BrisT1D Blood Glucose Prediction

Mansoor Ahmed[1] **Hafsa Akhtar**[1] **Bikesh Bimali**[1] **Alenka Tang**[1]
Paavani Aashika Maddi[1] **Shoaib**[1]
[1]Georgia State University, Atlanta, GA 30303
{mahmed76, hakhtar1, bbimali1, xtang8, pmaddi2, shoaib}@student.gsu.edu

Abstract

Insulin-producing beta cells in the pancreas are attacked by the immune system in type 1 diabetes (T1D), a chronic autoimmune disease that results in either total or almost total lack of insulin production. T1D patients have a difficult time controlling their blood glucose (BG) levels since they have to constantly check and modify their insulin dosages in reaction to changing circumstances, including blood glucose, activity, and carbs. Hence, it is challenging to develop robust estimation models that can precisely forecast future blood glucose levels in real time based on the previous few hours of data, allowing for prompt treatments and preparedness. In this work, we used the longitudinal data from 9 people with T1D collected over the span of 3 months, including continuous glucose monitoring (CGM) data, insulin dosages, carbohydrate intake, and activity levels collected from a Kaggle Competition, BrisT1D [1]. We provide a comparison of different machine learning-based methods for imputing the sparse dataset and blood glucose prediction. Overall, our results show that the forward-backward fill-based imputation strategy performed better, while the random forest (RF) regressor produced better blood glucose prediction accuracy with a mean squared error (MSE) of 2.61 years. Our source code is available at: https://github.com/mansorrahmed/brist1d.

1 Introduction

Type 1 diabetes (T1D) is an autoimmune disorder where the pancreas stops producing insulin, requiring patients to constantly monitor blood glucose levels and administer insulin. Predicting future glucose levels can improve disease management and prevent dangerous blood glucose extremes. Recent advances in continuous glucose monitoring and wearable devices have generated comprehensive datasets capturing factors affecting glucose dynamics. This creates an opportunity to leverage machine learning for improved glucose prediction. We investigate machine learning approaches to predict blood glucose levels one hour ahead using six hours of multimodal data from T1D patients. Our work focuses on developing models that generalize across patients, exploring various architectures from traditional regression to deep learning methods like LSTM and GRU networks. We evaluate different data preprocessing and imputation techniques to handle missing values.

2 Dataset

The dataset comprises young adults with T1D in the UK using continuous glucose monitors, insulin pumps, and smartwatches. Data includes blood glucose, insulin, carbohydrate intake, and activity levels, aggregated at five-minute intervals. Each sample contains six hours of historical data to predict

[1]https://www.kaggle.com/competitions/brist1d/overview

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

glucose levels one hour ahead. The training set uses three months of overlapping, chronological samples from nine participants. The test set contains non-overlapping samples in random order from fifteen different participants, ensuring evaluation on completely unseen individuals.

2.1 Exploratory Data Analysis (EDA)

The dataset contains 177,024 rows and 508 columns, with the target variable `bg+1:00` representing blood glucose one hour ahead. Features include historical blood glucose readings, insulin intake, carbohydrate consumption, heart rate, steps, and activity levels. Key findings from exploratory data analysis:

- Significant missing values exist, particularly in activity and carbohydrate data, while blood glucose, calories, heart rate, insulin, and steps have better coverage
- Blood glucose and insulin readings show clear temporal patterns in 24-hour cycles
- Features include both categorical and continuous variables
- The target variable follows a left-skewed normal distribution, suitable for regression modeling

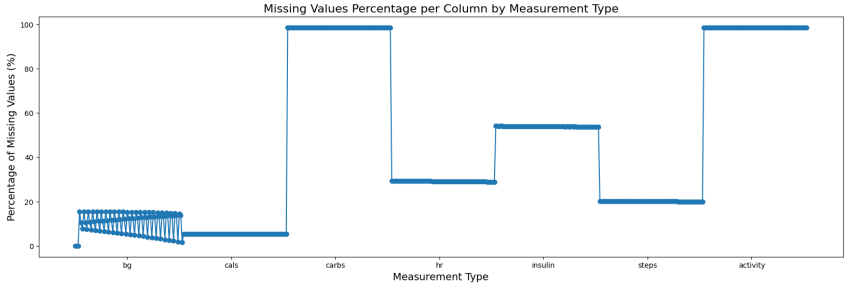

Figure 1: The percentage of missing values for all features.

