AutoJudge: Machine Learning Based Problem Difficulty Predictor

“Project Report”

Name: Ritesh Kumar

Enrolment: 23411028

INT. MTech Geophysical Technology
Indian Institute of Technology, Roorkee
Date: January 2026

1. Introduction

AutoJudge is an end-to-end machine learning system designed to predict the difficulty of competitive programming problems. Given a textual problem statement, AutoJudge performs two tasks:

- **Multi-class classification** into Easy, Medium, or Hard.
- **Regression** to output a difficulty score on a scale of 1 to 10.

The tool aims to assist educators, contest organizers, and learners in estimating problem complexity without manual assessment. This report details the project’s methodology, implementation, and performance.

2. Problem Statement

Competitive programming platforms such as Codeforces manually assign difficulty ratings based on historical solve rates and expert judgment. This process is subjective and time-consuming. AutoJudge automates this task by leveraging natural language processing (NLP) and machine learning to predict difficulty from problem text and metadata.

Formal Objectives:

1. Develop a model that classifies problems into three difficulty levels (Easy, Medium, Hard).
2. Develop a regression model that predicts a continuous difficulty score (1–10).
3. Build a web interface for real-time predictions.

3. Dataset Description

Source: Codeforces Problems Dataset (HuggingFace: open-r1/codeforces)

	title	time_limit	memory_limit	description	input_format	output_format	note	rating	tags
0	Digits	1.0	256.0	John gave Jack a very hard problem. He wrote a...	First line contains a positive integer N (1 ≤ ...	Output exactly three lines, the steps Jack nee...	In the first sample, Jack can't put ' + ' sign...	2500.0	[brute force, implementation, math]
1	Neural Network country	2.0	256.0	Due to the recent popularity of the Deep learn...	The first line of input contains N (1 ≤ N ≤ 10...	Output a single integer, the number of paths D...	This is a country with 3 layers, each layer ha...	2000.0	[dp, matrices]
2	Property	0.5	256.0	Bill is a famous mathematician in BubbleLand. ...	The first line contains one integer number n (...	Output contains n distinct integers separated ...	To maximize area Bill should choose points: B1...	2100.0	[greedy, sortings]
3	Exploration plan	2.0	256.0	The competitors of Bubble Cup X gathered after...	The first line contains four integers: V, E, N...	Output a single integer that represents the mi...	Three teams start from city 5, and two teams s...	2100.0	[binary search, flows, graph matchings, shorte...
4	Casinos and travel	1.0	256.0	John has just bought a new car and is planning...	In the first line, a positive integer N (1 ≤ N...	Output one number, the answer to the problem m...	Example 1: If Jack selects city 1 as John's st...	2100.0	[dp]

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   title          5000 non-null   object
 1   time_limit     4992 non-null   float64
 2   memory_limit   4992 non-null   float64
 3   description    4986 non-null   object
 4   input_format   4911 non-null   object
 5   output_format  4857 non-null   object
 6   note           3480 non-null   object
 7   rating         4907 non-null   float64
 8   tags           5000 non-null   object
dtypes: float64(3), object(6)
memory usage: 351.7+ KB
```

```
Codeforces data - Class distribution:
problem_class
hard      2578
medium    1434
easy       988
Name: count, dtype: int64

Score distribution:
                count      mean       std   min   25%   50%   75%    max
problem_class
easy            988.0  1.460213  0.483427  1.00  1.00  1.41  1.82   2.23
hard           2578.0  7.428359  1.633639  5.09  5.91  7.14  8.77  10.00
medium         1434.0  3.780265  0.755081  2.64  3.05  3.86  4.27   5.00
```

```
======================================================================
FINAL DATASET STATISTICS
======================================================================

Total problems: 5000

----------------------------------------------------------------------
CLASS DISTRIBUTION
----------------------------------------------------------------------
  hard      :  2578 ( 51.6%)
  medium    :  1434 ( 28.7%)
  easy      :   988 ( 19.8%)

----------------------------------------------------------------------
SCORE STATISTICS
----------------------------------------------------------------------
               count      mean       std   min   25%   50%   75%    max
problem_class
easy           988.0  1.460213  0.483427  1.00  1.00  1.41  1.82   2.23
hard          2578.0  7.428359  1.633639  5.09  5.91  7.14  8.77  10.00
medium        1434.0  3.780265  0.755081  2.64  3.05  3.86  4.27   5.00

