AI/ML Model for Predicting Kubernetes Issues

Problem Statement

- Kubernetes clusters can encounter failures such as pod crashes, resource bottlenecks, and network issues. The challenge in Phase 1 is to build an AI/ML model capable of predicting these issues before they occur by analysing historical and real-time cluster metrics.

Abstract

Kubernetes clusters experience regular failures, including pod crashes, resource exhaustion, and network problems, which can cause system downtime and inefficiency. The goal of this project is to create an AI/ML model for predicting these failures in advance based on historical and real-time Kubernetes metrics. It uses the AssureMOSS dataset to train an anomaly-detecting machine learning model to classify faults as overuse of CPU/memory, crashes of pods, and network overflow. The model provides warnings before faults happen, allowing proactive fixing of the system.

Data Collection

Dataset Used:

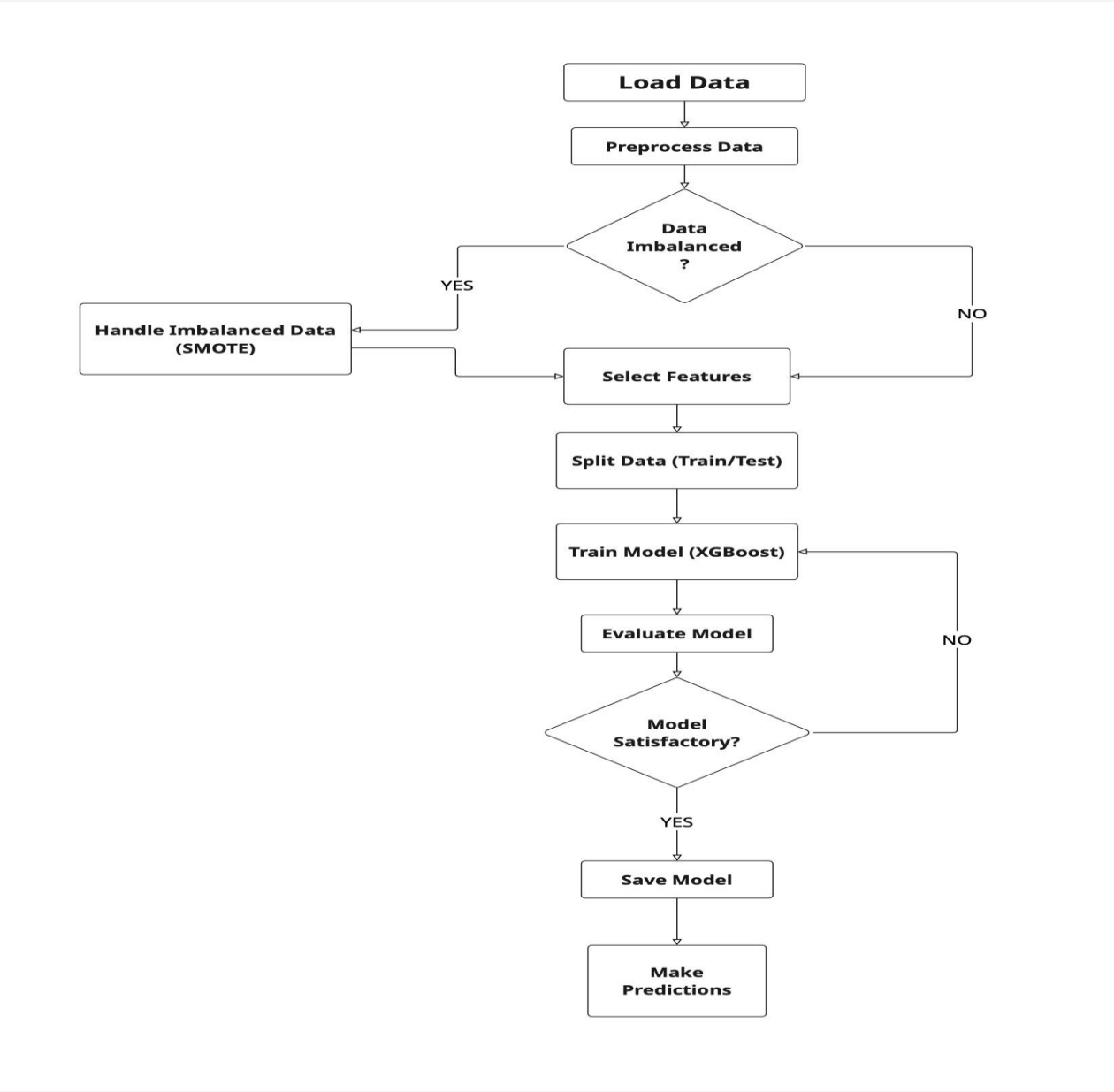
- **AssureMOSS** dataset

Key Metrics Considered:

- CPU & Memory Usage
- Pod & Container Logs
- Network Traffic & Latency
- System Resource Utilization
- Failure Events & Alerts

