AutoJudge — Predicting Programming Problem Difficulty

AANVI JHA (24114002)

Overview:

Online coding platforms such Codeforces, CodeChef assign difficulty labels and scores to programming problems.
These ratings are typically based on human judgment,which can be subjective and slow.

AutoJudge is a machine learning–based system that automatically predicts **Difficulty Class**: Easy / Medium / Hard and **Difficulty Score**: A numerical difficulty score using only the textual content of a problem statement.
The system is trained on a labeled dataset and deployed through an interactive Streamlit web application.

Dataset used- same as that provided already.

https://github.com/AREEG94FAHAD/TaskComplexityEval-24

Methodology:

1. **Data Preprocessing**
   Combined all textual fields into a single full_text feature.
   Handled missing values by replacing them with empty strings
2. **Feature extraction:**
   To convert text into numerical features:
   - TF-IDF Vectorization: Captures important words and phrase, using unigrams and bigrams
   - Hand-crafted features: Text length, Count of mathematical symbols (+-*/=<>)

   These features are concatenated to form the final input vector.

3. **Model — Classification:**

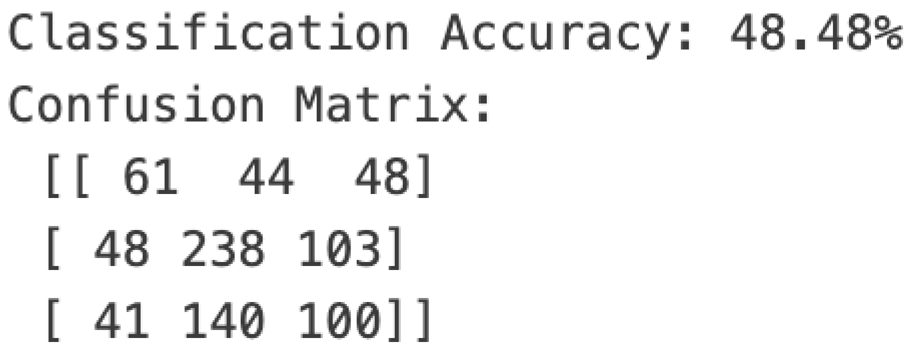
Support Vector Machine (LinearSVC) is used to predict the 3 difficulty classes- Easy, Medium and Hard. SVM was chosen as it performs well on high-dimensional text data and is robust for classification tasks.

Evaluation Metrics

- Accuracy

achieved - 48.48%

- Confusion Matrix



```
Classification Accuracy: 48.48%
Confusion Matrix:
 [[ 61  44  48]
 [ 48 238 103]
 [ 41 140 100]]
```

4. **Model- Regression:**

Random Forest Regressor is used to predict numerical difficulty score.
It captures non-linear relationships and is robust to noisy and subjective labels

Evaluation Metrics

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

Results

Achieved a classification **accuracy of 48.48** % and **MAE ~1.7, RSME~2.04.**

This shows reasonable MAE and RMSE given subjective difficulty scores.

