Name : Sagar Sidhu
Email : sagarsidhu33321@gmail.com
Assignment Name : Regression & Its Evaluation
Assignment code: DA-AG-010
Drive Link : N/A
Github Link : N/A

Question 1: What is Simple Linear Regression?
Answer =**Simple Linear Regression** is a statistical method used to model the relationship between two variables:

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

It assumes that the relationship between X and Y can be described with a straight line:

$Y=\beta 0+\beta 1X+\varepsilon$

Where:

- $\beta 0$ is the **intercept** (value of Y when X = 0)
- $\beta 1$ is the **slope** (change in Y for a one-unit change in X)
- ε is the **error term** (random variation not explained by the model)

Purpose: To predict the value of Y based on X or to understand the strength and direction of their linear relationship.

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

1. **Linearity**:
   The relationship between the independent variable (X) and the dependent variable (Y) is linear.
2. **Independence**:
   Observations (data points) are independent of each other — the residuals are not correlated.
3. **Homoscedasticity**:
   The variance of the residuals (errors) is constant across all values of X (no pattern in the spread of errors).
4. **Normality of Errors**:
   The residuals (differences between observed and predicted values) are normally distributed.
5. **No Multicollinearity** (only relevant in **multiple** regression):
   Since simple linear regression has only one independent variable, this assumption is naturally satisfied.

These assumptions ensure that the regression model is valid and the statistical inferences (like confidence intervals and p-values) are reliable.

Question 3: What is heteroscedasticity, and why is it important to address in regression models?
Answer=**Heteroscedasticity** refers to a situation in regression analysis where the **variance of the errors (residuals) is not constant across all levels of the independent variable(s)**. In simpler terms, as the value of the predictor changes, the spread (or variability) of the residuals changes — it may increase or decrease, forming patterns like funnels or cones when plotted.

Why is it important to address?

1. **Violation of OLS Assumptions**: Ordinary Least Squares (OLS) regression assumes constant variance of errors (homoscedasticity). Heteroscedasticity violates this assumption.

2. **Inefficient Estimates**: While the coefficient estimates remain unbiased, they are no longer efficient (i.e., they don't have the smallest possible variance).

3. **Invalid Inference**: It affects the **standard errors** of the coefficients, leading to **biased t-tests and F-tests**, which can result in **incorrect conclusions** about the significance of predictors.

4. **Misleading Confidence Intervals**: Confidence intervals and prediction intervals may be too narrow or too wide.

How to detect it?

- **Residual plots**: Plotting residuals vs. fitted values can reveal patterns.

- **Breusch-Pagan Test** or **White Test**: Formal statistical tests for heteroscedasticity.

How to address it?

- **Transform the dependent variable** (e.g., log transformation).

- **Use weighted least squares (WLS)** instead of OLS.

- **Use robust standard errors** to correct inference without transforming data.

Question 4: What is Multiple Linear Regression?
Answer:**Multiple Linear Regression (MLR)** is a statistical technique used to model the relationship between one **dependent variable** and **two or more independent variables**. It extends simple linear regression by allowing for **multiple predictors** to explain the outcome variable.

General Form:

$Y=\beta_0+\beta_1X_1+\beta_2X_2+\cdots+\beta_nX_n+\varepsilon$

- Y: Dependent (response) variable
- $X_1,X_2,\ldots,X_n$: Independent (predictor) variables
- β_0: Intercept
- $\beta_1,\beta_2,\ldots,\beta_n$: Coefficients (slopes)
- ε: Error term

Key Features:

- Each coefficient β_ii represents the effect of the corresponding independent variable $X_i$on Y, **holding other variables constant**.
- MLR helps in predicting the outcome and assessing the **relative importance** of each predictor.

Applications:

- Economics (predicting GDP from inflation, interest rates, etc.)
- Healthcare (predicting disease risk from age, weight, and lifestyle)
- Marketing (forecasting sales using advertising, pricing, and competition data)

Question 5: What is polynomial regression, and how does it differ from linear regression?
Answer:**Polynomial Regression** is a type of regression analysis in which the **relationship between the independent variable and the dependent variable is modeled as an nth-degree polynomial**. It is used when the data shows a **non-linear** relationship but can still be approximated by a curve that is a polynomial in nature.

