IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2024.0429000

Optimized Time Series Feature Selection for Manufacturing AI: Reducing Complexity and Improving Classification Accuracy

JAESEOK JANG[1], CHANYOUNG JUNG[1], and HAIL JUNG[2*], (Member, IEEE)

[1]Data Science Group, INTERX, Ulsan 44542, Republic of Korea; (e-mail: jay.jang@interxlab.com and cy.jung@interxlab.com)
[2]Department of Business Administration, Seoul National University of Science and Technology, Seoul 01811, Republic of Korea (e-mail: hail95@seoultech.ac.kr)

Corresponding author: Hail Jung (e-mail: hail95@seoultech.ac.kr).

Jaeseok Jang and Chanyoung Jung contributed equally to this work.

ABSTRACT

Although AI enhances productivity and quality in manufacturing, real-time validation remains difficult due to the complexity and high-dimensionality of time series data under strict inference time requirements. To address this, we propose two time series feature selection strategies. First, we extract meaningful variables by analyzing the time series patterns of non-defective and defective products. Through STL decomposition, we eliminate high-frequency noise and seasonal effects, and transform conventional feature-wise PCA into a production-wise approach by independently analyzing each sensor across products. We further quantify the pattern differences between non-defective and defective products for each sensor using DTW similarity, and perform clustering to systematically select significant variables. Second, we reconstruct lightweight statistical features through ANOVA and correlation analysis to further reduce feature dimensionality. In experiments on real-world manufacturing datasets, our techniques improve predictive performance and achieve reduced inference latency, successfully satisfying strict real-time production constraints.

INDEX TERMS Artificial intelligence, Computational Efficiency, Deep Learning, smart manufacturing, Machine Learning, time series classification, feature selection, Industrial AI

I. INTRODUCTION

THE manufacturing sector is increasingly driven by the need for automation to enhance productivity, reduce costs, and improve quality control, making the integration of artificial intelligence (AI) technologies essential. AI presents innovative solutions for key challenges, including reducing defect rates and waste in manufacturing processes, ultimately driving efficiency and sustainability [1]–[3]. Furthermore, studies suggest that AI integration can help resolve environmental issues, underscoring its importance not only as a tool for operational improvement but also as a critical enabler for the future of the industry [4], [5]. Within the Industry 4.0 framework, which emphasizes intelligent and interconnected manufacturing systems, AI plays a pivotal role in driving process innovation and transforming traditional manufacturing [6]–[8]. Developing these AI processes efficiently requires a cost-effective and sustainable development path [9]. However, successful implementation also requires highly accurate inference capabilities and reliable real-time performance to meet the demands of real-world industrial applications [10]. That is why research is continuously being conducted to increase performance while reducing AI inference time [6], [11] . This establishes AI as an indispensable foundation for advancing modern manufacturing practices while addressing both operational and strategic needs [12].

Time series data is one of the most critical data types in manufacturing, playing a crucial role in process optimization and achieving defect-free production [13]–[15]. In processes such as injection molding and material processing, continuous monitoring of variables such as temperature, pressure, and speed generates time series data that captures the dynamic characteristics of manufacturing operations, providing valuable insights for process improvement and quality control. However, since manufacturing data is generated in real time, the inference time should also be considered according to its characteristics when predicting product quality. The complexity of the time series model (requiring numerous learning parameters and processing 3D data) can significantly

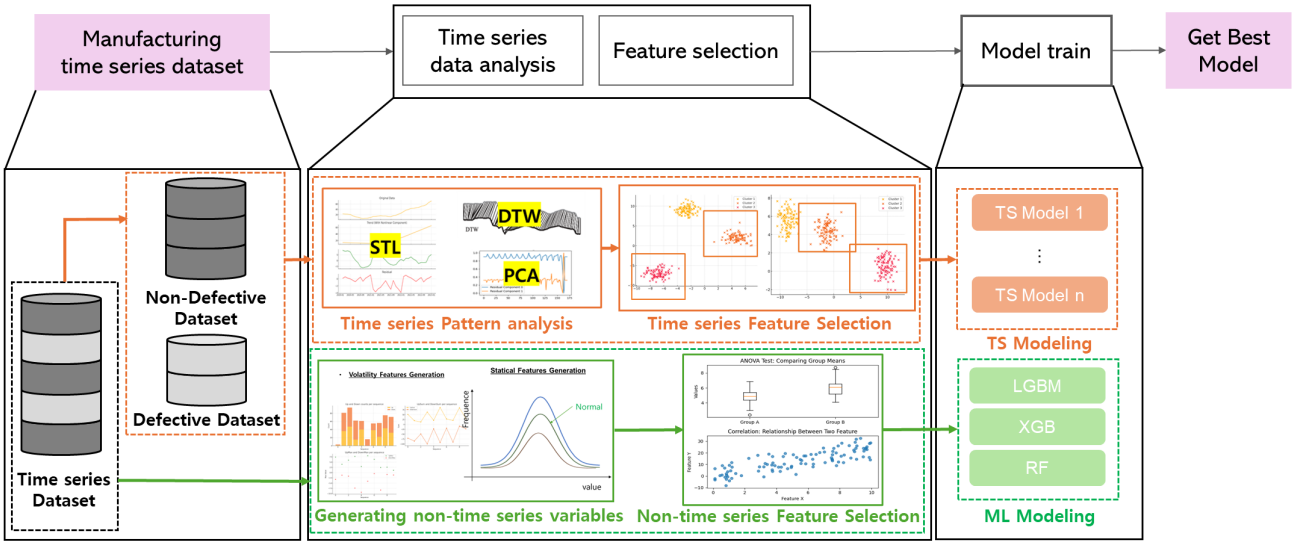


FIGURE 1. Overview analysis Pipeline of the proposed framework for feature selection and training model using time series and non-time series approaches. The first strategy utilizes Time Series Pattern Analysis for feature selection after classifying the dataset into non-defective and defective groups, leveraging time series characteristics for training model (highlighted in orange). The second strategy transforms time series data into non-time series features by extracting statistical properties and volatility, followed by non-time series feature selection and ensemble training model (highlighted in green). The final framework integrates both strategies to determine the best model.

increase the inference time, so reducing the inference time is essential for inferring products with short production cycles. Additionally, time series data is inherently complex, exhibiting temporal dependencies, multivariate interactions, variable sequence lengths, seasonality, trends, anomalies, and noise [16]. As a result, industries often rely on simplified statistical summaries to analyze time series data. However, simplified statistical summaries fail to fully leverage the rich temporal structure and interactions within the data, which can lead to performance degradation in predictive model [15]. Furthermore, extracting and utilizing meaningful variables for model training requires substantial domain expertise [17], but given the diversity of manufacturing processes, it is impractical for experts to possess deep knowledge across all domains, further complicating the model. Consequently, there is an increasing demand for automated AI solutions that can simplify and streamline the process of building AI systems without requiring extensive manual intervention or specialized expertise [18], [19].

Recent, end-to-end deep learning models offer the potential to learn complex representations directly from raw time series data, they often fail to adequately learn minority class patterns under severe class imbalance conditions [20], [21], which is particularly problematic in manufacturing defect detection where rare events are critical. Moreover, AutoML research efforts have attempted to automate feature engineering and selection to improve model performance, yet existing frameworks primarily focus on structured tabular, image, and text data, and do not fully support time series classification tasks due to sequence dependencies and variable-length inputs. To address the limitations of manual feature engineering, various time series feature extraction frameworks such as TSFresh [22] and Catch22 [23] have been proposed. These methods systematically extract interpretable statistical features from time series, enabling classical machine learning models to process sequential data. However, they are developed from a general feature engineering perspective and are not specifically tailored to reflect the unique requirements of manufacturing domains, distinguishing between defective and non-defective products based on subtle process variations.

To address these challenges, we propose two feature selection approaches that are tailored to the characteristics of manufacturing data and take into account inference time. Fig. 1 illustrates our proposed unified feature selection framework, which combines time series pattern analysis using STL, production-wise PCA, and DTW-based similarity, with statistical transformation strategies using ANOVA and correlation filtering. By identifying pattern-based discriminative features and removing redundant variables, this dual approach enables robust feature selection and enhances predictive accuracy and inference efficiency for real-time manufacturing applications.

The first approach involves directly extracting meaningful features from 3D-structured time series data through pattern analysis. Specifically, we utilize Seasonal-Trend Decomposition (STL) [24] to extract trends and apply Principal Component Analysis (PCA) [25] to identify key principal components that differentiate defective from non-defective data, enabling a comprehensive time series pattern analysis. To enhance this process, we transform feature-wise PCA into a production-wise approach by independently analyzing each sensor across products. Furthermore, to address scale differences across sensors in manufacturing processes, we propose a scale-invariant DTW similarity computation combined with

clustering for robust feature selection. Based on these pattern differences, we employ Dynamic Time Warping (DTW) [26] and clustering techniques [27] to identify the most critical variables. Although this methodology takes some time for inference because it trains a time series model, it not only facilitates the development of high-performance forecasting models, but also significantly reduces the computational time required for forecasting compared to existing time series models.

The second approach reconstructs time series data using statistical features while minimizing redundancy related to statistical and variability-based variables. Unlike simplified statistical summaries, this approach applies feature selection techniques (ANOVA [28] and correlation [29] analysis) to eliminate redundant features and retain only the most informative variables. By focusing on the most relevant statistical representations of the data, this method preserves crucial information while significantly reducing the dimensionality of the model. Once the key features are identified, training of the model is performed using only the selected features. This approach enables the development of lightweight models that significantly reduce prediction time. Its strength is particularly evident in manufacturing environments where data is generated at short intervals, making it highly suitable for processes with rapid production cycles. Consequently, this approach is especially advantageous for real-time applications, where minimizing inference time is essential.

These two proposed automated feature selection methods represent a significant advancement in enabling manufacturers to efficiently manage the complexity of time series data without relying on scarce domain experts. By addressing dimensional complexity, our approach supports real-time processing requirements in manufacturing environments while simultaneously improving predictive accuracy and computational efficiency. Ultimately, by providing a solution for selecting the most critical variables for model, our method lowers the barriers to AI adoption in the manufacturing industry and enhances both the accessibility and practical utility of AI in industrial settings. [18], [30].

Our main contribution is a robust and scalable feature selection framework specifically designed for manufacturing AI, with a focus on quality verification tasks. For time-series data, we propose a method that extracts stable and representative temporal features under varying production conditions using STL, production-wise PCA, and a scale-invariant DTW-based clustering approach. For transformed non-time-series data, we apply ANOVA and correlation analysis to select informative and non-redundant features. From a manufacturing perspective, our framework enhances the reliability and efficiency of defect detection and inference speed by stabilizing variable temporal inputs and eliminating irrelevant variables. We validate the effectiveness of our approach using two real-world datasets from distinct manufacturing domains, demonstrating its applicability to class-imbalanced and domain-sensitive quality verification tasks, and its potential for accelerating practical deployment in smart manufacturing environments.

In this paper progresses from foundational concepts to empirical validation, concluding with implications for future manufacturing systems. Section 2 reviews existing feature selection, AutoML techniques and time series feature engineering, outlining their limitations in manufacturing data contexts and introducing our proposed approach. Section 3 details the feature selection strategies employed to enhance defect detection performance. In Section 4, the proposed and baseline methods are evaluated using public and real-world datasets. The Discussion examines the impact of feature selection on model accuracy and computational efficiency, emphasizing its significance in real-time, AI-driven defect detection. The Conclusion underscores the necessity of adaptive AI automation in dynamic manufacturing environments and suggests future work on lightweight time series deep learning architectures to support efficient and sustainable production.

