Anomaly Detection in a Data Stream

Overview

This program simulates a data stream that generates time series data with anomalies and concept drift, then visualizes the data in real-time while detecting anomalies using an Exponentially Weighted Moving Average (EWMA) technique. The goal is to detect abnormal patterns in the data and differentiate between normal variations and true anomalies.

Key Components

1. **Simulation Class**:

- This class generates a synthetic time series dataset with regular, seasonal, and noise-based patterns.
- It also introduces random anomalies with a small probability and incorporates concept drift, meaning the underlying distribution of the data slowly changes over time.
- Each data point is generated based on a regular pattern (linearly increasing), a seasonal pattern (sinusoidal), and random noise. Anomalies are occasionally added as spikes or dips.

2. **EWMAAnomalyDetector Class**:

- Implements an anomaly detection algorithm using Exponentially Weighted Moving Average (EWMA).
- EWMA provides a smooth moving average of the data stream and is used to detect deviations from expected values.
- If the deviation of the current data point from the EWMA exceeds a threshold, the data point is flagged as an anomaly.

3. **EWMAAnimation Class**:

- This class visualizes the data stream and the detection of anomalies in real-time using Matplotlib.
- Data points are plotted on a line graph, with predicted anomalies highlighted on the graph as blue scatter points.
- The animation updates as the data stream progresses, adjusting both the x- and y-axes dynamically to accommodate new data.

Features

- **Anomaly Detection**: The program detects anomalies in a stream of data in real-time using the EWMA technique.

- **Concept Drift**: The underlying data generation model shifts slightly over time, simulating a real-world scenario where the data distribution evolves.
- **Interactive Plotting**: The data stream is visualized using Matplotlib, and updates are shown in an animated plot where both normal and anomalous data points are visible.
- **Logging**: Warnings are logged for both missed anomalies and false positives, providing insight into how the model is performing.

How to Run

1. **Requirements**:

    - Python 3.x
    - NumPy
    - Matplotlib

    You can install the necessary packages from requirements.txt:

    ```
    pip install -r requirements.txt
    ```

2. **Running the Program**: Simply run the program in a Python environment:

    ```
    python sim.py
    ```

    The program will open a Matplotlib window showing an animated line plot of the data stream, with detected anomalies highlighted.

3. **Stopping the Program**: The program runs until it processes the specified number of data points. You can interrupt it at any time by pressing `Ctrl+C` or closing the plot window.

Customization

- **Simulation Settings**:
- `max_steps`: Adjust the total number of data points to generate.
- `anomaly_prob`: Change the likelihood of an anomaly occurring.
- `drift_rate`: Modify how quickly the regular data pattern shifts over time.
- `seasonal_drift_rate`: Modify the rate of change for the seasonal pattern.
- **Anomaly Detector Settings**:
- `alpha`: The smoothing factor for the EWMA. Lower values give more weight to past data.

- `threshold`: The deviation threshold for flagging anomalies. Higher values make the detector less sensitive to fluctuations.

- **Animation Settings**:

- `max_data_points`: The total number of data points shown in the animation.

- `interval`: The update interval for the animation in milliseconds.

Example Output

The output consists of an animated plot where:

- The line represents the data stream.
- Blue dots indicate points where anomalies are predicted by the EWMA detector.
- Warnings in the console inform you of missed anomalies and false positives.

Future Improvements

- **Adaptive Thresholds**: The anomaly detection threshold could be adapted dynamically based on recent behavior.
- **Different Detection Algorithms**: Implement additional anomaly detection methods such as moving averages or machine learning-based models.
- **Improved Visualization**: Add more interactive features such as zooming or pausing the animation to investigate certain points.