

ISEQL-Glucose Analyzer

Lorenzo Tucceri Cimini

August 2, 2024

Abstract

Managing diabetes requires continuous monitoring of blood glucose levels to prevent complications. This project focuses on analyzing time-series glucose data to model different glycemic conditions: normal, extremely high, high, low, and extremely low. Using Python tools like Pandas and NumPy, we perform data cleaning, segmentation into intervals, and employ ISEQL for complex event detection. Our proposed architecture includes a Data Crawler for raw data extraction, an Interval Action Detector for event annotation, and a High-Level Event Detector for complex event identification. Results highlight effective glucose level analysis for diabetes management, emphasizing pattern recognition and anomaly detection across various time intervals. This approach enhances understanding of glucose behaviors critical for personalized treatment strategies.

1 Introduction

Continuous monitoring of blood glucose is essential for managing diabetes, a chronic condition that necessitates precise control of blood sugar levels to prevent complications. This project focuses on analyzing a dataset of glucose values recorded over time, with the aim of identifying and modeling time intervals associated with different glycemic conditions: **normal**, **high**, **extremely high**, **low**, and **extremely low**.

Utilizing data analysis tools in Python, such as Pandas for data manipulation and NumPy for numerical operations, we will extract meaningful insights from the raw data. Additionally, we will employ ISEQL (**Interval-based Surveillance Event Query Language**) to formally define and detect complex events in health data. ISEQL is particularly suitable for this analysis as it allows for the combination of low-level events into more sophisticated structures, enhancing the expressiveness and efficiency of the event detection process.

The analysis will proceed with data cleaning and preparation, followed by segmentation into relevant time intervals. We will then use ISEQL to define and detect complex events, concluding with a discussion of the results and their implications for diabetes management.

We will work with a CSV file containing glucose data from the past 90 days, specifically from February 13, 2024, to May 12, 2024. Our analysis will focus on:

- **Simple High-Level Event:**
 - **Time Swing:** Identifying individual glycemic events and significant changes in glucose levels, highlighting rapid fluctuations between different glucose categories.
 - **Too Long Glucose Anomalies:** Measuring the time spent in anomalous glucose ranges, which may indicate deteriorating health conditions.
- **Aggregated High-Level Event:** Aggregating multiple events to provide a comprehensive view of glucose level anomalies. This includes:
 - **Too Frequent Glucose Anomalies:** Assessing the frequency of anomalous glucose events within the observation period.
 - **Too Frequent Time Swings:** Evaluating the frequency of rapid glucose level changes within a specific time frame, indicating potential issues with glucose stability.
 - **Time Swing With Too Long Glucose Anomalies:** Analyzing swings where one of the intervals before or after the swing has a Too Long Glucose Anomalies, indicating extended instability.

These methods will help us identify significant correlations and trends in the data, which are crucial for effective diabetes management. By distinguishing between simple and aggregated high-level events, we can better understand and respond to complex glucose level patterns, ultimately improving patient care and outcomes.

2 State of Art

Diabetes represents a global health challenge, with artificial intelligence (AI) revolutionizing disease management. AI, leveraging its ability to analyze vast amounts of data and identify complex patterns, promises significant improvements in diagnosis, personalized treatment, and diabetes management. In this section, we will explore various tools used for studying and analyzing this disease, including advanced monitoring devices like the Dexcom G7, which provide essential data to optimize daily diabetes management.

- Dexcom G7
- DiaHealth
- Blood Glucose Level Prediction with Feedforward Neural Network

2.1 Dexcom G7

The Dexcom G7 represents one of the latest advancements in CGM technology, distinguished by its effectiveness and convenience. The device provides glucose level updates every 5 minutes, enabling users to closely monitor glycemic fluctuations throughout the day. A detailed graph displays glucose trends over the past 24 hours, facilitating the identification of significant glycemic peaks and valleys through customizable thresholds.

Additionally, Dexcom Clarity offers a long-term overview of glucose trends, allowing users to analyze data from the past 90 days through detailed charts and statistics on glycemic quality. This feature supports not only informed daily management but also preventive strategies to reduce the risk of long-term diabetes complications.

In-depth studies have shown that accurate and continuous glucose monitoring can lead to significant improvements in health management by early identification of issues such as prolonged periods of hyperglycemia or rapid shifts between hypoglycemia and hyperglycemia.

In our case, we use data about glucose provided by Dexcom.

2.2 DiaHealth

DiaHealth is an innovative mobile application designed to enhance daily diabetes management through a comprehensive range of integrated features. This application stands out in the landscape of digital diabetes solutions by integrating artificial intelligence to offer personalized suggestions and support more effective disease management.

Key features of DiaHealth include:

- **Blood Glucose Monitoring:** Allows users to regularly monitor their blood glucose levels through an intuitive and user-friendly interface.
- **Diet and Exercise Plans:** Provides personalized plans for diet and physical activity, helping users maintain a healthy and balanced lifestyle.
- **Health Data Logging:** Enables users to record and monitor various health parameters such as blood glucose, blood pressure, HbA1c, cholesterol, and weight.
- **Smart Suggestions:** Uses artificial intelligence to analyze user-entered data and provide personalized suggestions to improve diabetes control.
- **Diabetes Education:** Offers educational resources to help users better understand their condition and adopt healthy behaviors.

In our context, it is an excellent starting point to leverage High-Level Events and provide suggestions on glucose management, physical activity, and diet.

2.3 Blood Glucose Level Prediction with Feedforward Neural Network

Feedforward neural networks (FNNs) are applied for predicting blood glucose levels using historical data from continuous monitoring systems (CGM) such as Dexcom G7. These models are designed to process sequences of past glucose measurements. The architecture of an FNN includes an input layer for previous measurements, hidden layers that extract features and learn temporal dependencies, and an output layer that predicts future glucose levels over a specific time interval, such as 15 minutes.

FNNs have demonstrated high accuracy in predicting glucose levels, as evidenced by low Root Mean Square Error (RMSE) values. This indicates the model’s ability to effectively manage individual variations in diabetes management factors such as insulin sensitivity, meal timing, and physical activity levels. Thanks to these precise predictions, FNNs enable proactive management of glucose fluctuations, significantly reducing the incidence of hyperglycemic and hypoglycemic episodes.

This study could provide a deeper understanding of high-level events by predicting glucose levels and the possible events that follow.

3 Overview and Formalization

The overall architecture of the glucose level monitoring system we propose is shown in Figure 1. It consists of three levels: a *Data Crawler*, an *Interval Action Detector*, and a *High-Level Event Detector*.

The *Data Crawler* extracts data from a continuous glucose monitoring sensor, specifically the Dexcom. This sensor releases a dataset with blood glucose level measurements every 5 minutes. The raw events thus correspond to these periodic glucose level measurements. Consequently, this level is highly dependent on the application domain and allows us to obtain data with specific timestamps. Clearly, the *Data Crawler* must include all the necessary components to extract and process this timestamped data, such as a library to collect and manage data from the Dexcom.

In the next step, the *Interval Action Detector* takes the raw events and creates simple events whose format is largely independent of the application domain. This separates the High-Level Event Detection from the technical details of the raw events. As a result, this level produces a set M of mid-level annotations referring to intervals of glucose level readings, with timestamps, corresponding to the mid-level predicates Pr stored in the knowledge base (Figure 4). Additionally, the events generated by this level already contain some aggregated data, thus simplifying High-Level Event Detection.

Finally, at the highest level, a user can construct complex events of real interest using mid-level events and simpler high-level events as building blocks. We also provide a graphical user interface to formulate high-level events. Consequently, the *High-Level Event Detector* takes a set E of event patterns and

determines if any of these events occur in M .

In summary, compared to the approaches listed in Section 2, ours presents the following advantages:

- Generalizable to heterogeneous domains (Figure 1);
- Flexible in defining new event patterns, thanks to the robust support of an intelligent graphical user interface (Figure 1);
- Adaptable whenever a new low-level event representing a new glucose level pattern is detected by the *Data Crawler*; it is immediately and easily mapped to higher levels, and thus exploitable for defining new event patterns;
- Effective and efficient in detecting the occurrences of events of interest;
- Combinable with machine learning approaches; we could learn event patterns through machine learning approaches, then easily define and detect their occurrences;
- Significantly improves glucose level analysis;
- In addition to the aforementioned advantages, our system introduces novel high-level event types that are not commonly available in other applications. For example, we have defined events like "Time Swing" and its variants, which identify specific patterns in glucose level fluctuations over time. These new high-level events offer a more nuanced understanding of glucose level behavior, enhancing the monitoring and management of blood glucose levels. The inclusion of these unique event patterns significantly broadens the analytical capabilities of our system.

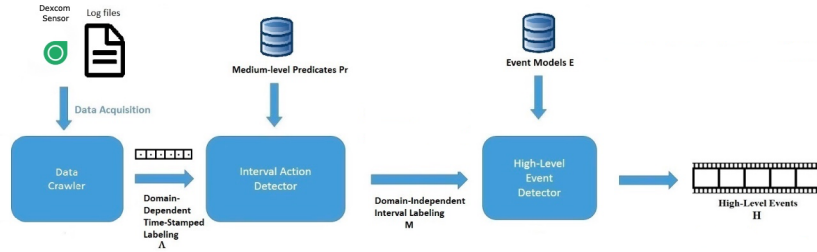


Figure 1: Overall Architecture

3.1 Data Crawler

To better understand various glucose level events, we use symbolic notation to represent each type of event and its corresponding thresholds. The following table shows the correspondence between symbols, events, and thresholds (i.e., the glucose level domain):

Symbol	Event	Threshold
a	Extremely High	≥ 250
b	High	$180 \leq x < 250$
c	Normal	$80 < x < 180$
d	Low	$55 < x \leq 80$
e	Extremely Low	≤ 55

Table 1: Mapping of Events and Thresholds with Symbols (S))

The Data Crawler will associate events in Table 1 with glucose levels extracted from the dataset. Here we introduce the first definition.

Definition 1 (Timestamp Labeling of Glucose Level). In the domain of glucose levels, timestamp labeling is a tuple $\lambda = (s, ts, gl)$, where $\lambda.s \in S$ (Table 1) represents the state associated with the event, $\lambda.ts$ is the timestamp related to the event state, and $\lambda.gl$ represents the glucose level at that specific moment. A timestamp labeling Λ is a finite set of entries with timestamps λ_i ordered by timestamp.

In the second table, we display the result from the Data Crawler, which includes only the columns of interest: "Symbol," "Timestamp," and "Glucose Level." These data represent glucose levels over time, as shown in Table 2.

