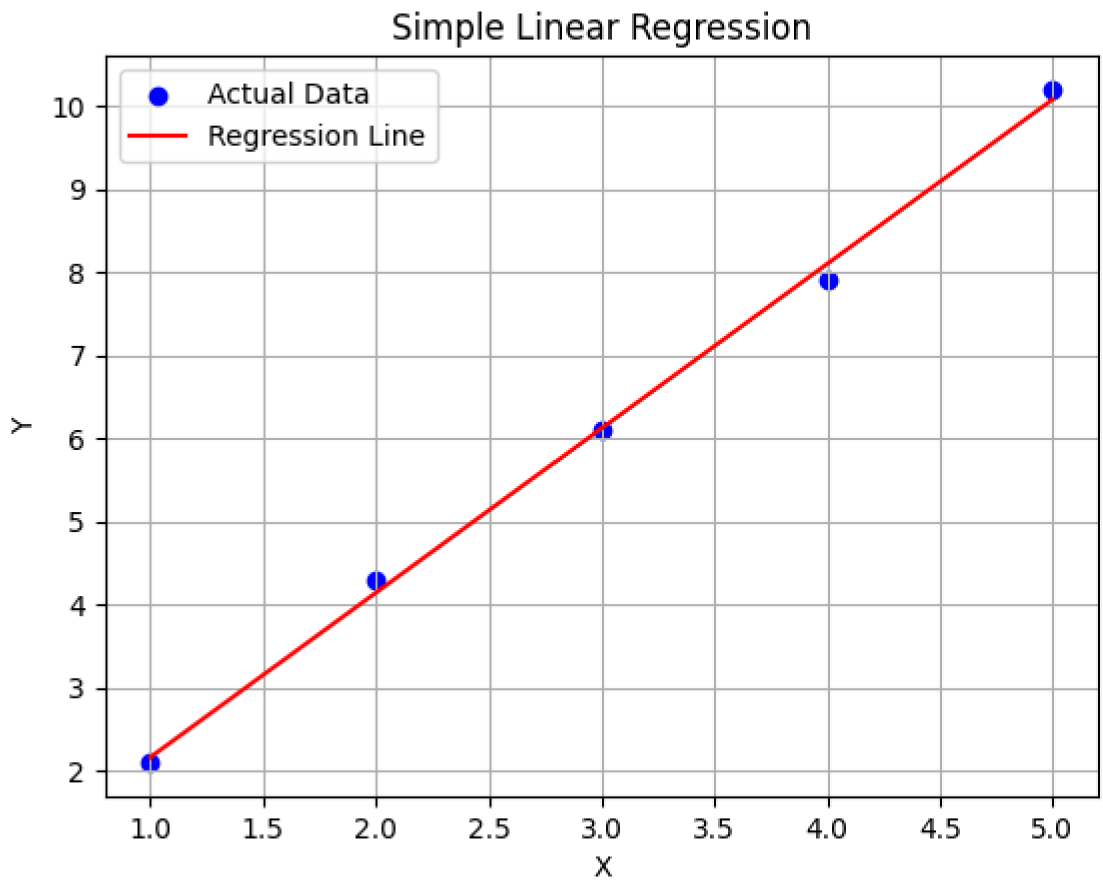
Ans :- Polynomial regression is a type of regression analysis where the relationship between the independent variable XXX and the dependent variable YYY is modeled as an **nth-degree polynomial**.

Feature	**Linear Regression**	**Polynomial Regression**
Relationship	Straight line	Curved line (nonlinear)
Equation form	Y=β0+β1XY = \beta_0 + \beta_1 X	Y=β0+β1X+β2X2+…Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \dots
Handles non-linearity	No	Yes
Flexibility	Less flexible (linear only)	More flexible for capturing complex patterns

