Jigsaw - Agile Community Rules Classification: Implementation Plan

Team of Five

July 26, 2025

Overview

The Jigsaw - Agile Community Rules Classification competition on Kaggle challenges participants to build a binary classifier to predict whether a Reddit comment violates a specific subreddit rule. The dataset includes moderated comments with associated rules, subreddits, and binary labels. The training data covers two rules, while the test data includes additional unseen rules, requiring models to generalize effectively. Submissions are evaluated using column-averaged AUC, and predictions must be probabilities of rule violations. This document outlines a detailed implementation plan, task division for a team of five, resources required, and machine learning models to explore.

1 Implementation Plan

The implementation plan is structured into five phases: Data Exploration, Preprocessing, Model Development, Evaluation, and Submission. Each phase includes specific tasks, assigned team members, and timelines to ensure efficient progress toward the competition deadline of October 23, 2025.

1.1 Phase 1: Data Exploration (July 26 - August 2, 2025)

- **Objective**: Understand the dataset structure, rules, and patterns to inform preprocessing and modeling strategies.
- **Tasks**:
  - Download and inspect `train.csv`, `test.csv`, and `sample_submission.csv`.
  - Analyze the distribution of `rule_violation` labels in `train.csv`.
  - Examine `subreddit`, `rule`, `positive_example_1`, `positive_example_2`, `negative_example_1`, and `negative_example_2` for context.
  - Perform exploratory data analysis (EDA) to identify text patterns, rule-specific keywords, and subreddit-specific norms.
  - Investigate potential data imbalances and missing values.

- **Assigned Team Members**:
  - Member 1: Lead EDA, visualize label and subreddit distributions.
  - Member 2: Analyze rule-specific patterns and example comments.
- **Deliverables**:
  - EDA notebook with visualizations (e.g., histograms, word clouds).
  - Report summarizing key findings (e.g., class imbalance, rule complexity).
- **Timeline**: 7 days (July 26 - August 2, 2025).

1.2 Phase 2: Data Preprocessing (August 3 - August 10, 2025)

- **Objective**: Clean and transform the dataset for model training, addressing text-specific challenges and unseen rules.
- **Tasks**:
  - Clean comment text (body) by removing special characters, URLs, and irrelevant symbols.
  - Apply text preprocessing: tokenization, lowercasing, stop-word removal, and lemmatization using libraries like NLTK or spaCy.
  - Create features from `subreddit` and `rule` (e.g., one-hot encoding or embeddings).
  - Incorporate `positive_example_1`, `positive_example_2`, `negative_example_1` and `negative_example_2` for context-aware feature engineering.
  - Develop a strategy to handle unseen rules in `test.csv` (e.g., rule embeddings or meta-learning).
  - Split `train.csv` into training (80%) and validation (20%) sets for local evaluation.
- **Assigned Team Members**:
  - Member 3: Lead text preprocessing and feature engineering.
  - Member 4: Develop rule and subreddit embeddings.
- **Deliverables**:
  - Preprocessing pipeline in a Kaggle Notebook.
  - Processed training and validation datasets.
- **Timeline**: 7 days (August 3 - August 10, 2025).

1.3 Phase 3: Model Development (August 11 - September 15, 2025)

- **Objective**: Build and train multiple machine learning models to predict rule violations, focusing on generalization to unseen rules.
- **Tasks**:
  - Implement baseline models (e.g., Logistic Regression, TF-IDF).
  - Develop advanced NLP models (e.g., BERT, RoBERTa, DistilBERT).
  - Experiment with ensemble methods to combine model predictions.
  - Use few-shot learning or meta-learning to handle unseen rules.
  - Optimize hyperparameters using grid search or random search.
  - Validate models locally using AUC on the validation set.
- **Assigned Team Members**:
  - Member 1: Implement baseline models.
  - Member 2: Develop transformer-based models (BERT, RoBERTa).
  - Member 3: Experiment with few-shot learning and meta-learning.
  - Member 4: Build ensemble models.
  - Member 5: Lead hyperparameter tuning and model validation.
- **Deliverables**:
  - Multiple trained models in Kaggle Notebooks.
  - Validation AUC scores for each model.
  - Ensemble model combining top performers.
- **Timeline**: 35 days (August 11 - September 15, 2025).

