**Efficient Data Stream Anomaly Detection**

Anomaly Detection Using Z-Score with Various Window Sizes.

**Introduction**

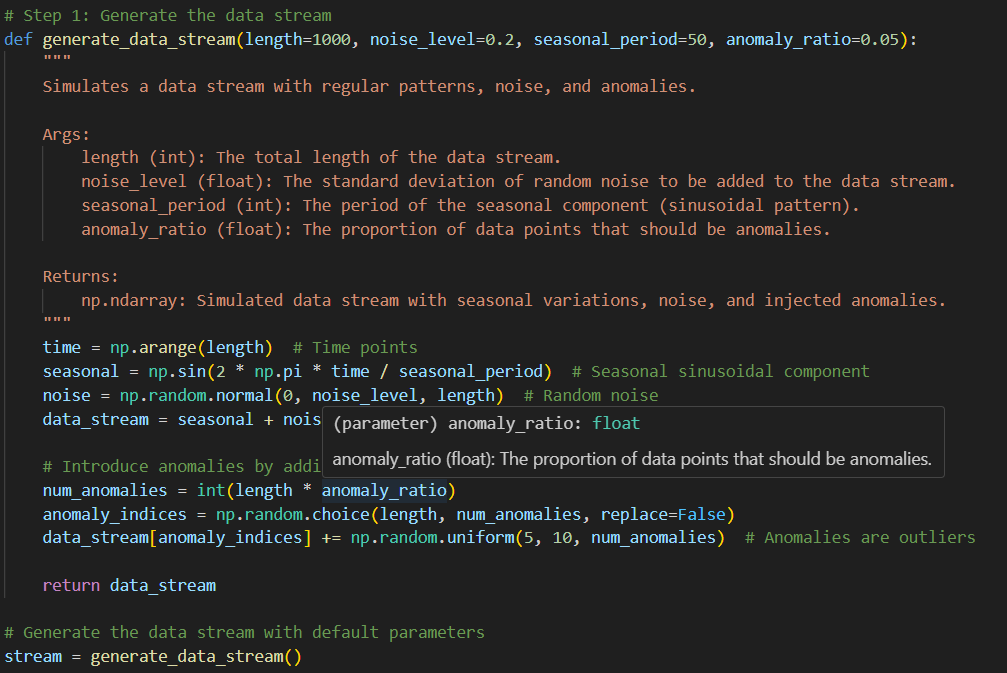
This project aims to detect anomalies in a data stream using a Z-Score-based algorithm. The algorithm adapts to seasonal variations and concept drift by applying a sliding window technique to maintain efficiency. Different window sizes were tested, and it was found that a window size of 30 produced the best results.

**Objectives**

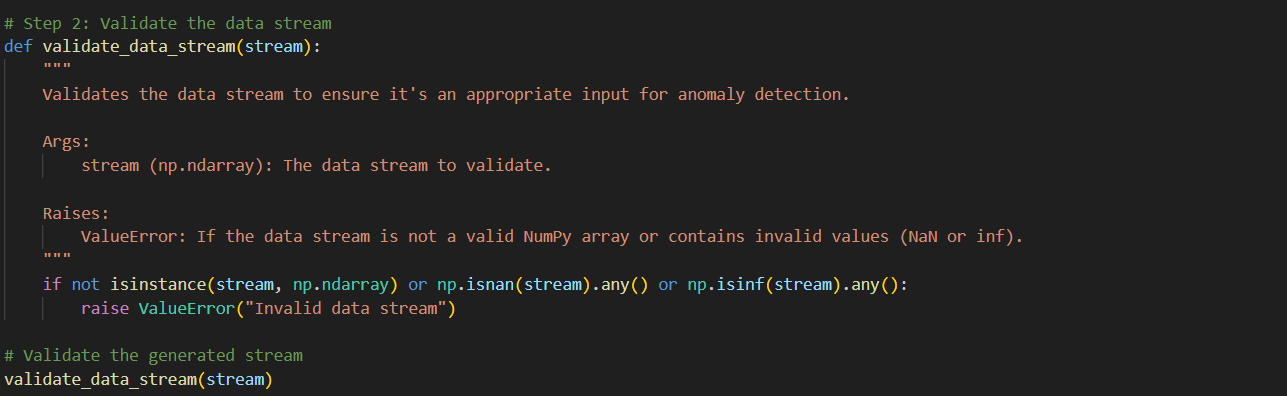
1. Algorithm Selection: Implement a suitable anomaly detection algorithm, adaptable to seasonal variations.
   * Given the need for efficiency and adaptability, Z-Score with a sliding window would be lightweight and sufficient for this project.