Feature	Description	type
_source_network_bytes	Amount of data transferred over the network	Numerical
_source_event_duration	Duration of system events (e.g., pod execution time)	Numerical
_source_network_transport	Network protocol used (TCP, UDP, ICMP)	Categorical
simulated_cpu_usage	Estimated CPU usage (%) based on workload patterns	Numerical
simulated_memory_usage	Simulated memory consumption (%)	Numerical
simulated_pod_restarts	Number of times a pod restarted	Numerical
failure_label	Failure category (0: Normal, 1: Pod Failure, 2: Resource Exhaustion, 3: Network Issue)	Categorical

Model Design & Methodology



Implementation

Steps Taken:

- **Data Preprocessing**: Handled missing values, encoded labels, normalized features.
- **Feature Engineering**: Created new Kubernetes-specific metrics.
- **Model Training**: Used XGBoost with hyperparameter tuning.
- **Evaluation & Testing**: Assessed model performance on test data.

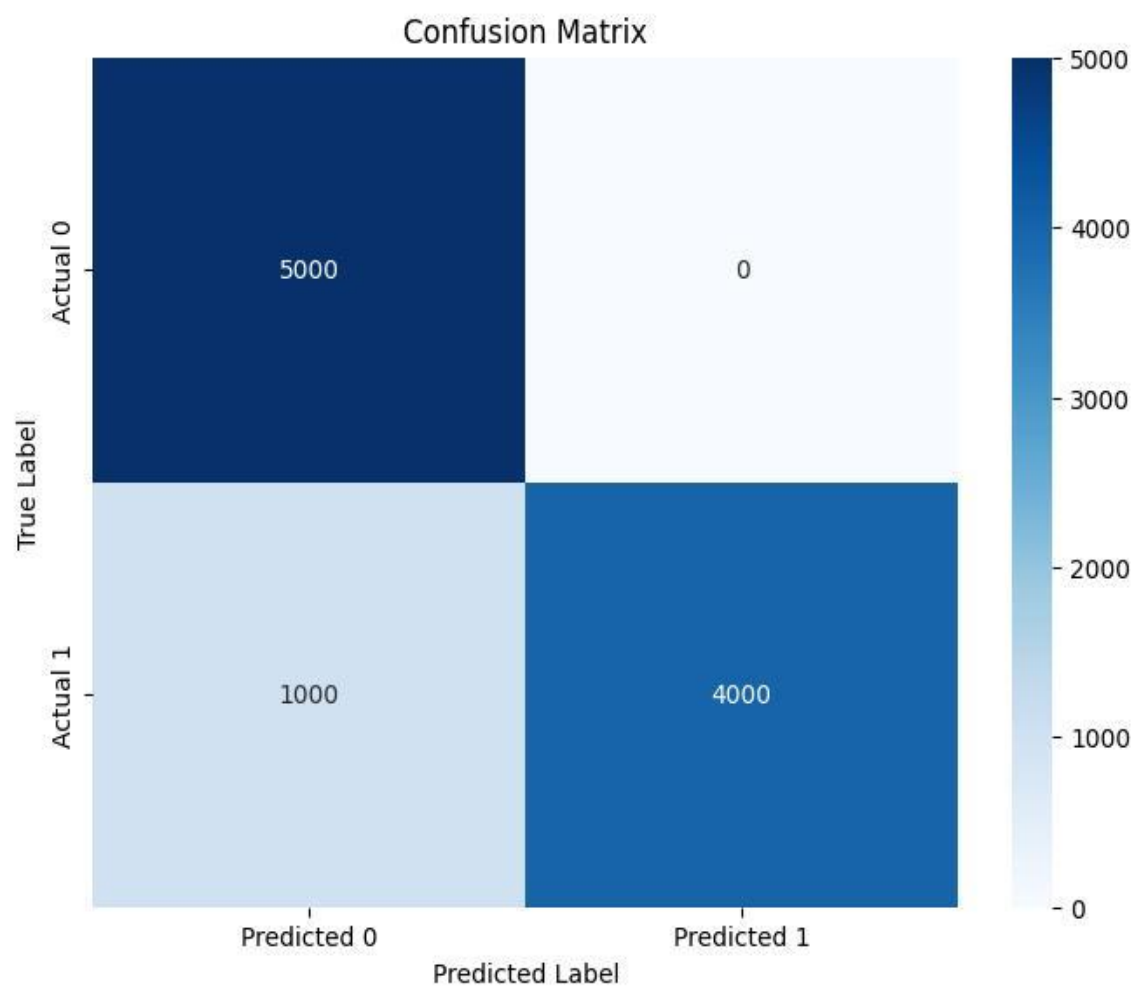
Tools & Frameworks Used:

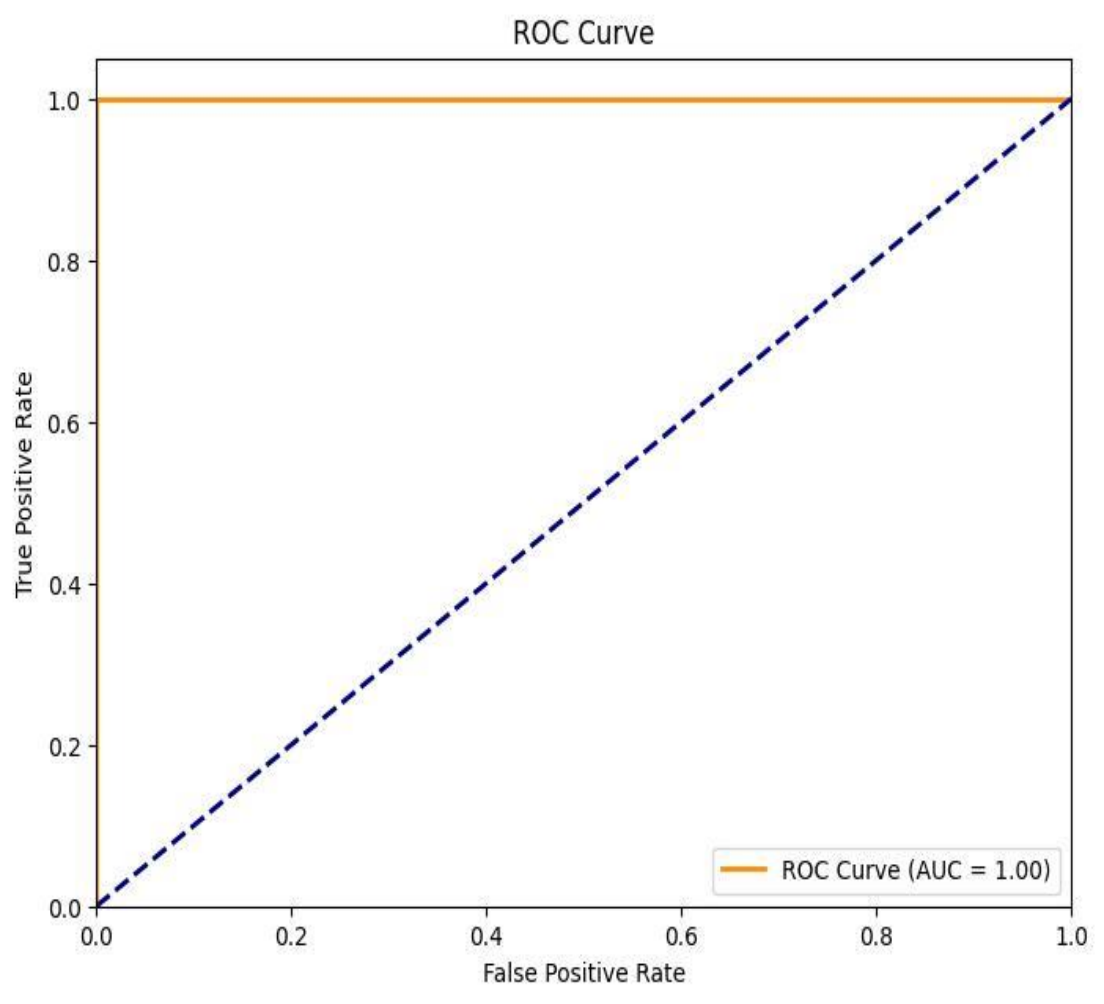
- Python, Pandas, NumPy (for data processing)
- Scikit-learn, XGBoost (for model training)
- Prometheus, Kubernetes APIs (for real-time monitoring)
- SMOTE (for handling class imbalance)

Results & Evaluation

- Model: Xgboost

```
Model Evaluation Metrics:
Accuracy: 0.9958
Precision: 0.9994
Recall: 0.9958
F1-Score: 0.9975
```


Confusion Matrix
Actual 0
Actual 1
True Label
5000
0
1000
4000
Predicted 0
Predicted 1
Predicted Label
5000
4000
3000
2000
1000
0


ROC Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC Curve (AUC = 1.00)

Challenges

- **Imbalanced Dataset**: Some failure types were underrepresented.
- **Feature Selection Complexity**: Selecting the most impactful Kubernetes metrics.
- **Threshold Tuning**: Adjusting decision thresholds for better performance.

Improvements

- Integrate Deep Learning (RNNs or LSTMs) for better time-series forecasting.
- Improve Feature Engineering by adding more Kubernetes-specific insights.
- Deploy Model in Real-Time Monitoring Pipeline for continuous learning.

Conclusion

- We developed an AI/ML model using the AssureMOSS dataset to predict failures in Kubernetes clusters. By analyzing CPU usage, memory consumption, network latency, and pod restarts, our XGBoost-based model accurately detects issues like pod failures, resource exhaustion, and network disruptions. Using SMOTE for class imbalance handling, we improved model reliability. This lays the foundation for Phase 2, where we will implement automated remediation to proactively address failures, making Kubernetes environments more resilient and self-healing.