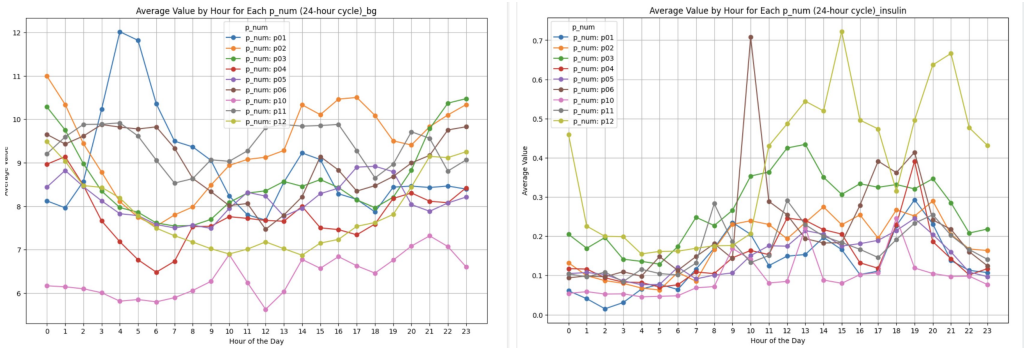

Figure 2: (from left to right) The blood glucose (BG) and insulin features aggregated at each hour for a period of 24 hours for the patients randomly sampled from the training set.

2.1.1 Correlation Analysis

One of the primary insights derived from the EDA was the correlation between the various features and the target variable `bg+1:00`. The correlation heatmap (Figure 4) provides a visual representation of the linear relationships between the features.

In this heatmap, values range from -1 to 1, where a correlation of +1 indicates a perfect positive linear relationship between two variables, -1 indicates a perfect negative linear relationship, and 0 indicates no linear relationship. Features like blood glucose levels at earlier times (e.g., `bg-5:55`, `bg-5:50`) were observed to have a strong positive correlation with the target variable. This suggests that recent blood glucose levels are good predictors of future values. Conversely, activity-related features had weaker correlations, which justified their removal during preprocessing.

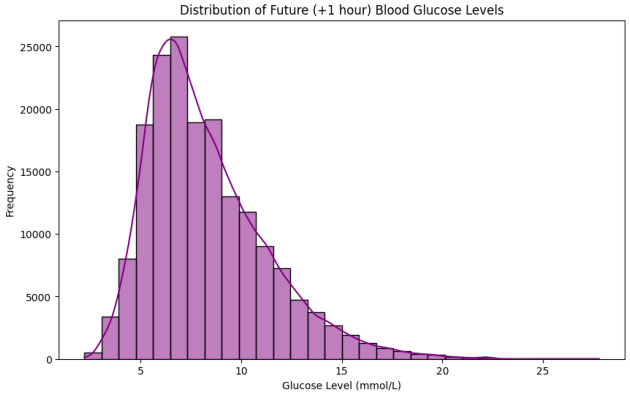

Figure 3: The distribution of the target future blood glucose (BG) variable (bg+1:00).

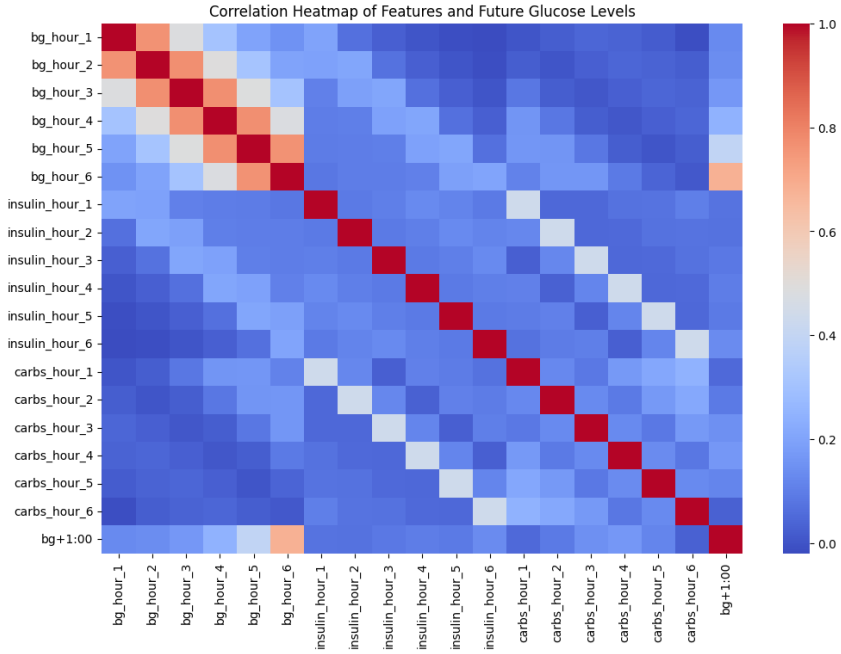

Figure 4: Correlation heatmap between features and the target variable.

