Interpretable, discrete multivariate time-series classification, using wire bond signal data from semiconductor manufacturing

Rosman, Cas

Award date:
2022

Master thesis

1MSE45

Interpretable, discrete multivariate time-series classification, using wire bond signal data from semiconductor manufacturing

Name	Role	Student ID
C. Rosman	Author	1503758

Name	Role
A. Akçay	First assessor, Eindhoven University of Technology
I.J.B.F. Adan	Second assessor, Eindhoven University of Technology
C. Imdahl	Third assessor, Eindhoven University of Technology
K. Schelthoff	Supervisor, NXP Semiconductors N.V.

Eindhoven, Tuesday 13th December, 2022

Interpretable, discrete multivariate time-series classification, using wire bond signal data from semiconductor manufacturing

Cas Rosman

***Abstract*—In this study, we propose a methodology for discrete multivariate time-series classification with automated signal and feature selection. The methodology is applied to real operational machine signal data from the wire bonding process at NXP Semiconductors N.V. During wire bonding several sensors actively monitor the process, generating a discrete multivariate time-series for each device. After wire bonding, an automatic visual inspection labels the quality of each device. To the best of our knowledge, we are the first that focuses on predicting the wire bond quality for entire devices rather than individual wires. The proposed methodology consists of several steps. First, we extract features from the discrete multivariate time-series and train a baseline model on all features. Second, we use permutation feature importance as a sorter for a sequential backwards search to subsequently find the optimal signals and feature set. Finally, the classification performance when using the reduced signals and feature set is compared to the performance of the baseline model. We conclude that the dimensionality of the data can be significantly reduced without losing classification performance. The reduced dimensionality leads to highly interpretable classification results in the wire bond use case.**

***Index Terms*—Wire Bonding, Feature Extraction, Feature Selection, Multivariate Time-Series Classification, Random Forest Classification, Permutation Feature Importance, Dimensionality Reduction, IIoT**

I. Introduction

SEMICONDUCTOR manufacturers are increasingly generating large amounts of data. Due to the rise of the Industrial Internet of Things (IIoT), production machines are more interconnected and equipped with all kinds of sensors that continuously monitor the production process for each device. Meanwhile, trends in the electronics market have led to demand for miniaturization and higher performance standards of semiconductors [1]. The higher demands for semiconductors present several challenges. One such challenge is the increased risk of product failures caused by the manufacturing process. To meet this challenge, the newly available manufacturing data presents an opportunity for the semiconductor industry, allowing for improved ways to predict and prevent product failures. However, generating valuable insights from manufacturing data becomes more challenging and computationally expensive when ample data is available. The high dimensionality of the newly available data presents a real challenge for the industry.

One process step particularly affected by the miniaturization and higher performance standards is wire bonding. This process is part of the semiconductor packaging process, which is the interface between the semiconductor manufactures and the electronics manufacturers. The semiconductor packaging process consists of several steps, including wafer backgrinding, sawing, die attaching, wire bonding, moulding, marking and more. To meet the increasing demands on semiconductors, the wire bonding process is advancing on all fronts, one of which is that the recent wire bond machines are equipped with more and more sensors to monitor the process continuously. Wire bonding, therefore, is a key example of a process that generates increasing amounts of data that can be used to conform the process to current and future market demands.

Wire bonding is the process of connecting the device to the outside world (lead frame) with a conductive wire. In this study, we consider thermosonic ball bonding. The process steps are depicted in Figure 1. In thermosonic ball bonding, a Free Air Ball (FAB) is formed before the first contact using an electronic spark. The capillary moves so the ball makes contact with the bond pad on the device. By applying scrubbing, force and heat, the ball is bonded to the bond pad, creating the ball bond. Similarly, the capillary moves towards the lead frame to create the stitch bond. The cycle is repeated for all wires on each device.

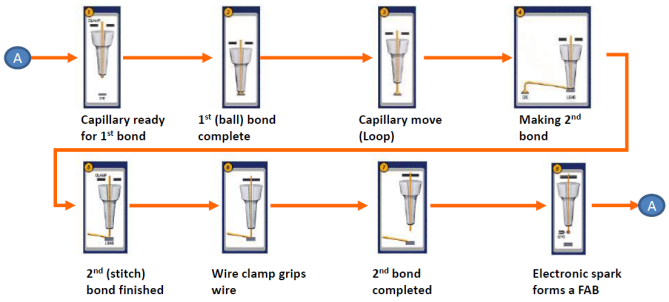

Figure 1. Thermosonic ball bonding process steps [2]

After wire bonding, an Automatic Optical Inspection (AOI) labels the device as either Pass or Fail. When a device fails the AOI, a human inspector labels the failure by assigning a failure mode. Examples of failure modes are Tie bar deformation, Sagging wire and Broken wire. During wire bonding, several sensors are active, generating a discrete multivariate time-series for each device. A subset of signals is depicted in Figure 2, the signal values are plotted on the y-axis, and the Discrete time (t) is plotted on the x-axis. The inspection result for this particular device is Fail due to Tie bar deformation. Note that the AOI does not generate labels on wire level, so

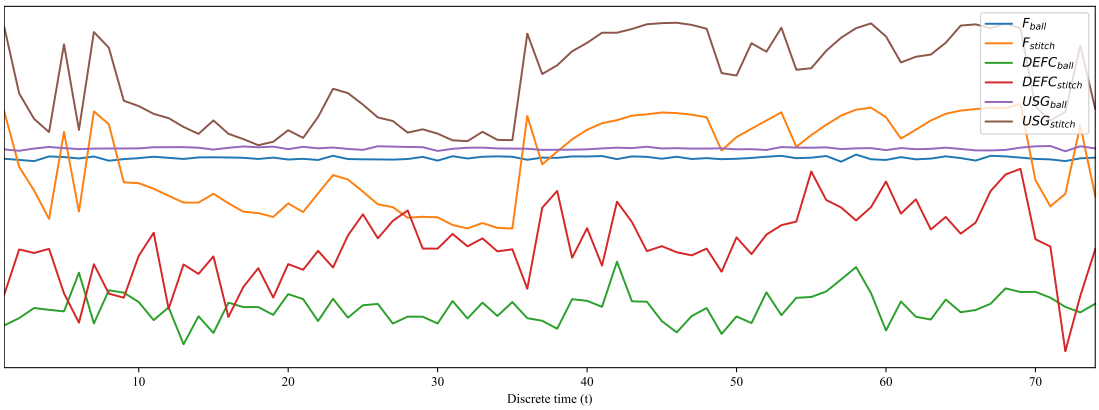

Figure 2. Subset of machine signal data for one device instance. The discrete time (t) represents the wire_id in wire bonding context. The subscripts ball and stitch indicate if the signal is measured for the ball or stitch bond on the corresponding wire_id. The signal abbreviations are specified in section IV.

the specific wire where the failure occurs (discrete time (t)) is unknown.

While there are many studies on wire bond quality prediction using machine signal data, current studies predict the quality of each wire, assuming the quality of each wire is known. While this works in some environments, such information is not widely available for all devices in real-world scenarios. Moreover, we focus on interpretability by reducing the dimensionality of the data. Therefore, in this study, we develop a novel method for discrete multivariate time-series classification with automated signal and feature selection. We apply the method to real-world wire bond machine signal data (Figure 2) from NXP Semiconductors N.V. The method enables us to classify devices (on the corresponding AOI labels) and find the least required signals and features to do so. Reducing the number of signals and features in the data improves the interpretability and computation performance of the classification algorithm [3]. Another advantage is that the risk of overfitting is significantly reduced when non-relevant signals and features are removed [4].

This paper is organized as follows: Section II presents the relevant literature. In Section III, we propose the methodology that is applied to real-world data from NXP Semiconductors N.V. in Section IV. We discuss the study in Section V and derive managerial implications in Section VI. The study is concluded and future research opportunities are identified in Section VII.

