AUTOJUDGE – AI BASED PROGRAMMING PROBLEM DIFFICULTY PREDICTOR

---

1. Introduction

Competitive programming platforms such as Codeforces, CodeChef, and LeetCode categorize problems into difficulty levels like *Easy*, *Medium*, and *Hard*. These difficulty labels play an important role in guiding learners, designing contests, and evaluating problem complexity. However, assigning difficulty manually is subjective and may vary across platforms and authors.

AutoJudge is an AI-based system designed to automate the prediction of programming problem difficulty using **Natural Language Processing (NLP)** and **Machine Learning**. The system analyzes problem statements and predicts both a **categorical difficulty level** and a **continuous difficulty score**, providing a more objective and fine-grained evaluation.

---

2. Objectives

The main objectives of this project are:

- To automatically analyze programming problem statements using NLP techniques
- To classify problems into **Easy, Medium, or Hard**
- To generate a **numerical difficulty score**
- To engineer meaningful textual and numerical features
- To build accurate classification and regression models
- To deploy the model using a user-friendly **Streamlit web interface**

---

3. Dataset Description

The dataset used in this project consists of competitive programming problems collected from online sources. Each data instance includes:

- Problem description
- Input description

- Output description
- Constraints
- Difficulty label (Easy / Medium / Hard)
- Difficulty score (numerical value)

The dataset contains both **textual** and **numerical** information, making it suitable for hybrid feature engineering.

---

4. Data Preprocessing

Text preprocessing is a crucial step to remove noise and improve model performance. The following preprocessing steps were applied:

- Conversion of text to lowercase
- Removal of special characters and unnecessary symbols
- Tokenization of text into words
- Removal of stopwords using NLTK
- Lemmatization using WordNet Lemmatizer

These steps help in normalizing the text and reducing dimensionality while preserving semantic meaning.

---

5. Feature Engineering

A hybrid feature engineering approach is used in AutoJudge.

5.1 Textual Features

- **TF-IDF (Term Frequency–Inverse Document Frequency)** vectorization is applied to cleaned text
- It captures the importance of words while reducing the effect of common terms

5.2 Handcrafted Numerical Features

Additional features were engineered to capture problem complexity:

- Length of the combined problem text
- Number of mathematical symbols (+, -, *, /, %, ^, etc.)

- Maximum constraint value extracted from constraints
- Presence of algorithmic keywords such as:
    - graph, dp, tree, greedy, shortest path, recursion, bitwise, modulo, etc.

These features significantly improve prediction accuracy.

---

6. Model Architecture

Two machine learning models are trained:

6.1 Classification Model

- **Random Forest Classifier**
- Predicts difficulty category: Easy, Medium, or Hard
- Chosen for robustness, interpretability, and ability to handle mixed feature types

6.2 Regression Model

- Machine Learning Regression model
- Predicts a continuous difficulty score
- Complements the classification output with finer granularity

---

7. Model Training

- The dataset is split into training and testing sets
- Feature selection using Random Forest feature importance is applied
- This reduces overfitting and improves generalization
- Hyperparameters are tuned to achieve optimal performance

---

8. Evaluation Metrics

The models are evaluated using appropriate metrics:

8.1 Classification Metrics

- **Accuracy**

```
Improved Accuracy: 0.5407047387606319
Classification Report:
              precision    recall  f1-score   support

        easy       0.47      0.48      0.47       153
        hard       0.58      0.81      0.68       389
      medium       0.44      0.20      0.27       281

    accuracy                           0.54       823
   macro avg       0.50      0.50      0.48       823
weighted avg       0.51      0.54      0.50       823


Confusion Matrix:
 [[ 73  50  30]
 [ 33 317  39]
 [ 49 177  55]]
```

8.2 Regression Metrics

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

```
MAE: 2.495633321790916
RMSE: 3.1508262679062633
```

These metrics provide both interpretability and precision for performance evaluation.

---

9. Web Application

AutoJudge is deployed as an interactive web application using **Streamlit**.

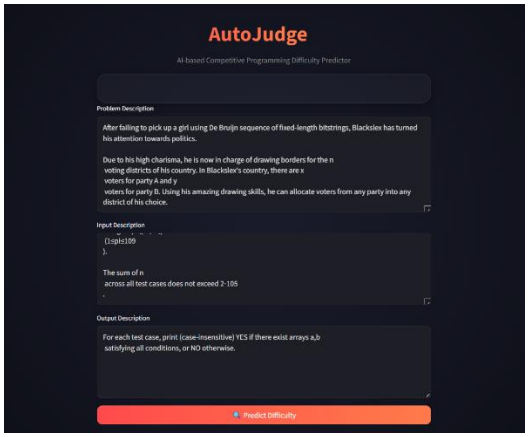
9.1 User Interface Flow

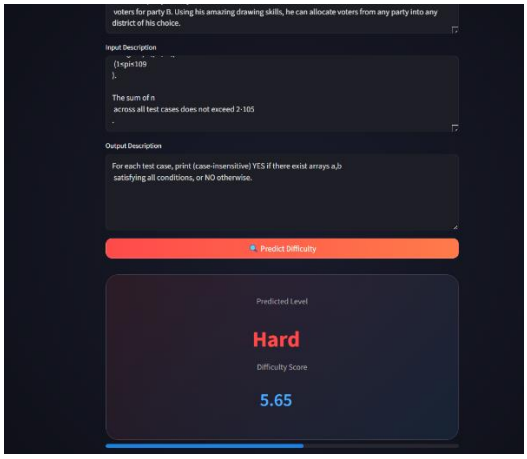
1. User enters:
    - Problem description
    - Input description
    - Output description
2. Text is cleaned and preprocessed
3. Features are extracted
4. Models generate predictions
5. Results are displayed:
    - Difficulty Level

- Difficulty Score

9.2 UI Features

- Real-time predictions
- Visual difficulty progress indicator





---

10. Results and Discussion

The system demonstrates strong predictive performance for both classification and regression tasks. Feature engineering plays a major role in improving accuracy. The numerical difficulty score provides more insight compared to categorical labels alone.

---

11. Limitations

- Model performance depends on dataset quality
- Highly ambiguous or poorly written problems may be misclassified
- Language bias may exist due to English-only training data

---

12. Future Enhancements

Possible future improvements include:

- Integration of transformer-based models such as BERT
- Support for multilingual problem statements
- Integration with competitive programming platforms
- Automated dataset expansion

---

13. Conclusion

AutoJudge successfully automates programming problem difficulty prediction using NLP and Machine Learning. The project demonstrates practical applicability, effective feature engineering, and real-world deployment through a web interface.

---

14. Author Details

- **Name:** Sushma
- **Enrollment Number**: 2315149
- **Domain:** Machine Learning and Natural Language Processing
- **Tools & Technologies:** Python, Scikit-learn, NLTK, Streamlit