Assignment Code: DA-AG-010

Regression & Its Evaluation | **Assignment**

Instructions: Carefully read each question. Use Google Docs, Microsoft Word, or a similar tool to create a document where you type out each question along with its answer. Save the document as a PDF, and then upload it to the LMS. Please do not zip or archive the files before uploading them. Each question carries 20 marks.

Total Marks: 100

Question 1: What is Simple Linear Regression?

Answer:

Simple linear regression is a **statistical method** used to model and analyze the relationship between two variables:

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

This is the equation >> Y=β0+β1X

Where:

- Y: dependent variable (what you're trying to predict)
- X: independent variable (the predictor)
- β0\beta: intercept (value of Y when X=0)
- β1\beta: slope (how much Y changes when X increases by 1)

Question 2: What are the key assumptions of Simple Linear Regression?

Answer:

1. Linearity

- The relationship between the **independent variable (X)** and the **dependent variable (Y)** must be **linear**
- like, the change in Y is proportional to the change in X.

2. Independence of Errors

- The residuals (errors) — the differences between observed and predicted y— must be **independent** of each other.
- This means one observation's error shouldn't depend on another's.

3. Homoscedasticity (Constant Variance of Errors)

- The residuals should have **constant variance** across all levels of x
- That is, the spread of errors should be the same for small and large values of x

4. Normality of Errors

- The residuals should be **normally distributed** (especially important for hypothesis testing and confidence intervals).

5. No Perfect Multicollinearity

- In *simple* linear regression, there's only one predictor — so this assumption is automatically satisfied.

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

Answer:

Heteroscedasticity means that the **variance of the errors (residuals)** in a regression model is **not constant** across all values of the independent variable(s).

In contrast, **homoscedasticity** means the residuals have a **constant variance** — which is what we *want*.

Example -> When you plot the residuals (errors) versus the predicted values (or versus XXX), the **spread of residuals should look even** (random cloud).If the residuals **fan out** or **narrow** as X increases, that's **heteroscedasticity**.

Question 4: What is Multiple Linear Regression?

Answer:

Multiple Linear Regression (MLR) is an extension of **Simple Linear Regression** that uses **two or more independent (predictor) variables** to explain or predict the value of a **single dependent (response) variable**.

It helps you understand how **several factors together** influence an outcome.

Equation of MLR =>

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

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

Answer:

Polynomial Regression is a type of **linear regression** in which the **relationship between the independent variable (X)** and the **dependent variable (Y)** is modeled as an **nth-degree polynomial**

Equation

The general form of a **polynomial regression model** is:

$$Y=\beta_0+\beta_1X+\beta_2X^2+\beta_3X^3+\cdots+\beta_nX^n$$

Where:

- Y : dependent variable
- X : independent variable
- n : degree of the polynomial
- ε : error term

Question 6: Implement a Python program to fit a Simple Linear Regression model to the following sample data:

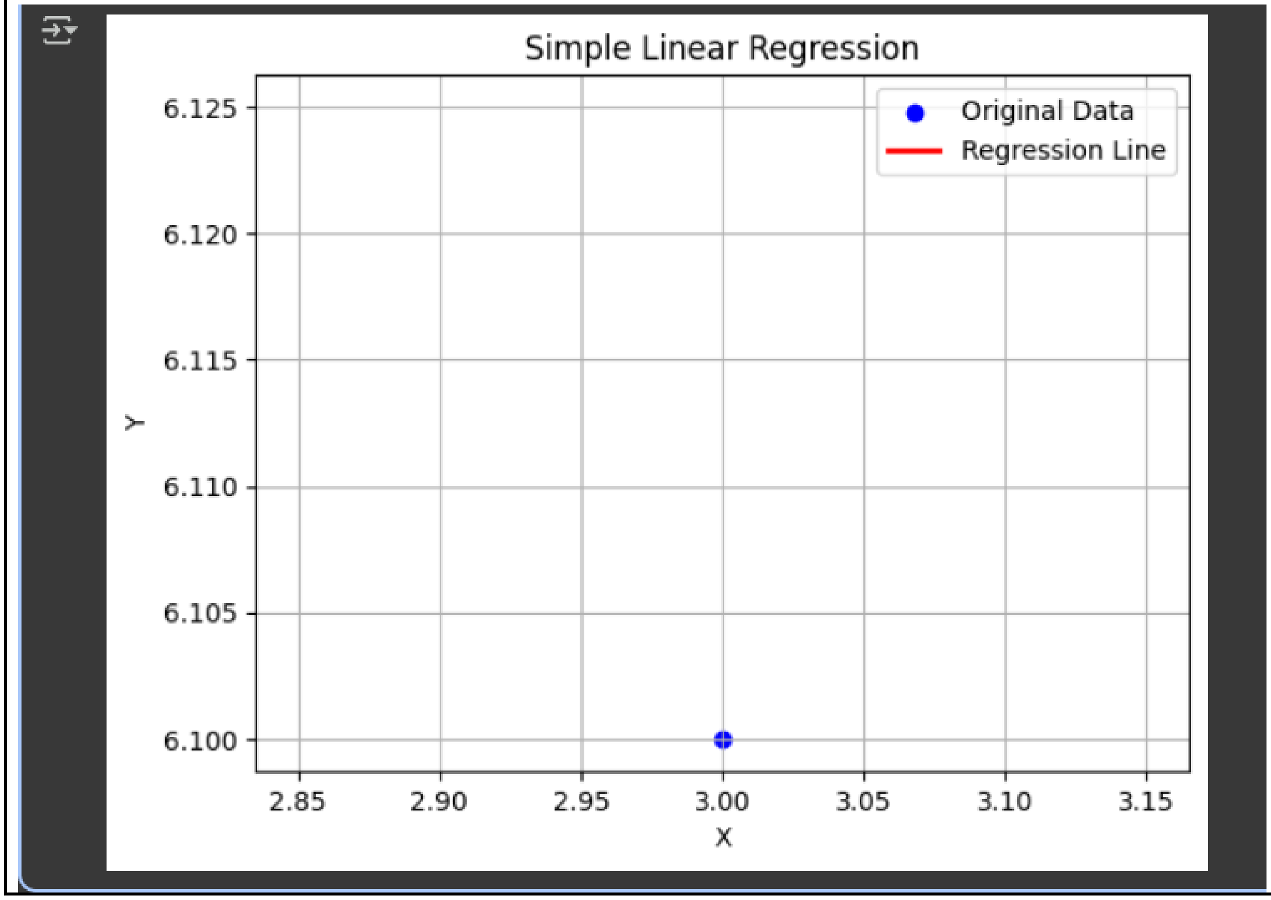
- X = [1, 2, 3, 4, 5]
- Y = [2.1, 4.3, 6.1, 7.9, 10.2]

Plot the regression line over the data points.

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

```
 # 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]

# Train Test Split
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train, Y_test = train_test_split(X,Y,test_size=0.20,
random_state = 1)

# Reshape X_train and X_test to be 2D arrays
import numpy as np
X_train = np.array(X_train).reshape(-1, 1)
X_test = np.array(X_test).reshape(-1, 1)
Y_train = np.array(Y_train)
Y_test = np.array(Y_test)


# importing Linear Regression
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model = model.fit(X_train,Y_train)
```

```
model

import matplotlib.pyplot as plt

# Predict the values using the trained model
y_pred = model.predict(X_test)

# Plot the original data points
plt.scatter(X_test, Y_test, color='blue', label='Original Data')

# Plot the regression line
plt.plot(X_test, y_pred, color='red', linewidth=2, label='Regression
Line')

plt.xlabel('X')
plt.ylabel('Y')
plt.title('Simple Linear Regression')
plt.legend()
plt.grid(True)
plt.show()
```



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:**

```
# 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]

# for multiple regression model we need to change the in dataframe
data = {'Area': Area, 'Rooms': Rooms, 'Price': Price}
df = pd.DataFrame(data)
```

```
display(df)

# separating x and y
X = df[['Area', 'Rooms']]
y = df['Price']

# import linear Regression
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X, y)


# checking VIF
from statsmodels.stats.outliers_influence import
variance_inflation_factor

vif_data = pd.DataFrame()
vif_data["feature"] = X.columns

vif_data["VIF"] = [variance_inflation_factor(X.values, i)
                          for i in range(len(X.columns))]

print(vif_data)


print("VIF Data:")
print(vif_data)

# Insight of the VIF
print("\nInterpretation of VIF values:")
print("The VIF values for both 'Area' (%.2f) and 'Rooms' (%.2f) are
significantly high." % (vif_data.loc[vif_data['feature'] == 'Area',
'VIF'].iloc[0], vif_data.loc[vif_data['feature'] == 'Rooms',
'VIF'].iloc[0]))
print("Generally, a VIF value greater than 5 or 10 indicates significant
multicollinearity.")
print("In this case, both features have VIF values exceeding 100, which
suggests a very strong linear relationship between 'Area' and 'Rooms'.")
```

```
print("This high multicollinearity can make it difficult to interpret
the individual coefficients of the linear regression model and may lead
to unstable coefficient estimates.")
```