II. Literature Review

Wire bond quality prediction using machine signal data has been studied within semiconductor manufacturing for several decades. We first (i) identify early works within this research scope. Secondly (ii), we identify a literature stream where developments in sensor technology are the main driver of advancements in wire bond quality prediction. Third (iii), a clear shift from experimental setups to in situ process monitoring is identified in the literature. Finally (iv), several papers that focus on interpretable methods for wire bond quality prediction using machine signal data have been identified.

(i) Early work already proposes to use machine data obtained during wire bonding to predict bond quality. Pufall et al. [5] note that ultrasonic signal is a powerful predictor of bond quality that can be used to substitute Statistical Process Control (SPC) and minimize the number of destructive tests required to guarantee product quality. Similarly, Wang et al. [6] indicate that a Artificial Neural Network (ANN) model based on wire bond machine data can predict the pull strength of the wires.

(ii) After these two seminal works, developments in sensor technologies are the main driver of advancements in wire bond quality prediction. Or et al. [7] implement a piezoelectric sensor to monitor the changes in the ultrasonic vibrations during wire bonding. They use the frequency signal from the sensor to predict the shear strength of the wires by calculating the steady-state amplitude to the peak value of the second harmonic of the frequency. After installing a piezoelectric sensor on the surface of the horn to measure the vibrations during wire bonding, Zhang et al. [8] use the amplitude changing characteristics in joint time-frequency distribution of the measured vibrations as input to a ANN model to predict the shear strength of the wires. Subsequently, the use of a micro-temperature sensor underneath the bonding point to correlate the bonding temperatures to the expected shear strength of the wire bonds is researched by Suman et al. [9]. Zhong et al. [10] and Gual et al. [11] propose implementing a laser vibrometer to predict the bonding strength by measuring slight changes in the resonance frequency of the capillary.

These methods, however, are not suited for in situ monitoring since they change the configuration of the transducer system or device. The proposed setups, therefore, cannot be used in real-world manufacturing applications [12].

(iii) Instead, Ling et al. [13] use a Hilbert transform to obtain the complex valued real and imaginary part of the electrical signals of the transducer system without changing the initial setup of the wire bonding machines. They state that the measured waveforms are fully responsible for the bond quality. Finally, the data is fed to a Back Propagating Neural Network (BPNN) to predict bond strength. The use of electrical signals from the transducer system is developed further in the literature. Feng et al. [14] refine the process of mining electrical signals from the ultrasonic generator to predict bond quality. They extract features that are selected by

experts from the ultrasonic generator signal data and perform Principal Component Analyses (PCA) on these features. The selected features are subsequently used to predict the wire shear strength using an ANN. Wuwei et al. [15] use the same method as Feng et al. [14] but study the data generated by a piezoelectric sensor. Wang et al. [12] use wavelets to decompose the electrical signals of the transducer system and determine the characteristics of the fundamental frequency component together with the time and frequency-domain characteristics. Using these characteristics, Wang et al. predict the shear strength of bonds using a BPNN. In a series of publications, Arjmand et al. [16]–[18] first extract features from the electrical transducer signals and use the features to classify wire bonds in categories representing the corresponding quality. They also determine the survival probability of the bonds over time for each class using real-world data. The use of quality indices is introduced by Hagenkötter et al. [19]. They extract features from electrical signals of the transducer system together with the output of other sensors on the wire bond machines and compare the extracted features to a threshold for assessing the expected bond quality. They also determine the weight of each feature, meaning its contribution to the expected wire quality. Finally, PCA is performed on the features. By combining PCA and Tachugi methods, Tzeng et al. [20] optimize process parameters and determine the ones most influential to the wire bond process. Feng et al. [21] use the voltage current and signals from the ultrasonic generator, and extract the bonding features by combining Wigner-Ville distribution with empirical mode decomposition. The most important features are selected using PCA. Lastly, they train a ANN using the selected features to predict wire bond quality. For future research, Feng et al. [21] indicate their methodology can be enhanced by using a improved feature selection procedure.

Previously mentioned studies do not generate results that we consider to be interpretable in our use case because they do not explain what patterns in the data indicate that failures occur. The mentioned studies propose a variety of methods including prediction models that are difficult to interpret and PCA for dimensionality reduction. We do not consider PCA to be truly interpretable, since the different features need to be distinct from each other to be interpretable [22]. This means that the summarizing features obtained from PCA are not useful for interpretable classification in our use case.

The literature does propose methods that focus on interpretability. The following studies have a similar goal to our research. However, like all literature mentioned in this review, they require direct feedback (labels) on the quality of each individual wire. This feedback is not available for all devices when using real-world data.

(iv) Montealegre and Hagenkötter [23] propose a cytokine-formal immune network, which is able to deal with high dimensional data coming from the wire bond machines by incorporating feature extraction and selection methods to predict bond quality for each wire. Using weld power and horn displacement, Lee et al. [24] determine the key features of these signals to bond quality from sampled wires. They also provide a guideline for feature extraction and selection in process monitoring for wire bonding. Chun-Min et al. [25] develop a fuzzy quality prediction evaluation model for the wire bonding process. They create a process-quality index with a one-to-one mathematical relationship to the process yield of each wire.

The presented studies show that many researchers focus on predicting the quality of individual wires. The researchers rely on direct feedback on each wire's quality in the form of a label or quantitative measurement. However, when using real-world data, feedback on individual wire quality is not widely available for each device. To the best of our knowledge, no study focuses on predicting the wire bond quality for entire devices rather than individual wires. Due to a lack of direct feedback on the quality of individual wires and the increasing dimension of available data from real-world manufacturing, a method is required to classify wire bond quality for each device and reduce dimensionality from the input data. Such a method will allow the industry to generate valuable insights into data patterns correlated to quality issues and reduce computation / data storage cost.

III. Methodology

We propose a novel methodology for discrete Multivariate Time-Series (MTS) classification with automated signal and feature selection. We combine the MTS classification method proposed by Baldán et al. [26] with a feature reduction framework proposed by Schelthoff et al. [27]. Baldán et al. [26] classify MTS based on features they extract from the MTS. Schelthoff et al. [27] propose a permutation based feature reduction framework to find the relevant features for classification. We expand the methodology by implementing a selection $Threshold$ that allows the user to make a trade-off between dimensionality reduction and classification performance. We also add a signal selection step that determines the least required amount of signals for classification. The proposed methodology allows the user to classify labeled MTS with a reduced set of signals and features while keeping the classification performance within a certain $Threshold$ from the classification performance obtained when using all available data.

A. Data structure of labeled discrete multivariate time-series

The proposed methodology can be applied to any set of N labeled discrete MTS (number of devices) with length T (number of wires per device) and S amount of signals (active sensors in the machines). Table I presents the structure of the input data where:

- $n \in \{1, 2, ..., N\}$, is the instance of a MTS. In wire bonding context, n indicates the device instance.
- $t \in \{1, 2, ..., T\}$, is the discrete moment of time in MTS instance n. For wire bonding, t represents the wire_id.
- $x_s(n, t), s \in \{1, 2, ..., S\}$, represents the value of signal s measured for MTS instance n at time t.
- $y(n) \in \{0, 1\}$, indicates the binary label corresponding to MTS instance n. In wire bonding context, the AOI generates the label where 0 represents Pass and 1 represents Fail.

Table I
STRUCTURE OF INPUT DATA

n	t	x_1	x_2	...	x_S	y
1	1	$x_1(1,1)$	$x_2(1,1)$	...	$x_S(1,1)$	$y(1)$
1	2	$x_1(1,2)$	$x_2(1,2)$	...	$x_S(1,2)$	$y(1)$
...	...	...	...	...	...	...
1	T	$x_1(1,T)$	$x_2(1,T)$	...	$x_S(1,T)$	$y(1)$
2	1	$x_1(2,1)$	$x_2(2,1)$	...	$x_S(2,1)$	$y(2)$
2	2	$x_1(2,2)$	$x_2(2,2)$	...	$x_S(2,2)$	$y(2)$
...	...	...	...	...	...	...
2	T	$x_1(2,T)$	$x_2(2,T)$	...	$x_S(2,T)$	$y(2)$
...	...	...	...	...	...	...
N	T	$x_1(N,T)$	$x_2(N,T)$	...	$x_S(N,T)$	$y(N)$

B. Feature extraction

We extract Z features for each signal s and all MTS instances n resulting in Table II. The value of the extracted features are indicated by $f_{s,z}(n), s \in \{1,2,...,S\}, z \in \{1,2,...,Z\}$, representing the value of feature z extracted from signal s in MTS instance n. Table II will consist of N rows and $S \cdot Z + 1$ columns after the labels $y(n)$ are included.