Question 6: Implement a Python program to fit a Simple Linear Regression model to the following sample data: ● X = [1, 2, 3, 4, 5] ● Y = [2.1, 4.3, 6.1, 7.9, 10.2]

```
import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear_model import LinearRegression


# Sample data

X = [1, 2, 3, 4, 5]

Y = [2.1, 4.3, 6.1, 7.9, 10.2]

# Reshape X for sklearn. (needs 2D array)

X = np.array(X).reshape(-1, 1)

Y = np.array(Y)

# Create and train the model

model = LinearRegression()

model.fit(X, Y)
```

```
# Get model parameters

slope = model.coef_[0]

intercept = model.intercept_

# Print the regression equation

print(f"Regression Equation: Y = {intercept:.2f} + {slope:.2f}X")

# Predict values

Y_pred = model.predict(X)

# Plot the results

plt.scatter(X, Y, color='blue', label='Actual Data')

plt.plot(X, Y_pred, color='red', label='Regression Line')

plt.xlabel('X')

plt.ylabel('Y')

plt.title('Simple Linear Regression')

plt.legend()

plt.grid(True)

plt.show()
```



Y = 0.06 + 2.02X

Question 7: Fit a Multiple Linear Regression model on this sample data: ● Area = [1200, 1500, 1800, 2000] ● Rooms = [2, 3, 3, 4] ● Price = [250000, 300000, 320000, 370000]

```
import numpy as np

from sklearn.linear_model import LinearRegression


# Sample data

area = [1200, 1500, 1800, 2000]

rooms = [2, 3, 3, 4]

price = [250000, 300000, 320000, 370000]


# Combine features into a 2D array (X)
```

X = np.column_stack((area, rooms))

y = np.array(price)

Create and train the model

model = LinearRegression()

model.fit(X, y)

Get coefficients

intercept = model.intercept_

coefficients = model.coef_

Print the regression equation

print(f"Regression Equation: Price = {intercept:.2f} + ({coefficients[0]:.2f} × Area) + ({coefficients[1]:.2f} × Rooms)")

```
Regression Equation: Price = 103157.89 + (63.16 × Area) +
(34736.84 × Rooms)
```

Question 8: Implement polynomial regression on the following data: ● X = [1, 2, 3, 4, 5] 3 ● Y = [2.2, 4.8, 7.5, 11.2, 14.7] Fit a 2nd-degree polynomial and plot the resulting curve.

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear_model import LinearRegression

from sklearn.preprocessing import PolynomialFeatures

Sample data

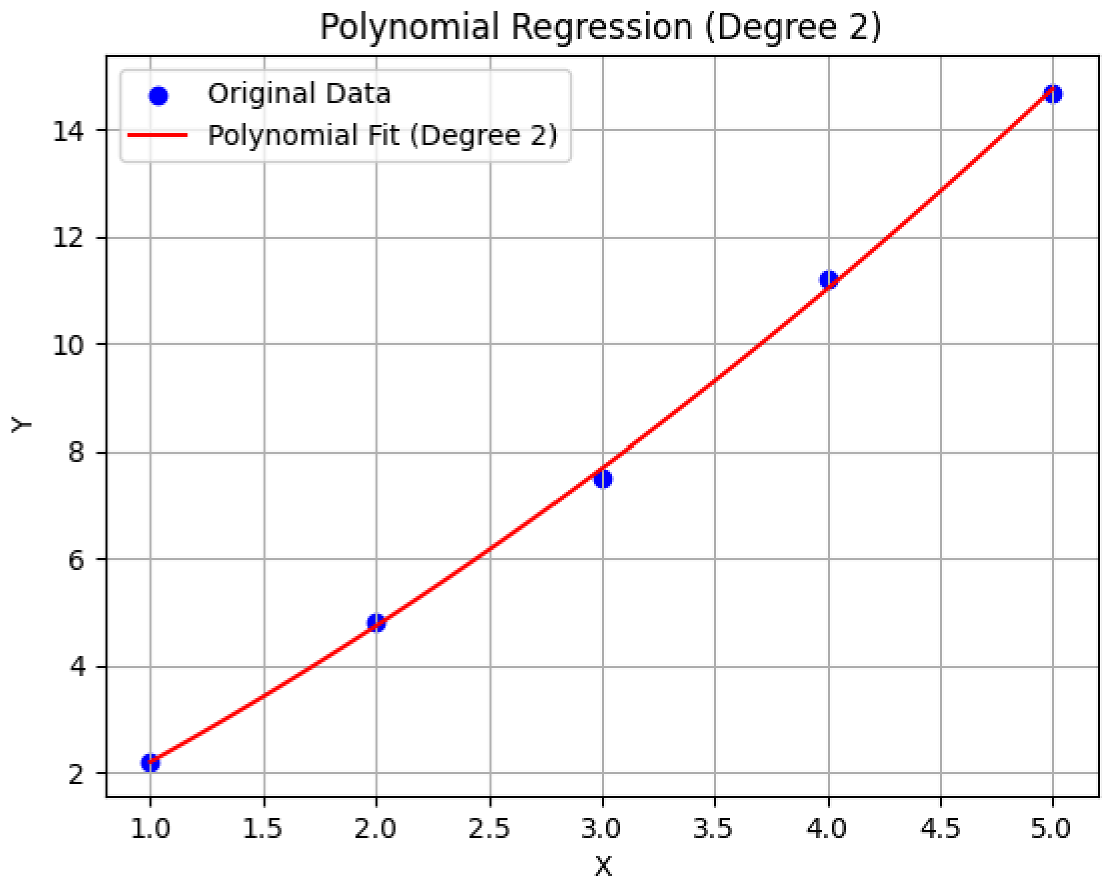
X = np.array([1, 2, 3, 4, 5]).reshape(-1, 1)