	Area	Rooms	Price
0	1200	2	250000
1	1500	3	300000
2	1800	3	320000
3	2000	4	370000

```
  feature         VIF
0    Area  127.796923
1   Rooms  127.796923
VIF Data:
  feature         VIF
0    Area  127.796923
1   Rooms  127.796923

Interpretation of VIF values:
The VIF values for both 'Area' (127.80) and 'Rooms' (127.80) are
significantly high.
Generally, a VIF value greater than 5 or 10 indicates significant
multicollinearity.
In this case, both features have VIF values exceeding 100, which suggests a
very strong linear relationship between 'Area' and 'Rooms'.
This high multicollinearity can make it difficult to interpret the
individual coefficients of the linear regression model and may lead to
unstable coefficient estimates
```

Question 8: Implement **polynomial regression** on the following data: ●

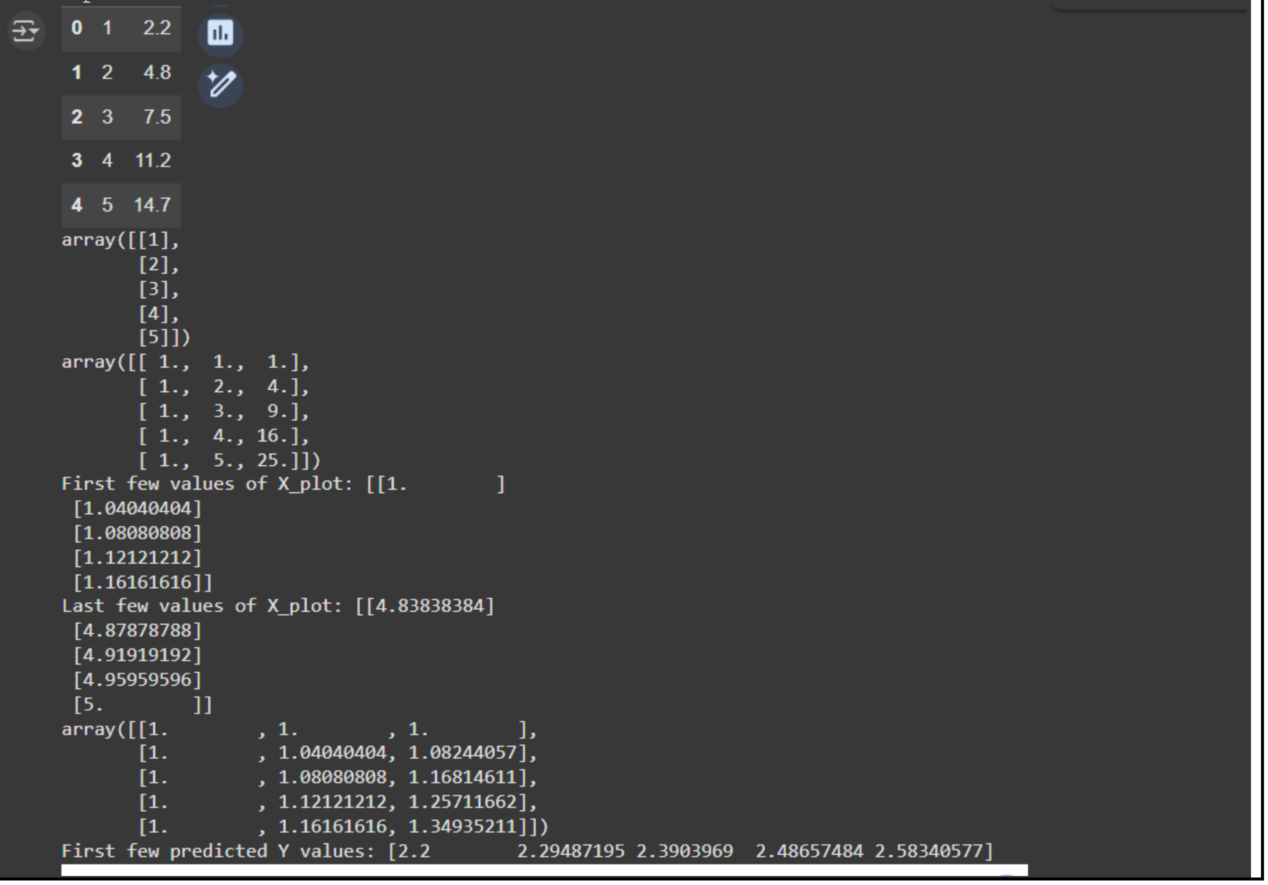
X = [1, 2, 3, 4, 5]

