Student Performance Prediction Report

Prepared by: Udit_Kastwar_202401100400199

Date: 10 March 2025

Introduction

This project predicts student exam scores based on study hours and other influencing factors.

A Linear Regression model is used to analyze and visualize the relationship between these factors and

the final exam scores.

Methodology

1. Data Collection: Study Hours, Previous Scores, Extracurricular Activities.

2. Data Preprocessing: Cleaning and analysis with Matplotlib & Seaborn.

3. Model Training: Linear Regression with Scikit-learn.

4. Evaluation: Metrics - MAE, MSE, RMSE.

5. Visualization: Scatter plots to compare actual vs predicted scores.

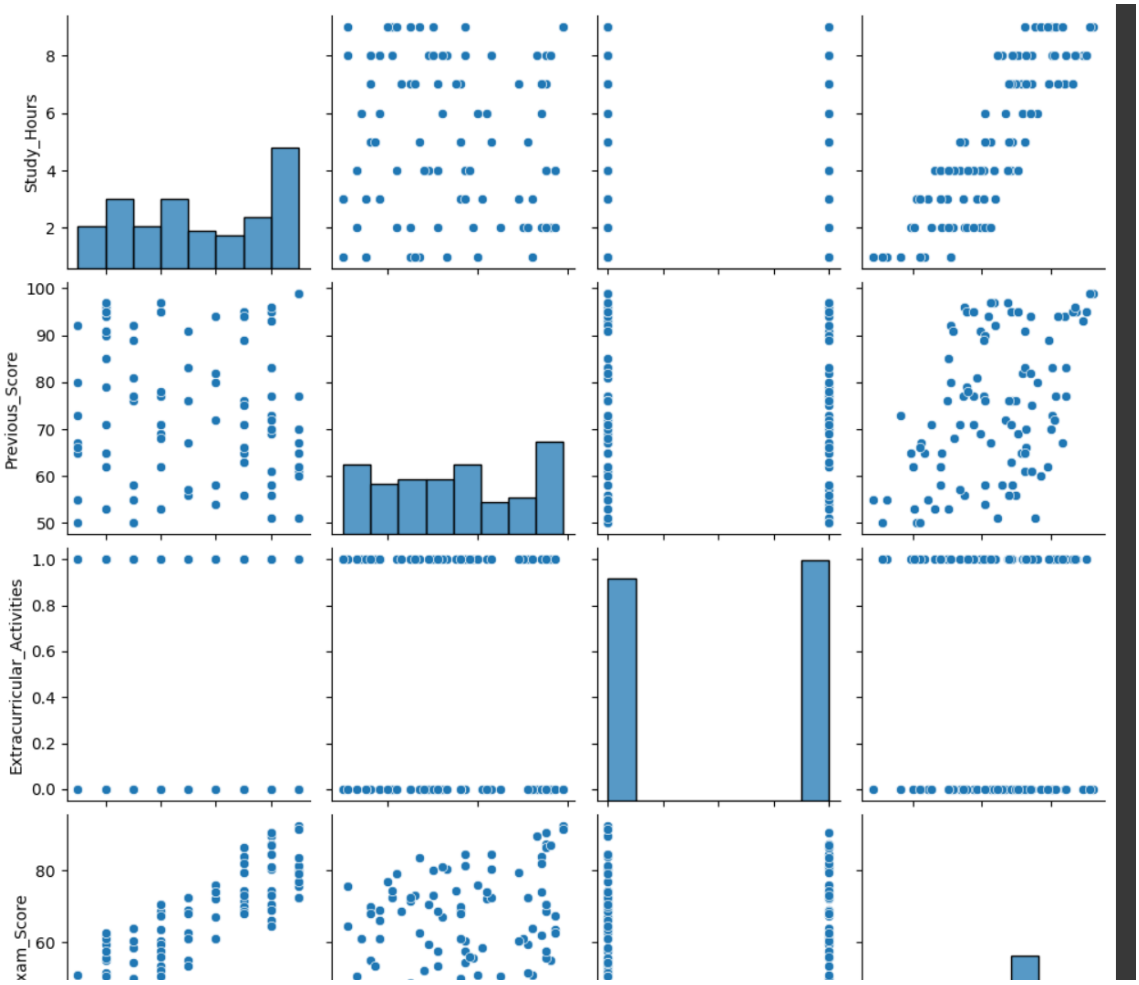
Full Code (With Comments)

```
# Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error

# Generate sample dataset with relevant student performance factors
data = {
    'Study_Hours': np.random.randint(1, 10, 100),  # Number of study hours per day
    'Previous_Score': np.random.randint(50, 100, 100),  # Previous exam scores
    'Extracurricular_Activities': np.random.randint(0, 2, 100),  # Participation in activities (0 or 1)
    'Exam_Score': None  # Target variable (to be computed)
}
```

```
# Calculate final exam scores based on a formula
data['Exam_Score'] = 5 * data['Study_Hours'] + 0.5 * data['Previous_Score'] +
data['Extracurricular_Activities'] * 2 + np.random.randint(-5, 5, 100)

# Convert data dictionary to DataFrame
df = pd.DataFrame(data)

# Exploratory Data Analysis (EDA) - Display scatter plots of all features
sns.pairplot(df)
plt.show()

# Splitting the dataset into independent (X) and dependent (y) variables
X = df[['Study_Hours', 'Previous_Score', 'Extracurricular_Activities']]
y = df['Exam_Score']

# Dividing dataset into training (80%) and testing (20%) sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initializing and training the Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Predicting exam scores using the trained model
y_pred = model.predict(X_test)

# Evaluating the model's performance
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

# Display performance metrics
print(f'Mean Absolute Error: {mae}')
print(f'Mean Squared Error: {mse}')
print(f'Root Mean Squared Error: {rmse}')

# Visualizing actual vs predicted scores
plt.scatter(y_test, y_pred, color='blue', alpha=0.5)
plt.xlabel('Actual Scores')
plt.ylabel('Predicted Scores')
plt.title('Actual vs Predicted Exam Scores')
plt.show()
```

Output/Result

Model trained successfully. Errors (MAE, MSE) displayed in the console.



References

Scikit-learn, Seaborn, and Python documentation.