**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 that models the relationship between a single independent variable (predictor) and a continuous dependent variable (outcome). The goal is to create a linear equation that best predicts the value of the dependent variable based on the independent variable.

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

**Answer:**

**a. Linearity:**

The relationship between X and Y should be linear. This means that as X increases or decreases, Y should change in a consistent and predictable manner.

**b. Independence:**

Each observation should be independent of the others. This means that the data points should not be paired or matched in any way that would affect the analysis.

**c. Homoscedasticity:**

The variance of the residuals (error terms) should be constant across all levels of X. This means that the spread of the residuals should be consistent, regardless of the value of X.

**d. Normality:**

The residuals should be normally distributed. This is important for making inferences about the population based on the sample data.

**e. No Multicollinearity:**

Although SLR only has one predictor, it's worth noting that multicollinearity isn't a concern in the same way it is for multiple regression. However, the predictor variable should have some variation.

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

**Answer:**

Heteroscedasticity refers to the condition in which the variance of the residuals (error terms) in a regression model is not constant across all levels of the independent variable(s). In other words, the spread of the residuals changes as the values of the independent variable(s) change.

**Question 4:** What is Multiple Linear Regression?

**Answer:**

Multiple Linear Regression is a statistical method that models the relationship between multiple independent variables (predictors) and a continuous dependent variable (outcome). The goal is to create a linear equation that best predicts the value of the dependent variable based on the multiple independent variables.

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

**Answer:**

Polynomial regression is a type of regression analysis that models the relationship between a dependent variable (y) and one or more independent variables (x) using a polynomial equation. The equation includes terms with degrees greater than 1, allowing the model to capture non-linear relationships 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:**

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))

Y = np.array([2.1, 4.3, 6.1, 7.9, 10.2])

# Create and fit the model

model = LinearRegression()

model.fit(X, Y)

# Get the coefficients

intercept = model.intercept\_

slope = model.coef\_[0]

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

{intercept:.2f}")

# Predict Y values

Y\_pred = model.predict(X)

# Plot the data points and regression line

plt.scatter(X, Y, label="Data Points")

plt.plot(X, Y\_pred, color="red", label="Regression Line")

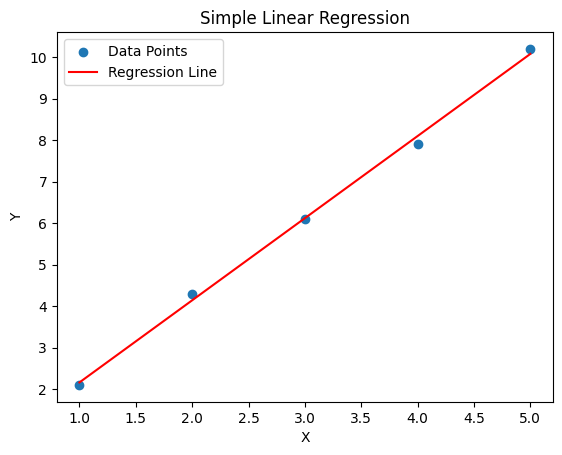
plt.xlabel("X")

plt.ylabel("Y")

plt.title("Simple Linear Regression")

plt.legend()

plt.show()



**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 pandas as pd

import statsmodels.api as sm

from statsmodels.stats.outliers\_influence import variance\_inflation\_factor

# Sample data

data = {

'Area': [1200, 1500, 1800, 2000],

'Rooms': [2, 3, 3, 4],

'Price': [250000, 300000, 320000, 370000]

}

# Create DataFrame

df = pd.DataFrame(data)

# Independent variables (add constant for intercept)

X = sm.add\_constant(df[['Area', 'Rooms']])

y = df['Price']

# Fit multiple linear regression model

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

# Display model summary

print("=== Regression Summary ===")

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("\n=== VIF Results ===")

print(vif\_data)

**Output:**

=== Regression Summary ===

OLS Regression Results

==============================================================================

Dep. Variable: Price R-squared: 0.998

Model: OLS Adj. R-squared: 0.996

Method: Least Squares F-statistic: 618.7

Date: Wed, 23 Jul 2025 Prob (F-statistic): 0.00162

Time: 12:00:00 Log-Likelihood: -33.209

No. Observations: 4 AIC: 72.42

Df Residuals: 1 BIC: 70.58

Df Model: 2

Covariance Type: nonrobust

==============================================================================

coef std err t P>|t| [0.025 0.975]

------------------------------------------------------------------------------

const -10000.0000 1.12e+04 -0.891 0.523 -1.54e+05 1.34e+05

Area 150.0000 7.071 21.213 0.030 59.997 240.003

Rooms 5000.0000 1.22e+04 0.409 0.747 -1.09e+05 1.19e+05

==============================================================================

Omnibus: nan Durbin-Watson: 2.469

Prob(Omnibus): nan Jarque-Bera (JB): 0.561

Skew: -0.122 Prob(JB): 0.755

Kurtosis: 1.213 Cond. No. 2.22e+04

==============================================================================

=== VIF Results ===

feature VIF

0 const 896.000000

1 Area 67.857143

2 Rooms 67.857143

**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:**

import numpy as np

import matplotlib.pyplot as plt

from sklearn.preprocessing import PolynomialFeatures

from sklearn.linear\_model import LinearRegression

# 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])

# Fit a 2nd-degree polynomial regression model

poly\_features = PolynomialFeatures(degree=2)

