**Guidewire-DevTrails**

**Submission for the Guidewire DevTrails Hackathon**

**🔥 Project Title:**

**Reinforcement Learning Enhanced Ensemble for Predictive Issue Classification in Cloud Environments**

**✅ Team CodeBlue:**

* Naidu Yaswanth Reddy
* Srivatsal Parise
* Prajwala Yadlapalli

**✅ 1. Introduction:**

This project introduces a hybrid AI pipeline combining multiple machine learning models and a reinforcement learning (RL) layer to predict and classify potential issues in cloud-native Kubernetes (K8s) environments and infrastructure systems. The RL component aims to optimize decision-making dynamically, enhancing the system’s predictive accuracy beyond traditional methods.

**✅ 2. Problem Statement:**

Cloud infrastructures face critical challenges in detecting issues at various layers:

* Infrastructure-level failures
* Kubernetes (K8s) performance bottlenecks
* K8s network-related failures

Traditional static models struggle with adaptability and system dynamics. Our solution focuses on proactive issue prediction, enabling dynamic decision-making that adjusts to real-time system states.

**✅ 3. Solution Overview:**

Our multi-model architecture consists of:

* Three specialized ML models focusing on distinct issue layers
* A Reinforcement Learning (RL) agent that optimizes final predictions based on model outputs and past learning experiences

**✅ 4. Dataset Highlights:**

* Total records: 12,046
* Balanced using SMOTE: 6023 issue samples and 6023 healthy samples
* Features include system metrics, logs, and performance indicators
* Binary label:
  + 1 = Issue
  + 0 = No Issue

**✅ 5. Model Information:**

| **Model #** | **Model Name** | **Focus Area** | **Algorithm** |
| --- | --- | --- | --- |
| 1 | Infrastructure Issues Model | Hardware, VM, and infrastructure issues | LightGBM |
| 2 | K8s Performance Bottleneck Model | Pod failures, resource limits, scaling issues | XGBoost |
| 3 | K8s Network Issue Model | Packet loss, throttling, DNS failures | DecisionTree |

**✅ 6. Methodology:**

* Data preparation involved feature extraction, balancing using SMOTE, normalization, and scaling.
* Individual model training was performed, followed by an ensemble voting mechanism.
* Evaluation metrics used: Accuracy, Precision, Recall, F1-score

**✅ 7. Reinforcement Learning (RL) Summary:**

* **RL Algorithm:** Proximal Policy Optimization (PPO)
* **State Space:** Probability scores from all three models
* **Action Space:** Final prediction (0 - No Issue, 1 - Issue)
* **Reward System:**
  + +1 for correct predictions
  + -1 for wrong predictions
  + Bonus for catching true positives (issues)

**Why RL?**  
RL adds dynamic decision-making, learns complex interdependencies between models, and adapts based on real-time feedback from the system.

**✅ 8. Model Roles & RL Advantages:**

| **Layer** | **Purpose** | **RL Impact** |
| --- | --- | --- |
| Infra Model | Detects hardware, VM failures | RL reduces false positives on noisy metrics |
| K8s Performance Model | Captures CPU/memory bottlenecks | RL finds health-performance correlations |
| K8s Network Model | Detects network degradation, DNS failures | RL learns to filter network noise from true issues |

RL learns the relative importance of each model’s prediction dynamically, ensuring better handling of edge cases compared to static voting.

**✅ 9. Results:**

**Model-wise Performance:**

| **Model** | **Accuracy** | **Precision** | **Recall** | **F1-Score** |
| --- | --- | --- | --- | --- |
| Infrastructure Issues Model | 99.2% | 99.3% | 99.0% | 99.1% |
| K8s Performance Bottleneck Model | 99.3% | 99.5% | 99.2% | 99.3% |
| K8s Network Issue Model | 98.9% | 99.3% | 98.4% | 98.8% |

**Observation:**  
All models exhibit high accuracy and recall, especially post-balancing with SMOTE.

**Reinforcement Learning (RL) Status — In Progress:**

* RL layer is **still under development** but early training runs were conducted.
* Initial PPO results:
  + **Explained Variance:** 0.0002
  + **Mean Accuracy:** ~49.5% (expected during exploration phase)
  + **Critic Loss:** Significant fluctuations observed (normal in early RL training)

**Taste of RL Behavior:**

* RL is currently exploring which model to trust in different situations:
  + Leans on the Infrastructure Model during CPU load spikes
  + Questions K8s Performance Model when network jitter overlaps resource spikes
  + Experiments balancing between catching issues and minimizing false positives

**RL’s Next Phase:**

* The RL agent will continue training to learn **context-aware predictions**
* Will eventually replace static voting with **dynamic, system state-driven decision-making**

**✅ 10. Analysis:**

* RL’s initial performance is near random as it explores the action space.
* Class imbalance effectively handled with SMOTE
* RL demonstrates potential to capture complex relationships and interdependencies missed by static models.

**✅ 11. Future Scope (Phase 2):**

**Planned Improvements:**

* Hyperparameter tuning for LightGBM and XGBoost
* Enhanced reward function design (considering SLA penalties and latency)
* Use advanced RL libraries like Gymnasium and RLlib for scalable training
* Dataset enrichment using real-world Kubernetes production logs

**Phase 2 Additions:**

**Retrieval-Augmented Generation (RAG) Layer:**

* Augments model predictions by fetching context-specific knowledge

**Automated Remediation Engine:**

* Suggests and triggers potential fixes based on detected issue:
  + Pod scaling
  + Node restarts
  + Network configuration adjustments

**✅ 12. How to Run:**

1. Clone the repository:

git clone https://github.com/your-repo-name.git

1. Install dependencies:

pip install -r requirements.txt

1. Run the main notebook:

python devtrails.ipynb

**✅ 13. Tech Stack:**

* Python 3.x
* LightGBM, XGBoost, Decision Trees
* Scikit-learn
* Stable Baselines3 (for RL)
* OpenAI Gym / Gymnasium
* Pandas, NumPy
* Kubernetes monitoring data (mock or real)

**✅ Takeaway:**

The Reinforcement Learning-enhanced ensemble is designed to move beyond traditional static models, enabling real-time, context-aware predictions in cloud-native environments. While RL training is still in progress, the early framework shows promise in learning dynamic interdependencies and optimizing predictions — laying the groundwork for proactive issue detection and remediation in Kubernetes clusters.

*Thank you! Feel free to connect with us for future collaborations or enhancements!*