**Project Title: Predicting Student Academic Performance**

**Project Overview**

This project aims to predict student performance (e.g., final grades) using the UCI Student Performance dataset. The dataset is provided in two files—one for Math and one for Portuguese language classes—which you can merge to gain richer insights into each student’s profile. The project covers the complete machine learning pipeline, from data cleaning and feature engineering to model training and evaluation.

**Data Sources**

* **Files:**
  + student-mat.csv (Math performance data)
  + student-por.csv (Portuguese performance data)
* **Description:**  
  Each file includes demographic, social, and academic attributes (such as study time, family background, and previous grades). Merging these files based on common attributes (e.g., school, sex, age) allows you to combine insights from both subjects and create a more comprehensive dataset.
* **Dataset Link:**  
  The UCI Student Performance dataset

**Project Workflow**

**1. Data Collection and Integration**

* **Load the Data:**
  + Import both CSV files into your working environment (e.g., using Pandas in Python).
* **Merge Datasets:**
  + Identify common columns (such as school, sex, age, etc.) and merge the Math and Portuguese datasets. Decide whether to combine records (using an inner join) or maintain separate attributes for each subject.
* **Initial Exploration:**
  + Get a sense of the dataset structure, missing values, and basic distributions.

**2. Data Preprocessing**

* **Missing Value Treatment:**
  + Identify missing data in both datasets.
  + Impute missing values using appropriate strategies (e.g., mean/mode imputation for numerical features, constant or new category for categorical features).
* **Data Cleaning:**
  + Standardize column names and data types.
  + Remove or correct any inconsistencies or erroneous entries.
* **Outlier Detection:**
  + Use visualizations (like box plots) and statistical methods to detect and decide how to handle outliers.

**3. Feature Engineering**

* **Combining Features:**
  + Create aggregate features such as the average grade across subjects or the difference between Math and Portuguese scores.
* **Encoding Categorical Variables:**
  + Convert categorical features (e.g., sex, school type, and family support) into numerical format using One-Hot Encoding or Label Encoding.
* **Temporal and Interaction Features:**
  + Derive interaction features, such as the relationship between study time and previous grades, to capture non-linear effects.
* **Scaling/Normalization:**
  + Standardize numerical features to ensure consistency for algorithms sensitive to feature scale.

**4. Model Creation**

* **Problem Definition:**
  + Choose a prediction task: you can either formulate it as a regression problem (predicting final grades) or a classification problem (predicting pass/fail status).
* **Baseline Models:**
  + Start with simple models like Linear Regression (for regression) or Logistic Regression (for classification) as baselines.
* **Advanced Models:**
  + Experiment with tree-based models (e.g., Random Forest, Gradient Boosting) and even ensemble methods to improve performance.
* **Pipeline Construction:**
  + Use Scikit-Learn pipelines to streamline preprocessing, feature engineering, and model training steps.

**5. Performance Evaluation**

* **Data Splitting:**
  + Split the merged dataset into training and testing subsets (e.g., an 80/20 split) or use cross-validation.
* **Metrics:**
  + For regression: evaluate using Mean Squared Error (MSE), Mean Absolute Error (MAE), and R² score.
  + For classification: assess performance using accuracy, precision, recall, F1-score, and ROC-AUC.
* **Error Analysis:**
  + Visualize residuals (for regression) or the confusion matrix (for classification) to analyze errors and identify areas for improvement.
* **Hyperparameter Tuning:**
  + Use techniques like grid search or random search to fine-tune model parameters.