Student Performance Prediction Report

Project Title: Student Performance Prediction
Name: Yatharth
Roll Number: 202401100300287
Course: INTRODUCTION TO AI
Date: 11/03/2025

Introduction

Student performance prediction is an essential task in the education sector. By analzing **study hours** and **previous scores**, we can estimate a student's **final exam performance**. This project aims to build a **Linear Regression model** to predict student scores based on these factors.

The dataset includes:

- **Study Hours**: Number of hours a student studied.
- **Previous Scores**: Scores from past exams.
- **Final Exam Score**: The actual score a student achieved in the final exam (Target variable).

This model helps educators and students by identifying those who may need **extra academic support** and enabling **early intervention strategies**.

Methodology

Step 1: Dataset Upload

The dataset is manually uploaded in **Google Collab** using:

python

CopyEdit

from google.colab import files

uploaded = files.upload()

Once uploaded, it is read using **pandas** into a DataFrame.

Step 2: Feature Selection

The dataset contains three key columns:

- StudyHours and PreviousScores → **Independent Variables (X)**
- FinalExamScore → **Dependent Variable (y)**

Step 3: Data Splitting

We divide the dataset into **training (80%)** and **testing (20%)** sets using train_test_split().

Step 4: Model Training

We use **Linear Regression**, a simple yet effective machine-learning model, to predict student performance. The model learns the relationship between **Study Hours, Previous Scores, and Final Exam Scores**.

Step 5: Model Prediction

After training, the model predicts the test data and generates predicted scores.

Step 6: Model Evaluation

The model is evaluated using the following metrics:

- **Mean Absolute Error (MAE)**
- **Mean Squared Error (MSE)**
- **Root Mean Squared Error (RMSE)**

These metrics help measure how accurately the model predicts student performance.

Code

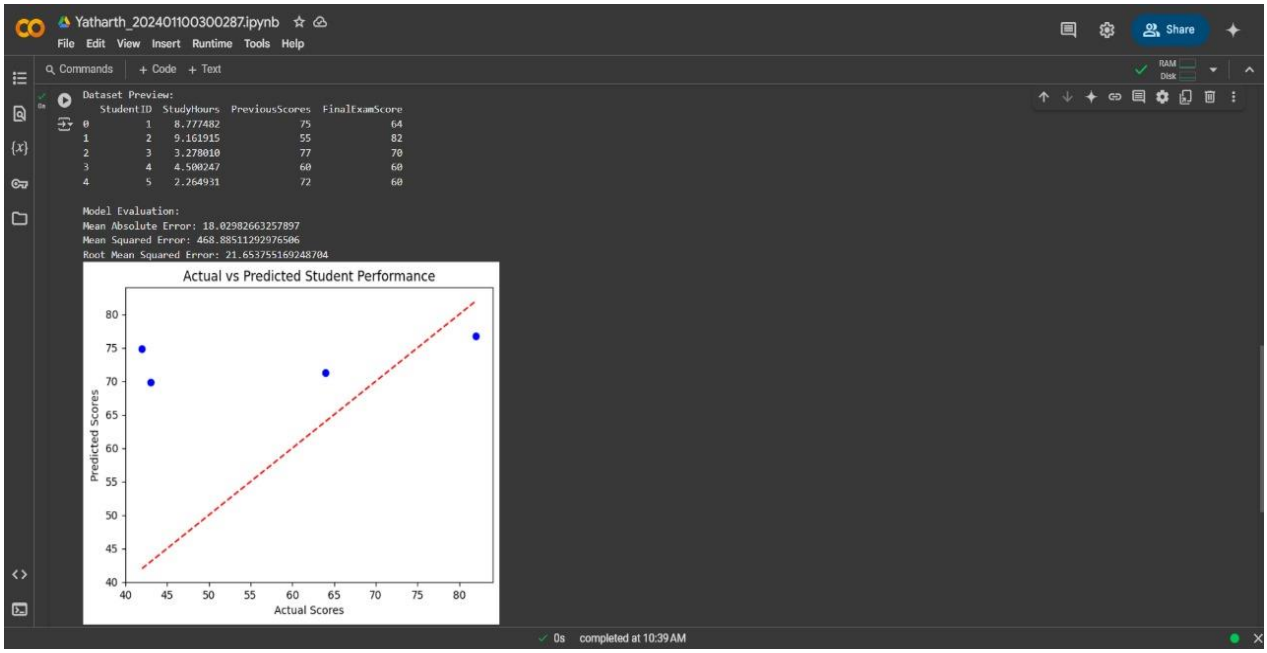
```
import pandas as pd  # Importing pandas for data handling

import numpy as np   # Importing numpy for numerical operations

import matplotlib.pyplot as plt  # Importing matplotlib for visualization

from sklearn.model_selection import train_test_split  # To split the dataset

from sklearn.linear_model import LinearRegression  # Importing Linear Regression model

from sklearn.metrics import mean_absolute_error, mean_squared_error  # For evaluating the model


# Step 1: Load the dataset (assuming it has already been uploaded)

filename = list(uploaded.keys())[0]  # Get the uploaded filename

df = pd.read_csv(filename)  # Read the dataset into a DataFrame

print("Dataset Preview:")

print(df.head())


# Step 2: Define independent (X) and dependent (y) variables

# Adjusting column names based on uploaded dataset

X = df[["StudyHours", "PreviousScores"]]  # Features
```

```
y = df["FinalExamScore"]  # Target variable

# Step 3: Split the dataset 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)

# Step 4: Train a Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)  # Training the model

# Step 5: Predict on the test set
y_pred = model.predict(X_test)

# Step 6: Evaluate the model
mae = mean_absolute_error(y_test, y_pred)  # Mean Absolute Error
mse = mean_squared_error(y_test, y_pred)  # Mean Squared Error
rmse = np.sqrt(mse)  # Root Mean Squared Error

print("\nModel Evaluation:")
print(f"Mean Absolute Error: {mae}")
print(f"Mean Squared Error: {mse}")
print(f"Root Mean Squared Error: {rmse}")
```

```
# Step 7: Visualizing the predictions vs actual values
plt.scatter(y_test, y_pred, color='blue')  # Scatter plot for actual vs predicted
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red', linestyle='dashed')
plt.xlabel("Actual Scores")
plt.ylabel("Predicted Scores")
plt.title("Actual vs Predicted Student Performance")
plt.show()
```

Output/Results

Below is a screenshot of the **Actual vs Predicted Scores** scatter plot generated by the model:



References/Credits

- **Dataset Source**: The dataset used for training is **manually uploaded** in Google Collab.
- **Libraries Used**:
  - pandas - For data handling
  - numpy - For numerical computations
  - matplotlib - For visualization
  - scikit-learn - For machine learning model training and evaluation