2. Data Stream Simulation: Emulate a data stream incorporating seasonal elements and random noise.
   * We can simulate a data stream by generating a sequence of values that combine regular patterns, noise, and occasional anomalies.
3. Anomaly Detection: Create a real-time mechanism to flag anomalies.
   * I implemented a real-time detection method using the Z-Score algorithm. The Z-Score can help identify values that are too far from the mean.
4. Optimization: Ensure the algorithm is optimized for both speed and efficiency.
   * the sliding window approach is efficient because it maintains only a small portion of the data in memory, avoiding performance bottlenecks.
   * While experimenting with different window sizes 30 was found to provide the best balance between sensitivity and accuracy.
5. Visualization: Provide a real-time visualization tool to display the data stream and detected anomalies.
   * I created a simple real-time plot using matplotlib that provides real-time detection for anomalies.

**Code Documentation**

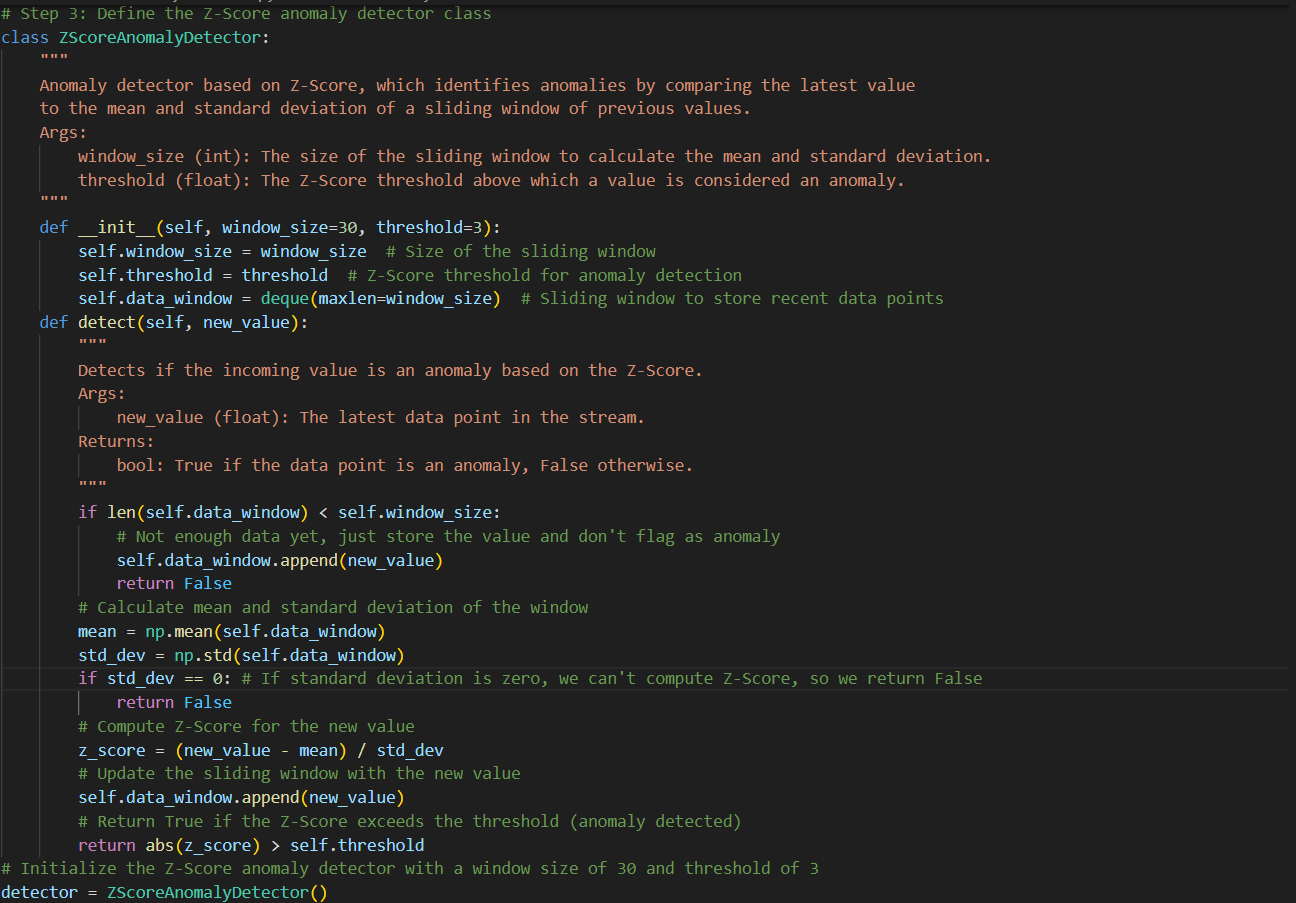
Step 1: Generate Data Stream The data stream is generated using a sinusoidal function to simulate seasonal variations, along with random noise and injected anomalies.

