Sens.AI : AI & ML Agent for Predicting and Remediating Kubernetes Cluster Issues

1. Data Collection & Preprocessing

Data Sources:

The approach starts by gathering historical and simulated metrics from Kubernetes clusters. Key metrics include CPU and memory usage, pod statuses, network I/O, and other operational logs.

Data Cleaning & Feature Engineering:

The collected data is then cleaned and normalized to remove noise and ensure consistency. Important features are extracted that are most likely to signal issues, such as unusual spikes in CPU usage or patterns in pod failures.

2. Model Development

Anomaly Detection & Time-Series Analysis:

Given the nature of cluster operations, the approach leverages techniques like anomaly detection and time-series forecasting. This helps in identifying deviations from normal behavior that could indicate a potential failure.

Model Training:

The preprocessed data is split into training and testing sets. Various machine learning algorithms (e.g., decision trees, support vector machines, or neural networks) may be experimented with to find the best predictor for issues like node or pod failures, resource exhaustion, or network connectivity problems.

3. Model Evaluation

Accuracy: Accuracy is the overall measure of how often the model correctly predicts the state of the Kubernetes clusters. It is calculated as the ratio of correct predictions (both true positives and true negatives) to the total number of predictions.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

In our scenario, accuracy gives an overall sense of model performance. However, when failures (e.g., pod crashes or resource bottlenecks) are rare compared to normal operations, accuracy might be high even if the model misses some critical failure events.

Precision: Precision quantifies the proportion of predicted failure events that were actually failures. It is especially important in scenarios where false alarms (false positives) must be minimized to avoid unnecessary remediation actions.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

In our use case, high precision is crucial if triggering an unnecessary remediation (like restarting a pod) could lead to wasted resources or service disruption.

Recall (Sensitivity): Recall measures the proportion of actual failure events that were correctly identified by the model. It is a critical metric when the cost of missing a failure (false negative) is high.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

For predicting Kubernetes issues, missing a true failure event (e.g., a resource exhaustion event) can have severe operational consequences. Therefore, recall is prioritized when ensuring the model captures as many failure events as possible.

F1 Score: The F1 Score is the harmonic mean of precision and recall, providing a balance between the two. It is particularly useful when both false positives and false negatives are critical.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

In this documentation, the F1 score serves as a comprehensive metric that reflects the trade-off between precision and recall, ensuring that neither false alarms nor missed failures are disproportionately impacting the model's performance.

Macro Average: Macro Average calculates the arithmetic mean of precision, recall, and F1 scores across all classes. It treats each class equally regardless of its frequency.

$$\text{Macro AVG} = \frac{1}{N} \sum_{i=1}^{N} \text{Score}_i$$

This metric is included in our documentation to show model performance in a balanced manner. It is particularly useful when we want to assess performance across different types of failures equally, even if some failure types occur less frequently.

Weighted Average: Weighted Average takes the average of the precision, recall, and F1 scores, but it weights each class by its frequency (support). This gives a more realistic picture when the dataset is imbalanced.

$$\text{Weighted AVG} = \sum_{i=1}^{N} \left(\frac{\text{Support}_i}{\text{Total Samples}} \times \text{Score}_i \right)$$

Given that certain failure events may be rarer than others, the weighted average is a key metric for our model. It ensures that the performance evaluation reflects the impact of each class in proportion to its occurrence in the dataset.

4. Remediation System Integration

Actionable Insights:

Once the model predicts an impending issue, the system triggers a remediation process. This might include actions such as scaling pods, restarting services, or reallocating resources to mitigate the predicted failure.

Automation & Agent Integration:

An automated agent can be integrated with the prediction model to execute or recommend remediation steps, ensuring a swift response to potential cluster issues. This integration is key to reducing downtime and maintaining service continuity.

5. Results:

Multi-Class Model Accuracy: 0.9895642621750275

Classification Report:

```
              precision    recall  f1-score   support

           0       0.95      0.97      0.96        86
           1       0.98      0.98      0.98      1335
           2       0.99      0.99      0.99       466
           3       0.99      0.99      0.99      3557
           4       1.00      0.83      0.91        18

    accuracy                           0.99      5462
   macro avg       0.98      0.95      0.97      5462
weighted avg       0.99      0.99      0.99      5462
```

Label :

0 – No Failure

1 - Node or pod failures.

2 - Resource exhaustion (CPU, memory, disk)

3 - Network or connectivity issues.

4 - Service disruptions based on logs and events.