● Y = [2.2, 4.8, 7.5, 11.2, 14.7]

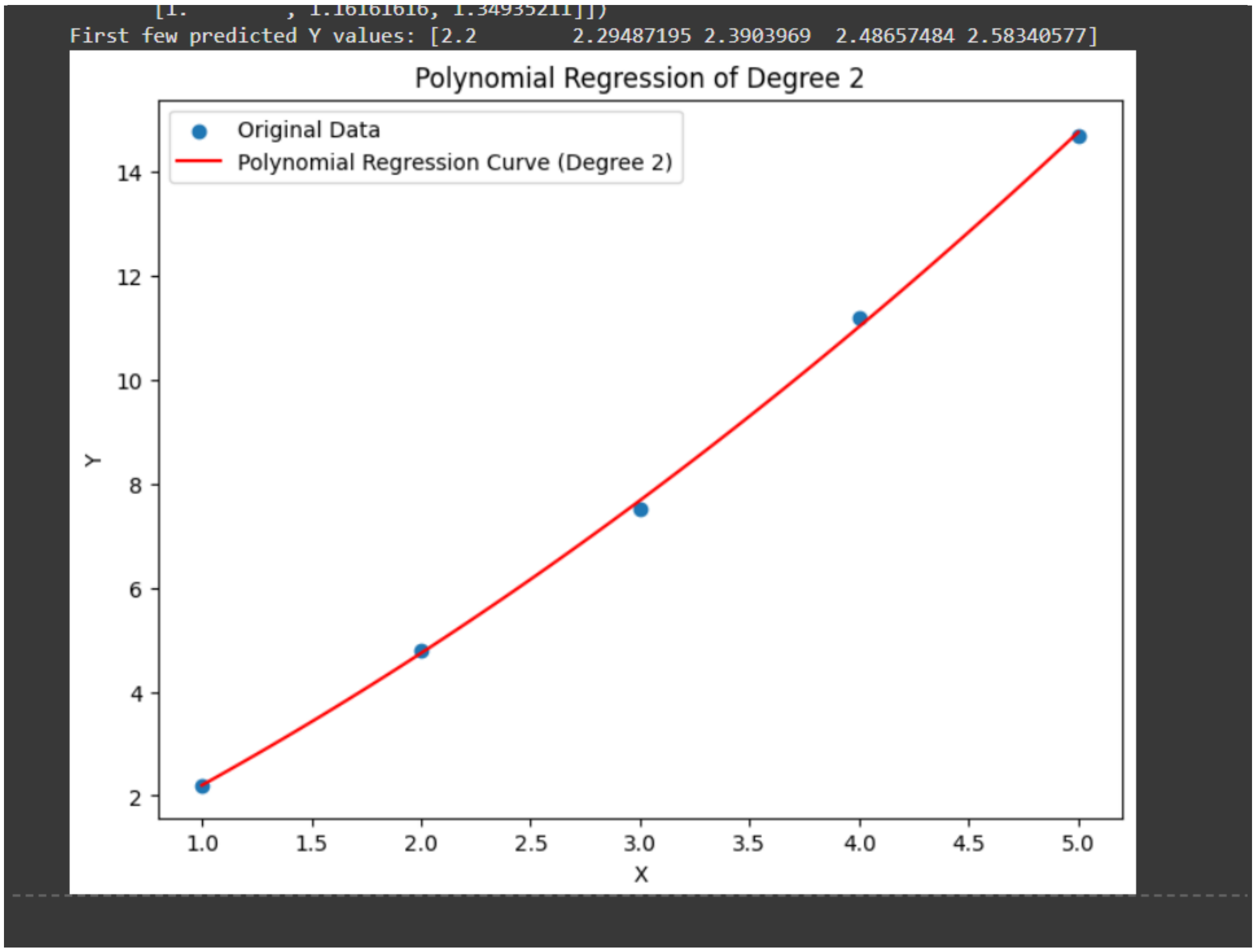
Fit a **2nd-degree polynomial** and plot the resulting curve.
(*Include your Python code and output in the code box below.*)
Answer:

```python
import pandas as pd

#converting into Dataframe
data = {'X': [1, 2, 3, 4, 5], 'Y': [2.2, 4.8, 7.5, 11.2, 14.7]}
df = pd.DataFrame(data)
display(df)

# reshaping
X_reshaped = df['X'].values.reshape(-1, 1)
display(X_reshaped)

# polynomial
from sklearn.preprocessing import PolynomialFeatures

poly_features = PolynomialFeatures(degree=2)
X_poly = poly_features.fit_transform(X_reshaped)
display(X_poly)


from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X_poly, df['Y'])

import numpy as np

# graphical represtation
X_plot = np.linspace(df['X'].min(), df['X'].max(), 100).reshape(-1, 1)
print("First few values of X_plot:", X_plot[:5])
print("Last few values of X_plot:", X_plot[-5:])


X_plot_poly = poly_features.transform(X_plot)
display(X_plot_poly[:5])

y_plot = model.predict(X_plot_poly)
print("First few predicted Y values:", y_plot[:5])


import matplotlib.pyplot as plt
```

```
plt.figure(figsize=(8, 6))
plt.scatter(df['X'], df['Y'], label='Original Data')