Symbol	Timestamp	Glucose Level
c	2024-02-13T00:16:32	120
c	2024-02-13T00:21:33	123
b	2024-02-13T01:31:32	205
b	2024-02-13T01:36:33	202
c	2024-02-13T02:05:32	160

Table 2: Glucose Level Domain

3.2 Interval Action Detector

In this section, we delve into Interval Action Detection, operating at an intermediate level where we work with temporal intervals. After labeling events based on their thresholds, we organize the data into temporal intervals.

To optimize the selection of intervals of interest, we use the Offline Interval Action Detection algorithm. This algorithm is particularly useful for managing

and selecting non-overlapping intervals, thus improving the efficiency of temporal analysis.

Offline Interval Action Detection Algorithm: The Offline Interval Action Detection algorithm selects the maximum subset of non-overlapping intervals from a set of temporal intervals based on specific actions or events. It is particularly effective in contexts where it is necessary to identify and analyze events or actions occurring at distinct points in time, avoiding temporal overlaps.

Algorithm 1 Offline Interval Action Detection

Data: *glucose_data*: DataFrame containing data on glucose levels

Input: *extreme_high_threshold* \leftarrow 250, *high_threshold* \leftarrow 180, *low_threshold* \leftarrow 80, *extreme_low_threshold* \leftarrow 55

Result: Time intervals associated with each glucose level represented as a list of tuples.

```
1 begin
2 | intervals  $\leftarrow$  []
3 | events  $\leftarrow$  []
4 | foreach row in glucose_data do
5 | | timestamp  $\leftarrow$  row['Date and hour (AAAA-MM-GGThh:mm:ss)']
6 | | glucose_value  $\leftarrow$  row['Glucose level (mg/dL)']
7 | | if glucose_value == "Basso" then
8 | | | event, symbol  $\leftarrow$  'extreme_low', 'd'
9 | | end
10 | | else
11 | | | glucose_level  $\leftarrow$  int(glucose_value)
12 | | | if glucose_level  $\geq$  extreme_high_threshold then
13 | | | | event, symbol  $\leftarrow$  'extremely_high', 'a'
14 | | | end
15 | | | if glucose_level  $\geq$  high_threshold then
16 | | | | event, symbol  $\leftarrow$  'high', 'b'
17 | | | end
18 | | | if glucose_level  $\leq$  extreme_low_threshold then
19 | | | | event, symbol  $\leftarrow$  'extremely_low', 'e'
20 | | | end
21 | | | if glucose_level  $\leq$  low_threshold then
22 | | | | event, symbol  $\leftarrow$  'low', 'd'
23 | | | end
24 | | | else
25 | | | | event, symbol  $\leftarrow$  'normal', 'c'
26 | | | end
27 | | if intervals is empty or intervals[-1][3]  $\neq$  event then
28 | | | intervals.append((symbol, timestamp, timestamp, event))
29 | | end
30 | | else
31 | | | intervals[-1]  $\leftarrow$  (intervals[-1][0], intervals[-1][1], timestamp, intervals[-1][3])
32 | | end
33 | | events.append((symbol, event, glucose_level if glucose_value  $\neq$  "Basso" else "extreme_low"))
34 | end
35 | return intervals, events
36 end
```

Definition 2 (Interval Labeling). A Medium-Level annotation is a tuple $\mu = (pred, p, tss, tse)$, where $\mu.pred \in S$ is the Medium-Level predicate describing a Medium-Level event. Here, $\mu.p$ specifies the patient associated with the state, $\mu.tss$ is the timestamp related to the beginning of the event, and $\mu.tse$ pertains to the end of the event.

The following table illustrates an example of how these events can be mapped into specific time intervals for the patient Lorenzo T. C.:

Pred	Patient	Start Time	End Time
Normal	Lorenzo T. C.	13/02/2024 00:01:32	13/02/2024 10:46:33
High	Lorenzo T. C.	13/02/2024 10:51:33	13/02/2024 11:06:32
Normal	Lorenzo T. C.	13/02/2024 11:11:33	13/02/2024 12:11:33
High	Lorenzo T. C.	13/02/2024 12:16:33	13/02/2024 13:06:33
Normal	Lorenzo T. C.	13/02/2024 13:11:33	13/02/2024 17:31:33
High	Lorenzo T. C.	13/02/2024 17:36:33	13/02/2024 19:41:34
Normal	Lorenzo T. C.	13/02/2024 19:46:33	13/02/2024 20:26:34
Low	Lorenzo T. C.	13/02/2024 20:31:34	13/02/2024 20:46:34
Normal	Lorenzo T. C.	13/02/2024 20:51:33	13/02/2024 21:46:34
Low	Lorenzo T. C.	13/02/2024 21:51:34	13/02/2024 22:16:33

Table 3: Temporal Intervals of Events for Lorenzo T. C.

In addition to time intervals, it is useful to consider the duration of each interval to better analyze the behavior of blood glucose events. To find these data, we simply calculated $\mu.tse - \mu.tss$, where $\mu.tse \geq \mu.tss$. An interval labeling \mathcal{M} is a finite set of Medium-Level annotations μ_i . Specifically, $\forall \mu \in \mathcal{M}, \mu.pred \in \mathcal{P}_r, \mu.tss, \mu.tse \in \mathbb{N}, \mu.tss \leq \mu.tse$.

The following table shows the duration of intervals for the patient Lorenzo T. C.:

Pred	Patient	Duration
Normal	Lorenzo T. C.	10 hours, 45 minutes, 1 second
High	Lorenzo T. C.	14 minutes, 59 seconds
Normal	Lorenzo T. C.	1 hour, 0 minutes, 0 seconds
High	Lorenzo T. C.	50 minutes, 0 seconds
Normal	Lorenzo T. C.	4 hours, 20 minutes, 0 seconds
High	Lorenzo T. C.	2 hours, 5 minutes, 1 second
Normal	Lorenzo T. C.	40 minutes, 1 second
Low	Lorenzo T. C.	15 minutes, 0 seconds
Normal	Lorenzo T. C.	55 minutes, 1 second
Low	Lorenzo T. C.	24 minutes, 59 seconds

Table 4: Interval Duration for Lorenzo T. C.

To further analyze daily patterns and trends in blood glucose levels, we have also created Medium-Level events by aggregating data into daily intervals.

This new event, referred to as *daily*, covers the period from midnight (00:00) to 11:59 PM of each day. This approach allows us to perform daily analyses of the events and identify any significant changes or trends that may occur from one day to the next. By examining these *daily* intervals, we can gain insights into the patient’s glucose control over time and make informed decisions regarding treatment and management strategies.

3.3 High-Level Event Detection

In this section, we focus on High-Level Event Detection, concentrating on analyzing time intervals associated with anomalous blood glucose events. We categorize these events into two distinct levels: Simple High-Level Event and Aggregated High-Level Event. The analysis covers a period of 90 days, specifically from February 13, 2024, to May 12, 2024, and considers both high and low glucose events. This analysis is based on the events identified at the medium level, as defined in Definition 2 (Interval Labeling).

Definition 3 (High-Level Event Detection): A high-level event is a tuple $o = (e, p, tss, tse, arg_1, \dots, arg_m)$, where $o.e \in E$ is the name of the detected high-level event, $o.p$ is the patient, $o.tss$ and $o.tse$ are the start and end times defining the event, and $o.arg_1, \dots, o.arg_m$ are the predicate arguments. The set of high-level events H is a finite sequence of high-level events o_i . Specifically, for each $o \in H$, $o.e \in E$, $o.tss, o.tse \in \mathbb{N}$, and $o.tss \leq o.tse$.

High-Level Event Detection, as i said before, is structured into two distinct levels: **Simple High-Level Event** and **Aggregated High-Level Event**.

3.3.1 Simple High-Level Event

In the Simple High-Level Event category, we identify individual events based on single occurrences of abnormal glucose levels.

1. **Time Swing:** This refers to a rapid change in blood glucose levels over a short period, such as a transition from hyperglycemia to hypoglycemia or vice versa. A potential *Time Swing* is defined as a change from high to low glucose levels or vice versa. However, to classify this change as a true *Time Swing*, we must verify that the duration of the change falls within a predefined threshold of two hours.

Example 1 (High-Level Events Simple - Time Swing):

$o = (\text{time_swing}, \text{Lorenzo T.C}, 2024-04-17 \text{ 08:00:00}, 2024-04-17 \text{ 08:50:00}, \text{high}, \text{low}, 00:50:00)$

In this example, the "Time Swing" event for patient Lorenzo T.C. starts on April 17, 2024, at 08:00:00 (end of the high event) and ends at 08:50:00 (start of the subsequent event) on the same day, with the glucose level changing from high to low in 50 minutes. In this case, it qualifies as a true *Time Swing* since it occurs within the two-hour threshold.

The following table shows several examples of Time Swings.

First event	Second event	Duration TS	Event Time
high	low	00:50:00	2024-04-17 08:00:00-08:50:00
low	high	02:20:00	2024-03-01 11:40:00-14:00:00
high	low	01:04:59	2024-03-06 22:20:23-23:24:59

Table 5: Time Swings for patient Lorenzo T. C.

2. **Too Long Glucose Anomalies:** Measures the total time a patient’s glucose events remain outside the normal range, if it exceeds a predefined duration. Prolonged periods of abnormal glucose events may indicate deteriorating health conditions.

Example 2 (Aggregated High-Level Events - Too Long Glucose Anomalies):

$o = (\text{too_long_glucose_anomalies}, \text{Lorenzo T.C}, 2024-02-20\ 08:30:00, 2024-02-20\ 14:25:00, \text{high}, 0\ \text{days}\ 05:55:00).$

In this example, the "Too Long Glucose Anomalies" event for patient Lorenzo T.C. Several intervals of Too Long Glucose Anomalies were found, including one with high blood glucose lasting 5 hours and 55 minutes.

The following table shows various examples of Too Long Glucose Anomalies:

Event	Too Long Glucose Anomalies	Event Time
low	0 days 01:14:59	2024-02-17 07:45:00-08:59:59
high	0 days 05:55:00	2024-02-20 08:30:00-14:25:00
low	0 days 01:09:59	2024-02-25 18:20:00-19:29:59

Table 6: Too Long Glucose Anomalies for patient Lorenzo T. C.

3.3.2 Aggregated High-Level Event

In the Aggregated High-Level Event category, we aggregate data from multiple events to provide a broader view of glucose level anomalies. This level includes:

1. **Too Frequent Glucose Anomalies:** This High-Level Event counts the number of times a patient’s glucose events exceed or fall below the normal thresholds within a specified time frame. Frequent anomalies can reveal patterns that require further investigation. For example, analyzing *daily* intervals—covering the period from midnight (00:00) to 11:59 PM each day—can help identify if there are recurring high or low glucose events on a daily basis, highlighting potential issues with glucose management that may need attention.