2.1.2 Temporal Relationship of Features with the Target Variable

In addition to the correlation heatmap, the temporal relationship between the features (especially blood glucose levels) and the target variable was visualized using line plots (Figure 5). These plots showed how past values of blood glucose influence future values.

A clear trend was observed where recent blood glucose levels (such as `bg-0:05`, `bg-0:10`) closely track the future blood glucose levels, with an almost linear progression. This highlights the temporal dependency of the target variable on its past values, reinforcing the time-series nature of the data.

3 Preprocessing

3.1 Handling Null Values

One of the primary challenges with the dataset was the presence of a significant number of missing values. The dataset exhibited a high percentage of null values in various columns, particularly in the

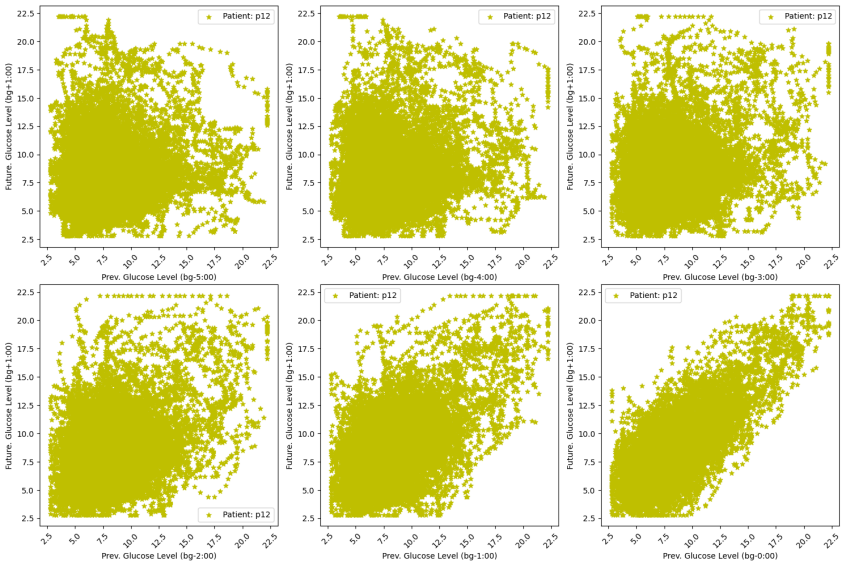

Figure 5: Temporal relationship between blood glucose features and the target variable.

blood glucose readings at different time intervals. To address this issue, several imputation strategies were explored and implemented.

3.1.1 Imputation Strategies

Different imputation strategies were employed to handle the missing values effectively:

- **Mean/Median Imputation:** Simple methods filling missing values with feature-wise mean or median. While computationally efficient, these methods ignore temporal dependencies and may not capture the data's dynamic nature
- **Forward/Backward Imputation:** Replaces missing values with the most recent preceding value or next available value. This method preserves temporal relationships and is particularly suitable for time-series data.
- **Linear Interpolation:** Performs linear interpolation between nearest known values, with forward/backward fill handling edge cases. This approach ensures smooth transitions between values in time-series data.
- **K-Nearest Neighbors:** KNN estimates missing values using k=5 nearest neighbors, considering the similarity between data points and capturing local patterns in the data.
- **Multiple Imputation by Chained Equations (MICE):** MICE iteratively predicts missing values by modeling relationships between all variables. It uses 5 imputations and 10 maximum iterations, providing more robust estimates despite higher computational costs.

3.2 Data Aggregation

To capture the temporal relationships in the data, the features were aggregated by category over different time intervals. The aggregation process involved extracting hourly data and computing the mean or sum for each category. This process helps in reducing the dimensionality of the data while retaining important temporal information.

3.3 Exclusion of High Null Percentage Columns

We excluded features with more than 50% missing values to improve data quality and model performance. These columns include carbohydrates, insulin, and activity.