General Form:

Y=β0+β1X+β2X2+β3X3+⋯+βnXn+ε

- Y: Dependent variable
- X: Independent variable
- β0,β1,…,βn: Coefficients
- ε: Error term
- n: Degree of the polynomial

Difference from Linear Regression:

Aspect	Linear Regression	Polynomial Regression
Relationship	Models a **straight-line** relationship	Models a **curved/non-linear** relationship
Equation	Y=β0+β1X+ε	Y=β0+β1X+β2X2+⋯+ε
Linearity in Variables	Only first-degree term (X)	Includes higher-degree terms (e.g., X2,X3,…)
Fit	May underfit curved data	Can capture complex patterns in data
Risk	Simpler, less prone to overfitting	Higher-degree polynomials may **overfit** the data

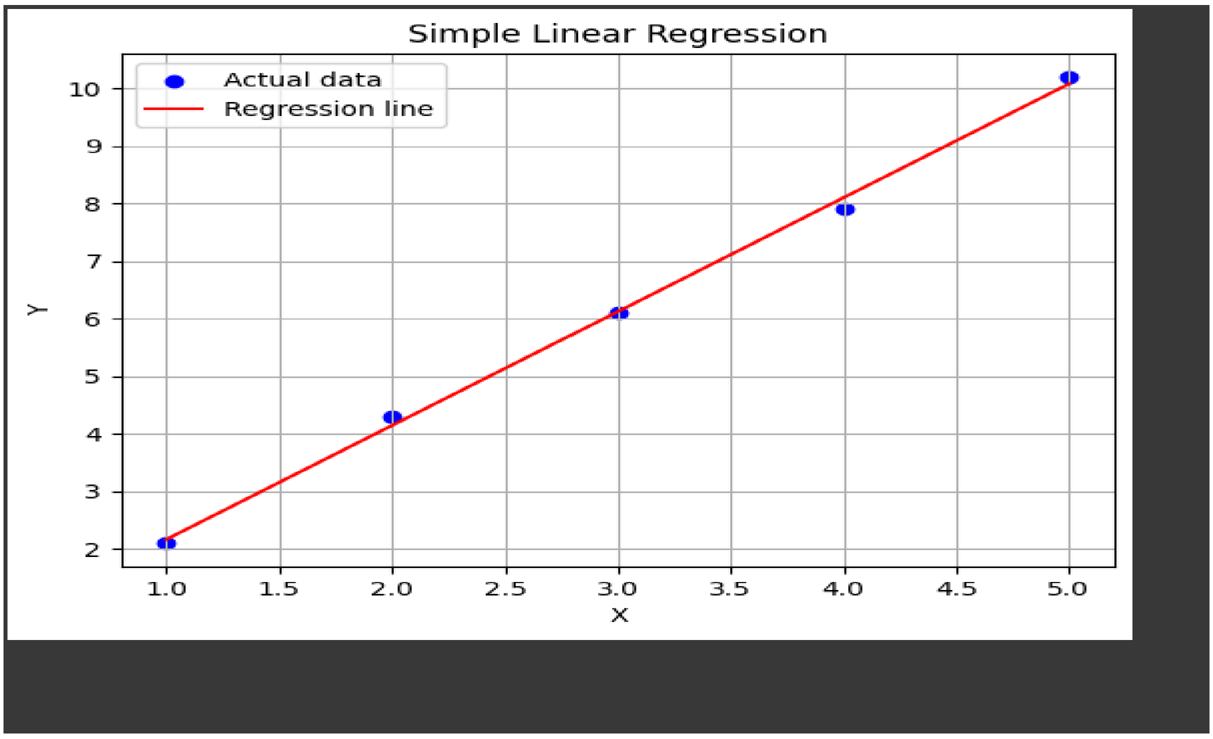
Question 6: Implement a Python program to fit a Simple Linear Regression model to the following sample data: ● X = [1, 2, 3, 4, 5] ● Y = [2.1, 4.3, 6.1, 7.9, 10.2] Plot the regression line over the data points. (Include your Python code and output in the code box below.)
Answer=`import numpy as np`

```
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

# Sample data
X = np.array([1, 2, 3, 4, 5]).reshape(-1, 1)  # Reshape for sklearn
Y = np.array([2.1, 4.3, 6.1, 7.9, 10.2])

# Create and fit the model
```

```
model = LinearRegression()
model.fit(X, Y)

# Predict values
Y_pred = model.predict(X)

# Print coefficients
print(f"Intercept (b0): {model.intercept_:.2f}")
print(f"Slope (b1): {model.coef_[0]:.2f}")

# Plotting
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()
```

OUTPUT:

```
Intercept (b0): 0.18
Slope (b1): 1.98
```



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] Check for multicollinearity using VIF and report the results. (Include your Python code and output in the code box below.)