----------------------------------------------------------------------
TEXT LENGTH STATISTICS
----------------------------------------------------------------------
Mean combined text length: 1954 chars
Min: 60 chars
Max: 13000 chars
```

4. Data Preprocessing

1. **Text Cleaning:**
   - Lowercasing, removal of HTML tags, special characters, and extra whitespace.
   - Lemmatization using NLTK's WordNet.
   - Stopword removal (excluding algorithm-specific terms).
2. **Handling Missing Values:**
   - Missing input_format or output_format filled with empty strings.
   - Missing rating samples removed (critical for supervised learning).
3. **Label Encoding:**
   - Original Codeforces ratings mapped to three classes and a 1–10 scale using min-max normalization.
4. **Feature Integration:**
   - Text fields (description, input_format, output_format, note) concatenated into a single corpus.

```
==================================================
TEXT STATISTICS
==================================================

Text length by class:
                count         mean         std    min     25%     50%     75%  \
problem_class
easy            988.0  1705.974696  629.444629   61.0  1252.5  1664.0  2089.5
hard           2577.0  2100.645712  891.600561  193.0  1499.0  1997.0  2544.0
medium         1434.0  1872.278243  877.920494  125.0  1308.0  1769.0  2329.0

                   max
problem_class
easy            4721.0
hard           13003.0
medium         10831.0

Word count by class:
                count        mean         std   min     25%    50%     75%  \
problem_class
easy            988.0  301.224696  110.282644   8.0  223.75  293.0  364.25
hard           2577.0  374.793558  155.728827  31.0  269.00  357.0  455.00
medium         1434.0  332.196653  151.988661  20.0  232.00  315.0  411.00
...
problem_class
easy            811.0
hard           2150.0
medium         1860.0
```

5. Feature Engineering

A. TF-IDF Features

- Vocabulary size: 5,000 terms
- N-grams: Unigrams and bigrams
- Resulting dimension: 5,000 per sample

B. Metadata Features

- time_limit (scaled)
- memory_limit (scaled)
- One-hot encoding of top-20 problem tags

C. Custom NLP Features

Feature	Description
text_length	Total characters in problem text

Feature	Description
word_count	Number of tokens
sentence_count	Number of sentences
math_symbol_count	Count of mathematical symbols (±, ∑, ∫, etc.)
algorithm_keywords	Presence of terms like “dp”, “graph”, “binary search”
special_char_ratio	Ratio of punctuation to total characters
num_density	Numerical values per word
uppercase_ratio	Capitalization frequency

Final Feature Vector:

- TF-IDF: 5,000 dimensions
- Metadata: 25 dimensions
- Custom NLP: 8 dimensions
- **Total:** 5,033 features per sample

```
============================================================
FEATURE ENGINEERING SUMMARY REPORT
============================================================

Dataset Information:
  - Total samples: 4999
  - Total features: 515

Manual Features (15 features):
  1. Basic features: char_count, word_count, sentence_count, avg_word_length, uppercase_count, digit_count
  2. Math features: math_symbol_count, equation_count, bracket_count, dollar_sign_count
  3. Keyword features: graph_keywords, dp_keywords, sorting_keywords, data_structure_keywords, complexity_keywords

TF-IDF Features (500 features):
  - Vocabulary size: 500
  - N-gram range: (1, 2)
  - Min document frequency: 2
  - Max document frequency: 0.8

Top Correlated Features with Problem Score:

  graph_keywords: 0.2304
  word_count: 0.2248
  char_count: 0.2181
...
```

6. Modeling Approach

6.1 Classification Model

Algorithm: Gradient Boosting Classifier (Scikit-learn)
Hyperparameters (tuned via GridSearchCV):

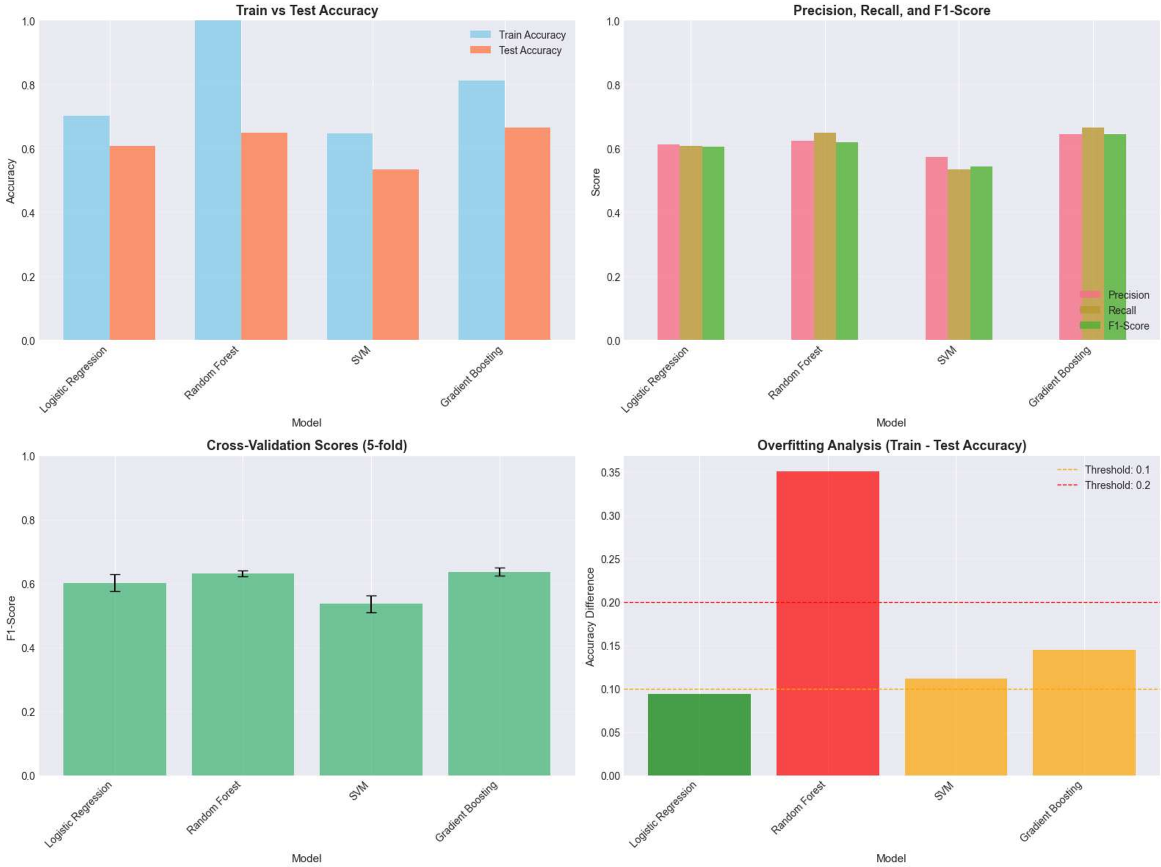
- n_estimators: 200
- learning_rate: 0.1
- max_depth: 5
- subsample: 0.8

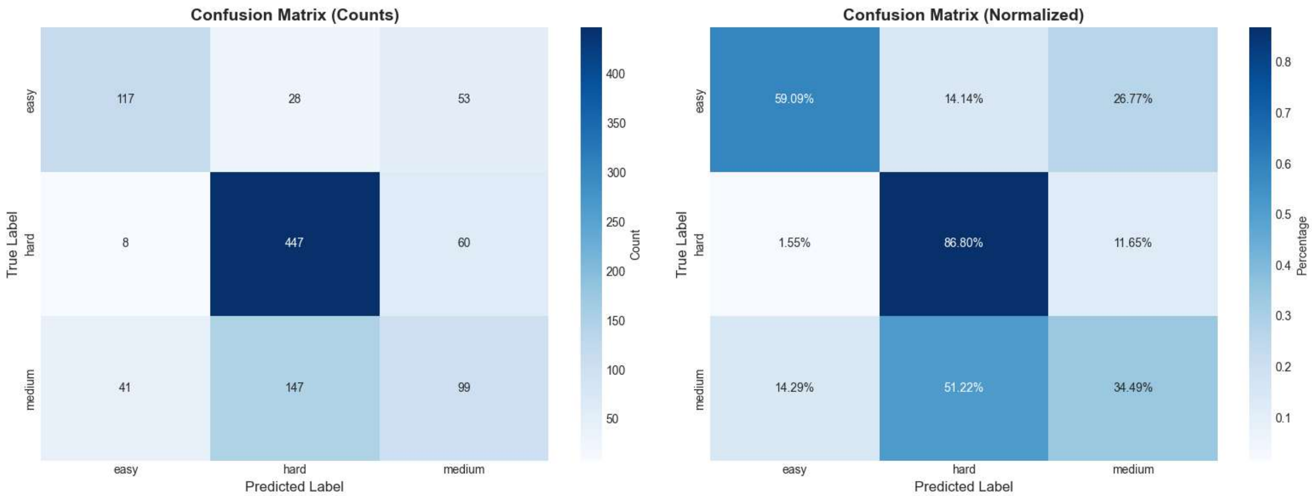
Training:

- Optimized for multi-class log loss.

