Automated Difficulty Prediction for Competitive Programming Problems

Classification and Regression Using Textual Features

By- Aryan 21324004

1. Introduction

Online competitive programming platforms such as Codeforces, CodeChef, and Kattis classify problems into difficulty levels (Easy, Medium, Hard) and often assign numerical difficulty scores. These labels are typically derived from human judgment and community feedback, which can be subjective and time-consuming.

This project aims to build an **automated system** that predicts:

1. **Problem Difficulty Class** (Easy / Medium / Hard) — *classification*
2. **Problem Difficulty Score** (continuous value) — *regression*

The system relies **only on textual information** from problem statements and uses **classical machine learning techniques**, without deep learning.

2. Dataset Description

Each data sample contains the following fields:

- title
- description
- input_description
- output_description
- sample_io
- problem_class (Easy / Medium / Hard)
- problem_score (numerical)
- url (dropped during preprocessing)

Dataset Statistics

- Total samples: **4112**
- Difficulty score range: **1.1 – 9.7**
- Class distribution:
    - Hard: 1941
    - Medium: 1405
    - Easy: 766

3. Data Preprocessing

3.1 Text Consolidation

All textual fields were merged into a single column (full_text) to simulate how programmers naturally read problems:

Title + Description + Input + Output + Sample I/O

3.2 Missing Values

Missing text fields were filled with empty strings to preserve labeled samples. No rows were dropped since all labels were present.

3.3 Text Normalization

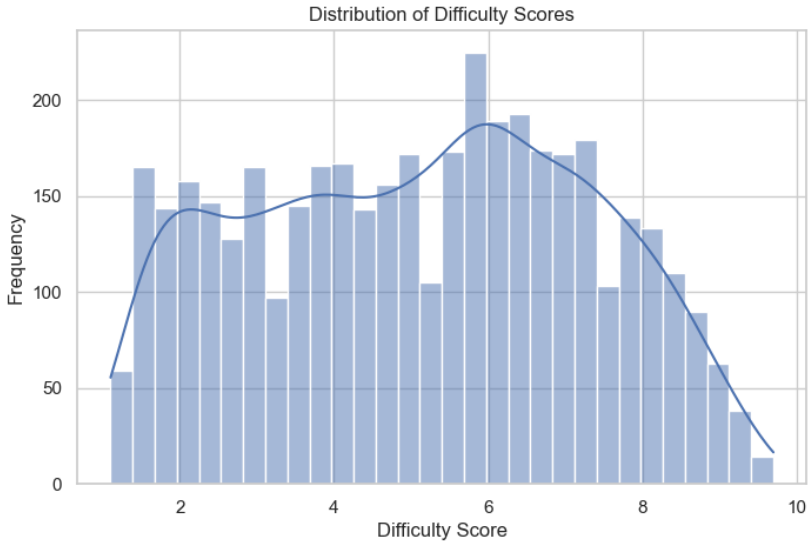
- Converted text to lowercase
- Normalized whitespace
- Retained numbers and mathematical symbols, as they convey difficulty-related information

4. Exploratory Data Analysis (EDA)

EDA was conducted to understand the relationship between textual characteristics and difficulty.

