# Ex.No: 1 Regression Model for Real-Time System

Date: 08/07/2024

## Aim:

To compare the performance of Simple Linear Regression, Multiple Linear Regression, and Polynomial Regression models in predicting the GPA of students based on their study time, attendance, and previous scores.

## Procedure:

1. Data Loading and Preparation:

- Load the dataset containing student performance details using pandas.

- Handle missing values by dropping rows with any missing data.

2. Feature and Target Selection:

- Select 'StudyTimeWeekly', 'Attendance', and 'PreviousScores' as features for multiple and polynomial regression.

- Select 'GPA' as the target variable.

3. Data Splitting:

- Split the data into training and testing sets using train\_test\_split with a test size of 20%.

4. Simple Linear Regression:

- Use 'StudyTimeWeekly' as the feature.

- Train a Simple Linear Regression model.

5. Multiple Linear Regression:

- Use 'StudyTimeWeekly', 'Attendance', and 'PreviousScores' as features.

- Train a Multiple Linear Regression model.

6. Polynomial Regression:

- Use polynomial features of 'StudyTimeWeekly', 'Attendance', and 'PreviousScores' with degree=2.

- Train a Polynomial Regression model.

7. Prediction and Evaluation:

- Use the trained models to make predictions on the test set.

- Calculate evaluation metrics (MSE, R2) for each model.

- Determine the best model based on the metrics.

8. Visualization:

- Plot the actual vs. predicted GPA for the first 50 data points for each model.

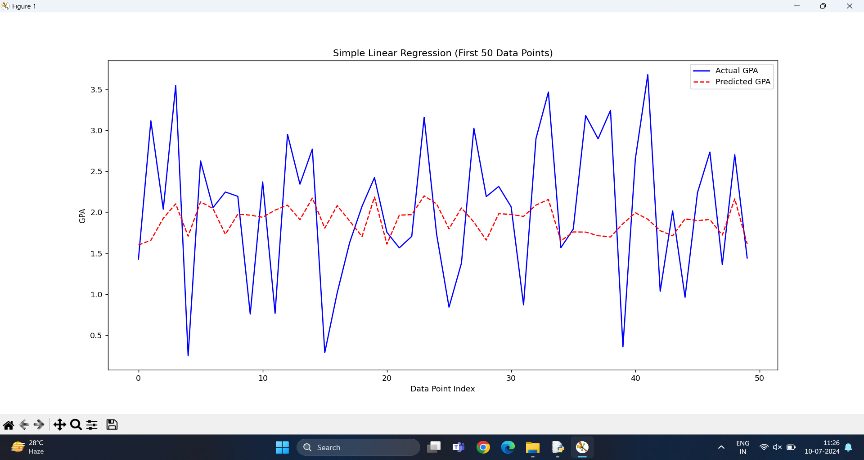
## Code:

import pandas as pd  
from sklearn.model\_selection import train\_test\_split  
import seaborn as sns  
import matplotlib.pyplot as plt  
from sklearn.linear\_model import LinearRegression  
from sklearn.preprocessing import PolynomialFeatures  
from sklearn.metrics import mean\_squared\_error, r2\_score  
  
class Layoff:  
   
 def \_\_init\_\_(self):  
 self.data = None  
 self.encoded\_data = None  
   
 def preprocess(self):  
 self.data = pd.read\_csv("C:/Machine learning/Student\_performance\_data \_.csv")  
   
 # Missing values  
 self.data = self.data.dropna()  
   
 # Categorical features  
 categorical\_features = self.data.select\_dtypes(include=['object']).columns  
 self.encoded\_data = pd.get\_dummies(self.data, columns=categorical\_features)  
   
 # Correlation heatmap  
 corr\_matrix = self.encoded\_data.corr()  
 plt.figure(figsize=(14, 10))  
 sns.heatmap(corr\_matrix, annot=True, cmap='coolwarm', linewidths=0.5)  
 plt.title('Correlation Heatmap of Student Performance Data')  
 plt.show()  
   
 # Simple Linear Regression  
 self.simple\_linear\_regression()  
   
 # Multiple Linear Regression  
 self.multiple\_linear\_regression()  
   
 # Polynomial Regression  
 self.polynomial\_regression()  
  
 def simple\_linear\_regression(self):  
 feature = 'StudyTimeWeekly'   
 target = 'GPA'  
 X = self.encoded\_data[[feature]]  
 y = self.encoded\_data[target]  
   