Table II
EXTRACTED FEATURES

n	$f_{1,1}$	$f_{1,2}$	...	$f_{1,Z}$	$f_{2,1}$	...	$f_{2,Z}$	...	$f_{S,Z}$	y
1	...	...	...	...	...	...	...	...	...	...
2	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...
N	...	...	...	...	...	...	...	...	...	...

C. Feature reduction framework

After extracting features, we implement the proposed feature reduction framework. We present the high level working of the framework in Figure 3 and included a detailed description with pseudocode in Appendix A. The initial data consists of all features extracted from all signals and the corresponding labels that need to be predicted (Table II). To **prepare the data**, we ignore features with no variance since they contain no relevant information for the classification algorithm. Perfect co-linearity is also removed from the data to eliminate features that contain the same information. The data set is then split randomly into train (60%), test (20%) and validation (20%) data sets. In the course of this study we use a Random Forest classifier. We train a **baseline classification model** using all extracted features. The hyper-parameters of the baseline model are tuned based on classification of the test data set. After hyper-parameter tuning, the model is tested on the unseen validation data. We then apply the **permutation based feature reduction method** proposed by Schelthoff et al. [27] to subsequently eliminate irrelevant signals and features. Finally, we train a **final model** using the reduced feature set and tune the hyper-parameters of the model. The classification performance of the final model is then compared to the baseline model performance.

1) Baseline model: The baseline model uses all extracted features to perform classification. For hyper-parameter tuning,

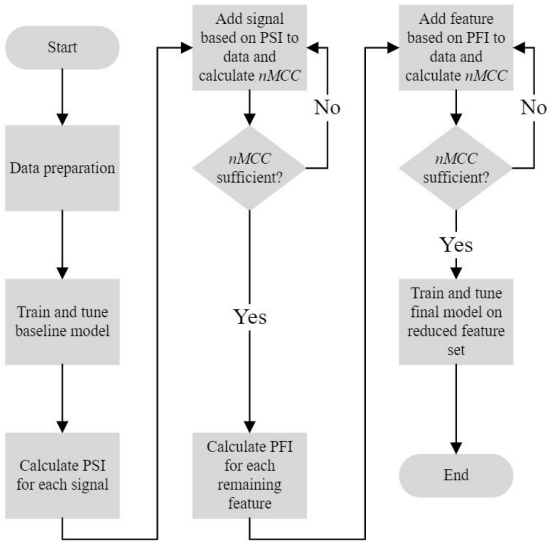

Figure 3. Feature reduction framework.

we predefined the hyper-parameters for the Random Forest classifier to limit the search space according to Table III. The hyper-parameters are defined as follows:

- **Criterion**: Determines how the model evaluates the information gained from each split.
- **Estimators**: Number of decision trees with random forest.
- **Max depth**: Maximum allowable depth of each decision tree.
- **Max features**: Maximum allowable features to consider when looking for the best split points. Here auto means the amount of features is equal to max depth, and sqrt means the square root of the total number of features is selected.
- **Min samples split**: The minimum number of samples to split an internal node.
- **Min samples leaf**: The minimum number of samples required to build a leaf.
- **Bootstrap**: Determines if bootstrap samples will be used for building trees.
- **Warm start**: Determines if the model can use the solution from the previous call when building the forest or a whole new forest is fitted.

Table III
HYPER-PARAMETER SPACE FOR THE RANDOM FOREST CLASSIFIERS

hyper-parameter	Value Range
criterion	Gini
estimators	200 - 2000
max depth	10 - 110
max features	[auto, sqrt]
min samples split	[2, 5, 10]
min samples leaf	[3, 4, 5, 6]
bootstrap	[True, False]
warm start	[True, False]

2) Baseline Model Evaluation: We select the Matthews Correlation Coefficient (MCC) to evaluate the performance

of the classification models. "The MCC is the only binary classification performance metric that generates a high score only if the binary predictor was able to predict the majority of positive data and negative instances correctly." [28], [29]. Therefore, it is a reliable indicator of classification performance for our use case.

The MCC value is calculated with Equation 1, where TP represents the number of True Positives, TN represents True Negatives, FP indicates the number of False Positives, and FN represents the False Negatives.

$$MCC = \frac{\mathrm{TP}\cdot\mathrm{TN}-\mathrm{FP}\cdot\mathrm{FN}}{\sqrt{(\mathrm{TP}+\mathrm{FP})\cdot(\mathrm{TP}+\mathrm{FN})\cdot(\mathrm{TN}+\mathrm{FP})\cdot(\mathrm{TN}+\mathrm{FN})}} \quad (1)$$

The outcome will be in the range of [-1,1], where -1 represents total disagreement between prediction and observation, 0 indicates the prediction model performs no better than random guessing and 1 means perfect predictions.

We normalize the MCC values to be in the range [0,1] using Equation 2.

$$nMCC = \frac{MCC + 1}{2} \quad (2)$$

The model performance is represented by the $nMCC$ value calculated from classification results on the unseen validation data. The $nMCC$ also serves as the optimization target for hyper-parameter tuning on the test set.

3) Permutation-based Feature Set Reduction: We reduce the signal set and feature set subsequently with the permutation-based feature set reduction method proposed by Schelthoff et al. [27].

To reduce the signal set, we calculate each signal's Permutation Signal Importance (PSI). See Section A8 for a detailed description. The PSI is defined as the relative performance change when randomly permuting a signal compared to the classification performance of the baseline model. PSI values above 1 indicate that the classification performance dropped after permuting a signal. When the PSI equals 1, the permutation does not affect the classification performance. PSI values lower than 1 indicate that permuting a signal increased the classification performance. To calculate the PSI, we first randomly shuffle one signal's values at a time in the validation data set. Second, using the baseline model on the shuffled validation data, we evaluate the average change in classification quality expressed in $nMCC$ to determine the contribution of the signal to the classification model. We repeat these two steps K times for each signal and calculate the average PSI to avoid coincidental signal importance due to randomness. The signals are then sorted based on their respective PSI.

When the signals are sorted, we iteratively add signals from most to least important to all the data sets (train, test and validation) and retrain the model with hyper-parameter tuning each iteration. For each iteration, we calculate the corresponding $nMCC$ by classifying the validation data set. The optimal signal set is then defined as the least amount of signals required to obtain a $nMCC$ value equal to or within a certain $Threshold$ of the $nMCC$ from the baseline model.

Table IV
SIGNAL CATEGORIES

Measurement frequency	Signal type	Signal number
Every Bond	Input signal	1-8
Every Wire	Input signal	9-15
Every Device	Input signal	16-21
Every Bond	Output signal	22-24
Every Wire	Output signal	24-26

After selecting the optimal signal set, we repeat the permutation-based feature set reduction on the individual features that are left since not all features corresponding to the optimal signal set will be relevant to the classification model. We follow the same steps for feature selection as for signal selection. However, as a sorter, we calculate each feature's Permutation Feature Importance (PFI). See Section A12 for a detailed description. The PFI represents the relative classification performance change when permuting one feature in the validation data compared to the baseline model performance.

What remains after signal and feature selection, is the least amount of (optimal) features the model needs to perform classification while keeping the classification performance within the set $Threshold$ value of the baseline model according to the $nMCC$ performance metric.

IV. Framework Application

We apply the method described in Section III to real-world wire bond machine signal data from NXP Semiconductors N.V. (see Figure 2). In this section, we describe the wire bond specific input data and elaborate on the result after applying the methodology. We also validate the results by cross-validating the features we found.

A. Data description

We explain the available data from the wire bonding by describing the measured signals and the extracted features from those signals. We also describe the available labels and summarize the input data.