3.4 Data Standardization

Data standardization was performed to ensure that all features have a mean of 0 and a standard deviation of 1. This step is crucial for many machine learning algorithms, as it helps in achieving faster convergence during training and improves the performance of the models. The standardization was applied to both the training and test datasets.

4 Problem Formulation

4.1 Regression Approach

The problem can be framed as a regression task where the goal is to learn a mapping from input features to the target variable, which is the blood glucose level one hour into the future. Let the dataset be represented as a matrix $X \in \mathbb{R}^{n \times m}$:

$$X = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1m} \\ x_{21} & x_{22} & \dots & x_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nm} \end{pmatrix}, \quad y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

where n represents the number of training samples, m represents the number of features, and y_i represents the target blood glucose level for the i-th sample. The goal is to learn a function f such that:

$$y_i = f(\mathbf{x}_i) + \epsilon_i$$

where $\mathbf{x}_i$ is the feature vector for the i-th sample, f is the regression function, and ϵ_i is the error term.

4.2 Forecasting Approach

The problem can also be framed as a forecasting task where the goal is to predict future blood glucose levels based on past observations. Given multivariate time series data $\mathbf{X}_t$ (blood glucose readings, insulin dosage, calories, and steps data), the objective is to forecast the blood glucose reading one hour into the future y_{t+1}:

$$y_{t+1} = g(\mathbf{X}_t) + \epsilon_t$$

where $\mathbf{X}_t$ represents the time series data up to time t, g is the forecasting function, and ϵ_t is the error term.

To capture the time-series relationship effectively, the data was transformed as illustrated in Figure 6. The transformed data includes various features such as blood glucose levels, heart rate, calories burned, and steps walked, measured over a six-hour period. The goal is to predict the future blood glucose level (indicated by the red star) based on these time-series features. It can be clearly seen in the given figure that the previous trends in the blood glucose levels (blue-colored signal) are highly predictive of future levels (shown with red label).

The structure of the data is transformed to capture the time-series relationship before being given as input to forecasting models, as shown in Figure 7. Each row represents a time step, and each column represents a different variable. The data is organized by patients and time steps, allowing the model to learn from the temporal relationships within the data.

5 Models

Keeping in view the above two formulations of the problem, we trained different baseline prediction models. Hence, in this section, we briefly discuss these models and the hyperparameters chosen to achieve the best performance.

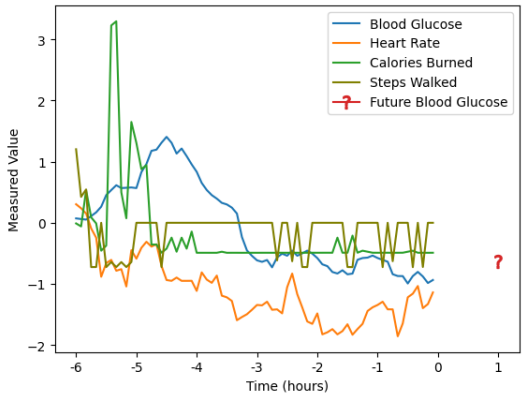

Figure 6: The time-series relationship of the previous blood-glucose-related signals or data with the target variable.

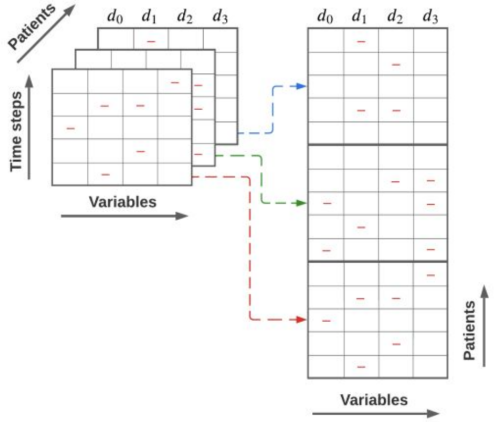

Figure 7: The transformed data for the forecasting problem (source: (1)).