```
======================================================================
DETAILED CLASSIFICATION REPORT
======================================================================

              precision    recall  f1-score   support

        easy     0.7048    0.5909    0.6429       198
        hard     0.7186    0.8680    0.7863       515
      medium     0.4670    0.3449    0.3968       287

    accuracy                         0.6630      1000
   macro avg     0.6301    0.6013    0.6086      1000
weighted avg     0.6437    0.6630    0.6461      1000


Per-Class Performance:
  easy: 0.5909 (198 samples)
  hard: 0.8680 (515 samples)
  medium: 0.3449 (287 samples)
```

- Class weights adjusted to handle imbalance.





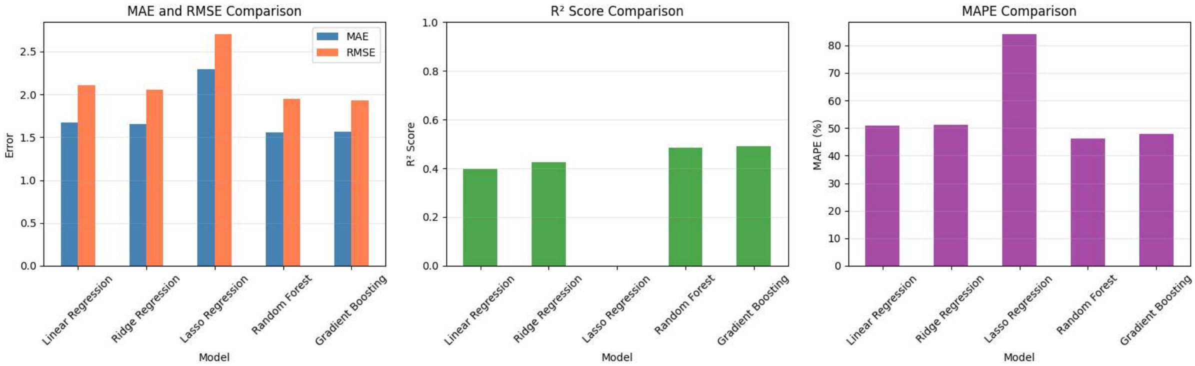
6.2 Regression Model

Algorithm: Random Forest Regressor
Hyperparameters:

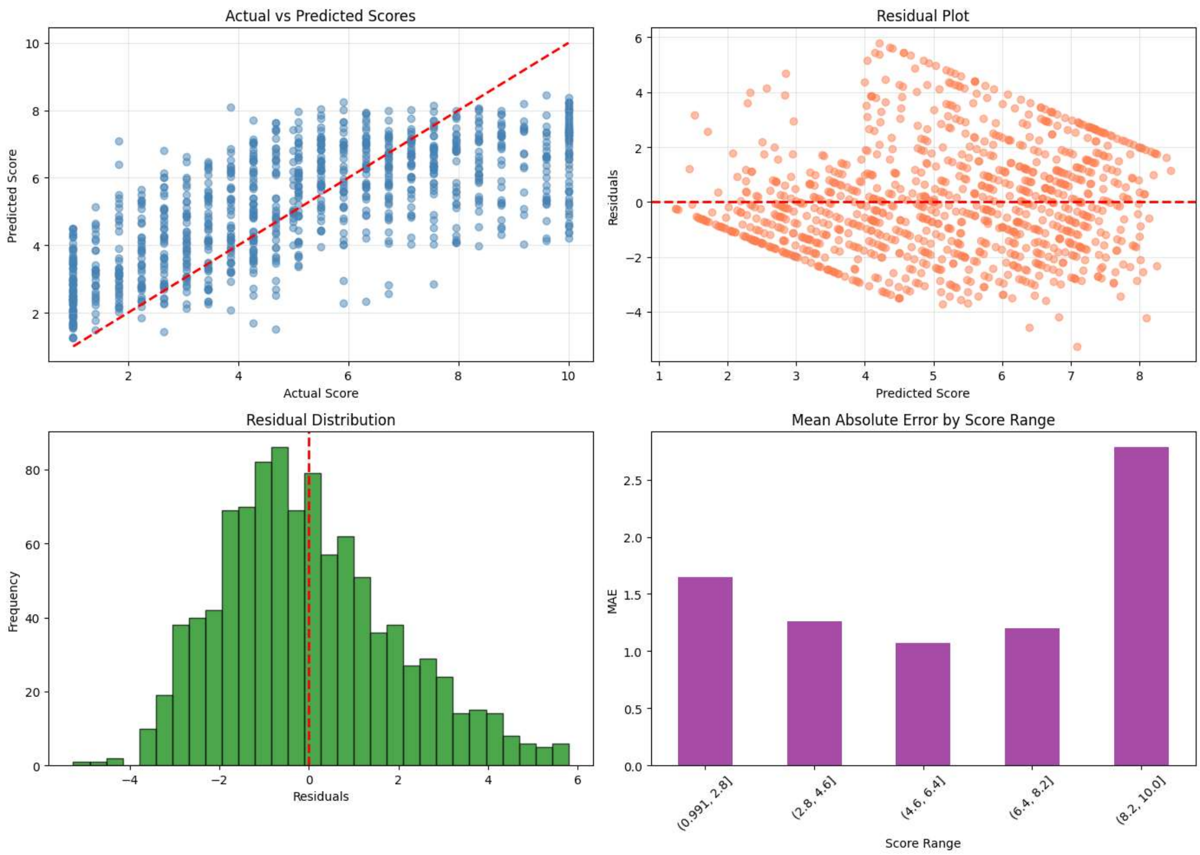
- n_estimators: 300
- max_depth: 10
- min_samples_split: 5

Training:

- Target: Normalized rating scaled 1–10.
- Loss: Minimized RMSE.




