Capstone 3 Project Report

Notebook

1. Problem Statement

This data science project aims to use data collected by an individual with type 1 diabetes, including blood glucose measured by a continuous glucose meter (CGM), manually recorded insulin injections and carbohydrate intake, and physical activity levels collected by a fitness tracker, to make predictions of how this person's blood glucose levels will develop in the short-term. The goal is to develop a predictive model that can anticipate blood sugar fluctuations by analyzing the interactions among these variables, providing individuals with type 1 diabetes a tool for better daily management of their condition.

The challenge is to forecast blood glucose levels for four hours into the future, with an error margin below 15%. Achieving this accuracy is required in order to be a useful guide for people with type 1 diabetes to incorporate these forecasts when deciding on the optimal dosage for their next insulin injection.

Type 1 diabetes is an autoimmune disease in which the body's immune system attacks and destroys insulin-producing cells in the pancreas. This leads to insufficient insulin production, causing high blood sugar levels. Treatment involves daily insulin injections or an insulin pump to regulate blood sugar. Patients must also monitor their blood sugar levels, follow a balanced diet, and engage in regular physical activity to manage the condition.

A **continuous glucose monitor** (CGM) is a small device that continuously tracks a person's blood glucose levels and provides real-time data via a sensor inserted under the skin. Blood glucose levels are influenced by carbohydrate intake, insulin administration, physical activity, and other factors such as stress.

Understanding the **timeline of insulin and carbohydrates in blood glucose management** is vital for diabetes. Carbohydrates lead to a swift blood sugar rise, faster than rapid-acting insulin's peak action, necessitating taking insulin some time before a meal. Rapid-acting insulin typically begins within 15 minutes, peaks after 1-2 hours, and returns to normal levels within approximately 4-6 hours. The glucose increase from carbohydrates usually ends within 2-3 hours after a meal.

The data for this particular individual show some corrective insulin injections (see EDA), presumably based on current blood glucose readings. These doses may be unnecessary and could be avoided if a model predicts that blood glucose will decrease in the near term without intervention.

2. Data

The data set was downloaded from Kaggle and consists of csv files with blood glucose measurements (in mmol/L) from a CGM, JSON files from an activity tracker, and one csv file in which the amount (in grams) and type (glycemic index categories) of carbohydrates, and amount (international units, IU) and type of insulin (rapid and basal) were recorded. The common time frame for which all data are available is January 9th 2022 to April 24th 2022. The CGM data included three columns: scan glucose (current value), historic glucose (averaged values over the last 15 minutes), and data from glucose measurement sticks (strips requiring a blood drop). The recorded insulin and glucose data consisted of seven columns, including basal and rapid insulin, carbohydrate grams and glycemic index, meal type and tags.

Kaggle data, Github data

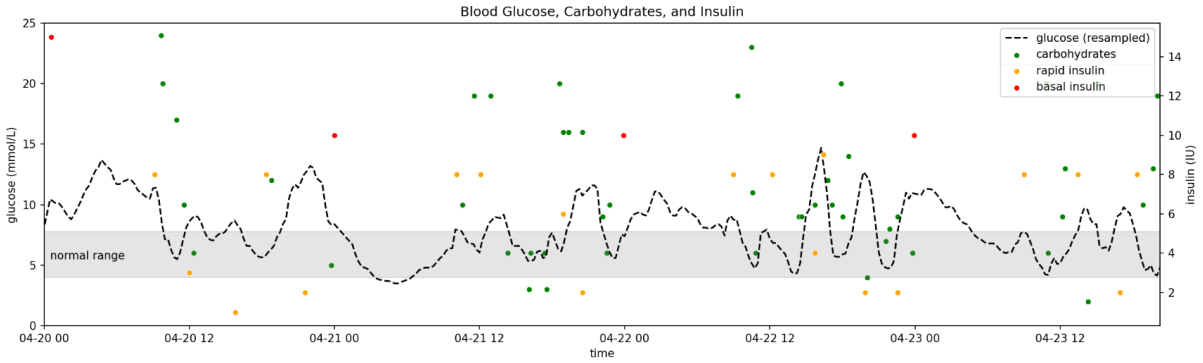
3. Data Cleaning, Wrangling, and Pre-processing

