**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:**

|  |
| --- |
| a statistical method used to model the linear relationship between a single independent variable and a single dependent variable |

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

**Answer:**

|  |
| --- |
| linearity (the relationship between variables is a straight line), independence (errors are not correlated), homoscedasticity (errors have constant variance), and normality (errors are normally distributed) |

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

**Answer:**

|  |
| --- |
| It is important to address because it violates a key assumption of ordinary least squares (OLS) regression, which can lead to unreliable standard errors, inefficient estimates, and invalid confidence intervals and hypothesis tests. |

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

**Answer:**

|  |
| --- |
| What is Multiple Linear Regression? |

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

**Answer:**

|  |
| --- |
| Polynomial regression models a non-linear relationship between a dependent and independent variable by fitting a curved line, while linear regression fits a straight line and assumes a linear relationship |

**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:**

|  |
| --- |
| Polynomial regression models non-linear relationships by fitting a curved line to data, while linear regression fits a straight line and assumes a linear relationship |

**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  # 1. Prepare the data  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()  print("### Model Summary ###")  print(model.summary())  vif\_data = pd.DataFrame()  vif\_data["Variable"] = X\_with\_const.columns  vif\_data["VIF"] = [variance\_inflation\_factor(X\_with\_const.values, i) for i in range(X\_with\_const.shape[1])]  print("\n### Multicollinearity Check (VIF) ###")  print(vif\_data) |

**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  # 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\_features = PolynomialFeatures(degree=2)  X\_poly = poly\_features.fit\_transform(X)  # Fit the polynomial regression model  model = LinearRegression()  model.fit(X\_poly, Y)  # Predict Y values for plotting the curve  X\_plot = np.linspace(min(X), max(X), 100).reshape(-1, 1)  X\_plot\_poly = poly\_features.transform(X\_plot)  Y\_pred = model.predict(X\_plot\_poly)  # Print the coefficients  print(f"Coefficients: {model.coef\_}")  print(f"Intercept: {model.intercept\_}")  # Plot the data and the fitted curve  plt.scatter(X, Y, label='Original Data')  plt.plot(X\_plot, Y\_pred, color='red', label='2nd Degree Polynomial Fit')  plt.xlabel('X')  plt.ylabel('Y')  plt.title('Polynomial Regression (Degree 2)')  plt.legend()  plt.grid(True)  plt.show()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])  # Create polynomial features (degree 2)  poly\_features = PolynomialFeatures(degree=2)  X\_poly = poly\_features.fit\_transform(X)  # Fit the polynomial regression model  model = LinearRegression()  model.fit(X\_poly, Y)  # Predict Y values for plotting the curve  X\_plot = np.linspace(min(X), max(X), 100).reshape(-1, 1)  X\_plot\_poly = poly\_features.transform(X\_plot)  Y\_pred = model.predict(X\_plot\_poly)  # Print the coefficients  print(f"Coefficients: {model.coef\_}")  print(f"Intercept: {model.intercept\_}")  # Plot the data and the fitted curve  plt.scatter(X, Y, label='Original Data')  plt.plot(X\_plot, Y\_pred, color='red', label='2nd Degree Polynomial Fit')  plt.xlabel('X')  plt.ylabel('Y')  plt.title('Polynomial Regression (Degree 2)')  plt.legend()  plt.grid(True)  plt.show() |

**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  # Data  X = np.array([10, 20, 30, 40, 50]).reshape(-1, 1)  Y = np.array([15, 35, 40, 50, 65])  # Train a linear regression model  model = LinearRegression()  model.fit(X, Y)  # Predict Y values  Y\_pred = model.predict(X)  # Calculate residuals  residuals = Y - Y\_pred  # Create the residuals plot  plt.figure(figsize=(8, 6))  plt.scatter(X, residuals, color='blue')  plt.axhline(y=0, color='red', linestyle='--')  plt.xlabel("Independent Variable (X)")  plt.ylabel("Residuals")  plt.title("Residuals Plot")  plt.grid(True)  plt.show()  print("Assessing Heteroscedasticity:")  print("Examine the residuals plot for a consistent spread of points around the zero line.")  print("If the spread of residuals changes systematically with the independent variable (e.g., a fan shape),")  print("it suggests heteroscedasticity. If the spread is relatively constant, it suggests homoscedasticity.")  print("\nResiduals:")  for i, res in enumerate(residuals):  print(f"X={X[i][0]}, Residual={res:.2f}") |

**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:**

|  |
| --- |
| 1**. Detect heteroscedasticity and multicollinearity**  2. **Detecting multicollinearity**  **3.** **Correlation matrix**  **4.** **Variance Inflation Factor (VIF)**  **5.** **Detecting heteroscedasticity**  **5.** **Residual plot**  **6.** **Statistical tests:** |