5.1 Linear Regression

After preprocessing, a baseline linear regression model was trained using the preprocessed dataset. The primary goal of this model was to predict the blood glucose level one hour in the future (`bg+1:00`) based on historical data. The model's performance was evaluated using standard regression metrics, such as mean squared error (MSE), and serves as a benchmark for future models.

$$\hat{y} = \beta_0 + \sum_{j=1}^{p} \beta_j x_j$$

5.2 Ridge Regression

A variant of linear regression that includes a penalty term to reduce overfitting by constraining the size of the model's coefficients. The objective function for ridge regression is:

$$\text{Minimize } \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

λ: Regularization strength. A higher λ shrinks coefficients more aggressively. Features like blood glucose measurements at consecutive intervals are highly correlated.

5.3 Lasso Regression

Least Absolute Shrinkage and Selection Operator (lasso) is a linear regression model. Useful for feature selection because it can shrink some feature coefficients to exactly zero, effectively excluding those features from the model. Useful in dealing with high-dimensional datasets with irrelevant or redundant features. Features with little to no predictive power can have their coefficients shrunk to zero, simplifying the model. Lasso regression might identify that only a subset of features significantly contribute to future predictions, while setting other coefficients to zero.

5.4 Decision Tree Regressor

Predicts continuous numerical values by splitting data into subsets based on feature values. Effective for blood glucose prediction, as relationships between features and future glucose levels may not be linear. Proper tuning and constraints are necessary to prevent overfitting and achieve reliable predictions. The dataset is split into smaller subsets based on feature thresholds. To make a prediction for a new input, the model traverses the tree using the feature values until it reaches a leaf node.

5.5 Random Forest

An ensemble learning method that combines multiple decision trees to improve prediction accuracy and reduce overfitting. Trains several decision trees on different subsets of the data and averaging their predictions for regression tasks. The training dataset is randomly sampled with replacement to create multiple subsets. At each split in a tree, only a random subset of features is considered for splitting. Identifies the most relevant factors influencing future blood glucose levels. Missing values in data have less impact since predictions are based on multiple trees.

5.6 Gradient Boosting

An advanced ensemble learning algorithm that builds models sequentially, where each new model focuses on correcting the errors of the previous ones. In each iteration, a new decision tree (weak learner) is trained to predict the residual errors of the current model. The predictions from the new tree are added to the overall model with a small weight (learning rate) to improve performance while avoiding overfitting. Without proper regularization, it may overfit small datasets.

5.7 Hist Gradient Boosting

Uses histogram-based binning for feature discretization, faster on large datasets. Faster and memory-efficient, especially for large datasets. L2 regularization (ridge) to prevent overfitting. May struggle with complex feature interactions. Optimized for parallelism using histogram-based approach.

5.8 XGBoost

Traditional decision trees without binning, slower on large datasets. Can be slower and more memory-intensive for large datasets. L1 and L2 regularization (lasso and ridge) for more flexibility. Better at capturing complex, non-linear relationships between features.

5.9 Long Short-Term Memory networks (LSTMs)

Blood glucose levels are influenced by a combination of factors (e.g., food intake, exercise, insulin, stress) that interact over time. Capturing these relationships requires remembering events and patterns from the past. LSTMs can analyze the sequential input data (e.g., six hours of past glucose readings) and learn patterns that predict the glucose levels an hour into the future. LSTMs are robust to noisy or missing data when paired with effective imputation strategies. They can learn to infer patterns even if the input sequence has gaps. The ability to retain long-term information helps improve prediction accuracy for blood glucose levels, especially for detecting trends and preventing extreme fluctuations.

5.10 Gated Recurrent Unit

Unlike LSTMs, GRUs rely solely on a single hidden state to store information. This lack of a separate cell state simplifies the architecture and reduces memory requirements. The single hidden state still retains sufficient context for most sequential tasks, especially when long-term dependencies are less critical. GRUs are effective for tasks where the relevant dependencies are within shorter time frames, such as the six-hour input window used for blood glucose prediction. Like LSTMs, GRUs can handle noisy or imputed data well, especially when combined with effective preprocessing techniques.

6 Results

6.1 Evaluation Metrics

The performance of the prediction models is evaluated using the MSE, which is a measure of the average squared difference between the predicted values and the actual values. MSE is defined as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

where y_i is the actual target value for the i-th sample, $\hat{y}_i$ is the predicted value for the i-th sample, and n is the total number of samples.

6.2 Performance of Models Trained with Different Imputation Strategies

We compared the MSE of various machine learning models as shown in fig 8. Our results show that XGBoost and Random Forest generally performed better by consistently showing low MSE across all imputation strategies, making them robust choices for blood glucose prediction. Decision Tree Shows the highest MSE across most imputation strategies, indicating poorer performance compared to other models. LSTM and GRU Performed well, but GRU shows slightly higher MSE for some imputation strategies (e.g., KNN).

