Regression & Its Evaluation | Assignment

Question 1: What is Simple Linear Regression?

Ans:

| Simple Linear Regression is a basic statistical method that finds the relationship between two  variables: one that predicts (X) and one that is predicted (Y). It creates a straight line that best  fits the data points. The line has this formula: Y = a + bX, where "a" is where the line crosses  the Y-axis and "b" is the slope of the line. The main goal is to find values for "a" and "b" that  make the line fit the data as closely as possible. |
| --- |

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

Ans:

| Simple Linear Regression relies on these key assumptions:  1. Linearity: The relationship between X and Y can be drawn as a straight line.  2. Independence: Each data point doesn't influence other data points.  3. Equal Variance: The spread of points around the line is the same everywhere  (homoscedasticity).  4. Normal Errors: The mistakes in predictions follow a bell curve distribution.  5. No Extreme Values: There aren't unusual data points that pull the line in odd directions.  6. Correct Measurement: The X variable is measured accurately.  If these assumptions aren't met, the regression results might be misleading. |
| --- |

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

Ans:

| Heteroscedasticity means the errors in a regression model have uneven spread. Think of it like  this: when you plot prediction errors, they form a pattern (like a funnel shape) instead of being  randomly scattered.  It's important to fix heteroscedasticity because:  1. It makes your confidence intervals and p-values wrong  2. Your statistical tests become unreliable  3. Some predictions will be much less accurate than others  4. It suggests your model might be missing important variables  To fix it, you can transform your data (like using logarithms), use weighted methods that give  less importance to more variable data points, or use special robust techniques that work better  with uneven error patterns. |
| --- |

Question 4: What is Multiple Linear Regression?

Ans:

| Multiple Linear Regression is an extension of Simple Linear Regression that uses two or more  independent variables to predict a dependent variable. The equation has the form Y = a + b₁X₁  + b₂X₂ + ... + bₙXₙ, where each X represents a different predictor variable and each b  represents that variable's coefficient. |
| --- |

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

regression?

Ans:

| Polynomial regression is a form of regression analysis where the relationship between the  independent variable and the dependent variable is modeled as an nth degree polynomial  function. Instead of fitting a straight line to the data, polynomial regression fits a curved line.  It differs from linear regression in these ways:  1. It models curved relationships rather than straight lines  2. It uses transformed versions of the original variables (squared, cubed terms)  3. It can capture more complex patterns in the data  4. It has higher risk of overfitting, especially with higher-degree polynomials  5. The effect of X on Y varies depending on the value of X, rather than being constant |
| --- |

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.)

Ans:

| 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) # Feature matrix  Y = np.array([2.1, 4.3, 6.1, 7.9, 10.2]) # Target vector  # Create and fit the model  model = LinearRegression()  model.fit(X, Y)  # Predict values  Y\_pred = model.predict(X)  # Print the slope and intercept  print("Slope (Coefficient):", model.coef\_[0])  print("Intercept:", model.intercept\_)  # Plot the data points and the regression line  plt.scatter(X, Y, color='blue', label='Actual Data')  plt.plot(X, Y\_pred, color='red', label='Regression Line')  plt.title('Simple Linear Regression')  plt.xlabel('X')  plt.ylabel('Y')  plt.legend()  plt.grid(True)  plt.show()  **Output:** |
| --- |

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.)

Ans:

| import pandas as pd  import statsmodels.api as sm  from statsmodels.stats.outliers\_influence import variance\_inflation\_factor  # 1. Create DataFrame  data = pd.DataFrame({  'Area': [1200, 1500, 1800, 2000],  'Rooms': [2, 3, 3, 4],  'Price': [250000, 300000, 320000, 370000]  })  # 2. Define independent (X) and dependent (y) variables  X = data[['Area', 'Rooms']]  y = data['Price']  # 3. Add constant term for intercept  X\_const = sm.add\_constant(X)  # 4. Fit the model  model = sm.OLS(y, X\_const).fit()  # 5. Print the model summary  print(model.summary())  # 6. 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 Factor (VIF):")  print(vif\_data)  **Output:** |
| --- |

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.)

Ans:

| import numpy as np  import matplotlib.pyplot as plt  from sklearn.linear\_model import LinearRegression  from sklearn.preprocessing import PolynomialFeatures  # Input 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 polynomial regression model  model = LinearRegression()  model.fit(X\_poly, Y)  # Predict using the model  X\_range = np.linspace(1, 5, 100).reshape(-1, 1)  X\_range\_poly = poly.transform(X\_range)  Y\_pred = model.predict(X\_range\_poly)  # Plotting  plt.scatter(X, Y, color='red', label='Original Data')  plt.plot(X\_range, Y\_pred, color='blue', label='Polynomial Fit (degree 2)')  plt.title('Polynomial Regression (Degree 2)')  plt.xlabel('X')  plt.ylabel('Y')  plt.legend()  plt.grid(True)  plt.show()  **Output:** |
| --- |

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.)

Ans:

| 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 Linear Regression  model = LinearRegression()  model.fit(X, Y)  Y\_pred = model.predict(X)  # Calculate Residuals  residuals = Y - Y\_pred  # Residuals plot  plt.scatter(X, residuals, color='purple', marker='o')  plt.axhline(y=0, color='black', linestyle='--')  plt.title('Residuals Plot')  plt.xlabel('X')  plt.ylabel('Residuals')  plt.grid(True)  plt.show()  **Output:** |
| --- |

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.

Ans:

| To address heteroscedasticity and multicollinearity in a real estate price prediction model:  For Heteroscedasticity:  1. Transform variables: Log transform price and/or features  2. Weighted Least Squares: Give different weights to observations  3. Robust standard errors: Use heteroscedasticity-consistent SEs  For Multicollinearity:  1. Calculate VIF: Identify highly correlated features  2. Feature selection: Remove or combine highly correlated features  3. Regularization techniques: Consider Ridge or Lasso regression  By addressing these issues, we can improve the model's reliability and accuracy for predicting  house prices. |
| --- |