Efficient Stream Anomaly Detection

Objective:

The primary objective of this project is to develop an efficient Python script for real-time anomaly detection in a continuous data stream. This involves implementing an adaptive algorithm capable of handling concept drift and seasonal variations, creating a function to simulate realistic data streams, and designing a mechanism to flag anomalies in real-time. The project also requires optimising the algorithm for speed and efficiency, and building a simple visualisation tool to display the data stream and detected anomalies. Overall, this task aims to showcase the ability to create a robust, accurate system for identifying unusual patterns in streaming data, demonstrating technical skills and problem-solving abilities in a practical context relevant to fields such as financial transaction monitoring or system metrics analysis.

Method Employed:

The **Isolation Forest** algorithm is an effective choice for **real-time anomaly detection** in data streams. It works on the principle that anomalies are **rare** and **different**, making them easier to isolate in a random forest structure. In this implementation, the algorithm adapts to evolving data patterns by **periodically retraining** on a **sliding window of recent observations**. The use of an **adaptive threshold**, combining an **exponential moving average with standard deviation**, allows the model to adjust its **sensitivity to anomalies** as the **data distribution changes** over time. This approach is particularly effective for detecting both **sudden spikes** and **gradual shifts** in the data stream. The algorithm's efficiency stems from its ability to operate with **low time and memory complexity**, making it well-suited for real-time processing of continuous data streams. Its effectiveness is further enhanced by the **periodic model updates** and the **adaptive thresholding mechanism**, which help maintain high accuracy in detecting anomalies while **minimising false positives** in dynamic, non-stationary environments. A better explanation of my code is given below, which highlights what each class and function does:

1. **ImprovedAnomalyDetector class:**
    - Uses Isolation Forest from scikit-learn for anomaly detection.
    - Maintains a sliding window of data and scores.
    - Implements an adaptive threshold using an exponential moving average.
    - Periodically updates the model to adapt to changing data patterns.
2. **generate_complex_data() function:**
    - Creates synthetic data with trends, seasonality, and occasional anomalies.
    - Simulates a realistic data stream for testing the detector.
3. **update_plot() function:**
    - Handles the real-time updating of the visualisation.
    - Processes new data points, detects anomalies, and updates the plot.
    - Calculates and displays performance metrics (precision, recall, F1-score, etc.).

4. **Visualisation setup:**
    - Uses matplotlib to create an animated plot showing the data stream, detected anomalies, and a moving average.
    - Includes a text box displaying real-time statistics.

A snapshot of the real time anomaly detection animation is given below: