AutoJudge: Predicting Programming Problem Difficulty

Online coding platforms (like Codeforces, Kattis, CodeChef) classify programming problems as **Easy, Medium, or Hard**, and also assign a **difficulty score**.
However, this process usually depends on human judgment and user feedback.

In this project, your task is to **build an intelligent system that can automatically predict**:

1. **Problem Class** → `Easy / Medium / Hard` *(classification task)*

2. **Problem Score** → a numerical difficulty score *(regression task)*

The prediction should be based **only on the textual description of the problem**, such as:

- Problem description
- Input description
- Output description

You will use the **provided dataset only**, which already contains problems with difficulty labels and scores.

Finally, you will build a **simple web interface** where a user can paste a new problem description and get:

- Predicted difficulty class
- Predicted difficulty score

Dataset Description → Link **(this dataset is just for reference you can use some other as well)**

Each data sample contains:

- `title`
- `description`
- `input_description`
- `output_description`
- `problem_class` → *(Easy / Medium / Hard)*
- `problem_score` → *(numerical value)*

You **do not need to create or label the dataset yourself**.

Project Objectives

By the end of this project, your system should:

★ Predict **problem difficulty class**
★ Predict **problem difficulty score**
★ Work using **only textual information**
★ Provide results through a **simple web UI**

Expected Deliverables

1. **Data Preprocessing**
    - Clean and combine text fields
    - Handle missing values if any

2. **Feature Extraction**

    - Convert text into numerical features

3. **Model 1: Classification**
    - Predict `problem_class` (Easy / Medium / Hard)

4. **Model 2: Regression**
    - Predict `problem_score`

5. **Evaluation**
    - Classification: accuracy / confusion matrix
    - Regression: MAE / RMSE

6. **Web Interface**
    - Text boxes for:
        - Problem description
        - Input description
        - Output description
    - Display predicted class and score

7. **Documentation**
    - Clear README explaining approach and results

Hints (To Help You Get Started)

🔹 **Feature Engineering (Important!)**

- Combine all text fields into **one single text input**
- Useful features:
    - Text length
    - Number of mathematical symbols

    - Keyword frequency (e.g., graph, dp, recursion)
    - TF-IDF vectors

🔹 **Models to Try (Simple & Effective)**

- **Classification**
    - Logistic Regression
    - Random Forest
    - Support Vector Machine (SVM)

- **Regression**
    - Linear Regression
    - Random Forest Regressor
    - Gradient Boosting

*You are **not required** to use deep learning.*

🌐 Web UI Expectations (Simple)

The UI should allow a user to:

1. Paste:
    - Problem description
    - Input description
    - Output description

2. Click **Predict**

3. See:
    - Predicted difficulty class
    - Predicted difficulty score

No authentication or database required.