Compiled by:
Ananya Pal

Data Mining - Diabetes Data Analysis (Python)

PART 1

Dependencies: pandas, datetime

Input: CGMData.csv, InsulinData.csv

CGM data - CGM sensor readings every 5 minutes.

InsulinData - Small bursts of insulin called Micro bolus delivered to the user by insulin pump to maintain blood glucose levels.

Results:

- segregate into manual and auto mode

- OverNight, Day, Whole Day

(Percentage time in hyperglycemia : CGM > 180 mg/dL, Percentage time in hyperglycemia critical : >250, 70-180, 70-150, percentage time in hypoglycemia level 1 <70, percentage time in hypoglycemia level 2 <54)

(18 metrics extracted)

Procedure:

- **Data was in reverse order of time** i.e. the first row is the end of the data collection whereas the last row is the beginning of the data collection. So, I had to **traverse the data from the end**.
- Data **started with manual mode** and continued till an **alarm AUTO MODE** ACTIVE PLGM OFF in the **column** "Q" (Alarms) of the **InsulinData**.csv, and then onwards **Auto mode started**. So first we had to determine the timestamp when Auto mode started which was **difficult** since the **time stamp of the CGM data** was **not the same as the time stamp of the insulin pump data** as the **two different devices operated asynchronously**. (there was a time gap between the readings from cgm and insulin pump sensors)
- **Once the start date and time of AutoMode from InsulinData.csv was found, the consequent date and time needed to be found for the CGMData.csv.** Considering the condition for the **same date and the time Nearest to & Later than the Automode-start-time in the InsulinData.csv**, I found the **start time for Automode in the CGMData.**
- **Based on this Automode timestamp,** the **CGMData is divided into segments**, where **each segment** has **a day worth of data** (starting at 12 am, ending at 11:59 pm). If there is **no CGM data loss, then there should be 288 samples in each segment** (per date there should be 288 readings). 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: daytime sub-segment and overnight subsegment**. For each subsegment, had to **count the number of samples that belong to the ranges specified in the metrics** and computed the **per day percentage** (with respect to 24 hours). The **total number of samples in the specified range is divided by 288.**

- **I had to group the no. of samples in a particular range in a segment by Date and find Datewise percentage** (sum of datewise values divided by 288 samples per segment) **and then find the average of the Datewise percentages for the whole segment** (Datewise percent/length of the segment e.g. cgmAutoDatewise) **that a person was hyperglycemic or hypoglycemic and so on.**
- **Result: Basically, we found the Average Percentage time under the conditions that are hyperglycemia, hypergycemia critical, hypoglycemia level 1, hypoglycemia level 2, Per day (with respect to 24 hours).**

PART 2

Dependencies: pandas, datetime, numpy, pickle, sklearn, pywt, scipy

Meal data:

In the InsulinData, **start of a meal** consumption is **indicated by a non-NAN** in the Y column (**BWZ Carb Input** (grams)). Consider this time as t_m. **Meal data** comprises of a **2hr 30 min stretch of CGM data** that **starts from $\mathbf{t_m - 30min}$** till **$\mathbf{t_m + 2hrs}$.**

There are three conditions:

a) If there is **no meal between** time **$\mathbf{t_m}$ and $\mathbf{t_m + 2hrs}$**, use this stretch as **Meal** data
b) If there is **a meal** at some time t_p **between $\mathbf{t_m}$ & $\mathbf{t_m}$ +2 hrs** ($t_p > t_m$ & $t_p < t_m + 2$) then consider the **Meal at time $\mathbf{t_p}$** instead of t_m.
c) If there is **a meal** at time **$\mathbf{t_m + 2hrs}$,** then **consider the stretch** from:
$\mathbf{t_m + 1hr\ 30min}$ to **$\mathbf{t_m + 4hrs}$** as **Meal** data.

- **CsvExtraction.py:**
  Taking these **three conditions** into consideration was **confusing and time-consuming** as the Meal values needed to be taken from CGMData but the time conditions were given based on the Y column of corresponding InsulinData. However, I resolved it by **iterating through the InsulinData** and then **taking the difference between start time** (for each non-NAN value in Y column of InsulinData **(using numpy.NaN)**) **and end time,** and **checking for above conditions,** and adding it to a **list of meal times** and **no meal times**. Then **checking if the time corresponding to Glucose sensor values in CGMData lied in between the start and end times in the time list from InsulinData, I added the Glucose Sensor values to a Separate list of Meal data.** The **rest** of the data were added to **No meal** data, which did not satisfy above conditions. **Had to make a separate file for the meal & no-meal extraction as the code was long**.

No meal data:

No meal data comprises of **2 hrs of raw data** that **does not have meal intake**. **Start of no meal** is at time $\boldsymbol{t_m + 2hrs}$ where t_m is the start of some meal. Basically, all 2 hr stretches in a day that are **not in the meal data** and **do not fall within 2 hrs of start of a meal.** (there should not be a non-NAN value at $t_m + 2$ also)

- The Meal and No Meal data were individually exported into two csv files of size Nx25 each.
- **Train.py:**
  Taking the **Mealdata.csv** as **Input**, calculated the mean, kurtosis, skew and other features column wise. (e.g. df.mean(axis=1)) **Normalized the feature matrix** and **added a column for 'Class'** with all values as 1. Similarly, made the feature matrix for No Meal data and normalized it, and added 'Class' with column values as 0.
  Concatenated all the **Meal and No meal** data after this step and **trained SVM classifier (sklearn**). Calculated **Accuracy**, F1 score, **Precision and Recall** (sklearn.metrics) and performed **K-fold Cross validation.** Generated a **pickle** file and **saved the model**.
- **Test.py:**

Similarly, using the test file, the features were generated and feature matrix was normalized and the **Class of the test data (Meal=1, NoMeal=0) was predicted** by the pickle model already saved.

- **Result: Achieved an accuracy of 80% and successfully predicted class of test data.**

PART 3

Dependencies: pandas, datetime, numpy, sklearn, scipy

- From the **InsulinData Y column**, found the max and min values and **discretized** the **corresponding meal amounts** in **bins of size 20**. Considering each row in the meal data matrix generated in Part 2, according to their meal amount label, they were put in the respective bins. In total, $n = \frac{max - \min}{20}$ bins exist. These bins form the **Ground Truth**.
- Without using the above meal amount data (ground truth), used the **features in Part 2** to perform **K-means clustering** on the meal data into n=6 clusters and visualized the data using scatterplot in Matplotlib. Also, performed **DBSCAN** on the features to compare the performance between the two algorithms.

Result: The accuracy of clustering based on **SSE** and **entropy** were calculated for both the algorithms. The bins from step 1 served as Ground Truth for the SSE calculation.