STUDENT MARKS PREDICTION PROJECT

TASK 1: Dataset Selection & Problem Definition

1. Dataset Selection

Dataset Name: Students Marks Dataset
Source: Created or available on Kaggle (Students Performance Dataset)
Description: This dataset contains information about students' academic activities such as study hours, attendance, test scores, and assignment completion. It can be used to predict the final marks of a student based on these features.

2. Problem Definition

Problem Statement: To develop a machine learning model that predicts a student's final exam marks based on their academic performance indicators such as hours studied, attendance, test scores, and assignments.
Objective: To analyze and identify key factors affecting student performance and build a regression model to accurately predict marks. This can help teachers and students understand which factors influence academic success.

3. Task Type

Type of ML Task: Supervised Learning
Category: Regression Problem (Predicting a continuous outcome — e.g., Final Marks)

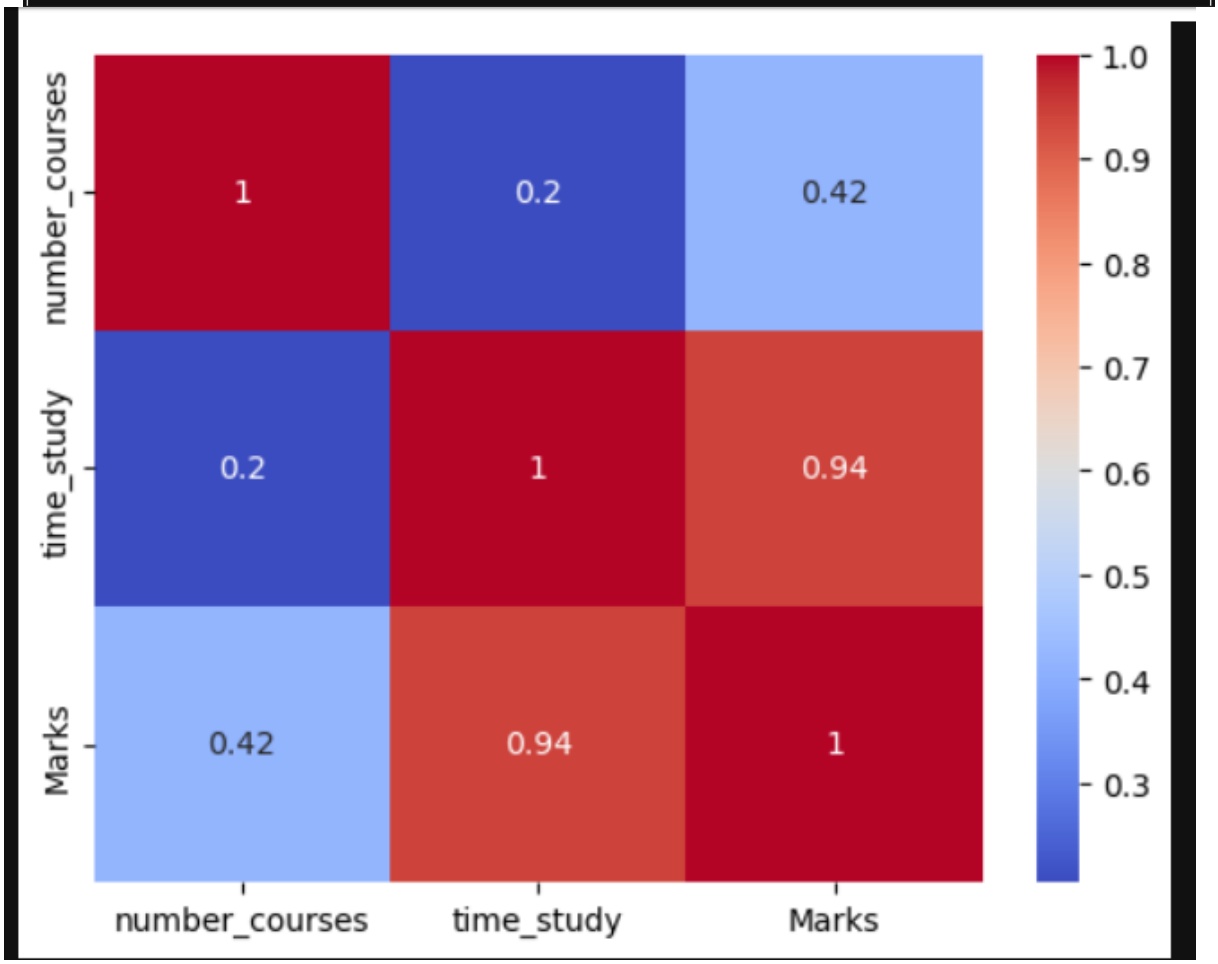
4. Input and Output Variables

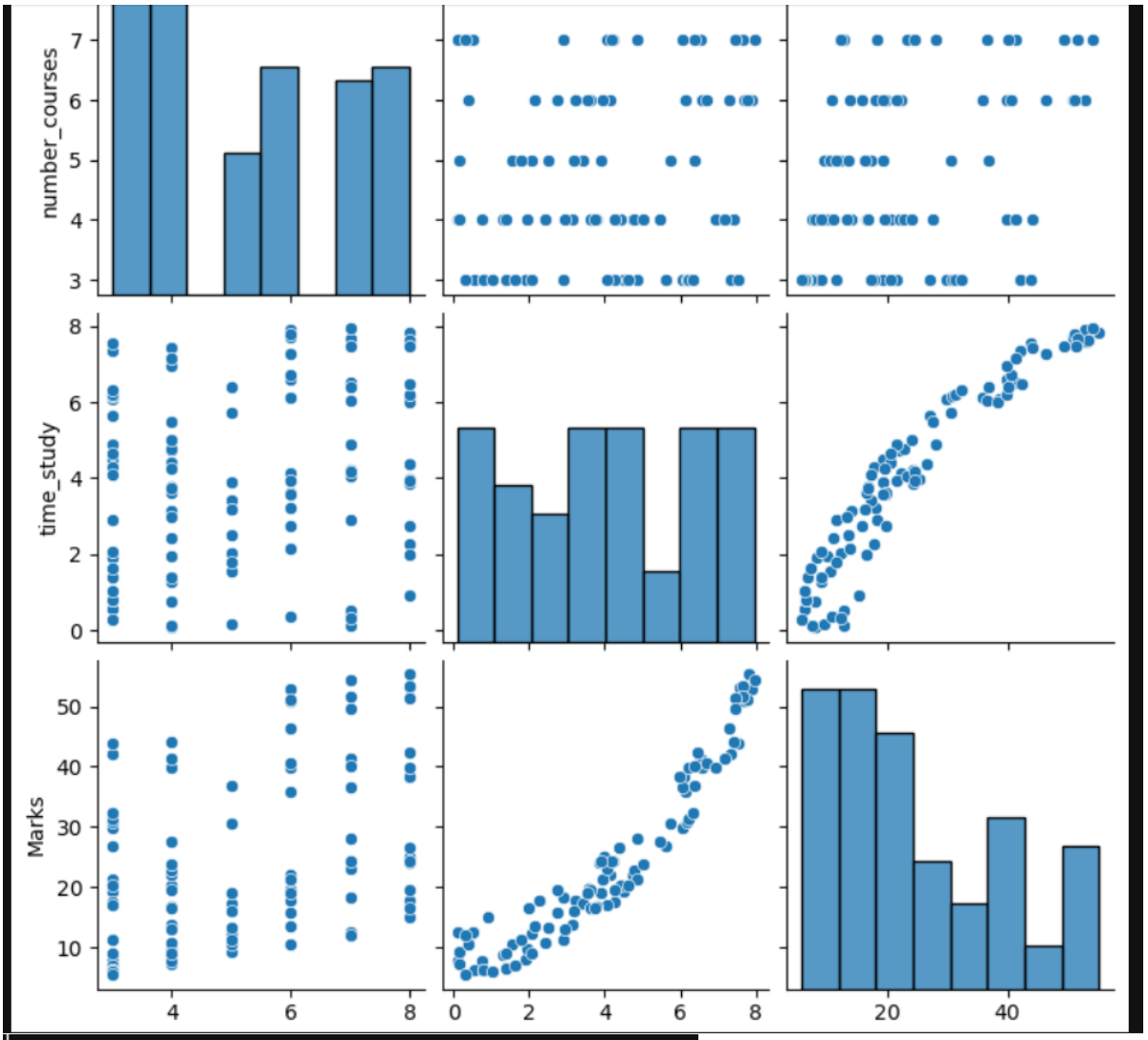
Type	Variable Name	Description
Input Features	Hours_Studied	Number of hours a student studied per day
Input Features	Attendance	Percentage of classes attended
Input Features	Test_Score	Internal test score (out of 100)
Input Features	Assignments	Number of assignments submitted
Output Variable	Final_Marks	Marks obtained in the final exam