After combining glucose data from nine csv files, rows were removed if they fell outside of the common time frame, did not have any glucose data, or were exact duplicates. Also, the stick glucose column was dropped because it contained few values. Remaining duplicates based on the same time index were collated. Since a plot of the remaining columns showed that their values were well aligned, they were combined using the historic values when both existed. Next, the combined glucose column was resampled using 15 minute intervals, with linear interpolation to fill any gaps (the largest gap was almost three hours, and the mean gap time was about 10 minutes).

Of the columns in the custom log file for insulin and carbohydrates, meal type and tags were dropped because they had few values and unclear predictive utility. Glycemic index values were unified into four categories from low to very high. Small intervals between carbohydrate intake times were checked to exclude accidental double entries. Basal insulin, rapid insulin, and carbohydrate grams were resampled using 15 minute intervals, filling any missing values with zero

4. Exploratory Data Analysis (EDA)

Seasonal decomposition was performed on the combined glucose data, and plots showed an irregular trend line with one to three daily peaks (values roughly between 4 and 12 mmol/L), negligible seasonal component, and residuals of considerable size (up to 4 mmol/L). Stationarity tests were inconclusive, with Kwiatkowski–Phillips–Schmidt–Shin (KPSS) indicating stationarity, while augmented Dickey-Fuller (ADF) suggested non-stationarity. After differencing once, both KPSS and ADF point to stationarity and therefore a differencing order of one for an ARIMA model was chosen. Plots for ACF and PACF both trailed off, which did not allow to determine lag order for the autoregressive component and the moving average through this method. An example of four days of glucose, insulin, and carbohydrate data toward the end of the available time frame is shown below.



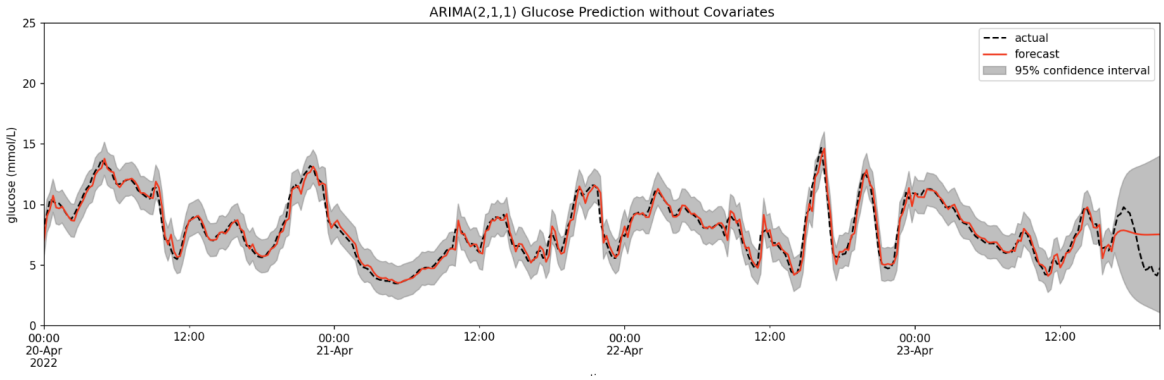
5. Training and Model Selection

Parameters for the autoregressive component (p) and the moving average (q) of an ARIMA model for glucose data were determined using a plot of ordered MSE values of various ARIMA models of the differenced series (p from 0 to 12 and q from 0 to 6, with differencing order d held constant at 0). Two models with small orders of p and q with low MSE were (2, 0, 1) and (3, 0, 0). Based on this, a (2, 1, 1) ARIMA model was fitted on the undifferenced glucose values for the entire time frame except for the last four hours.

