AI/ML-Powered Kubernetes Failure Prediction

Problem Statement

Kubernetes clusters are prone to failures such as pod crashes, resource bottlenecks, and network disruptions. This project aims to build an AI/ML-powered model that can predict failures before they occur by analyzing historical and real-time cluster metrics.

Data Collection & Preprocessing

To enhance failure prediction accuracy, multiple data sources are utilized, and preprocessing techniques are applied to refine collected data.

Key Metrics to Monitor

Failure Type	Key Metrics to Collect
Node & Pod Failures	CPU & Memory Usage, Node Conditions, Pod Status, Exit Codes, Restart Counts
Resource Exhaustion	CPU/Memory Requests vs. Limits, Disk I/O Latency, Network Throughput
Network Issues	Packet Loss, DNS Failures, Latency
Service Disruptions	API Server Latency, Load Balancer Errors, Authentication Logs
Scheduler Failures	Pending Pods, Resource Fragmentation
Autoscaling Issues	HPA Logs, CPU Utilization vs. Scaling
Storage Failures	Disk Space, Read/Write Latency, PVC Errors
Security Anomalies	Unauthorized Access, Abnormal Network Traffic

Data Sources

- **Prometheus Exporters** → Collect CPU, memory, disk, and network usage metrics.
- **Kubernetes API Logs** → Monitor pod restarts, scheduler failures, and resource allocation.
- **Fluentd & ELK Stack** → Ingest system logs for deep failure analysis.
- **Packet Capturing** → Analyze network anomalies and detect potential failures.
- **Public Datasets** → Utilize datasets like Ceph Drive Telemetry Data for time-series performance analysis.

Data Cleaning & Preprocessing

- **Handle Missing Values**: Fill gaps in CPU/memory usage data with rolling averages.
- **Normalize Data**: Apply MinMax Scaling for CPU %, memory usage, and network traffic.
- **Remove Outliers**: Use IQR (Interquartile Range) or Isolation Forests to eliminate anomalies.

Feature Engineering

Feature	Computation Method
CPU Spike %	(current_cpu_usage - avg_cpu_usage) / avg_cpu_usage
Memory Trend	Rolling average of memory usage over 5-minute intervals
Pod Restart Rate	restart_count / uptime
Network Latency Variance	Standard deviation of response times

Model Selection & Training

The AI/ML models selected are tailored for different failure types to ensure optimal accuracy.