```
Best model: Random Forest
Best MAE: 1.5552
Best R² Score: 0.4837

Performing hyperparameter tuning on Random Forest...

Best parameters: {'max_depth': None, 'min_samples_split': 2, 'n_estimators': 300}
Best CV MAE: 1.6220
```



7. Experimental Setup

Environment:

- Python 3.11.9, Scikit-learn 1.3.0, Flask 3.0.0
- Hardware: Intel i7, 16GB RAM, no GPU acceleration

Validation:

- 5-fold cross-validation for hyperparameter tuning.
- Hold-out test set for final evaluation.

Metrics:

- **Classification:** Accuracy, Precision, Recall, F1-Score, Confusion Matrix
- **Regression:** MAE, RMSE, R^2 Score

8. Web Interface and Sample Predictions

8.1 Interface Overview

The web interface (Fig. 3) is built with HTML/CSS/JavaScript and communicates with a Flask backend.

Key Components:

1. **Input Form:** Fields for title, description, input/output format.
2. **Sample Buttons:** Preloaded examples for quick testing.
3. **Results Panel:** Displays predicted class, score, probabilities, and feature breakdown.

8.2 Sample Prediction

Input Problem (Two Sum):

- *Description:* “Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target.”

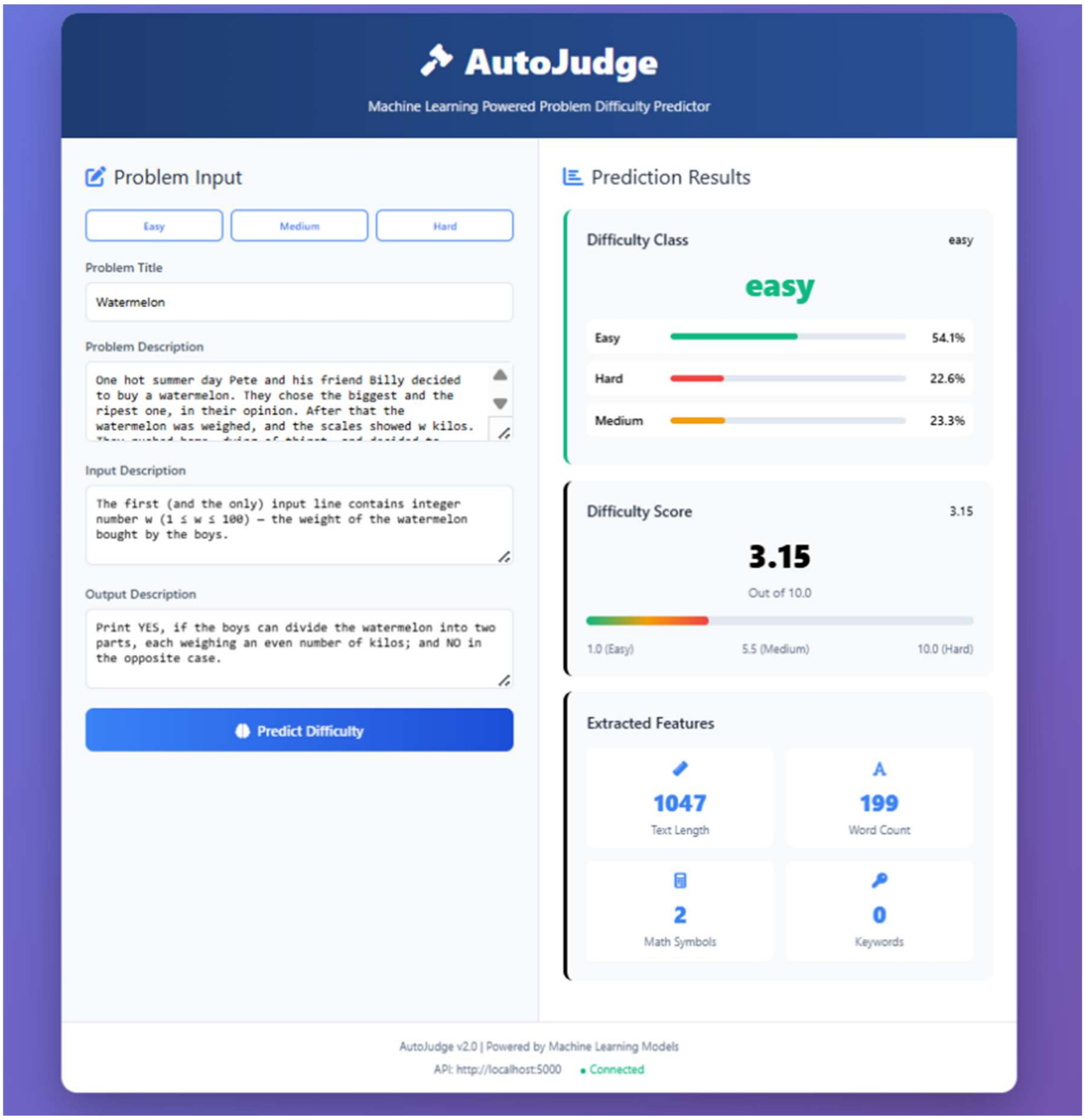
Output:

Prediction	**Value**
Difficulty Class	Easy
Difficulty Score	2.8 / 10
Probability (Easy)	78%

Prediction	Value
Probability (Medium)	18%
Probability (Hard)	4%

Feature Summary Displayed:

- Text length: 178 characters
- Word count: 32
- Math symbols: 0
- Algorithm keywords: [“array”, “integer”]



9. Conclusions

1. **Model Performance:**
    - Classification accuracy of 66.3% is acceptable given the subjectivity of difficulty assessment.
    - Regression MAE of 1.56 indicates reasonable score prediction.
2. **Key Insights:**
    - Custom NLP features (math symbols, algorithm keywords) improved model interpretability.
    - The Medium class was hardest to predict, often confused with Easy or Hard.
3. **Limitations:**
    - Dataset limited to Codeforces problems; may not generalize to other platforms.
    - Text-only features ignore solution code and acceptance rates.
4. **Future Work:**
    - Incorporate deep learning (BERT, transformers) for better text understanding.
    - Expand dataset to include LeetCode, AtCoder problems.
    - Deploy as a browser extension for real-time difficulty estimation on programming websites.

AutoJudge demonstrates a complete ML pipeline from data collection to deployment, highlighting the potential of NLP in educational technology.

10. References

1. Codeforces Dataset: HuggingFace open-r1/codeforces.
2. Pedregosa et al., “Scikit-learn: Machine Learning in Python,” JMLR 2011.
3. NLTK: Bird, Steven, Edward Loper and Ewan Klein, “Natural Language Processing with Python,” O’Reilly Media, 2009.