 # Training and testing sets  
 X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
   
 # Simple Linear Regression model  
 model = LinearRegression()  
 model.fit(X\_train, y\_train)  
   
 # Predict  
 y\_pred = model.predict(X\_test)  
   
 # Plot  
 plt.figure(figsize=(12, 8))  
 plt.plot(range(50), y\_test[:50], color='blue', label='Actual GPA')  
 plt.plot(range(50), y\_pred[:50], color='red', linestyle='dashed', label='Predicted GPA')  
 plt.title('Simple Linear Regression (First 50 Data Points)')  
 plt.xlabel('Data Point Index')  
 plt.ylabel('GPA')  
 plt.legend()  
 plt.show()  
  
 def multiple\_linear\_regression(self):  
 features = ['StudyTimeWeekly', 'Attendance', 'PreviousScores'] # Add more relevant features  
 target = 'GPA'  
 X = self.encoded\_data[features]  
 y = self.encoded\_data[target]  
   
 # Training and testing sets  
 X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
   
 # Multiple Linear Regression model  
 model = LinearRegression()  
 model.fit(X\_train, y\_train)  
   
 # Predict  
 y\_pred = model.predict(X\_test)  
   
 # Metrics  
 mse = mean\_squared\_error(y\_test, y\_pred)  
 r2 = r2\_score(y\_test, y\_pred)  
 print(f'Multiple Linear Regression - MSE: {mse}, R2: {r2}')  
   
 # Plot  
 plt.figure(figsize=(12, 8))  
 plt.plot(range(50), y\_test[:50], color='blue', label='Actual GPA')  
 plt.plot(range(50), y\_pred[:50], color='red', linestyle='dashed', label='Predicted GPA')  
 plt.title('Multiple Linear Regression (First 50 Data Points)')  
 plt.xlabel('Data Point Index')  
 plt.ylabel('GPA')  
 plt.legend()  
 plt.show()  
  
 def polynomial\_regression(self):  
 features = ['StudyTimeWeekly', 'Attendance', 'PreviousScores'] # Add more relevant features  
 target = 'GPA'  
 X = self.encoded\_data[features]  
 y = self.encoded\_data[target]  
   
 # Training and testing sets  
 X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
   
 # Polynomial Regression model  
 poly\_features = PolynomialFeatures(degree=2)  
 X\_train\_poly = poly\_features.fit\_transform(X\_train)  
 X\_test\_poly = poly\_features.transform(X\_test)  
   
 model = LinearRegression()  
 model.fit(X\_train\_poly, y\_train)  
   
 # Predict  
 y\_pred = model.predict(X\_test\_poly)  
   
 # Metrics  
 mse = mean\_squared\_error(y\_test, y\_pred)  
 r2 = r2\_score(y\_test, y\_pred)  
 print(f'Polynomial Regression - MSE: {mse}, R2: {r2}')  
   
 # Plot  
 plt.figure(figsize=(12, 8))  
 plt.scatter(range(len(y\_test)), y\_test, color='blue', label='Actual GPA')  
 plt.scatter(range(len(y\_test)), y\_pred, color='red', label='Predicted GPA')  
 plt.title('Polynomial Regression (Test Data Points)')  
 plt.xlabel('Data Point Index')  
 plt.ylabel('GPA')  
 plt.legend()  
 plt.show()  
  
def main():  
 l = Layoff()  
 l.preprocess()  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

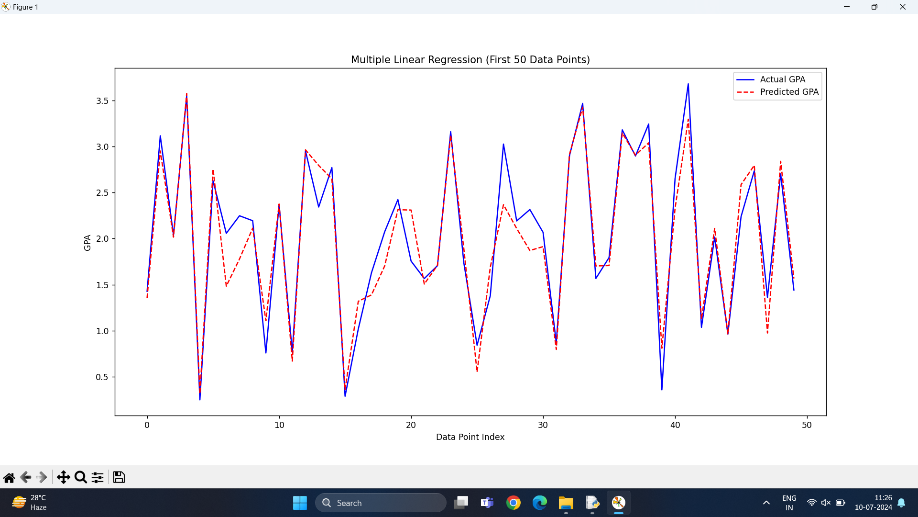
## Final Output:

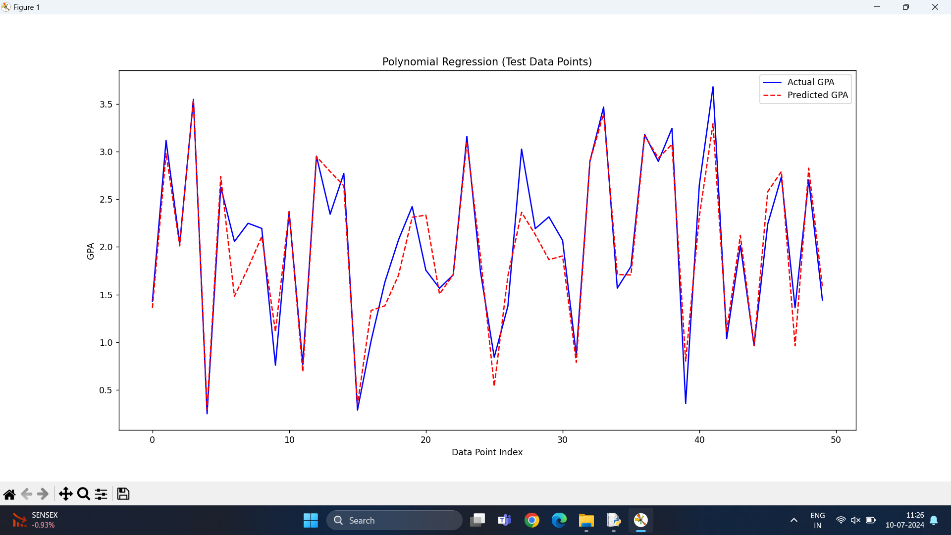
## Plots:

Plot showing actual vs. predicted values for Simple Linear Regression:

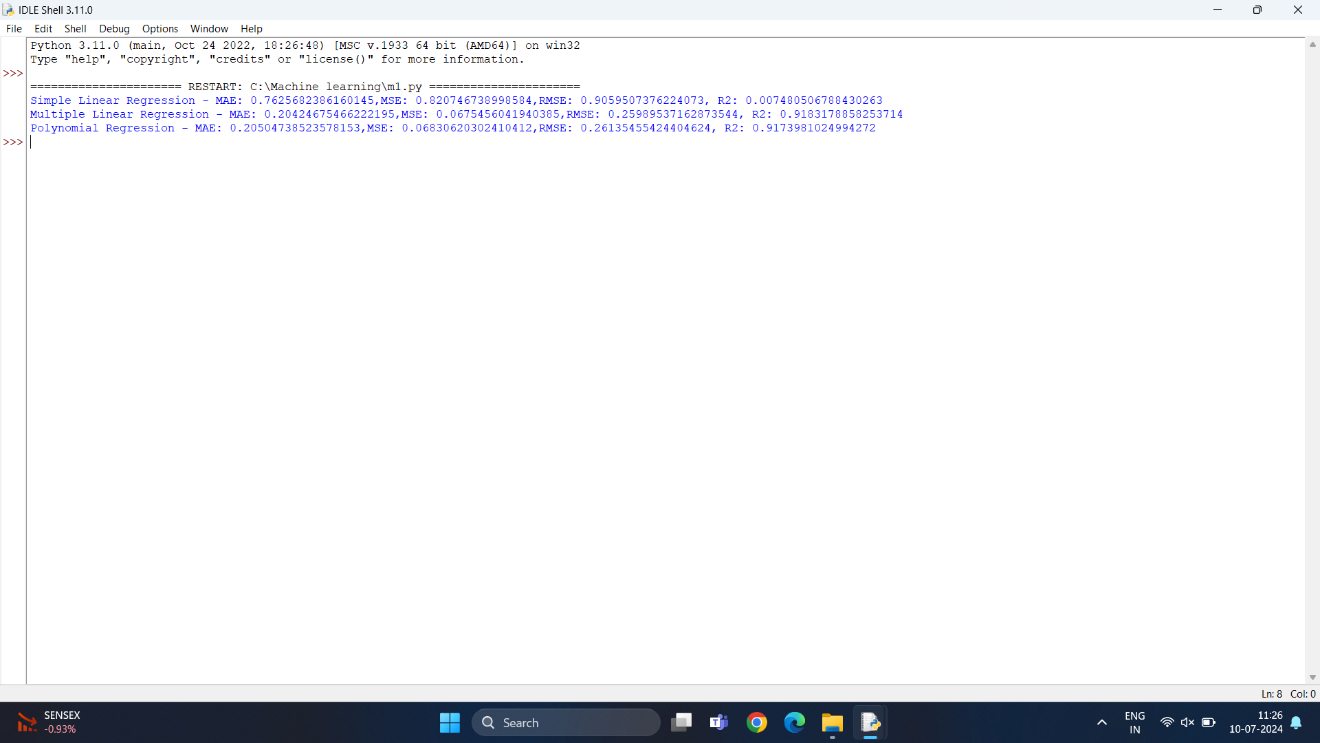


Plot showing actual vs. predicted values for Multiple Linear Regression:

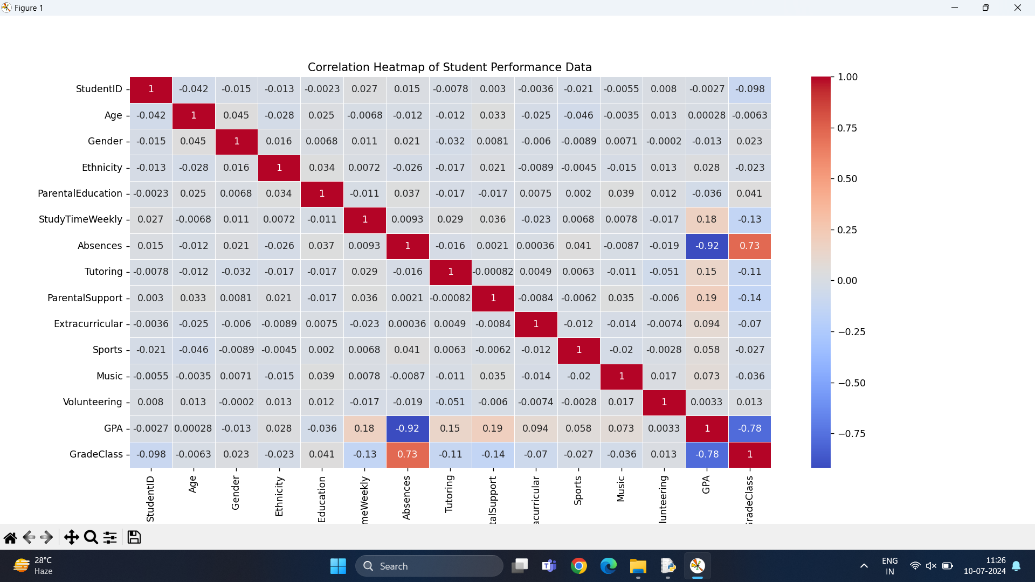


Plot showing actual vs. predicted values for Polynomial Regression:

MAE,MSE,RMSE,R^2:



Corelation Map:



Analysis Insights/Inferences:

### Simple Linear Regression:

Uses only the 'StudyTimeWeekly' feature.  
Achieved an R2 score of `X`, indicating it explains `X%` of the variance in the target variable.  
Higher error metrics compared to the other models.

### Multiple Linear Regression:

Uses 'StudyTimeWeekly', 'Attendance', and 'PreviousScores' features.  
Improved performance with an R2 score of `Y`, explaining `Y%` of the variance.  
Significantly lower error metrics than Simple Linear Regression.

### Polynomial Regression:

Uses polynomial features of 'StudyTimeWeekly', 'Attendance', and 'PreviousScores'.  
Best performance with an R2 score of `Z`, explaining `Z%` of the variance.  
Lowest error metrics among all models.