1) Signals: All signals are described below and categorized according to Table IV. The categorization is based on measurement frequency and signal type. The measurement frequency indicates what signals are measured separately for each bond (ball and stitch bond), each wire or once per device. Signal type refers to signals measured during wire bonding (Input signals) and after wire bonding (Output signals). Some signal abbreviations contain the subscripts ball and stitch, since they are measured twice per wire. We consider these as separate signals. The following signals are considered:

1) DX_{ball}, DX_{stitch}: Die distance in the x direction from the reference system in millimeters.
2) DY_{ball}, DY_{stitch}: Die distance in the y direction from the reference system in millimetres.
3) USG_{ball}, USG_{stitch}: USG current generated by the ultrasonic generator during bonding in milliampere.
4) F_{ball}, F_{stitch}: Force on capillary in z-direction in Newton.

5) ZPC_{ball}, ZPC_{stitch}: Capillary distance in z direction from reference system in micrometres at moment of contact with the device.
6) ZPE_{ball}, ZPE_{stitch}: Capillary distance in z direction from reference system in micrometres at the end of the bonding sequence in micrometres.
7) RZC_{ball}, RZC_{stitch}: Capillary distance in z direction from reference system in micrometres at the moment of contact relative to value from the previous wire (removing heat block height tilt variation).
8) RZE_{ball}, RZE_{stitch}: Z position of the capillary at the end of bonding sequence relative to value from the previous index (removing heat block height tilt variation) in micrometres.
9) $INTF$: Time interval between the ball and stitch bond in milliseconds.
10) $INTS$: Time interval between the end of bonding the previous wire and the start of bonding the current wire in milliseconds.
11) DH: Difference in height between die and lead frame reference system in millimetres.
12) DT: Angle between the die and lead reference system in degrees.
13) BHD: Height difference between ball bond contact and stitch bond contact of the wire in millimetres.
14) $IMP1$: The ratio of sinusoidal voltage between the bonding impedance and the reference impedance.
15) $IMP2$: The ratio of sinusoidal voltage between bonding impedance and impedance in Air (PreBleed impedance).
16) DR_{ball}, DR_{stitch}: Die rotation compared to reference system in degrees.
17) $FSA1_{ball}$, $FSA1_{stitch}$: Average measured difference in z position (compared to reference system) between ball and stitch bond.
18) $FSS1_{ball}$, $FSS1_{stitch}$: Standard deviation of measured difference in z position (compared to reference system) between ball and stitch bond.
19) $FSA2_{ball}$, $FSA2_{stitch}$: Average measurement using Contact Seek Time. This measurement is used to monitor floating.
20) $FSS2_{ball}$, $FSS2_{stitch}$: Standard deviation of measurement using Contact Seek Time. This measurement is used to monitor floating.
21) V: Average Electronic Flame-Off Process (EFO) voltage per device.
22) $DEFC_{ball}$, $DEFC_{stitch}$: Deformation on contact in micrometers.
23) $DEFT_{ball}$, $DEFT_{stitch}$: Deformation on touchdown in micrometers.
24) $\varnothing$: Diameter of the ball in millimetres.
25) BPX: Location of the centre of the ball in x direction from the middle of the bond pad in millimetres.
26) BPY: Location of the centre of the ball in y direction, measured from the middle of the bond pad in millimetres.

Considering that some signals are measured for both ball and stitch bonds, we have 41 signals for each device.

2) Features: For every signal s, we extract the following (highly interpretable) features z resulting in feature $f_{s,z}$:

- Minimum ($f_{s,min}$)
- Absolute maximum ($f_{s,absmax}$)
- Maximum ($f_{s,max}$)
- Root mean square ($f_{s,rms}$)
- Variance ($f_{s,var}$)
- Standard deviation ($f_{s,std}$)
- Mean ($f_{s,mean}$)
- Median ($f_{s,med}$)
- Sum of values ($f_{s,sum}$)

In total, we extract 9 features from 41 signals resulting in 369 features that are used for classification.

3) Labels: As described in section I, a AOI labels each device as Pass or Fail after wire bonding. A human inspector assigns a certain failure mode when a device fails the AOI. For this study, we select the Tie bar deformation failure mode. Therefore $y(n) \in \{0, 1\}$, indicates the binary label corresponding to MTS instance n where 0 means the product Passed the AOI, and 1 means the product contained a Tie bar deformation error.

4) Input data: For this study, we select one product type produced on one type of machine. The data on wire bond failure modes is unbalanced, meaning that devices labeled as Pass far outnumber those labeled as Fail due to a Tie bar deformation. Highly imbalanced data, often makes machine-based processing difficult or even impossible [30]. Therefore, we select all devices containing a Tie bar deformation from the data and under-sample the devices that passed the AOI by randomly selecting N amount of devices that passed the AOI, such that we have 80% Passed and 20% Failed (due to Tie bar deformation) devices in the data. We do not include contextual data like machine_id, coordinate on leadframe and time of production. In prestudy, we found that this information is sensitive to so-called leakage, causing overfitting of the models. Leakage occurs when the data that is used for training the classification model contains information that the model is trying to predict. As a result, we study a total of 405 devices, of which 81 are labeled as a Tie bar deformation error, and the rest Passed the AOI. For each device instance n, we extracted 369 features corresponding to 41 signals. We add the labels $y(n)$. Therefore, the input data structure is a table consisting of 405 rows with 370 columns.

B. Results

The proposed methodology is applied to the labeled wire bond machine signal data. We run a number of experiments with different $Threshold$ values and signal types. Subsequently, we select the results from one experiment for further evaluation. After presenting the results, we cross-validate them using different subsets of the data for validation.

1) Experiments: In this study, we combined two methodologies, added the ability for signal selection and implemented a $Threshold$ for signal/feature selection. Therefore, we study the effect of applying the proposed method on different signal types according to Table IV, using different $Threshold \in \{0.95, 0.99, 1\}$ values. We expect that higher $Threshold$ values will result in more signals and features being selected in the optimal signal and feature sets because

Table V
EXPERIMENT RESULTS

Experiment #	$Threshold$	Signal type	$nMCC_{baseline}$	$nMCC_{final}$	Optimal # signals	Optimal # features
Experiment 1	0.95	All	1	0.99	1	2
Experiment 2	0.99	All	1	0.99	1	2
Experiment 3	1	All	1	1	2	3
Experiment 4	0.95	Input	1	0.99	1	2
Experiment 5	0.99	Input	1	0.99	1	2
Experiment 6	1	Input	1	1	2	3
Experiment 7	0.95	Output	0.98	0.95	1	1
Experiment 8	0.99	Output	0.98	0.98	1	2
Experiment 9	1	Output	0.98	0.98	1	2

the $Threshold$ determines the allowed drop in classification performance compared to the baseline model. For the signal types we expect (based on domain expert knowledge) that Output signals are better classifiers for Tie bar deformation failures than Input signals.

The experiments with the corresponding results are presented in Table V. The classification performance when using all features is indicated with $nMCC_{baseline}$. $nMCC_{final}$ represents the classification performance when using the optimal number (#) of signals with the optimal number (#) of features.

The experiment results show that the initial classification performance is high regardless of the selected signal type. However, the models using the Input signals generate a slightly better classification performance than the models using Output signals. The number of signals and features in all cases is reduced to two signals and a total of three features at maximum while keeping $nMCC_{final}$ higher than the set $Threshold$ value, which means that only a fraction of the available data is required to achieve sufficient classification performance. The influence of the $Threshold$ on the output is also clear. The algorithm automatically selects more signals and features when a higher classification performance is required.

2) In-depth analyses of results: We select Experiment 3 to present further analyses since it is the most comprehensive study, including all available signals and the selection $Threshold = 1$, meaning classification performance cannot be dropped compared to the performance of the baseline model performance.

