# HELP zscore\_score Current Z-score for topic's message rate

# TYPE zscore\_score gauge

zscore\_score{topic="job-metrics"} 2.8

# Helps quantify how far the current rate deviates from the norm (mean)

# HELP zscore\_anomaly\_detected 1 if current rate is anomalous

# TYPE zscore\_anomaly\_detected gauge

zscore\_anomaly\_detected{topic="job-metrics"} 1

# Indicates if an anomaly was detected in the current window

# HELP zscore\_msg\_rate Current message rate per topic (e.g., per 60s)

# TYPE zscore\_msg\_rate gauge

zscore\_msg\_rate{topic="job-metrics"} 20

# Raw message count seen in the last time interval

# HELP zscore\_baseline\_mean Historical mean rate from sliding window

# TYPE zscore\_baseline\_mean gauge

zscore\_baseline\_mean{topic="job-metrics"} 100

# Shows the normal/expected rate based on past N intervals

# HELP zscore\_baseline\_std Standard deviation of historical rates

# TYPE zscore\_baseline\_std gauge

zscore\_baseline\_std{topic="job-metrics"} 15.6

# Used to compute Z-score: helps gauge volatility of topic rates

# HELP zscore\_last\_anomaly\_time Unix timestamp of last detected anomaly

# TYPE zscore\_last\_anomaly\_time gauge

zscore\_last\_anomaly\_time{topic="job-metrics"} 1720521600

# Useful for tracking when the last drop/spike occurred

# HELP zscore\_anomaly\_type Type of anomaly detected: drop or spike

# TYPE zscore\_anomaly\_type gauge

zscore\_anomaly\_type{topic="job-metrics", type="drop"} 1

# Adds detail about \*why\* it was marked anomalous

# HELP zscore\_recovery\_detected Indicates recovery after anomaly

# TYPE zscore\_recovery\_detected gauge

zscore\_recovery\_detected{topic="job-metrics"} 1

# Signals that topic rate has returned to normal

# HELP zscore\_confidence Confidence of anomaly (smoothed z-score)

# TYPE zscore\_confidence gauge

zscore\_confidence{topic="job-metrics"} 0.91

# Optional smoothing metric to reduce false positives

# HELP zscore\_status Overall topic health status

# TYPE zscore\_status gauge

zscore\_status{topic="job-metrics", state="normal"} 1

zscore\_status{topic="job-metrics", state="anomaly"} 0

# One metric to feed into dashboards or status panels

Metric NameDescriptionzscore\_score{topic="..."}Current Z-score valuezscore\_anomaly\_detected{topic="..."}1 if anomaly detected, 0 otherwisezscore\_baseline\_mean{topic="..."}Sliding window mean of message ratezscore\_baseline\_std{topic="..."}Sliding window std deviationzscore\_msg\_rate{topic="..."}Latest observed message count (e.g., per 60s)zscore\_last\_anomaly\_time{topic="..."}Unix timestamp of last anomaly (optional)

### Data Quality Scoring – ML-Driven Design (Training + Inference)

#### Purpose

To automatically detect malformed, missing, or anomalous records in data flowing through our cluster pipeline (e.g., Redfish telemetry, job metrics, etc.) using a unified machine learning model. This prevents low-quality data from corrupting downstream systems like OpenSearch, VictoriaMetrics, or ML models.

---

#### 🧠 Training Phase (One Model for All Topics)

1. \*\*Collect Sample Records\*\* from multiple Kafka topics (e.g., `redfish-telemetry`, `job-metrics`).

2. \*\*Assign `topic\_id`\*\* to each record to indicate its origin.

3. \*\*Extract Features\*\* (topic-specific fields like `cpu\_temp`, `mem\_used`, etc.).

4. \*\*Label Records (`label = 1 or 0`)\*\* using external rule-based logic via a config file:

- For example:

- `cpu\_temp` must be between 0–100

- `mem\_used` must not be null

- `status` must not be "UNKNOWN"

5. \*\*Build Unified Training Dataset\*\* with:

- Mixed features from all topics (missing values allowed)

- `topic\_id` to help model distinguish formats

- `label` as target (1 = clean, 0 = bad)

6. \*\*Train a Binary Classifier\*\* (e.g., XGBoost or LightGBM):

- `model.fit(X, y)` where X includes all features + `topic\_id`

---

#### 🔮 Inference Phase (Live Scoring)

1. \*\*Kafka Consumer\*\* reads new data from any topic.

2. \*\*Extract Features\*\* from the incoming record.

3. \*\*Assign `topic\_id`\*\* based on topic name.

4. \*\*Predict Cleanliness\*\*:

```python

quality\_score = model.predict\_proba([features\_with\_topic\_id])[0][1]