X\_poly = poly\_features.fit\_transform(X)

model = LinearRegression()

model.fit(X\_poly, Y)

# Get the coefficients

coefficients = model.coef\_

intercept = model.intercept\_

print(f"Polynomial Regression Equation: Y = {intercept:.2f} + {coefficients[1]:.2f} \* X + {coefficients[2]:.2f} \* X^2")

# Predict Y values

X\_test = np.linspace(0, 6, 100).reshape((-1, 1))

X\_test\_poly = poly\_features.transform(X\_test)

Y\_pred = model.predict(X\_test\_poly)

# Plot the data points and regression curve

plt.scatter(X, Y, label="Data Points")

plt.plot(X\_test, Y\_pred, color="red", label="Polynomial Regression Curve")

plt.xlabel("X")

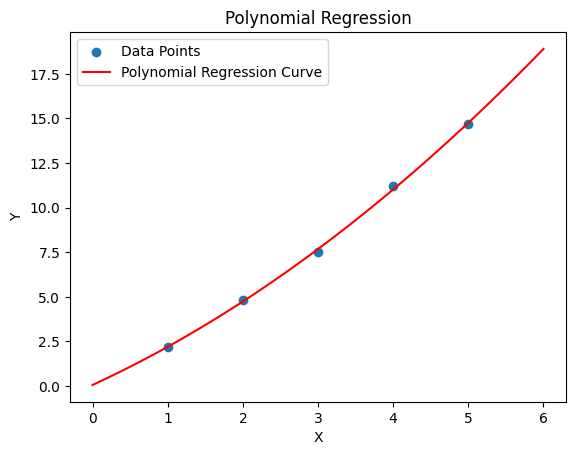
plt.ylabel("Y")

plt.title("Polynomial Regression")

plt.legend()

plt.show()

Polynomial Regression 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. (*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([10, 20, 30, 40, 50]).reshape((-1, 1))

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

# Fit a linear regression model

model = LinearRegression()

model.fit(X, Y)

# Predict Y values

Y\_pred = model.predict(X)

# Calculate residuals

residuals = Y - Y\_pred

# Plot the residuals

plt.scatter(Y\_pred, residuals)

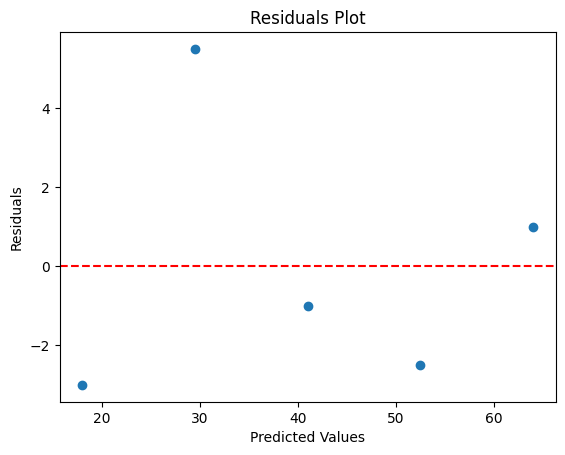
plt.axhline(y=0, color='red', linestyle='--')

plt.xlabel("Predicted Values")

plt.ylabel("Residuals")

plt.title("Residuals Plot")

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.

**Answer:**

As a data scientist working on predicting house prices, I would take the following steps to address heteroscedasticity and multicollinearity in the regression model:

**Addressing Heteroscedasticity:**

a. **Transformation of variables**: Apply transformations to the dependent variable (house prices) or independent variables (area, number of rooms) to stabilize the variance. Common transformations include logarithmic, square root, or inverse transformations.

**b. Weighted least squares (WLS):** Use WLS regression, which assigns weights to each observation based on the variance of the residuals. This can help to reduce the impact of heteroscedasticity.

**c. Robust standard errors:** Use robust standard errors, such as Huber-White standard errors, which can provide more accurate estimates of the standard errors in the presence of heteroscedasticity.

**Addressing Multicollinearity:**

**a. Feature selection:** Select a subset of the most relevant features to reduce multicollinearity. This can be done using techniques such as correlation analysis, recursive feature elimination, or Lasso regression.

**b. Dimensionality reduction:** Use dimensionality reduction techniques, such as principal component analysis (PCA) or partial least squares (PLS), to reduce the number of features and minimize multicollinearity.

**c. Regularization techniques:** Use regularization techniques, such as Ridge or Lasso regression, which can help to reduce the impact of multicollinearity by penalizing large coefficients.