plt.plot(X_plot, y_plot, color='red', label='Polynomial Regression Curve
(Degree 2)')
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Polynomial Regression of Degree 2')
plt.legend()
plt.show()


# Insight
print("""Insights
The 2nd-degree polynomial regression model appears to fit the given data
well based on the generated plot..""")

# output
```



```
0  1   2.2
1  2   4.8
2  3   7.5
3  4  11.2
4  5  14.7
array([[1],
       [2],
       [3],
       [4],
       [5]])
array([[ 1.,  1.,  1.],
       [ 1.,  2.,  4.],
       [ 1.,  3.,  9.],
       [ 1.,  4., 16.],
       [ 1.,  5., 25.]])
First few values of X_plot: [[1.        ]
 [1.04040404]
 [1.08080808]
 [1.12121212]
 [1.16161616]]
Last few values of X_plot: [[4.83838384]
 [4.87878788]
 [4.91919192]
 [4.95959596]
 [5.        ]]
array([[1.        , 1.        , 1.        ],
       [1.        , 1.04040404, 1.08244057],
       [1.        , 1.08080808, 1.16814611],
       [1.        , 1.12121212, 1.25711662],
       [1.        , 1.16161616, 1.34935211]])
First few predicted Y values: [2.2        2.29487195 2.3903969  2.48657484 2.58340577]
```



```
          [1.          , 1.16161616, 1.34935211]])
First few predicted Y values: [2.2        2.29487195 2.3903969  2.48657484 2.58340577]
```