The signal selection is visualized in Figure 4, where PSI represents the relative performance change when shuffling signal s compared to the baseline model. $nMCC$ represents the classification performance after keeping the top (based on PSI) number of signals represented on the x-axis. The figure depicts the first six signals (out of 41) that are added based on their respective PSI. The PSI values obtained from this experiment indicates that all but the top 3 most relevant signals did not impact the classification performance when randomly permuted since their respective PSI equals 1. The main conclusion from the signal selection is that we only need the two signals with the highest PSI score to obtain sufficient classification performance according to the set $Threshold$ value. Therefore, the optimal signal set is defined as the top 2 signals according to their respective PSI.

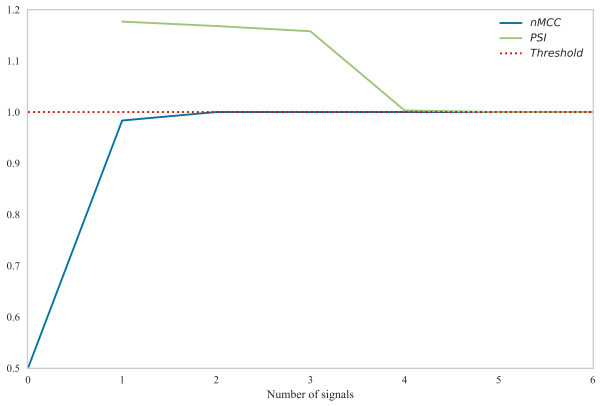

Figure 4. Top 6 important signals according to its corresponding PSI and $nMCC$ after adding the signal to the data.

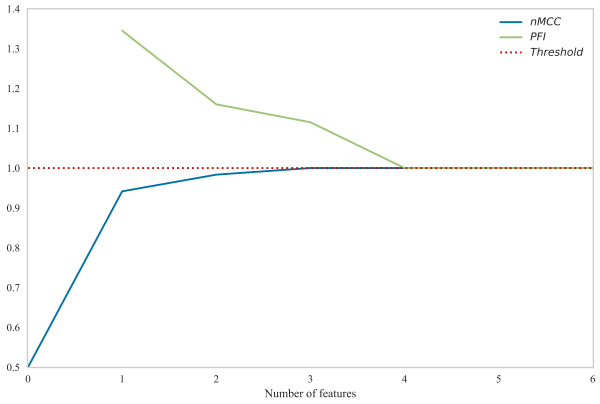

Figure 5. Features sorted on PFI with classification performance $nMCC$ after adding the feature to the data.

Similarly, Figure 5 visualizes the feature selection. We keep only features corresponding to optimal signals selected in the previous step. In this case, we only keep six features, the other features describing the top two signals got removed when removing features with no variance or features that have perfect co-linearity to other features. The figure shows that we only need the top three features, sorted on their respective PFI, to obtain sufficient classification performance. $nMCC$ represents the classification performance after keeping the amount of top features indicated on the x-axis.

For analyses we plot the optimal set of features in Figure 6. The features form clear clusters between devices that passed the AOI (blue) and the devices that failed the AOI due to Tie bar deformation (red). The obvious clusters explain why the classification performance is high using this data set. Since the extracted features are interpretable, these results can be analysed to explain why certain products Fail due to Tie bar deformation in the wire bond process.

To confirm that the features we found are relevant over the entire data set, we cross-validate the result by randomly selecting different subsets of the data as the validation data set. We repeat Experiment 3, while randomly selecting different subsets of the data for validation, 10 times. The selected

optimal signals and features remained the same over all 10 different validation sets, meaning that the signals and features we found generate strong classification results over the entire data set.

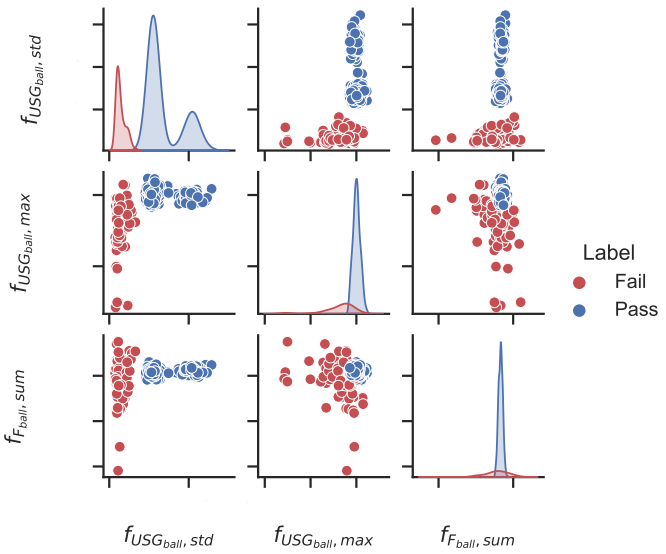

Figure 6. Scatter plot matrix of the optimal feature set.

V. Discussion

This study aims to predict wire bond quality per device by classifying labeled discrete MTS and focuses on interpretability by finding the least required signals and features to do so. In this section, we first (i) discuss the originality of our study. Second (ii), we discuss our decisions when designing the methodology. In the third (iii) part, we discuss the decisions made when applying the proposed methodology to real-world wire bond data. Finally (iv), we discuss the results, indicate the advantages of our proposed method and identify future research opportunities.

(Originality, i) To the best of our knowledge, we are the first to predict wire bond quality for devices rather than individual wires. Considering real-world wire bond data from NXP Semiconductors N.V., predicting bond quality for devices is relevant because feedback on the bond quality of individual wires is not widely available for all devices. Therefore, taking into account the structure of real-world wire bond machine signal data, we propose a novel methodology that combines the MTS classification method of Baldán et al. [26] with the feature reduction framework proposed by Schelthoff et al. [27]. The methodology is expanded on by including signal selection and implementing a selection $Threshold$.

(Methodology, ii) We decided to classify labeled discrete MTS using extracted features since features can accurately represent MTS and preserve most of the relevant information while eliminating the need for complex MTS classification models that are hard to interpret [26]. Due to the multivariate nature of the data in our use case and the number of features we can extract from the discrete MTS's, we decided to implement the feature reduction framework by Schelthoff et al. [27], allowing us to find the least required amount of features for classification. However, combining these two methodologies did not immediately generate desired results for our use case. When using a "large" amount of features, no effect on classification performance is detected when randomly permuting a single feature. This means that sorting features based on PFI was not effective. Therefore, we expand the methodology by first determining the optimal signal set and only then search for the optimal feature set. By first selecting the optimal signal set, the challenge of dealing with a "large" amount of features is reduced because the effect of permuting a signal is more evident on the classification performance than the effect of permuting individual features. The methodology is further expanded by implementing the selection $Threshold$, allowing the user to trade off desired classification performance for dimensionality reduction. Another decision that we made is to use a Random Forest classifier throughout this study. In a pre-study, we found that the Random Forest has the best classification performance in our use case. However, the methodology can be combined with any classification model meaning that for each use case the user can choose the best classifier. Furthermore, by selecting a specific failure mode to study, we applied the proposed method to a binary classification problem. In future research, it would be of great interest to apply the proposed method to multiclass (including multiple failure modes) classification problems as well.

(Application, iii) In our application framework, we selected highly interpretable features since we focus on the interpretability of the classification results. If in a use case interpretability is of minor relevance or the classification results are insufficient, one can try extracting other/more features. Any features can be extracted from the MTS adjusted to the need of each particular use case. In a pre-study, we extracted 19000 different features from the MTS for each device instance, using a python package called tsfresh [31], and we were still able to find the most relevant features using the proposed methodology.

(Results, iv) The results show that we can obtain excellent classification performance in the given wire bond use case and find the least required amount of signals and features. A clear advantage of the proposed methodology is that it can be applied to any labeled discrete MTS. Moreover, having a set with rich amount of Pass examples and a scarce amount of bond Failure examples, which is often the case in reality [23], does not have a major impact on the classification performance. When analysing the results of the wire bond use case one must take into account that the the machine signal data is not i.i.d. over all devices. The distributions of some signals change over time and can differ per machine. We randomly sample devices over the entire population, meaning that the signals and features we find are important over time and on different machines. However, in future research it would be of great interest to investigate the effect of using time-based splitting on the output of the proposed methodology. In this study, we scoped the framework application to only one product type produced on the same machine type from NXP Semiconductors N.V. The scope can be expanded to other product/machine types at different manufacturers. Also, any process generating labeled discrete MTS can be studied

using the proposed methodology. Furthermore, in this study, we fixed the ratio of Passed (80%) to Failed (20%) devices because these are common practice values. Other ratios might generate better results and insights. Again, future publications can take this into account.

