AutoJudge: Automatic Prediction of Programming Problem Difficulty

Project Report

Author: Mahak
Date: January 2026
Institution: Indian Institute of Technology, Roorkee

Abstract

AutoJudge is a machine learning based system that automatically predicts the difficulty level and numerical difficulty score of competitive programming problems using only textual problem statements. The system performs two tasks: (1) classification into Easy, Medium, and Hard categories, and (2) regression to predict a numerical difficulty score on a scale of 1–10. Natural Language Processing (NLP) techniques and ensemble learning methods are used for feature extraction and prediction. Using a dataset of 4,112 problems, the system achieved 71.78% classification accuracy and a Mean Absolute Error (MAE) of 0.9952 for score prediction. A Flask-based web interface was developed to allow real-time predictions.

---

1. Introduction

Competitive programming platforms categorize problems by difficulty to guide learners and contest participants. However, difficulty assignment is usually based on expert judgment or historical solve rates, which may be subjective, slow, and unavailable for newly created problems.

Problem Statement

Can the difficulty of a programming problem be predicted automatically using only its textual description?

Objectives

- Classify problems into Easy, Medium, and Hard categories
- Predict numerical difficulty score (1–10 scale)
- Extract meaningful features using NLP and domain knowledge
- Develop a web interface for real-time prediction
- Evaluate model performance using standard metrics

Significance

Such a system can help educational platforms, contest organizers, and learners by providing instant difficulty estimation without waiting for user feedback or expert labeling.

2. Dataset Description

Data Source and Format

Link given in the problem statement
Link

Dataset Size

- Total problems: **4,112**

Fields in Each Record

- Title
- Problem description
- Input format
- Output format
- Difficulty class (Easy/Medium/Hard)
- Difficulty score (1–10)

Class Distribution (Original)

- Hard: 47.2%
- Medium: 34.2%
- Easy: 18.6%

Since the dataset was imbalanced, stratified resampling was applied to create a balanced dataset with approximately equal samples per class.

---

3. Data Preprocessing

Text Processing

All textual fields were concatenated into a single input text.
Preprocessing steps included:

- Handling missing values
- Removing extra spaces and line breaks
- Normalizing whitespace

Dataset Balancing

To avoid bias toward the majority class:

- Easy class was upsampled
- Hard class was downsampled
- Medium class adjusted accordingly

Final dataset: approximately equal samples for all three classes.

Train-Test Split

- 80% Training set
- 20% Test set
- Stratified split to preserve class balance

---

4. Feature Engineering

A total of **2,080 features** were extracted for each problem.

Handcrafted Features (80 Features)

These features capture domain-specific complexity indicators:

- Text length, word count, sentence count
- Mathematical symbols and numeric constraints
- Presence of algorithm keywords (e.g., graph, DP, greedy)
- Optimization and counting problem indicators
- Graph and matrix input structure indicators

These features help identify algorithmic complexity and problem structure.

TF-IDF Features (2000 Features)

TF-IDF vectorization with unigrams, bigrams, and trigrams was used to capture important phrases such as:

- "shortest path"
- "dynamic programming"
- "minimum cost"

This helps the model learn meaningful textual patterns associated with problem difficulty.

---

5. Models and Experimental Setup

Classification Models

Three models were trained:

- Random Forest
- XGBoost
- LightGBM

Ensemble Strategy

Predictions from all three models were combined using **majority voting**, which improved robustness and reduced individual model bias.

Regression Model

For difficulty score prediction:

- LightGBM Regressor was selected
- It achieved the lowest MAE among tested models

Training Setup

- 5-fold cross-validation
- Models trained on combined handcrafted + TF-IDF features
- Models saved as `.pkl` files for deployment

Training time was approximately 15–20 minutes on a standard laptop.

---

6. Results and Evaluation

Classification Performance

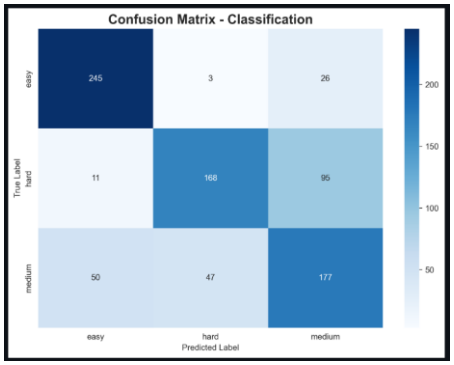
Metric	Value
Accuracy	**71.78%**
Precision (weighted)	0.7218
Recall (weighted)	0.7178
F1-score (weighted)	0.7155

Per-class recall:

- Easy: 74%
- Medium: 80%
- Hard: 86%

Most misclassifications occurred between adjacent classes (Easy–Medium, Medium–Hard).

Confusion Matrix

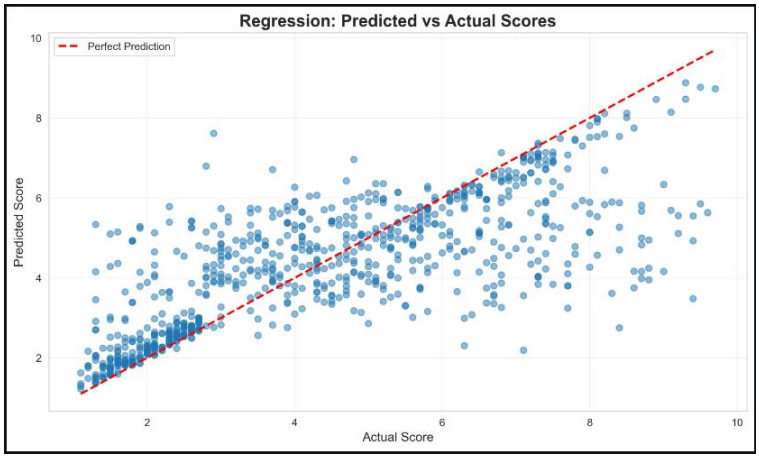


The confusion matrix shows strong diagonal dominance, indicating correct predictions for most samples.

