AutoJudge: Programming Problem Difficulty Prediction Using Machine Learning

Author: Ronak Kumar
Enrollment No: 23113130
Department: Civil Engineering (3rd Year)

This report presents AutoJudge, a machine learning system designed to predict the difficulty of programming problems using only textual descriptions. The system performs both classification and regression to estimate difficulty class and a relative numerical difficulty score.

1. Introduction and Problem Statement

Competitive programming platforms host thousands of problems categorized by difficulty levels such as Easy, Medium, and Hard. These labels are critical for learning paths and contest design, but they are often assigned manually.

Manual difficulty labeling is subjective and inconsistent. The same problem may receive different labels across platforms. This motivates the need for an automated system that predicts difficulty using textual information alone.

2. Dataset Description

The dataset used in this project is sourced from publicly available competitive programming problems.

Dataset Link: https://github.com/AREEG94FAHAD/TaskComplexityEval-24

The dataset is provided in JSONL format. Each entry represents a single problem and contains fields such as title, description, input description, output description, sample input-output, difficulty class, and difficulty score.

3. Data Preprocessing

All textual fields were combined into a single text representation to capture full context. Missing values were replaced with empty strings to ensure no data loss.

Basic normalization was applied to ensure consistent formatting across the dataset.

4. Feature Engineering

Textual features were extracted using TF-IDF vectors, which capture term importance within the corpus.

In addition to TF-IDF, handcrafted features were used, including text length, numeric symbol count, and frequency of algorithmic keywords such as graph, BFS, DFS, DP, and recursion.

These features help capture both semantic and structural complexity of programming problems.

5. Models and Experimental Setup

Random Forest models were used for both classification and regression due to their robustness and ability to handle non-linear relationships.

The dataset was split into training and testing sets using a standard hold-out strategy. The same feature pipeline was used for both tasks, and default hyperparameters were retained to avoid overfitting.

6. Evaluation and Results

The classification model was evaluated using accuracy and confusion matrix. The achieved accuracy was approximately 50%, which is significantly higher than random guessing for a three-class problem.

The regression model was evaluated using Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE). These metrics measure the deviation between predicted and actual difficulty scores.

7. Web Interface and Sample Prediction

A Flask-based web interface was developed to demonstrate real-time predictions. Users can input problem descriptions and receive predicted difficulty class, relative difficulty score, confidence value, and explanation.

The explanation highlights detected keywords, text length, and numeric constraints to justify predictions.

8. Conclusion and Future Work

This project demonstrates an end-to-end machine learning pipeline for predicting programming problem difficulty using textual information alone.

Future work includes training on larger datasets, exploring deep learning models, and enhancing explainability techniques.