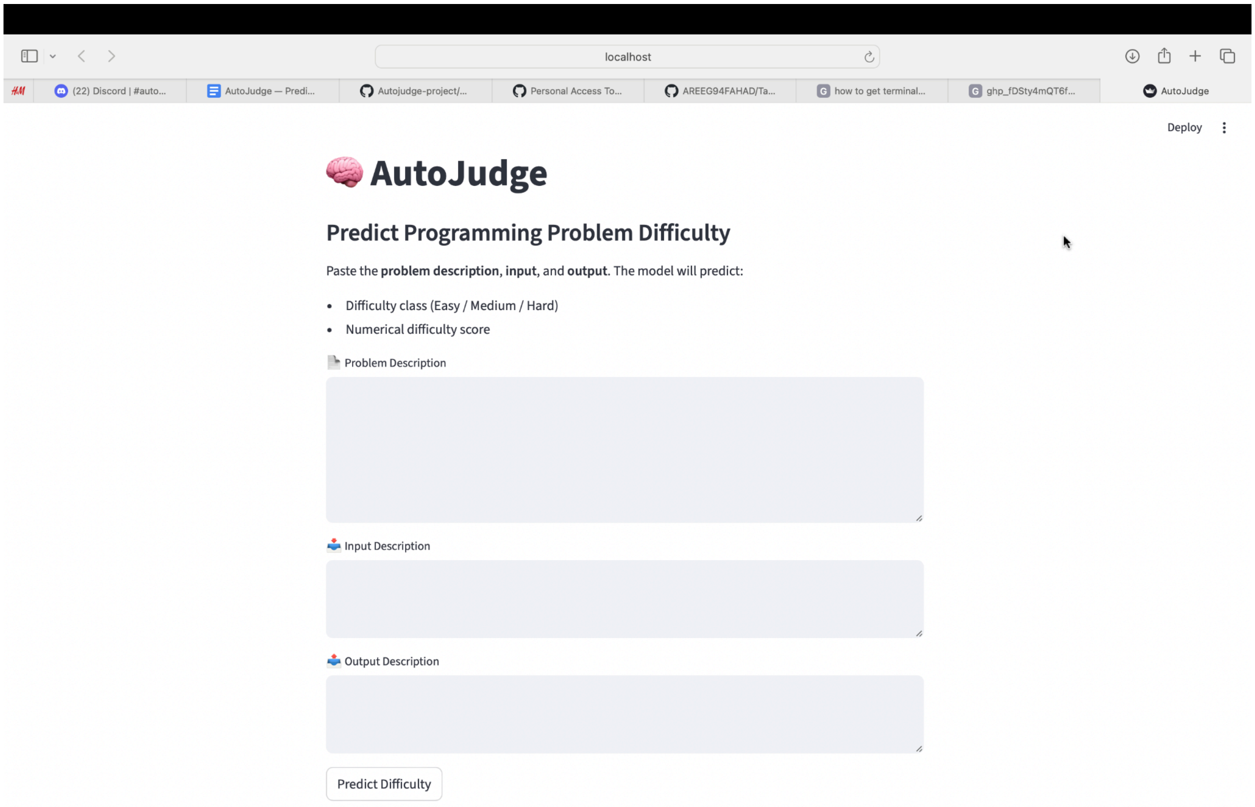
Deployment

The trained models are deployed using **Streamlit**, allowing users to:

1. Paste a new problem description
2. Click predict
3. Instantly see predicted difficulty class and predicted difficulty score

Steps to run the program-

1. Clone the github repo on your terminal
2. Create new virtual environment.
3. Ensure that all the packages on requirements.txt are installed.
4. Run:- streamlit run app.py
5. You will be directed to the streamlit Web UI.
6. Insert the problem whose difficulty and score you want to know.
7. Give proper input and output description
8. Click on the “Predict Difficulty” button to know the results.



