

Project Report: Auto Judge

1. Problem Statement

This project is designed as a tool to assist competitive programmers and Data Structures and Algorithms (DSA) solvers in analyzing the difficulty of a problem *before* attempting it. By providing a problem's description, the tool predicts two key metrics:

- **Classification:** Whether the question is **HARD**, **MEDIUM**, or **EASY**.
- **Regression:** The problem's precise difficulty score on a scale out of **10**.

2. Dataset Used

The model was trained using the following external dataset:

- **Dataset Source:** [TaskComplexityEval-24](#)

3. Data Preprocessing and Feature Engineering

Certain key observations were made from the initial analysis of data that certain range of difficulty score was classified into a particular class for eg

	min	max	range
problem_class			
hard	5.5	9.7	4.2
medium	2.8	5.5	2.7
easy	1.1	2.8	1.7

The initial approach involved consolidating all textual information (Problem Description, Input Description, Output Description, etc.) into a single column for processing. Data Preprocessing

- The raw textual information was aggregated into a unified text column.
- Textual data was cleaned removing unnecessary punctuation and doing lemmatization.
- The textual data was then converted into numerical features using **TF-IDF (Term Frequency-Inverse Document Frequency)** to create embeddings that the Machine Learning models could understand.

Feature Engineering

The following features were engineered from the problem text to capture key aspects of problem complexity:

- Length of the problem description.

- Number of words used in the problem description.
- The presence and count of mathematical symbols used.

Notably topics like graph, dp and recursion were given a count to signify its importance in difficulty but the count was too low and sparse was not viable for taking it in account.

4. Models Used for Classification and Regression

Two separate machine learning models were developed to address the distinct prediction tasks (Classification and Regression).

The models used in this project include:

Task	Models Tested	Saved Best Model
Classification (EASY, MEDIUM, HARD)	RandomForest Classifier and XGBoost Classifier	RandomForestClassifier
Regression (Difficulty Score out of 10)	RandomForest Regressor and XGBoost Regressor	RandomForestRegressor

The idea to create only a single model for regression and mapping its result based on training data for classification led to poor results, so the idea had to be discarded. This could be because of rigid boundary change in the class

```
--- Classification Report (Derived from Regression Output) ---
Accuracy: 0.4520
```

Detailed Report:

	precision	recall	f1-score	support
easy	0.67	0.04	0.07	153
hard	0.64	0.42	0.51	389
medium	0.36	0.72	0.48	281
accuracy			0.45	823
macro avg	0.56	0.39	0.35	823
weighted avg	0.55	0.45	0.42	823

Separate results of RandomForest and XgBoost without any hyperparameter tuning.

RandomForest

```
Regressor R2 Score: 0.1548
Regressor RMSE: 4.1017
```

Classifier Accuracy: 0.5140

Classification Report:

	precision	recall	f1-score	support
easy	0.56	0.29	0.38	153
hard	0.54	0.80	0.64	389
medium	0.41	0.25	0.31	281
accuracy			0.51	823
macro avg	0.50	0.44	0.44	823
weighted avg	0.50	0.51	0.48	823

XGBoost

XGBoost Regressor R2: 0.1565

XGBoost Regressor RMSE: 4.0934

XGBoost Classifier Accuracy: 0.5091

	precision	recall	f1-score	support
easy	0.49	0.31	0.38	153
hard	0.56	0.76	0.65	389
medium	0.38	0.27	0.31	281
accuracy			0.51	823
macro avg	0.48	0.45	0.45	823
weighted avg	0.49	0.51	0.48	823

5. Experimental Setup

The core experimental process focused on **hyperparameter tuning** to select the optimal model for both the classification and regression tasks. The models were evaluated using the respective metrics for each task, and the best-performing model was then saved for deployment.

6. Results and Evaluation Metrics

The final model selection was based on the performance of each model on the test data. Regression Metrics (R^2)

Model	R ² Score
RandomForest Regressor	0.1321
XGBoost Regressor	0.1179
Saved Best Regressor	RandomForestRegressor

Classification Metrics (Accuracy)

Model	Accuracy
RandomForest Classifier	0.5081
XGBoost Classifier	0.5056
Saved Best Classifier	RandomForestClassifier

7. Web Interface and Sample Predictions

Web Interface Description

The tool's user interface is built using **Streamlit** and is available for local deployment or via the hosted application link. The interface prompts the user to submit detailed inputs for the problem they wish to analyze, specifically:

- **Problem Description**
- **Input Description**
- **Output Description**
- **Sample Input/Output**

Upon submitting the problem details, the system processes the inputs through the saved best models (**RandomForestClassifier** and **RandomForestRegressor**) and displays the predicted results to the user.

Problem Difficulty Analyzer

Problem Description

But yesterday, he came to see "her" in the real world and found out "she" is actually a very strong man! Our hero is very sad and he is too tired to love again now. So he came up with a way to recognize users' genders by their user names.

This is his method: if the number of distinct characters in one's user name is odd, then he is a male, otherwise she is a female. You are given the string that denotes the user name, please help our hero to determine the gender of this user by his method.

Input Description

that contains only lowercase English letters — the user name. This string contains at most 100 letters.

Output Description

"CHAT WITH HER!" (without the quotes), otherwise, print "IGNORE HIM!" (without the quotes)

Sample Input/Output

sevenkplus
OutputCopy
CHAT WITH HER!

Analyze Difficulty

Complexity Score

4.42

Category: **EASY**

This project successfully implements a dual-approach machine learning system for competitive programming problem difficulty analysis, offering both a categorical classification and a granular score. The **RandomForest** algorithm was selected as the best performer for both the classification and regression tasks based on the evaluation metrics. The final application provides a user-friendly web interface via Streamlit, allowing competitive programmers to quickly analyze the complexity of a problem before beginning their solution.