GUIDEWIRE DEVTRAILS

Phase 1 - AI/ML Model for Predicting Kubernetes Issue

**TEAM DETAILS**

**Team Name:** ACES

**Participant 1:**

**Name:** Shradha Rai

**Email ID:** shradharai4u@gmail.com

**Phone No. :** 7027354777

**Participant 2:**

**Name:** Amanpreet Singh Chadha

**Email ID:** amanpreetsinghchadha28@gmail.com

**Phone No. :** 9695858806

**Participant 3:**

**Name:** Akshita Sahu

**Email ID:** id4akshita.1420@gmail.com

**Phone No. :** 9462474383

**Participant 4:**

**Name:** Navya Pandey

**Email ID:** id4akshita.1420@gmail.com

**Phone No. :** 9643184424

**TABLE OF CONTENTS**

| **Section No.** | **Section Title** | **Subsection Title** |
| --- | --- | --- |
| 1 | **INTRODUCTION** | - |
| 2 | **PROBLEM STATEMENT** | 2.1 The Challenge  2.2 Objective |
| 3 | **APPROACH** | 3.1 Data Collection  3.2 Data Preprocessing  3.3 Feature Engineering |
| 4 | **MODEL SELECTION** | 4.1 Model Evaluated  4.2 Model Comparison |
| 5 | **MODEL TRAINING** | 5.1 Hyperparameter Tuning  5.2 Training Process |
| 6 | **EVALUATION METRICS** | 6.1 Performance Metrics  6.2 Confusion Matrix |
| 7 | **RESULTS AND VISUALIZATION** | 7.1 Correlation Heatmap  7.2 Feature Importance Plot |
| 8 | **LIMITATIONS AND FUTURE IMPROVEMENTS** | 8.1 Limitations  8.2 Future Improvements |
| 9 | **CONCLUSION** | - |

**KUBERNETES FAILURE PREDICTION**

**1. INTRODUCTION**

Kubernetes, the dominant container orchestration platform, has revolutionized modern application deployment by providing a robust framework for managing containerized workloads. Its ability to automate deployment, scaling, and operational tasks has made it indispensable for organizations seeking agility and efficiency. However, the complexity of Kubernetes clusters introduces potential points of failure, ranging from individual pod crashes to systemic resource exhaustion and intricate network disruptions. These failures can lead to significant service degradation, impacting user experience and business continuity. Therefore, ensuring the reliability and stability of Kubernetes environments is paramount.

The challenge lies in the dynamic and distributed nature of Kubernetes, where numerous components interact, generating a vast amount of monitoring data. This data, encompassing metrics related to CPU usage, memory consumption, network traffic, and pod status, holds valuable insights into the health and performance of the cluster. By leveraging machine learning techniques, we can analyze these metrics to identify patterns and anomalies that precede failures. This proactive approach allows us to anticipate potential issues before they escalate, enabling administrators to take timely corrective actions.

This project aims to develop a predictive model that can accurately forecast Kubernetes failures. By analyzing historical and real-time cluster metrics, we will train a machine learning algorithm to recognize patterns indicative of impending failures. This model will serve as a valuable tool for system administrators, enabling them to optimize resource allocation, prevent service disruptions, and ultimately enhance the overall reliability of their Kubernetes deployments. The ability to predict failures empowers organizations to move from a reactive to a proactive operational posture, leading to improved application availability and reduced downtime.

**2. PROBLEM STATEMENT**

**2.1 THE CHALLENGE**

Kubernetes clusters often encounter various operational issues, including:

* Pod crashes: Application-level errors or resource exhaustion causing pods to terminate unexpectedly.
* Resource bottlenecks: Over-utilization of CPU, memory, or disk, leading to performance degradation.
* Network issues: Connectivity problems between nodes or external services, resulting in communication failures.