Example 3 (Aggregated High-Level Events - Too Frequent Glucose Anomalies):

o = (too_frequent_glucose_anomalies, Lorenzo T.C, 2024-02-13 00:00:00, 2024-02-13 11:59:00, count_extremely_high, 1, count_high, 3, count_low, 2, count_extremely_low, 0, total_count, 6).

In this example, the "Too Frequent Glucose Anomalies" for the patient Lorenzo T.C. is analyzed over a time interval from February 13, 2024, at 00:00:00 to February 13, 2024, at 11:59:00. During this period, 3 episodes of high blood glucose, 1 of extremely high blood and 2 episodes of low blood glucose were found.

In the following table, we show the count of various examples of the Too Frequent Glucose Anomalies .

Extremely high	high	Low	Extremely low	Total count
1	3	2	0	6

Table 7: Too frequent Glucose Anomalies for patient Lorenzo T. C.

2. **Too Frequent Time Swings:** This metric evaluates the occurrence of Time Swings—rapid fluctuations in glucose levels—within a specified time period. It measures how frequently these swings happen, which can indicate issues with glucose stability. For instance, analyzing *daily* intervals, from midnight (00:00) to 11:59 PM each day, can help determine if rapid glucose fluctuations are a daily occurrence, potentially signaling a need for more intensive glucose monitoring and management.

Example 4 (Aggregated High-Level Events - Too Frequent Time Swings)):

o = (too_frequent_time_swings, Lorenzo T.C, 2024-04-22 00:00:00, 2024-04-23 11:59:00, Time Swing, 2, 07:30:00, 08:00:00, 00:30:00, high, low, 14:30:00, 14:50:00, 00:20:00, low, high).

In this scenario, we analyzed a time frame encompassing the entire day of April 22, 2024. During this period, we observed two notable Time Swings. The first Time Swing occurred from 07:30:00 to 08:00:00, lasting 30 minutes, and involved a shift from high to low glucose events. The second Time Swing happened later in the day, from 14:30:00 to 14:50:00, lasting 20 minutes, and also involved a significant change from low to high.

Count TS	Duration TS	Event Before and After TS
2	00:30:00 and 00:20:00	high to low and low to high
2	00:15:00 and 00:25:00	high to low and high to low

Table 8: Too Frequent Time Swings for patient Lorenzo T.C.

In this table, we illustrate examples showing the number of Time Swings (TS) within a day, the duration of each swing, and the type of events before and after each Time Swing. The "Count TS" column indicates the number of Time Swings observed, "Duration TS" provides the duration of each swing in hours, minutes, and seconds, and "Event Before and After TS" describes the transitions between glucose level categories (e.g., High to Low, Low to High) that occur before and after each Time Swing.

3. **Time Swing With Too Long Glucose Anomalies :** This metric captures Time Swings where either the interval immediately before or after the swing exhibits an abnormal duration. It reflects periods where glucose levels fluctuate significantly, and the abnormal duration of the surrounding intervals indicates prolonged instability. Extended durations of such swings can be concerning and may warrant medical attention.

Example 5 (Aggregated High-Level Events - Time Swing With Too Long Glucose Anomalies):

o = (time_swing_with_too_long_glucose_anomalies , Lorenzo T.C, 2024-04-17 06:00:00, 2024-04-17 08:50:00, high, low, 08:00:00, 08:50:00, 00:50:00, high duration: 02:00:00)

In this example, the "Time Swing" event for patient Lorenzo T.C. starts on April 17, 2024, at 08:00:00 (the end of the high glucose level event) and ends at 08:50:00 (the beginning of the next glucose level event) on the same day. The glucose level transitions from high to low over a duration of 50 minutes. This constitutes a valid *Time Swing* because it falls within the 2-hour threshold for duration. Additionally, we observe an *Too Long Glucose Anomalies* where the high glucose level event lasts for 2 hours, from 06:00:00 to 08:00:00.

The following table provides examples of Time Swing With Too Long Glucose Anomalies.

First event	Second event	Duration TS	Time Swing With Too Long
high	low	00:50:00	high event: 02:00:00
high	low	01:04:59	low event: 01:05:01
low	high	02:30:00	high event: 03:10:00

Table 9: Overlap between Time Swings with Too Long Glucose Anomalies for patient Lorenzo T.C.

4 Examples and Charts

To illustrate our approach, we present three examples with accompanying graphs extracted from the Dexcom G7 application.

4.1 Example 1: Normal Glucose Levels

In the first example, we analyze a time interval where glucose levels mostly remained within the normal range, with occasional high spikes. The graph below shows the glucose trends.

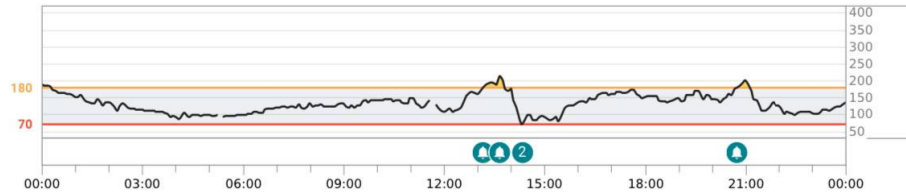


Figure 2: Normal Glucose Levels

4.2 Example 2: High Glucose Levels

In the second example, we examine a period where glucose levels were elevated, especially between 03:00 and 06:00. The graph depicts the high glucose peaks.

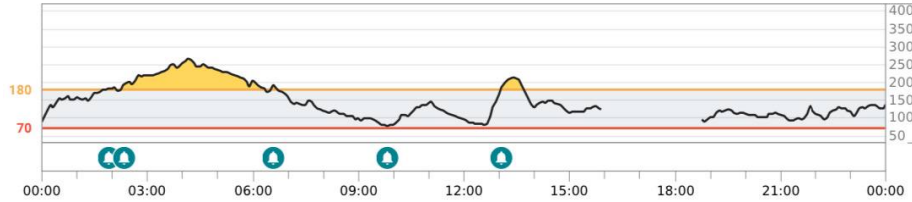


Figure 3: High Glucose Levels

4.3 Example 3: Low Glucose Levels

In the third example, we analyze a period with low glucose levels, mainly occurring during the nighttime hours. The graph highlights fluctuations below the normal threshold.

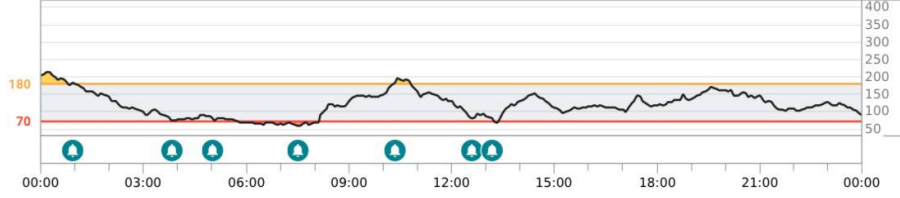


Figure 4: Low Glucose Levels

These examples demonstrate how analyzing glucose data through our approach can provide significant insights into variations in glucose levels over time, identifying periods of normalcy, high levels, and lows that can be crucial for diabetes management and overall health improvement.

5 ISEQL

We have proposed a system based on an extension of relational algebra, ISEQL (Interval-based Surveillance Event Query Language), enriched with powerful temporal operators. ISEQL significantly enhances the ability to monitor and interpret events in the context of continuous glucose monitoring. For example, it allows:

- Rapid identification of prolonged hyperglycemia or hypoglycemia episodes and quantification of their duration and frequency.
- Generation of detailed glucose pattern reports to assist doctors in making informed decisions on diabetes management.

The integration of ISEQL into our glucose level monitoring system provides a powerful tool for conducting complex temporal analyses more efficiently and robustly, improving the ability to monitor and interpret significant events in glucose data. Through ISEQL, it is also possible to express Allen’s interval relations to define high-level models more efficiently.

5.1 Interval Relations

We begin by defining an interval relation, which is simply a relation where each tuple is associated with the start and end points of an interval.

Definition 4 (Interval Relation). An interval relation R is a relation where each tuple $r = (A, Ts, Te)$ includes attributes $Ts \in \mathbb{N}$ and $Te \in \mathbb{N}$, denoting the start and end points of a closed interval. Here, A is the attribute related to glucose level events in that time interval. We define $size(r) = r.Te - r.Ts$, representing the duration of the event.

The interval denoted by the two points can be written as $[Ts, Te]$ or simply as T . We use a dot (\cdot) to refer to an attribute of a particular tuple: $r_i.T$ or $r_i.Ts$. Figure 2 lists the interval relations.

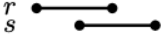
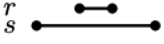

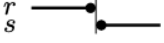
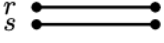
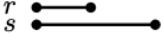

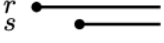
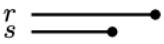
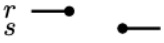
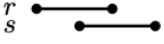
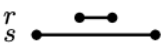
Relation	Doodle	Formal definition
OVERLAPS		$r.T_s < s.T_s < r.T_e < s.T_e$
DURING		$s.T_s < r.T_s \wedge r.T_e < s.T_e$
BEFORE		$r.T_e < s.T_s$
MEETS		$r.T_e = s.T_s$
EQUALS		$r.T_s = s.T_s \wedge r.T_e = s.T_e$
STARTS		$r.T_s = s.T_s \wedge r.T_e < s.T_e$
FINISHES		$s.T_s < r.T_s \wedge r.T_e = s.T_e$
START PRECEDING		$r.T_s \leq s.T_s < r.T_e$ $s.T_s - r.T_s \leq \delta$
END FOLLOWING		$r.T_s < s.T_e \leq r.T_e$ $r.T_e - s.T_e \leq \varepsilon$
BEFORE		$r.T_e \leq s.T_s$ $s.T_s - r.T_e \leq \delta$
LEFT OVERLAP		$r.T_s \leq s.T_s < r.T_e \leq s.T_e$ $s.T_s - r.T_s \leq \delta$ $s.T_e - r.T_e \leq \varepsilon$
DURING		$s.T_s \leq r.T_s \wedge r.T_e \leq s.T_e$ $r.T_s - s.T_s \leq \delta$ $s.T_e - r.T_e \leq \varepsilon$

Figure 5: Allen's and ISEQL Interval relations

5.2 CARDINALITY CONSTRAINTS

The **CARDINALITY CONSTRAINTS** specify the number of occurrences required for a particular interval within the result set of an operation. These constraints are useful for analyzing the frequency of events and filtering results to focus on specific intervals based on their occurrence.