ANSWER=

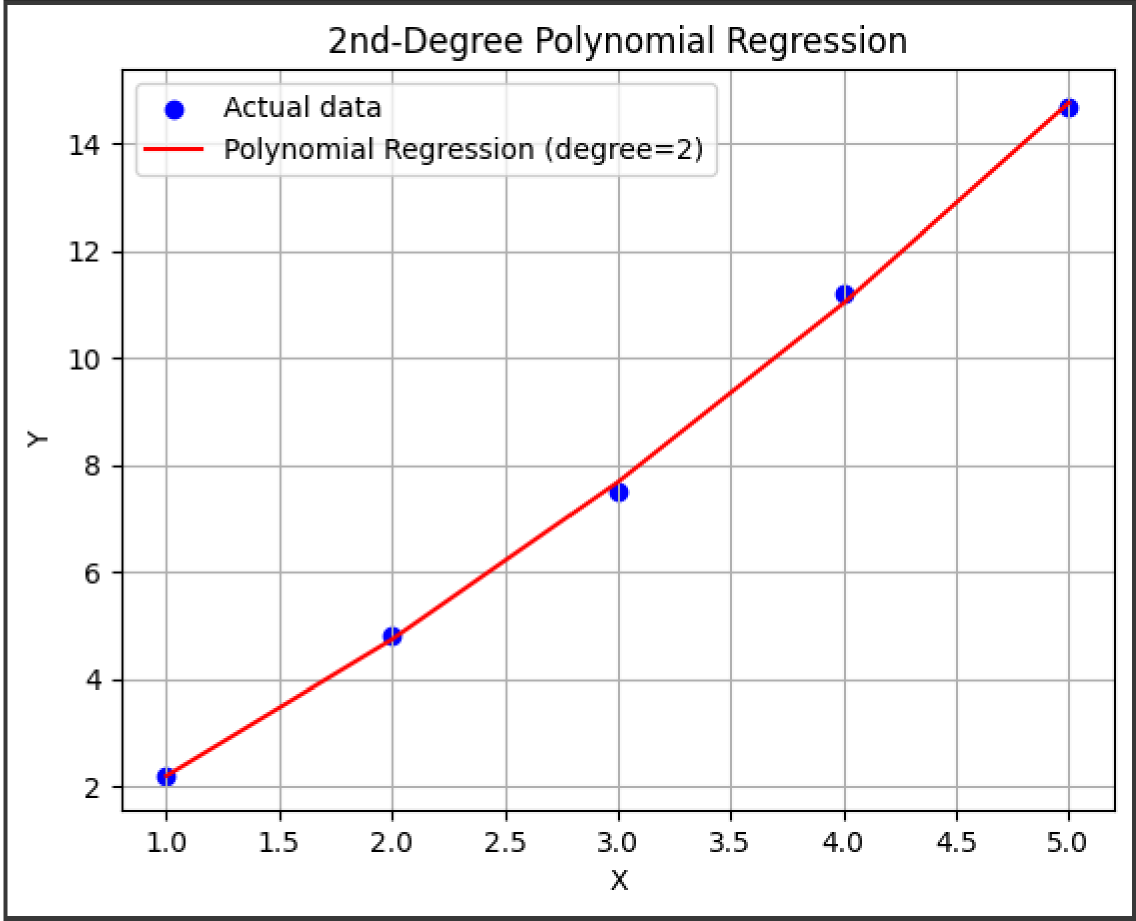
```
import numpy as np
import pandas as pd
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import
variance_inflation_factor

# Sample data
data = pd.DataFrame({
    'Area': [1200, 1500, 1800, 2000],
    'Rooms': [2, 3, 3, 4],
    'Price': [250000, 300000, 320000, 370000]
})

# Define independent and dependent variables
X = data[['Area', 'Rooms']]
Y = data['Price']

# Add constant for intercept
X_const = sm.add_constant(X)

# Fit model
model = sm.OLS(Y, X_const).fit()
print(model.summary())

# Calculate VIF
vif_data = pd.DataFrame()
vif_data["Feature"] = X.columns
vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in
range(X.shape[1])]

print("\nVariance Inflation Factors (VIF):")
print(vif_data)


OUTPUT:
                          OLS Regression Results
=======================================================================
=======
Dep. Variable:                  Price   R-squared:
0.999
Model:                            OLS   Adj. R-squared:
0.996
Method:                 Least Squares   F-statistic:
351.0
Date:                Fri, 18 Jul 2025   Prob (F-statistic):
0.0377
Time:                        10:18:11   Log-Likelihood:
-35.242
No. Observations:                   4   AIC:
76.48
Df Residuals:                       1   BIC:
74.64
Df Model:                           2
Covariance Type:            nonrobust
========================================================================
=======
                 coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------
-------
const       1.032e+05   9488.293     10.872      0.058   -1.74e+04
2.24e+05
Area          63.1579     14.886      4.243      0.147    -125.992
252.308
Rooms       3.474e+04   6381.240      5.444      0.116   -4.63e+04
1.16e+05
========================================================================
=======
Omnibus:                          nan   Durbin-Watson:
2.053
Prob(Omnibus):                    nan   Jarque-Bera (JB):
0.554
Skew:                          -0.154   Prob(JB):
0.758
Kurtosis:                       1.202   Cond. No.
1.01e+04
========================================================================
=======

Notes:
```

```
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 1.01e+04. This might indicate that
there are
strong multicollinearity or other numerical problems.

Variance Inflation Factors (VIF):
  Feature          VIF
0     Area  127.796923
1    Rooms  127.796923
```