Y = np.array([2.2, 4.8, 7.5, 11.2, 14.7])

Create polynomial features (degree 2)

poly = PolynomialFeatures(degree=2)

```
X_poly = poly.fit_transform(X)

# Fit the model
model = LinearRegression()
model.fit(X_poly, Y)

# Predict values
X_fit = np.linspace(1, 5, 100).reshape(-1, 1)
X_fit_poly = poly.transform(X_fit)
Y_pred = model.predict(X_fit_poly)

# Print the polynomial equation
coeffs = model.coef_
intercept = model.intercept_
print(f"Polynomial Equation: Y = {intercept:.2f} + ({coeffs[1]:.2f} × X) + ({coeffs[2]:.2f} × X²)")

# Plot the results
plt.scatter(X, Y, color='blue', label='Original Data')
plt.plot(X_fit, Y_pred, color='red', label='Polynomial Fit (Degree 2)')
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Polynomial Regression (Degree 2)')
plt.legend()
plt.grid(True)
plt.show()
```

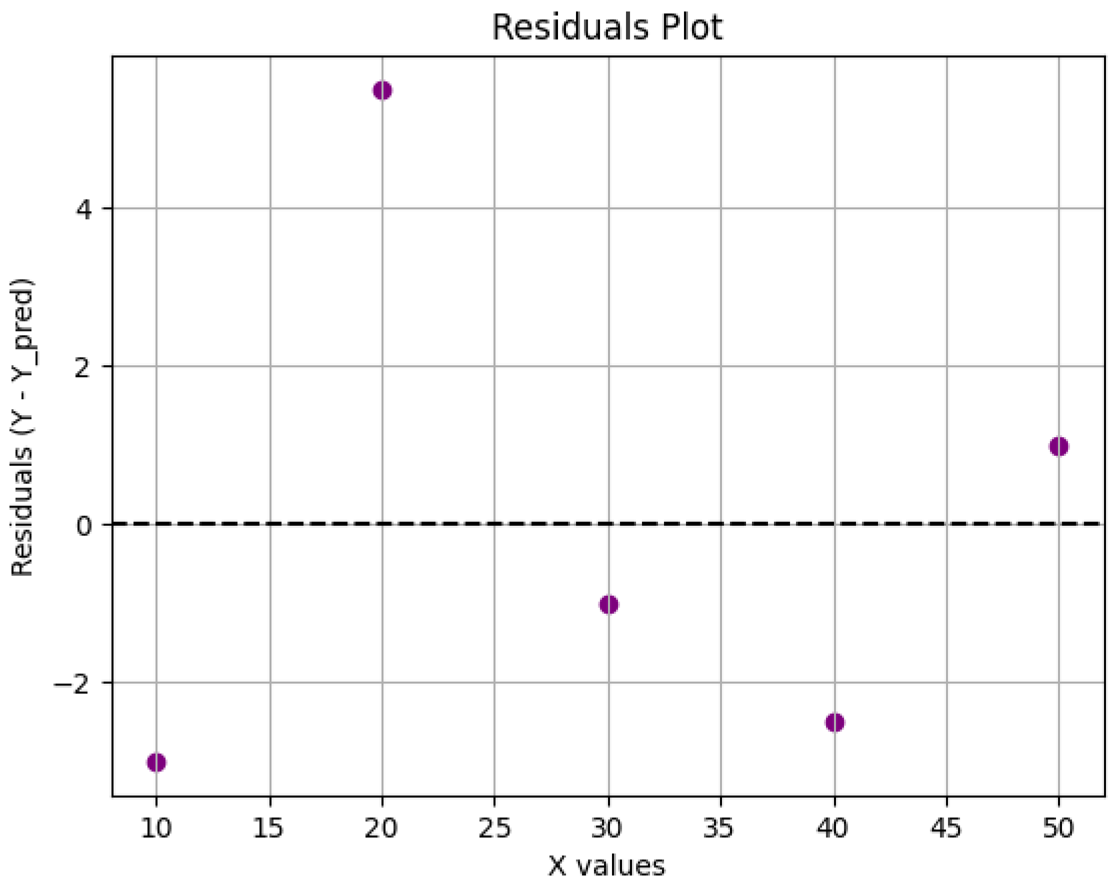


Polynomial Equation: Y = 0.06 + (1.94 × X) + (0.20 × X^2)

Question 9: Create a residuals plot for a regression model trained on this data: ● X = [10, 20, 30, 40, 50] ● Y = [15, 35, 40, 50, 65] Assess heteroscedasticity by examining the spread of residuals.

```
import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear_model import LinearRegression


# Input data

X = np.array([10, 20, 30, 40, 50]).reshape(-1, 1)

Y = np.array([15, 35, 40, 50, 65])
```

```
# Train the linear regression model

model = LinearRegression()

model.fit(X, Y)

# Predict Y values

Y_pred = model.predict(X)

# Calculate residuals

residuals = Y - Y_pred

# Plot residuals

plt.scatter(X, residuals, color='purple')

plt.axhline(y=0, color='black', linestyle='--')

plt.xlabel('X values')

plt.ylabel('Residuals (Y - Y_pred)')

plt.title('Residuals Plot')

plt.grid(True)

plt.show()
```