II. RELATED WORK

Time series and high-dimensional structured data play a critical role in manufacturing quality prediction. To address the complexity of building effective models, Automated Machine Learning (AutoML) frameworks and feature selection techniques have been introduced, aiming to automate model selection, hyperparameter tuning, and feature engineering [18], [31]. However, despite their success in well-structured and low-noise environments, the direct application of these methods to manufacturing data remains challenging due to high variability, domain-specific dependencies, and the need for real-time inference [32]. Traditional feature selection methods also struggle with issues such as multicollinearity [33], which further limits their effectiveness. This section reviews existing research on AutoML and feature selection for time series and manufacturing data, highlighting their limitations in addressing the specific requirements of industrial applications.

AutoML, or automated machine learning, simplifies the complex and resource-intensive process of building ML models by automating key steps such as model selection, hyperparameter optimization, preprocessing, and feature engineering. Platforms like AutoGluon, H2O.ai [34], and autoSklearn [35] support structured data, while AutoGluon TimeSeries [36] and AutoTS [37] specialize in time series data. These frameworks have made AutoML more accessible across domains. However, applying AutoML to manufacturing is challenging due to the need for extensive preprocessing of multivariate time series data, which most frameworks do not address. Darts and AutoTS excel in model stacking, they do not support product-level time series segmentation, which is essential in manufacturing to isolate meaningful sequences for each production cycle or serial number.

In addition, AutoGluon expand the number of features through feature engineering and combine multiple models using a stacking approach, which increases inference time. As a result, they may not be suitable for manufacturing processes that require real-time processing [38], [42]. Manufacturing

data is deeply intertwined with domain-specific contexts, including variability, complex process conditions, and strong temporal dependencies. In manufacturing, each product can have a slightly different production cycle, which causes sensor data to be temporally misaligned and creates irregular dependencies across time. However, Auto-Sklearn and H2O.ai assume fixed-length, uniformly aligned time series. Therefore cannot effectively model this temporal structure.

Another limitation is that time series classification, which is crucial in manufacturing (e.g., distinguishing defective from non-defective products), is not well-supported by popular AutoML frameworks like AutoGluon, Prophet [39], and AutoTS. This is primarily due to the difficulty of identifying patterns across entire sequences rather than forecasting single future values. Existing feature selection techniques are often insufficient due to varying product sequence lengths and the absence of predefined feature engineering mechanisms [40], [41]. As a result, statistical summaries are often used before the AutoML pipeline, but they do not properly represent the dynamic structure of the data, resulting in suboptimal performance. Although AutoML research is continuously advancing, time series customization for manufacturing is still in its infancy [38], [42].

In time series analysis, feature selection is a critical step to identify the most relevant variables for predictive models and improve model performance. Existing feature selection techniques, such as LASSO-based [43] methods, PCA and AutoEncoder-based [44] dimensionality reduction, have been widely used for general-purpose time series data. Manufacturing time series data is often high-dimensional and incorporates variables that reflect complex production conditions, such as temperature, pressure, and time intervals.These features often exhibit nonlinear dependencies, requiring more sophisticated techniques, such as deep learning-based feature selection or domain-specific algorithms. However, AutoEncoder-based [44] dimensionality reduction methods are unsupervised and prioritize reconstruction accuracy over task relevance. As a result, they often retain features that are easy to reconstruct rather than those that are predictive. In addition, manufacturing data is often collected at irregular intervals across multiple features, resulting in variable-length time series. This structure poses significant challenges for traditional techniques such as PCA and LASSO-based methods, which assume fixed-length, fully aligned input features and are not designed to handle asynchronous or incomplete sequences. Existing studies provide a foundation for feature selection, but they have proven effective only in specific cases and have not fully addressed the complexity of manufacturing-related requirements [45], [46].

Recently, researchers have explored feature engineering techniques to enhance the efficiency of time-series applications. Lubba et al [23]. proposed Catch22 to extract a small set of time series features with strong classification performance and minimal redundancy. Although Catch22 offers domain-independent features, its univariate design and lack of multivariate-aware variable selection limit its ability to capture complex inter-sensor dynamics essential in manufacturing systems. In contrast, In contrast, TSFresh [22] extracts a large number of interpretable time-series features and ranks them via significance tests, while also supporting variable selection that accounts for multivariate contexts. However, its high computational cost, the inclusion of noisy or redundant features, and the lack of mechanisms to address class imbalance make it unsuitable for large-scale or real-time manufacturing environments. As a result, This approaches fall short in capturing the absolute magnitudes, outliers, and complex temporal dynamics that are critical for robust manufacturing AI applications.

For non-time series data, feature selection plays an equally important role in improving model performance and reducing computational complexity. Traditional approaches fall into three main categories. Filter methods, such as Analysis of Variance (ANOVA) F-tests, correlation analysis, and Chi-square tests, evaluate features based on statistical metrics, making them computationally efficient and model-independent. However, they often fail to address multicollinearity issues [33]. Wrapper methods evaluate feature subsets based on model performance, yielding strong results but suffering from high computational costs and a tendency to overfitting in small datasets [47], [48]. Embedded methods, such as LASSO regression, perform feature selection during model training, offering efficiency but limiting generalizability due to their reliance on specific models [49]. More recently, automated feature selection techniques that leverage evolutionary algorithms have been developed to optimize feature selection along with model tuning. One notable tool is TPOT (Tree-based Pipeline Optimization Tool) [50], which uses genetic programming to automatically identify the best feature subset while also optimizing preprocessing and model training steps. However, although TPOT offers automation, it has limitations such as reduced interpretability and vulnerability to overfitting. [50]. Furthermore, it does not explicitly address multicollinearity, potentially allowing redundant or highly correlated features to be included in the final selection.

In summary, while AutoML and feature selection methods have made significant progress in simplifying machine learning model development, their application to manufacturing data remains challenging. Existing AutoML frameworks struggle with the complexities of manufacturing, particularly in time series classification and real-time processing. Similarly, traditional feature selection methods fail to fully accommodate the high dimensionality, varying data collection intervals, irregular sequence lengths, domain-specific dependencies, and overfitting issues inherent in manufacturing data. Addressing these challenges in the manufacturing domain requires domain knowledge, advanced preprocessing techniques, and tailored approaches optimized for industrial environments. In this paper, We propose a feature selection framework that combines product-level time series analysis using STL and production-wise PCA with scale-invariant DTW clustering to address sensor scale and sampling differences. For non-time-series, we apply ANOVA and correlation

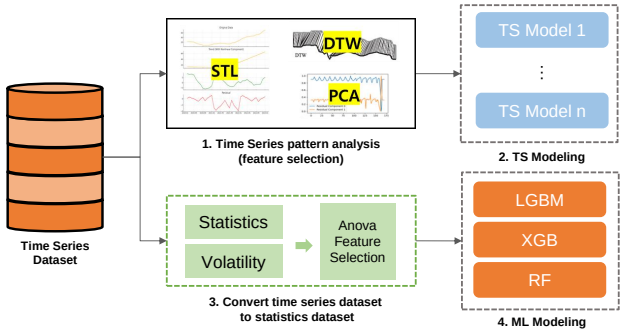


FIGURE 2. **Proposed pipeline structure for classification model. Our first strategy, time series feature selection, utilizes a GRU-based time series model. Meanwhile, our second strategy, non-time series feature selection, employs ensemble model.**

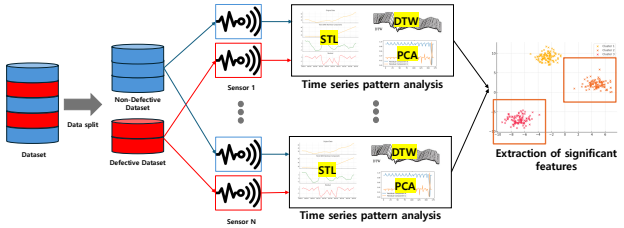


FIGURE 3. **Proposed time series feature selection pipeline using time series pattern analysis. The dataset is split into non-defective and defective groups, followed by STL, PCA, and DTW analysis on sensor data to extract significant features**

analysis to remove redundancy and retain relevant variables.

III. METHOD

Fig. 2 shows how our methodology is designed to fully leverage time series data by adopting two distinct approaches within a single dataset. First, the time series data undergo a preprocessing step to normalize the sequence lengths across different data instances. Following this, we apply our proposed feature selection method to identify and extract the most significant variables before training the model. Afterwards, we train the time series model using only the selected data. The second approach involves computing statistical summaries for each sequence within the time series data, effectively transforming the three dimensional time series dataset into a structured two-dimensional format. After performing a separate feature selection process, the newly structured dataset is used to train a machine learning model. Depending on the characteristics of the dataset, our methodology implements two types of feature selection so that meaningfully extracted features contribute to improving the performance of each model. Importantly, our work does not focus on the model development itself but rather on proposing feature selection techniques tailored for manufacturing data. Therefore, this section provides a detailed explanation of each proposed feature selection methodology, highlighting their implementation and advantages.

A. FEATURE SELECTION FOR TIME SERIES DATA

Fig. 3 represents the overview of time series feature selection strategy. First, our feature selection technique separates the non-defective and defective data sets from the original dataset. Then, we analyze the time series patterns that appear in the non-defective and defective datasets for each sensor. To achieve this, STL is utilized to extract the trend and residual components from each time series. Then, PCA is applied to derive the principal components for each sensor. Following this, DTW is employed to measure the similarity between non-defective and defective for each sensor. Finally, a clustering-based [27] approach is used to identify meaningful variable groups, leading to the extraction of the final set of significant features.

The pattern analysis method should specify that it analyzes the difference in the non-defective and defective data patterns for each sensor. At this step, zero-padding is applied to address the issue of varying lengths of time series data. This technique ensures that all time series are standardized to a uniform length, facilitating consistent feature extraction and comparison across different sensors. We use the STL technique to separate the seasonal, trend, and residual patterns in order to extract the time series patterns of the non-defective and defective data for one sensor. The STL decomposition process repeatedly extracts seasonal component (S_t) from the given time series data (X_t), removes it, extracts the trend component (T_t) through Loess [51] smoothing, and finally calculates the residual component (R_t), as shown in (1),

$$X_t = S_t + T_t + R_t, \quad \begin{cases} S_t & \text{the seasonal component} \\ T_t & \text{the trend component} \\ R_t & \text{the residual component.} \end{cases} \tag{1}$$

Here, S_t captures the periodic (seasonal) component of the time series, where the periodicity is either predefined or estimated during the STL decomposition. Loess is a nonlinear regression technique that locally smooths data to extract trends or approximate curves. Loess selects data from a specific range at each data point and weights it to perform linear or polynomial regression. The seasonal component (S_t) is computed as a weighted average of periodic data points across cycles. This process can be expressed mathematically as:

$$S_t = \frac{1}{p}\sum_{i=1}^{p} w_i \cdot x_{t-i}, \quad \begin{cases} p: \text{ cycle length} \\ w_i: \text{ weight of the } i\text{-th observation in the cycle.} \\ x_{t-i}: \text{ observed value } (i) \text{ periods before } (t). \end{cases} \tag{2}$$

The weights (w_i) ensure that certain points, such as those closer to the current cycle, contribute more to the calculation of (S_t). This allows (S_t) to reflect periodic patterns accurately while reducing the influence of noise. Once (S_t) is estimated,

the detrended data $(X_t - S_t)$ is used to compute the trend component (T_t). The trend is extracted using Loess smoothing, which incorporates the Tri-cube weight (w_{ti}) to estimate the local regression coefficients. The weight function is defined as,

$$w_{ti} = \begin{cases} \left(1 - \left|\frac{x_i - x_t}{h}\right|^3\right)^3, & \text{if } |x_i - x_t| \le h, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Using these weights, the trend (T_t) is calculated through locally weighted regression,

$$\begin{aligned} T_t &= \beta_0 + \beta_1(x_t - x), \\ \min_{\beta_0,\beta_1} \sum_{i=1}^{n} w_{ti} \left(X_{t_i} - \beta_0 - \beta_1(x_i - x_t)\right)^2, \end{aligned} \tag{4}$$

where β_0 and β_1 are regression coefficients obtained by minimizing the weighted least squares error. Equation (4), X_{t_i} represents the observed value at time (t_i), where t_i refers to the i-th time point in the local neighborhood of t. The local regression uses these X_{t_i} values, weighted by their distance to the central point X_t, to estimate the trend component (T_t). The extracted T_t captures the smooth, long-term trajectory of the time series, isolating it from short-term fluctuations and periodic patterns.

