

LLM-as-a-Judge: By-Peril Model Evaluation

Colin Gibbons-Fly

Objective

This framework simulates real-world deployment conditions by leveraging a generative AI “Judge” to determine which peril classification (Current System vs. By-Peril Model) is more accurate based on unstructured claim data.

- **Business Goal:** Improve peril specificity for enhanced pricing segmentation.
- **Research Goal:** Evaluate model performance without relying on human labels (which won’t exist in production) by using a robust LLM-based surrogate judge trained on insurance reasoning.

System Architecture

Core Components

Component	Purpose
PredictiveModelPerilJudge	LLM-based evaluator that picks the best peril label for a claim
ModelPerformanceAnalytics	Tracks performance across evaluations
load_and_prepare_data()	Normalizes input CSV into expected schema
run_comprehensive_evaluation()	Runs judge on a batch of claims
generate_final_report()	Provides summary statistics and recommendations
run_full_evaluation_first_n_rows()	Main entry point (used in production/test runs)

Judge Design

PredictiveModelPerilJudge

- **Model:** Claude 3.5 Sonnet via AWS Bedrock
- **Prompt Style:** Chain-of-Thought (CoT) + Domain-Specific Instructions
- **Inputs:**
 - DESCRIPTION, NOTES_SUMMARY: Claim narrative
 - CURRENT_PERIL_TYPE: Legacy label
 - MODEL_PERIL_TYPE: Prediction from By Peril model
 - PERIL_AGENT_1, PERIL_AGENT_2, AGENT_COMMENTS: Used for validation only

Decision Options

- "MODEL" — Model classification more accurate
- "CURRENT" — Current label more accurate
- "BOTH CORRECT" — Both are valid
- "NEITHER ADEQUATE" — Both are flawed or unclear

Threshold Logic

If confidence < 7.0, decision is overridden to "NEITHER ADEQUATE" and flagged accordingly.

Prompt Structure

Each prompt sent to the LLM includes:

1. **System Role Prompt**
Describes a senior claims adjuster with 15+ years of experience.
2. **Peril Type Definitions**
Includes key indicators for Wind, Hail, Water (sudden vs. gradual), Theft, Vandalism, etc.
3. **Chain-of-Thought Prompt**
Requires step-by-step reasoning:

- Evaluate damage pattern
- Assess peril coverage fit
- Calibrate confidence
- Final decision as JSON

4. Claim Evidence Section

Injects specific claim text and model vs. current labels. Agent assessments are included for context only.

Data Preprocessing

```
def load_and_prepare_data(csv_file_path, num_rows=3):
    # Loads CSV and maps columns to expected format for evaluation
    # Includes mapping logic for flexible production use
```

Evaluation Process

`run_comprehensive_evaluation(sample_data)`

- Iterates over each claim, calling `judge_single_claim()`
- Logs structured reasoning and confidence
- Records outcomes as one of the four decisions
- Adds all results to `ModelPerformanceAnalytics`

ModelPerformanceAnalytics

Tracks: - Agreement patterns by peril - Confidence scores by decision type - System success rates

Output Metrics

From `generate_final_report()`

- Counts of:
 - Model wins
 - Current wins
 - Both correct
 - Neither adequate
- Confidence statistics:
 - Mean, Std. Dev.
 - High-confidence thresholds
- Agreement metrics:
 - System agreement rate (both correct + neither adequate)
 - Success rate (at least one correct)
- Recommendations for:
 - Production deployment
 - Retraining strategy
 - Domain complexity insights

Results Storage

Results are saved as:

```
evaluation_results_first_n.json
```

Includes: - `claim_id` - `winner` - `confidence` - `reasoning` - `justification` - `processing_time`
- `model_vs_current`

Production Considerations

- In production, agent labels will not be available.
- The prompt is designed to work independently of `PERIL_AGENT_1` and `AGENT_COMMENTS`.
- The mapping function allows for loose column naming, increasing flexibility.