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. (Include your Python code and output in the code box below.)

```
ANSWER=import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression

# 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])

# Transform to 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
Y_pred = model.predict(X_poly)

# Print coefficients
print("Intercept:", model.intercept_)
print("Coefficients:", model.coef_)

# Plotting
```

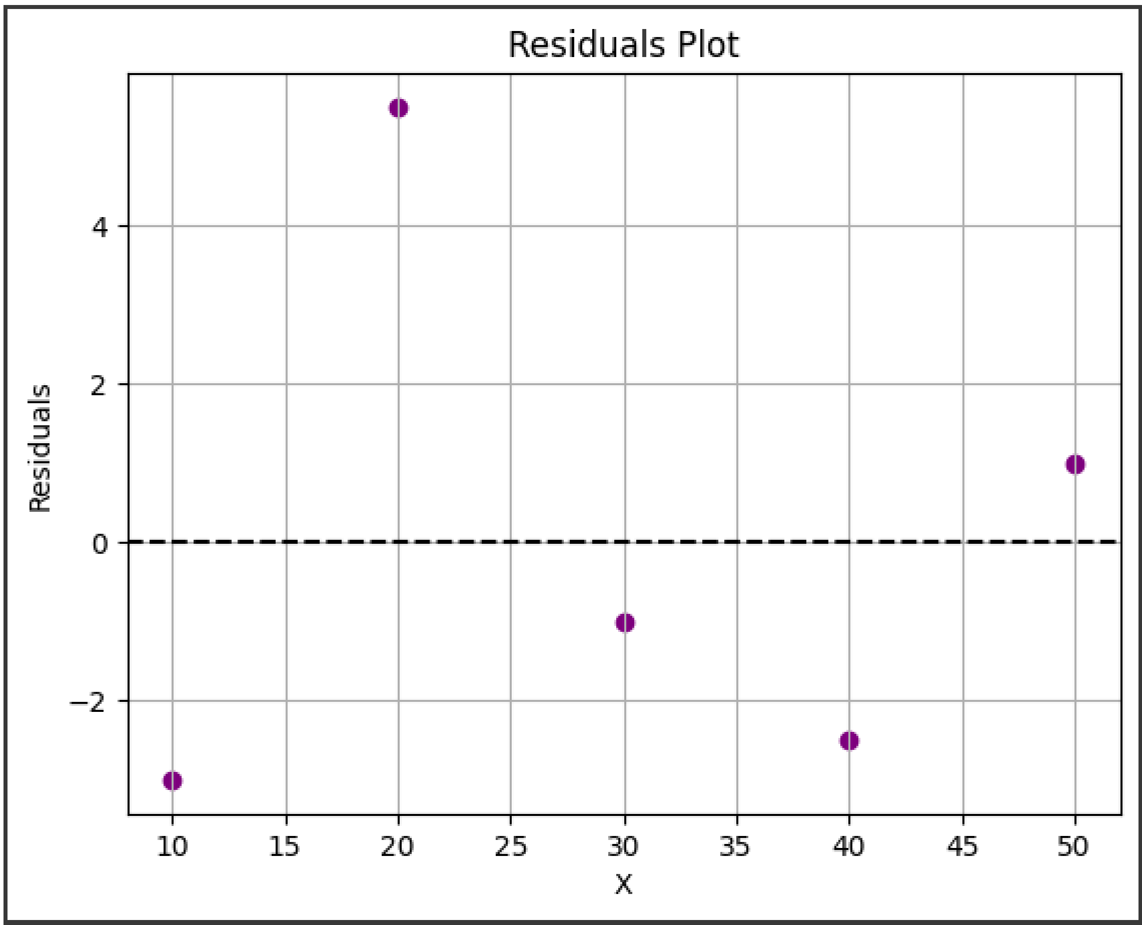
```
plt.scatter(X, Y, color='blue', label='Actual data')
plt.plot(X, Y_pred, color='red', label='Polynomial Regression
(degree=2)')
plt.title('2nd-Degree Polynomial Regression')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.grid(True)
plt.show()



OUTPUT:
Intercept: 0.06000000000000938
Coefficients: [0.   1.94 0.2 ]
```



