Efficient Data Stream Anomaly Detection

Project Description

This project implements a Python script for real-time anomaly detection in a simulated data stream. The script detects unusual patterns or outliers (anomalies) using the **Z-Score anomaly detection algorithm**. The data stream can represent various metrics like financial transactions or system metrics.

Features

- **Algorithm Selection**: The Z-Score method is used for anomaly detection. It adapts to changes in data by flagging points that deviate significantly from the current norm (in terms of standard deviations).
- **Data Stream Simulation**: A function simulates real-time data with patterns, seasonal variations, and random noise. Occasional anomalies are introduced to test detection.
- **Anomaly Detection**: A sliding window is used to compute Z-scores for real-time anomaly detection, allowing the algorithm to detect significant deviations as data is streamed.
- **Efficiency**: The implementation is designed for real-time processing, optimized for speed with a sliding window mechanism.
- **Real-Time Output**: Detected anomalies are printed in real-time during the streaming process.

Algorithm Explanation

Z-Score Anomaly Detection

```
def z_score_anomaly_detection(window, threshold=3):
   """
   Detect anomalies in the data stream using Z-Score.

   Parameters:
      window (list or np.ndarray): The current window of data.
      threshold (float): The Z-Score threshold to consider a point an anomaly.

   Returns:
      np.ndarray: Boolean array indicating whether each point is an anomaly.
   """
   z_scores = zscore(window)  # Compute Z-scores for the window
   anomalies = np.abs(z_scores) > threshold  # Flag points with Z-score >
threshold
   return anomalies
```

We use the **Z-Score anomaly detection** algorithm, which identifies data points that are far from the mean of a sliding window. The Z-Score tells us how many standard deviations a point is from the window's mean.

- Formula: [Z =] Where:
  - x is the data point,
  - μ is the mean of the window,
  - σ is the standard deviation of the window.

The absolute value of the Z-score is compared to a **threshold** (e.g., 3). If the absolute Z-score is greater than the threshold, the point is flagged as an anomaly.

Code Walkthrough

Data Stream Simulation (`data_stream_simulation`)

```python
def data_stream_simulation(size=1000, seasonal=True):
   """
   Simulates a data stream with noise and occasional anomalies.

   Parameters:
      size (int): The number of data points to generate.
      seasonal (bool): Whether to include a sinusoidal pattern in the data.

   Returns:
      numpy.ndarray: Simulated data stream with potential anomalies.
   """
   t = np.linspace(0, 100, size)  # Generate time steps
   data = 10 * np.sin(t) if seasonal else 5 * np.ones(size)  # Sinusoidal or flat
trend
   noise = np.random.normal(0, 2, size)  # Add random noise
   anomalies = np.random.choice([0, 20], size=size, p=[0.98, 0.02])  # 2% chance
of anomalies
   return data + noise + anomalies  # Combine the normal data with anomalies
```

The Z-Score detection works by calculating how far each data point in the window is from the mean. If the deviation is greater than the threshold, the point is flagged as an anomaly.