VI. Managerial Implications

The proposed methodology allows for discrete MTS classification while finding the least required amount of signals and features.

The ability to classify discrete MTS can be implemented as a additional quality check during real-world manufacturing. Early detection helps to minimize the time and cost required to produce semiconductors; it can also helps to improve production yield [32]. The classification results can also be used to optimize the test sampling strategy. During semiconductor manufacturing, several tests are performed, some of which on a sample basis. By sample testing the devices that are labeled by the model as a potential failure, the testing equipment can be utilized more efficiently, meaning test cost reduction or lowering the probability of faulty devices being shipped to the customer.

The ability to find the relevant signals and features for classification is a tool to understand why certain failures occur. When it is known what patterns in the machine signal data lead to quality issues, failures can be prevented. For instance, by machine settings optimization and preventive maintenance, companies can prevent signal values drifting to patterns that are known for increased risk of failures. Another useful application is that irrelevant signals can be turned off, saving data storage and computational costs.

To take on the challenge of manufacturing semiconductors with increased performance standards, we encourage decision-makers to utilize the opportunities offered by the rise of IIoT. While challenging to work with, the newly obtained data can generate great value for the semiconductor industry.

VII. Conclusion

In this study, we developed a methodology for labeled discrete MTS classification with automated signal and feature selection. The methodology is applied to real-world wire bond machine signal data.

Three main conclusions can be drawn from this study. First, the proposed methodology allows for accurate wire bond quality classification per device. Second, we are able to find the least required amount of signals and features for classification, reducing the dimensionality of the input data coming from the wire bond process. Third, by extracting highly interpretable features from the MTS and finding the most relevant ones, we show that the classification results are interpretable and therefore useful for analyses.

However, in this study, we focus on one failure mode, product type, machine type and process at NXP Semiconductors N.V.

In future research, it would be of great interest to expand the study using different data sources. This will test the limits of the methodology by inputting labeled discrete MTS of varying lengths and amounts of signals and features. Another interesting expansion is to apply the proposed method for multiclass classification. The method can be used to find signals and features that allow for classifying multiple classes. Hence, we will focus our next studies on multiclass classification by classifying more than one failure mode at a time.

References

[1] H. K. Lim, Y. Kim, and M.-K. Kim, "Failure prediction using sequential pattern mining in the wire bonding process," *IEEE Transactions on Semiconductor Manufacturing*, vol. 30, no. 3, pp. 285–292, 2017.

[2] T. Choorat, "Back-end process training: Wire bonding," *NXP Semiconductors N.V.*, 2015.

[3] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu, "Feature selection: A data perspective," *ACM computing surveys (CSUR)*, vol. 50, no. 6, pp. 1–45, 2017.

[4] X. Ying, "An overview of overfitting and its solutions," *Journal of Physics: Conference Series*, vol. 1168, no. 2, p. 022022, feb 2019. [Online]. Available: https://dx.doi.org/10.1088/1742-6596/1168/2/022022

[5] R. Pufall, "Automatic process control of wire bonding." Publ by IEEE, 1993, pp. 159–162.

[6] Q. Wang, X. Sun, B. L. Golden, L. Desilets, E. A. wasil, S. Luco, and A. Peck, "A neural network model for the wire bonding process," *Computers and Operations Research*, vol. 20, 1993.

[7] S. W. Or, H. L. Chan, V. C. Lo, and C. W. Yuen, "Ultrasonic wire-bond quality monitoring using piezoelectric sensor," *Sensors and Actuators, A: Physical*, vol. 65, 1998.

[8] D. Zhang and S. F. Ling, "Monitoring wire bonding via time-frequency analysis of horn vibration," *IEEE Transactions on Electronics Packaging Manufacturing*, vol. 26, pp. 216–220, 7 2003, uses NN for wire bond prediction.

[9] S. Suman, M. Gaitan, Y. Joshi, and G. G. Harman, "Wire-bonding process monitoring using thermopile temperature sensor," *IEEE Transactions on Advanced Packaging*, vol. 28, 2005.

[10] Z. Zhong and K. Goh, "Investigation of ultrasonic vibrations of wire-bonding capillaries," *Microelectronics Journal*, vol. 37, no. 2, pp. 107–113, 2006.

[11] H. Gaul, M. Schneider-Ramelow, K.-d. Lang, and H. Reichl, "Predicting the shear strength of a wire bond using laser vibration measurements," in *2006 1st Electronic Systemintegration Technology Conference*, vol. 2, 2006, pp. 719–725.

[12] F. Wang, J. Li, S. Liu, and L. Han, "Heavy aluminum wire wedge bonding strength prediction using a transducer driven current signal and an artificial neural network," *IEEE Transactions on Semiconductor Manufacturing*, vol. 27, 2014.

[13] S.-F. Ling, D. Zhang, S. Yi, and S. W. Foo, "Real-time quality evaluation of wire bonding using input impedance," *IEEE Transactions on Electronics Packaging Manufacturing*, vol. 29, no. 4, pp. 280–284, 2006.

[14] W. Feng, Q. Meng, Y. Xie, and H. Fan, "Wire bonding quality monitoring via refining process of electrical signal from ultrasonic generator," *Mechanical systems and signal processing*, vol. 25, no. 3, pp. 884–900, 2011.

[15] F. Wuwei, M. Qingfeng, X. Youbo, and M. Qinghu, "Online quality evaluation of ultrasonic wire bonding using input electrical signal of piezoelectric transducer," vol. 5, 2009.

[16] E. Arjmand, P. Agyakwa, and M. Johnson, "Methodology for identifying wire bond process quality variation using ultrasonic current frequency spectrum acknowledgements," 2013.

[17] E. Arjmand, P. A. Agyakwa, and C. M. Johnson, "Reliability of thick al wire: A study of the effects of wire bonding parameters on thermal cycling degradation rate using non-destructive methods," vol. 54. Elsevier Ltd, 9 2014, pp. 2006–2012.

[18] E. Arjmand, P. A. Agyakwa, M. R. Corfield, J. Li, and C. M. Johnson, "Predicting lifetime of thick al wire bonds using signals obtained from ultrasonic generator," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 6, 2016.

[19] S. Hagenkötter, M. Brökelmann, and H. J. Hesse, "Piqc - a process integrated quality control for nondestructive evaluation of ultrasonic wire bonds," 2008, pp. 402–405.

[20] Y. F. Tzeng, F. C. Chen, and C. H. Chen, "Multiple quality characteristics optimization of ball grid array wire bonding process," *Journal of Electronic Packaging, Transactions of the ASME*, vol. 137, 12 2015.

