Assignment Code: DA-AG-010

Regression & Its Evaluation | **Assignment**

Name: Dibyojyoti Dutta
Email-Id:- ddduttaoffice184@gmail.com
Assignment Name:- Regression
Drive Link:-N/A
Github Link:- Link

Total Marks: 100

Question 1: What is Simple Linear Regression?

Answer:

Simple linear regression is a statistical method that models the relationship between two variables by fitting a straight line to the data. It predicts the value of one variable (the dependent variable) based on the value of another variable (the independent variable), using the equation of a line: $y=mx+c$, where m is the slope and c is the intercept

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

Answer:

- **Linearity:** The relationship between the independent variable (X) and the dependent variable (Y) must be linear.
- **Independence:** The residuals (errors) should be independent; that is, the value of one observation should not influence another.
- **Homoscedasticity:** The size of the errors should be about the same everywhere along the line. This means the dots are not spreading out more (or less) as we move along the x-axis—this is called "constant variance" or "homoscedasticity"
- **Errors are Normal:** If we look at all the errors, their pattern looks like a typical bell curve (normal distribution)

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

Answer:

Heteroscedasticity is a condition in regression analysis where the variance of the residuals /Error is not constant across all levels of the independent variable(s).

Why Important?-

the core problem is that the changing spread of errors makes regression results look more certain or uncertain than they truly are.

If heteroscedasticity is not addressed, Standard Errors would be wrong:

For example,

1) **Fake Significance:** Suppose a variable is actually not related to the outcome, but because heteroscedasticity makes standard errors too small or too large, our regression might incorrectly show this variable as statistically significant—so we draw the wrong conclusion about what matters in our model

2) **Bad Predictions:** If we use the model to make predictions (like forecasting spending based on income), our prediction intervals will be too wide or too narrow in different parts of the data, making our forecasts unreliable or misleading

Question 4: What is Multiple Linear Regression?

Answer:

Multiple linear regression is a statistical technique used to predict the value of a single dependent variable based on the values of two or more independent (explanatory) variables

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

Answer:

Polynomial regression is a statistical method used to model the relationship between a dependent variable and an independent variable as a polynomial of degree n, which allows it to capture curved, non-linear trends in data. In contrast, **linear regression** models the relationship using a straight line, assuming a constant rate of change between the variables

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

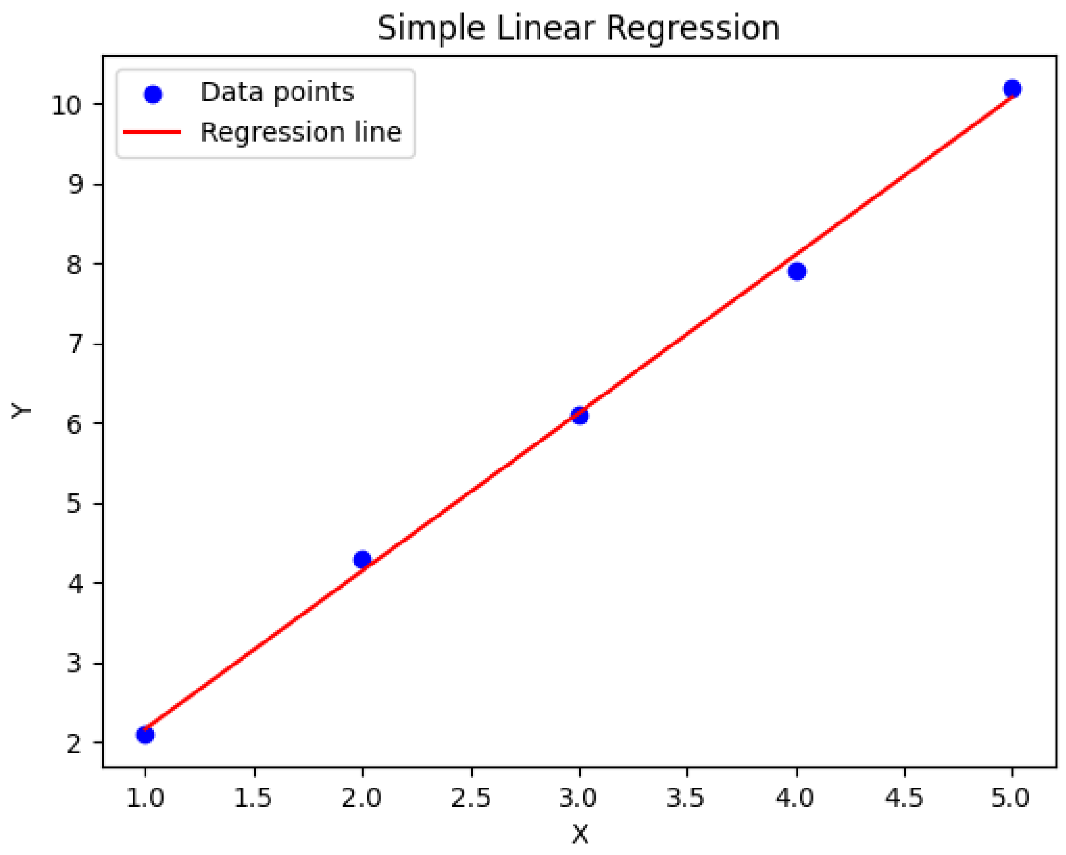
- `X = [1, 2, 3, 4, 5]`
- `Y = [2.1, 4.3, 6.1, 7.9, 10.2]`

Plot the regression line over the data points.

