Student Academic Performance Predictor

Objective: To assess your understanding of integrating machine learning models with web applications, handling different datasets, and deploying practical prediction tools.

Part 1: Scenario and Dataset Understanding (20 points)

You are tasked with building a web application that predicts a student's academic performance (e.g., 'Good', 'Average', 'Poor') based on certain input features.

Dataset: Create dataset for student performance. Let's call it student_data.csv.

student_data.csv (Example Structure):

StudyHours,Attendance,PreviousGrade,Performance

2,70,C,Poor

5,85,B,Average

8,95,A,Good

3,75,C,Poor

6,90,B,Good

1,60,D,Poor

4,80,C,Average

7,92,A,Good

Questions:

1. **Data Features & Target:**
    - Identify the input features (independent variables) and the target variable (dependent variable) in the student_data.csv dataset.
    - Explain why each chosen feature might be relevant for predicting academic performance.

2. **Data Types & Preprocessing:**
    - What are the data types of each feature in student_data.csv?
    - If 'PreviousGrade' were represented as 'A', 'B', 'C', 'D', 'F' instead of numerical values, what preprocessing steps would be necessary before feeding it to a machine learning model? Justify your answer.
3. **Model Selection:**
    - Given the Performance column as the target, what type of machine learning problem is this (e.g., classification, regression)?
    - Suggest at least two suitable machine learning algorithms that could be used to build a model for this prediction task. Briefly explain why each is appropriate.

Part 2: Model Training and Serialization (30 points)

Task: Assume you have trained a machine learning model using the student_data.csv dataset and saved it as model_performance.pkl.

Questions:

1. **Model Training Process (Conceptual):**
    - Outline the general steps you would take to train a machine learning model using a dataset like student_data.csv in Python (e.g., using scikit-learn).
    - What is the purpose of "training" a model?
2. **Pickle File:**
    - Explain the role of pickle in the provided app.py example (model1 = pickle.load(open('E:/movie_interest_webapp/data/model.pkl', 'rb'))).
    - Why is it beneficial to save a trained model using pickle for a web application?

3. **Input Data Format for Prediction:**
    - If your model_performance.pkl expects input in the format [[StudyHours, Attendance, PreviousGrade]], how would you modify the predict function in app.py to prepare the user's input for a single student?
    - Consider an input form that takes study_hours, attendance_percentage, and previous_grade_numeric from the user.

Part 3: Web Application Development

Task: Modify the provided app.py, index.html, and result.html files to create the "Student Academic Performance Predictor".

Requirements:

1. **app.py Modifications:**
    - Change the model loading path and variable name appropriately (e.g., model_performance).
    - Update the predict function to accept three input features (study_hours, attendance_percentage, previous_grade_numeric) from the HTML form.
    - Ensure the input data is correctly formatted for prediction by your model_performance (e.g., [[val1, val2, val3]] for a single student prediction).
    - Modify the render_template call in predict to pass the predicted performance to result.html.
2. **index.html Modifications:**
    - Change the title to "Student Academic Performance Predictor".
    - Update the form to include three input fields: "Study Hours", "Attendance Percentage", and "Previous Grade (Numeric e.g., 1-100 or GPA)".

    - Ensure the name attributes for the input fields match what app.py expects (e.g., name="study_hours").
    - The form should submit to the /predict route.
3. **result.html Modifications:**
    - Change the title to "Academic Performance Prediction Result".
    - Display the predicted performance for the student (e.g., "Predicted Performance: {{ prediction }}").
    - Keep the "Try Again" link.

Code Snippets to Provide (as answers):

- Modified app.py (relevant sections only, or full file if easier).
- Modified index.html (full file).
- Modified result.html (full file).

Part 4: Deployment and Real-World Considerations

Questions:

1. **Error Handling & Validation:**
    - What kind of input validation would be crucial to add to the app.py to make the application more robust (e.g., handling non-numeric input, out-of-range values)?
    - How would you inform the user about invalid input?
2. **Scalability & Deployment:**
    - If this application were to be used by thousands of students simultaneously, what are some performance considerations you would need to address?
    - Briefly describe a typical deployment strategy for a Flask web application in a production environment (e.g., using Gunicorn/Waitress and Nginx/Apache).

3. **Ethical Considerations & Bias:**
    - Discuss potential biases that could exist in a student_data.csv dataset and how these biases might lead to unfair or inaccurate predictions.
    - What are the ethical implications of using an AI model to predict student performance?