Student Performance Prediction

**Report By:**

**Kushagra srivastava**

**[202401100300143]**

**Course:**

**B. Tech – CSE-AI**

**Institution:**

**KIET Group of Institutions**



**Date:**

**11 March 2025**

# 1. Introduction

This project is a Python-based predictor of student’s performance. It allows users to check how good would a student will perform in the test.

# 2. Methodology

In this project, we predict **Final Exam Scores** of students based on **Study Hours** and **Previous Scores** using **Linear Regression**. Below is the summarized methodology:

1. **Data Creation**: A dataset containing **Study Hours**, **Previous Scores**, and **Final Exam Scores** was created and saved as a CSV file.
2. **Data Preprocessing**: The dataset was loaded and inspected. The features used for prediction are **Study Hours** and **Previous Scores**, with **Final Exam Score** as the target.
3. **Train-Test Split**: The data was split into 80% training and 20% testing to evaluate the model on unseen data.
4. **Model Training**: A **Linear Regression** model was trained using the training data, learning the relationship between the features and the target variable.
5. **Evaluation**: The model's performance was assessed using:
   * **Mean Absolute Error (MAE)**
   * **Mean Squared Error (MSE)**
   * **Root Mean Squared Error (RMSE)**
6. **Visualization**: Scatter plots and bar graphs were used to visualize actual vs. predicted scores and the relationship between the features and the target variable.

# 3. Code

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error

# Create and save dataset as CSV

csv\_filename = "student\_performance.csv"

data\_dict = {

    "StudentID": list(range(1, 21)),

    "StudyHours": [8.777482, 9.161915, 3.278010, 4.500247, 2.264931, 5.178765, 5.195977, 7.551904, 1.940879, 3.315756,

                    4.599099, 2.471536, 6.708061, 6.692461, 9.654504, 6.392792, 4.655716, 6.339880, 6.195849, 7.819022],

    "PreviousScores": [75, 55, 77, 60, 72, 87, 45, 68, 73, 78, 99, 68, 63, 44, 85, 86, 57, 69, 52, 90],

    "FinalExamScore": [64, 82, 70, 60, 60, 81, 85, 57, 65, 68, 81, 96, 85, 93, 52, 43, 99, 42, 76, 79]

}

data = pd.DataFrame(data\_dict)

data.to\_csv(csv\_filename, index=False)

print(f"CSV file '{csv\_filename}' has been created successfully.")

# Load dataset

data = pd.read\_csv(csv\_filename)

# Display first few rows

print("Dataset Preview:")

print(data.head())

# Define independent variables (features) and dependent variable (target)

X = data[['StudyHours', 'PreviousScores']]

y = data['FinalExamScore']

# Split data into training and testing sets (80% training, 20% testing)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Initialize and train the model

model = LinearRegression()

model.fit(X\_train, y\_train)

# Predict on test data

y\_pred = model.predict(X\_test)

# Evaluate the model

mae = mean\_absolute\_error(y\_test, y\_pred)

mse = mean\_squared\_error(y\_test, y\_pred)

rmse = np.sqrt(mse)

print("\nModel Performance:")

print(f"Mean Absolute Error (MAE): {mae:.2f}")

print(f"Mean Squared Error (MSE): {mse:.2f}")

print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")

# Plot actual vs predicted values

plt.scatter(y\_test, y\_pred, color='blue')

plt.xlabel("Actual Final Exam Score")

plt.ylabel("Predicted Final Exam Score")

plt.title("Actual vs Predicted Final Exam Scores")

plt.show()

# Display the model's coefficients and intercept

print("\nModel Coefficients:")

print(f"StudyHours Coefficient: {model.coef\_[0]:.2f}")

print(f"PreviousScores Coefficient: {model.coef\_[1]:.2f}")

print(f"Intercept: {model.intercept\_:.2f}")

# Additional Visualizations

plt.figure(figsize=(12, 6))

# Bar graph for Study Hours vs Final Exam Score

plt.subplot(1, 2, 1)

plt.bar(data['StudentID'], data['StudyHours'], color='skyblue', label='Study Hours')

plt.bar(data['StudentID'], data['FinalExamScore'], color='orange', alpha=0.7, label='Final Exam Score')

plt.xlabel("Student ID")

plt.ylabel("Values")

plt.title("Study Hours & Final Exam Score Comparison")

plt.legend()

# Line graph for Previous Scores vs Final Exam Score

plt.subplot(1, 2, 2)

plt.plot(data['StudentID'], data['PreviousScores'], marker='o', linestyle='-', color='red', label='Previous Scores')

plt.plot(data['StudentID'], data['FinalExamScore'], marker='s', linestyle='--', color='green', label='Final Exam Score')

plt.xlabel("Student ID")

plt.ylabel("Scores")

plt.title("Previous Scores vs Final Exam Score")

plt.legend()

plt.tight\_layout()

plt.show()

# 4. Screenshots of Output

Add screenshots of the program execution here.

