Project Purpose

In this project, I aimed to develop a system that could be applied in real-world scenarios, specifically focusing on pressure measurement. The core scenario I envisioned involves monitoring pressure values, where an alert is generated if the pressure falls below 1, indicating a need to increase the pressure. Conversely, if the pressure exceeds 6, the system indicates the necessity to decrease it.

I designed the normal pressure levels to range between 3 and 4. To illustrate this scenario effectively, I frequently sent values above 6 and below 1 from the server to the client. For the project to function properly from the outset, it is essential to transmit 30 data points from the server to the client. Subsequently, these incoming values are displayed on a live table, highlighting which values are normal (in blue) and which are abnormal (in red).

Server Code Documentation

1. Project Overview

- Purpose: A SocketIO server that generates and streams real-time data to connected clients.
- Use Cases: Suitable for projects involving data visualization, real-time monitoring, and analysis.

2. Requirements

- Python Version: 3.x
- Necessary Libraries:
  - Flask: Web framework for building web applications.
  - Flask-SocketIO: Provides WebSocket support for real-time communication.

3. Installing the Libraries

You can install the required libraries using the following command:

```
pip install Flask Flask-SocketIO
```

4. Explanation of Server Code

generate_data():

Description: Continuously generates random data values. Depending on the current time, it produces values in three different ranges:

- Between 6 and 7 approximately every 20 seconds.
- Between 0 and 1 approximately every 20 seconds.
- Between 3 and 4 for all other times.

Purpose: Simulates a data stream for real-time analysis.

handle_connect()

Description: Triggered when a client connects to the WebSocket server.

Purpose: Starts the background data generation task when a client connects and prints a connection message.

5. Running the Server

To start the server, execute the following command in your terminal:

```
python server.py
```

6. Testing

- To verify that the server is running, you can visit http://127.0.0.1:5000 in your web browser or connect a client that listens to the data_update event.

Client Code Documentation

1. Project Overview

- **Purpose**: A client application that receives data from a SocketIO server, detects outliers, and visualizes the results.
- **Use Cases**: Suitable for real-time data analysis and visualization projects.

2. Requirements

- **Python Version**: 3.x
- **Necessary Libraries**:
  - socketio: WebSocket client
  - matplotlib: For data visualization
  - math: For mathematical operations

3. Installing the Libraries

```
pip install python-socketio matplotlib
```

4. Explanation of Client Code

add_data_point(data, label)

Description: Adds a new data point and its corresponding label to the buffers. Maintains the size of the buffers to the last n data points.

Purpose: Ensures that only a fixed number of the most recent data points are stored for processing and analysis.

euclidean_distance(x1, x2)

Description: Calculates the Euclidean distance between two points.

Purpose: Used in the KNN algorithm to measure the distance between the new data point and existing data points in the buffer.

handle_data(data)

Description: Handles incoming data updates from the WebSocket server. It processes the received value, detects outliers, and stores the data accordingly.

Purpose: Manages real-time data analysis by adding data points to the buffer and determining if they are normal or outliers.

update_plot(frame)

Description: Updates the live plot with the most recent normal data and outliers. It visualizes the data points using scatter plots.

Purpose: Provides real-time visualization of the data stream, allowing users to see trends and outliers as they occur.

start_socketio()

Description: Connects the SocketIO client to the WebSocket server.

Purpose: Establishes the communication channel necessary for receiving real-time data updates.

plt.show()

- **Description**: Displays the plot window for the data visualization.
- **Purpose**: Initiates the graphical interface to visualize the real-time data stream.

5. Running the Client

- To start the client, execute the following command in the terminal:
  - python client.py

6. Testing

- Once the client is running, it will visualize the incoming data and identify outliers in real-time.

Additional Concepts

Euclidean Distance

Euclidean distance is a measure of the straight-line distance between two points in a multi-dimensional space. It is calculated using the formula:

$$d = \sqrt{[(x_2 - x_1)^{**}2 + (y_2 - y_1)^{**}2]}$$

In the context of this project, it is used to measure the distance between a new data point and existing data points in the buffer to determine whether it is an outlier.

K-Nearest Neighbors (KNN) Algorithm

The K-Nearest Neighbors (KNN) algorithm is a simple, non-parametric classification algorithm. It classifies a new data point based on the majority class of its k nearest neighbors in the feature space. In this project, KNN is used for outlier detection by analyzing the distances of a new data point to its nearest neighbors. If the average distance of the k nearest neighbors exceeds a defined threshold, the data point is classified as an outlier.

Summary

This documentation provides a comprehensive overview of the SocketIO server and client implementation, including their functionalities, dependencies, and operational instructions. Additionally, it introduces key concepts like Euclidean distance and the KNN algorithm that underpin the outlier detection process in the client application.