The extraction of S_t and T_t is inherently interconnected. After estimating S_t using (2), the detrended data $(X_t - S_t)$ serves as input for computing T_t using (4). Conversely, the updated T_t is used to refine S_t, ensuring that the seasonal and trend components are optimized iteratively. This iterative process continues until convergence, resulting in a robust decomposition where S_t and T_t are effectively separated while maintaining consistency with the original data X_t.

Based on specific sensor data, we extracted the trend and residual patterns of the data by utilizing STL to analyze the patterns of good and defective products. However, in the actual manufacturing process, the good data accounts for a significantly larger proportion than the defective data, which makes it difficult to directly compare the two data.

To solve this problem, secondly, we propose a method to analyze the trend and residual patterns of the non-defective and defective data extracted by STL using PCA and extract one principal component each of the trend and residual representing the non-defective and defective products, respectively. In conventional time series analysis, feature-wise PCA is typically applied across different variables to capture inter-variable correlations. However, in our study, we propose a novel application of production-wise PCA across multiple products for a single sensor variable to extract a representative trend that characterizes good and defective products separately.

Formally, let $T_i^j(t)$ denote the trend component extracted via STL decomposition for the i-th product sensor data. For a fixed j-th sensor s_j, we collect the trends across n products into a matrix:

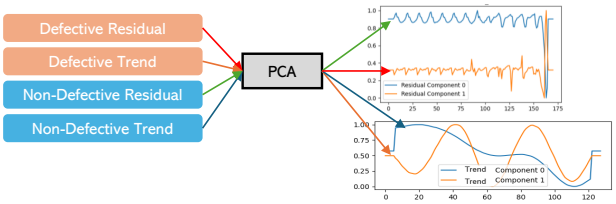


FIGURE 4. Extracting principal components of residual and trend components from non-defective and defective patterns using PCA to analyze feature differences across sensors.

$$X^{(s_j)} = \begin{bmatrix} T_1^j(t) \\ T_2^j(t) \\ \vdots \\ T_n^j(t) \end{bmatrix} \in \mathbb{R}^{n \times d} \tag{5}$$

where d is the number of time points in the trend. We then compute the covariance matrix:

$$C^{(s_j)} = \frac{1}{n}(X^{(s_j)})^T X^{(s_j)} \tag{6}$$

and obtain the first principal component $v_1^{(s_j)}$ by solving:

$$v_1^{(s_j)} = \arg\max_{\|v\|=1} \sum_{i=1}^{n} (T_i^j(t) \cdot v)^2 \tag{7}$$

This production-wise principal component $v_1^{(s_j)}$ represents the dominant trend pattern across the products for the given sensor. This is equivalent to finding the leading eigenvector of the covariance matrix $C^{(s_j)}$, corresponding to the direction of maximum variance in the production-wise trend space. The same procedure is applied to the residual components $R_i^j(t)$ to compute the dominant residual pattern $v_{\text{res}}^{(s_j)}$, which captures the common high-frequency variation across products for sensor s_j.

Fig. 4 shows the production-wise PCA flowchart of feature selection based on pattern analysis of the non-defective and defective process data. By first applying STL decomposition, we eliminate high-frequency noise and seasonal effects, ensuring that production-wise PCA focuses on meaningful long-term behaviors rather than transient fluctuations. Simple pointwise averaging is highly sensitive to outliers, whereas production-wise PCA aggregates structural variance across samples, making the representative trend more resilient to anomalous products. By independently summarizing the behavior of each sensor across multiple products in a production-wise manner, the extracted representative trend has a clear physical meaning and captures the most common long-term behavior patterns typical of non-defective or defective products for that sensor.

Third, we calculate the similarity through DTW to compare whether the sensor is a sensor that shows a significant difference between non-defective and defective products through

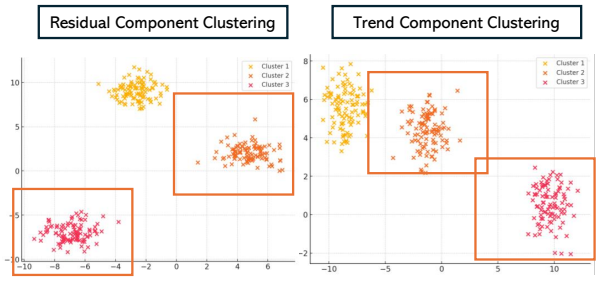


FIGURE 5. Extracting significant features using clustering from principal components and DTW similarity of residual and trend components in non-defective and defective patterns.

the extracted non-defective/defective trend and residual principal components. Previously, zero-padding was applied to handle variable-length time series, automatically aligning all sequences to the length of the longest product sequence. However, this approach can introduce noise and distort the time series pattern analysis. To mitigate this issue, we employ the DTW. DTW is an algorithm that calculates the similarity by considering the non-linear time alignment between two time series data. DTW works effectively even on data with different lengths or data that are not synchronized on the time axis, and is mainly used to compare the similarity of patterns or to align time series data. That is, DTW inherently accounts for inconsistent sequence lengths and irregular time intervals, making it well-suited for handling time series data without being influenced by the meaningless padded regions introduced by zero-padding [52], [53].

DTW calculates the optimal path between two time series data using the following formula and measures the similarity based on these paths. DTW is a method for measuring similarity between two time series $A = \{a_1, a_2, \ldots, a_n\}$ and $B = \{b_1, b_2, \ldots, b_m\}$, by allowing non-linear alignments. The cost matrix $D(i,j)$ is recursively defined as,

$$D(i,j) = \|a_i - b_j\| + \min \begin{cases} D(i-1,j), \\ D(i,j-1), \\ D(i-1,j-1), \end{cases} \tag{8}$$

where $\|a_i - b_j\|$ is the distance between the elements a_i and b_j. The DTW distance between the two sequences is given by

$$\mathrm{DTW}(A,B) = D(n,m) \tag{9}$$

where $D(n,m)$ is the accumulated cost at the last cell of the cost matrix.

The DTW algorithm ensures that the two time series are optimally aligned by calculating the minimum cumulative cost along the warping path, with the final value $D(n,m)$ representing the similarity between the sequences.

We apply this to all sensors and extract the similarity of the non-defective/defective data set pairs of all sensors. However, although DTW measures similarity, it represents the relative distance between two time series (defective, non-defective), so it is not an absolute indicator that can be compared with other features. To solve this, we apply clustering to divide it into three similarity groups, and then define the features of Class 2 and 3, excluding Class 1 with the highest similarity, as significant features. We proceed to the DTW similarity of Trend and Residual, and extract the Residual Cluster (Class 2, 3) and Trend Cluster (Class 2, 3) as the final significant features. The following Fig. 5 shows the clustered part of the values of Residual and Trend.

While DTW and clustering are individually well-established techniques, our approach is not a naive combination of the two but a novel formulation that transforms the problem of sensor selection into a structured optimization on relational distance space. Traditional DTW focuses on aligning time series with non-linear temporal shifts and calculating a minimum cumulative distance. However, the resulting DTW distance values vary in scale across different sensors, making direct comparison between sensors problematic and potentially introducing selection bias. Conventional clustering methods, on the other hand, operate on raw feature vectors and assume linear Euclidean structures, thus failing to capture the inherent time-warped similarities of sequence data.

To address these issues, we propose a structured method that first normalizes the principal component trends and residuals for each sensor individually to the range $[0,1]$, ensuring scale-invariant DTW similarity computation.

$$d_s^{\text{scaled}} = D_{\text{DTW}}\left(\left(PC'^{(s_j)}_{\text{defective}}(t)\right), \left(PC'^{(s_j)}_{\text{non-defective}}(t)\right)\right) \tag{10}$$

where PC′ denotes the min-max normalized version of the original principal component $PC(t)$, scaled to the range $[0,1]$.

The set of all DTW distances, $\mathcal{D} = \{d_1, d_2, \ldots, d_S\}$, represents a relational similarity structure across sensors. Rather than applying heuristic thresholding, we partition $\mathcal{D}$ into clusters by solving:

$$\min_{\mathcal{C}} \sum_{k=1}^{3} \sum_{d_s^{\text{scaled}} \in \mathcal{C}_k} \left\| d_s^{\text{scaled}} - \mu_k \right\|^2 \tag{11}$$

where μ_k denote the centroids of each cluster, and $K = 3$ is predetermined to divide the sensors into high, moderate, and low similarity groups. Finally, we define significant sensors as those belonging to clusters with relatively larger DTW distances (moderate and low similarity groups):

$$\mathcal{S}_{\text{significant}} = \{s \mid d_s \in \mathcal{C}_2 \cup \mathcal{C}_3\} \tag{12}$$

Through per-sensor normalization prior to DTW computation and clustering over relational distances, our method enables robust, scale-invariant, and structure-aware feature selection tailored to time series data.

Lastly, the feature selection based on pattern analysis of the data from the non-defective and defective products process that we propose compares the principal components of the

Algorithm 1 Feature Selection for Time Series dataset

Require: Time series $\{X_i^j(t)\}$, where i is the product index and j is the sensor index

Ensure: Significant sensor set $\mathcal{S}_{\text{significant}}$

for each sensor j **do**
  for each product i **do**
    $\{T_i^j(t), S_i^j(t), R_i^j(t)\} \leftarrow \text{STL}(X_i^j(t))$
    Normalize $T_i^j(t)$ and $R_i^j(t)$ to $[0,1]$
  end for
  $X_{\text{trend}}^{(j)} = \{T_i^j(t)\}_{i=1}^n$
  $X_{\text{res}}^{(j)} = \{R_i^j(t)\}_{i=1}^n$
  Compute $v_{\text{trend}}^{(j)}$, $v_{\text{res}}^{(j)}$: production-wise PCA $X_{\text{trend}}^{(j)}$, $X_{\text{res}}^{(j)}$
  Compute DTW distances:

$$d_{\text{trend}}^{(j)} = D_{\text{DTW}}(v_{\text{trend,non-defective}}^{(j)}(t), v_{\text{trend,defective}}^{(j)}(t))$$

$$d_{\text{res}}^{(j)} = D_{\text{DTW}}(v_{\text{res,non-defective}}^{(j)}(t), v_{\text{res,defective}}^{(j)}(t))$$

end for
$\mathcal{D}_{\text{trend}} = \{d_{\text{trend}}^{(j)}\}, \quad \mathcal{D}_{\text{res}} = \{d_{\text{res}}^{(j)}\}$
Cluster $\mathcal{D}_{\text{trend}}$ and $\mathcal{D}_{\text{res}}$ into $K = 3$ groups
Define:

$$\mathcal{S}_{\text{significant}} = \{j \mid d_{\text{trend}}^{(j)} \in \mathcal{C}_2 \cup \mathcal{C}_3 \quad \text{or} \quad d_{\text{res}}^{(j)} \in \mathcal{C}_2 \cup \mathcal{C}_3\}$$

return $\mathcal{S}_{\text{significant}}$

time series patterns, Trend and Residual patterns, in the non-defective and defective products to extract the final significant features with low similarity, and uses only the data with low similarity for the time series model training.

Algorithm 1 summarizes the overall feature selection procedure for the time series dataset. For each sensor, STL decomposition is first applied to extract trend and residual components, which are then normalized to ensure scale consistency. Production-wise PCA is subsequently performed on each component to extract the first principal component, representing the dominant pattern across products. DTW distances between non-defective and defective representative components are computed, and clustering is applied to systematically select sensors that exhibit significant pattern differences. This approach enables robust and efficient feature selection tailored to the characteristics of real-world manufacturing time series data.