1.4 Phase 4: Evaluation and Optimization (September 16 - October 10, 2025)

- **Objective**: Evaluate and refine models to maximize AUC on the public test set and ensure robustness.
- **Tasks**:
  - Submit predictions to Kaggle's public leaderboard.
  - Analyze public leaderboard feedback to identify model weaknesses.
  - Fine-tune models based on leaderboard performance.
  - Implement error analysis to understand false positives/negatives.
  - Test model robustness on unseen rules using synthetic data or cross-validation.

- **Assigned Team Members**:
  - Member 5: Lead evaluation and submission process.
  - Member 1 and 2: Conduct error analysis.
  - Member 3 and 4: Fine-tune models and test robustness.
- **Deliverables**:
  - Optimized models with improved AUC.
  - Error analysis report.
  - Final submission notebook.
- **Timeline**: 25 days (September 16 - October 10, 2025).

1.5 Phase 5: Final Submission (October 11 - October 23, 2025)

- **Objective**: Prepare and submit the final model predictions to Kaggle.
- **Tasks**:
  - Generate predictions on `test.csv` using the best model or ensemble.
  - Format predictions as per `sample_submission.csv` (columns: `row_id`, `rule_violation`).
  - Verify submission file format and content.
  - Submit final predictions to Kaggle.
  - Document the final approach and results.
- **Assigned Team Members**:
  - Member 5: Lead final submission and documentation.
  - All Members: Review submission file and documentation.
- **Deliverables**:
  - Final `submission.csv` file.
  - Comprehensive project report.
- **Timeline**: 13 days (October 11 - October 23, 2025).

2 Resources Required

- **Computational Resources**:
  - Kaggle Notebooks (CPU and GPU, limited to 12 hours run-time each).
  - Optional: Local machines or cloud instances (e.g., Google Colab, AWS) for experimentation beyond Kaggle limits.
- **Software and Libraries**:

  - Python 3.x, Jupyter Notebooks.
  - Data processing: `pandas`, `numpy`.
  - NLP: `nltk`, `spaCy`, `transformers` (Hugging Face), `scikit-learn`.
  - Deep learning: `tensorflow`, `pytorch`.
  - Visualization: `matplotlib`, `seaborn`, `wordcloud`.
- **Datasets**:
  - `train.csv`, `test.csv`, `sample_submission.csv` from Kaggle.
  - External datasets: Pre-trained models (e.g., BERT, RoBERTa) from Hugging Face.
  - Optional: Additional Reddit datasets (e.g., from Pushshift, if permitted) for context.
- **Team Resources**:
  - Collaboration tools: GitHub for version control, Slack/Discord for communication.
  - Documentation tools: Google Docs, LaTeX for report generation.

3 Machine Learning Models

The following models are recommended, balancing simplicity, performance, and generalization to unseen rules.

3.1 Baseline Models

- **Logistic Regression with TF-IDF**:
  - Convert comment text to TF-IDF vectors using `scikit-learn`.
  - Train a logistic regression model to predict `rule_violation`.
  - Pros: Simple, interpretable, fast to train.
  - Cons: Limited ability to capture complex text patterns.
- **Naive Bayes with Bag-of-Words**:
  - Use bag-of-words representation for comment text.
  - Train a Multinomial Naive Bayes classifier.
  - Pros: Computationally efficient, good for text classification.
  - Cons: Assumes word independence, may struggle with nuanced rules.

