# Dataset:

There are a total of 30 patients, including adults #001-010, adolescents #001-010, and children #001-010. Each patient has 20 days of data. The data comprises three variables: "CGM" (representing glucose values), "CHO" (representing carbohydrate intake), and "insulin" (representing insulin values). Data for each variable is recorded every 3 minutes. It's important to note that the carbohydrate intake remains consistent across all patients, both in terms of timing and amount.

# Algorithm:

There are four types of algorithms included, each serving a different purpose in training and utilizing the model.

* Named "transfer\_learning," this method doesn't require training the model from scratch. Instead, each patient utilizes a pre-trained model (obtained from the final model of federated\_deep\_learning) and applies transfer learning to create personalized models by fine-tuning them according to each patient's data.
* Named "centralized\_learning", this method simulates a scenario where all patients send their data to a centralized center. The center utilizes the collective data to train a single model. Each patient then utilizes that model and applies transfer learning to create personalized models by fine-tuning them according to each patient's data.
* Named "self\_train\_RNN", this method simulates a scenario where each patient exclusively uses their own data to train a personalized model.
* Named "federated\_deep\_learning", this method simulates a scenario where only a few patients share their data with a centralized center. The center trains an initial global model using the shared data, which is then sent to the remaining patients (referred to as local patients) to train their local models. Each patient utilizes their own data to train their personalized local model. The parameters of the local models are then shared with the centralized server to generate a global model. This process undergoes several rounds, where the global model is sent to the local models and vice versa, constituting one round. Ultimately, a well-trained federated deep learning model is obtained, which all patients can use.

Note: The four algorithms can perform independently without affecting the other algorithms, except for "transfer\_learning." This algorithm relies on a pre-trained model from the federated\_deep\_learning algorithm when 5 days of data are included. If more or fewer days of data are included, or if changes are made to the number of layers and neurons in the RNN framework, the optimal pre-trained model may change. In such cases, it is necessary to run the "federated\_deep\_learning" algorithm to create a new model, which can then be used for transfer\_learning.

# Code guidelines:

All the functions used in the algorithm are stored in the folder named "functions".

These functions are designed to be flexible, allowing the parameters to be adjusted to meet various requirements. For each algorithm listed above, both the model and the results (RMSE calculated based on the test data) are automatically saved in a folder generated based on the number of days of data and the algorithm name. This allows for easy performance evaluation.

The following parameters can be adjusted to meet the performance requirements:

num\_days = 5

mins\_before\_predict\_list = [30] # 30 mins = 0.5 hour

window\_slide\_list = [180] # 180 mins = 3 hours

monitor = "val\_loss"

learning\_rate = 0.001

epochs = 10

neuron\_nums = [64,32]

batch\_size = 32

num\_rounds = 5

seed\_number = 1

A black background with white text

Description automatically generated

I want to clarify the layers and framework of the RNN. The RNN architecture has been redefined to include one Input Layer, two hidden layers, and an output layer. The number of neurons in the input layer depends on the size of the training data. As it is time-series data, we cannot randomly choose the data for training. Therefore, we designate the last day's data as test data, the day before the last day's data as validation data, and the remaining days' data as training data. Consequently, the number of neurons in the first layer depends on the input number of days' data. The two hidden layers are configured with 64 and 32 neurons, respectively, and the output layer consists of a single neuron.