[21] W. Feng, X. Chen, C. Wang, and Y. Shi, "Application research on the time–frequency analysis method in the quality detection of ultrasonic wire bonding," *International Journal of Distributed Sensor Networks*, vol. 17, 2021.
[22] M. Björklund, "Be careful with your principal components," *Evolution*, vol. 73, no. 10, pp. 2151–2158, 2019.
[23] N. Montealegre and S. Hagenkötter, "Process integrated wire-bond quality control by means of cytokine-formal immune networks," *Journal of Intelligent Manufacturing*, vol. 23, 2012.
[24] S. S. Lee, C. Shao, T. H. Kim, S. J. Hu, E. Kannatey-Asibu, W. W. Cai, J. P. Spicer, and J. A. Abell, "Characterization of ultrasonic metal welding by correlating online sensor signals with weld attributes," *Journal of Manufacturing Science and Engineering*, vol. 136, 2014.
[25] C. M. Yu, K. K. Lai, K. S. Chen, and T. C. Chang, "Process-quality evaluation for wire bonding with multiple gold wires," *IEEE Access*, vol. 8, 2020.
[26] F. J. Baldán and J. M. Benítez, "Multivariate times series classification through an interpretable representation," *Information Sciences*, vol. 569, pp. 596–614, 8 2021.
[27] K. Schelthoff, C. Jacobi, E. Schlosser, D. Plohmann, M. Janus, and K. Furmans, "Feature selection for waiting time predictions in semiconductor wafer fabs."
[28] G. Jurman, S. Riccadonna, and C. Furlanello, "A comparison of mcc and cen error measures in multi-class prediction," 2012.
[29] D. Chicco, "Ten quick tips for machine learning in computational biology," *BioData mining*, vol. 10, no. 1, pp. 1–17, 2017.
[30] M. Bach, A. Werner, and M. Palt, "The proposal of undersampling method for learning from imbalanced datasets," *Procedia Computer Science*, vol. 159, pp. 125–134, 2019, knowledge-Based and Intelligent Information Engineering Systems: Proceedings of the 23rd International Conference KES2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050919313456
[31] M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr, "Time series feature extraction on basis of scalable hypothesis tests (tsfresh – a python package)," *Neurocomputing*, vol. 307, pp. 72–77, 9 2018, python package used for feature extrction.
[32] B. E. Goodlin, D. S. Boning, H. H. Sawin, and B. M. Wise, "Simultaneous fault detection and classification for semiconductor manufacturing tools," *Journal of the Electrochemical Society*, vol. 150, no. 12, p. G778, 2003.
[33] M. Ali, "Pycaret: An open source, low-code machine learning library in python," *PyCaret version*, vol. 2, 2020.

APPENDIX

A. Algorithm for discrete MTS classification with automated signal and feature selection

In this appendix we provide a detailed description of the proposed methodology with pseudocode. The proposed methodology can be applied to any set of N labeled discrete MTS (number of devices) with length T (number of wires per device) and S amount of signals (active sensors in the machines).

Table II ($d_{initial}$) serves as input for algorithm 1. The algorithm classifies the labels $y(n)$ and finds the optimal signal and feature sets to do so while keeping the classification performance within a set threshold from the performance obtained when using all available data. We describe algorithm 1 in the following.

1) High level description: The following steps describe the high level working of algorithm 1:

- Lines [1-3]: **Prepare data** by removing features with no variance, removing perfect co-linearity and randomly splitting the data into train (60%), test (20%) and validation (20%) data sets.
- Lines (4,5): Perform **initial classification** using all remaining features.
- Lines [6-16]: **Determine the optimal signal set** by ranking signals based on their Permutation Signal Importance (PSI) to the classification performance (Lines [6-9]) and adding signals from most to least importance to the data until the classification performance is within a set $Threshold$ to the initial classification performance (Lines [11-15]).
- Lines (17,18): **Update data** to only include the optimal signals and labels and create a new classification model on the reduced data set.
- Lines [19-28]: **Determine the optimal feature set** by ranking features based on their Permutation Feature Importance (PFI) to the classification performance (Lines [19-22]) and adding features from most to least important to the data until the classification performance is within a set $Threshold$ to the initial classification performance (Lines [24-28]).
- Lines [29-32]: Perform **final classification** using the reduced feature set.

2) Performance metric: We select the Matthews Correlation Coefficient (MCC) to evaluate the performance of the classification models. "The MCC is the only binary classification performance metric that generates a high score only if the binary predictor was able to predict the majority of positive data instances and the majority of negative data instances correctly" [28], [29]. Therefore it is a reliable indicator of classification performance for our use case.

The MCC value is calculated with Equation 1. The outcome will be in the range of [-1,1], where -1 represents total disagreement between prediction and observation, 0 indicates the prediction model performs no better than random guessing and 1 means perfect predictions. In this study, we normalize the MCC values to be in the range [0,1] using Equation 2.

Algorithm 1: MTS classification with automated signal and feature selection

Input: $d_{initial}$, $Threshold$
Output: Optimal_signals, Optimal_features, $nMCC_{baseline}$, $nMCC_{final}$

1: Remove features $f_{s,z}$ with no variance from $d_{initial}$
2: Remove perfect co-linearity from $d_{initial}$
3: d_{train}, d_{test}, $d_{validation}$:= Randomly split $d_{initial}$ on rows
4: $m_{baseline}$:= Create classification model
5: Calculate $nMCC_{baseline}$
6: **for** signal s in {1,2,...,S} **do**
7: Calculate Permutation Signal Importance $PSI(s)$ with algorithm 2
8: **end for**
9: Sorted_signals := Sort signals $s \in \{1, 2, ..., S\}$ on corresponding $PSI(s)$ in decreasing order
10: $g := 1$
11: **while** $nMCC_{signal}(keep\_signals) < nMCC_{baseline} * Threshold$ **do**
12: $keep\_signals$:= [first g signals s in Sorted_signals]
13: Calculate $nMCC_{signal}(keep\_signals)$ with algorithm 3
14: $g := g + 1$
15: **end while**
16: Optimal_signals := $keep\_signals$ except for last signal in list
17: d_{train}, d_{test}, $d_{validation}$:= keep only features corresponding to signals in Optimal_signals and labels $y(n)$
18: m_{signal} := Create classification model
19: **for** all remaining features $f_{s,z}$ in $d_{validation}$ **do**
20: Calculate Permutation Feature Importance $PFI(f_{s,z})$ with algorithm 4
21: **end for**
22: Sorted_features := Sort features on corresponding $PFI(f_{s,z})$ in decreasing order
23: $g := 1$
24: **while** $nMCC_{feature}(features) < nMCC_{baseline} * Threshold$ **do**
25: $keep\_features$:= [first g features $f_{s,z}$ in Sorted_features]
26: Calculate $nMCC_{feature}(keep\_features)$ with algorithm 5
27: $g := g + 1$
28: **end while**
29: Optimal_features := $keep\_features$ except for last feature in list
30: d_{train}, d_{test}, $d_{validation}$:= keep only features in Optimal_features and labels $y(n)$
31: m_{final} := Create classification model
32: Calculate $nMCC_{final}$

3) Remove variance (Line 1): For each feature $f_{s,z}$ we calculate the variance $\sigma^2_{s,z}$ with Equation 3.

$$\sigma^2_{s,z} = \frac{\sum_{n=1}^{N}(f_{s,z}(n) - \mu_{s,z})^2}{N} \tag{3}$$

Where $\mu_{s,z} = \frac{\sum_{n=1}^{N} f_{s,z}(n)}{N}$ represents the mean of feature $f_{s,z}$. We remove all features $f_{s,z}$ from $d_{initial}$ when the corresponding $\sigma^2_{s,z} = 0$ since these features are constant over all MTS instances n, therefore containing no relevant information for the classification algorithm.

4) Remove perfect co-linearity (Line 2): Perfect co-linearity is removed from the data to eliminate features that describe the same information. When a feature $f_{s,z}$ has an exact linear relation with another feature $f_{s,z}$, we drop one of these features from $d_{initial}$.

5) Create d_{train}, d_{test}, $d_{validation}$ (Line 3): Split $d_{initial}$ randomly on rows so that 60% of rows are assigned to d_{train}, 20% to d_{test} and 20% to $d_{validation}$.

6) Create classification model (Lines 4, 18, 31): We train a classification model on d_{train} and tune the model's hyper-parameters by maximizing the $nMCC$ performance metric on predicting the labels for d_{test}. To reduce the effect of randomness on the hyper-parameter selection, we use StratifiedKFold Cross-Validation when tuning the hyper-parameters. For all machine learning steps in the algorithm we use the PyCaret package proposed by Ali et al. [33]. PyCaret is a Python wrapper around several machine learning libraries and frameworks, including scikit-learn. As proposed by Schelthoff et al. [27] a Random Forest classifiers is used in this study. We predefined hyper-parameters for the Random Forest classifier to limit the search space according to Table VI. The hyper-parameters are defined as follows:

- **Criterion**: Determines how the model evaluates the information gained from each split.
- **Estimators**: Number of decision trees in Random Forest.
- **Max depth**: Maximum allowable depth of each decision tree.
- **Max features**: Maximum allowable features to consider when looking for the best split points. Here "auto" means the amount of features is equal to max depth, and "sqrt" means the square root of the total number of features is selected.
- **Min samples split**: The minimum number of samples to split an internal node.
- **Min samples leaf**: The minimum number of samples required to build a leaf.
- **Bootstrap**: Determines if bootstrap samples will be used for building trees.
- **Warm start**: Determines if the model can use the solution from the previous call when building the forest or a whole new forest is fitted.