4.1 Target Variable Analysis

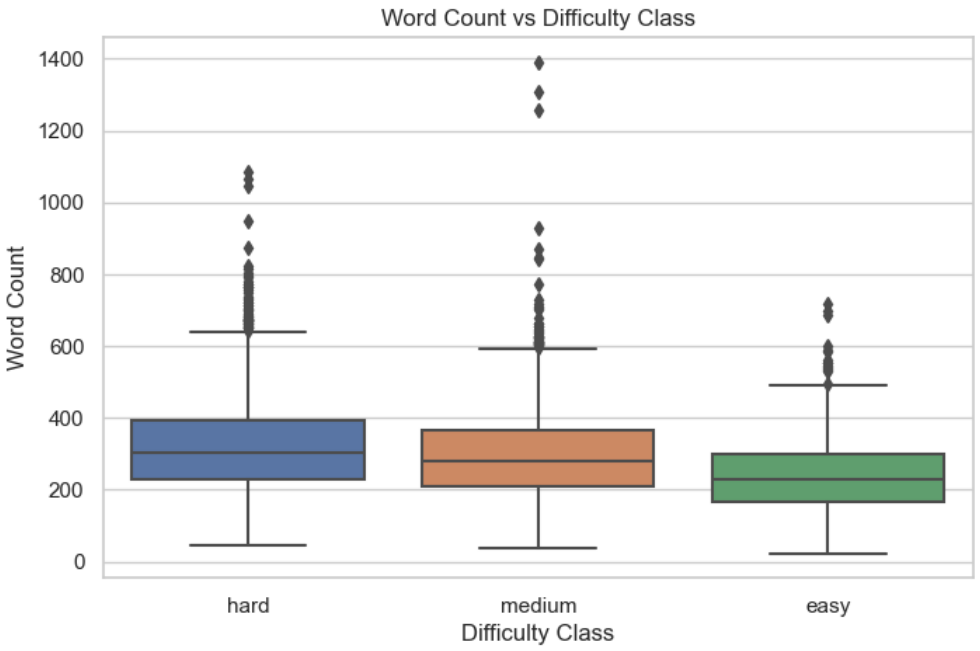
- **Problem Class Distribution** showed moderate class imbalance.
- **Difficulty Score Distribution** was approximately continuous and centered around medium difficulty.



4.2 Text Length & Structure Analysis

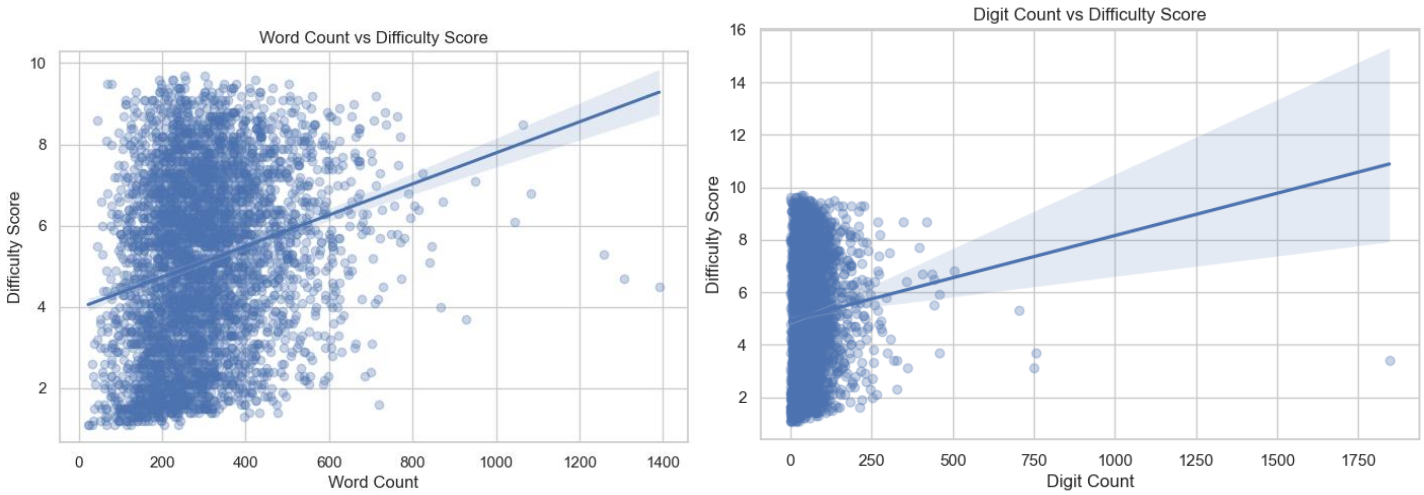
Hard problems tend to be longer and more structured.

