Introduction

This project builds an Automated AI Red-Teaming System that generates adversarial prompts, attacks multiple LLMs, and evaluates model safety using a dedicated Judge LLM. Since the system does not train any ML model, it only orchestrates API calls to external LLMs; traditional ML components, such as hyperparameter tuning, MLflow-style experiment tracking, and model sensitivity analysis, do not apply. The pipeline focuses on evaluating model safety behavior, not optimizing or training predictive models.

1. Hyperparameter Tuning

Hyperparameter tuning is relevant when:

- A model is trained locally
- The model's behavior depends on adjustable hyperparameters
- Techniques such as grid search, random search, or Bayesian optimization can improve performance

Why It Does Not Apply Here

- The project does not train any machine learning model.
- All LLMs (Llama-3, Moonshot, MiniMax, etc.) are pretrained and hosted externally via APIs.
- Their internal hyperparameters or weights are not accessible and cannot be tuned.
- The goal is to *evaluate* model behavior, not *optimize* it.

Therefore:

Hyperparameter tuning is irrelevant, since there is no training loop, no model weights, and no optimization objective involved.

2. Experiment Tracking & Model Selection

Traditional experiment tracking includes:

- Logging training runs
- Tracking loss curves, accuracy curves
- Logging model checkpoints
- Comparing trained models

Why It Does Not Apply

- There are no training experiments.
- There are no model versions generated inside the project.

- There are no confusion matrices or learning curves because no model is being trained.

What *is* applicable

The project *does* track:

- Unsafe response rates
- Refusal behavior
- Violation categories
- Per-model judgement CSV files

This is not ML experiment tracking, but operational tracking of safety outcomes.

Therefore:

Only lightweight evaluation tracking applies—not MLflow/W&B experiment tracking for model training.

3. Model Sensitivity Analysis

Sensitivity analysis tools like SHAP and LIME require:

- A trained model
- A feature-based input vector
- A need to measure feature importance or model explainability

Why It Does Not Apply

- LLMs are accessed through text prompts and not feature vectors.
- No supervised learning model is trained within the project.
- SHAP/LIME cannot be applied to closed-weight LLM APIs without:
    - Access to internal model gradients
    - Access to the model's token-level logits
    - A feature-engineering pipeline

None of these conditions is met.

Therefore:

Sensitivity analysis is not applicable because the system deals with prompt-response evaluation, not model explainability.

4. CI/CD for Model Training

Typical CI/CD pipelines for machine learning are designed to automate:

- Model training
- Model validation
- Bias detection on trained models
- Deployment of newly trained model versions

Why It Does Not Apply Here

- No model is being trained in this project.
  All LLMs (Llama-3, Moonshot, MiniMax, etc.) are external API models, not models we train, fine-tune, or deploy.
- The system's output is evaluated responses, not a trained model artifact.
- Bias detection in CI/CD applies to trained models, not to third-party LLM responses.
- There is no model registry, no model versioning, and no deployment step needed.

What CI/CD *is* used for here

The only automation required is:

- DAG-based automated preprocessing (Airflow)
- Automated response generation workflows
- Automated Judge LLM scoring

This is pipeline automation, not ML training automation.

Therefore:

CI/CD for model training, validation, bias detection, or model deployment is not applicable, as the project does not involve producing or deploying machine learning models.