We will describe:

1. Mathematical Definition
2. Step-by-Step Algorithm
3. Pseudocode
4. Time Complexity
5. How it contributes to final Drift decision

📘 Production Signal Drift Analyzer

Algorithms for Measuring Data Drift

1️⃣ Mean Shift (Location Drift)

📌 Purpose

Measures change in central tendency between reference and live data.

If the center of distribution moves → location drift.

📐 Mathematical Definition

$$\mu_R = \frac{1}{n} \sum_{i=1}^{n} R_i$$

$$\mu_L = \frac{1}{m} \sum_{i=1}^{m} L_i$$

$$\Delta_\mu = |\mu_R - \mu_L|$$

🧮 Algorithm Steps

1. Compute mean of reference dataset.
2. Compute mean of live dataset.
3. Compute absolute difference.
4. Compare with threshold.

🧠 Pseudocode

```
FUNCTION MeanShift(reference, live):

    mean_ref = SUM(reference) / LENGTH(reference)
    mean_live = SUM(live) / LENGTH(live)

    delta_mu = ABS(mean_ref - mean_live)

    RETURN delta_mu
```

⏱️ Complexity

O(n + m)

🎯 Interpretation

Small Δμ → Stable
Moderate Δμ → Slight Drift
Large Δμ → Major Drift

2️⃣ Variance Shift (Spread Drift)

📌 Purpose

Measures change in dispersion (spread) of values.

Even if means match, variance may change.

📐 Mathematical Definition

$$\sigma^2 = \frac{1}{n}\sum(x-\mu)^2$$

$$\Delta_\sigma = |\sigma_R - \sigma_L|$$

🧮 Algorithm Steps

1. Compute mean.
2. Compute variance.
3. Take square root → standard deviation.
4. Compute absolute difference.

🧠 Pseudocode

```
FUNCTION VarianceShift(reference, live):

    mean_ref = compute_mean(reference)
    mean_live = compute_mean(live)

    std_ref = sqrt(SUM((x - mean_ref)^2) / n)
    std_live = sqrt(SUM((x - mean_live)^2) / m)

    delta_sigma = ABS(std_ref - std_live)

    RETURN delta_sigma
```

⏱️ Complexity

O(n + m)

🎯 Interpretation

Spread widening → increased uncertainty
Spread shrinking → over-concentration

3️⃣ Distribution Shape Shift (Modality Change)

📌 Purpose

Detects structural change (unimodal → bimodal)

Mean and variance may look similar, but shape differs.

📐 Concept

Convert data into histogram bins.
Count number of local peaks.

If peak count changes → structural drift.

🧮 Algorithm Steps

1. Find global min and max.
2. Create fixed number of bins.
3. Build histogram.
4. Count local maxima.

🧠 Pseudocode

```
FUNCTION CountPeaks(data, bins):

    min_val = MIN(data)
    max_val = MAX(data)

    IF min_val == max_val:
        RETURN 1

    width = (max_val - min_val) / bins

    histogram = ARRAY[bins] initialized to 0

    FOR each value in data:
        index = FLOOR((value - min_val) / width)
        IF index == bins:
            index = bins - 1
        histogram[index] += 1

    peaks = 0

    FOR i from 1 to bins-2:
        IF histogram[i] > histogram[i-1] AND histogram[i] > histogram[i+1]:
            peaks += 1

    RETURN MAX(peaks, 1)
```

⏱️ Complexity

O(n)

🎯 Interpretation

Peak change → Bimodal Shift
Highest priority drift type

4️⃣ Distribution Divergence (Histogram Distance)

📌 Purpose

Measures overall probability mass difference.

Unlike mean or variance, this captures full distribution difference.

📐 Mathematical Definition

Let:

$$P(i) = \frac{count_{ref}(i)}{n}$$

$$Q(i) = \frac{count_{live}(i)}{m}$$

Histogram Distance:

$$D = \frac{1}{2} \sum |P(i) - Q(i)|$$

Range:

0 ≤ D ≤ 1

🧮 Algorithm Steps

1. Define common bins.
2. Build histograms for both datasets.
3. Normalize to probabilities.
4. Compute total variation distance.

🧠 Pseudocode

```
FUNCTION HistogramDistance(reference, live, bins):

    min_val = MIN(global)
    max_val = MAX(global)
    width = (max_val - min_val) / bins

    hist_ref = ARRAY[bins]
    hist_live = ARRAY[bins]

    FILL histograms

    FOR i in bins:
        p = hist_ref[i] / n
        q = hist_live[i] / m
        divergence += ABS(p - q)

    RETURN divergence / 2
```

⏱️ Complexity

O(n + bins)

🎯 Interpretation

0 → identical
0.1–0.3 → moderate

> 0.4 → strong drift

🔥 Final Data Drift Decision Algorithm

Combine all metrics:

🧠 Unified Drift Algorithm

```
FUNCTION DetectDrift(reference, live):

    delta_mu = MeanShift(reference, live)
    delta_sigma = VarianceShift(reference, live)

    peaks_ref = CountPeaks(reference)
    peaks_live = CountPeaks(live)
    divergence = HistogramDistance(reference, live)

    IF peaks_ref != peaks_live:
        RETURN "Bimodal Shift"

    ELSE IF divergence >= 0.4:
        RETURN "Major Drift"

    ELSE IF divergence >= 0.15:
        RETURN "Slight Drift"

    ELSE:
        RETURN "Stable"
```

📊 Summary Table

Component	Detects	Sensitive To
Mean Shift	Location change	Translation
Variance Shift	Spread change	Scaling
Shape Shift	Structure	Multi-peak
Histogram Distance	Overall probability change	Global differences

🏁 Overall Time Complexity

O(n + m)