- Word count vs difficulty class



4.3 Difficulty Score vs Text Complexity

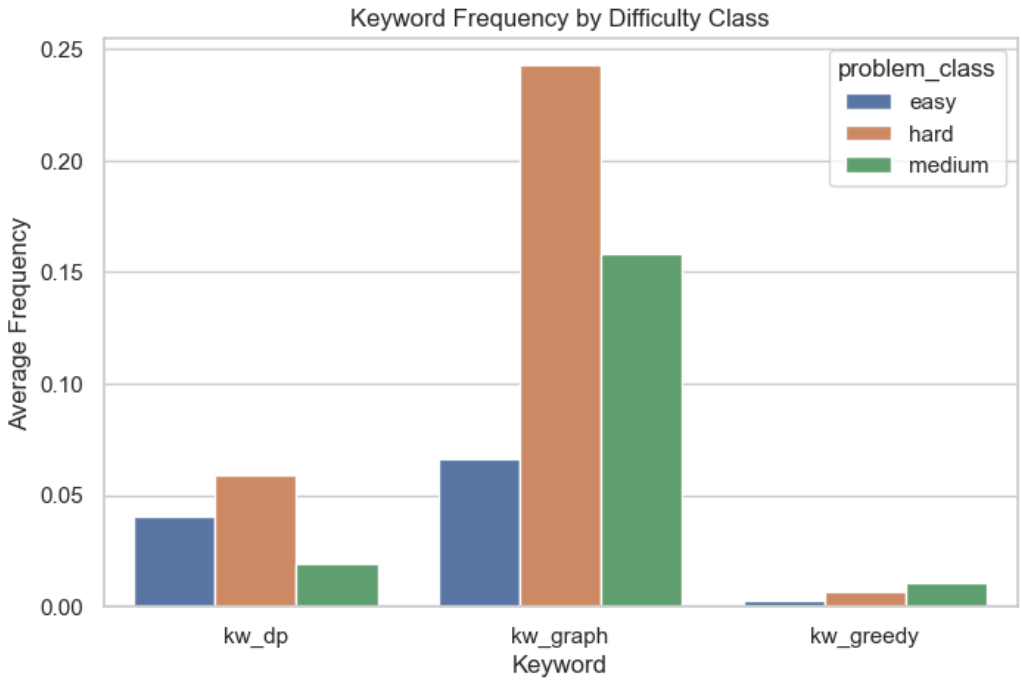
Text complexity correlates positively with difficulty score, though with significant variance.



4.4 Keyword & Symbol Analysis

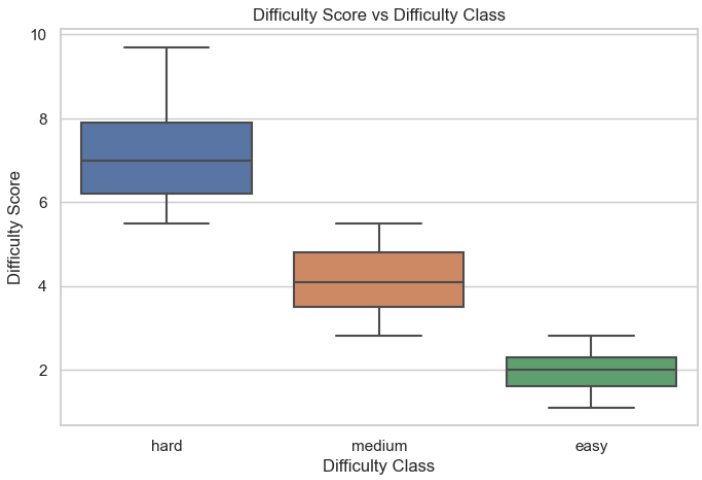
Algorithmic keywords and constraint symbols are strong indicators of difficulty.

