Student Performance Prediction

1. Project Overview

The project aims to predict students' math scores based on various features such as gender, race/ethnicity, parental level of education, lunch, test preparation course, reading score, and writing score. This is achieved using a machine learning model integrated into a Flask web application for user interaction.

2. System Architecture

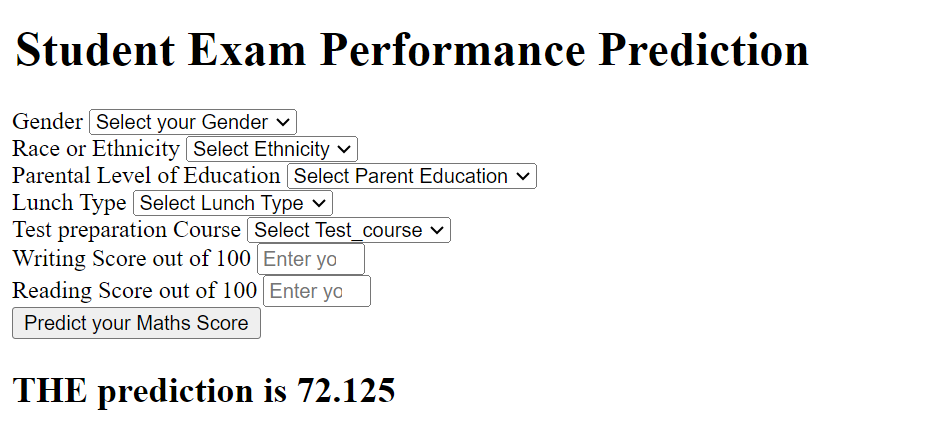
* Architecture Type: Client-Server
* Technologies and Frameworks:
  + Backend: Flask (for serving the machine learning model and handling HTTP requests)
  + Frontend: HTML (handled by Flask templates)
  + Development Environment: VS Code (for modular development)
  + Version Control: GitHub (for version control and portfolio)

3. Key Components

1. Artifact Folder:
   * Purpose: Contains data files and artifacts related to the model training and testing phases.
2. Notebook Folder
   * Purpose: Contains Jupyter notebooks for exploratory data analysis (EDA) and CSV file handling.
3. Template Folder
   * Purpose: Contains HTML templates for the web application's user interface.
4. Source (src) Folder
   * Purpose: Contains the core functionality of the application.
   * Components:
     + data\_ingestion.py: Reads and splits data into training and testing sets, stores them in the artifact folder.
     + data\_transformation.py: Transforms data (e.g., encoding categorical features, scaling numerical features).
     + model\_trainer.py: Performs model training, hyperparameter tuning, and returns performance metrics like R² score.
     + pipeline/: Contains utility modules:
       - exception.py: Handles exceptions.
       - logger.py: Logs events and errors.
       - utils.py: Provides helper functions.
5. Log Folder
   * Purpose: Stores logs generated during the execution of the application.
6. Virtual Environment (venv) Folder
   * Purpose: Contains a separate Python environment with specific packages required for the project.
7. Supporting Files
   * .gitignore: Specifies files and directories to be ignored by Git.
   * requirements.txt: Lists the Python packages required for the project.
   * setup.py: Provides metadata and configuration for the project.

4. Data Flow

1. User Input:
   * Users input their data through the web application's forms.
2. Data Processing:
   * User inputs are sent to the Flask backend.
3. Prediction:
   * The Flask application processes the input data using the prediction model.
4. Output:
   * The predicted math score is returned to the user and displayed on the web interface.



5. External Interfaces

* Web Application Interface: Collects user inputs via HTML forms and displays prediction results.
* Machine Learning Model: The core of the prediction functionality, accessed via Flask.

6. Security Considerations

* Data Validation: Ensure that input data is validated and sanitized to prevent security issues.
* Error Handling: Implement robust exception handling and logging to track and manage errors.

7. Performance Requirements

* Response Time: Aim for minimal response time to ensure a smooth user experience.
* Scalability: While currently deployed locally, the project is designed with cloud deployment in mind to handle scalability and performance requirements.

8. Deployment Strategy

* Local Deployment: Currently deployed locally due to budget constraints.
* Cloud Deployment: Outlined strategy for cloud deployment, including considerations for scaling, security, and maintenance.

9. Tools and Technologies

* Development: VS Code
* Web Application: Flask
* Version Control: GitHub
* Language: Python