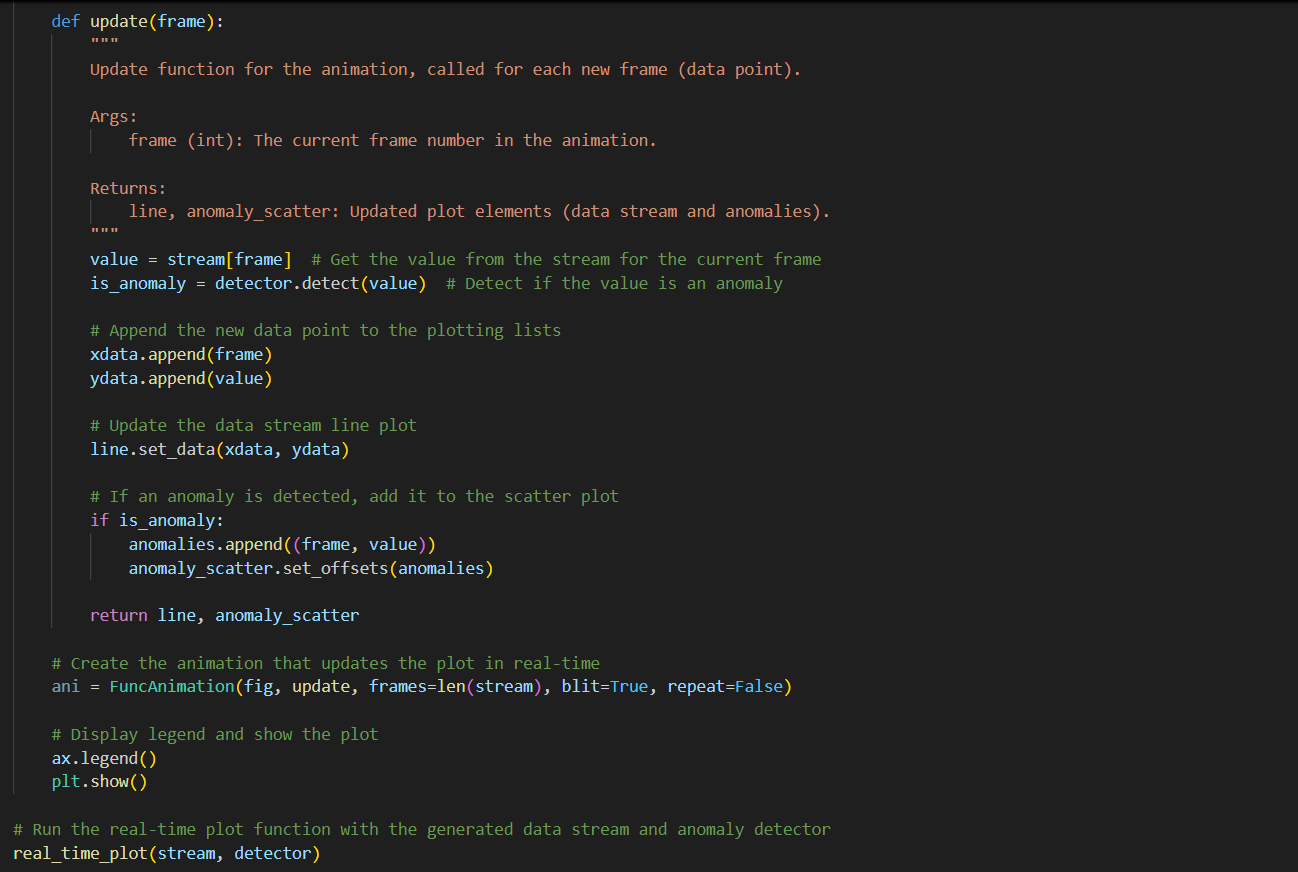
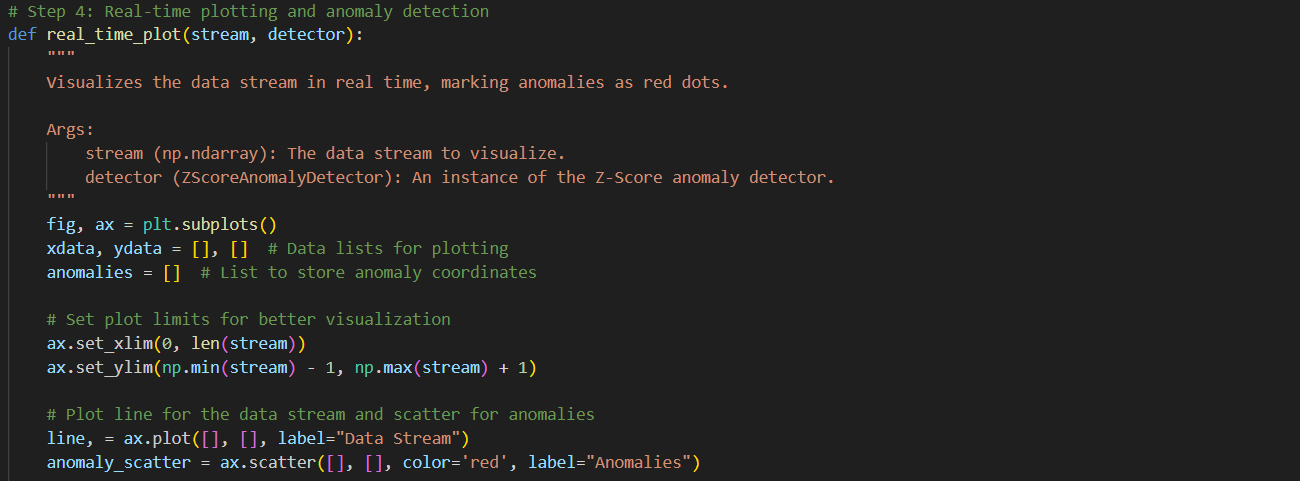


Step 2: Validate Data Stream The generated data stream is validated to ensure it is