# ISYE6501 Course Project

My personal interests lie in the fields of education, medicine and finance, and naturally I am excited to apply analytics solutions to these fields. For my project, I am choosing to analyze the case in which Medtronic partnered with IBM Watson to develop the "Sugar.IQ with Watson" diabetes monitoring smartphone application.

The IBM link to this case is here: https://www.ibm.com/case-studies/medtronic

Also the site below provides more information from Medtronic:
https://www.medtronicdiabetes.com/products/sugar.iq-diabetes-assistant

I am neither a doctor or a diabetes patient, but I am very interested in the application of machine learning and analytics to help people make better decisions. I will focus mostly on the possible modeling approaches this product could have used.

Using the websites above we have gathered a list of services that the application provides, which also potentially use modeling techniques discussed in this class:

1) Using a wearable monitor, the app continuously (at 5 minute intervals) monitors your blood glucose and makes diet/activity recommendations based on these readings. It offers notifications 15-60 minutes ahead of time if a user is predicted to be high or low in their blood sugar levels.
2) There is likely to be some type of general meal recommendation system for when you are preparing meals at home, so that you can effectively keep your blood sugar within a reasonable range.
3) Clients may use the app when at a restaurant; they can enter their prospective meal order into the app to see if their meal choice would keep their blood sugar levels within the reasonable range.

All of these features are then combined to create the solution, which is the smartphone application in question.

Part 1: Using a wearable monitor, the app continuously (at 5 minute intervals) monitors your blood glucose and makes diet/activity recommendations based on these readings. It offers notifications 15-60 minutes ahead of time if a user is predicted to be high or low in their blood glucose levels.

For someone without diabetes, a fasting blood sugar on awakening should be under 100 mg/dl. Before-meal normal sugars are 70–99 mg/dl. "Postprandial" sugars taken two hours after meals should be less than 140 mg/dl. These are the normal numbers for someone without diabetes. If you have diabetes, the American Diabetes Association (ADA) advises keeping your blood sugar levels before meals from 80–130 mg/dl and your levels 1–2 hours after meals under 180. Many people with diabetes and doctors shoot for levels closer to those of people without diabetes, because they are more protective against complications. Less extreme numbers require more careful diet and more frequent monitoring to prevent lows, but they are doable for many people. [2]

**Given the time-series blood glucose values for the specified user, use a CUSUM model (with both increase and decrease detection) to detect when blood glucose is above/below the normal values.**

For people with diabetes, blood glucose values above 180 mg/dl are considered high, and values less than 50 mg/dl are considered hypoglycemic. However values in these ranges are quite unhealthy, so we would hope that our CUSUM will alert us well before these values are reached. This will also help the user take preventative measures before it becomes problematic. The general form of the CUSUM models are as follows:

Increase detection: $S_t = max\{0, S_{t-1} + (x_t - C - \mu)\}\, Is\, S_t \geq T\,?$
Decrease detection: $S_t = max\{0, S_{t-1} + (\mu - x_t - C)\}\, Is\, S_t \geq T\,?$

Blood glucose values tend to fluctuate throughout the day. We could not find good estimates regarding how much variance we could expect – we imagine that it depends on the specific patient and their habits/routines throughout the day. However, we can easily calculate the standard deviation for the specific patient using the historical data collected by the attached monitor. Once we have that value (perhaps it can update daily, based on the past month or week of data), we can set the threshold T-value for the CUSUM. We can calculate the mean for the same interval as well. How high to set the threshold would likely depend on the specific patient, but as a first try we could try 1 or 2 times the standard deviation of the user calculated previously. This will ensure that the threshold is specific to each patient. Also we would try setting C equal to the standard deviation, since C is supposed to account for some of the observed variability.

When CUSUM detects an increase, we could recommend that the user take action to lower their blood glucose levels by drinking water, exercising or eating a protein-based snack. If CUSUM detects a decrease, we could recommend that the user consume carbohydrates such as fruit juice or ceasing physical activity. The recommendations could also take into account the user food preferences that the app keeps track of (we explain these more in Part 2).

As an alternative or supplement to the CUSUM model, we can use a different time-series model to get more concrete predictions for the next period's blood glucose values:

**Given the time-series blood glucose values for the specified user, use an exponential smoothing model to forecast the blood glucose values for the next time period.**

General exponential smoothing model: $S_t = \alpha x_t + (1-\alpha) S_{t-1},\ 0 < \alpha < 1$
with initial condition $S_1 = x_1$

The “alpha” value for exponential smoothing would depend on the user-specific blood glucose variance within the past week or month – the higher the variance, the closer alpha would be to 0.