B. FEATURE SELECTION FOR NON-TIME SERIES DATA

Three dimension time series data comprises temporal, variable, and sample dimensions, making it complex for machine learning applications [15]. To effectively utilize this data in standard models, it is often necessary to transform it into a two-dimension format by extracting statistical attributes from the temporal dimension [54]. Fig. 6 illustrates the process of transforming our time series data into non-time series data. Typically, Time series data is structured as (N, T, F),

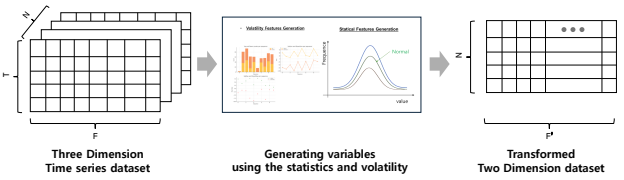


FIGURE 6. Variables are extracted using statistical and volatility attributes to transform time series data into non-time series data.

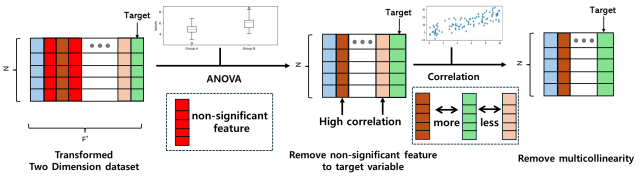


FIGURE 7. Proposed pipeline for feature selection in non-time series data. First, Using an ANOVA F-test, we eliminate variables that are not significantly associated with the target variable. Second, We address multicollinearity by performing correlation analysis.

where N is the number of samples, T is the sequence length, and F is the number of features. Our transformation process involves computing these statistical attributes across T for each feature. Additionally, we create volatility-related variables to reflect the inherent fluctuations driven by temporal patterns. This transformation result in a structured (N, F') dataset where F' represents the newly derived features. This transformed data format facilitates the application of various machine learning models without the need for specialized time series approaches, thereby improving computational efficiency and model interpretability [55].

After generating variables using the statistics and volatility of each feature in the time series dataset, we apply feature selection techniques to ensure optimal model performance. Fig. 7 represents the feature selection flowchart of the non-time series data. Our proposed non-time series feature selection approach employs a two step process to identify significant variables. Specifically, we applied ANOVA [28] to evaluate the statistical significance of each feature with respect to the target variable. ANOVA analysis is conducted to statistically determine which variables show significant differences in their means based on the target categories and to extract meaningful variables accordingly. For every independent feature, we computed the F-statistic and its corresponding p-value. Features with a p-value less than the predefined significance level of 0.05 were considered to have a meaningful impact on the target variable. These statistically significant features were retained for further analysis.

After identifying features that show statistically significant associations with the target variable, we analyzed their relationships to mitigate potential multicollinearity issues. Using the Pearson correlation coefficient, we calculated the pairwise correlation between all selected features. If two features were found to have a high correlation (In this methodology, it is defined as a Pearson correlation coefficient of 0.8 or higher),

Algorithm 2 Feature Selection for Non-Time Series Data

Require: Time series $\{X_i^j(t)\}$
Ensure: Selected feature set $\mathcal{F}_{\text{selected}}$
for each sample i, feature j **do**
  Extract statistical and volatility features across t
end for
Form transformed dataset $\{X_i^{j'}\}$
$\mathcal{F}_{\text{anova}} \leftarrow \{j' \mid \text{ANOVA}(X^{j'}) < 0.05\}$
$\mathcal{F}_{\text{selected}} \leftarrow \mathcal{F}_{\text{anova}}$
for each $(j'_a, j'_b) \in \mathcal{F}_{\text{selected}}$ **do**
  if $\rho(j'_a, j'_b) \geq 0.8$ **then**
    Retain $\arg\max_{j' \in \{j'_a, j'_b\}} \rho(j', \text{target})$
  end if
end for
return $\mathcal{F}_{\text{selected}}$

we investigated their individual relationships with the target variable. Specifically, we calculated the correlation of each feature with the target variable and retained the feature with the stronger correlation to the target. This step ensured that the final set of selected features was not only statistically significant, but also sufficiently independent of the relationships among the features to avoid multicollinearity.

Algorithm 2 describes the procedure for transforming time series data into non-time series features and selecting significant variables. Here, i denotes the sample index, j denotes the original feature index, and j' denotes the transformed feature index after statistical and volatility feature extraction. The process involves extracting statistical attributes across the temporal dimension, applying ANOVA to filter features based on their association with the target, and removing highly correlated features to mitigate multicollinearity.

We train a machine learning model using a refined set of robust 2D features, obtained through a two step process of feature selection via ANOVA and multicollinearity mitigation using correlation analysis with the target variable.

IV. EXPERIMENTAL

A. EXPERIMENTAL SET UP

1) Baseline

For performance comparison shown in Fig. 1, we evaluate AutoGluon, PyCaret, TPOT, TSFresh, and Catch22. AutoGluon provides both Tabular and TS modules, but AutoGluon.TS does not support time series classification. Therefore, we exclude Autogluon.TS from the experiments. PyCaret [58] and TPOT are used for classification on non-time series data. TSFresh and Catch22 are used to extract time series features, which are then passed to conventional classifiers. All methods include automated processes for feature selection or extraction and model optimization. Their performance is compared with our time series pattern analysis strategy.

Bidirectional GRU architecture [59], [60], as shown in Fig. 2. These models use concatenation-based feature fusion to capture both short and long-term temporal dependencies, with variations in hidden size and dropout settings. For comparison, we also evaluate state-of-the-art models including TimesNet [56] and SegRNN [57].

2) Dataset

We utilized three datasets in our experiments. One of these is an open-source dataset from Kaggle. It consists of thousands of sixty-second sequences of biological sensor data recorded from several hundred participants in one of two possible activity states. The dataset includes approximately 26,000 records with a total of 13 sensor features. The dataset has a binary class distribution of approximately 55:45, with no missing values. Although this dataset originates from a biological domain, it has been selected due to its structured multivariate time series format and its relevance in testing classification and prediction methodologies, which are applicable to autonomous manufacturing scenarios.

As a second company F dataset originates from an automotive parts manufacturing process, with sensor readings recorded every 2 seconds. Each sample represents a minute window, comprising 30 time steps. Collected over a six-month period, the dataset includes approximately 100,000 product records with 196 sensor features. Among them, about 1,800 are labeled as defective, resulting in a defect rate of 1.8%. The dataset is complete, containing no missing values, and provides a realistic representation of the operational environment in modern smart factories.

The last company D dataset originates from an arginine fermentation process. The Company D dataset was collected from an arginine fermentation process, with sensor readings recorded every 10 seconds. It includes variables such as tank temperature, air supply, sugar feed rate, and cell state indicators, with no missing values. Each sample covers a 20-minute window, resulting in 120 time steps. The objective is to classify whether a proper cellular response occurred, based on labeled data. Out of approximately 14,000 samples, around 3,100 are labeled abnormal, indicating a defect rate of 22.1%.

3) Performance Metrics

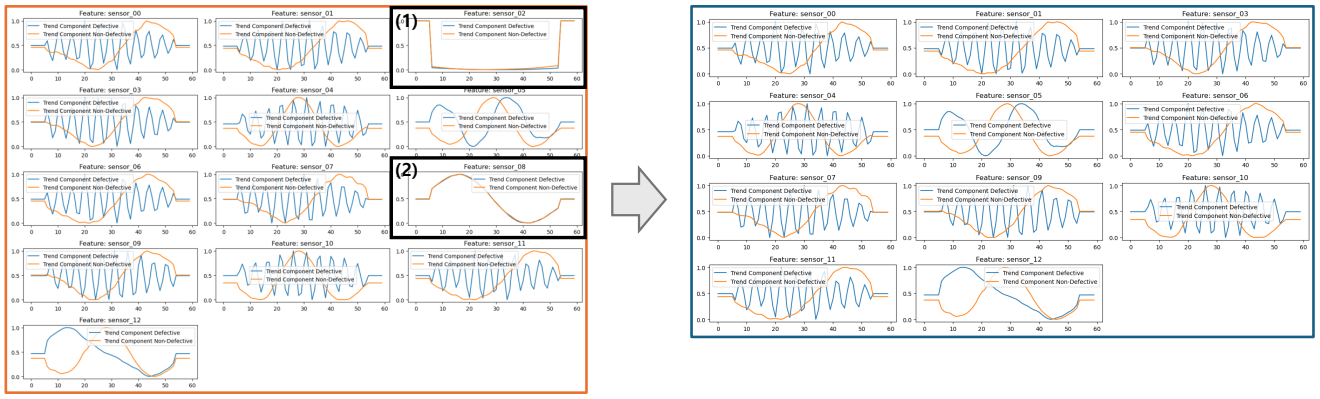
To evaluate model performance in manufacturing scenarios, we use accuracy, recall, F1-score, and inference time. As defective items are typically rare, accuracy alone may be misleading. Thus, recall and F1-score are emphasized for assessing how well models detect defects while balancing false positives and false negatives.

Accuracy reflects the overall proportion of correct predictions:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{13}$$

Recall measures how well the model identifies actual defective items:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{14}$$



(a) ALL Feature Trend PCA Result

(b) Trend component Feature selection Result

FIGURE 8. Feature selection results using time series pattern analysis on the Kaggle dataset. (a) Results of trend component patterns extracted using STL and PCA applied to all features, where (1) and (2) represent non-significant features. (b) Results after selecting significant features using DTW and clustering outcomes.

F1-score, the harmonic mean of precision and recall, is especially useful in imbalanced datasets:

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{15}$$

Here, TP (true positive), TN (true negative), FP (false positive), and FN (false negative) refer to standard classification outcomes. Lastly, we also measure inference time over the full test set to evaluate real-time suitability in industrial applications.

B. EXPERIMENTAL 1

In this section, we conduct experiments on an open-source dataset from Kaggle. The Kaggle dataset has a binary class distribution of approximately 55:45. We split the dataset into approximately 15,000 data points for training, 5,000 for validation, and 6,000 for testing.

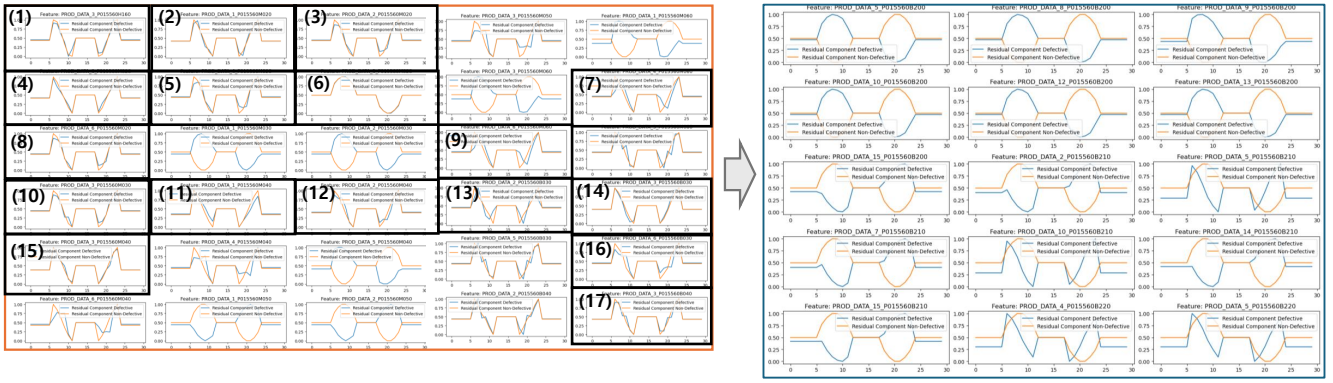
Fig. 8 presents the results of analyzing trend component time series patterns for each sensor in the company F manufacturing dataset, as illustrated in Fig. 3, to select significant features. As shown in Fig. 8 (1)-(2), the trend component time series patterns of certain sensors exhibit similar patterns, and thus, these sensors are excluded from the set of significant features. After applying the same process to residual components, we ultimately selected 12 significant features out of a total of 13 sensors. Table 1 presents the test dataset performance for time series classification experiments conducted on the Kaggle dataset. For AutoGluon.Tabular, the result shown represent the top 3 models with the best performance. The computing time indicates the total time required to perform classification on the entire test dataset. For AutoML tools, we allowed sufficient tuning time by configuring the maximum training time or iterations, and reported the best-performing results selected by the internal model selection process in the table.

