Ira A. Fulton Schools of Engineering
Arizona State University

CSE 572: Data Mining

Extracting Time Series Properties of Glucose Levels in Artificial Pancreas Project

Purpose

In this project, you will extract several performance metrics of an Artificial Pancreas system from sensor data.

Objectives

Learners will be able to:

- Extract feature data from a data set.
- Synchronize data from two sensors.
- Compute and report overall statistical measures from data.

Technology Requirements

- Python 3.11
- scikit-learn 1.3.2
- pandas 1.5.3
- numpy 1.26.3
- scipy 1.11.4

Project Description

In this project, we are considering the Artificial Pancreas medical control system, specifically the Medtronic 670G system. The Medtronic system consists of a continuous glucose monitor (CGM) and the Guardian Sensor (12), which is used to collect blood glucose measurements every 5 minutes. The sensor is single-use and can be used continuously for 7 days after which it has to be replaced. The replacement procedures include a recalibration process that requires the user to obtain blood glucose measurements using a Contour NextLink 2.4 glucosemeter ®.

Note that this process also requires manual intervention. The Guardian Link Transmitter® powers the CGM sensor and sends the data to the MiniMed 670G® insulin pump. The insulin pump utilizes the Smart Guard Technology that modulates the insulin delivery based on the CGM data. The SmartGuard Technology uses a Proportional, Integrative, and Derivative controller to derive small bursts of insulin also called Micro bolus to be delivered to the user. During meals, the user uses a BolusWizard to compute the amount of food bolus required to maintain blood glucose levels. The user manually estimates the amount of carbohydrate intake and enters it into the Bolus Wizard.

The Bolus Wizard is pre-configured with the correction factor, body weight, and average insulin sensitivity of the subject, and it calculates the bolus insulin to be delivered. The user can then program the MiniMed 670G infusion pump to deliver that amount. In addition to the bolus, the MiniMed 670G insulin pump can also provide a correction bolus. The correction bolus amount is provided only if the CGM reading is above a threshold (typically 120 mg/dL) and is a proportional amount with respect to the difference between the CGM reading and the threshold.

The SmartGuard technology has two methods of suspending insulin delivery: a) Suspend on low, where the insulin delivery is stopped when the CGM reading is less than a certain threshold, or b) suspend on predicted low, where the insulin delivery is stopped when the CGM reading is predicted to be less than a certain threshold. Apart from these options, insulin delivery can also be suspended manually by the user or can be suspended when the insulin reservoir is running low.

Accessing Ed Lessons

You will complete and submit your work through Ed Lessons. Follow the directions to correctly access the provided workspace:

1. Go to the Canvas Assignment, “**Submission: Extracting Time Series Properties of Glucose Levels in Artificial Pancreas Project**”

2. Click the “**Load Submission…in new window**” button.

3. Once in Ed Lesson, select the assignment titled "**Submission: Extracting Time Series Properties of Glucose Levels in Artificial Pancreas Project**".

4. In the code challenge, first review the directions and resources provided in the description.

5. When ready, start working in the Python file titled “**main.py**”

Directions

Dataset:

You will be given two datasets:

1. From the Continuous Glucose Sensor (CGMData.csv) and
2. from the insulin pump (InsulinData.csv)

The output of the CGM sensor consists of three columns:

1. **Data time stamp (Columns B and C combined)**,
2. **the 5-minute filtered CGM reading in mg/dL**, (Column AE) and
3. the ISIG value which is the raw sensor output every 5 mins.

The output of the pump has the following information:

1. **Data time stamp**,
2. Basal setting,
3. Micro bolus every 5 mins,
4. Meal intake amount in terms of grams of carbohydrate,
5. Meal bolus,
6. correction bolus,
7. correction factor,
8. CGM calibration or insulin reservoir-related alarms, and
9. **auto mode exit events and unique codes representing reasons (Column Q)**.

The bold items are the columns you will use in this project.

Metrics to be extracted:

1. Percentage time in hyperglycemia (CGM > 180 mg/dL),
2. percentage of time in hyperglycemia critical (CGM > 250 mg/dL),
3. percentage time in range (CGM >= 70 mg/dL and CGM <= 180 mg/dL),
4. percentage time in range secondary (CGM >= 70 mg/dL and CGM <= 150 mg/dL),
5. percentage time in hypoglycemia level 1 (CGM < 70 mg/dL), and
6. percentage time in hypoglycemia level 2 (CGM < 54 mg/dL).