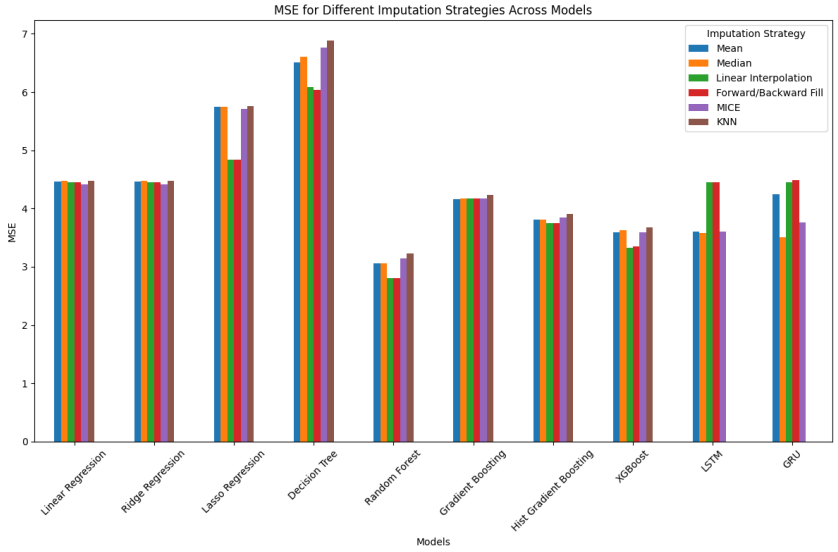

Figure 8: MSE for different imputation strategies across different trained models.

6.3 Preprocessing Times

Figure 9 shows the preprocessing times for the different imputation strategies we used. Simple methods like Mean, Median, and Linear Interpolation are computationally efficient and require minimal time. Forward/Backward Fill is slightly more time-consuming but still efficient. MICE and

KNN are significantly more time-intensive, likely due to their complexity. MICE involves iterative modeling, and KNN searches for similar instances to impute missing values, which is computationally expensive.

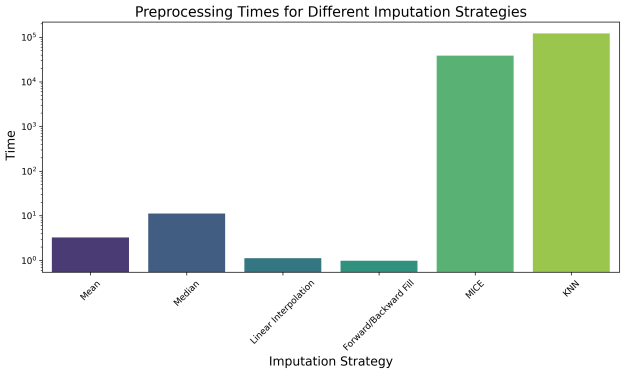

Figure 9: Preprocessing times for different imputation strategies across models (y-axis transformed to log-scale to make the comparison easier).

6.4 Models Training Time Complexity

The bar chart in Figure 10 compares the training times (in seconds) for different models across various imputation strategies. The y-axis represents the training time on a logarithmic scale, while the x-axis lists the different models. Each bar represents a different imputation strategy, color-coded for clarity. Here are some of the key observations from the figure.

Linear Regression, Ridge Regression, and Lasso Regression generally have the shortest training times across all imputation strategies. This is due to their simplicity and the efficiency of linear algorithms. Decision Tree and Random Forest models showed moderate training times. Random Forest, being an ensemble of decision trees, has a longer training time compared to a single decision tree but is still relatively efficient. Gradient Boosting and Hist Gradient Boosting models exhibited longer training times due to their sequential nature and the need to build multiple models. Hist Gradient Boosting shows a slight improvement in training time efficiency compared to traditional Gradient Boosting. XGBoost model has one of the longest training times, reflecting its complexity and the need for multiple iterations to optimize the boosting process. The neural network models such as LSTM and GRU had the longest training times, especially when using more complex imputation strategies like MICE and KNN. This is due to the computational intensity of training neural networks and the additional complexity introduced by these imputation methods.