For example, consider two interval relations R and S , and an interval join operation $R \bowtie S$ producing a result set ψ . We can apply cardinality constraints to limit the tuples included in ψ based on the frequency of intervals from either relation.

Definition 5 (Left Cardinality). Given two interval relations R and S ,

tuples $r_i \in R$ and $s_j \in S$, an interval join operator $\omega \in \text{Op}$ with $\psi = R\omega S$, a standard comparison operator θ' , and a left cardinality value $k \in \mathbb{N}$, the constraint is defined as:

$$R(\theta'k)\omega S = \{\langle r_i, s_j \rangle \in \psi \mid r_i \text{ appears in } \psi \text{ the number of times imposed by } \theta'k\}$$

where $\theta' \in \{<, \leq, =, \neq, \geq, >\}$.

Similarly, the Right Cardinality constraint can be applied to the intervals from the relation S . For example:

Definition 6 (Right Cardinality). Given two interval relations R and S , tuples $r_i \in R$ and $s_j \in S$, an interval join operator $\omega \in \text{Op}$ with $\psi = R\omega S$, a standard comparison operator θ' , and a right cardinality value $k \in \mathbb{N}$, the constraint is defined as:

$$R\omega S = \{\langle r_i, s_j \rangle \in \psi \mid s_j \text{ appears in } \psi \text{ the number of times imposed by } \theta'k\}$$

where $\theta' \in \{<, \leq, =, \neq, \geq, >\}$.

By applying both left and right cardinality constraints, the analysis can be further refined, ensuring the inclusion of only the most relevant data points. For instance, if we set $R(\geq 2)\omega S$, we require that intervals from R appear at least twice in the result set ψ , highlighting frequent occurrences within the specified analysis parameters.

In the context of glucose monitoring, CARDINALITY CONSTRAINTS can be used to:

- Analyze the frequency of specific events, such as anomalies or Time Swings, within a given time frame.
- Filter results to focus on intervals that meet or exceed a specified occurrence threshold.

5.3 DURING

The DURING operator in ISEQL is used to verify if one time interval occurs completely within another. This operator is useful for analyzing the presence and patterns of specific events relative to a broader time interval. In our case we use a variant of DURING like the DURING JOIN.

The DURING JOIN operator is defined by:

- $s.T_s \leq r.T_s \wedge r.T_e \leq s.T_e$: Interval r starts after or when interval s starts and ends before or when interval s ends.
- $r.T_s - s.T_s \leq \delta$: The difference between the start times of intervals r and s is within a threshold δ .
- $s.T_e - r.T_e \leq \varepsilon$: The difference between the end times of intervals s and r is within a threshold ε .

This means that the start time of interval r is greater than or equal to the start time of interval s , and the end time of interval r is less than or equal to the end time of interval s . Additionally, the difference between the start times of r and s is within a threshold δ , and the difference between the end times of s and r is within a threshold ε .

In the context of High-Level Event Detection, we can use the DURING JOIN operator to:

- Identify the occurrence of specific high-level events, such as "Too Frequent Time Swings," within a defined time period (e.g., daily, weekly, monthly, or yearly).
- Analyze other high-level events, including significant glucose fluctuations or other patterns, to understand their distribution within specified intervals.

The notation for this operator in ISEQL queries is DJ.

5.4 BEFORE

The BEFORE operator in ISEQL is used to express a temporal relationship between two intervals, indicating that one interval occurs completely before another interval, without any temporal overlap. This is particularly useful for analyzing the sequence of events in the context of continuous glucose monitoring, allowing the identification of causal or temporal relationships between different glycemic states.

Formally, the BEFORE operator can be defined as:

- $r.T_e \leq s.T_s$
- $s.T_s - r.T_e \leq \delta$

This means that the end time of interval r is less than or equal to the start time of interval s , and the difference between the start time of s and the end time of r is less than or equal to a threshold δ .

In our context, this allows us to visualize potential Time Swings, such as transitions from hyperglycemia to hypoglycemia and vice versa within a short time frame. The notation for this operator in ISEQL queries is Bef.

6 Modeling High-Level Events via ISEQL

The modeling of high-level events in continuous glucose monitoring is crucial for understanding diabetes management and for identifying critical situations that require intervention. ISEQL enables detailed temporal analysis to identify and characterize such events.

In this context, high-level events can be categorized into two main types: simple events and aggregated events. Simple High-Level Events are characterized by specific temporal attributes and are detected based on straightforward

conditions. These events provide fundamental insights into the occurrences and patterns in the glucose data.

Aggregated high-level events, on the other hand, involve the combination of multiple events to analyze broader patterns and trends. By examining the frequency and interactions of events over a specified period, aggregated events offer a more comprehensive view of the data, revealing complex phenomena that may not be apparent from individual events alone.

By leveraging ISEQL’s capabilities, both simple and Aggregated High-Level Events can be effectively modeled. This approach allows for a nuanced analysis of glucose data, helping to identify significant changes and trends, thereby supporting better management and intervention strategies in diabetes care.

6.1 Simple High-Level Events

Simple High-Level Events are fundamental units of analysis in continuous glucose monitoring. These events are characterized by distinct temporal patterns and are identified through specific conditions applied to the glucose data. By focusing on these basic events, we can gain immediate insights into glucose control and identify critical situations.

To detect these events, we use ISEQL operators such as `BEFORE`. The `BEFORE` operator identifies when one event occurs entirely before another event. These tools enhance the analysis and understanding of the temporal relationships between glucose events, allowing for more effective detection of rapid glucose level fluctuations.

6.1.1 Time Swing

The concept of *Time Swing* refers to rapid changes in glucose levels that occur over a short period of time. These changes can indicate instability in glucose control and potentially put the patient at risk of complications. Using ISEQL, we can identify these rapid oscillations between hyperglycemic and hypoglycemic intervals.

Detecting *Time Swings* is crucial for monitoring diabetes management. Fast fluctuations in glucose levels may indicate difficulty in maintaining stable control, increasing the risk of complications. The ability to quickly identify *Time Swings* allows for timely intervention and adjustments to the diabetes management plan, preventing episodes of hypoglycemia or hyperglycemia with immediate and serious consequences. Furthermore, accurate monitoring can help identify patterns that require therapeutic adjustments, thereby improving the overall quality of life for the patient.

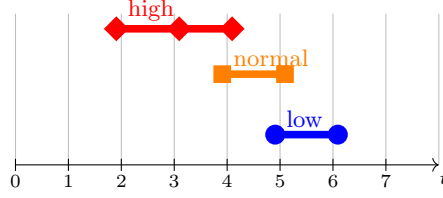


Figure 6: Time Swing from high to low in one hour

We used the *BEFORE* operator to verify that the first interval ends before the second interval begins, thereby identifying a *potential Time Swing*. Specifically, we checked that the hyperglycemic interval ends before the hypoglycemic interval begins (or vice versa), and that the duration between these two intervals is less than the specified time threshold using a δ of two hours to determine if it is indeed a *Time Swing* and not just a *potential Time Swing*. This approach allows us to effectively detect rapid oscillations in glucose levels. In this case we show an example of query to find a Time Swing from high to low or vice versa.

$$\pi_{M_1.Patient, M_1.end_time, M_2.start_time, M_1.event, M_2.event, (M_2.start_time - M_1.end_time)} \text{ AS Duration } \left(\begin{aligned} &\sigma_{M_1.Patient = M_2.Patient} \left(\begin{aligned} &\left(\sigma_{M_1.event = \text{"high"}}(M_1) \text{ Bef}(\delta : 2) \sigma_{M_2.event = \text{"low"}}(M_2) \right) \\ &\text{UNION} \\ &\left(\sigma_{M_1.event = \text{"low"}}(M_1) \text{ Bef}(\delta : 2) \sigma_{M_2.event = \text{"high"}}(M_2) \right) \end{aligned} \right) \end{aligned} \right)$$

6.1.2 Too Long Glucose Anomalies

The *Too Long Glucose Anomalies* events refers to the period during which a patient's glucose level remains outside normal thresholds. Monitoring these events is crucial to understanding how long the patient spends in potentially dangerous conditions.

Measuring the duration of anomalous events is essential for assessing the effectiveness of glycemic control. Too Long periods of hyperglycemia or hypoglycemia may indicate that the current treatment is inadequate and may require therapeutic adjustments.

Prolonged anomalous events are associated with an increased risk of acute and chronic complications. Monitoring their duration allows for timely intervention to reduce the risk of long-term complications.

Using ISEQL, we can determine the duration of each anomalous interval by subtracting the start timestamp (tss) from the end timestamp (tse). This data can help identify how long a patient has spent in hyperglycemia or hypoglycemia. A prolonged anomalous event with a duration exceeding the average may indicate an issue in diabetes management that requires attention.



Figure 7: Prolonged glucose anomalies with high glucose for four hours

To evaluate these events, we check if the duration exceeds the specific constraint by calculating the difference between the end timestamp and the start timestamp. The constraints used are:

- 45 minutes for "extremely high" events,
- 1.5 hours for "high" events,
- 30 minutes for "low" events,
- 10 minutes for "extremely low" events.

This approach allows for detailed analysis of prolonged and potentially harmful glucose fluctuations, providing critical insights into patient health.

$$\begin{aligned}
 &\pi_{M_1.Patient, M_1.start_time, M_1.end_time, M_1.event, (M_1.end_time - M_1.start_time)} \text{ AS Duration } \left(\right. \\
 &\quad \sigma \left(\right. \\
 &\quad \quad (M_1.event = \text{"extremely high"} \text{ AND } (M_1.end_time - M_1.start_time) \geq 45 \text{ minutes}) \text{ OR } \\
 &\quad \quad (M_1.event = \text{"high"} \text{ AND } (M_1.end_time - M_1.start_time) \geq 1.5 \text{ hours}) \text{ OR } \\
 &\quad \quad (M_1.event = \text{"low"} \text{ AND } (M_1.end_time - M_1.start_time) \geq 30 \text{ minutes}) \text{ OR } \\
 &\quad \quad (M_1.event = \text{"extremely low"} \text{ AND } (M_1.end_time - M_1.start_time) \geq 10 \text{ minutes})) \left(\right. \\
 &\quad \quad \sigma_{M_1.event \in \{\text{"extremely high"}, \text{"high"}, \text{"low"}, \text{"extremely low"}\}}(M_1) \\
 &\quad \left. \right) \left. \right)
 \end{aligned}$$

6.2 Aggregated High-Level Events

Aggregated High-Level Events offer a comprehensive view of glucose control by combining multiple high-level events into more complex patterns. These aggregated events are essential for identifying broader trends and making informed decisions about diabetes management.