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

Answer:

Code:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')

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

X = np.array([1, 2, 3, 4, 5]).reshape(-1, 1)
Y = np.array([2.1, 4.3, 6.1, 7.9, 10.2])

from sklearn.linear_model import LinearRegression
model = LinearRegression()
model

model.fit(X, Y)

Y_pred = model.predict(X)

plt.scatter(X, Y, color='blue', label='Data points')
plt.plot(X, Y_pred, color='red', label='Regression line')

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


Simple Linear Regression
Data points
Regression line
Y
X
10
9
8
7
6
5
4
3
2
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

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:

Code:
```
import pandas as pd
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor

data = {
    'Area': [1200, 1500, 1800, 2000],
    'Rooms': [2, 3, 3, 4],
    'Price': [250000, 300000, 320000, 370000]
}

df = pd.DataFrame(data)

X = df[['Area', 'Rooms']]
y = df['Price']

X_with_const = sm.add_constant(X)

model = sm.OLS(y, X_with_const).fit()

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 Factor (VIF):\n")
print(vif_data)
```

Output:-

```
Variance Inflation Factor (VIF):

  Feature          VIF
0    Area   127.796923

1   Rooms   127.796923
```

<u>Code for Summary:</u>

```
print("Regression Summary:\n")
print(model.summary())
```

Output:-

```
Regression Summary:

                            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:                Tue, 29 Jul 2025   Prob (F-statistic):             0.0377
Time:                        16:17:38   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
==============================================================================
```

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.*)

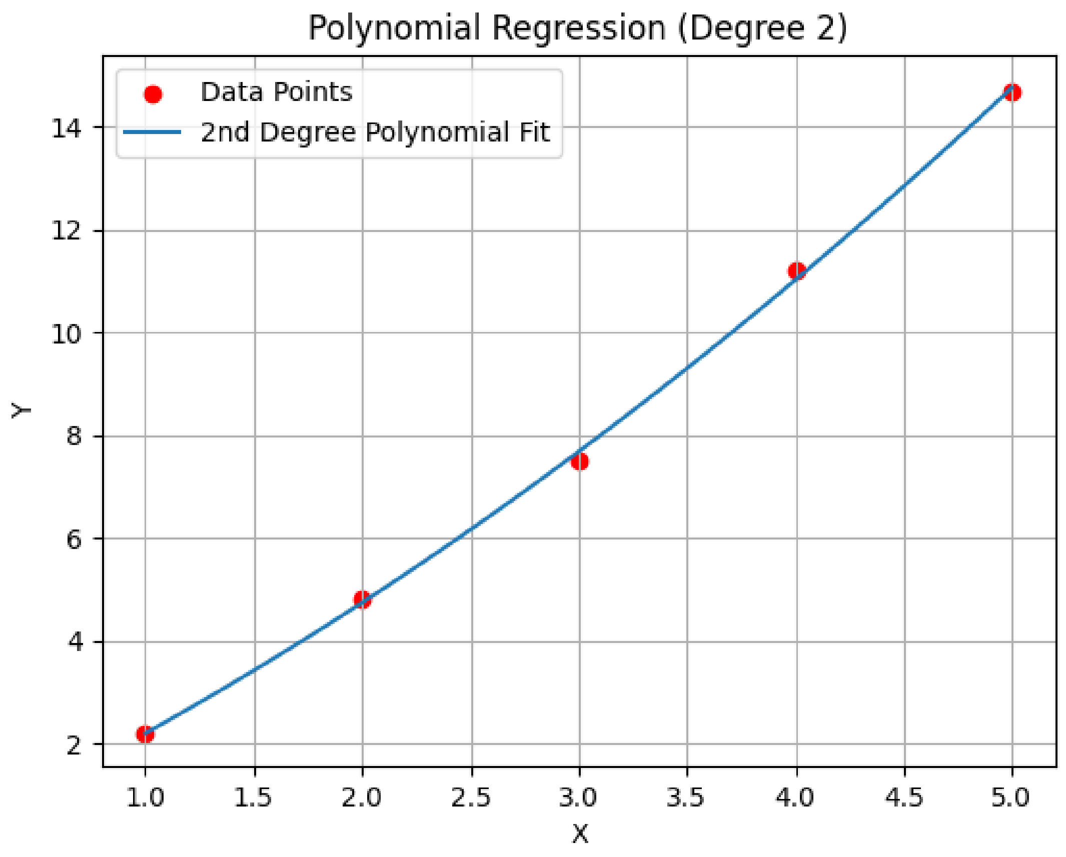
Answer:

Code:

```
import numpy as np
import matplotlib.pyplot as plt
from numpy.polynomial.polynomial import Polynomial

X = np.array([1, 2, 3, 4, 5])
Y = np.array([2.2, 4.8, 7.5, 11.2, 14.7])

coef = Polynomial.fit(X, Y, 2).convert().coef
poly = Polynomial(coef)

x_plot = np.linspace(X.min(), X.max(), 100)
y_plot = poly(x_plot)

plt.scatter(X, Y, color='red', label='Data Points')
plt.plot(x_plot, y_plot, label='2nd Degree Polynomial Fit')
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Polynomial Regression (Degree 2)')
plt.legend()
plt.grid(True)
plt.show()

print('Coefficients:', coef)
```

Output:-

```
Coefficients: [0.06 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.*)

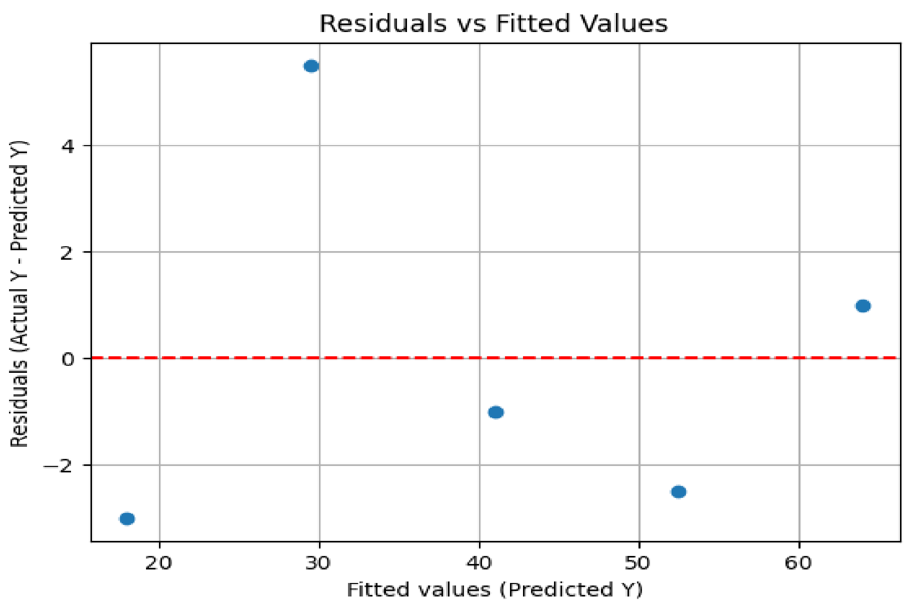
Answer:

Assessing Heteroscedasticity:

- If residuals are randomly scattered around zero without a clear pattern or changing spread, heteroscedasticity is unlikely.
- Patterns like funnel shape or increasing/decreasing spread indicate heteroscedasticity (non-constant variance).

Code:-
```
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm

X = np.array([10, 20, 30, 40, 50])
Y = np.array([15, 35, 40, 50, 65])

X_const = sm.add_constant(X)

model = sm.OLS(Y, X_const).fit()

Y_pred = model.predict(X_const)

residuals = Y - Y_pred

plt.scatter(Y_pred, residuals)
plt.axhline(y=0, color='r', linestyle='--')
plt.xlabel('Fitted values (Predicted Y)')
plt.ylabel('Residuals (Actual Y - Predicted Y)')
plt.title('Residuals vs Fitted Values')
plt.grid(True)
plt.show()

print(model.summary())
```

Output:-



```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.965
Model:                            OLS   Adj. R-squared:                  0.954
Method:                 Least Squares   F-statistic:                     83.53
Date:                Tue, 29 Jul 2025   Prob (F-statistic):            0.00277
Time:                        16:39:25   Log-Likelihood:                -12.723
No. Observations:                   5   AIC:                             29.45
Df Residuals:                       3   BIC:                             28.66
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          6.5000      4.173      1.558      0.217      -6.781      19.781
x1             1.1500      0.126      9.139      0.003       0.750       1.550
==============================================================================
Omnibus:                          nan   Durbin-Watson:                   2.716
Prob(Omnibus):                    nan   Jarque-Bera (JB):                0.698
Skew:                           0.845   Prob(JB):                        0.705
Kurtosis:                       2.298   Cond. No.                         77.8
==============================================================================
```

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:

Steps:-

- First, diagnose and reduce multicollinearity via feature selection, regularization, or PCA.
- Then, address heteroscedasticity by transforming variables or using robust regression methods.
- This improves coefficient reliability, prediction accuracy, and model interpretability for pricing predictions in real estate.