Each of the metrics mentioned above is extracted in three different time intervals: daytime (6 am to midnight), overnight (midnight to 6 am), and whole day (12 am to 12 am).

The percentage is for the total number of CGM data that should be available each day. Assume that the total number of CGM data that should be available is 288. There will be days such that the number of data available is less than 288, but still, consider the percentage to be with respect to 288.

You have to extract these metrics for each day and then report the mean value of each metric over all days. Hence there are 18 metrics to be extracted.

The metrics will be computed for two cases:

- Case A: Manual mode
- Case B: Auto mode

Analysis Procedure:

The data is in reverse order of time. This means that the first row is the end of the data collection whereas the last row is the beginning of the data collection. The data starts with manual mode. Manual mode continues until you get a message “AUTO MODE ACTIVE PLGM OFF” in the column “Q” of the InsulinData.csv. From then onwards Auto mode starts. *You may get multiple* “AUTO MODE ACTIVE PLGM OFF” *in column “Q” but only use the earliest one to determine when you switch to auto mode. There is no switching back to manual mode,* so the first task is to determine the time stamp when Auto mode starts. **Remember that the time stamp of the CGM data is not the same as the timestamp of the insulin pump data because these are two different devices that operate asynchronously.**

Once you determine the start of Auto Mode from InsulinData.csv, you have to figure out the timestamp in CGMData.csv where Auto Mode starts. This can be done simply by searching for the time stamp nearest to (and later than) the Auto mode start time stamp obtained from InsulinData.csv.

For each user, CGM data is first parsed and divided into segments, where each segment corresponds to a day's worth of data. One day is considered to start at 12 am and end at 11:59 pm. If there is no CGM data loss, then there should be 288 samples in each segment. The segment as a whole is used to compute the metrics for the whole day time period. Each segment is then divided into two sub-segments: the daytime sub-segment and the overnight sub-segment. For each subsegment, the CGM series is investigated to count the number of samples that belong to the ranges specified in the metrics. To compute the percentage with respect to 24 hours, the total number of samples in the specified range is divided by 288.

Note that here you have to tackle the “missing data problem”, so a particular may not have all 288 data points. In the data files, those are represented as NaN. You need to devise a strategy to tackle the missing data problem. Popular strategies include deletion of the entire day of data, or interpolation.

Write a Python script that accepts two CSV files: CGMData.csv and InsulinData.csv and runs the analysis procedure and outputs the metrics discussed in the metrics section in another CSV file using the format described in Result.csv.

Submission Directions for Project Deliverables

This project will be auto-graded. You must complete and submit your work through Ed Lesson's code challenges to receive credit for the course:

1. To get started, use the "**main.py**' file provided in your workspace.
2. All necessary datasets are already loaded into the workspace.
3. Execute your code by running the "**python3 main.py**" command in the terminal to test your work.
4. When you are ready to submit your completed work, click on "**Test**" at the bottom right of the screen.
5. You will know you have completed the assignment when feedback appears for each test case with a score.
6. If needed: to resubmit the assignment in Ed Lesson
    a. Edit your work in the provided workspace
    b. Execute your code again by running the commands in the terminal
    c. Click "**Test**" at the bottom of the screen
7. Once you have finished working on the project, please submit it by clicking the **"Submit"** button at the top right corner of your submission space.

Your submission will be reviewed by the course team and then, after the due date has passed, your score will be populated from Ed Lesson into your Canvas grade.

Note:

1. Do not change the code file name; it must remain '**main.py**' for the auto-grader to recognize your submission.
2. When the auto-grader runs your Python file, it should generate a '**Result.csv**' file with the specified format. The '**Result.csv**' file should not include any headers and should only contain the metrics in a 2x18 matrix.

3. Avoid using absolute paths when accessing other files.
4. Before submitting to the grader, it is recommended to run your code file via the terminal to catch any potential runtime errors.

Evaluation

The auto-grader will run your code on a new CGMData.csv and InsulinData.csv from another subject and generate a Result.csv file. We will have a baseline Result.csv file generated from the code on our end. We will match the results of your code and our code. A mean square error limit of 10% will be set as the baseline threshold for getting a full grade.

- 140 points: Successful execution of your code.
- 60 points: The total number of metrics you get within the 10% error limit.