ARIMA models future values of a time series as a **linear combination of past values** (this is the AR part) **and past errors** (this is the MA part). The model works on **stationary** data (mean and std remain constant over time) **,** so the original series may have to be **differenced** one or more times (this is the I part). The number of past values (so called **lags**) p, the number of past error terms q that factor into the equation, and number of times the series is differenced d, is expressed as the **model order (p, d, q)**.

In a **SARIMAX** model, a separate order for the **seasonal component** (the S part) is specified as **(P, D, Q, s),** where **s is the number of periods in a season** (for example, 4 for quarterly data or 12 for monthly data). If a future value depends not only on its own past values (**endogenous variable**), but also on past values of additional time series (**exogenous variables**, the X part), these form another set of terms representing a **'linear combination of previous values of the exogenous variables'** that **are added on** to the equation (including error terms).

The one-step-ahead predictions show a close match to the actual values. While MAE for the one-step-ahead forecast was 0.48 mmol/L, the 95% confidence interval is up to 4 mmol/L wide. The 95% confidence interval for the forecast for the next four hours shows a range from 4 to 16 mmol/L after 4 hours, with a mean forecast of about10 mmol/L. The model diagnostics plots indicate a residual distribution with higher peak and heavier tails, but no failure of the model to capture patterns in the data. The four hour prediction had a mean average percentage error (MAPE) of 34%.



The same model parameters were used to fit a SARIMAX model, with the addition of three covariates: carbohydrates, basal insulin, and rapid insulin. The prediction for SARIMAX, which included future covariates, did not differ much from the ARIMA model in either plot shape or MAPE.

Next, a Darts TCN model was fitted on the same glucose time series, again leaving four hours of data at the end for the prediction interval (MSE is the default metric when training the model). The best MAPE for the forecast of this model was 31%. When using carbohydrate data as a past covariate the best MAPE increases to 34%, but after additionally including both types of insulin in the covariates, the model again achieves a best MAPE of 31%. However, the forecast plots show that the TCN model with one covariate aligns fairly well with the actual data for about half of the forecast timeframe.

The **Temporal Convolutional Network (TCN)** approach borrows from CNN, but in the simplest case, applies the **convolution** in one dimension only instead (or, in case of multiple input channels, at least it still **moves along a single axis**; note that channels are not exactly the same as covariates - no information on covariate convolution..). Another difference is that in TCN the value of an **output element is derived only from input elements that precede it** timewise. If there are n network layers and the filter or kernel size is k, an element in the last layer depends on r = 1+n (k-1) number of elements from the bottom layer. This r is called the **receptive field**. **Dilation** is a concept that ensures that a greater part of the input can be included in the receptive field without requiring a large number of layers. It works by **skipping over input cells,** e.g., every other one. In order to achieve a certain dilation in the bottom layer, the dilation factor needs to increase exponentially with each step up to the top layer, expressed as **dilation base b**. Thus, the number of layers depends on the dilation and **n is now a log function of the receptive field (or input)** instead of linear. In addition the kernel size should be at least as large as the dilation base to avoid large holes between elements

of the bottom layers that are included in the receptive field.

Medium TCN article

A **Convolutional Neural Network (CNN),** as it is used in image classification, 1. reduces the number of input nodes, 2. tolerates small shifts in the image, and 3. takes advantage of the correlation between adjacent pixels in complex images, compared to a regular Deep Neural Network. It uses, e.g., a 3x3 pixel filter (aka kernel, which represents a certain pattern) to form the dot product with each successive 3x3 part of the image (moving by one pixel or more each step) - this is called **convolving** the input with the filter (the output contains information not just about one pixel but also its neighbors, so it **incorporates correlation**). A bias term is added to the output of the filter and the resulting value is added to a feature map, which has the same size (or smaller if moving by more than one pixel) as the image but contains the degree by which a pixel conforms to a certain pattern. Next, a ReLU activation function and max **pooling** is applied to the feature map, assigning the highest value from a sub-region (e.g., 2x2 area) of the feature map to the output. This reduces the size of the map which is then used as input to a regular neural network (so pooling **achieves the reduction of input nodes**). Max pooling selects spots where the filter did best matching the input image. **Robustness against shifts** in the image is the result of several aspects: a **moving filter** detects features across the entire image, **pooling** means averaging over an area, so it doesn't matter where in that area the feature is, and **fully connected layers** mean input from the entire image is considered for the next layer.