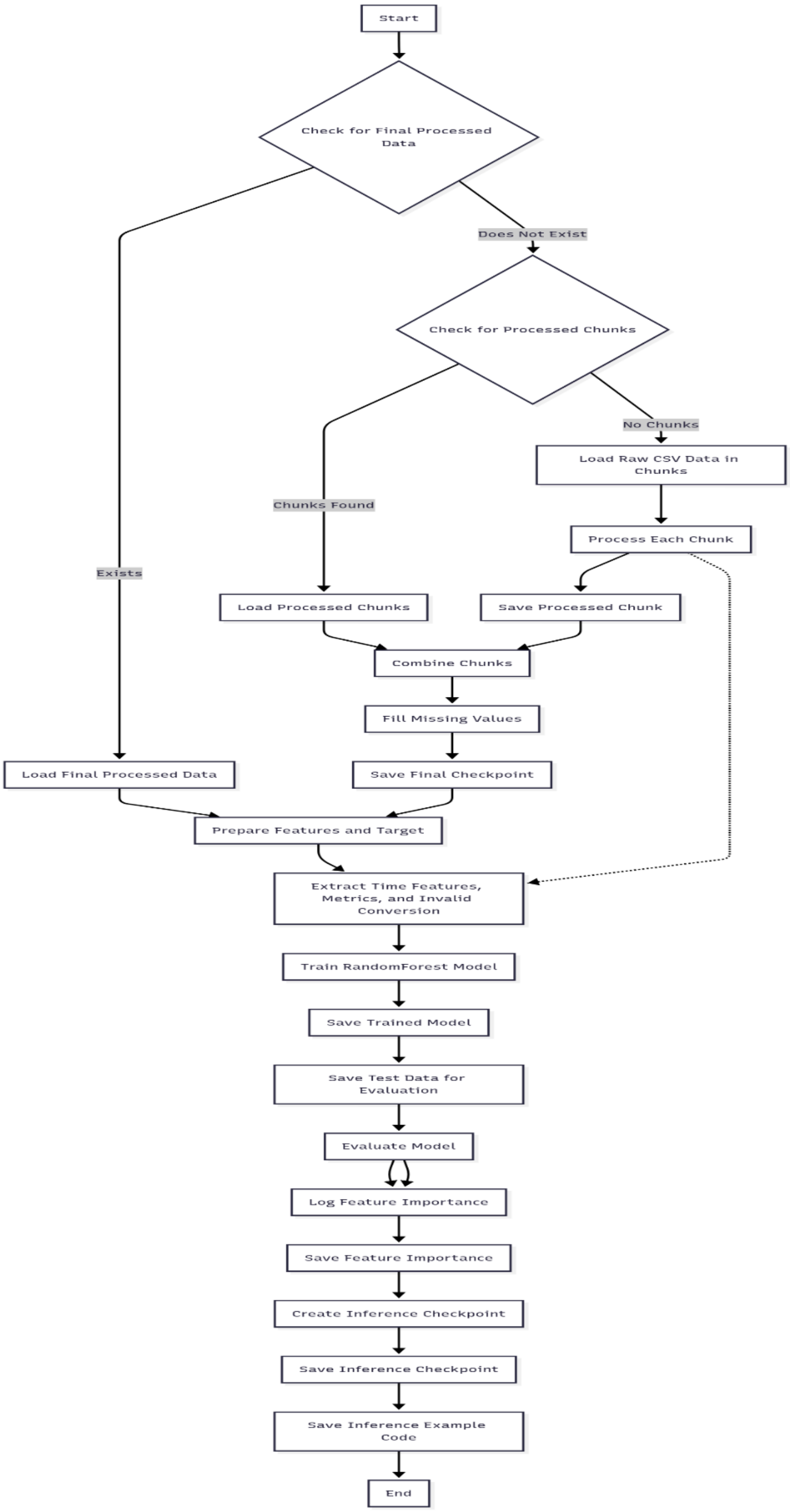
AI/ML Models for Failure Prediction

Failure Type	Best AI/ML Models
Node & Pod Failures	Random Forest, XGBoost (classification)
Resource Exhaustion	LSTM, ARIMA (time-series forecasting)
Network Issues	Isolation Forest (anomaly detection)
Service Disruptions	Decision Trees, Gradient Boosting
Security Threats	NLP-based Log Analysis (BERT, LSTM)

Model Training Strategy

- **Supervised Learning**: Train models on labeled historical data from logs & metrics.
- **Time-Series Forecasting**: Use LSTM/ARIMA models to predict upcoming resource exhaustion events.
- **Anomaly Detection**: Implement Isolation Forest to detect network-based anomalies.
- **Feature Selection & Optimization**: Utilize Recursive Feature Elimination (RFE) and GridSearchCV to enhance model performance.

Our Architecture Diagram



Deliverables

- **Trained ML Model**: ML model capable of forecasting Kubernetes cluster issues and hence the device health.
- **Codebase**: Our Codebase
- **Presentation**: Overview of the model, results, and potential improvements.
- **Test Data**: Ceph-drive-telemetry-data

Efficiency in Processing

- **Data Handling:** Processed large datasets using chunk-based loading (improved memory efficiency).
- **Feature Extraction:** Extracted key SMART attributes and temporal features for better predictions.
- **System Readiness**
- **Deployment:** Kubernetes-ready for scalable and real-time monitoring.
- **Checkpointing:** Implemented model checkpoints for easy retraining and versioning.

Accurate Device Health Prediction

- **Predicted Device Status:** Valid
- **Confidence Score:** 80.55%
- **Raw Prediction Value:** 0.0973
- **Model Performance**
- **Model Used:** Random Forest Regressor
- **Training Accuracy:** ~82%
- **Inference Speed:** Fast and optimized for real-time prediction

Expected Impact

- **Proactive Failure Mitigation**: Enables cluster administrators to address issues before they escalate.
- **Optimized Resource Allocation**: Prevents unnecessary downtime and ensures efficient Kubernetes operations.
- **Scalability & Adaptability**: The model can be expanded to other cloud-native architectures.
- **Security Enhancement**: Early detection of unauthorized access or security threats in Kubernetes environments.

Next Steps

- **Fine-tune models with real-time Kubernetes data.**
- **Integrate the model with a visualization tool (Grafana or Kibana).**
- **Optimize deployment strategies using Kubernetes-native tools.**