Real-Time Anomaly Detection (detect_anomalies_in_stream)

```python
def detect_anomalies_in_stream(stream_size=1000, window_size=50, threshold=3):
    """
    Simulate a real-time data stream and detect anomalies using a sliding window.

    Parameters:
        stream_size (int): The total size of the data stream.
        window_size (int): The size of the sliding window for anomaly detection.
        threshold (float): The Z-score threshold for anomaly detection.

    Returns:
        list: A list of detected anomalies (index, value).
    """
    stream = data_stream_simulation(size=stream_size)  # Generate data stream
    data_window = deque(maxlen=window_size)  # Sliding window to store recent
data
    anomalies = []  # List to store detected anomalies

    # Initialize the real-time plot
    plt.ion()  # Enable interactive mode for live updates
    fig, ax = plt.subplots(figsize=(10, 5))  # Create a plot figure
    ax.set_title("Real-Time Data Stream with Anomalies")
    ax.set_xlabel("Time")
    ax.set_ylabel("Data Value")
    ax.set_ylim(-30, 40)

    # Line objects for data stream and anomalies
    line, = ax.plot([], [], lw=2, label="Data Stream")
    anomaly_points, = ax.plot([], [], 'ro', label="Anomalies")  # Red dots for
anomalies

    # Lists to store data for plotting
    x_data = []
    y_data = []
    anomaly_x = []
    anomaly_y = []

    for i, value in enumerate(stream):  # Loop over the data stream
        data_window.append(value)  # Add the latest value to the window
        # Update plot data
        x_data.append(i)
        y_data.append(value)

        if len(data_window) == window_size:  # Check if window is full
            window_data = np.array(data_window)
            anomaly_flags = z_score_anomaly_detection(window_data, threshold)

            if anomaly_flags[-1]:  # If the most recent point is an anomaly
                anomaly_x.append(i)
                anomaly_y.append(value)
                anomalies.append((i, value))  # Store anomaly details

            # Update the plot with new data
            line.set_data(x_data, y_data)  # Update the data stream line
            anomaly_points.set_data(anomaly_x, anomaly_y)  # Update the anomaly
points
            ax.set_xlim(0, max(x_data) + 1)  # Adjust the x-axis range

            plt.legend()
            plt.draw()  # Redraw the plot
            plt.pause(0.01)  # Short pause for live plotting

        time.sleep(0.1)  # Simulate delay for real-time stream

    plt.ioff()  # Turn off interactive mode once the stream is done
    plt.show()

    return anomalies  # Return the list of detected anomalies
```

This function simulates the data stream in real-time and detects anomalies using a sliding window of the last `window_size` data points. When an anomaly is detected, it prints the index and value of the anomalous point.

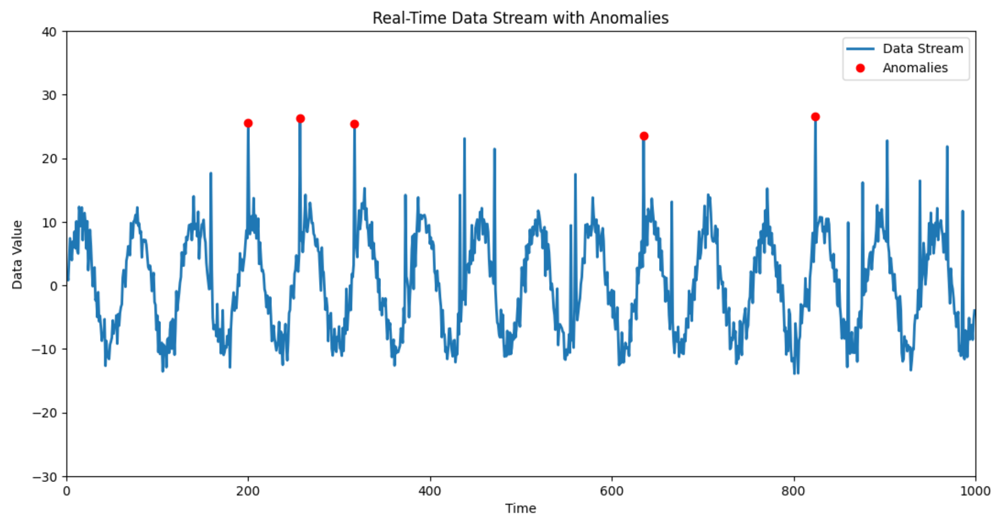
Conclusion

This project provides a simple, efficient method to detect anomalies in a data stream using the Z-Score algorithm. It's designed for real-time data and can easily be adapted to handle various types of data streams.

Summary of the `README.md`:

- **Project Description**: High-level overview of the project.
- **Algorithm Explanation**: Description of Z-Score anomaly detection.

- **Code Walkthrough**: Explanation of each key function with comments.
- **Instructions**: Steps to install libraries and run the script.
- **Error Handling**: Mention of error-handling mechanisms.
- **Conclusion**: Wrap-up of the project.


Real-Time Data Stream with Anomalies
Data Stream
Anomalies
Data Value
Time