**AML Project Code Contribution**

We reused a portion of the code found in the Python notebooks shared by users in the Kaggle community for the blood glucose prediction competition. Mainly, we drew inspiration for some of the visuals in our presentation and report from the analysis found in the ‘Explorations’ Kaggle notebook because the creator did well in visualizing the important variables that contained many values. To save time, we formatted our plots in a similar way to effectively visualize the key components of our data. Also, when studying what models we could incorporate as part of our project we stumbled into useful notebooks that contained models we learned in the course, which we used as references for how we developed, modified and trained our models and data:

(Sources)

[🩸Blood Glucose Prediction - Explorations](https://www.kaggle.com/code/docxian/blood-glucose-prediction-explorations)

[BrisT1D-AutoGlucon Ensemble [Baseline]](https://www.kaggle.com/code/suvroo/brist1d-autoglucon-ensemble-baseline)

[Key Insights from XGBoost 🔮 score: 2.200](https://www.kaggle.com/code/vineetgupta2023/key-insights-from-xgboost-score-2-200)

[BrisT1D Blood Glucose Prediction | LightBGM](https://www.kaggle.com/code/valentinemwai/brist1d-blood-glucose-prediction-lightbgm)

[{ ANN 🩸 📊 } Blood Glucose Prediction](https://www.kaggle.com/code/shahzaibmalik44/ann-blood-glucose-prediction)

Some of our imputation and data preprocessing methods were inspired by the ‘ANN Blood Glucose Prediction’ notebook especially, since it detailed a process flow from data analysis all the way to neural network results nicely. For the neural network portion, however, we expanded a bit further by considering a recurrent neural network since our data exhibits characteristics of time-series data.

**LLM Usage**

For this project, we utilized ChatGPT to assist with several key aspects of development. During the data preprocessing stage, ChatGPT provided valuable insights into techniques such as forward/backward fill and KNN imputation, which were instrumental in handling missing values in our time-series data. These suggestions helped streamline the data cleaning process and ensured that the dataset was well-prepped for training. In addition, while we did have Kaggle code files to refer to for some of our initial ensemble, boosting, and Elastic Net models, we utilized LLMs to troubleshoot occasional errors. Several charts in the tree and hybrid model approaches were generated with the assistance of LLMs to ensure that correct insights were being conveyed and/or to enhance visual elements such as labels and aesthetics.

The idea to incorporate a transformer into our hybrid model originated from Prof. Ghosh’s recommendation. Building on this suggestion, we conducted research on transformer architectures and their applications in time-series forecasting. To implement the hybrid RNN + transformer model, we turned to ChatGPT for guidance on handling sequential time-step tensor matrices and the gated recurrent unit required for the RNN and initializing the transformer layer using PyTorch’s nn.TransformerEncoder model. ChatGPT also suggested hyperparameters to tune for the RNN and transformer components, such as hidden\_dim, ff\_dim, num\_heads, and num\_layers, which helped optimize the model’s performance. While these contributions informed specific implementation details, the codebase, training workflow (including the train/test splits and train-epoch loops that we learned about in class), and model refinement were primarily our own efforts.