The AutoGluon.Tabular model demonstrated better performance, achieving an accuracy of approximately 0.7925 and an F1-Score of around 0.8065, outperforming our non-time series pipeline by approximately 1 to 2%. Additionally, other AutoML tools, such as PyCaret and TPOT, showed slightly lower performance than our non-time series pipeline. These results suggest that current time series AutoML tools still face limitations in effectively handling time series classification tasks.

Although Our ML exhibited lower performance compared to existing non-time series AutoML tools, Our TS DL produced the opposite results. As shown in the table above, the performance of TS using Non-Feature Selection and Our TS DL method using time series pattern analysis feature selection were the highest. When comparing our TS DL model with feature selection to the Non-Feature Selection TS model, we observed significant improvements in both performance and speed, consistent with the results of Experiment 1. Specifically, our TS DL model achieved an increase of approxi-

TABLE 1. Classification performance comparison of the kaggle dataset

Method	Model	Accuracy	Recall	F1-Score	Computing Time
AutoGluon.Tabular	WeightedEnsemble [61]	0.7925	0.8673	0.8062	4.7 (sec)
	NeuralNet	0.7918	0.8717	0.8065	3.8 (sec)
	CatBoost [62]	0.7965	0.8461	0.8054	3.4 (sec)
PyCaret	Best model	0.728	0.755	0.734	0.26 (sec)
TPOT	Best model	0.761	0.715	0.751	0.1 (sec)
TSFresh	Best Model	0.795	0.791	0.797	1.2 (sec)
Catch22	Best Model	0.728	0.729	0.729	0.45 (sec)
Non-Feature Selection	TS DL 1	0.8843 ±0.002	0.8864 ±0.004	0.8842 ±0.023	11.8 (sec)
	TS DL 2	0.8813 ±0.013	0.8912 ±0.005	0.8824 ±0.006	10.6 (sec)
	TS DL 3	0.8352 ±0.023	0.8192 ±0.013	0.8411 ±0.012	14.07 (sec)
	TimesNet	0.8086 ±0.012	0.8061 ±0.02	0.8074 ±0.035	7.1 (sec)
	SegRNN	0.8024 ±0.03	0.8168 ±0.02	0.8045 ±0.03	5.3 (sec)
Our TS DL	TS DL 1	**0.8973 ±0.003**	**0.8915 ±0.003**	**0.8971 ±0.002**	10.7 (sec)
	TS DL 2	0.8832 ±0.001	0.8927 ±0.003	0.8837 ±0.003	10.1 (sec)
	TS DL 3	**0.8917 ±0.001**	**0.9165 ±0.001**	**0.8952 ±0.001**	13.8 (sec)
	TimesNet [56]	0.8282 ±0.002	0.8142 ±0.002	0.8136 ±0.005	6.9 (sec)
	SegRNN [57]	0.8017 ±0.02	0.8138 ±0.01	0.8142 ±0.04	5.1 (sec)
Our ML	Random Forest [63]	0.7502	0.7158	0.7441	0.13 (sec)
	XGB [64]	0.7864	0.7776	0.7869	0.15 (sec)
	LightGBM [65]	0.781	0.7702	0.7811	0.03 (sec)



(a) ALL Feature Residual PCA Result (b) Residual component Feature selection Result

FIGURE 9. Feature selection results using time series pattern analysis on the F Company dataset. (a) Results of residual patterns extracted using STL and PCA applied to a subset of features, where (1)–(17) represent non-significant features. (b) Results after selecting significant features using DTW and clustering outcomes.

mately 1.69% in accuracy and 1.55% in F1-Score. In addition to this performance improvement, the computing time of our TS DL model was reduced by approximately 9.3%. These results demonstrate that, even in complex and highly interactive time series datasets, eliminating unimportant single features among the 13 not only reduces model complexity but also optimizes computational efficiency.

C. EXPERIMENTAL 2

In this section, we conduct experiments on a real-world dataset from the manufacturing process of automotive parts by company F. The dataset has a distribution of approximately 98% non-defective and 2% defective products. We split the dataset into approximately 60,000 data points for training, 20,000 for validation, and 20,000 for testing.

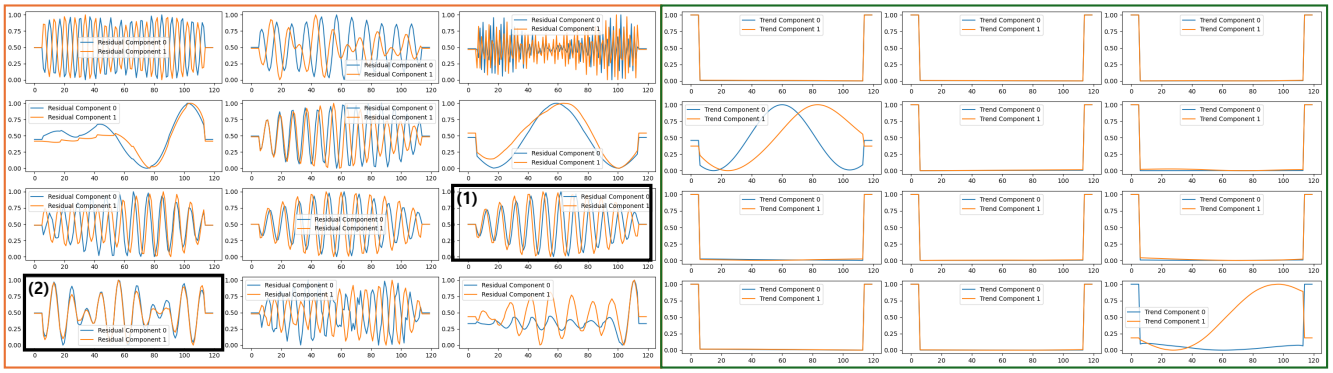
Fig. 9 presents the results of analyzing residual component time series patterns for each sensor in the company F manufacturing dataset, as illustrated in Fig. 3, to select significant features. As shown in Fig. 9 (1)-(17), the residual time series patterns of certain sensors exhibit similar patterns, and thus, these sensors are excluded from the set of significant features. After applying the same process to trend components, we ultimately selected 126 significant features out of a total of 196 sensors. Table 2 presents the results of comparing the classification performance of various AutoML methods and models based on the manufacturing dataset from Company F. For AutoGluon.Tabular, the results shown represent the top 3 models with the best performance. The computing time indicates the total time required to perform classification on the entire test dataset.

When transforming the time series dataset into a non-time series format, AutoGluon.Tabular, PyCaret, and TPOT demonstrated relatively good performance. Despite achieving 99.7% accuracy, all three AutoML methods exhibited comparatively low F1-Score and Recall. This suggests that they failed to address the challenges posed by high-dimensional features, an extreme 98:2 class imbalance, and the difference in sequence length for each product.

TABLE 2. Classification performance comparison of the Company F dataset

Method	Model	Accuracy	Recall	F1-Score	Computing Time
AutoGluon.Tabular	WeightedEnsemble	0.997	0.790	0.881	6.8 (sec)
	NeuralNet	0.997	0.805	0.881	7.6 (sec)
	Random forest	0.997	0.794	0.883	6.3 (sec)
PyCaret	Best Model	0.997	0.7941	0.879	1.88 (sec)
TPOT	Best Model	0.997	0.8125	0.891	0.18 (sec)
TSFresh	Best Model	0.997	0.831	0.890	1.58 (sec)
Catch22	Best Model	0.996	0.757	0.843	0.82 (sec)
Non-Feature Selection	TS DL 1	0.999 ±0.0002	0.958 ±0.002	0.979 ±0.003	16.6 (sec)
	TS DL 2	0.999 ±0.0001	0.963 ±0.001	0.975 ±0.001	16.9 (sec)
	TS DL 3	0.999 ±0.0001	955 ±0.003	0.964 ±0.004	17.3 (sec)
	TimesNet	0.999 ±0.0002	0.963 ±0.004	0.971 ±0.005	10.1 (sec)
	SegRNN	0.998 ±0.001	0.951 ±0.011	0.963 ±0.008	9.8 (sec)
Our TS DL	TS DL 1	0.999 ±0.0001	0.963 ±0.002	0.980 ±0.002	14.2 (sec)
	TS DL 2	0.999 ±0.0002	0.959 ±0.003	0.979 ±0.002	14.4 (sec)
	TS DL 3	**0.999 ±0.0002**	**0.964 ±0.002**	**0.981 ±0.003**	14.5 (sec)
	TimesNet	**0.999 ±0.0002**	**0.973 ±0.002**	**0.986 ±0.003**	9.2 (sec)
	SegRNN	0.999 ±0.0004	0.963 ±0.003	0.980 ±0.003	8.7 (sec)
Our ML	Random Forest	0.993	0.684	0.807	0.32 (sec)
	XGB	0.997	0.894	0.934	0.12 (sec)
	LightGBM	0.998	0.914	0.948	0.26 (sec)

In contrast, Our ML demonstrated superior performance in handling the highly imbalanced manufacturing dataset with 196 variables, outperforming AutoGluon.Tabular, PyCaret, and TPOT in this scenario. While AutoGluon.Tabular achieved the highest accuracy among the three AutoML tools, with CatBoost at 0.7965 accuracy and 0.8054 F1-score, it still struggled with class imbalance, as indicated by lower recall scores. PyCaret and TPOT, despite their relatively high accuracy of 0.728 and 0.761, respectively, exhibited lower recall and F1-scores, suggesting difficulty in distinguishing defective from non-defective cases. In contrast, Our ML models, particularly XGBoost and LightGBM, achieved 0.9976 and 0.9981 accuracy, respectively, with well-balanced recall (0.8947and 0.9144) and F1-scores (0.9347and 0.9488), making them more effective for high-dimensional, imbalanced



(a) ALL Feature Residual PCA Result (b) ALL Feature Trend PCA Result

FIGURE 10. Feature selection results using time series pattern analysis on the D Company dataset. (a) Residual pattern results extracted via STL and PCA, where (1)–(2) represent non-significant features. (b) Trend results showing that most sensors exhibit similar trends, indicating potential redundancy.

manufacturing data. Also, The Our ML models, which applied the method described in Fig. 7, proved to be highly efficient, particularly in terms of Computing Time. Notably, XGB achieved the fastest inference time of 0.12 seconds, while simultaneously maintaining high performance, demonstrating both efficiency and accuracy.

The comparison between the Non-Feature Selection models, which do not apply the structure shown in Fig. 3, and Our TS DL models, which incorporate this structure, demonstrates that the TS DL models achieved high Accuracy, Recall, and F1-Score, showcasing excellent performance. This indicates that the multi-layer Bidirectional GRU structure we adopted is well-suited for manufacturing process datasets with a very low defect rate. Furthermore, the Our TS DL models with Feature Selection outperformed the Non-Feature Selection models. Notably, the TS DL 1 and TS DL 3 models achieved the highest Accuracy (0.9995), Recall (0.9632), and F1-Score (0.9812). Additionally, the Computing Time of the Our TS DL models was successfully reduced by approximately 15.1% compared to the Non-Feature Selection TS DL models. These results clearly demonstrate the effectiveness of Feature Selection in enhancing both performance and efficiency.

