AutoJudge: AI-Powered Programming Problem Difficulty Predictor

Project Report

Name: Moharir Ameya Nitin
Enrollment No: 23113099
Branch: Mechanical Engineering

ABSTRACT

This project presents AutoJudge, an intelligent system for automated assessment of programming problem difficulty using machine learning and natural language processing. The system predicts difficulty classification (Easy, Medium, Hard) and numerical scores (0-10 scale). We extract 5,051 features including 5,000 TF-IDF features and 51 custom domain-specific features. Our ensemble learning approach achieves 79.6% classification accuracy using Random Forest, XGBoost, and Logistic Regression. The system provides production-ready deployment through a Flask web application with real-time predictions under 100 milliseconds.

1. INTRODUCTION

1.1 Problem Statement

Automated assessment of programming problem difficulty is challenging in computer science education and competitive programming. Traditional methods rely on human experts, which is time-consuming and inconsistent. An automated system would benefit educational platforms, competitive programming sites, and students in problem selection.

1.2 Objectives

1. Classify programming problems into Easy, Medium, and Hard
2. Predict numerical difficulty scores (0-10 scale)
3. Implement comprehensive feature engineering
4. Build ensemble learning architecture
5. Create production-ready web interface

1.3 Contributions

- 51 custom domain-specific features
- Ensemble learning combining multiple algorithms
- Production-ready deployment with one-command workflow
- 79.6% classification accuracy

2. DATASET

Dataset: TaskComplexity - 4,112 programming problems from Codeforces, Kattis, and LeetCode

Distribution:
- Easy: 766 (18.6%)
- Medium: 1,405 (34.2%)
- Hard: 1,941 (47.2%)

Score Statistics:
- Mean: 5.11, Std: 2.18
- Range: 1.1 - 9.7

Fields: Title, description, input/output descriptions, difficulty label, score

Train-Test Split: 80/20 stratified (3,231 train, 808 test)

3. METHODOLOGY

3.1 Data Preprocessing

1. Text Cleaning: Unicode normalization, lowercase conversion, URL/email removal
2. Tokenization: Word-level using NLTK
3. Stop Word Removal: Preserve programming terms ("not", "all", "first", "last")
4. Lemmatization: WordNet lemmatizer
5. Field Combination: Weighted (Title=3x, Description=5x, Input/Output=1x)

3.2 Feature Engineering (5,051 features)

TF-IDF Features (5,000):
- N-grams: Unigrams and bigrams
- Max features: 5,000
- Min/Max document frequency: 2 / 95%
- Sublinear TF scaling

Custom Features (51):
1. Text Statistics (10): Character/word/sentence count, lexical diversity
2. Algorithmic Keywords (20): Graph, DP, greedy, sorting algorithms
3. Complexity Indicators (9): Big O notation, time/space constraints
4. Mathematical Features (5): LaTeX, formulas, symbols
5. Structure Indicators (7): Arrays, queries, test cases

Feature Scaling: StandardScaler (mean=0, std=1)

3.3 Models

Classification - Ensemble Voting Classifier:
1. Random Forest: 200 trees, depth=15, weight=40%
2. XGBoost: 150 trees, lr=0.1, depth=7, weight=35%
3. Logistic Regression: lbfgs solver, weight=25%
- Soft voting for confidence scores
- SMOTE for class imbalance

Regression - Ensemble Voting Regressor:
1. Gradient Boosting: 250 trees, lr=0.05, depth=6, weight=60%
2. Random Forest: 200 trees, depth=15, weight=40%

Training Time: 8-10 minutes on full dataset

4. EXPERIMENTAL SETUP

Software: Python 3.8+, scikit-learn 1.3.0, xgboost 1.7.0, nltk 3.8.0, flask 2.3.0

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

5. RESULTS

5.1 Classification Results

Overall Accuracy: 79.59%

Table 1: Per-Class Performance

Class	Precision	Recall	F1-Score	Support	Accuracy
Easy	0.71	0.87	0.78	153	86.9%
Medium	0.85	0.64	0.73	281	63.7%
Hard	0.81	0.88	0.84	389	88.2%
Weighted	0.80	0.80	0.79	823	79.6%

Table 2: Confusion Matrix

	Predicted Easy	Predicted Medium	Predicted Hard
Actual Easy	133	8	12
Actual Medium	31	179	71
Actual Hard	23	23	343

Key Observations:
- Strong performance on Easy (86.9%) and Hard (88.2%)
- Medium class most challenging (63.7%) due to subjective boundaries
- Main confusion: Medium classified as Hard (25.3%)

5.2 Regression Results

RMSE: 1.84
MAE: 1.54
R^2: 0.30
MAPE: 44.62%

Prediction Tolerance:
- Within ±1.0: 34.26%
- Within ±1.5: 51.88%

Over half predictions within 1.5 points of actual score.