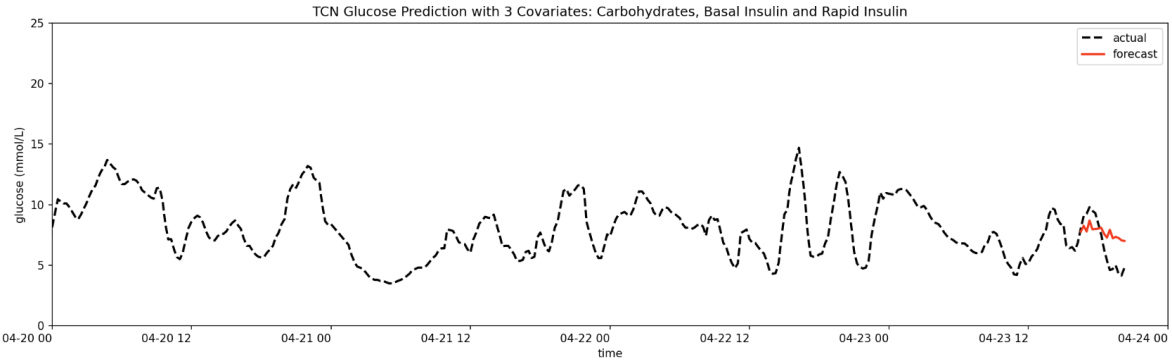
StatQuest CNN video

Model	**MAPE [%]**	**Parameters**	**Comment**
ARIMA	34	p= 2 (number of autoregressive lags), d=1 (order of differencing), q= 1(number of moving average lags)	No covariates
SARIMAX, 3 covariates	34	See ARIMA	Past and future covariates
Darts ARIMA	34	See ARIMA	No covariates, uses statsmodels
Darts TCN, no covariates	1. 36 2. 31 3. 32	1. input_chunk_length=288 output_chunk_length=16, n_epochs=10, weight_norm=True (defaults: kernel_size=3, num_filters=3, dilation_base=2, dropout=0.2) 2. change to input_chunk_length=192, n_epochs=5, weight_norm= False 3. change to kernel_size=5, num_filters=6	No covariates, uses torch
Darts TCN, 1 covariate	1. 37 2. 34 3. 35	See Darts TCN without covariates	Past covariates only, uses torch
Darts TCN, 3 covariates	1. 35 2. 35 3. 31	See Darts TCN without covariates	Past covariates only, uses torch

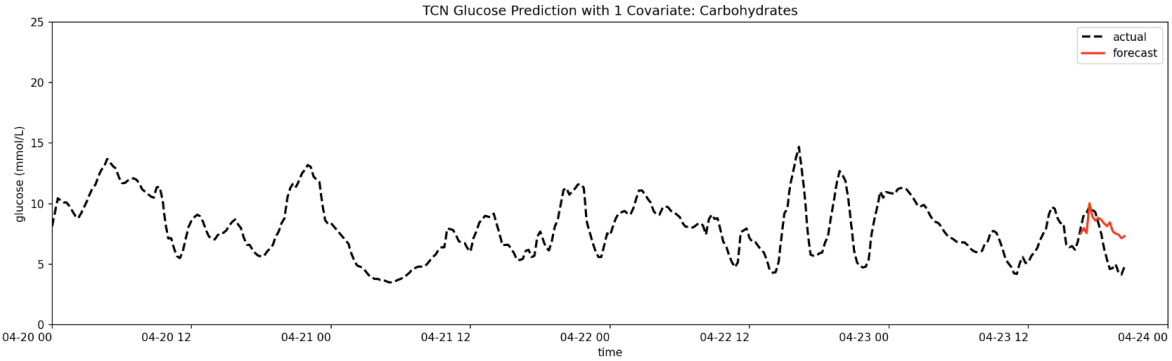
Model metrics (csv)

