AutoJudge: Predicting Programming Problem Difficulty Using Textual Features

Abstract

Difficulty assignment of programming problems is a crucial yet subjective task in competitive programming platforms.

AutoJudge is a machine learning based system that predicts the difficulty of programming problems using only textual information.

The system performs both classification into Easy, Medium, and Hard categories and regression to predict a numeric difficulty score.

By leveraging natural language processing techniques such as TF-IDF along with classical machine learning models,

AutoJudge eliminates the dependency on user submissions and historical performance data.

1. Introduction

Competitive programming platforms rely heavily on accurate difficulty labels to ensure balanced contests and structured learning paths.

Traditionally, difficulty levels are manually assigned by problem setters, making the process time-consuming and prone to inconsistency.

This project aims to automate the difficulty prediction process using only problem statements.

2. Dataset Description

The dataset used in this project consists of approximately 4100 programming problems stored in JSON Lines format.

Each problem includes a title, detailed description, input specification, output specification, a categorical difficulty label,
and a numeric difficulty score.
The dataset is pre-labeled and suitable for supervised learning.

3. Data Preprocessing

Data preprocessing plays a critical role in ensuring model performance.
All missing textual fields were replaced with empty strings.
Text was converted to lowercase, unnecessary whitespaces were removed, and English stopwords were filtered using the NLTK library.
All text fields were then concatenated into a single combined_text feature to capture complete semantic information.

4. Feature Extraction

Textual data was converted into numerical form using TF-IDF vectorization.
The vocabulary was limited to the top 5000 features, stop words were removed, and both unigrams and bigrams were used.
In addition to TF-IDF, a numerical feature representing text length was introduced.
This feature captures verbosity and indirectly reflects problem complexity.
The TF-IDF vectors and numerical features were combined using sparse matrix operations.

5. Handling Class Imbalance

The dataset exhibits class imbalance across difficulty categories.
To mitigate this, SMOTE (Synthetic Minority Over-sampling Technique) was applied on

the training data.
This ensured equal representation of all difficulty classes and prevented model bias toward majority classes.

6. Model Architecture

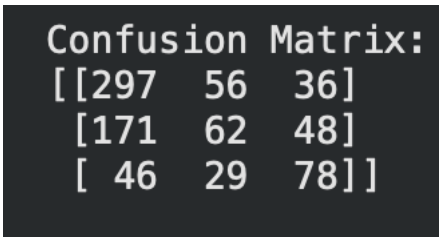
Multiple classification models were evaluated including Logistic Regression, Random Forest Classifier, and Linear Support Vector Machine.
Random Forest performed best due to its ability to capture non-linear relationships in high-dimensional feature spaces.
For regression, Linear Regression, Random Forest Regressor, and Gradient Boosting Regressor were tested.
Gradient Boosting Regressor achieved the lowest error metrics.

7. Experimental Setup

The dataset was split into training and testing sets using an 80-20 stratified split.
The same preprocessing pipeline was applied consistently across all models.
Model performance was evaluated using accuracy for classification and MAE and RMSE for regression.

8. Results and Evaluation

The Random Forest Classifier achieved an accuracy of approximately 53.10 percent.
The confusion matrix shows strong performance on Easy and Hard classes, while Medium problems remain ambiguous.



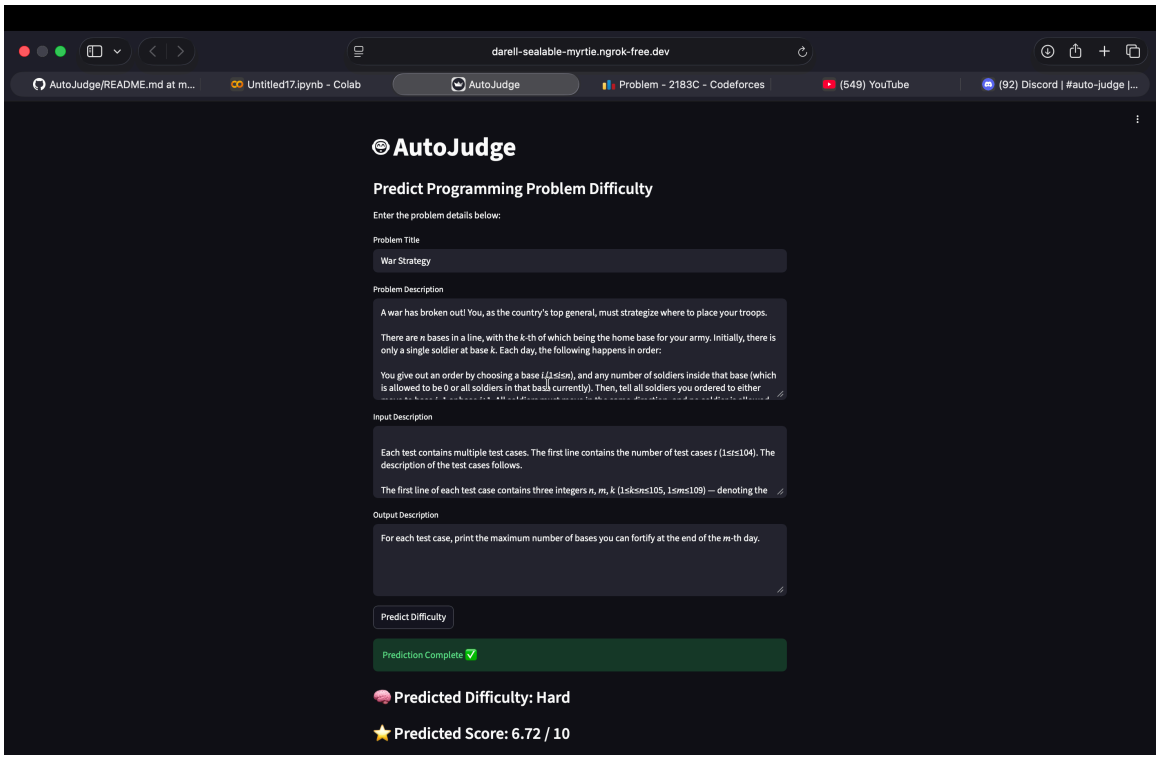
The Gradient Boosting Regressor achieved a Mean Absolute Error of approximately 1.617 and an RMSE of 1.940.

These results indicate that textual features alone can provide meaningful difficulty estimates.

9. Web Interface

A web application was developed using Streamlit to demonstrate real-time predictions. Users can input problem details and instantly receive predicted difficulty class and score. The interface loads pre-trained models and ensures consistent preprocessing during

inference.



10. Conclusion

AutoJudge demonstrates that programming problem difficulty can be reasonably predicted using textual information alone.
Although the system has limitations, particularly with medium difficulty problems, it provides a strong baseline.

AGASTYA AGARWAL

24112009

CHEMICAL ENGINEERING