free from NaN or infinite values.



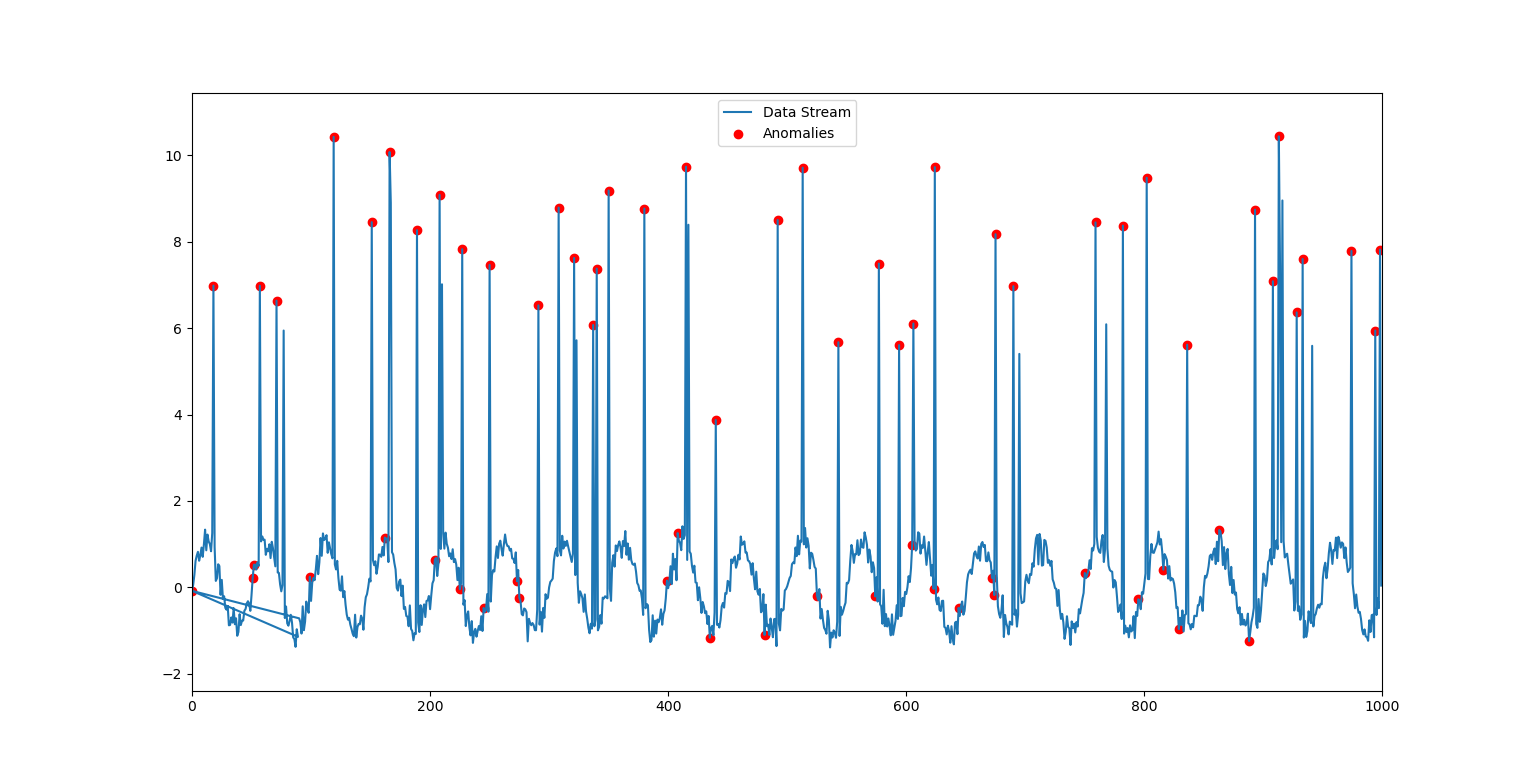
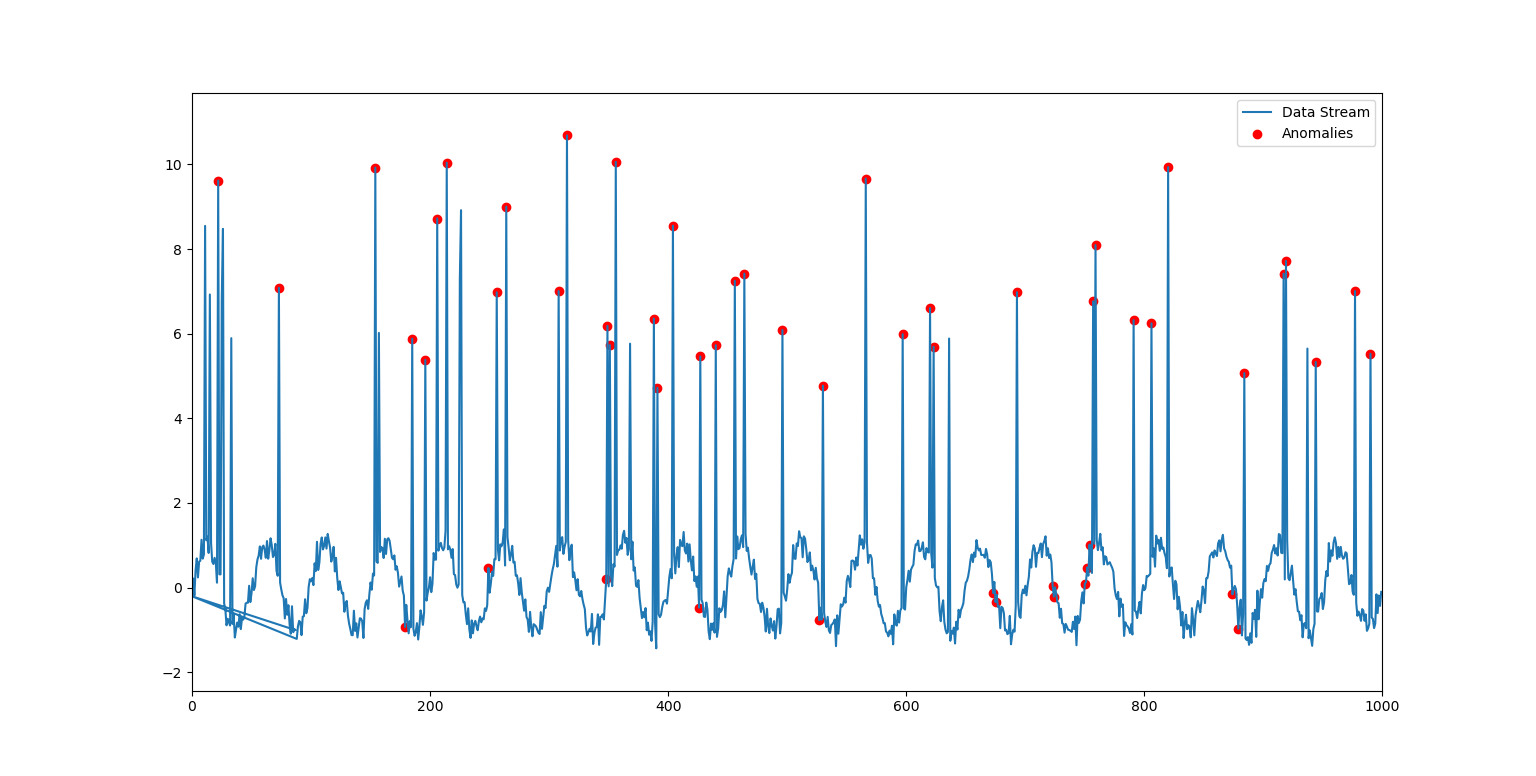
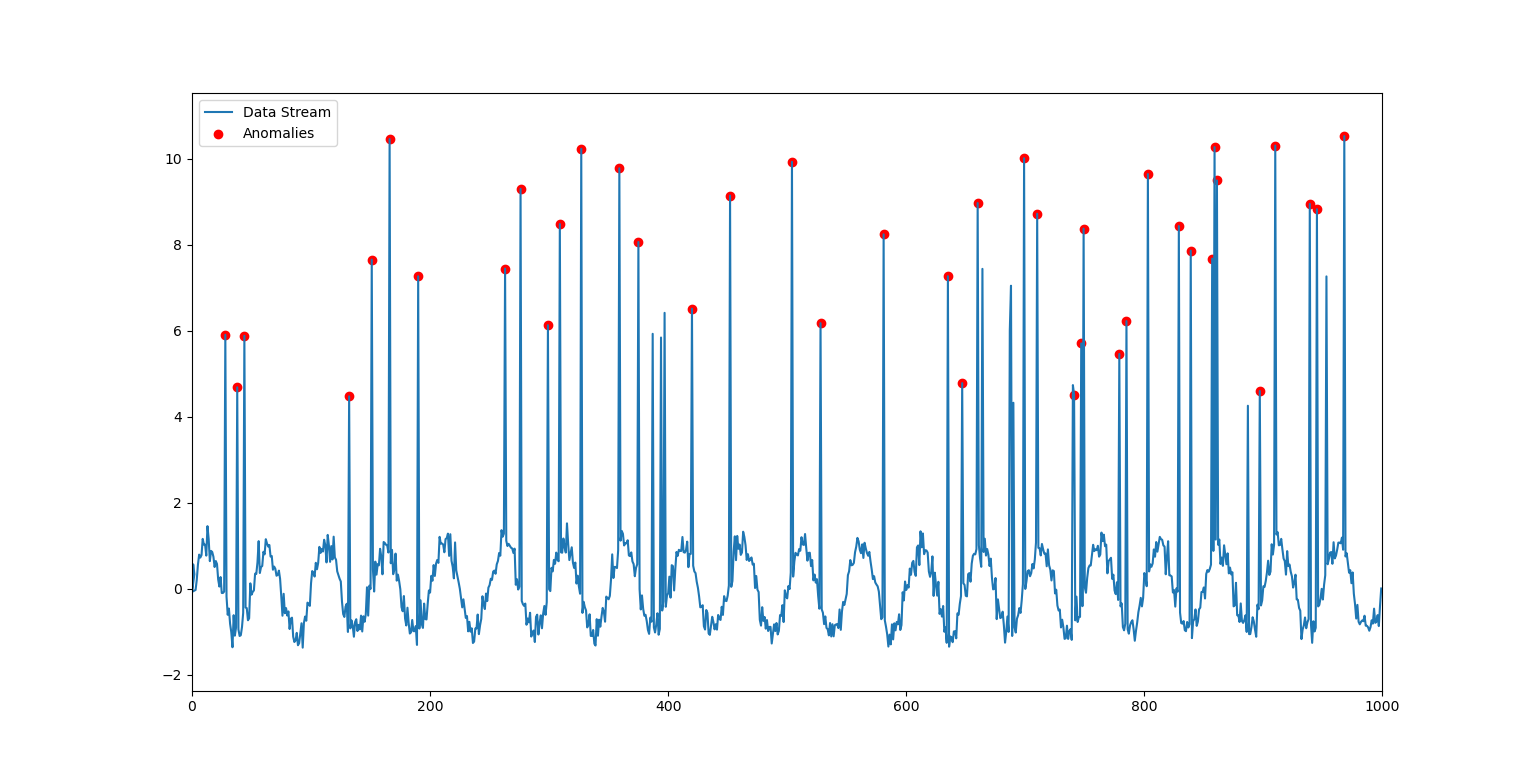