6. Modeling: Glucose Prediction

The model with the best MAPE for a four hour glucose forecast was Darts TCN with three covariates. Despite the lower MAPE compared to SARIMAX, it is not clear that the shape of the prediction aligns better with the actual data.



Even though the TCN model with one covariate (carbohydrates only) has a higher MAPE, it appears to have a better match between forecast and actual glucose for the first part of the prediction interval.



7. Recommendations

For a 4-hour blood glucose forecast, all (S)ARIMA(X) and TCN models have a MAPE of over 30%, regardless of whether covariates (carbohydrates, basal insulin, rapid insulin) are included or not. This error margin is too large to base any decisions on it, such as insulin amount for the next injection. However, the TCN model with one covariate could be used to estimate how blood glucose will develop in the next two hours before making decisions to correct current glucose levels with injections of rapid insulin. This would allow to avoid unnecessary injections (in case glucose is projected to fall) or to pick a higher dose (if glucose is predicted to rise). In addition, a newer deep learning architecture for time series forecasting, Temporal Fusion Transformer (TFT) should be explored. TFT is implemented in Darts and promises better performance on the type of data set used here.

8. Limitations and Future Work

The plots of all (S)ARIMA(X) and some TCN models show that the forecast diverges from the actual values very quickly. In addition, the (S)ARIMA(X) models' confidence intervals (statsmodels package) show that the range of possible blood glucose values may potentially be outside of the MAPE. In comparison, the TCN model with one or three covariates is closer to the actual values for almost half of the forecast timeframe. However, Darts does not automatically add confidence intervals[1], and SARIMAX and TCN models differ in their support for past and future covariates (the latter does not consider future covariates), so a direct comparison is

[1] See Darts TCN likelihood parameter.

problematic. To better judge whether the TCN forecasts' small fluctuations and seemingly better match is a sign of a better prediction or potentially overfitting to noise, several predictions throughout the entire time frame should be looked at.

The MAPE of the TNC model with three covariates could likely be reduced with a shorter forecast timeframe of 2 hours. This is still a useful prediction length for a diabetic who is considering a corrective insulin injection. In addition, increasing the time interval from 15 to 30 minutes would reduce the number of steps the model needs to forecast for the same forecast timeframe and may lead to better results. Also, glycemic index and activity data should be incorporated into the covariate models and hyperparameter tuning tools should be explored (gridsearch for TCN is only available for time series without covariates). A different model, Darts TFT, is expected to perform well in multi-step forecasts and supports future and past covariates.

A **Temporal Fusion Transformer (TFT)** is a form of deep neural network that borrows part of its approach from Transformers (an algorithm known for its use in large language models), such as sequence-to-sequence layers to learn short-term relationships and self-attention to learn long-term dependencies.

TFT paper

9. Conclusion

The soberingly large MAPEs of all models (twice as large as required) mean that time series forecasting of blood glucose for treatment decisions is not feasible, neither with classic ARIMA based models, nor with newer, neural network based models - at least not at the level explored in this project. Yet so-called closed-loop systems that monitor blood glucose and automatically administer insulin have been FDA approved since 2016. These devices use classic control algorithms that are decades old. As more machine learning algorithms are being developed or adapted to times series prediction (see TFT), their performance will hopefully improve.

FDA approves insulin auto-delivery device, Closed Loop Control Algorithms