To analyze these aggregated events, we use ISEQL operators such as **CARDINALITY CONSTRAINTS**. The **CARDINALITY CONSTRAINTS** focus on the frequency of events, specifying the minimum number of occurrences needed for an event to be considered significant within a given timeframe. This approach helps in recognizing patterns based on the frequency of certain types of events.

By applying this operator, we can thoroughly analyze the frequency of glucose events, providing a deeper and more actionable understanding of glucose control.

6.2.1 Too Frequent Glucose Anomalies

Too Frequent Glucose Anomalies refers to the number of times a patient’s glucose levels exceed or fall below normal thresholds within a specified period. For our analysis, this period is defined using the Medium-Level event *daily*, which spans from midnight to 23:59. Monitoring these anomalies is crucial for evaluating the stability of glucose control.

Tracking *Too Frequent Glucose Anomalies* helps identify patterns of instability. Frequent hyperglycemic or hypoglycemic events may indicate issues with treatment or other underlying factors. Such anomalies can significantly impact quality of life, potentially increasing the risk of accidents or accelerating diabetic complications.

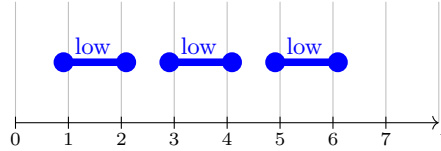


Figure 8: Too Frequent Glucose Anomalies with low glucose for one hour every two hours

In ISEQL, we analyze *Too Frequent Glucose Anomalies* by counting occurrences of hyperglycemic or hypoglycemic intervals within each daily period. The analysis employs:

- **CARDINALITY CONSTRAINTS:** This method ensures that event frequencies meet specified thresholds within daily intervals. Constraints are applied as follows:
 - At least 3 occurrences of "high" glucose events,
 - At least 3 occurrences of "low" glucose events,
 - At least 2 occurrences of "extremely high" glucose events,
 - At least 2 occurrences of "extremely low" glucose events.
- **DJ (During Join):** This operator ensures that all counts are correctly aggregated within daily periods.
- **DAILY:** This operator determines the daily intervals.
- **Union of Results:** This technique combines results from different glucose level checks using a union operation. Each check counts occurrences of a glucose level category within the daily interval, with the *DJ* operator ensuring accurate inclusion of events within each day. The results are then aggregated to provide a comprehensive count of anomalies per day.

By employing these methods, we can effectively monitor and analyze *Too Frequent Glucose Anomalies* on a daily basis, thereby enhancing the management of glucose control.

Let Union_Results be defined as:

$$\text{Union} \left(\begin{aligned} &\pi_{M.event, \text{Count}(M.event)} \text{ AS count_extremely_high} \left(\right. \\ &\quad \left. \sigma_{M.event = \text{"extremely high"}}(M) \text{ }^{(\geq 2)} \text{ DJ DAILY} \right), \\ &\pi_{M.event, \text{Count}(M.event)} \text{ AS count_high} \left(\right. \\ &\quad \left. \sigma_{M.event = \text{"high"}}(M) \text{ }^{(\geq 3)} \text{ DJ DAILY} \right), \\ &\pi_{M.event, \text{Count}(M.event)} \text{ AS count_low} \left(\right. \\ &\quad \left. \sigma_{M.event = \text{"low"}}(M) \text{ }^{(\geq 3)} \text{ DJ DAILY} \right), \\ &\pi_{M.event, \text{Count}(M.event)} \text{ AS count_extremely_low} \left(\right. \\ &\quad \left. \sigma_{M.event = \text{"extremely low"}}(M) \text{ }^{(\geq 2)} \text{ DJ DAILY} \right) \\ &\left. \right) \\ &\pi_{\text{start_time_Time}, \text{end_time_Time}, \text{Total_Count}} \text{ AS total_Count} \left(\right. \\ &\quad \left. \text{Count(Union_Results)} \right) \end{aligned} \right)$$

6.2.2 Too Frequent Time Swings

Too Frequent Time Swings assesses how often rapid fluctuations in glucose levels, referred to as *Time Swings*, occur within a specified period. A high frequency of these swings may indicate issues with glucose stability, suggesting that the patient's glucose management might need closer monitoring. Using ISEQL, we can identify and analyze these swings to improve glycemic control and reduce the risk of adverse events.

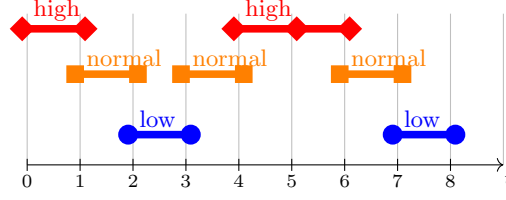


Figure 9: Rapid Time Swings between high and low glucose events within a few hours

To identify *Time Swings*, we use the *BEFORE* operator, which specifies the allowable time gap between successive swings. We then apply *CARDINALITY CONSTRAINTS* to determine how frequently these swings occur. Specifically, the constraint is set to identify instances where at least two Time Swings occur within a specified timeframe.

The analysis focuses on identifying frequent swings from high to low glucose levels or vice versa within the daily interval. The *DAILY* operator defines the daily period for the analysis, while the *DJ* (*During Join*) operator ensures that swings are accurately aggregated within each day. The *CARDINALITY CONSTRAINTS* are set to require at least two Time Swings within the daily period for swings to be considered frequent.

In summary, the query detects daily Time Swings characterized by rapid glucose changes, using the *BEFORE* with δ of the hours operator to define the time gap and *CARDINALITY CONSTRAINTS* to set the frequency threshold. The results provide insights into the occurrence of these swings and help in evaluating glucose stability and management effectiveness.

$$\begin{aligned}
& \pi_{Patient, Count(*)} \text{ AS count_time_swings, } M_1.event, M_2.event, (M_2.start_time - M_1.end_time) \text{ AS Duration} \left(\right. \\
& \quad \sigma_{M_1.Patient = M_2.Patient} \left(\right. \\
& \quad \quad \left(\sigma_{M_1.event = \text{"high"}}(M_1) \text{ Bef}(\delta : 2) \sigma_{M_2.event = \text{"low"}}(M_2) \right. \\
& \quad \quad \text{UNION} \\
& \quad \quad \left. \left. \left(\sigma_{M_1.event = \text{"low"}}(M_1) \text{ Bef}(\delta : 2) \sigma_{M_2.event = \text{"high"}}(M_2) \right) \right)^{(\geq 2)} \text{ DJ DAILY} \right)
\end{aligned}$$

6.2.3 Time Swing With Too Long Glucose Anomalies

The concept of *Time Swing With Too Long Glucose Anomalies* refers to rapid changes in glucose levels that occur within a short period and overlap with *Too Long Glucose Anomalies* of glycemic events. Using ISEQL, we can identify these rapid oscillations and overlaps between intervals of hyperglycemia and hypoglycemia with abnormal durations.

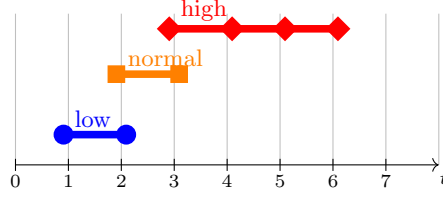


Figure 10: Time Swing from low to high with Too Long Glucose Anomalies on high

After identifying the *Time Swings* and *Too Long Glucose Anomalies*, we verify if the patients for these events are the same. Specifically, we check if the patient for the Time Swing events (TS1 and TS2) matches those for the Too Long Glucose Anomalies (TL1 and TL2). We then examine the following two cases:

- **Case 1:** TL1 occurs before TS1. This means that a Too Long Glucose Anomaly (TL1) happens before the Time Swing (TS1).
- **Case 2:** TS2 occurs before TL2. This indicates that a Time Swing (TS2) happens before a Too Long Glucose Anomaly (TL2).

By analyzing these cases, we can assess the relationship between rapid glucose changes and prolonged anomalies, providing insights into potential patterns or issues in glucose management.

$$\pi_{TS.Patient, TS.M2_event, TS.Duration, AD.Anomalous_Duration, AD.event} \left(\right. \\ \sigma_{TL1.Patient=TS1.Patient \text{ AND } TS2.Patient=TL2.Patient} \left(\right. \\ \left. TL1 \text{ Bef}(\delta : 0) TS1 \text{ UNION } TS2 \text{ Bef}(\delta : 0) TL2 \right) \left. \right)$$

7 System Architecture and Diagrams

The system architecture is designed to be efficient, scalable, and adaptable to changing requirements. It integrates a range of technologies tailored to specific tasks: Laravel for comprehensive web application development, the Flask micro-framework for lightweight web operations, and MySQL for structured and efficient data management. This section provides a detailed overview of the system architecture, emphasizing how these technologies interact and support the overall functionality.

In this section, we will describe three key types of UML diagrams that illustrate different aspects of our system's design:

- **Entity-Relationship (ER) Diagram:** Shows the data structure, including entities, attributes, and relationships. It illustrates how Laravel's ORM and MySQL manage and interact with data.
- **Sequence Diagram:** Depicts the sequence of interactions between various components for different use cases, including how Flask processes requests and how Laravel handles user interactions.
- **Component Diagram:** Highlights the high-level structure of the system, including the main components such as Laravel, Flask, and MySQL, and their interactions within the system.

7.1 Entity-Relationship Diagram

The ER diagram illustrates the data structure of the system, showing the entities involved and their relationships. MySQL is used to manage the relational database, ensuring data integrity and efficient query processing.

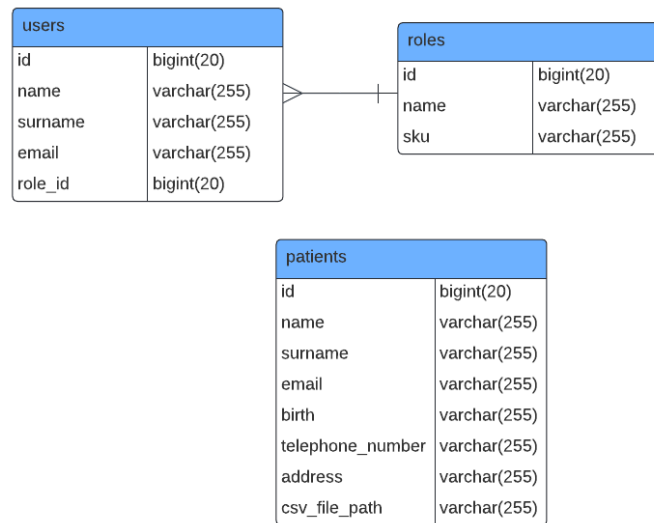


Figure 11: Entity-Relationship Diagram

7.1.1 Entity Descriptions

- **Users:**
 - Manages the information of operators/admins.
 - Stores essential details such as username, password, email, and other personal information.