6.5 Learning Curves for LSTM and GRU

As shown in Figure 11, both GRU and LSTM models show effective initial learning, with a rapid decrease in losses within the first 10 epochs. The GRU model exhibits more stability in the loss values compared to the LSTM model, which shows more fluctuations. Unlike LSTM, The GRU model shows a noticeable spike in losses around epoch 70, indicating potential overfitting. Both models show a trend towards convergence, with losses stabilized by the end of the training process.

6.6 Kaggle Leaderboard

We submitted our best-performing models to the Kaggle competition and got the best results for random forest (MSE=2.61) and XGBoost (MSE=2.60) regressors. The private leaderboard is calculated with approximately 71% of the test data. The test data includes records from 4 patients who were not present in the training and validation sets.

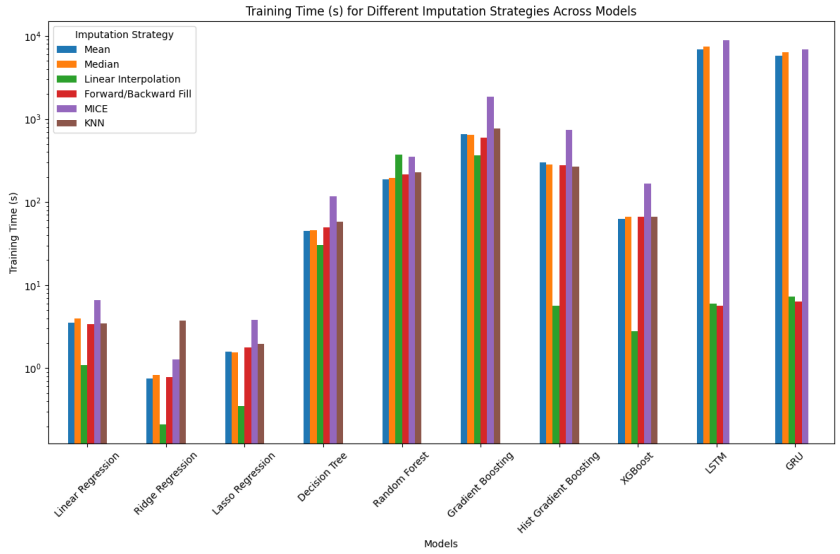

Figure 10: Training time complexity of different models with different imputation strategies.

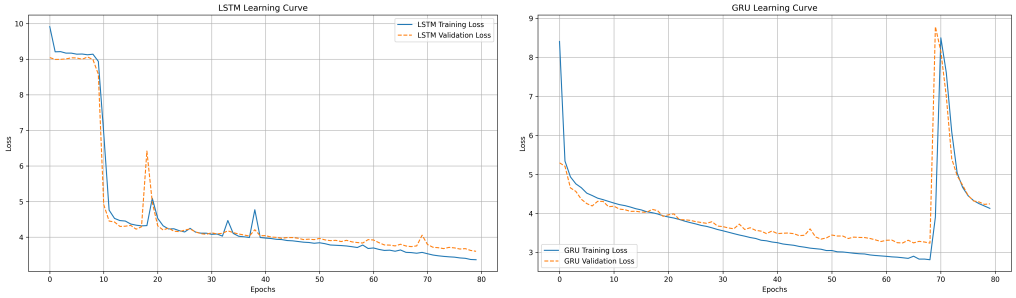

Figure 11: The learning curves of LSTM and GRU.

7 Conclusion & Future Work

In conclusion, the random forest regressor produced the best results for predicting future blood glucose based on the previous few hours of data. Moreover, random forest, having fewer parameters, can potentially be deployed to the T1D patient's smartwatches for real-time forecasting. This is not a traditional time series forecasting problem, and there were some issues with the data. Instead of relying on aggregated features, structuring the data as a true time series problem could unlock additional temporal patterns. Techniques such as Recurrent Neural Networks (RNNs), Long Short-Term Memory networks (LSTMs), or Temporal Convolutional Networks (TCNs) could then be applied. Issues with data quality or structure might have affected the preprocessing and results, highlighting the need for better handling of missing values or noisy data. Leveraging additional data from smartwatches (e.g., heart rate, activity levels) can provide richer features to improve predictive accuracy. Another future direction could be to stratify the training-testing samples based on the patient IDs so as to avoid data leakage and make the models more generalizable.

References

[1] M. Kazijevs and M. D. Samad, "Deep imputation of missing values in time series health data: A review with benchmarking," *Journal of biomedical informatics*, p. 104440, 2023.