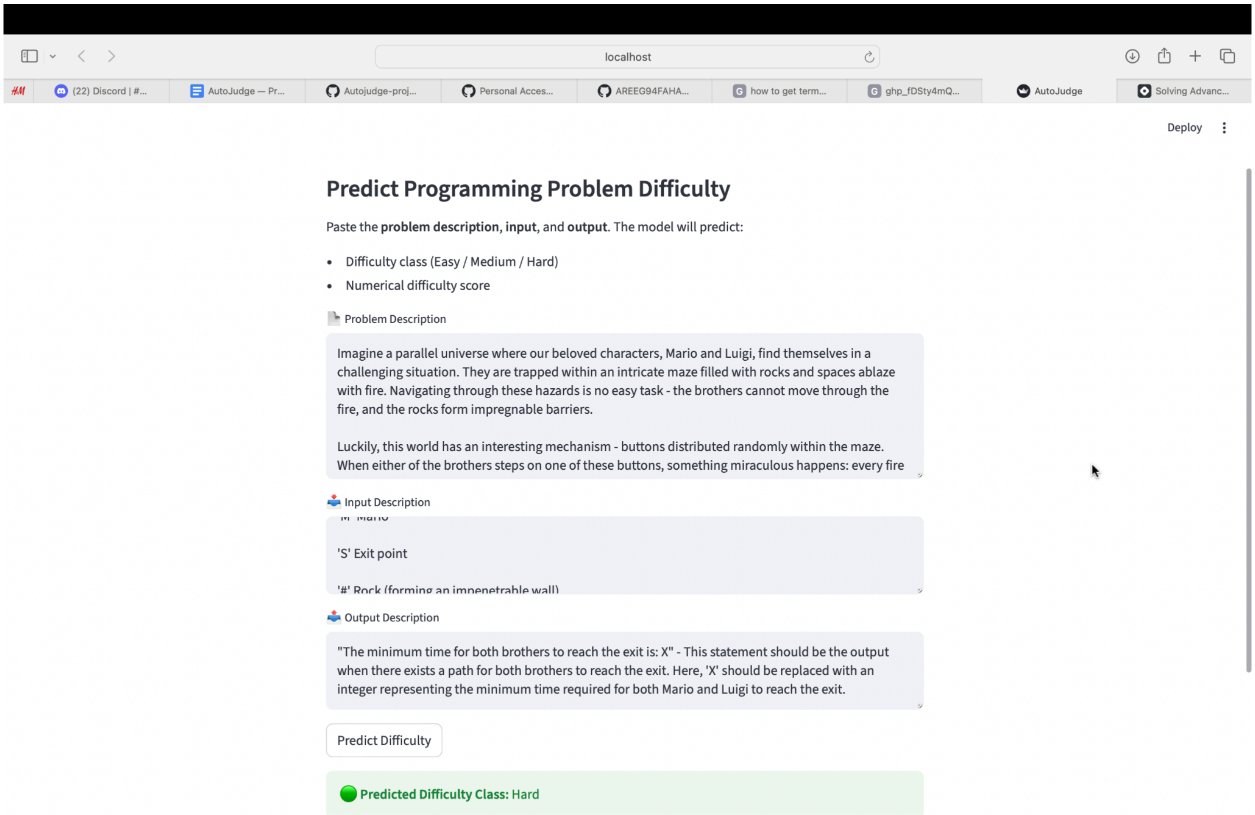
🧠 AutoJudge

Predict Programming Problem Difficulty

Paste the **problem description**, **input**, and **output**. The model will predict:

- Difficulty class (Easy / Medium / Hard)
- Numerical difficulty score

Here is how the web interface looks like.



Predict Programming Problem Difficulty

Paste the **problem description**, **input**, and **output**. The model will predict:

- Difficulty class (Easy / Medium / Hard)
- Numerical difficulty score

Problem Description

Imagine a parallel universe where our beloved characters, Mario and Luigi, find themselves in a challenging situation. They are trapped within an intricate maze filled with rocks and spaces ablaze with fire. Navigating through these hazards is no easy task - the brothers cannot move through the fire, and the rocks form impregnable barriers.

Luckily, this world has an interesting mechanism - buttons distributed randomly within the maze. When either of the brothers steps on one of these buttons, something miraculous happens: every fire

Input Description

'M' Mario

'S' Exit point

'#' Rock (forming an impenetrable wall)

Output Description

"The minimum time for both brothers to reach the exit is: X" - This statement should be the output when there exists a path for both brothers to reach the exit. Here, 'X' should be replaced with an integer representing the minimum time required for both Mario and Luigi to reach the exit.

● **Predicted Difficulty Class:** Hard

Here is the result for a problem classified as hard in the dataset.

CONCLUSION:

AutoJudge provides a practical, end-to-end pipeline, from data preprocessing and model training to deployment via a Streamlit-based web interface highlighting how machine learning models can be transitioned from experimentation to real-world applications. This system can help students in selecting problems suited to their skill level and support platforms in organizing and tagging large collections of programming problems.

Overall, the project illustrates how machine learning can be effectively applied to software engineering and education-oriented domains, while also emphasizing challenges such as data quality, feature engineering, and model generalization.

DEMO VIDEO LINK- https://youtu.be/QmiGmmTMetw