- Each user is associated with a specific role which determines their permissions and access levels within the system.
- **Roles:**
 - Defines the different roles available within the system, such as admin, operator, etc.
 - Each role has specific permissions that dictate what actions a user assigned to that role can perform.
- **Patients:**
 - Contains all the information related to patients.
 - Includes personal details such as name, date of birth, address, phone number, and the file path of a CSV containing the patient's glucose level data.
 - The CSV file is an important component as it provides critical health information for each patient.

7.1.2 Entity-Relationship Description

There is a one-to-many relationship between **Roles** and **Users**. This means that each user is assigned exactly one role, but each role can be assigned to multiple users. This relationship is crucial for managing user permissions and ensuring that each user has the appropriate access level within the system.

7.2 Sequence Diagrams

Sequence diagrams describe the flow of operations within the system for various use cases. They provide a detailed view of how components interact over time. Flask handles specific web requests and responses, ensuring seamless communication between the client and the server.

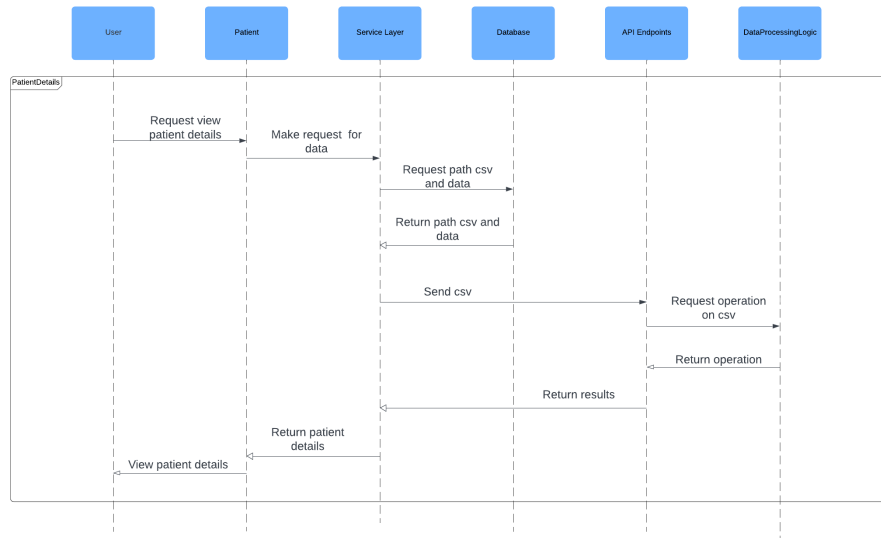


Figure 12: Sequence Diagram for New Patient

The user requests to view the details of a patient. This request is forwarded to the service layer, which is responsible for handling the application's business logic. The service layer requests the patient data and the CSV file path from the database. Upon receiving this information, the service layer uses Flask to send the CSV file to the API Endpoints.

The API Endpoints, in turn, request data processing operations on the CSV file from the Data Processing Logic. Once the processing is complete, the results are sent back to the API Endpoints, which then forward them to the service layer. The service layer compiles all the necessary patient data and forwards it to the patient view, which is displayed to the user.

7.3 Component Diagram

The component diagram provides a high-level view of the system's structure, highlighting the various components and their interactions.

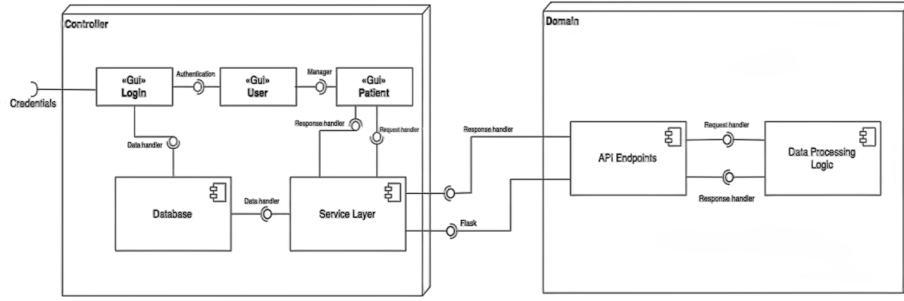


Figure 13: System Component Diagram

The component diagram highlights the following main areas:

- **Controller Block:**
 - **Login:** Manages user authentication via the **Database**, verifying credentials and ensuring correct access to the web application.
 - **Gui User:** User interface for managing user accounts.
 - **Gui Patient:** User interface for viewing and managing patient data. Users request and view patient information through this interface.
 - **Service Layer:** Manages business logic and coordinates communications between the **Database** and the **API Endpoints**. Interacts with the **Domain** to request data analysis.
- **Domain Block:**
 - **API Endpoints:** Handles data analysis requests received from the **Service Layer**, forwards them to the **Data Processing Logic**, and returns the results to the **Service Layer**.
 - **Data Processing Logic:** Performs data analysis operations received from the **API Endpoints**, using specific classes and methods (e.g., `IntervalActionDetector`, `ISEQL`) and returns the results to the **API Endpoints**.

7.4 Flask Application

The **Flask** micro-framework is utilized for handling lightweight web tasks within the system. Flask is chosen for its simplicity and flexibility, allowing quick development and easy integration with other components. It manages specific web requests and responses, particularly for real-time data processing and API endpoints. Flask's minimalistic design makes it ideal for microservices, which handle distinct functionalities such as data extraction, interval action detection, and interaction with the MySQL database. Flask's built-in development server and debugger also facilitate a smooth development process.

7.5 Laravel Framework

The **Laravel** framework is used as the primary technology for building and maintaining the web application. Laravel offers a comprehensive suite of tools and libraries that streamline development, including an expressive ORM (Eloquent), powerful templating engine (Blade), and robust routing capabilities. Laravel’s modular packaging system and extensive ecosystem allow easy integration of additional functionalities. It also provides built-in features for authentication, authorization, and session management, ensuring a secure and maintainable codebase. The framework’s emphasis on elegant syntax and developer productivity makes it an excellent choice for large-scale web applications.

7.6 Database Management

The website’s data is managed using **MySQL**, a robust relational database management system. We utilize **PhpMyAdmin**, a web-based interface for MySQL, to administer the database. PhpMyAdmin simplifies tasks such as creating databases, managing tables, and executing SQL queries directly through a user-friendly interface.

The database stores various types of information critical to the website’s functionality, including user credentials, patient data, and system configurations. Laravel’s ORM (Object-Relational Mapping) capabilities streamline database interactions, allowing developers to define database schema and perform database operations using PHP syntax.

PhpMyAdmin also facilitates database backups, ensuring data integrity and disaster recovery readiness. This comprehensive database management setup ensures reliable storage and efficient retrieval of glucose data for analysis and visualization within the website.

8 Implementation

In this section, we explore the implementation of glucose data analysis using **Python** and the pandas library. **Pandas** is a powerful library for data manipulation and analysis, offering high-level data structures and a wide range of tools for dataset management. We use a CSV file containing glucose data from the past 90 days, specifically from February 13, 2024, to May 12, 2024, including essential time-related information for monitoring and analyzing blood glucose level variations.

8.1 Project Structure

The project is organized into several files, each with a specific role in the glucose data analysis pipeline:

- **interval_action_detector.py:**
 - Contains the `IntervalActionDetector` class, responsible for analyzing glucose data. This class includes the method `offline_interval_action_detection`, which categorizes glucose levels and identifies intervals based on predefined thresholds.
- **interval.py:**
 - Defines the `Interval` class. This class represents a time interval with a start and end time, along with associated glucose level events and their duration.
- **iseql.py:**
 - Contains the `ISEQL` class. This class is used for further analysis of the intervals, such as detecting Too Long Glucose Anomalies and Too Frequency, finding Time Swings, and other advanced analyses.
- **data_processing_logic.py:**
 - Implements a Flask web application to provide an API endpoint for processing glucose data. This file facilitates the interaction with the analysis logic and can be integrated with Laravel or any other front-end framework to create a comprehensive web application.
- **analysis.py:**
 - This file contains code for evaluating the performance and scalability of the glucose data analysis pipeline. It includes tools to test the efficiency of various operations, measure execution times, and assess how the system handles increasingly larger datasets.

8.2 Data Extraction

To analyze glucose data, we extract relevant information from daily logs. Using pandas, we load data from a CSV file that includes significant columns such as event type, subtype, date, time of each measurement, and glucose value in mg/dL. This allows us to focus on pertinent columns for analysis.

8.3 IntervalActionDetector Class

In our glucose data analysis, we have implemented the class `IntervalActionDetector` with a function called `offline_interval_action_detection` to create time intervals associated with different glucose levels in the dataset. This function is designed to automatically identify and label intervals based on detected glucose levels.

The function takes input from a DataFrame `glucose_data` containing glucose level data, along with various predefined thresholds to categorize glucose

levels as high, low, or extremely low. By iterating through the first 10,000 records of the DataFrame, the function analyzes each glucose measurement and determines its glucose level category.

Key threshold parameters include:

- `extreme_high_threshold`: Threshold for extremely high glucose level.
- `high_threshold`: Threshold for high glucose level.
- `low_threshold`: Threshold for low glucose level.
- `extreme_low_threshold`: Threshold for extremely low glucose level.

During iteration, the function evaluates each glucose measurement against the defined thresholds and assigns a corresponding event label such as 'extremely high', 'high', 'extremely low', 'low', or 'normal'. Time intervals are then created based on significant changes in detected events.

This methodology segments and clearly visualizes periods of high, low, or extremely low glucose levels over time, enabling in-depth analysis of glucose level variations and related intervention actions.

8.4 Interval Class

The `Interval` class represents a time interval associated with a specific glucose event. Each instance of `Interval` includes the following properties:

- `symbol`: Identifying symbol for the event.
- `start_time` and `end_time`: start time and end time of the interval.
- `event`: Type of glucose event ('extremely_high', 'high', 'normal', 'low', 'extremely_low').
- `duration`: Interval duration, automatically calculated if not specified.

8.5 ISEQL Class

The `ISEQL` class manages a collection of time intervals (`intervals`). It provides useful methods to add, retrieve, and filter intervals, manage ISEQL relations, and find all high-level events.

Primary methods for interval management include:

- `add_interval(interval)`: Adds an interval to the collection.
- `get_intervals()`: Retrieves all intervals.
- `find_event_interval(event_name)`: Finds a specific interval based on the event type.
- `check_anomalies_event(interval)`: Checks if the interval is anomalous.

- `create_daily_intervals()`: Creates and returns a dictionary of daily intervals.