Overall, the results clearly indicate that Our ML and TS DL approaches are highly effective for defect classification in manufacturing datasets, especially when dealing with high-dimensional, class-imbalanced time series data. While existing AutoML tools such as AutoGluon.Tabular, PyCaret, and TPOT provide acceptable performance in traditional machine learning settings, they struggle with distinguishing defective from non-defective products due to their limited handling of sequential dependencies, feature selection, and class imbalance.

D. EXPERIMENTAL 3

Fig. 10 presents the trend and residual analysis results for the Company D fermentation dataset. Similar to the procedure applied to Company F, we analyzed time series patterns to identify discriminative features. Although the trend components across different classes appeared highly similar, residual patterns revealed distinct differences. Based on this observation, we excluded the top two features that exhibited highly similar residual patterns across non-defective and defective cases, as they were not considered effective for classification. As a result, 10 significant features were selected out of the original 12. The experimental setup and evaluation protocol remained consistent with the previous two experiments.

TABLE 3. Classification performance of the Company D dataset

Method	Model	Accuracy	Recall	F1-Score	Computing Time
AutoGluon.Tabular	WeightedEnsemble	0.922	0.745	0.804	2.1 (sec)
	LightGBM	0.918	0.750	0.797	2.1 (sec)
	NeuralNet	0.917	0.735	0.790	2.2 (sec)
PyCaret	Best Model	0.782	0.015	0.028	0.1 (sec)
TPOT	Best Model	0.7608	0.744	0.830	0.1 (sec)
TSFresh	Best Model	0.802	0.814	0.796	1.32 (sec)
Catch22	Best Model	0.788	0.716	0.708	0.62 (sec)
Non-Feature Selection	TS DL 1	0.980 ±0.002	0.905 ±0.003	0.950 ±0.004	15.6 (sec)
	TS DL 2	0.977 ±0.001	0.905 ±0.002	0.945 ±0.001	14.2 (sec)
	TS DL 3	0.963 ±0.004	0.831 ±0.013	0.907 ±0.005	16.1 (sec)
	TimesNet	0.985 ±0.002	0.941 ±0.013	0.964 ±0.005	9.2 (sec)
	SegRNN	0.964 ±0.002	0.872 ±0.019	0.912 ±0.0007	7.3 (sec)
Our TS DL	TS DL 1	0.980 ±0.003	0.906 ±0.005	0.949 ±0.004	15.1 (sec)
	TS DL 2	0.979 ±0.002	0.905 ±0.001	0.950 ±0.002	13.8 (sec)
	TS DL 3	0.976 ±0.003	0.886 ±0.011	0.939 ±0.002	15.9 (sec)
	TimesNet	**0.988 ±0.002**	**0.954 ±0.008**	**0.972 ±0.005**	8.1 (sec)
	SegRNN	0.969 ±0.004	0.875 ±0.028	0.923 ±0.012	6.8 (sec)
Our ML	Soft	0.923	0.760	0.811	0.4 (sec)
	XGB	0.7436	0.336	0.361	0.1 (sec)
	Hard	0.851	0.383	0.525	0.4 (sec)

Table 3 represents the results of classification performance of various methods and models based on the manufacturing dataset from Company D. Compared to the non-feature selection setting, our TS DL models showed consistent improvements across all metrics. On average, accuracy increased by more than 1.5%, recall by approximately 3.4%, and F1-score by over 2.5%. These results highlight the effectiveness of incorporating feature selection in time-series modeling, especially for capturing meaningful temporal patterns and

improving generalization performance.

The performance improvement is primarily attributed to our proposed deep feature selection method, which filters out features that show minimal temporal pattern differences between non-defective and defective production. Instead of removing noisy inputs, the method evaluates the discriminative power of each feature based on residual and trend pattern analysis and retains only those that contribute meaningfully to the classification task. As shown in Fig. 10, although the overall trends of sensor signals appear similar across non-defective and defective cases in the fermentation process of Company D, distinct differences emerge in the residual patterns. This is because fermentation processes often involve complex biological dynamics where surface-level trends may mask subtle deviations in cellular responses. Residual analysis allows us to isolate these deviations, which are critical for identifying abnormal behavior that is not apparent from raw signal trends alone. This selective representation enhances the model focus on critical signals while reducing the influence of irrelevant temporal dynamics.

E. ABLATION STUDY

In Section Experimental 1, we conducted experiments using a dataset with balanced classes and a fixed window size. The experimental results confirmed that our deep learning (TS DL) models exhibited outstanding performance on such a well-processed time series dataset. However, in terms of computing time, TS DL models required significantly higher computational resources.

On the other hand, the our ML model, which utilized the method shown in Fig. 7, demonstrated an advantage in computing time over the AutoML framework AutoGluon.Tabular. However, its performance was relatively lower. In contrast, when applying the same models to real-world manufacturing data in Experimental 2 and 3, the results were completely reversed. As shown in Table 2 and 3, the our ML model, using the method from Fig. 7, outperformed AutoGluon.Tabular in terms of computing time as well as recall and F1-score.

This differences of result between experimental 2, 3 and 1 in ML models is attributed to the unique characteristics of real manufacturing data. Unlike well curated datasets such as those on Kaggle, real-world manufacturing data contains over 100 high-dimensional features and exhibits severe class imbalance issues [13]–[15], [66]. In manufacturing environments where computing time is not a major constraint, our TS DL demonstrates performance suitable for autonomous manufacturing, as observed in Experimental 2. However, in scenarios where computing time is a critical factor, our TS DL may not be the optimal choice, necessitating the use of ML models with relatively shorter inference times.

Given this context, a comparison of the results from Experimental 1 and Experimental 2 highlights that, unlike standard AutoML approaches, our ML is more suitable for handling class imbalance and high-dimensional manufacturing data with over 100 variables. To further analyze these findings in depth, we conduct an ablation study to examine whether our ML is suitable for handling the challenges posed by real-time computing constraints, the high dimensionality of manufacturing data, and severe class imbalance. This study aims to systematically evaluate the impact of these factors on model performance and verify the robustness of our ML under real-world manufacturing conditions.

TABLE 4. Classification performance comparison of the PhysioNet Dataset

Method	Model	Recall	F1-Score	Computing Time
AutoGluon.Tabular (300)	WeightedEnsemble	0.345	0.455	1.7 (sec)
	CatBoost	0.345	0.455	1.7 (sec)
	LightGBM	0.290	0.410	1.7 (sec)
9+ AutoGluon.Tabular (1200)	WeightedEnsemble	0.336	0.445	5.2 (sec)
	LightGBM	0.336	0.445	5.2 (sec)
	CatBoost	0.305	0.421	5.3 (sec)
AutoGluon.Tabular (2400)	WeightedEnsemble	0.324	0.436	5.3 (sec)
	CatBoost	0.324	0.436	5.4 (sec)
	XGBoost	0.326	0.440	5.1 (sec)
AutoGluon.Tabular (3600)	WeightedEnsemble	0.326	0.436	5.1 (sec)
	CatBoost	0.326	0.436	5.1 (sec)
	XGBoost	0.314	0.424	5.4 (sec)
PyCaret	Best Model	0.280	0.396	0.18 (sec)
TPOT	Best Model	0.537	0.456	0.04(sec)
Original ML	Random Forest	0.126	0.216	0.15 (sec)
	XGB	0.353	0.407	0.07 (sec)
	LightGBM	0.338	0.401	0.1 (sec)
Our ML	Random Forest	0.393	0.421	0.12 (sec)
	XGB	0.610	**0.505**	**0.05** (sec)
	LightGBM	**0.611**	0.484	0.08 (sec)

We use PhysioNet dataset in this ablation study. This dataset, form Physionet Challenge 2012 dataset [67], is a publicly available collection of multivariate clinical time series from 12,000 intensive care unit (ICU) records. The original dataset consists of 33 variables with a window size of approximately 80. To train the ML models, we generated statistical and volatility features 8for each Record ID, resulting in a total of 126 variables. Additional, The PhysioNet dataset is sensor-based data with over 100 high-dimensional features. Moreover, it exhibits severe class imbalance, making it similar in nature to manufacturing data. In particular, the binary class ratio is highly imbalanced at 14:86. Therefore, we utilize this dataset as the experimental data for our Ablation Study. The experiments were conducted using this extended feature set. Out of a total of 12,000 data points (with 4,000 in each of set-a, set-b, and set-c), we merged set-a and set-b and split them into training and validation datasets at a 6,000:2,000 ratio. Meanwhile, set-c was used as the test dataset.

To validate this, we designed the experiment as follows: We train the AutoGluon.Tabular model for 300, 1200, 2400, and 3600 seconds, respectively, and compare their performance. Additionally, we evaluate the original ML model (without our Feature Selection technique), as well as the best models from PyCaret and TPOT, against our ML model (with Feature Selection applied) to further analyze the impact of our approach.

These varied training durations (300s, 1200s, 2400s, 3600s) allow us to assess the trade-off between computational time and model performance in AutoML models. In manufacturing environments, real-time processing capability is a critical factor. While longer training times may improve model performance, they also increase computational costs. Therefore, it is essential to determine the optimal training duration that balances efficiency and performance under limited

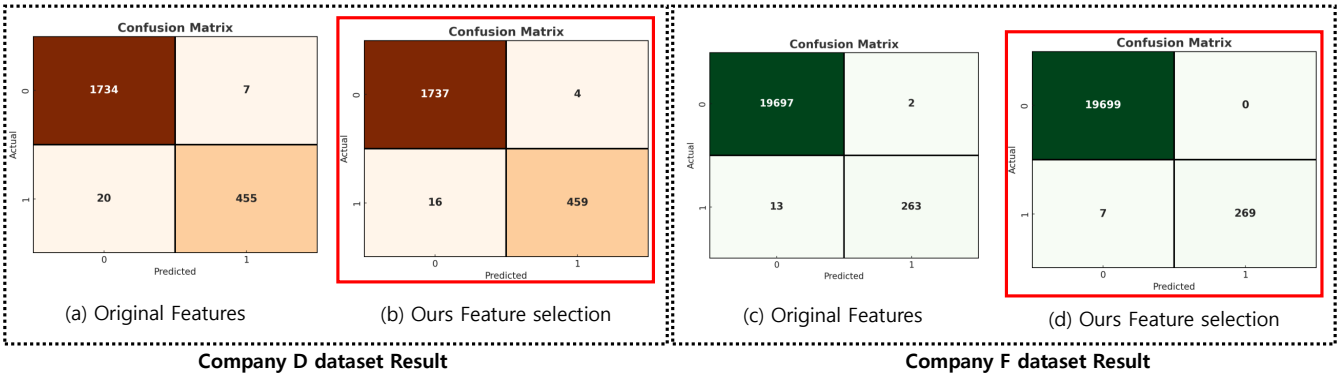


FIGURE 11. Confusion matrix illustrating the classification results of the best model on the real-world dataset.

computational resources. Furthermore, by comparing this to Our ML with Feature Selection, we aim to verify whether our approach can maintain high performance while significantly improving computational efficiency.

The results in Table 4 indicate that Our ML is highly effective in addressing challenges posed by real-world manufacturing-like data, particularly in handling high dimensionality, severe class imbalance, and real-time computing constraints. These results systematically evaluate the impact of these factors on model performance and confirm the robustness of non-time series feature selection method under realistic manufacturing conditions.

Our ML significantly outperform AutoGluon.Tabular, which was trained with various time settings. While AutoGluon.Tabular (3600s) achieves a maximum recall of 0.326 and an F1-score of 0.440, Our ML models (XGB and LightGBM) achieve recall scores of 0.610 and 0.611, respectively, and F1-scores of 0.505 and 0.484, demonstrating superior classification ability, especially in handling imbalanced datasets. XGB of Our ML achieves the best F1-score (0.505) while requiring only 0.05 seconds for inference, making it the best-performing model in terms of both speed and accuracy. Similarly, LightGBM achieves a strong recall (0.611) with a short inference time of 0.08 seconds, demonstrating the effectiveness of our non-time series feature selection method in reducing inference costs.

