Kubernetes Failure Prediction - Phase 1 Report

1. Introduction

Problem Statement

Kubernetes clusters can experience failures such as:

- **Pod crashes**
- **Resource exhaustion (CPU, Memory, Disk)**
- **Network bottlenecks & service disruptions**

The goal of this project is to **build an AI/ML model that predicts these issues before they occur** using historical and real-time Kubernetes metrics.

2. Dataset

The dataset contains **100,000 records** of Kubernetes system metrics collected over time.

Features Used

Feature Name	Description	Type
CPU_Usage(%)	CPU utilization in percentage	Continuous
Memory_Usage(%)	Memory usage in percentage	Continuous
Disk_Usage(%)	Disk usage in percentage	Continuous
Network_IO(MB/s)	Network traffic in MB/s	Continuous
Error_Logs_Count	Number of error logs in a time	Integer
Pod_Status	Categorical (Running, Failed, etc.)	Encoded
Issue_Label	Target variable (0 = No Issue, 1 =	Binary

3. Data Preprocessing

Preprocessing Steps

- **Dropped** Timestamp **column** (not needed for training).
- **Encoded** Pod_Status using LabelEncoder.
- **Standardized numerical features** using StandardScaler.
- **Handled class imbalance using SMOTE** (as dataset had more "No Issue" cases).

4. Model Development

Models Used

1. **Random Forest Classifier**
   - Tuned using RandomizedSearchCV for best hyperparameters.
   - Achieved **99.95% accuracy on original test data**.
2. **XGBoost Classifier**
   - Trained on balanced data.
   - Achieved **99.75% accuracy on original test data**.

5. Model Evaluation

Final Performance on Test Data (100,000 samples):

Model	Accuracy	Precision	Recall	F1-Score
Random Forest	99.95%	1.00	1.00	1.00
XGBoost	99.75%	1.00	1.00	1.00

Feature Importance Analysis (SHAP & Random Forest Importance Plot)

- **Most Important Features:**
  - **CPU Usage**
  - **Memory Usage**
  - **Pod Status**
  - **Error Log Count**

6. Testing on a Different Dataset (2,000 records)

After training on **100,000 records**, the model was tested on a completely **different dataset of 2,000 records**.

Performance on New Test Data:

Model	Accuracy	Precision	Recall	F1-Score
Random Forest	98.80%	99%	99%	99%
XGBoost	93.30%	94%	93%	93%

Confusion Matrix Analysis

Random Forest:

- **14 False Positives** (Predicted Issue, but No Issue)
- **10 False Negatives** (Missed an actual Issue)

XGBoost:

- **125 False Positives** (Predicted more failures than actual)
- **9 False Negatives** (Missed actual Issue)

Conclusion:

- **Random Forest is highly accurate & precise.**
- **XGBoost is more aggressive in predicting failures (higher False Positives).**
- **Both models generalize well to unseen data.**

7. Deployment & Model Saving

Trained models were saved as .**pkl files** for deployment .

Feature Importance & SHAP values were also saved for future analysis.

8. Conclusion

In **Phase 1**, we successfully developed an **AI/ML model** to predict **Kubernetes failures** using historical and real-time system metrics. **Random Forest** achieved **98.8% accuracy**, demonstrating strong generalization to unseen data, while **XGBoost** provided a more aggressive failure prediction approach.

The model was tested on **a different dataset (2,000 records)** and showed **high precision and recall**, confirming its reliability. The trained models were **saved for deployment**, and feature importance analysis identified key contributors to failures, such as **CPU usage, memory usage, and error log count**.

9. Next Steps (Phase 2 Preview)

Deploying the model in Kubernetes monitoring systems

Automating failure remediation using AI-driven alerts

Optimizing model for real-time inference in production