1. Algorithm Choice

For simplicity and effectiveness, we'll use a moving average with threshold method. This technique is quick, adapts well to changes, and can handle some concept drift and seasonality with slight modifications.

Explanation:

We calculate a moving average over a defined window to understand the expected value.

We flag any values that exceed a threshold (like 3 standard deviations from the mean) as anomalies.

This approach is efficient and understandable, suitable for real-time scenarios.

2. Data Stream Simulation

We'll simulate a continuous data stream with random noise and a seasonal pattern using the sine function. We'll also inject some random anomalies.

3. Real-time Anomaly Detection

We'll implement a basic real-time detection mechanism with moving statistics (mean and standard deviation) and use a threshold for anomaly detection.

4. Visualization

We'll create a simple visualization using matplotlib to display the data stream and anomalies.

Explanation of Key Sections:

1. Data Stream Simulation:

The simulate_data_stream() function creates a sequence with a sine wave pattern and Gaussian noise, adding a seasonal component. Randomly injected spikes represent anomalies.

2. Anomaly Detection:

The detect_anomalies() function uses a sliding window to calculate the moving average and standard deviation.

A point is flagged as an anomaly if it deviates from the mean by more than a set number of standard deviations (threshold_factor).

3. Visualization:

The visualize_data() function displays the data stream, moving average, and thresholds. Anomalies are marked with red ‘x’ markers.

4. Execution:

The main() function orchestrates the entire process by generating the data stream, running the anomaly detection, and visualizing results.