

# AutoJudge: Programming Problem Difficulty Prediction

## Project Report:

Name: Devansh Pandey

Enrollment Number: 23113049

Branch: Civil Engineering, 3rd Year

*This report presents the design and implementation of an automated system to predict the difficulty level of competitive programming problems using machine learning techniques.*

# 1. Introduction and Problem Statement

Competitive programming platforms host thousands of problems with varying difficulty levels. Manually assigning difficulty labels is subjective and time-consuming. This project aims to automate difficulty prediction using machine learning by analyzing problem statements and associated metadata.

# 2. Dataset Description

The dataset used in this project is provided in JSON Lines (JSONL) format and consists of programming problems scraped from online judges. Each record includes the problem title, description, input/output specifications, sample input-output examples, difficulty class (Easy, Medium, Hard), and a numerical difficulty score.

# 3. Data Preprocessing

Textual fields are cleaned and combined for feature extraction. Numerical features are scaled using standard normalization. The dataset is split into training and testing subsets to evaluate model generalization.

# 4. Feature Engineering

Feature engineering is implemented in `feature_utils.py`. TF-IDF vectorization is applied to textual data to capture keyword importance, while handcrafted numerical features represent structural characteristics such as text length and numeric density.

# 5. Machine Learning Models

Two Random Forest models are employed. The classification model predicts discrete difficulty classes, while the regression model predicts a continuous difficulty score. Both models are implemented using scikit-learn pipelines to ensure consistent preprocessing during inference.

# 6. Experimental Setup

The dataset is divided into training and testing sets with an 80-20 split. Models are trained on the training set and evaluated on the test set using standard performance metrics.

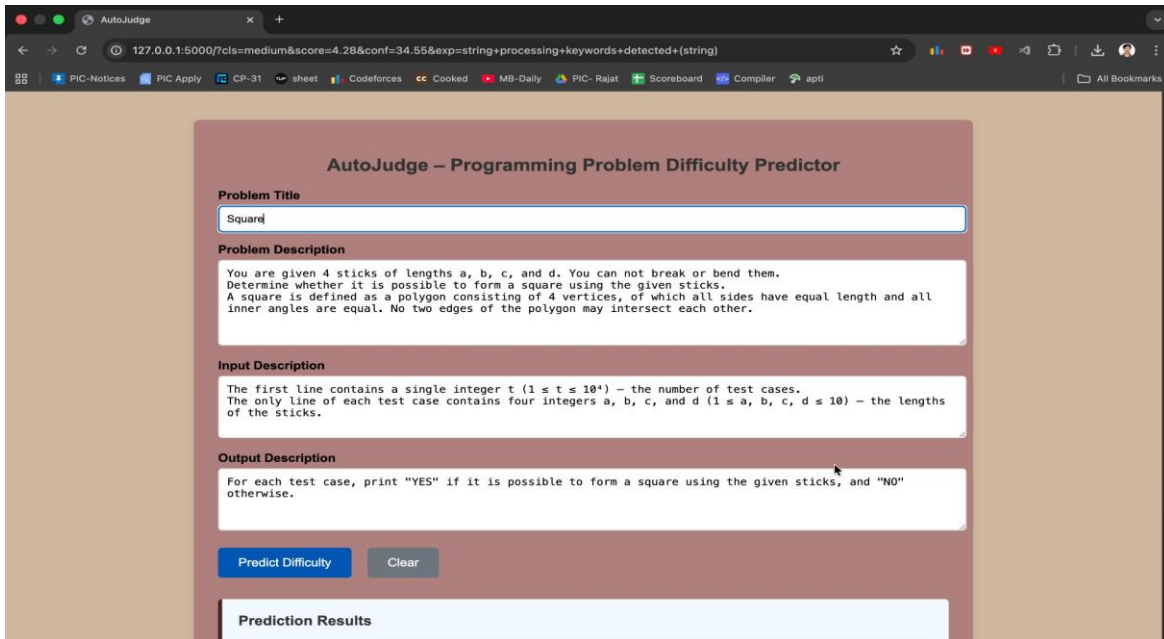
# 7. Evaluation Metrics and Results

Metric	Exact Value
Classification Accuracy	54.19%
Mean Absolute Error (MAE)	1.696
Root Mean Square Error (RMSE)	2.043

The classification accuracy reflects the challenging and subjective nature of difficulty prediction. The confusion matrix analysis shows that the model performs best on Medium difficulty problems, which dominate the dataset. Regression metrics indicate acceptable error margins for predicting difficulty scores.