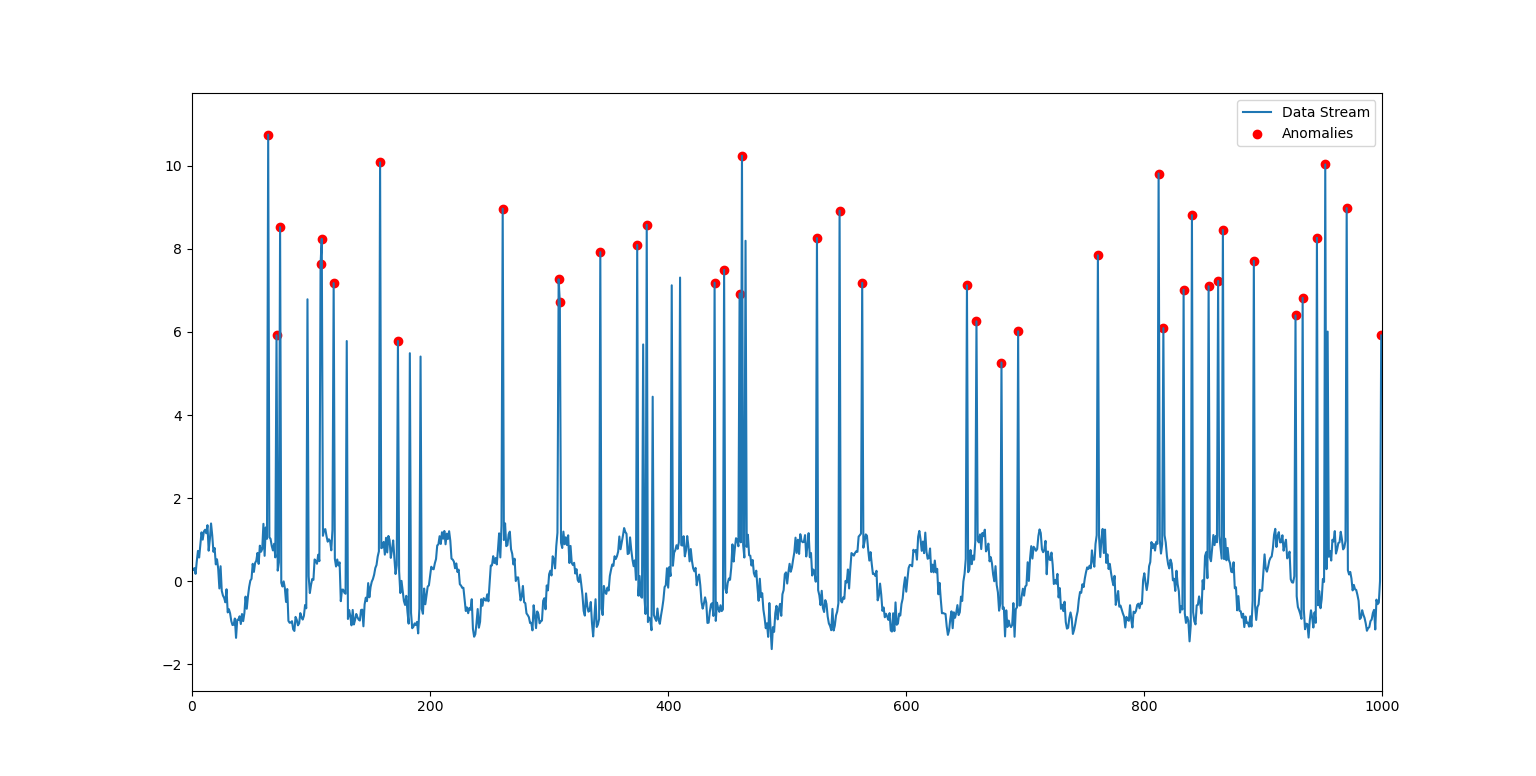
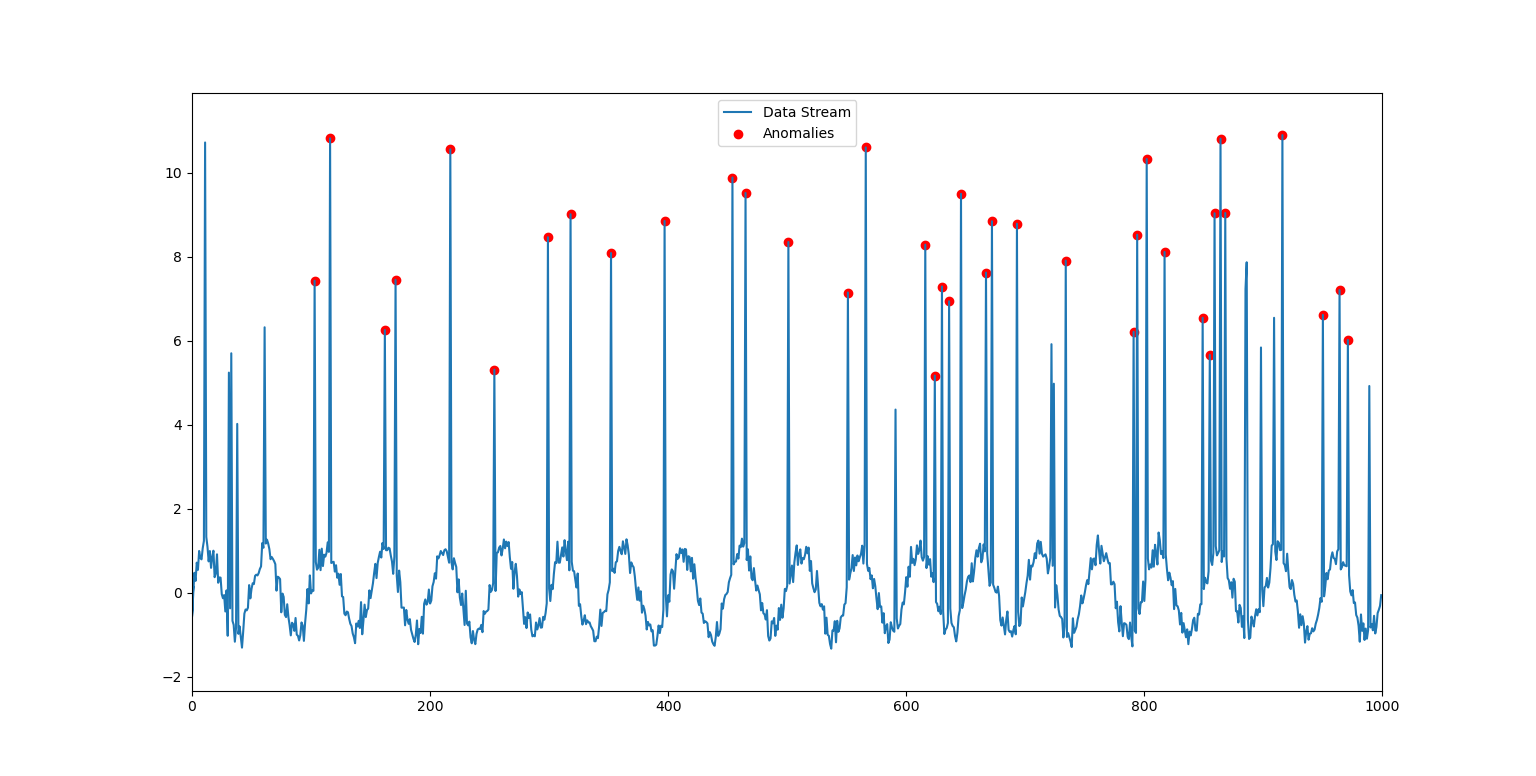
Step 3: Z-Score Anomaly Detector A Z-Score anomaly detector is implemented to flag anomalies based on deviations from the sliding window's mean and standard deviation.



Step 4: Real-time Plotting The data stream is visualized in real-time using matplotlib, with anomalies highlighted in red.

**Results**

The following images show the anomaly detection results for different window sizes. Window size 30 was found to provide the best balance between sensitivity and accuracy.

* **Window Size = 10**
* **Window Size = 20**
* **Window Size = 30**
* **Window Size = 40**
* **Window Size = 50**
* **Window Size = 100**