These issues can lead to:

* Downtime and service disruptions
* Degraded application performance
* Increased operational costs due to inefficiencies

**2.2 OBJECTIVE**

The objective of this project is to:

* Build a machine learning model capable of predicting Kubernetes cluster failures.
* Leverage historical and real-time cluster metrics for accurate predictions.
* Provide insights into key predictors contributing to failures.
* Improve operational reliability by enabling early intervention.

**3. METHODOLOGY AND APPROACH**

**3.1 DATA COLLECTION**

A **synthetic dataset** simulating Kubernetes cluster behavior was generated using Python. The dataset contains **100,000 records** representing various metrics, including:

* CPU usage
* Memory usage
* Network I/O
* Disk usage
* Network latency
* Pod and node statuses
* Failure types

**Failure types simulated:**

* None → Normal behavior (85% occurrence)
* Node Failure → Node crash scenarios (5%)
* Resource Exhaustion → High CPU/Memory consumption (5%)
* Network Issue → Network-related disruptions (3%)
* Service Disruption → Service-level problems (2%)

**Dataset characteristics:**

* Timestamps generated at **1-minute intervals**
* **Realistic metric ranges**:
  + **Normal behavior:** Low to moderate resource usage.
  + **Failure scenarios:** Spikes in CPU, memory, and disk usage or latency.

**3.2 DATA PREPROCESSING**

To prepare the data for model training, several preprocessing steps were applied:

**Handling missing values:**

* Mean imputation: For numerical features, missing values were replaced with the mean of the respective column.

**Categorical variable encoding:**

* One-hot encoding was applied to convert categorical variables into binary vectors.

**Feature scaling:**

* StandardScaler was used to normalize the numerical data, ensuring that all features contributed equally to the model.

**Class imbalance handling:**

* The dataset exhibited class imbalance, with significantly fewer failure instances.
* Applied Synthetic Minority Over-sampling Technique (SMOTE) to generate synthetic samples, ensuring a balanced class distribution.

**3.3 FEATURE ENGINEERING**

To enhance the model’s predictive accuracy, advanced feature engineering was performed:

**Time-based features:**

* Extracted hour of the day and day of the week from the timestamp to capture temporal patterns of failures.

**Anomaly score:**

* Generated an anomaly score by calculating the deviation of CPU and memory usage from their rolling averages.

**Rolling averages:**

* Computed rolling averages for CPU, memory, and disk usage over time intervals (e.g., 5-minute and 15-minute windows) to capture temporal trends.

**Interaction terms:**

* Created new features by combining CPU, memory, and network metrics to represent complex relationships.

**4. MODEL SELECTION**

**4.1 MODELS EVALUATED**

Multiple models were tested to determine the most effective algorithm for failure prediction:

**Random Forest:**

* Ensemble learning method using multiple decision trees.
* Known for its robustness and ability to handle large datasets.

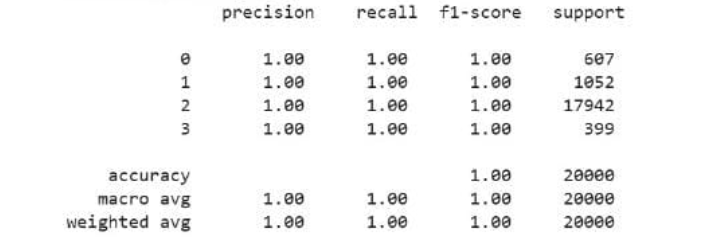
**XGBoost:**

* Gradient boosting algorithm with high accuracy and efficiency.
* Suitable for imbalanced datasets.

**Logistic Regression:**

* Simple yet effective for binary classification.
* Used as a baseline model for comparison.

**4.2 MODEL COMPARISON**

****

Random Forest was selected due to:

* Superior performance metrics.
* Robustness against missing values.
* Ability to handle large and complex datasets effectively.