## 8. Web Interface and Sample Predictions

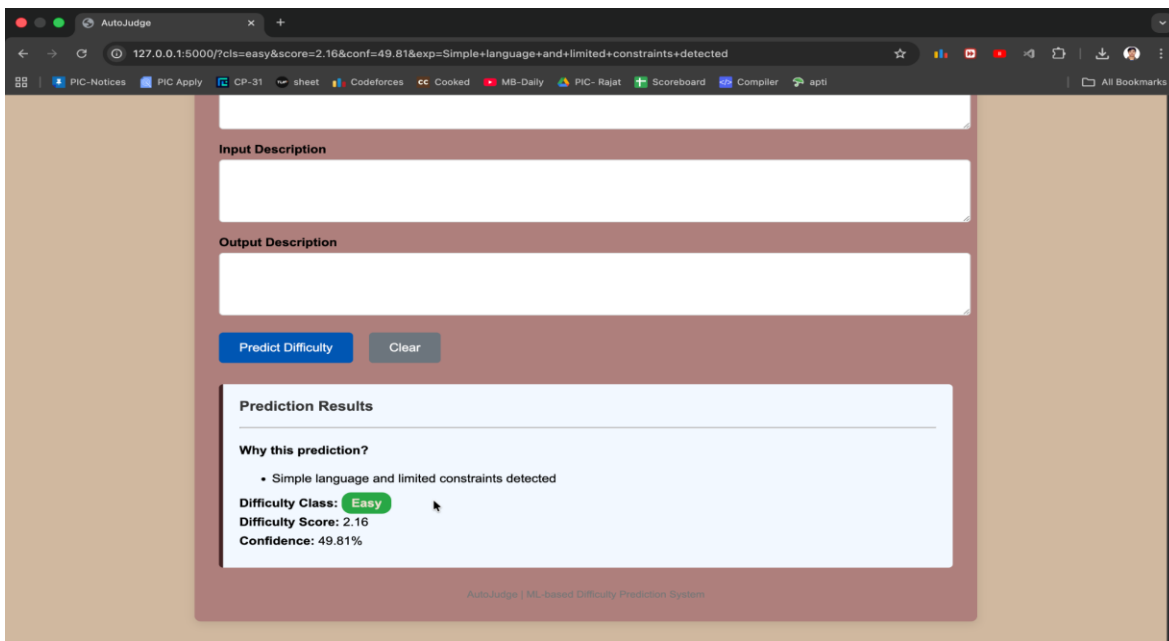
A Flask-based web interface allows users to enter problem details and obtain instant difficulty predictions. The screenshots below demonstrate the system behavior for Easy and Hard problems during local execution.



The screenshot shows a web browser window with the URL `127.0.0.1:5000/?cls=medium&score=4.28&conf=34.55&exp=string+processing+keywords+detected+(string)`. The page title is "AutoJudge – Programming Problem Difficulty Predictor". The form contains the following sections:

- Problem Title:** A text input field containing "Square".
- Problem Description:** A text area containing the problem statement: "You are given 4 sticks of lengths a, b, c, and d. You can not break or bend them. Determine whether it is possible to form a square using the given sticks. A square is defined as a polygon consisting of 4 vertices, of which all sides have equal length and all inner angles are equal. No two edges of the polygon may intersect each other."
- Input Description:** A text area containing: "The first line contains a single integer t ( $1 \leq t \leq 10^4$ ) – the number of test cases. The only line of each test case contains four integers a, b, c, and d ( $1 \leq a, b, c, d \leq 10$ ) – the lengths of the sticks."
- Output Description:** A text area containing: "For each test case, print 'YES' if it is possible to form a square using the given sticks, and 'NO' otherwise."
- Buttons:** "Predict Difficulty" (blue) and "Clear" (grey).
- Prediction Results:** A light blue box at the bottom, currently empty.

Figure 1: Easy problem input (Square problem)



The screenshot shows the same web browser window after clicking "Predict Difficulty". The URL is `127.0.0.1:5000/?cls=easy&score=2.16&conf=49.81&exp=Simple+language+and+limited+constraints+detected`. The "Prediction Results" section is now populated:

- Why this prediction?** A list containing: "Simple language and limited constraints detected".
- Difficulty Class:** "Easy" (highlighted in a green pill).
- Difficulty Score:** 2.16
- Confidence:** 49.81%

At the bottom of the results box, it says "AutoJudge | ML-based Difficulty Prediction System".

Figure 2: Easy problem prediction result

**AutoJudge – Programming Problem Difficulty Predictor**

**Problem Title**  
Mani and Segments

**Problem Description**  
An array  $b$  of length  $|b|$  is cute if the sum of the length of its Longest Increasing Subsequence (LIS) and the length of its Longest Decreasing Subsequence (LDS), is exactly one more than the length of the array. More formally, the array  $b$  is cute if  $LIS(b) + LDS(b) = |b| + 1$ .

**Input Description**  
Each test contains multiple test cases. The first line contains the number of test cases  $t$  ( $1 \leq t \leq 10^4$ ). The description of the test cases follows.

**Output Description**  
For each test case, output the number of cute non-empty subarrays of permutation  $a$ .

**Predict Difficulty** **Clear**

**Prediction Results**

Figure 3: Hard problem input (Mani and Segments)

**Input Description**

**Output Description**

**Predict Difficulty** **Clear**

**Prediction Results**

**Why this prediction?**

- Long problem description (higher complexity)
- Many numeric values / constraints present

**Difficulty Class:** **Hard**

**Difficulty Score:** 6.01

**Confidence:** 38.82%

AutoJudge | ML-based Difficulty Prediction System

Figure 4: Hard problem prediction result

## 9. Discussion

The results indicate that textual complexity and numeric constraints significantly influence difficulty prediction. Confusion between Medium and Hard categories arises due to overlapping problem characteristics. Despite moderate accuracy, the system demonstrates reliable behavior for practical use cases.

## 10. Conclusion and Future Work

This project successfully implements an end-to-end machine learning pipeline for predicting programming problem difficulty. Future work may explore dataset balancing, transformer-based language models, and advanced feature representations to improve performance.