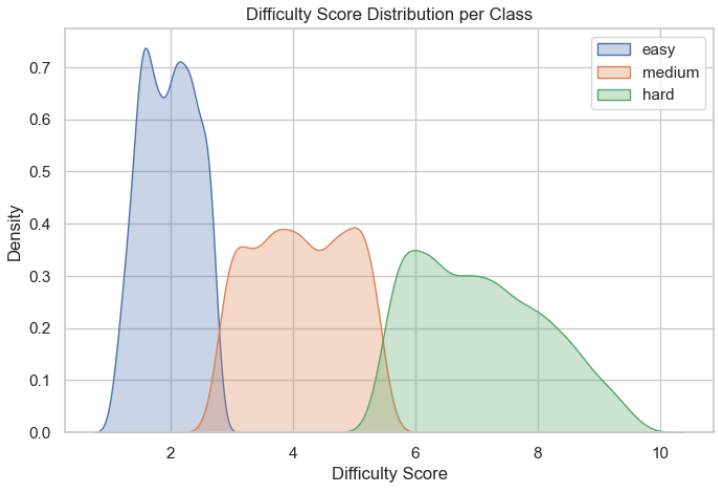
- Hard problems frequently contain terms such as dp, graph, union find
- Constraint symbols (<=, %, ^) appear more often in harder problems



4.5 Class–Score Relationship

There is substantial overlap between difficulty classes in terms of score, explaining why perfect classification is inherently difficult.





5. Feature Engineering

Three categories of features were used:

5.1 Semantic Features

- **TF-IDF vectors** (unigrams + bigrams)
- Limited vocabulary to reduce noise

5.2 Structural Features

- Character count
- Word count
- Line count
- Digit count

5.3 Algorithmic Signals

- Keyword frequencies (dp, graph, greedy, etc.)
- Mathematical and comparison symbol counts

All numeric features were standardized and concatenated with TF-IDF features.

6. Modeling Approach

6.1 Baseline Model

A trivial baseline classifier that **always predicts "Medium"** was implemented.

- Purpose: establish a lower bound
- Accuracy ≈ **0.34143377885783716**, matching class distribution

This confirms that learned models significantly outperform naive guessing.

7. Classification Models

7.1 Random Forest Classifier

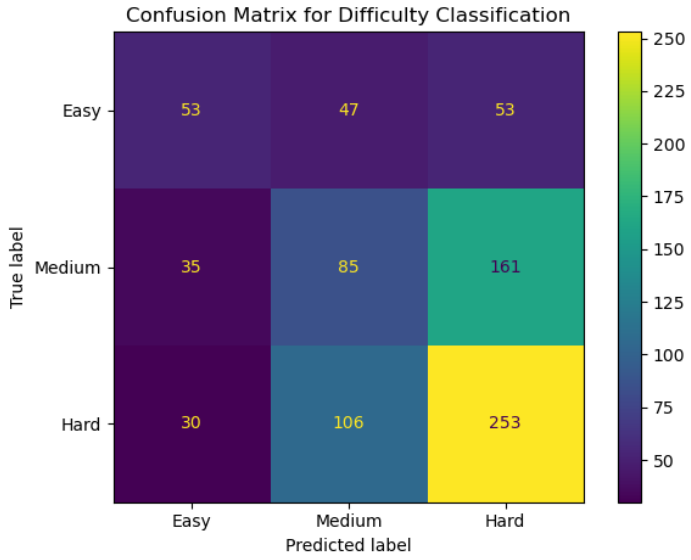
A Random Forest classifier was trained on combined TF-IDF and engineered features.

