Assignment Code: DA-AG-010

Regression & Its Evaluation | **Assignment**

Name : Sourabh Ranbhise
Email : sourabhranbhise@gmail.com
Assignment Name : Regression & Its Evaluation |
Github Link : LINK

Question 1: What is Simple Linear Regression?

Answer :

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?

Answer:

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?

Answer:

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?

Answer:

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 $Y = a + b_1X_1 + b_2X_2 + ... + b_nX_n$, 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?

Answer:

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

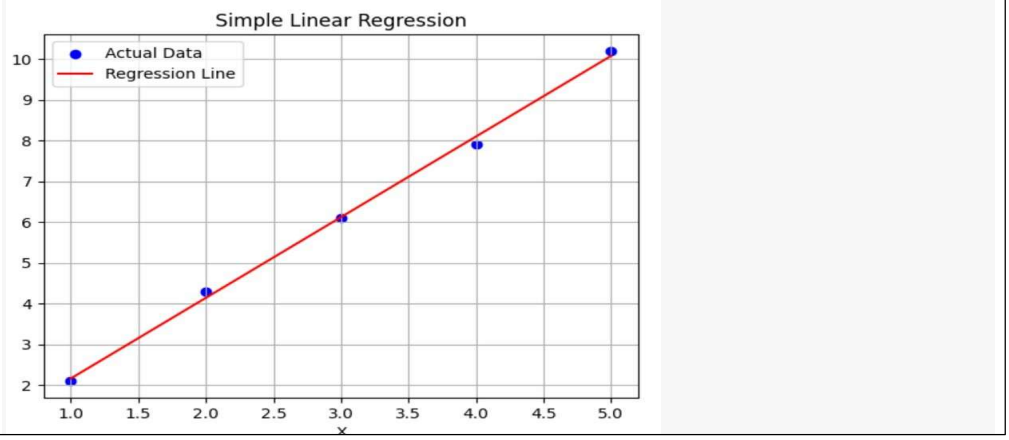
Question 6: Implement a Python program to fit a Simple Linear Regression model to the following sample data:

- 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.*)
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 train the model
model = LinearRegression()
model.fit(X, Y)

# Predict values
Y_pred = model.predict(X)

# Plotting
plt.scatter(X, Y, color='blue', label='Actual Data')  # data points
plt.plot(X, Y_pred, color='red', label='Regression Line')  # 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.*)

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]
}

df = pd.DataFrame(data)

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

# Add constant for intercept
X_with_const = sm.add_constant(X)

# Fit Multiple Linear Regression model
model = sm.OLS(y, X_with_const).fit()

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

# Calculate VIF for each feature
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 :

```
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:                Thu, 17 Jul 2025   Prob (F-statistic):             0.0377
Time:                        10:25:41   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
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.01e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

Variance Inflation Factor (VIF):

  Feature         VIF
0    Area  127.796923
1   Rooms  127.796923
```

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]

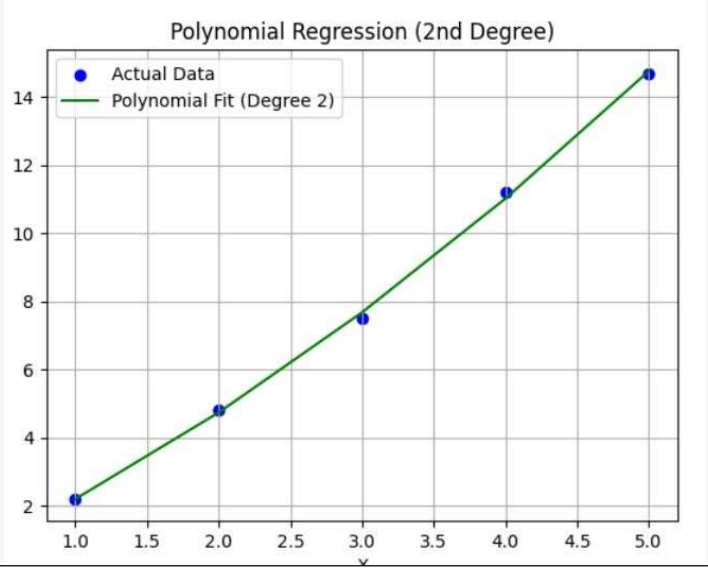
Fit a **2nd-degree polynomial** and plot the resulting curve.
(*Include your Python code and output in the code box below.*)

Answer:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures

# 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 the model
model = LinearRegression()
model.fit(X_poly, Y)

# Predict
Y_pred = model.predict(X_poly)

# Plot
plt.scatter(X, Y, color='blue', label='Actual Data')
plt.plot(X, Y_pred, color='green', label='Polynomial Fit (Degree 2)')
plt.title('Polynomial Regression (2nd Degree)')
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:

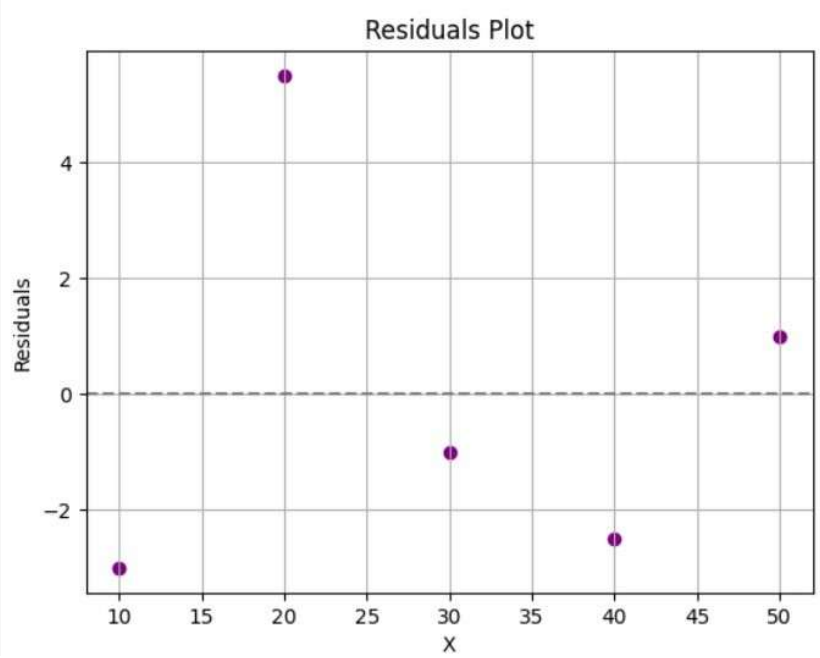
- X = [10, 20, 30, 40, 50]
- Y = [15, 35, 40, 50, 65]

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

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 linear regression model
model = LinearRegression()
model.fit(X, Y)
Y_pred = model.predict(X)

# Calculate residuals (actual - predicted)
residuals = Y - Y_pred

# Plot residuals
plt.scatter(X, residuals, color='purple')
plt.axhline(y=0, color='gray', 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.

Answer:

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.