Additionally, while TPOT achieves a recall score of 0.537, its F1-score (0.456) is lower than that of Our ML models, suggesting that TPOT struggles to balance precision and recall effectively. In contrast, PyCaret exhibits the lowest recall (0.280) and F1-score (0.396), confirming its limitations in handling complex, high-dimensional medical datasets such as PhysioNet. Finally, compared to the Original ML models, Our ML models show significant improvements. Overall, recall and F1-score performance improved by approximately 26% and 13%, emphasizing the effectiveness of our non-time series feature selection approach in handling imbalanced datasets.

TABLE 5. Effect size (Cohen's *d*) and confidence intervals (CI) of F1-score improvements on Company D and Company F datasets

Dataset	Model	Mean Diff	Cohen's *d*	CI Lower	CI Upper	Significant
Company D	TS DL 1	-0.001	-0.250	-0.0060	0.0040	No
	TS DL 2	0.005	3.162	0.0030	0.0070	Yes
	TS DL 3	0.032	8.404	0.0273	0.0367	Yes
	TimesNet	0.008	1.600	0.0018	0.0142	Yes
	SegRNN	0.011	1.294	0.0005	0.0215	Yes
Company F	TS DL 1	0.001	0.408	-0.0023	0.0043	No
	TS DL 2	0.004	2.828	0.0023	0.0057	Yes
	TS DL 3	0.017	4.717	0.0119	0.0221	Yes
	TimesNet	0.015	3.536	0.0093	0.0207	Yes
	SegRNN	0.017	2.400	0.0046	0.0294	Yes

V. DISSCUSSION

Our time series feature selection framework, which consists of STL decomposition, production wise PCA, DTW similarity computation, and clustering, provides a systematic approach to identify time series features that effectively distinguish between defective and non-defective products. Rather than altering the input signals, the method selects informative sensors by analyzing the trend and residual patterns across multiple products. These selected features are then used in their original form to train deep learning models, preserving the full fidelity of the data.

What distinguishes our method is its ability to flexibly detect class-separating signals either in the long-term trend or short-term fluctuations (residuals). In the Company D dataset, no meaningful separation could be observed from the trend component. However, when we applied our method to the residuals, we identified several sensors that strongly distinguished defective from non-defective cases. This demonstrates that often overlooked residual patterns carry crucial information about localized anomalies or deviations indicative of product defects.

Table. 5 presents the effect sizes and 95% confidence intervals for F1-score improvements across Company D and Company F. TS DL 2 and TS DL 3 consistently demonstrate large effect sizes and statistically significant improvements. These results highlight the importance of balancing model size and stability to fully benefit from feature selection. However, In both Company D and Company F, TS DL 1, which contains over 150 million parameters, did not show statistically signif-

icant improvements despite feature selection. This suggests that when model size becomes excessively large, the benefits of feature selection may diminish due to increased training variance and reduced stability.

As shown in Fig. 8, utilizing the feature selection process in the TS DL model led to significant improvements in both performance and computation speed compared to models without feature selection. These results align with those of Experiment 1 in Table 1. Specifically, the TS DL model demonstrated notable improvements, with accuracy increasing by 1.69%, F1-score improving by 1.55%, and computational time reducing by 9.3%. Similarly, in Experiment 2, as presented in Table 2, the TS DL model applied to the manufacturing dataset not only outperformed traditional statistical models in terms of performance and computational efficiency but also exhibited enhanced performance over previous models through feature selection, achieving an average reduction of 20% in computational time. These results highlight that effectively excluding irrelevant features in complex and interactive time series datasets not only reduces model complexity but also optimizes computational efficiency.

These effects help deep learning models focus on structurally relevant inputs. Figure 11 shows the confusion matrices of the highest-performing deep learning models on the Company D and Company F datasets. This leads to clearer decision boundaries and more reliable predictions. The confusion matrices support this: in Company D, false negatives decreased from 20 to 16, and true positives increased from 455 to 459. In Company F, false negatives dropped from 13 to 7, and false positives were completely eliminated.

While time series models show high performance, we have observed that the inference time is somewhat longer than other standard ML models. Therefore, we looked at the results for a machine learning (ML) feature selection process with short inference time. As shown in the experimental results in Table 1, the ML model incorporating our proposed feature selection method achieved faster computation speeds compared to other ML-based approaches. In terms of performance, our model surpassed most ML models. However, when compared to AutoGluon, it achieved an F1-score of 78%, slightly lower than 80% of AutoGluon. Nevertheless, in Experiment 2 and 3, which utilized real-world manufacturing process data (Table 2 and 3), our ML model achieved an F1-score of approximately 95%, surpassing the 88% of conventional ML models and demonstrating superior classification performance over AutoML.

To further investigate this, an ablation study was conducted using the PhysioNet dataset, which contains a highly imbalanced class distribution (14:86) and a large number of features, similar to manufacturing environments. The results showed that, in terms of F1-score, the proposed model achieved at least a 5% improvement compared to existing models, with an average increase of 5 to 10%. Furthermore, regarding recall, particularly in detecting the minority class, the proposed model demonstrated an improvement of at least 8%, with an average increase exceeding 10%. Additionally, inference time was significantly shorter compared to traditional models, further validating the efficiency of the proposed approach. These findings indicate that the ML-based feature selection process is particularly effective in environments characterized by numerous manufacturing sensors, complex feature interactions, and imbalanced datasets.

In time series feature engineering, Catch22 extracts features from the entire time series without window segmentation, making it difficult to capture localized defect patterns [23]. As the window size increases (30 for Company F, 60 for Kaggle, 120 for Company D), its F1-score drops by approximately 9%, 11%, and 18% compared to the best performing models. Likewise, TSfresh extracts hundreds of features per univariate time series, resulting in high computational cost and an increased risk of overfitting [22]. Our results showed excellent training and validation performance but a clear degradation on the test dataset. Specifiy, In Experiment 1, the training and validation accuracy reached 100%, whereas the test accuracy dropped to 70%. Similarly, for the company D, we observed over 90% accuracy during training and validation phases, but only 80% range accuracy on the test dataset. This clearly demonstrates that excessive feature generation can lead to overfitting, highlighting the need for careful feature selection and engineering that reflects the high-dimensional characteristics of time series data in manufacturing.

Neverless, Compared to time series feature engineering methods, our ML approach achieved faster inference but only marginal performance gains. That is, our non-time series feature selection method outperforms AutoML approaches that do not support time series classification or ignore temporal dependencies. However, the performance gains were limited compared to time series feature engineering methods. This indicates that integrating our feature selection approach with strategies that can preserve or enhance temporal representations remains an open challenge for achieving both efficiency and accuracy in time series classification tasks.

VI. CONCLUSION

Manufacturing companies are making substantial investments in automation to enhance operational efficiency. However, manufacturing processes vary significantly depending on product types, leading to diverse data characteristics. This variability makes it challenging to ensure consistent model performance when applying the same feature engineering techniques or derived variable generation methods. Additionally, to enable real-time AI-driven validation of products, the model prediction time must be shorter than the production time of a single product. This highlights the need for rapid and efficient adaptability in model training processes to accommodate new products or changing process conditions [68]–[70].

To improve both speed and performance, we proposed two feature selection strategies that yielded significant enhancements in both TS DL and ML models. Our study has shown that integrating advanced feature selection techniques into

time series and ML-based models can significantly improve predictive performance and computational efficiency. The proposed methodology improves model accuracy, optimizes resource utilization, and enables real-time quality validation, making it particularly suitable for complex manufacturing environments. Additionally our automated pipeline successfully reduces the time required for AI deployment by efficiently transforming time series data into a format suitable for ML-based model. This approach facilitates real-time quality inspection and rapid adaptation to new products and process modifications.

There are a few limitations to be acknowledged in this research. For the ML model, the improvement in inference time was very remarkable, but it did not satisfy sufficient model performance. On the other hand, for the TS DL model, the improvement in performance was remarkable, but there was a limit to the improvement in inference speed. This limitation arises because TS DL models necessitate more complex algorithms and numerous parameter computations to analyze the inherent patterns in time series data, which restricts their speed. Therefore, developing a TS DL optimized for feature selection is essential to address this challenge.

In future work, we plan to explore methods for effectively interpreting multivariate time-series datasets. By combining feature engineering and feature selection techniques, we aim to mitigate overfitting and further enhance model performance. Additionally, to maintain the performance of our time series deep learning (TS DL) model while increasing its speed by using shallow convolutional layers to extract the significant sequences from significant features [71]. Instead of using the entire sequence as in conventional approaches, we intend to investigate lightweight GRU or Transformer [72] models that train solely on these extracted significant sequences.

In addition, there are still important challenges in automating the manufacturing process, as well as the performance of the model itself. If future research can effectively use and research AI to analyze the data itself and actually make decisions about not only whether or not defects actually occurred, but also why these defects occurred, it will promote sustainable and intelligent growth in various industries.

ACKNOWLEDGMENT

This research was financially supported by the Ministry of Trade, Industry, and Energy (MOTIE), Korea, under the "Global Industrial Technology Cooperation Center(GITCC) program" supervised by the Korea Institute for Advancement of Technology (KIAT).(Task No. P0028468)

REFERENCES

[1] R. Cioffi, et al., 'Artificial intelligence and machine learning applications in smart production: Progress, trends, and directions,' *Sustainability*, vol. 12, no. 2, p. 492, 2020.

[2] M.-L. Tseng, et al., 'Sustainable industrial and operation engineering trends and challenges toward Industry 4.0: A data-driven analysis,' *Journal of Industrial and Production Engineering*, vol. 38, no. 8, pp. 581–598, 2021.

[3] H. Jung, et al., 'Application of machine learning techniques in injection molding quality prediction: Implications on sustainable manufacturing industry,' *Sustainability*, vol. 13, no. 8, p. 4120, 2021.

[4] L. Waltersmann, et al., 'Artificial intelligence applications for increasing resource efficiency in manufacturing companies—a comprehensive review,' *Sustainability*, vol. 13, no. 12, p. 6689, 2021.

[5] Y. Chen and S. Jin, 'Artificial intelligence and carbon emissions in manufacturing firms: The moderating role of green innovation,' *Processes*, vol. 11, no. 9, p. 2705, 2023.

[6] R. Rakholia, et al., 'Advancing manufacturing through artificial intelligence: Current landscape, perspectives, best practices, challenges, and future direction,' *IEEE Access*, 2024.

[7] R. S. Peres, et al., 'Industrial artificial intelligence in Industry 4.0-systematic review, challenges, and outlook,' *IEEE Access*, vol. 8, pp. 220121–220139, 2020.

[8] J. F. Arinez, et al., 'Artificial intelligence in advanced manufacturing: Current status and future outlook,' *Journal of Manufacturing Science and Engineering*, vol. 142, no. 11, p. 110804, 2020.

[9] C.-J. Wu, et al., 'Beyond efficiency: Scaling AI sustainably,' *IEEE Micro*, 2024.

[10] R. S. Peres, et al., 'Industrial artificial intelligence in Industry 4.0-systematic review, challenges, and outlook,' *IEEE Access*, vol. 8, pp. 220121–220139, 2020.

[11] B. Wang, 'The future of manufacturing: A new perspective,' *Engineering*, vol. 4, no. 5, pp. 722–728, 2018.

[12] J. Li and X. Jin, 'The impact of artificial intelligence adoption intensity on corporate sustainability performance: The moderated mediation effect of organizational change,' *Sustainability*, vol. 16, no. 21, p. 9350, 2024.

[13] C.-Y. Hsu and W.-C. Liu, 'Multiple time-series convolutional neural network for fault detection and diagnosis and empirical study in semiconductor manufacturing,' *Journal of Intelligent Manufacturing*, vol. 32, no. 3, pp. 823–836, 2021.

[14] M. A. Farahani, et al., 'Time-series classification in smart manufacturing systems: An experimental evaluation of state-of-the-art machine learning algorithms,' *Robotics and Computer-Integrated Manufacturing*, vol. 91, p. 102839, 2025.