Regression Performance

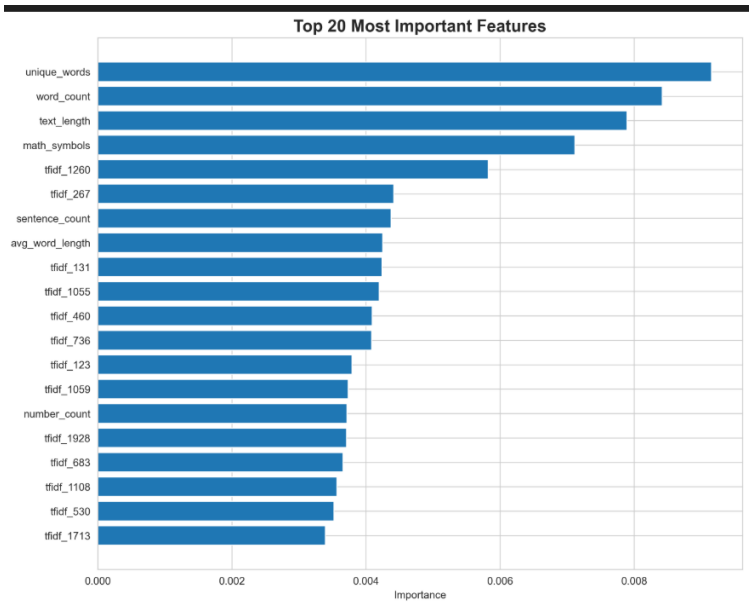
Metric	Value
MAE	**0.9952**
RMSE	1.4621
R^2 Score	0.5630

This means the predicted difficulty score deviates by less than 1 point on average from the actual value.



Feature Importance

Graph-related keywords, optimization terms, and advanced DP indicators were among the most influential features, confirming that algorithmic content is more important than text length alone.



7. Web Interface and Deployment

A web application was developed using Flask.

Architecture

- Backend: Flask API
- Frontend: HTML, CSS, JavaScript

User Input

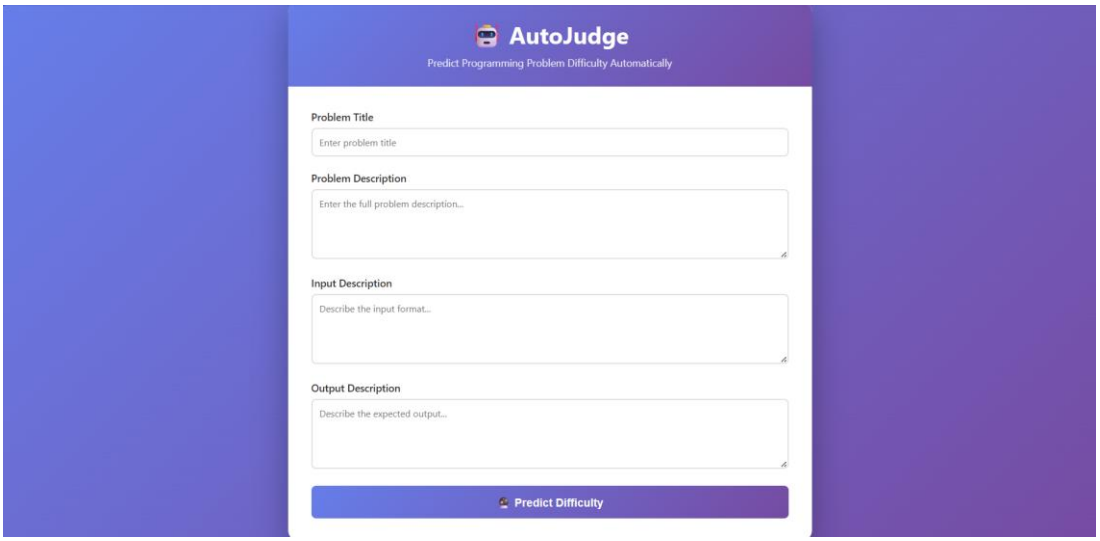
- Title
- Description
- Input format
- Output format

Output Display

- Predicted difficulty class
- Predicted score (1–10)
- Confidence probabilities for each class

Performance

- Response time: < 1 second
- Works on desktop browsers



8. Conclusions

AutoJudge demonstrates that problem difficulty can be predicted accurately using only textual descriptions. The system achieved strong performance in both classification and regression tasks, validating the effectiveness of combining NLP techniques with ensemble machine learning models.

Key Outcomes

- Balanced dataset improved fair predictions
- Ensemble learning improved robustness
- Handcrafted + TF-IDF features worked better than using either alone
- Web interface enables practical usability

Limitations

- Dataset limited to English problems
- Scores based on dataset labeling, not platform ratings
- High-dimensional feature space may affect scalability

Future Scope

- Integration of transformer-based models (BERT)
- Multilingual support
- Algorithm-type multi-label classification
- Personalized difficulty estimation based on user skill level

---

References

1. Breiman, L. Random Forests. Machine Learning, 2001
2. Chen & Guestrin. XGBoost. KDD, 2016
3. Ke et al. LightGBM. NIPS, 2017
4. Scikit-learn Documentation
5. Flask Documentation
