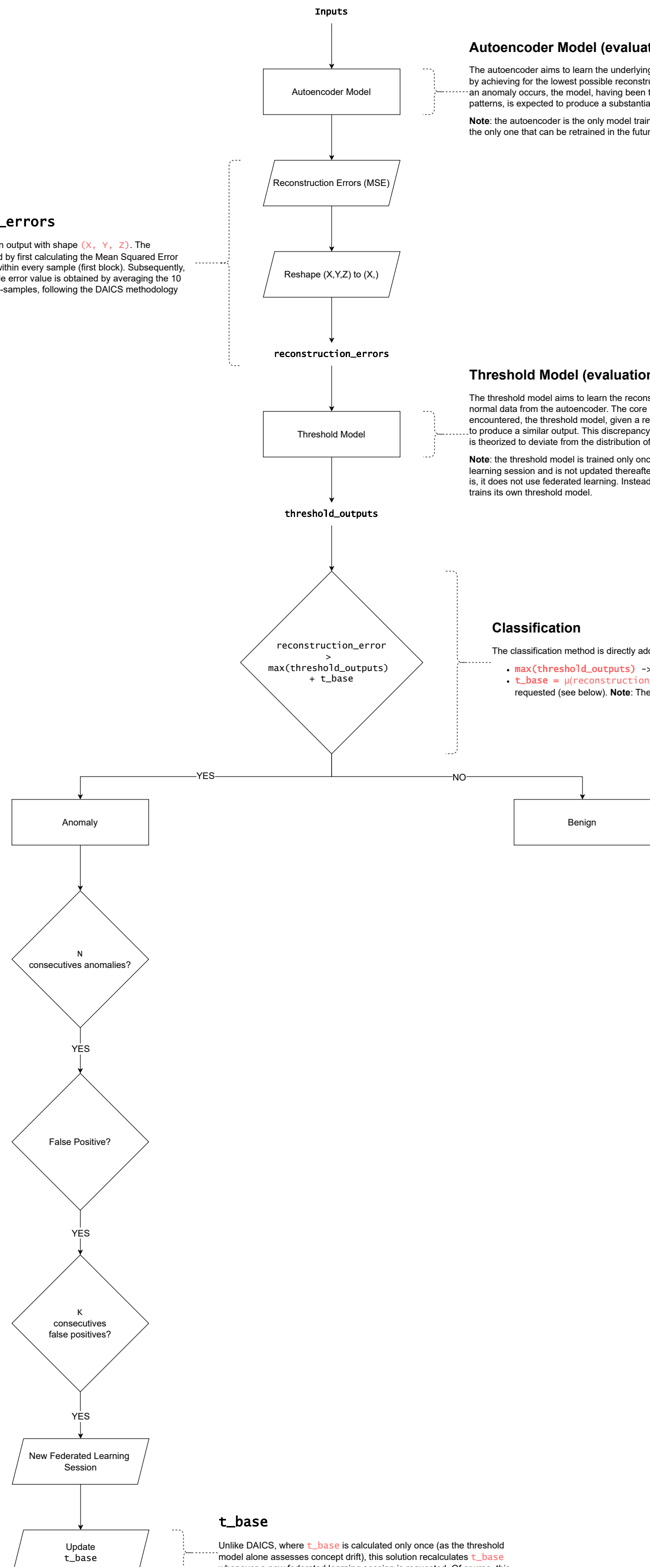
`Inputs`

The autoencoder receives mixed samples (normal + attack). The shape is `(X, Y, Z)`, where `X` is the number of samples, `Y` is the window size (10), and `Z` is the number of features (51).



Autoencoder Model (evaluation mode)

The autoencoder aims to learn the underlying distribution of normal data by achieving for the lowest possible reconstruction error. Consequently, when an anomaly occurs, the model, having been trained solely on normal patterns, is expected to produce a substantially higher reconstruction error.

Note: the autoencoder is the only model trained using federated learning and the only one that can be retrained in the future if needed.

`reconstruction_errors`

The autoencoder produces an output with shape `(X, Y, Z)`. The reconstruction error is derived by first calculating the Mean Squared Error (MSE) for each sub-sample within every sample (first block). Subsequently, for each main sample, a single error value is obtained by averaging the 10 MSEs from its respective sub-samples, following the DAICS methodology (second block).

Threshold Model (evaluation mode)

The threshold model aims to learn the reconstruction error characteristics of normal data from the autoencoder. The core idea is that when anomalies are encountered, the threshold model, given a reconstruction error, will struggle to produce a similar output. This discrepancy arises because anomalous data is theorized to deviate from the distribution of normal data.

Note: the threshold model is trained only once during the initial federated learning session and is not updated thereafter. Its training is centralized; that is, it does not use federated learning. Instead, each client independently trains its own threshold model.

Classification

The classification method is directly adopted from DAICS.

- **`max(threshold_outputs)`** -> Calculated each time.
- **`t_base =`** `μ(reconstruction_errors) + σ(reconstruction_errors)` -> Recalculated only when a new federated learning session is requested (see below). **Note**: The system initializes with a `t_base` value derived from the training session.

Anomaly Classifications

Controlling false positives is easier than false negatives. In this scenario, an operator can verify the classification, and their feedback can then be used to update the entire system.

Benign Classifications

Controlling false negatives in benign classifications is extremely challenging, if not impossible. Unlike false positives, where an operator can verify the classification, there isn't a similarly specific method to address false negatives.

N

Unlike other values, `N` is hardcoded. Dynamically determining `N` could introduce too many dynamic variables into the system, significantly increasing its complexity.

Operator

In this system, the operator provides optional feedback used to improve the entire system. While the specific method an operator uses to check for false positives is out of scope, their feedback is crucial: in cases of false positives, it is used to improve the local model and, under certain conditions, also the global one. The ultimate objective is to create a system that minimizes the need for operator intervention.

K

Unlike other values, `K` is hardcoded. Dynamically determining `K` could introduce too many dynamic variables into the system, significantly increasing its complexity.

New Federated Learning Session

The client sends a request to initiate a new federated learning session. The starting model is the `global_model`. Once convergence is achieved, the system would theoretically have internalized the false positives.

`t_base`

Unlike DAICS, where `t_base` is calculated only once (as the threshold model alone assesses concept drift), this solution recalculates `t_base` whenever a new federated learning session is requested. Of course, this is done at the end, once convergence is achieved.