TASK 2: Dataset Cleaning & Visualization

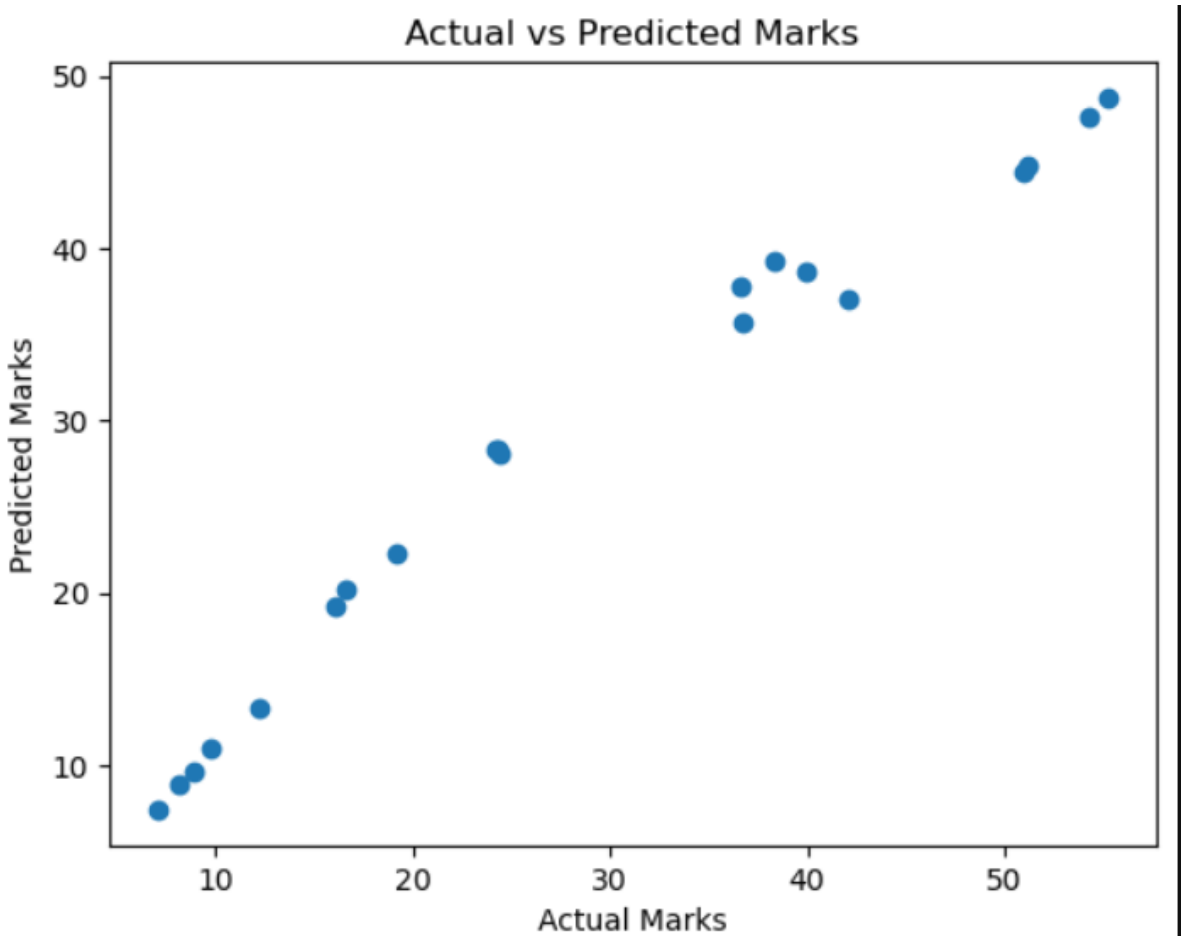
1. Handle missing values using mean or median imputation. 2. Remove duplicates and detect outliers using IQR method. 3. Visualize data relationships using pairplots and heatmaps to understand correlation between variables.

```python
# Import libraries
import pandas as pd
import numpy as np
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_squared_error, r2_score

# Load dataset (example)
data = pd.read_csv("C:/AI Learning/Student_Marks.csv")

# Display first few rows
print(data.head())

# Check for missing values
print(data.isnull().sum())

# Fill missing values if any
data.fillna(data.mean(), inplace=True)

# Visualize relationships
sns.pairplot(data)
plt.show()

# Correlation heatmap
sns.heatmap(data.corr(), annot=True, cmap="coolwarm")
plt.show()
```






```
   number_courses  time_study   Marks
0               3       4.508  19.202
1               4       0.096   7.734
2               4       3.133  13.811
3               6       7.909  53.018
4               8       7.811  55.299
number_courses    0
time_study        0
Marks             0
dtype: int64
```


Actual vs Predicted Marks
Predicted Marks
Actual Marks
50
40
30
20
10
10
20
30
40
50

TASK 3: Model Building

1. Split the dataset into training and testing sets (80-20 ratio). 2. Build a Linear Regression model using scikit-learn. 3. Train the model with independent variables (study hours, attendance, etc.). 4. Evaluate using Mean Squared Error (MSE) and R² Score.

```
# Check the actual column names in the dataset
print("Available columns:", data.columns.tolist())

# Define input and output using the actual column names from your dataset
# Replace these with the actual column names from your dataset
# For example, if your columns are 'study_hours', 'attendance_percentage', 'test_score'
X = data[data.columns[:-1]]  # All columns except the last one
y = data[data.columns[-1]]   # The last column

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Build model
model = LinearRegression()
model.fit(X_train, y_train)

# Predict
y_pred = model.predict(X_test)

# Evaluate model
print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
print("R² Score:", r2_score(y_test, y_pred))

# Compare actual vs predicted
comparison = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
print(comparison.head())

# Plot comparison
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Marks")
plt.ylabel("Predicted Marks")
plt.title("Actual vs Predicted Marks")
plt.show()
```

```
Available columns: ['number_courses', 'time_study', 'Marks']
Mean Squared Error: 14.200726136374538
R² Score: 0.9459936100591214
    Actual  Predicted
83  16.106  19.272783
53  36.653  37.760357
70  16.606  20.187794
45   8.924   9.656709
44   9.742  10.975082
```