Question 10: Imagine you are a data scientist working for a real estate company. You need to predict house prices using features like area, number of rooms, and location. However, you detect heteroscedasticity and multicollinearity in your regression model. Explain the steps you would take to address these issues and ensure a robust model.

1. **Visual Diagnosis**
   - **Plot residuals vs. predicted values to confirm heteroscedasticity (look for funnel or pattern shape).**
2. **Transform the Dependent Variable**
   - **Apply transformations such as:**
     - **Logarithmic:** `log(Price)`
     - **Square root or Box-Cox transformation**
   - **This often stabilizes variance and linearizes relationships.**

3. **Use Weighted Least Squares (WLS)**
    - **Assign less weight to observations with higher variance.**
4. **Use Robust Standard Errors**
    - **Apply heteroscedasticity-consistent standard errors (e.g., White's correction) to make inference more reliable.**

Steps to Address Multicollinearity:

1. **Detect It**
    - **Use Variance Inflation Factor (VIF) to check each variable:**
        - **VIF > 5 or 10 indicates a problem.**
2. **Remove or Combine Correlated Predictors**
    - **For example, if area and number of rooms are highly correlated, consider keeping only one or combining them into a new feature (e.g., area per room).**
3. **Dimensionality Reduction**
    - **Use Principal Component Analysis (PCA) to reduce collinearity while preserving variance.**
4. **Regularization Techniques**
    - **Use Ridge Regression (L2) or Lasso Regression (L1):**
        - **Ridge reduces impact of collinearity by shrinking coefficients.**
        - **Lasso can eliminate irrelevant features by forcing some coefficients to zero.**

Final Steps for a Robust Model:

- **Re-train and Re-evaluate the model after adjustments.**
- **Cross-validate to ensure generalizability.**

- **Check residuals again to confirm heteroscedasticity is reduced.**
- **Communicate results clearly with uncertainty quantified (e.g., confidence intervals).**