**Title**: Student Performance Prediction using Linear Regression

**Problem Statement**:  
In this project, we aim to predict the final exam scores of students based on two key factors: **Study Hours** and **Previous Scores**. Using a **Linear Regression** model, we analyze the relationship between these two features and the final exam score to build a predictive model. This project helps understand how well study habits and prior performance can predict final exam outcomes.

**Personal Details**:

* **Name**: Ankit kumar Gupta
* **Roll Number**: 202401100300048
* **Course**: AI
* **Instructor**: Bikki kumar
* **Date**: 11/03/2025

**Introduction:**

In the modern educational landscape, predicting students' performance plays an essential role in helping instructors, students, and institutions improve learning outcomes. By analyzing factors such as **Study Hours** and **Previous Scores**, educators can identify students who might need additional assistance or resources to perform better. In this project, we focus on predicting the final exam scores of students using these two features, with the goal of determining how well these factors can predict student performance. To achieve this, we use a **Linear Regression** model, which is one of the most straightforward and interpretable regression techniques.

The dataset consists of information on **Study Hours**, **Previous Scores**, and **Final Exam Scores**. Using this dataset, we will build a linear regression model to predict the **Final Exam Score** of students based on the given features. In the subsequent sections, we will explain the methodology followed to solve the problem, provide the code used for implementation, and present the results of the model's performance.

**Methodology:**

The methodology of this project revolves around the process of building a linear regression model to predict student performance based on **Study Hours** and **Previous Scores**.

First, the dataset was loaded and examined for missing values or inconsistencies. After ensuring the data was clean, we selected **Study Hours** and **Previous Scores** as the features (independent variables) and **Final Exam Score** as the target (dependent variable).

The data was then split into training and testing sets using the **train\_test\_split** function, where 80% of the data was used for training and 20% for testing. Next, we initialized a **Linear Regression** model, trained it using the training data, and made predictions on the test data.

To evaluate the performance of the model, we used two primary metrics: **Mean Squared Error (MSE)** and **R2 Score**. **MSE** measures the average squared difference between actual and predicted values, while **R2 Score** indicates how well the model explains the variance in the target variable.

Additionally, we visualized the results by plotting the actual versus predicted final exam scores. The plot allowed us to compare how closely the predicted values matched the actual values, which provided insight into the model's performance.

**Code:**

Here is the code used to implement the student performance prediction:

# Import necessary libraries

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

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

from sklearn.linear\_model import LinearRegression # Import LinearRegression

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

# Load the dataset

df = pd.read\_csv('/content/sample\_data/student\_data.csv') # Adjust the file path if needed

print(df.head()) # Display the first few rows of the dataset

# Define features (X) and target (y)

X = df[['StudyHours', 'PreviousScores']] # Features (StudyHours and PreviousScores)

y = df['FinalExamScore'] # Target (FinalExamScore)

# Split the dataset into training and testing sets (80% train, 20% test)

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

# Initialize the model

model = LinearRegression()

# Train the model on the training data

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

# Make predictions on the test set

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

# Evaluate the model's performance

mse = mean\_squared\_error(y\_test, y\_pred) # Mean Squared Error

r2 = r2\_score(y\_test, y\_pred) # R2 Score

# Print evaluation metrics

print(f'Mean Squared Error: {mse}')

print(f'R2 Score: {r2}')

# Plot the Actual vs Predicted Final Exam Scores

plt.scatter(y\_test, y\_pred)

plt.plot([min(y\_test), max(y\_test)], [min(y\_test), max(y\_test)], color='red', linewidth=2)

plt.xlabel('Actual Final Exam Scores')

plt.ylabel('Predicted Final Exam Scores')

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

plt.show()

**Output/Result:**

The model's performance was evaluated based on two primary metrics:

1. **Mean Squared Error (MSE)**: The MSE value is the average of the squared differences between the actual and predicted final exam scores. A lower MSE indicates better model performance.
2. **R2 Score**: The R2 score, also known as the coefficient of determination, measures the proportion of the variance in the dependent variable (Final Exam Score) that is predictable from the independent variables. A value closer to 1 indicates that the model explains a large portion of the variance.

The scatter plot below shows the relationship between the actual and predicted final exam scores. The closer the points are to the red line, the better the model's predictions.

**Example of printed output:**

Mean Squared Error: 182.85903450958284

R2 Score: 0.5240106722313811

**Scatter Plot (Actual vs Predicted Final Exam Scores):** The scatter plot visually represents how well the predicted values match the actual values. Points closer to the red line indicate accurate predictions, while points farther away suggest discrepancies.

**References/Credits:**

* **Dataset**: The dataset used for this project is hypothetical and created for educational purposes. In real-world applications, datasets like these can be found on platforms like Kaggle or the UCI Machine Learning Repository.
* **Libraries Used**:
  + **Pandas**: For data manipulation and handling missing values.
  + **NumPy**: For numerical operations and array manipulations.
  + **Matplotlib**: For creating the scatter plot to visualize predictions.
  + **Scikit-learn**: For implementing the Linear Regression model and evaluating its performance.

**Uploading Files to GitHub:**

To complete the project submission, upload the following files to a GitHub repository:

* The **Jupyter Notebook** (.ipynb file) containing the code.
* The **Report** in **PDF format** that includes the title page, introduction, methodology, code, output/results, and references.
* A **README file** that explains the project and how to run the code.

**Example of README file**:

# Student Performance Prediction

## Description

This project aims to predict student performance (Final Exam Scores) based on two features: Study Hours and Previous Scores. A Linear Regression model is used for prediction.

## Files:

- \*\*student\_performance\_prediction.ipynb\*\*: Jupyter notebook containing the code for training the model and evaluating the results.

- \*\*report.pdf\*\*: A detailed report of the project, including the problem statement, methodology, and results.

- \*\*README.md\*\*: This file.

## How to Run:

1. Clone the repository.

2. Install the necessary libraries:

```bash

pip install scikit-learn numpy pandas matplotlib

1. Open the student\_performance\_prediction.ipynb file and run the code to train the model and evaluate its performance.

**Credits:**

* Dataset: [Dataset Source] (if applicable)
* Libraries: Pandas, Numpy, Scikit-learn, Matplotlib

---

With this structure, you'll be able to create a well-organized report and easily upload your project files to GitHub. Let me know if you need further assistance!