Efficient Data Stream Anomaly Detection

1. Problem Definition

The project task was to develop a Python script that can help detect anomalies in a continuous data stream. These streams could represent metrics such as financial transactions or system data, where anomalies are defined as unusual patterns like exceptionally high values or deviations from the norm.

2. Objectives

The developed script should be able to achieve the following objectives:

1. Data Stream Simulation: Contain a function to emulate a data stream, incorporating regular patterns, seasonal elements, and random noise
2. Anomaly Detection: A real-time mechanism to accurately flag anomalies as the data is streamed.
3. Optimization: Ensure the algorithm is optimized for both speed and efficiency.
4. Visualization: Create a straightforward real-time visualization tool to display both the data stream and any detected anomalies

3. Algorithm

3.1 Exponential Moving Average

For this problem, I selected to use the Exponential Moving Average (EMA) because of the following reasons:

1. **Concept Drift Adaptability:** With the EMA more weight is given to recent data points making it well-suited to handle concept drift, where the data distribution may change over time.
2. **Low Time & Space Complexity**: The EMA can be updated in O(1) time for each new data point, making it ideal for real-time streams. The EMA adapts quickly to new data without needing to store a large history of values, which keeps the space complexity low. Also, only the most recent EMA value is stored, keeping the memory footprint small.

3.2 Sliding Window with Standard Deviation

For setting dynamic thresholds around the EMA to detect anomalies, I implemented a sliding window to calculate the standard deviation of recent data points in respect to the EMA. The sliding window maintains the local context of the data stream, allowing thresholds to adjust dynamically based on recent variability in the data.
By focusing on recent data within the sliding window, the threshold adapts to local fluctuations or noise in the data. The EMA and standard deviation are used to define upper and lower thresholds that adjust in response to the stream’s variability.

4. Data Stream Simulation

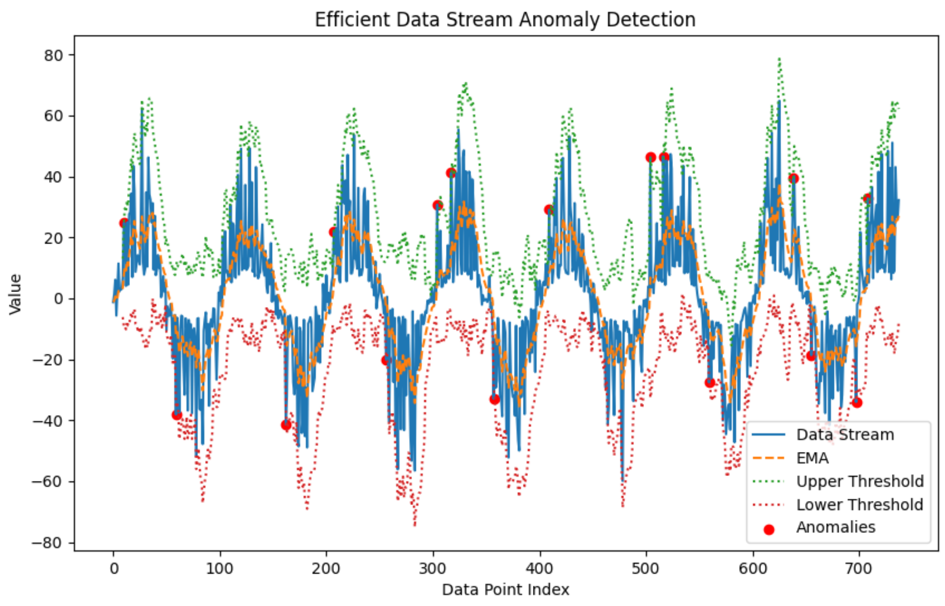
To test the system, I designed a data stream simulation function that generates a continuous flow of floating data points. This simulated data includes:

1. **Seasonal Patterns**: Modelled using a sine wave to represent regular periodic behaviour.
2. **Random Noise:** Added to mimic the inherent variability in real-world data.

3. **Anomalies**: Randomly introduced by amplifying some data points, simulating outliers that the anomaly detection system must flag.

5. Testing

The anomaly detection program was tested on the simulated data stream, which included regular patterns, random noise, and anomalies. The script effectively flags outliers and deviations from the normal patterns based on the dynamic thresholds set by the sliding window. The most impactful setting to the anomaly detection sensitivity is the threshold multiplier as it sets the bounds for what the system considers “normal” based on the EMA and recent deviations. Below is a sample output



6. Conclusion

This project showcases an efficient and responsive approach to real-time anomaly detection for continuous data streams. The project leverages the Exponential Moving Average to track the trend together with a sliding window to dynamically adjust thresholds. For improvement, more advanced algorithms like Isolation Forest or One-Class SVM could be explored for more complex anomaly detection tasks.