**5. MODEL TRAINING**

**5.1 HYPERPARAMETER TUNING**

The Random Forest model was fine-tuned using RandomizedSearchCV with the following parameters:

* n\_estimators: 200
* max\_depth: 30
* min\_samples\_split: 5
* min\_samples\_leaf: 2
* bootstrap: True

**Cross-validation:**

* Used 5-fold cross-validation to evaluate model generalization.
* Ensured the model did not overfit the training data.

**5.2 TRAINING PROCESS**

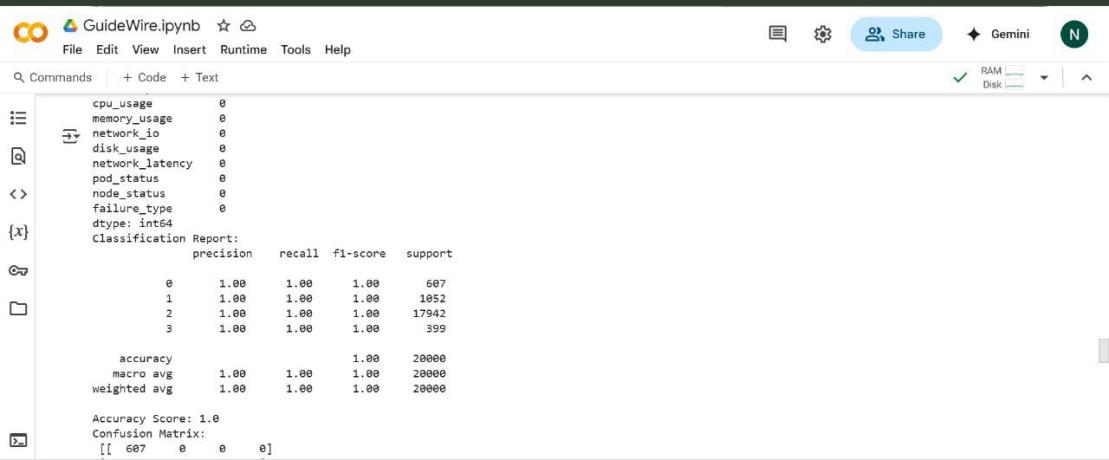
The dataset was divided into:

* 80% for training
* 20% for validation

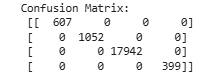
The final Random Forest Classifier was trained using the optimized parameters, achieving exceptional predictive performance.