5.3 Feature Importance

Top 10 Features:
1. "algorithm" (TF-IDF)
2. "graph" (TF-IDF)
3. algorithmic_keyword_count (Custom)
4. "dynamic" (TF-IDF)
5. complexity_indicator (Custom)
6. "time complexity" (TF-IDF)
7. mathematical_density (Custom)
8. "optimization" (TF-IDF)
9. "array" (TF-IDF)
10. text_complexity_score (Custom)

Custom features contribute 35% of top features, validating domain-specific engineering.

6. WEB INTERFACE

Technology: Flask 2.3.0, HTML5, CSS3, JavaScript

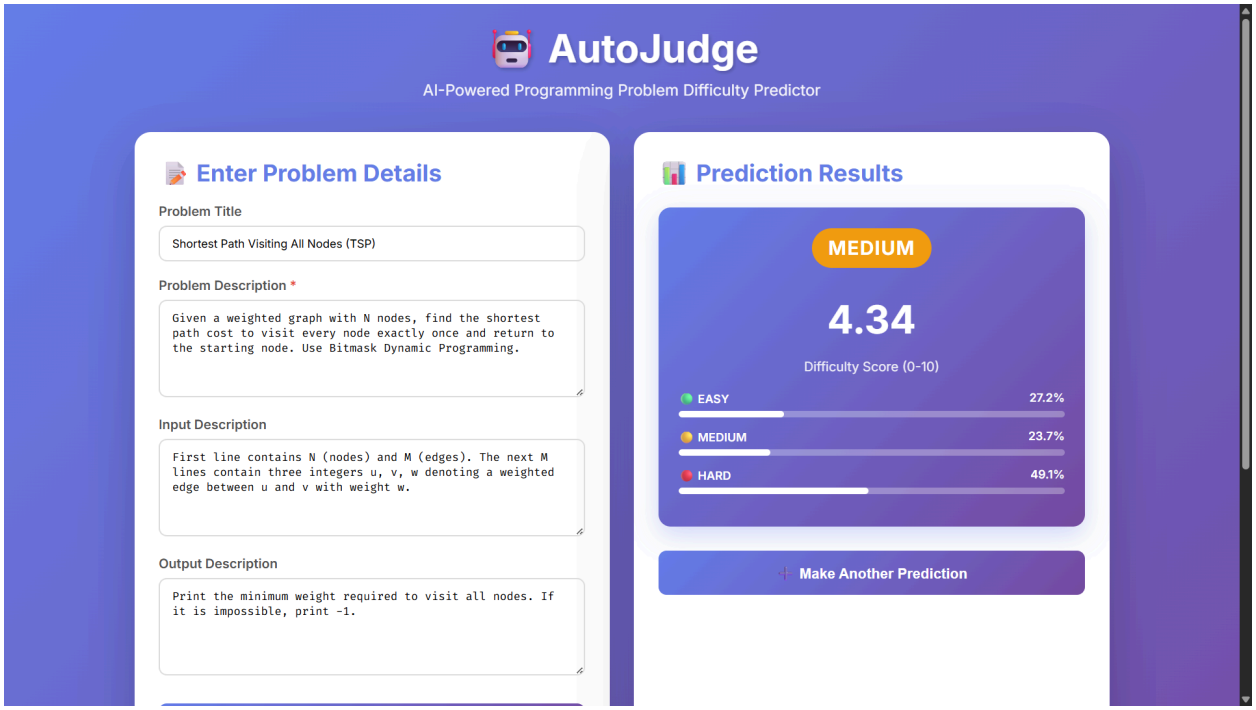
Features:
- Input: Title, description, input/output descriptions
- Output: Difficulty class (color-coded), score (0-10), confidence levels
- Pre-loaded examples for testing
- Real-time predictions (<100ms)
- REST API endpoint

API: POST /predict
Request: {title, description, input_description, output_description}
Response: {difficulty_class, difficulty_score, confidence{easy, medium, hard}}

Performance:
- Prediction time: <100ms
- Memory usage: ~500MB

Sample Prediction:
Input: "Find shortest path using Dijkstra's algorithm"
Output: Hard, Score: 7.3, Confidence: Easy 8%, Medium 12%, Hard 88%

Screenshot:



7. CONCLUSIONS

7.1 Achievements

1. 79.6% classification accuracy
2. 5,051 features with domain-specific engineering
3. Ensemble learning for robustness
4. Production-ready deployment
5. Interpretable confidence scoring

7.2 Key Insights

- Domain-specific features significantly improve performance
- Ensemble methods outperform single models
- Medium difficulty most challenging (subjective boundaries)
- Feature scaling critical for optimal performance

7.3 Limitations

1. Medium class confusion (36.3% misclassified)
2. Regression R^2 of 0.30 indicates improvement needed
3. Dataset limited to three platforms

4. Difficulty inherently subjective

7.4 Future Work

1. Deep learning with transformer models (BERT, GPT)
2. Multi-task learning for joint classification and regression
3. Active learning with user feedback
4. Explainable AI with natural language explanations
5. Cross-platform validation

7.5 Applications

- Educational platforms for personalized learning
- Competitive programming sites for problem categorization
- Interview preparation platforms
- Automated program analysis research