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. (Include your Python code and output in the code box below.)

ANSWER=
```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

# Data
X = np.array([10, 20, 30, 40, 50]).reshape(-1, 1)
Y = np.array([15, 35, 40, 50, 65])

# Fit the linear model
model = LinearRegression()
model.fit(X, Y)
Y_pred = model.predict(X)

# Calculate residuals
residuals = Y - Y_pred

# Print residuals
print("Residuals:", residuals)

# Residuals plot
plt.scatter(X, residuals, color='purple')
plt.axhline(y=0, color='black', linestyle='--')
plt.title("Residuals Plot")
plt.xlabel("X")
plt.ylabel("Residuals")
plt.grid(True)
plt.show()
```

OUTPUT:
```
Residuals: [-3.   5.5 -1.  -2.5  1. ]
```



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?
Answer=

As a data scientist building a **house price prediction model**, detecting **heteroscedasticity** and **multicollinearity** in your regression model indicates that some of the key assumptions of linear regression are violated. Here's how I would address each issue to build a **robust and reliable model**:

✅ Step-by-Step Approach

1. Addressing Multicollinearity

Multicollinearity occurs when two or more independent variables are highly correlated, leading to unstable coefficient estimates and inflated standard errors.

🔍 ***How to Detect:***

- **Variance Inflation Factor (VIF)**: VIF > 10 indicates high multicollinearity.

🛠️ ***Solutions:***

- **Remove or combine correlated features**:
  - Drop one of the correlated variables (e.g., if `area` and `number_of_rooms` are highly correlated).
- **Principal Component Analysis (PCA)**:
  - Reduce dimensionality while preserving variance.
- **Regularization Techniques**:
  - Use **Ridge Regression (L2)** or **Lasso Regression (L1)** to penalize large coefficients and reduce multicollinearity.

---

2. Addressing Heteroscedasticity

Heteroscedasticity violates the assumption of constant variance of residuals, leading to inefficient estimates and invalid inference.

🔍 ***How to Detect:***

- **Residual plots**: Residuals should be randomly scattered around zero.
- **Breusch-Pagan test**: Formal statistical test.

🛠️ ***Solutions:***

- **Transform the dependent variable**:
  - Use **logarithmic, square root, or Box-Cox transformation** on `Price` to stabilize variance.
- **Weighted Least Squares (WLS)**:
  - Assign weights inversely proportional to the variance of the errors.
- **Use Robust Standard Errors**:
  - Helps correct inference without changing the model.

---

3. Improve the Overall Model

🛠️ ***Additional Best Practices*:**

- **Feature Engineering**:
  - Create more meaningful features from location (e.g., encode proximity to amenities, neighborhood rating).
- **Categorical Variable Encoding**:
  - Use **One-Hot Encoding** or **Target Encoding** for `location`.
- **Train-Test Split & Cross-Validation**:
  - Ensure model generalizes well using **K-Fold Cross-Validation**.
- **Model Comparison**:
  - Try advanced models like **Random Forest, Gradient Boosting, or XGBoost**, which are more robust to heteroscedasticity and multicollinearity.

✅ Final Robust Model Strategy

- Perform **feature correlation analysis** and drop/reduce correlated features.
- Apply **log transformation on `Price`** if heteroscedasticity exists.
- Use **Ridge Regression** to handle multicollinearity.
- Validate with **cross-validation** and examine **residual plots**.
- Use **SHAP or feature importance** to interpret model decisions.