Subsequently, methods were implemented for ISEQL relations:

- `DURING(interval1, interval2)`: Checks if the first interval occurs completely during the second interval.
- `BEFORE(interval1, interval2, delta)`: Checks if the first interval ends before the start of the second interval, considering a time delta margin.
- `CARDINALITY_CONSTRAINTS(intervals, event, min_count)`: Checks if the number of intervals with a specific level within a given time frame meets the minimum count.

Finally, the following methods were implemented to find and manage high-level events:

- `find_time_swing(time_threshold=timedelta(hours=2))`: Detects Time Swing, which are significant changes in glucose levels occurring within a defined time interval.
- `find_too_frequent_glucose_anomalies(min_high=3, min_low=3, min_extremely_high=2, min_extremely_low=2)`: Identifies days with significant anomalies in the frequency of glucose level events.
- `find_too_frequent_time_swings(time_swing_threshold=timedelta(hours=2), min_ts=2)`: Finds significant glucose level changes that occur too frequently within a specified time period.
- `find_too_long_glucose_anomalies()`: Identifies time intervals where glucose levels show significant duration anomalies.
- `find_time_swing_with_too_long_glucose_anomalies()`: Finds Time Swings that are associated with Too Long Glucose Anomalies intervals.

8.6 Data Processing Logic

The data processing logic is implemented in the file `data_processing_logic.py`, which configures a Flask application to handle incoming CSV files and analyze glucose data. The main functionality is encapsulated in the `/process-csv` endpoint. Below is an overview of its operations:

- **File Handling:**
 - The endpoint receives a CSV file from the client. The file is saved in a predefined directory and a pandas DataFrame is created by reading this file.
 - If an error occurs during file handling or reading, an appropriate error message is returned to the client.

- **Data Preprocessing:**

- Relevant columns are selected from the `DataFrame` and initial rows are skipped to clean the data. This preprocessing ensures that only the necessary data is retained for analysis.

- **Interval Detection:**

- An instance of the `IntervalActionDetector` class is created to process the cleaned glucose data.
- The `offline_interval_action_detection` method is used to categorize glucose levels and identify periods of interest.

- **Interval Management:**

- The detected intervals are stored in instances of the `Interval` class, which are then added to an instance of `ISEQL`.
- The `ISEQL` class is used to manage these intervals and perform various analyses, including searching for anomalies and glucose fluctuations.

- **Results Analysis and Formatting:**

- Several analysis methods from the `ISEQL` class are called to detect anomalies and significant patterns in the glucose data.
- The results are formatted into a structured JSON response, including details on anomalous frequencies, durations, glucose fluctuations, and more.
- Utility functions such as `format_duration`, `format_datetime`, and `format_day` are used to ensure that the output is user-readable.

- **Additional Analysis:**

- The `analyze_glucose_data` function from the `analysis` module is called to perform additional analysis on the intervals. The results from this function are included in the final JSON response.

- **Endpoint Execution:**

- The Flask application is run with debugging enabled, allowing real-time feedback and easier troubleshooting.

This setup ensures that glucose data is processed efficiently and effectively, providing meaningful insights into glucose levels and trends. The Flask application acts as a bridge between raw data and analytical insights, facilitating the integration of data processing into larger systems or applications.

For a detailed explanation of the results and the analyses performed, please refer to Section 9.

8.7 Website Development

In the implementation phase, the website integrates **Laravel** and **Flask** to provide a comprehensive solution for web application development and data processing.

Laravel handles the core functionalities of the web application, including user authentication, data management, and interaction with the MySQL database. It utilizes its ORM (Eloquent) to manage database operations and streamline development.

For specialized data processing tasks, such as analyzing glucose data, we employ a Flask application. Defined in the file **data_processing_login.py**, Flask provides API endpoints for processing data, managing HTTP requests, and returning results in JSON format. Flask receives data from Laravel, performs the necessary computations, and returns the processed results.

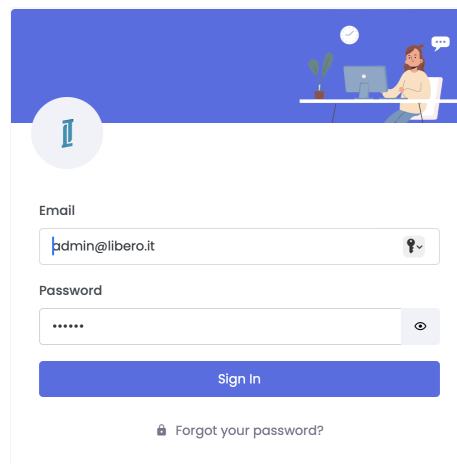
This integration of Laravel and Flask ensures effective data handling and processing, while PhpMyAdmin is used for managing backups and ensuring data integrity.

8.7.1 Login

The login functionality is a crucial part of the website's user management system. It ensures that only authorized users can access the system and its functionalities. The login page will use Laravel's built-in authentication mechanisms to handle user sign-in.

Users will be prompted to enter their credentials (username and password) on the login page. Upon successful authentication, they will be redirected to the dashboard.

The login page will include "Forgot Password" to enhance user security.



The image shows a login form with a blue header. On the left, there is a circular logo with a stylized 'I'. On the right, there is an illustration of a person sitting at a desk with a laptop. Below the header, the form has two input fields: 'Email' with the text 'admin@libero.it' and a key icon, and 'Password' with a masked password '.....' and an eye icon. A blue 'Sign In' button is below the password field. At the bottom, there is a link that says 'Forgot your password?' with a key icon.

Figure 14: Login

8.7.2 User Management

The website will feature user authentication and authorization functionalities using Laravel's built-in authentication system. Users will be categorized into two main roles:

- **Admin:** Admin users have full access to all functionalities of the website. They can manage operators (create, edit, delete), view patient data, and perform administrative tasks. By default, there is a principal admin who has the authority to modify and delete other admins and change their roles (Admin or Operator).
- **Operator:** Operators have restricted access compared to admins. They can view patient data and perform data analysis tasks but cannot manage other users.

8.7.3 Dashboard

The dashboard serves as the main interface upon logging into the website. It provides an overview of key metrics such as statistics on the number of operators, patients, and CSV files uploaded, as well as the latest operators added. The dashboard is customizable based on user preferences and provides interactive visualizations for deeper insights into patient data.

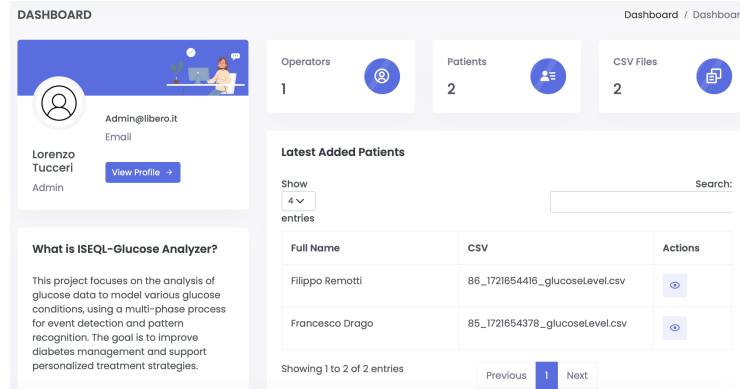


Figure 15: Dashboard

8.7.4 Profile and Operator Management

Users, both admins and operators, can manage their profiles through dedicated settings. They can update personal information, change passwords, and adjust notification preferences. Admin users also have access to a separate section for managing operators, where they can add new operators, modify existing profiles, and delete accounts if necessary.

The image shows two form sections. The first section, titled 'Update Profile', contains three input fields: 'Email' with the value 'admin@libero.it', 'First Name' with the value 'Lorenzo', and 'Last Name' with the value 'Tucceri'. Below these fields is a blue 'Update' button. The second section, titled 'Update Password', contains three input fields: 'Current Password' with the placeholder 'Enter current password', 'New Password' with the placeholder 'Enter new password', and 'Confirm Password' with the placeholder 'Confirm password'. Below these fields is another blue 'Update' button.

Figure 16: Personal profile

8.7.5 Patient Management

The website includes functionalities for managing patient records and data. Admins and operators can add new patients to the system, update patient information, and view detailed glucose data for each patient. Additionally, for each patient, CSV files containing their glucose data can be uploaded for studies and analysis.

The image shows a form titled 'Add Patient'. It contains several input fields: '*First Name' with the value 'Miriana', '*Last Name' with the value 'Baldassarra', '*Email' with the value 'miri.baldo@libero.it', '*Phone Number' with the value '3393921565', '*Date of Birth' with the value '2008-07-13', '*Address' with the value 'Via Maschera 12', and '*CSV' with a file upload button labeled 'Scegli file' and a file named 'glucoseLevel.csv'. Below the form is a blue 'Save' button.

Figure 17: New patient

In this subsection, we have detailed patient information; all statistics and results from the uploaded CSV file are displayed. This includes metrics such as Time Swing, high glucose levels, low glucose levels, and other relevant data

points derived from the glucose records. Additionally, it will be possible to download a PDF containing all the data related to the patient.

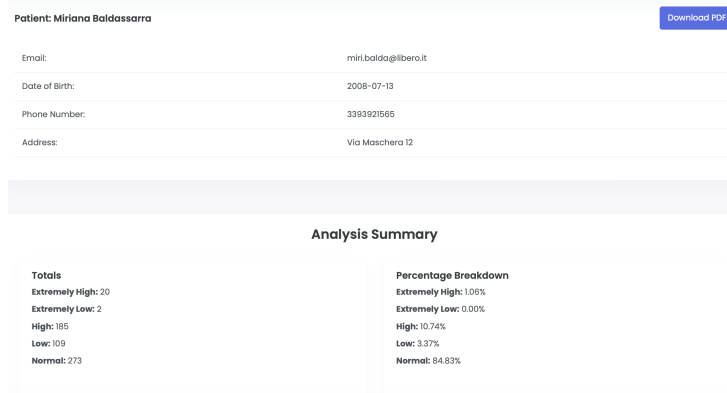


Figure 18: Patient Details

8.7.6 Website Limitations

In this version, the website enforces a restriction of one CSV file per patient, simplifying data management but potentially limiting in-depth analysis. Currently, the platform lacks graphical representations to visualize data trends and events. While detailed information is available in the patient detail section, the absence of charts may hinder the clarity and interpretability of glucose data and associated metrics. Additionally, the application only accepts CSV files exported from Dexcom.

9 Result and Analysis

In this section, we present the results obtained from analyzing glucose data using the functions and classes implemented in the file `analysis.py`. We evaluated the glycemic data by identifying extremely high, high, low, and extremely low glucose levels based on predefined thresholds and segmented these data into time intervals for detailed analysis.

