**Chapter 2: Sliding Window Algorithm**

Version 1: Without ANN- Not automatic

1. Load and Prepare Data:

* Select only numeric data and fill in missing values with respective column means.

**Stage (I):**

2. Define Sliding Window Parameters

* Set the window size and step size for creating sliding windows from the time series data:
  + The sliding window technique involves breaking the time series data into smaller overlapping segments (windows). The window size determines the length of each segment, and the step size determines how much to move the window for each new segment. If the results of all performance metrics calculated the last Step (10) is not more than 99%, the algorithm returns to this Step (2) and increase window size 10 minutes and repeat.

3. Create Sliding Windows:

* Generate sliding windows from the time series data, creating subsamples:
  + The original time series data is divided into multiple overlapping segments (subsamples) using the sliding window parameters. Each subsample is treated as a smaller instance of the original time series for further analysis.

4. Extract Features from Subsamples:

* Calculate statistical features (mean, standard deviation, skewness, and kurtosis) from each sliding window (subsample):

5. Anomaly detection with IF clustering model:

* An IF model on the extracted features is trained:
  + The IF is an unsupervised anomaly detection algorithm. It works by randomly selecting features and then partitioning the data. Anomalies are more likely to be isolated quickly since they differ significantly from the majority of the data.
  + In this step, the features extracted from each subsample (mean, standard deviation, skewness, kurtosis) are fed into the IF. The model learns the structure of the data by identifying patterns that are common across the majority of the samples and detecting outliers that deviate from these patterns.
* Generate new labels based on the anomaly detection (1 for anomaly, 0 for normal). These labels are referred to as the "new labels" and are used in subsequent steps of the analysis.

**Stage (II):**

6. Train-Test Split, Features’ normalization, and Oversampling

* Split the subsamples into training (80%) and testing (20%) sets based on the new labels generated by the IF.
* Apply feature scaling to the training and testing sets to standardize the features (mean=0, std=1):
  + This scaling ensures that all features contribute equally to the model, preventing any feature with a larger range from dominating the analysis.
* Use RandomOverSampler to balance the classes in the training data by oversampling the minority class (the dataset includes more normal subsamples than anomalies).

7. Train RF Model with Early Stopping:

* Train an RF classifier on the resampled training data:
  + The RF model is a supervised learning algorithm that builds multiple decision trees during training. Each tree is trained on a random subset of the data, and the final classification is based on the majority vote of all the trees.
  + The resampled training data (with balanced classes) is used to train the RF model. The model learns to classify the subsamples based on the features and the new labels provided by the IF.
* Implement early stopping based on training accuracy to avoid overfitting:
  + Early stopping is a technique used to prevent the model from overfitting the training data.
  + During training, the model's performance is monitored over several epochs (iterations). If the model's accuracy on the training data stops improving after a certain number of epochs (indicating that further training may lead to overfitting), the training process is stopped early. The best model from the earlier epochs is then selected for making predictions on the test set.

8. Make Predictions:

* Use the trained RF model to make predictions on the test set:
  + The trained RF model is used to predict the labels of the subsamples in the test set.
* Calculate probabilities and initial predictions for each test subsample:
  + The model outputs the probability of each subsample belonging to each class (normal or anomaly), as well as an initial prediction based on these probabilities.

9. Classify with Abstaining (Reject Option):

* For various lambda thresholds, classify the test set, abstaining from predictions that are uncertain
  + A range of lambda thresholds is applied to the model's predictions. If the model's confidence in a prediction (based on probability) is below the threshold, the prediction is abstained from.

10. Calculate Performance Metrics. If the target performances are not met, return to step 2.

* For each lambda threshold, calculate confusion matrix elements (TP, TN, FP, FN) and compute accuracy, precision, recall, F1 score, and specificity.