**2311cs020268-day24**

**Using the same data set of Civil\_Engineering\_Regression\_Dataset.csv**

**Part 4: Multiple Linear Regression**

**12. Compare the R-squared values of simple and multiple linear regression. Which model performs better?**

**13. What does the Adjusted R-squared value indicate about the multiple regression model?**

**14. How does multicollinearity affect the model? Check Variance Inflation Factor (VIF) to detect multicollinearity.**

**Code:**

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**from sklearn.linear\_model import LinearRegression**

**from sklearn.metrics import mean\_squared\_error, r2\_score**

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

**try:**

**df = pd.read\_csv("Civil\_Engineering\_Regression\_Dataset.csv")**

**print("CSV file loaded successfully!\n")**

**except FileNotFoundError:**

**print("Error: CSV file not found. Check the file path.")**

**exit()**

**required\_columns = ["Building Height", "Material Quality", "Labor Cost", "Concrete Strength", "Foundation Depth", "Construction Cost"]**

**for col in required\_columns:**

**if col not in df.columns:**

**print(f"Error: Column '{col}' is missing. Check your dataset headers.")**

**exit()**

**X\_multi = df[["Building Height", "Material Quality", "Labor Cost", "Concrete Strength", "Foundation Depth"]]**

**y = df["Construction Cost"]**

**# Handle missing or non-numeric values**

**X\_multi = X\_multi.apply(pd.to\_numeric, errors="coerce")**

**y = pd.to\_numeric(y, errors="coerce")**

**df.dropna(inplace=True)**

**X\_simple = df[["Building Height"]]**

**simple\_model = LinearRegression()**

**simple\_model.fit(X\_simple, y)**

**r2\_simple = r2\_score(y, simple\_model.predict(X\_simple))**

**multi\_model = LinearRegression()**

**multi\_model.fit(X\_multi, y)**

**r2\_multi = r2\_score(y, multi\_model.predict(X\_multi))**

**n = len(y)**

**p = X\_multi.shape[1] # Number of predictors**

**adjusted\_r2\_multi = 1 - ((1 - r2\_multi) \* (n - 1) / (n - p - 1))**

**print(f"📊 Model Performance Comparison:")**

**print(f"Simple Linear Regression R-squared: {r2\_simple:.4f}")**

**print(f"Multiple Linear Regression R-squared: {r2\_multi:.4f}")**

**print(f"Multiple Linear Regression Adjusted R-squared: {adjusted\_r2\_multi:.4f}")**

**print("\n📌 Adjusted R-squared Interpretation:")**

**if adjusted\_r2\_multi > r2\_simple:**

**print("The multiple regression model explains more variance while accounting for additional predictors.")**

**else:**

**print("The additional variables do not significantly improve the model, and some may be unnecessary.")**

**# Checking Multicollinearity using VIF**

**vif\_data = pd.DataFrame()**

**vif\_data["Feature"] = X\_multi.columns**

**vif\_data["VIF"] = [variance\_inflation\_factor(X\_multi.values, i) for i in range(X\_multi.shape[1])]**

**print("\n🔍 Variance Inflation Factor (VIF) Results:")**

**print(vif\_data)**

**print("\n📌 Multicollinearity Interpretation:")**

**high\_vif\_features = vif\_data[vif\_data["VIF"] > 10]["Feature"].tolist()**

**if high\_vif\_features:**

**print(f"Warning: High multicollinearity detected in {high\_vif\_features}. Consider removing or combining correlated variables.")**

**else:**

**print("No significant multicollinearity detected. The predictors are independent enough for a reliable model.")**