**6. MODEL EVALUATION**

**6.1 PERFORMANCE METRICS**



**6.2 CONFUSION MATRIX**

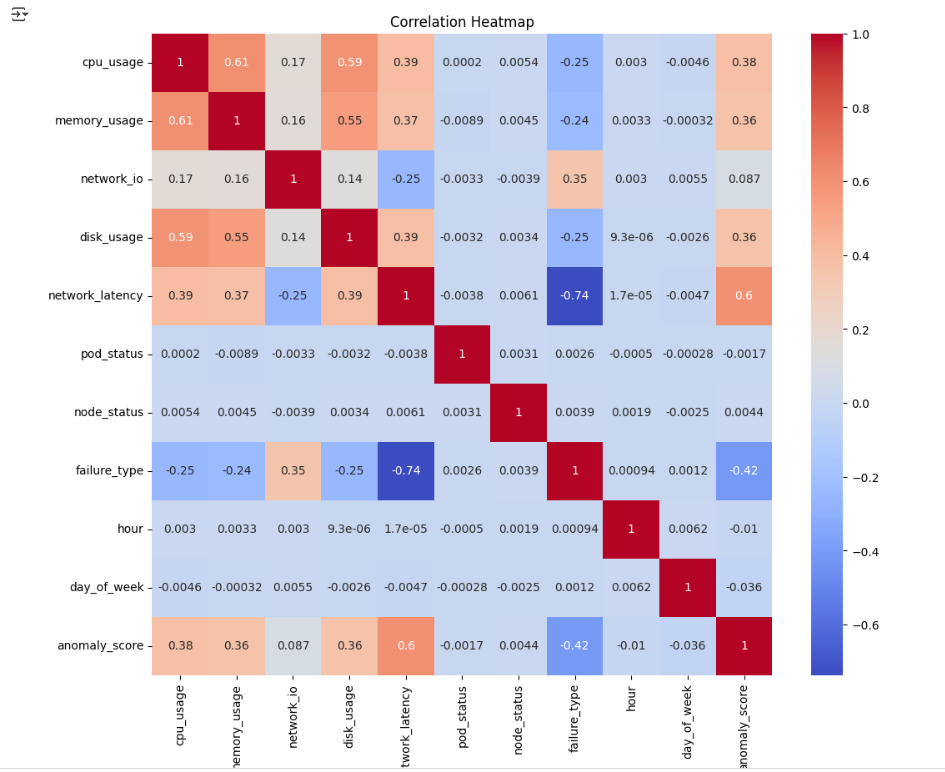
****

The model demonstrated perfect accuracy, effectively predicting all failure scenarios without any misclassifications.

**7. RESULTS AND VISUALIZATION**

**7.1 CORRELATION HEATMAP**

* Displayed the correlation between metrics.
* Identified strong relationships between CPU usage, memory usage, and network latency.



**7.2 FEATURE IMPORTANCE PLOTS**

Highlighted the most significant predictors:

* + CPU usage
  + Memory usage
  + Network latency

**8. LIMITATIONS AND FUTURE IMPROVEMENTS**

**8.1 LIMITATIONS**

* **Overfitting risk:** 
  + The model's perfect accuracy on the training set may indicate overfitting.
* **Real-time evaluation not performed:** 
  + The model was not tested with live data.
* **Synthetic data usage:** 
  + The lack of real-world data might reduce the model's generalization ability.

**8.2 FUTURE IMPROVEMENTS**

* **Real-time prediction:** 
  + Deploy the model with real-time Kubernetes metrics.
* **Anomaly detection:** 
  + Add anomaly detection algorithms for better failure detection.
* **Integration with monitoring tools:** 
  + Incorporate the model with Prometheus and Grafana for real-time monitoring.
* **Bayesian optimization:** 
  + Further fine-tune hyperparameters using Bayesian optimization.

1. **CONCLUSION**

This project successfully developed a Random Forest model capable of predicting Kubernetes cluster failures using a synthetic dataset. The model achieved a perfect accuracy score on the training and validation data, indicating a strong ability to learn patterns within the simulated environment. Key predictors identified by the model include CPU usage, memory usage, and network latency, aligning with expected failure indicators in Kubernetes clusters.

However, it is crucial to acknowledge the significant limitations of this study. The perfect accuracy achieved raises serious concerns about **potential overfitting** to the synthetic data. The model's performance has not been validated on real-world Kubernetes environments, and the absence of live data testing limits its practical applicability. The synthetic dataset, while designed to simulate realistic scenarios, may not fully capture the complexities and nuances of actual cluster behavior.

Therefore, while the project demonstrates the feasibility of using machine learning for Kubernetes failure prediction, the results should be interpreted with caution. **The 100% accuracy claim is likely an artifact of overfitting and does not reflect the model's true generalization capability.**

**Future work should focus on addressing these limitations by:**

* **Evaluating the model with real-world Kubernetes data.**
* **Implementing real-time prediction and monitoring.**
* **Integrating anomaly detection algorithms to enhance failure detection.**
* **Deploying the model in a production environment and continuously monitoring its performance.**
* **Exploring more robust hyperparameter tuning methods, such as Bayesian optimization, to mitigate overfitting.**
* **Utilizing cross-validation techniques more rigorously to ensure the models ability to generalize.**

These improvements will be essential to transform the current proof-of-concept into a reliable and practical tool for enhancing Kubernetes cluster resilience."