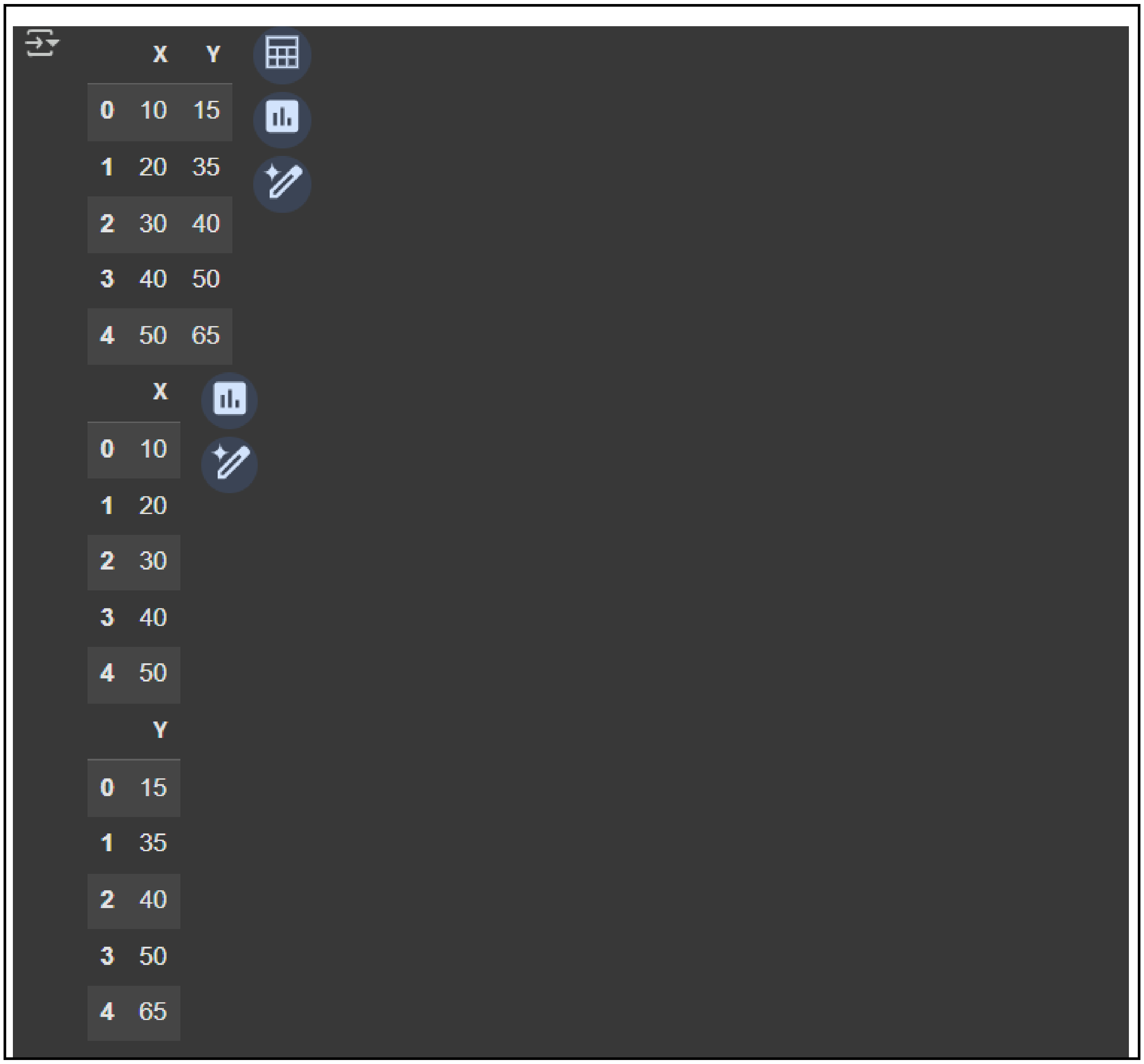
Question 9: Create a **residuals plot** for a regression model trained on this data:

- X = [10, 20, 30, 40, 50] ● Y = [15, 35, 40, 50, 65]

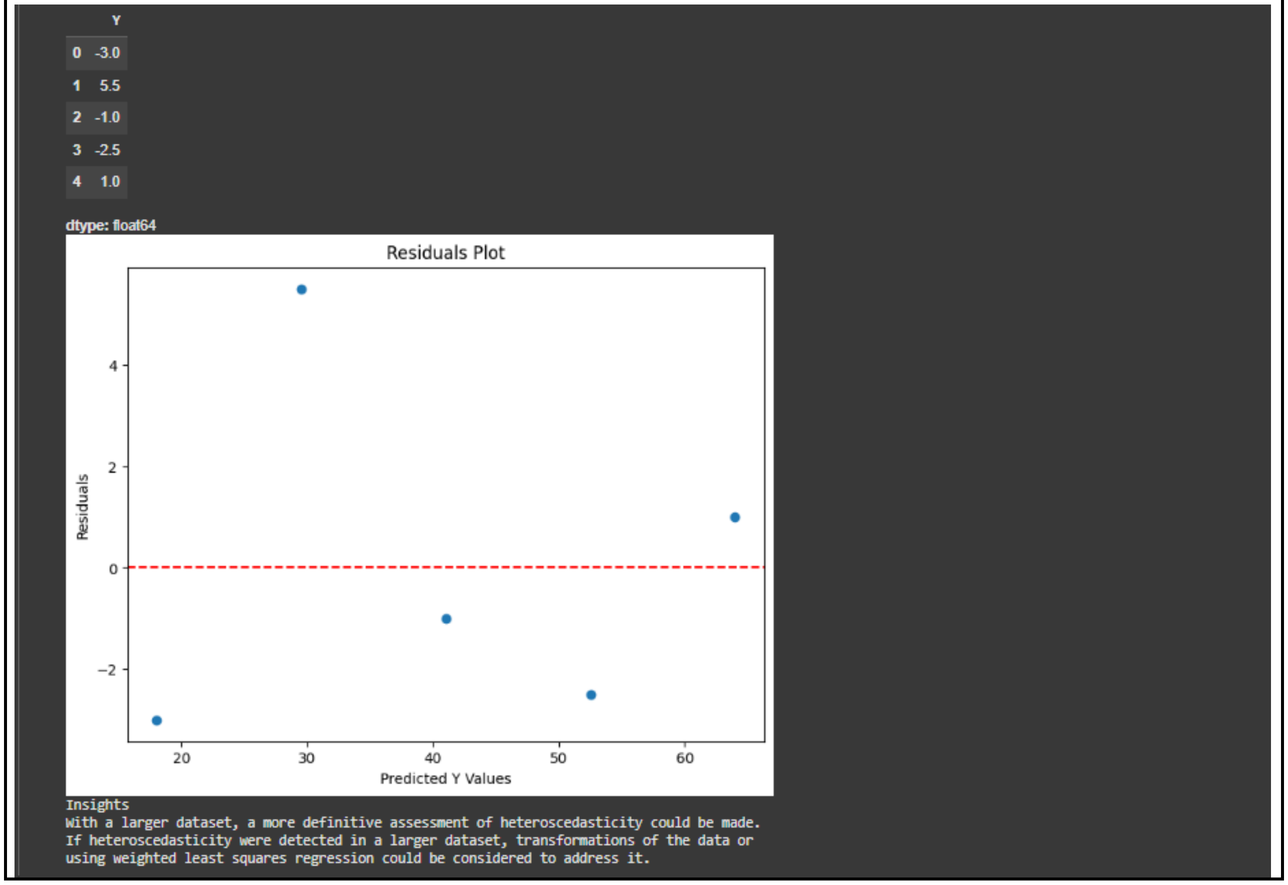
Assess heteroscedasticity by examining the spread of residuals.
(*Include your Python code and output in the code box below.*)
Answer:

```
 import pandas as pd

data = {'X': [10, 20, 30, 40, 50], 'Y': [15, 35, 40, 50, 65]}
df = pd.DataFrame(data)
display(df)
```

```python
X = df[['X']]
y = df['Y']
display(X)
display(y)

from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X, y)


y_pred = model.predict(X)
residuals = y - y_pred
display(residuals)

import matplotlib.pyplot as plt

plt.figure(figsize=(8, 6))
plt.scatter(y_pred, residuals)
plt.axhline(y=0, color='r', linestyle='--')
plt.xlabel('Predicted Y Values')
plt.ylabel('Residuals')
plt.title('Residuals Plot')
plt.show()


# Insight
print("""Insights
With a larger dataset, a more definitive assessment of heteroscedasticity
could be made.
If heteroscedasticity were detected in a larger dataset, transformations
of the data or
using weighted least squares regression could be considered to address
it.""")

# output
```



```
    X   Y
0  10  15
1  20  35
2  30  40
3  40  50
4  50  65
    X
0  10
1  20
2  30
3  40
4  50
    Y
0  15
1  35
2  40
3  50
4  65
```



```
   Y
0  -3.0
1   5.5
2  -1.0
3  -2.5
4   1.0

dtype: float64
```

```
Insights
With a larger dataset, a more definitive assessment of heteroscedasticity could be made.
If heteroscedasticity were detected in a larger dataset, transformations of the data or
using weighted least squares regression could be considered to address it.
```

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.

Answer:

To handle **heteroscedasticity**, I would first check the residual plots to confirm it. Then I'd try:

1. **Transforming the target variable** (e.g., using log(price)) to stabilize variance.
2. **Using Weighted Least Squares** so that data points with high variance get less weight.

3. If those don’t help, I might switch to a **robust regression** method that’s less sensitive to unequal variance.

For **multicollinearity**, I’d:

1. **Check VIF (Variance Inflation Factor)** to see which variables are highly correlated.
2. **Remove or combine correlated features** (like total rooms and bedrooms).
3. Consider **regularization methods** like **Ridge or Lasso regression**, which handle multicollinearity well.

Finally, I’d **retrain and validate** the model to make sure performance and residuals look good.