[15] M. A. Farahani, et al., 'Time-series pattern recognition in Smart Manufacturing Systems: A literature review and ontology,' *Journal of Manufacturing Systems*, vol. 69, pp. 208–241, 2023.

[16] Z. Liu, et al., 'Forecast methods for time series data: A survey,' *IEEE Access*, vol. 9, pp. 91896–91912, 2021.

[17] S. S. W. Fatima and A. Rahimi, 'A review of time-series forecasting algorithms for industrial manufacturing systems,' *Machines*, vol. 12, no. 6, p. 380, 2024.

[18] A. Gerling, et al., 'Results from using an AutoML tool for error analysis in manufacturing,' *ICEIS (1)*, 2022.

[19] W. Zhai, et al., 'Explainable AutoML (xAutoML) with adaptive modeling for yield enhancement in semiconductor smart manufacturing,' *arXiv preprint*, arXiv:2403.12381, 2024.

[20] Ghosh, K., Bellinger, C., Corizzo, R., Branco, P., Krawczyk, B., and Japkowicz, N. (2024). The class imbalance problem in deep learning. Machine Learning, 113(7), 4845-4901.

[21] Khan, A., Rasheed, M. T., and Khan, H. (2023). An empirical study of deep learning-based feature extractor models for imbalanced image classification. Advances in Computational Intelligence, 3(6), 20.

[22] Christ, M., Braun, N., Neuffer, J., and Kempa-Liehr, A. W. (2018). Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package). Neurocomputing, 307, 72-77.

[23] Lubba, C. H., Sethi, S. S., Knaute, P., Schultz, S. R., Fulcher, B. D., and Jones, N. S. (2019). catch22: CAnonical Time-series CHaracteristics: Selected through highly comparative time-series analysis. Data mining and knowledge discovery, 33(6), 1821-1852.

[24] R. B. Cleveland, et al., 'STL: A seasonal-trend decomposition,' *J. Off. Stat.*, vol. 6, no. 1, pp. 3–73, 1990.

[25] K. Pearson, 'LIII. On lines and planes of closest fit to systems of points in space,' *Philos. Mag.*, vol. 2, no. 11, pp. 559–572, 1901.

[26] M. Müller, 'Dynamic time warping,' in *Information Retrieval for Music and Motion*, pp. 69–84, 2007.

[27] S. Na, X. Li, and G. Yong, 'Research on k-means clustering algorithm: An improved k-means clustering algorithm,' in *Proc. 2010 Third International Symposium on Intelligent Information Technology and Security Informatics*, IEEE, 2010.

[28] L. St and S. Wold, 'Analysis of variance (ANOVA),' *Chemometrics and Intelligent Laboratory Systems*, vol. 6, no. 4, pp. 259–272, 1989.

[29] D. C. Montgomery and G. C. Runger, *Applied statistics and probability for engineers*, 5th ed., John Wiley & Sons, 2010.
[30] A. Gerling, 'Towards an implementation of an AutoML tool for manufacturing,' Ph.D. dissertation, Université de Haute Alsace-Mulhouse, 2022.
[31] J. Cação, et al., 'Intelligent assistant for smart factory power management,' *Procedia Computer Science*, vol. 232, pp. 966–979, 2024.
[32] E. L. Naour, L. Serrano, L. Migus, Y. Yin, G. Agoua, N. Baskiotis, ... and V. Guigue, 'Time series continuous modeling for imputation and forecasting with implicit neural representations,' *arXiv preprint arXiv:2306.05880*, 2023.
[33] J. Y. L. Chan, S. M. H. Leow, K. T. Bea, W. K. Cheng, S. W. Phoong, Z. W. Hong, and Y. L. Chen, 'Mitigating the multicollinearity problem and its machine learning approach: a review,' Mathematics, vol. 10, no. 8, p. 1283, 2022.
[34] E. LeDell and S. Poirier, 'H2o AutoML: Scalable automatic machine learning,' in *Proceedings of the AutoML Workshop at ICML*, 2020.
[35] M. Feurer, et al., 'Auto-sklearn 2.0: The next generation,' *arXiv preprint*, arXiv:2007.04074, 2020.
[36] O. Shchur, et al., 'AutoGluon–TimeSeries: AutoML for probabilistic time series forecasting,' *International Conference on Automated Machine Learning*, PMLR, 2023.
[37] C. Wang, et al., 'AutoTS: Automatic time series forecasting model design based on two-stage pruning,' *arXiv preprint*, arXiv:2203.14169, 2022.
[38] A. Alsharef, et al., 'Review of ML and AutoML solutions to forecast time-series data,' *Archives of Computational Methods in Engineering*, vol. 29, no. 7, pp. 5297–5311, 2022.
[39] S. J. Taylor and B. Letham, 'Forecasting at scale,' *The American Statistician*, vol. 72, no. 1, pp. 37–45, 2018.
[40] H. I. Fawaz, 'Deep learning for time series classification,' *arXiv preprint arXiv:2010.00567*, 2020.
[41] J. F. Torres, D. Hadjout, A. Sebaa, F. Martínez-Álvarez, and A. Troncoso, 'Deep learning for time series forecasting: a survey,' *Big Data*, vol. 9, no. 1, pp. 3-21, 2021.
[42] K. Azevedo, et al., 'A multivocal literature review on the benefits and limitations of automated machine learning tools,' *arXiv preprint*, arXiv:2401.11366, 2024.
[43] V. Fonti and E. Belitser, 'Feature selection using LASSO,' *VU Amsterdam Research Paper*, vol. 30, pp. 1–25, 2017.
[44] K. Han, et al., 'Autoencoder inspired unsupervised feature selection,' in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.
[45] B. D. Fulcher and N. S. Jones, 'Highly comparative feature-based time-series classification,' *IEEE Trans. on Knowledge and Data Engineering*, vol. 26, no. 12, pp. 3026–3037, 2014.
[46] L. Huang, et al., 'Time series feature selection method based on mutual information,' *Applied Sciences*, vol. 14, no. 5, p. 1960, 2024.
[47] R. Kohavi and G. H. John, "Wrappers for feature subset selection," Artificial Intelligence, vol. 97, no. 1-2, pp. 273-324, 1997.
[48] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," Machine Learning, vol. 46, no. 1-3, pp. 389-422, 2002.
[49] R. Tibshirani, "Regression shrinkage and selection via the lasso," Journal of the Royal Statistical Society: Series B (Methodological), vol. 58, no. 1, pp. 267-288, 1996.
[50] R. S. Olson and J. H. Moore, 'TPOT: A tree-based pipeline optimization tool for automating machine learning,' in *Workshop on Automatic Machine Learning*, pp. 66–74, Dec. 2016, PMLR.
[51] K. Pye, 'Loess,' *Progress in Physical Geography*, vol. 8, no. 2, pp. 176–217, 1984.
[52] D. J. Berndt and J. Clifford, 'Using dynamic time warping to find patterns in time series,' in *KDD Workshop*, vol. 10, no. 16, pp. 359–370, 1994.
[53] T. Rakthanmanon, et al., 'Searching and mining trillions of time series subsequences under dynamic time warping,' in *Proc. 18th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, pp. 262–270, 2012.
[54] M. G. Plessen, 'Integrated time series summarization and prediction algorithm and its application to COVID-19 data mining,' in *Proceedings of the 2020 IEEE International Conference on Big Data (Big Data)*, Dec. 2020, pp. 4945-4954.
[55] P. Schäfer, 'Scalable time series classification,' *Data Mining and Knowledge Discovery*, vol. 30, no. 5, pp. 1273-1298, 2016.
[56] Wu, H., Hu, T., Liu, Y., Zhou, H., Wang, J., and Long, M. Timesnet: Temporal 2d-variation modeling for general time series analysis. In The Eleventh International Conference on Learning Representations, 2023.
[57] Shengsheng Lin, Weiwei Lin, Wentai Wu, Feiyu Zhao, Ruichao Mo, and Haotong Zhang. Segrnn: Segment recurrent neural network for long-term time series forecasting. arXiv preprint arXiv:2308.11200, 2023.
[58] N. Sarangpure, V. Dhamde, A. Roge, J. Doye, S. Patle, and S. Tamboli, 'Automating the machine learning process using PyCaret and Streamlit,' in *Proc. 2023 2nd Int. Conf. Innovation in Technology (INOCON)*, pp. 1–5, Mar. 2023, IEEE.
[59] K. Cho, et al., 'Learning phrase representations using RNN encoder-decoder for statistical machine translation,' *arXiv preprint*, arXiv:1406.1078, 2014.
[60] M. Schuster and K. K. Paliwal, 'Bidirectional recurrent neural networks,' *IEEE Trans. on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.
[61] R. Caruana, et al., 'Ensemble selection from libraries of models,' in *Proc. 21st International Conference on Machine Learning*, 2004.
[62] L. Prokhorenkova, et al., 'CatBoost: Unbiased boosting with categorical features,' *Advances in Neural Information Processing Systems*, vol. 31, 2018.
[63] S. J. Rigatti, 'Random forest,' *Journal of Insurance Medicine*, vol. 47, no. 1, pp. 31–39, 2017.
[64] T. Chen and C. Guestrin, 'XGBoost: A scalable tree boosting system,' in *Proc. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.
[65] G. Ke, et al., 'LightGBM: A highly efficient gradient boosting decision tree,' *Advances in Neural Information Processing Systems*, vol. 30, 2017.
[66] Sridhar, S., Sanagavarapu, S., and Sanagavarapu, S., "Handling data imbalance in predictive maintenance for machines using SMOTE-based oversampling," 2021 13th International Conference on Computational Intelligence and Communication Networks (CICN). IEEE, 2021.
[67] I. Silva, G. Moody, D. J. Scott, L. A. Celi, and R. G. Mark, 'Predicting In-Hospital Mortality of ICU Patients: The PhysioNet/Computing in Cardiology Challenge 2012,' *Computing in Cardiology*, vol. 39, pp. 245–248, 2012. PMID: 24678516; PMCID: PMC3965265.
[68] S. Fahle, et al., 'Systematic review on machine learning (ML) methods for manufacturing processes,' *Procedia CIRP*, vol. 93, pp. 413–418, 2020.
[69] I. K. Nti, et al., 'Applications of artificial intelligence in engineering and manufacturing: A systematic review,' *Journal of Intelligent Manufacturing*, vol. 33, no. 6, pp. 1581–1601, 2022.
[70] S. W. Kim, et al., 'Recent advances of artificial intelligence in manufacturing industrial sectors: A review,' *International Journal of Precision Engineering and Manufacturing*, 2022.
[71] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, 'Convolutional sequence to sequence learning,' in *Proc. Int. Conf. on Machine Learning*, Jul. 2017, pp. 1243–1252.
[72] S. Mehta and M. Rastegari, 'MobileViT: light-weight, general-purpose, and mobile-friendly vision transformer,' *arXiv preprint arXiv:2110.02178*, 2021.

JAESEOK JANG has been a researcher at INTERX since 2021 and currently holds the position of Senior Researcher. His research interests lie in time series analysis, image classification, and ensemble learning, with a focus on developing robust machine learning models for real-world applications.

CHANYOUNG JUNG was born in Seoul, South Korea, in 1997. He earned a B.S. degree in Mathematics in 2022 and an M.S. degree in computer engineering and convergence engineering for intelligent drone at both from Sejong University, South Korea.

Since 2024, he has been working as a researcher employed by a data science group called INTERX. Since 2021, He is the author of more than five papers, and more than two inventions. His research interests include machine learning, predictive model, data analytics and autonomous manufacturing using machine learning and data analysis.

HAIL JUNG is currently an Assistant Professor with the Seoul National University of Science and Technology. He is also a founding member and the CTO of INTERX, a manufacturing AI solution provider. He has served ad-hoc referee for multiple academic journals, such as the Journal of Business Research.