Evaluation Metrics

- **Accuracy**
- **Confusion Matrix**

Accuracy achieved: 0.5540704738760632

MAE: 1.6605750294768074



	precision	recall	f1-score	support
easy	0.53	0.42	0.47	153
hard	0.57	0.84	0.68	389
medium	0.52	0.23	0.32	281
accuracy			0.55	823
macro avg	0.54	0.50	0.49	823
weighted avg	0.54	0.55	0.52	823

Observations

- Performs significantly better than baseline
- Struggles with high-dimensional sparse text features
- Most confusion occurs between Medium and Hard classes

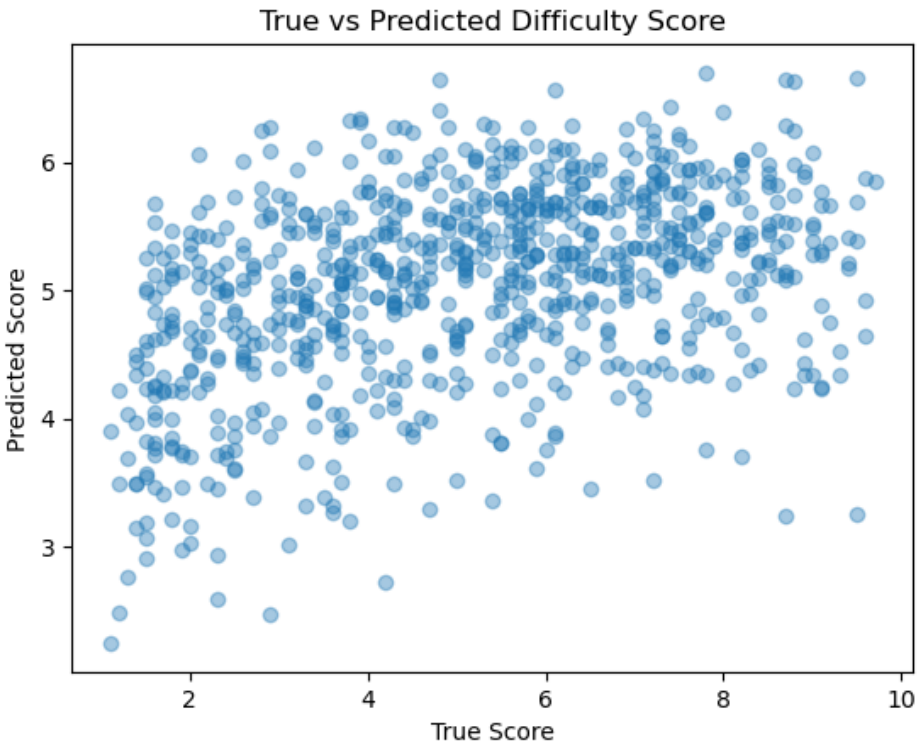
8. Regression Models

8.1 Random Forest Regressor

A Random Forest Regressor was used to predict numerical difficulty scores.

Evaluation Metrics

- **Mean Absolute Error (MAE)**
- **Root Mean Squared Error (RMSE)**



Interpretation

- MAE indicates reasonable average deviation from true difficulty
- RMSE highlights increased error for mid-range scores due to class overlap

9. Evaluation Summary

Classification

- Metric used: **Accuracy**
- Visualization: **Confusion Matrix**
- Baseline accuracy: ~0.34
- Random Forest accuracy: **~0.554**

Regression

- Metrics used: **MAE**, **RMSE**
- Performance demonstrates meaningful learning beyond mean prediction baseline

10. Discussion

- Text length and structure strongly correlate with difficulty
- Keyword and symbol features provide crucial signals
- Overlap between classes limits maximum achievable accuracy

- Random Forest models are interpretable but less suited for high-dimensional sparse text compared to linear models

11. Conclusion

This project demonstrates that:

- Programming problem difficulty can be reasonably predicted using **text alone**
- Classical ML models combined with careful feature engineering are effective
- Both classification and regression tasks provide complementary insights
- Proper EDA is essential to understand model limitations and performance

Future work could include hybrid models or hierarchical classification strategies to further improve accuracy.