AUTOJUDGE – INTELLIGENT PROGRAMMING PROBLEM DIFFICULTY ESTIMATION SYSTEM

Overview

Online competitive programming platforms such as Codeforces, CodeChef, and LeetCode organize problems into categories like Easy, Medium, and Hard. These difficulty levels help learners choose suitable challenges, assist contest designers, and measure problem complexity.
However, assigning difficulty manually is often subjective and inconsistent across platforms and authors.

AutoJudge is an AI-driven system that automatically estimates the difficulty of programming problems using Natural Language Processing (NLP) and Machine Learning techniques. By analyzing problem descriptions, the system outputs both a categorical difficulty label and a continuous numerical score, enabling a more objective and detailed assessment.

Project Goals

The primary goals of AutoJudge are:

- To analyze programming problem text using NLP techniques
- To automatically classify problems as Easy, Medium, or Hard
- To compute a numerical difficulty score for finer evaluation
- To extract meaningful textual and numerical features
- To design robust classification and regression models
- To deploy the solution as an interactive web application using Streamlit

---

Dataset Information

The dataset used in this project consists of programming problems collected from various competitive coding platforms. Each record contains:

- Problem statement
- Input specification
- Output specification

- Constraints
- Difficulty category (Easy / Medium / Hard)
- Numerical difficulty score

The dataset combines unstructured text data with structured numerical attributes, making it ideal for hybrid feature extraction.

Data Cleaning and Preprocessing

To improve model performance and reduce noise, several text preprocessing steps were applied:

- Conversion of all text to lowercase
- Removal of special characters and irrelevant symbols
- Tokenization of sentences into individual words
- Elimination of stopwords using NLTK
- Lemmatization using WordNet Lemmatizer

These steps normalize textual data while preserving semantic relevance and reducing dimensional complexity.

Feature Extraction and Engineering

AutoJudge employs a combination of textual and handcrafted numerical features.

Text-Based Features

- TF-IDF vectorization is applied to processed text
- It highlights important terms while minimizing the impact of frequently occurring words

Engineered Numerical Features

To better capture problem complexity, additional numerical features were created:

- Total length of the combined problem text
- Count of mathematical and logical symbols (e.g., +, -, *, /, %, ^)
- Maximum numerical constraint extracted from the problem

- Detection of algorithm-related keywords such as:
    - graph, dynamic programming, tree, greedy, shortest path, recursion, bit manipulation, modulo

These features significantly enhance predictive accuracy.

Model Design

Two separate machine learning models are implemented:

Difficulty Classification

- Random Forest Classifier
- Predicts categorical difficulty (Easy, Medium, Hard)
- Selected for its robustness, interpretability, and ability to handle mixed data types

Difficulty Score Prediction

- Regression-based machine learning model
- Outputs a continuous difficulty score
- Provides finer granularity alongside categorical prediction

---

Training Strategy

- The dataset is divided into training and testing subsets
- Feature selection is performed using Random Forest feature importance
- This step reduces overfitting and enhances generalization
- Hyperparameters are tuned to achieve optimal model performance

---

Performance Evaluation

Different metrics are used to assess model effectiveness.

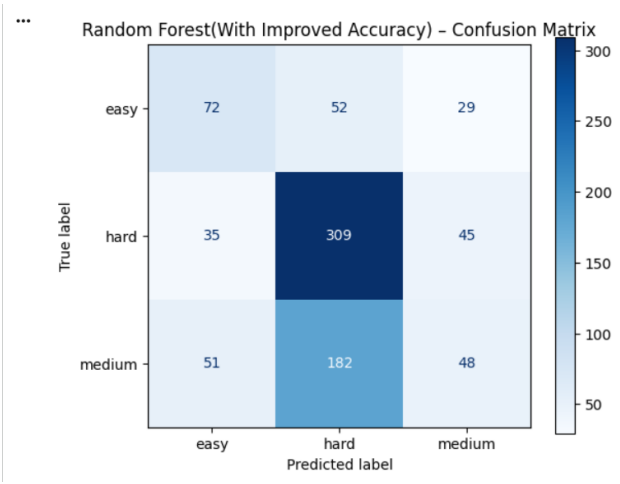
Classification Evaluation

- Accuracy

```
Improved(RF) 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
```



Regression Evaluation

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

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

These metrics ensure both clarity and precision in performance measurement.

Web Application Deployment

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

Application Workflow

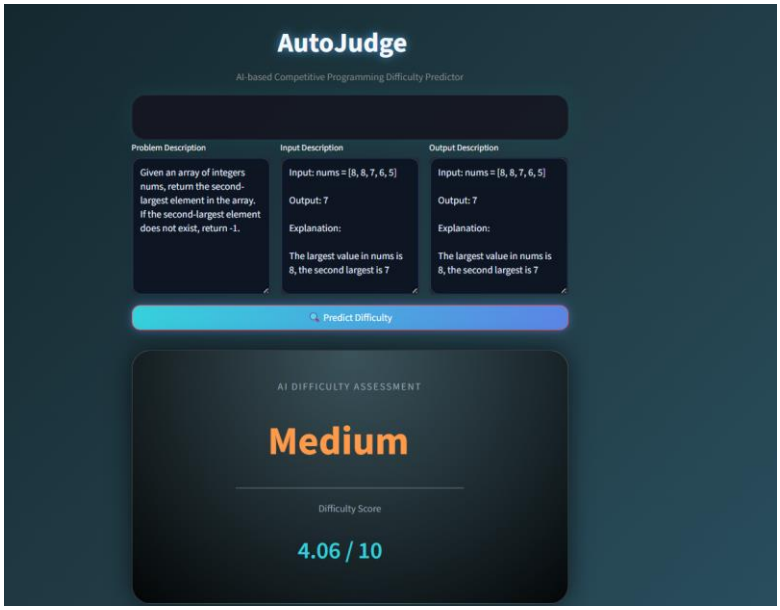
1. User inputs:

- Problem description
- Input details
- Output details

2. Text is cleaned and preprocessed
3. Feature extraction is performed
4. Models generate predictions
5. Results are displayed:
    - Predicted difficulty level
    - Estimated difficulty score

Interface Highlights

- Instant predictions
- Visual indicators representing problem difficulty



Observations and Analysis

The system demonstrates reliable performance in both classification and regression tasks. Feature engineering plays a critical role in improving accuracy. The numerical difficulty score adds depth and insight beyond simple categorical labels.

Constraints and Challenges

- Model performance depends heavily on dataset quality
- Ambiguous or poorly structured problems may lead to incorrect predictions
- Training data is limited to English, introducing potential language bias

---

Future Scope

Potential enhancements include:

- Incorporating transformer-based architectures such as BERT
- Supporting multilingual problem descriptions
- Direct integration with competitive programming platforms
- Automated expansion of the training dataset

Final Summary

AutoJudge provides an automated, intelligent solution for predicting programming problem difficulty using NLP and machine learning techniques. The project showcases effective feature engineering, reliable modeling, and practical deployment through a web interface.

Author Information

- **Name:** Ritesh Kumar Ratnakar
- **Enrollment Number:** 23113128

- **Technologies Used:** Python, Scikit-learn, NLTK, Streamlit