9.1 General Statistics

The general statistics of the analyzed glucose data are as follows:

- **Total Counts:**
 - Extremely High: 20
 - High: 185
 - Low: 109

- Extremely Low: 2
- Normal: 273

- **Total Durations:**

- Extremely High: 08:20
- High: 84:30
- Low: 26:33
- Extremely Low: 00:00
- Normal: 667:44, representing 84.83% of the total time.

These statistics provide an overview of the distribution and duration of different glucose level events within the dataset.

9.2 Simple High-Level Event

The `find_time_swing` function identified significant changes in glucose levels, characterized by rapid fluctuations between high and low glucose levels. The results are as follows:

- A total of 39 significant Time Swings were detected.

This indicates that significant Time Swings are frequent and often lead to periods of prolonged instability.

9.3 Aggregated High-Level Event

In this subsection, we analyze various aspects related to aggregation high-level anomalous events, with a focus on Too Frequent Glucose Anomalies, Too Frequent Time Swings, Too Long Glucose Anomalies, and Time Swings With Too Long Glucose Anomalies.

9.3.1 Too Frequent Glucose Anomalies

The `find_too_frequent_glucose_anomalies` function revealed the day with the highest number of anomalies:

- On February 23, 2024, 3 high glucose events, 13 low glucose events, 0 extremely high glucose events, and 1 extremely low glucose event were recorded, totaling 17 anomalous events.

This highlights specific days when glucose levels were particularly unstable.

9.3.2 Too Frequent Time Swings

The analysis of Time Swings revealed:

- There are 7 days with too frequent time swings, amounting to 19 time swings.
- These account for 17.95% of the total Time Swings detected.

This indicates that some Time Swings occur with such frequency that they may warrant closer monitoring.

9.3.3 Too Long Glucose Anomalies

The `find_too_long_glucose_anomalies` function showed:

- For extremely high glucose intervals, 1.06% exhibited Too Long Glucose Anomalies.
- For high glucose intervals, 10.74% exhibited Too Long Glucose Anomalies.
- For low glucose intervals, 3.37% exhibited Too Long Glucose Anomalies.
- Extremely low glucose intervals showed 0.00% Too Long Glucose Anomalies.

The duration data highlights which glucose levels are more prone to anomalies and requires specific attention.

9.3.4 Time Swings With Too Long Glucose Anomalies

The analysis of Time Swings and their duration yielded:

- A total of 11 Time Swings were found to have Too Long Glucose Anomalies.
- This represents 28.21% of all Time Swings detected.

This suggests that a significant portion of Time Swings is associated with periods of Too Long Glucose Anomalies, emphasizing the need to address prolonged instability in glucose levels.

These observations collectively offer insights into the patterns and anomalies in glucose levels, enabling targeted interventions to manage and mitigate fluctuations effectively.

9.4 Evaluation Metrics

In this subsection, evaluation metrics such as efficiency, precision, recall, etc., will be added.

Here's the updated section with the provided data:

9.5 Scalability and Efficiency

The analysis of scalability and efficiency was conducted using datasets of varying sizes: 25%, 50%, 75%, and 100% of the total dataset. The following results summarize execution times and growth rates for different methods:

9.5.1 Execution Times

Table 10 presents the execution times for each method at different dataset sizes.

Method	25%	50%	75%	100%
offline_interval_action_detection	0.1405	0.1936	0.1950	0.1940
find_time_swings	0.00007	0.00012	0.00012	0.00012
find_too_frequent_glucose_anomalies	0.00014	0.00022	0.00021	0.00022
find_too_frequent_time_swings	0.00117	0.00282	0.00281	0.00281
find_too_long_glucose_anomalies	0.00002	0.00003	0.00003	0.00003
find_time_swing_with_too_long_glucose_anomalies	0.00012	0.00022	0.00022	0.00022

Table 10: Execution Times for Different Methods

9.5.2 Average Execution Times

Table 11 shows the average execution times for each method across all dataset sizes.

Method	Average Time (seconds)
offline_interval_action_detection	0.18076124
find_time_swings	0.00011016
find_too_frequent_glucose_anomalies	0.00019583
find_too_frequent_time_swings	0.00240318
find_too_long_glucose_anomalies	0.00002658
find_time_swing_with_too_long_glucose_anomalies	0.00019381

Table 11: Average Execution Times

9.5.3 Average Growth Rates

Table 12 provides the average growth rates of execution times as the dataset size increases.

Method	Average Growth Rate
offline_interval_action_detection	0.1267
find.time.swings	0.2431
find.too.frequent.glucose.anomalies	0.1968
find.too.frequent.time.swings	0.4690
find.too.long.glucose.anomalies	0.2123
find.time.swing.with.too.long.glucose.anomalies	0.2923

Table 12: Average Growth Rates

9.5.4 Graphs

Figure 19 illustrates the execution times for each method across different dataset sizes.

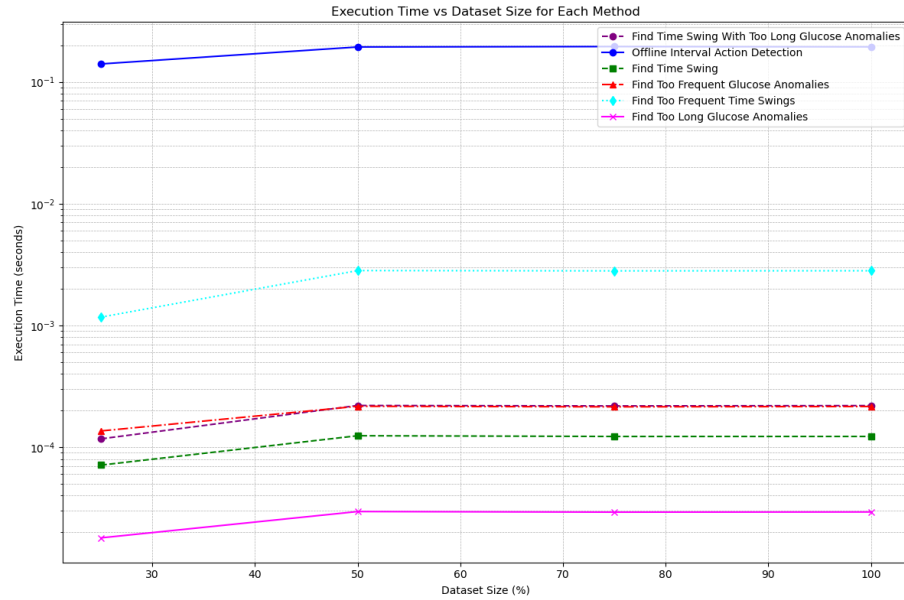


Figure 19: Execution Times for Different Methods

Figure 20 shows the average growth rates of execution times for each method as the dataset size increases.

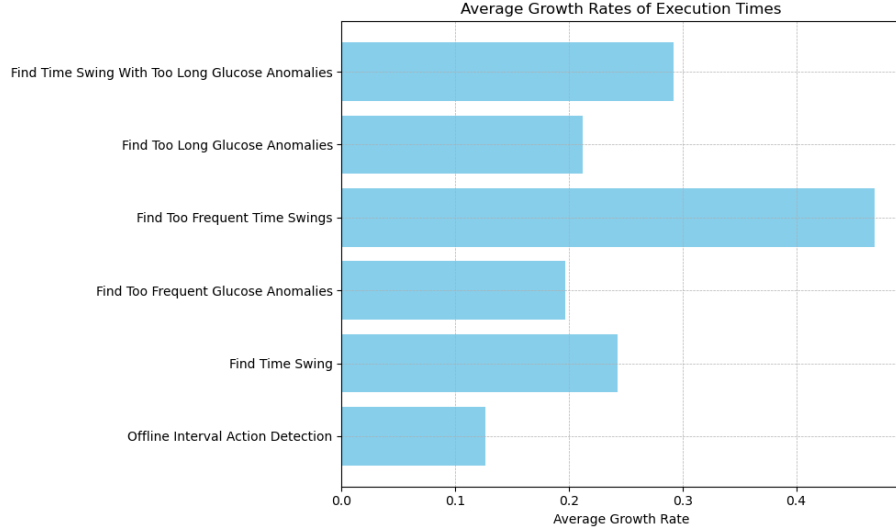


Figure 20: Average Growth Rates of Execution Times

9.5.5 Final Considerations

- **General Efficiency:** Most methods show very short execution times, typically in the millisecond range, with minimal impact from dataset size.
- **Scalability of the Main Method:** The `offline_interval_action_detection` method exhibits moderate scalability with an average execution time of 0.1808 seconds and a growth rate of 0.1267.
- **Other Methods:** Methods like `find_too_long_glucose_anomalies` and `find_too_frequent_glucose_anomalies` show negligible growth with average execution times of 0.00002658 and 0.00019583 seconds, respectively.
- **Growth Trends:** Growth rates are higher for methods like `find_too_frequent_time_swings` and `find_time_swing_with_too_long_glucose_anomalies`, but overall, execution times increase moderately with dataset size.

In summary, the system demonstrates good efficiency and acceptable scalability. The `offline_interval_action_detection` method requires further optimization attention, while other methods handle increasing data sizes well.

9.6 Conclusions and Future Work

The analysis of glucose data using implemented functions and classes has provided precise insights into glucose level changes over time, identification of significant anomalies in the duration and frequency of glycemic events, and recognition of important Time Swings. These results are critical for enhancing diabetes management, enabling timely interventions, and customizing treatment based on specific data points.

Future work will focus on improving the analysis of high-level events identified in the current implementation and researching and studying additional significant events. Additionally, the website will address and remove existing limitations, such as the restriction of one CSV file per patient and the lack of graphical representations, to provide a more comprehensive data management and analysis tool.

Moreover, there are plans to develop a mobile version of the website. This will offer several advantages, including increased accessibility for users on-the-go, real-time data monitoring and updates, and a more user-friendly interface for managing patient records and analyzing glucose data.

By enhancing event analysis, removing current limitations, and creating a mobile version, the website will significantly improve its utility and effectiveness in diabetes management and patient care.

References

1. **Flask**

Micro framework per Python. <https://flask.palletsprojects.com>

2. **Laravel**

Framework PHP per lo sviluppo web. <https://laravel.com>

3. **MySQL**

Database open-source. <https://www.mysql.com>

4. **ISEQL**

F. Persia and S. Helmer, "A Framework for High-Level Event Detection in a Social Network Context Via an Extension of ISEQL," 2018 IEEE 12th International Conference on Semantic Computing (ICSC), Laguna Hills, CA, USA, 2018, pp. 140-147, doi: 10.1109/ICSC.2018.00028.

5. **SOCO**

Project by Fabio Persia. Full project title: "Modeling and Detecting High-Level Events in Healthcare Applications Exploiting ISEQL+."