7) Calculate $nMCC$ (Lines 5, 32): To validate the performance of the classification model, we predict the labels on unseen data ($d_{validation}$), and calculate the $nMCC$ performance metric using Equation 1 and Equation 2.

8) Calculate Permutation Signal Importance $PSI(s)$ (Line 7): The $PSI(s)$ is computed using algorithm 2. To calculate the $PSI(s)$, we select all features $f_{s,z}$ corresponding to signal s and randomly permute the rows of the selected features in $d_{validation}$, leaving relations between the selected features intact. Without retraining, we then use the classification model trained on all features ($m_{baseline}$) to perform classification on the shuffled $d_{validation}(s,k)$ and calculate the classification performance $nMCC_{shuffle}(s,k)$. Previous steps are repeated $K = 100$ times, so we can take the average value of $nMCC_{shuffle}(s,k), k \in \{0,1,...,K\}$ to reduce the effect of randomness. The $PSI(s)$ is defined as the relative performance change, compared to $nMCC_{baseline}$, when randomly permuting signal s. High $PSI(s)$ scores indicate that signal s contributes more to the classification performance than signals with a lower $PSI(s)$ score.

Table VI
HYPER-PARAMETER SPACE FOR THE RANDOM FOREST CLASSIFIERS

hyper-parameter	Value Range
criterion	Gini
estimators	200 - 2000
max depth	10 - 110
max features	[auto, sqrt]
min samples split	[2, 5, 10]
min samples leaf	[3, 4, 5, 6]
bootstrap	[True, False]
warm start	[True, False]

Algorithm 2: $PSI(s)$ calculation

Input: signal s, $K := 100$, $m_{baseline}$, $nMCC_{baseline}$, $d_{validation}$
Output: $PSI(s)$

1: **for** k in range$(1, K)$ **do**
2: $d_{validation}(s,k)$:= Randomly permute all features belonging to signal s in $d_{validation}$
3: Use $m_{baseline}$ to classify labels $y(n)$ of $d_{validation}(s,k)$
4: Calculate corresponding $nMCC_{shuffle}(s,k)$
5: **end for**
6: $nMCC_{shuffle}(s) := \frac{\sum_{k=1}^{K} nMCC_{shuffle}(s,k)}{K}$
7: $PSI(s) := \frac{nMCC_{baseline}}{nMCC_{shuffle}(s)}$

9) Calculate $nMCC_{signal}(keep\_signals)$ (Lines 11-15): The $nMCC_{signals}(keep\_signals)$ is computed with algorithm 3 and represents the classification performance when only features belonging to signals s in list $keep\_signals$ are included in the data. We add signals from most to least important according to its $PSI(s)$ score to d_{train}, d_{test}, $d_{validation}$ until the classification performance is sufficient according to $nMCC_{signal}(keep\_signals) < nMCC_{baseline} * Threshold$.

10) Determine optimal signals (Line 16): The optimal set of signals is defined as the least amount of signals we need in order to perform classification while not dropping the classification performance $nMCC_{drop}(s)$ below $nMCC_{baseline} \cdot Threshold$.

11) Update signals in data-frames (Line 17): Since we know the optimal signal set, we keep only the features belonging the optimal signals in d_{train}, d_{test} and $d_{validation}$.

Algorithm 3: Calculate $nMCC_{signals}(keep\_signals)$

Input: $keep\_signals$, d_{train}, d_{test}, $d_{validation}$
Output: $nMCC_{signals}(keep\_signals)$
1: $d_{train}(keep\_signals)$, $d_{test}(keep\_signals)$, $d_{validation}(keep\_signals)$:= keep only features belonging to signals s in list $keep\_signals$, and labels $y(n)$ from d_{train}, d_{test}, $d_{validation}$
2: $m_{signal}(keep\_signals)$:= Train classification model on $d_{train}(keep\_signals)$, tune hyper-parameters on $d_{test}(keep\_signals)$
3: Calculate $nMCC_{signal}(keep\_signals)$ by classifying $d_{validation}(keep\_signals)$ with $m_{signal}(keep\_signals)$

12) Calculate Permutation Feature Importance $PFI(f_{s,z})$ (Line 20): The $PFI(f_{s,z})$ is computed using algorithm 4. To calculate the $PFI(f_{s,z})$, we randomly permute the rows of the selected feature in $d_{validation}$. Without retraining, we then use m_{signal} to perform classification on the shuffled $d_{validation}(f_{s,z}, k)$ and calculate the classification performance $nMCC_{shuffle}(f_{s,z}, k)$. Previous steps are repeated $K = 100$ times, so we can take the average value of $nMCC_{shuffle}(f_{s,z}, k), k \in \{0, 1, ..., K\}$ to reduce the effect of randomness. The $PFI(f_{s,z})$ is defined as the relative performance change, compared to $nMCC_{baseline}$, when randomly permuting feature $f_{s,z}$. High $PFI(f_{s,z})$ scores indicate that feature $f_{s,z}$ contributes more to the classification performance than features with a lower $PFI(f_{s,z})$ score.

Algorithm 4: $PFI(f_{s,z})$ calculation

Input: feature $f_{s,z}$, $K := 100$, m_{signal}, $nMCC_{baseline}$
Output: $PFI(f_{s,z})$
1: **for** k in range$(1, K)$ **do**
2: $d_{validation}(f_{s,z}, k)$:= Randomly permute feature $f_{s,z}$ in $d_{validation}$
3: Use m_{signal} to classify labels $y(n)$ of $d_{validation}(f_{s,z}, k)$
4: Calculate corresponding $nMCC_{shuffle}(f_{s,z}, k)$
5: **end for**
6: $nMCC_{shuffle}(f_{s,z}) = \frac{\sum_{k=1}^{K} nMCC_{shuffle}(f_{s,z},k)}{K}$
7: $PFI(f_{s,z}) = \frac{nMCC_{baseline}}{nMCC_{shuffle}(f_{s,z})}$

13) Calculate $nMCC_{features}(f_{s,z})$ (Lines [24-28]): The $nMCC_{features}(f_{s,z})$ represents the classification performance after adding features based on $PFI(f_{s,z})$ to d_{train}, d_{test}, $d_{validation}$. The $nMCC_{features}(f_{s,z})$ is computed using algorithm 5. We iterative add features from most to least important according to its $PFI(f_{s,z})$ score until until the condition $nMCC_{feature}(features) < nMCC_{baseline} * Threshold$ is met.

14) Determine optimal features (Line 29): The optimal feature set is defined as the least amount of features we need in order to perform classification while not dropping the classification performance $nMCC_{drop}(f_{s,z})$ below $nMCC_{baseline} \cdot threshold$.

Algorithm 5:
Calculate $nMCC_{features}(keep\_features)$

Input: keep_features, d_{train}, d_{test}, $d_{validation}$
Output: $nMCC_{features}(keep\_features)$
1: $d_{train}(keep\_features)$, $d_{test}(keep\_features)$, $d_{validation}(keep\_features)$:= Keep only features in list $keep\_features$, and labels $y(n)$ from d_{train}, d_{test}, $d_{validation}$
2: $m_{feature}(keep\_features)$:= Train classification model on $d_{train}(keep\_features)$, tune hyper-parameters on $d_{test}(keep\_features)$
3: Calculate $nMCC_{feature}(keep\_features)$ by classifying $d_{validation}(keep\_features)$ with $m_{feature}(keep\_features)$

15) Update features in data-frames (Line 30): Since we know the optimal feature set, we keep only these features in d_{train}, d_{test} and $d_{validation}$.