To use this approach, we would need to first modify our input data. The app claims to provide predictions 15, 30 or 60 minutes in advance, so we would need to create three data sets based on our initial 5-minute interval blood glucose data. We do this carefully as follows:

Let N be the number of 5-minute interval data points that we keep a record of (say N=2040 – that would be roughly equivalent to one week). We construct the relevant data intervals as follows:

15 minute intervals: average data points 1-3, 4-6, 7-9,...,(N-2)-N together to obtain 680 new data points
30 minute intervals: average data points 1-6, 7-12, 14-19,...,(N-5)-N together to obtain 240 new data points

60 minute intervals: average data points 1-12, 13-24, 25-36,...,(N-11)-N together to obtain 120 new data points

Then we construct smoothed time-series with each of these data sets. This will allow us to create predictions for the next time period using each data set - 15, 30 and 60 minutes into the future, using the forecast $x_{t+1}=S_t$ - please note here that we would also include trend and seasonality factors if they were relevant features of the observed data. If $x_{t+1}$ was above or below a specified threshold value (for example, above "high = 180 mg/dl" and below "low = 70 mg/dl") we would send a notification to the user's phone informing them of such, as well as the time interval (15, 30 or 60 minutes) that the user will have to take precautionary measures in. Then we could make similar recommendations to the CUSUM situation, based on whether we predicted a high or low value.

Part 2: There is likely to be some type of general meal recommendation system for when users are preparing food at home, so that users can effectively keep their blood sugar within a reasonable range.

When the app starts up for the first time, it would not know which foods the user prefers. We could ask the user to complete a short survey of how they would rank 100 dishes in terms of how much they like or dislike the prospective dish. Once we have these initial rankings, we can add data to it as the user eats and drinks more things and inputs feedback into the app. This will give a clearer picture of the user's preferences over time and will allow the app to produce better recommendations. However this might be too bothersome to be realistic.

Another way to determine food preferences is to use a modeling approach. We could start with some baseline such as the 100 most-preferred meals of other users of this app, and then personalize according to the Multi-Arm Bandit (MAB) approach.

**Given the preferred meals of other users of this app, use a multi-arm bandit model to update the meal preferences of this specific user.**

In the MAB approach, the expected preference value of each of the 100 prospective meals are equal at initialization. Then as the user consumes foods among the initial options, we can adjust the expected preference value according to how many times a user chooses this meal, how much they liked the meal (0-5 scale), or a weighted average of both factors. Then after the first 100 or so meals eaten by the user, we could reduce the number of prospective meals by keeping the 50 meals with the highest updated expected values. We could repeat this process an additional step or two to narrow down to 20 or 30 meals – this would be specific enough to the user's preferences, while still providing the user plenty of variety.

**Given user food preferences, use an optimization model to recommend a meal while meeting nutritional constraints relevant to the predicted blood glucose state.**

This problem could be formulated in a similar way to the Week 7 homework. The main difference is that instead of minimizing costs, we want to maximize the meal preference score of the user such that the nutritional constraints are satisfied.

An important input into this model is the predicted state of the user in Part 1, as this will determine our optimization problem constraints. If a user is predicted to be in a high blood sugar state, we would set

the carbohydrate constraints to be quite low (say 0 <= carbs <= 10g), while our protein constraint would be set to at least 10-15g or so (15<= protein). If the user is predicted to be in a low blood-sugar state, then we would set the carbohydrate restraint to be much higher (30g<= carbs) while reducing the protein requirements (0<= protein < 10g). Finally if the user is not in either of these two "concerning" states, we could use normal nutritional constraints. Our linear program would select and then recommend to the user the meal that has the highest expected preference value that also satisfies the constraints.

Part 3: Clients may use the app when out to eat; they can enter their prospective meal order into the app to see if their meal choice would keep their blood sugar levels within the reasonable range.

**Given restaurant food options, user food preferences and predicted blood glucose levels determined in Part 1, use an optimization model to recommend a meal.**

The main difference here from Part 2 is that our possible food options are limited by what the restaurant has to offer. We would hope that the app would be able to reference some database where common restaurant food items and their nutritional breakdown are stored. It would then be simple to reference this database and see if the input meal (e.g. "chicken Alfredo" or "pad Thai") satisfies the constraints that are governed by the predicted blood glucose state. If it does not, we could respond to the user input by telling them that the meal is not recommended.

We can also use this system to recommend a meal at the restaurant without having the user input anything. If we use the above database filtered by items available at the restaurant in question, we could solve the linear program and recommend the most preferred item that satisfies the relevant constraints. One potential difficulty is that the user preferences saved in the app may not match up with any item that the restaurant offers – perhaps the user is being adventurous and trying some new ethnic cuisine. Then we would simply recommend any meal that satisfies the dietary constraints of the predicted blood glucose state.

References:

[1] https://www.diabetesselfmanagement.com/blog/what-is-a-normal-blood-sugar-level/

[2] https://www.healthline.com/health/hypoglycemia