3.2 Advanced NLP Models

- **BERT (Bidirectional Encoder Representations from Transformers)**:
  - Use pre-trained BERT (e.g., `bert-base-uncased`) from Hugging Face.
  - Fine-tune on `train.csv` with `subreddit` and `rule` as additional inputs.
  - Pros: Captures contextual relationships, strong performance on text tasks.
  - Cons: Computationally intensive, requires GPU.
- **RoBERTa (Robustly optimized BERT approach)**:
  - Similar to BERT but optimized for robustness.
  - Fine-tune with rule-specific context embeddings.
  - Pros: Improved performance over BERT, handles nuanced text.
  - Cons: High computational cost.
- **DistilBERT**:
  - Lightweight version of BERT with fewer parameters.
  - Suitable for Kaggle's resource constraints.
  - Pros: Faster training, lower memory usage.
  - Cons: Slightly lower accuracy than BERT/RoBERTa.

3.3 Few-Shot Learning and Meta-Learning

- **Few-Shot Learning with Prototypical Networks**:
  - Use `positive_example_1`, `positive_example_2`, `negative_example_1`, and `negative_example_2` to create prototypes for each rule.
  - Classify comments based on distance to prototypes.
  - Pros: Effective for unseen rules with limited examples.
  - Cons: Requires careful design of prototype embeddings.
- **Meta-Learning (MAML - Model-Agnostic Meta-Learning)**:
  - Train a model to adapt quickly to new rules using few examples.
  - Use `train.csv` to simulate meta-learning tasks.
  - Pros: Strong generalization to unseen rules.
  - Cons: Complex implementation, high computational cost.

3.4 Ensemble Models

- Combine predictions from multiple models (e.g., BERT, RoBERTa, Logistic Regression) using weighted averaging or stacking.
- Use cross-validation to determine optimal weights.
- Pros: Improves robustness and AUC by leveraging model diversity.
- Cons: Increases complexity and submission time.

4 Task Division for Team of Five

- **Member 1 (EDA and Baseline Models)**:
  - Lead EDA in Phase 1.
  - Develop baseline models (Logistic Regression, Naive Bayes) in Phase 3.
  - Conduct error analysis in Phase 4.
- **Member 2 (Rule Analysis and Transformers)**:
  - Analyze rule-specific patterns in Phase 1.
  - Implement BERT and RoBERTa models in Phase 3.
  - Assist with error analysis in Phase 4.
- **Member 3 (Preprocessing and Few-Shot Learning)**:
  - Lead text preprocessing and feature engineering in Phase 2.
  - Develop few-shot learning models in Phase 3.
  - Fine-tune models in Phase 4.
- **Member 4 (Embeddings and Ensembles)**:
  - Develop rule and subreddit embeddings in Phase 2.
  - Build ensemble models in Phase 3.
  - Test model robustness in Phase 4.
- **Member 5 (Evaluation and Submission)**:
  - Lead hyperparameter tuning and validation in Phase 3.
  - Manage evaluation and submissions in Phase 4.
  - Oversee final submission and documentation in Phase 5.

5 Risks and Mitigation

- **Risk**: Models fail to generalize to unseen rules.

  - **Mitigation**: Use few-shot learning, meta-learning, and rule embeddings to improve generalization.

- **Risk**: Kaggle Notebook runtime limits (12 hours CPU/GPU).
  - **Mitigation**: Optimize code, use DistilBERT for efficiency, and test on local machines if needed.

- **Risk**: Data imbalances affect model performance.
  - **Mitigation**: Apply class weighting or oversampling during training.

- **Risk**: Team coordination issues.
  - **Mitigation**: Use GitHub for code sharing, Slack for communication, and weekly check-ins.

6 Conclusion

This implementation plan provides a structured approach to tackle the Jigsaw - Agile Community Rules Classification competition. By dividing tasks among five team members, leveraging a mix of baseline and advanced NLP models, and addressing generalization to unseen rules, the team can maximize its chances of achieving a high AUC score. The plan adheres to Kaggle's constraints and timelines, ensuring timely submission by October 23, 2025.