AutoJudge ACM – Predicting Programming Problem Difficulty Report

Name: Killada Dhanur Vishnu
Institution: IIT Roorkee
Roll No: 22118037

Club: ACM Student Chapter, IIT Roorkee

1. Introduction

Programming competitions often classify problems into difficulty levels (Easy, Medium, Hard). Accurate difficulty prediction can help:

- Curate problem sets for contests
- Recommend problems for learners
- Analyze problem distribution on online judges

AutoJudge is a machine learning-based system that predicts the difficulty of programming problems from their textual descriptions. It performs:

1. **Classification:** Predicts difficulty class (Easy / Medium / Hard)
2. **Regression:** Predicts a numeric difficulty score

2. Dataset Description

The dataset contains problems scraped from online judges such as Kattis. Key columns:

Column Name	Description
title	Problem title
description	Problem statement
input_description	Description of input format
output_description	Description of output format
sample_io	Sample input/output
problem_class	Difficulty class (Easy / Medium / Hard)
problem_score	Numeric difficulty score

Text fields are combined for feature extraction.

3. Data Preprocessing

Text preprocessing steps:

1. Lowercasing all text
2. Removing special characters
3. Removing extra whitespaces
4. Combining multiple text fields into a single full_text

These steps ensure cleaner input for feature extraction.

4. Feature Engineering

We use two types of features:

4.1 TF-IDF Features

- Maximum features: 5000
- Captures semantic meaning from problem statements

4.2 Handcrafted Features

Feature	Description
Total text length	Number of characters in combined text
Description length	Number of characters in problem description
Input description length	Number of characters in input section
Output description length	Number of characters in output section
Number of digits	Counts numeric characters
Number of math symbols	Counts symbols like + - * / % = < >

These features complement TF-IDF by capturing structural aspects of the problem.

5. Models Used

5.1 Classification Model

- Model: Logistic Regression
- Task: Predict difficulty class (Easy / Medium / Hard)

5.2 Regression Model

- Model: Random Forest Regressor
- Task: Predict numeric difficulty score (out of 10)

Both models are trained offline and saved for inference.

6. Training & Experimental Setup

Training is documented in “notebooks/training.ipynb”, which includes:

- Loading the dataset
- Preprocessing and feature extraction
- Model training
- Performance evaluation

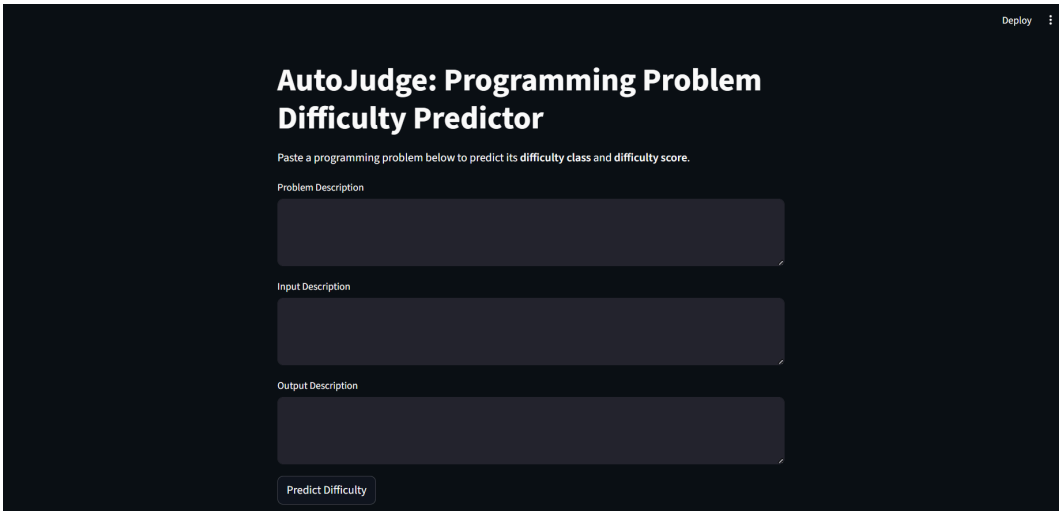
Environment: Python 3.9, scikit-learn, pandas, NumPy

7. Web Interface

The web application allows users to:

1. Copy Paste problem descriptions
2. Click **Predict Difficulty**
3. View:
    - Predicted difficulty class
    - Predicted difficulty score

Interface: Streamlit (runs locally)



Steps to run locally:

> git clone https://github.com/vish1466/AutoJudge_ACM.git
> cd AutoJudge_ACM
> pip install -r requirements.txt
> python app.py

8. Results & Sample Predictions

(Based on the testing dataset)

The results of the regressor are :

- MAE : 1.66
- RMSE : 2.01
-

The results of the classifier are :

- Accuracy : ~51.2%
- Confusion Matrix : (present in the AutoJudge_ACM/notebooks/training.ipynb file)

(Demonstration can be viewed in the video file link attached below)

https://drive.google.com/file/d/1PRpc67jxZvC7k57m2WR-5vtpXOhbdBHS/view?usp=sharing

9. Conclusion

- AutoJudge successfully predicted both categorical and numeric difficulty of programming problems.
- Combining TF-IDF with handcrafted features improves performance.
- The web interface demonstrates real-time predictions, making it useful for learners and contest organizers.

References

1. Kattis Online Judge – Dataset source
2. Scikit-learn Documentation – Machine learning models
3. Python Pandas / NumPy Documentation