Journal of Intelligent Learning Systems and Applications, 2026, 18(2), 105-122
https://www.scirp.org/journal/jilsa
ISSN Online: 2150-8410
ISSN Print: 2150-8402

Bottleneck Identification in Semiconductor Manufacturing: A Machine Learning Approach

Adar Kalir[1*], Alon Yaakobi[2], Sylvain Bouhnik[2], Einav Gilboa[2], Moshik Ohayon[2]

[1]Intel Corporation and Ruppin Academic Center, Emek Hefer, Israel
[2]Intel Corporation, Qiriat-Gat, Israel
Email: *adar.kalir@intel.com, alon.yaakobi@intel.com, sylvain.bouhnik@intel.com, einav.gilboa@intel.com, moshik.ohayon@intel.com

How to cite this paper: Kalir, A., Yaakobi, A., Bouhnik, S., Gilboa, E. and Ohayon, M. (2026) Bottleneck Identification in Semiconductor Manufacturing: A Machine Learning Approach. *Journal of Intelligent Learning Systems and Applications*, **18**, 105-122.
https://doi.org/10.4236/jilsa.2026.182007

Received: February 26, 2026
Accepted: May 12, 2026
Published: May 15, 2026

Abstract

The semiconductor industry is known for its complex production. Thousands of machines (tools) perform thousands of operations over a diverse range of products with re-entrant flows and shifting bottlenecks. Detecting and predicting these bottlenecks in real-time and promptly responding to them at the shop-floor level can make a big difference in terms of efficiently utilizing the high capital equipment and achieving operational excellence. So far, approaches to this problem have used traditional optimization methods. This paper pioneers the application of big data ML models to the problem, tested on industry data. We propose a dual-phased approach that leverages Machine Learning (ML) for this task. The first phase involves evaluating the most relevant production parameters (features) in the production line, along with their predefined target values, while the second phase enhances and refines these parameters at the segment level of the production line to reflect aspects of balancing and leveling. We show that the most effective ML model is attained using the XGBoost algorithm, which achieves 95% accuracy. This outcome enables a more precise classification of the dynamically shifting bottlenecks within the production line relative to other methods in practice, thereby allowing production staff better control and performance in the long run.

Keywords

Semiconductor Manufacturing, Production Line, Bottleneck Identification and Prediction, Machine Learning

1. Introduction

Semiconductor production lines are highly complex, consisting of thousands of

process steps, a diverse range of products, and re-entrant flows with vast amounts of data. As indicated by Ehm and Ponsignon [1], managing this high complexity of semiconductor manufacturing, combined with the extreme volatility of the market, is challenging and typically results in bottlenecks with significant productivity losses. Schmenner and Swink [2] proposed the Theory of Swift Even Flow (TSEF) to explain differences in factory productivity beyond explanations provided by microeconomic theories, and the first step to improving productivity in their TSEF is identifying bottlenecks correctly to remove or reduce their negative impact. According to Alavian *et al.* [3], up to 30% of productivity losses are attributed to inefficient bottleneck identification and management. There are two types of bottlenecks: tactical bottlenecks and strategic bottlenecks. Tactical bottlenecks are dynamic and change with material flow, while strategic bottlenecks persist over longer periods and are therefore denoted more appropriately as constraints. The identification of bottlenecks is challenging due to the complexity and stochasticity of the problem.

Currently, the process of identifying bottlenecks in semiconductor manufacturing relies heavily on the expertise of engineers and line (production) operators, making it time-consuming. At the beginning of each shift, and throughout the shift, operators make decisions on which tools to prioritize in terms of feeding WIP from upstream, giving them priority in metrology and inspection tools, giving them priority for maintenance resources, changing their drumbeats (*i.e.*, the output goal per layer in the process flow of the same toolset) in the scheduling system, etc.

Automating this process could save time and improve overall factory productivity and output. This work leverages Machine Learning (ML) methods to identify tactical (and dynamically shifting) bottlenecks. We focus on utilizing real production line data, such as the WIP (Work-In-Progress) levels and Cycle Time (CT) at each tool and each process step, tool availability, and other operational measurements. The proposed algorithm runs frequently (hourly), enabling shift personnel to manage the (dynamic) bottlenecks more effectively based on the frequent changes occurring in the production line due to the high variability nature of semiconductor manufacturing. In the long run, this enables a continuous, high-quality, rapid identification of bottlenecks, thereby mitigating productivity losses and increasing output.

This paper is organized as follows. The next section provides a literature review of papers dealing with bottleneck identification (or detection) as well as papers using ML models to address similar problems. Then, details of the methods and ML models used in the proposed approach are described, followed by results of an actual application of the model to industry that are presented and discussed. We conclude by discussing how bottleneck identification and prediction in real-time enhances productivity and overall performance.

2. Literature Review

The topic of identifying bottlenecks in production lines has long been a subject of

interest in both academic and industrial circles. One of the first definitions for a bottleneck and the distinction between a bottleneck and a constraint is due to Goldratt (1990) [4], where bottleneck is defined as a production (or system) resource that limits the production flow and causes WIP congestion in the short term. On the other hand, a constraint is a production (or system) resource that limits the output in the long term. Hence, bottlenecks are primarily characterized by the amount of WIP held by them, whereas constraints are characterized by both the amount of WIP held by them and their overall capacity, which is typically the lowest.

Theories such as the "Theory of Constraints" (ToC) [4] and "Swift Even Flow" [2] are among the conceptual frameworks that have been developed to address the issue. In recent years, there has been a noticeable increase in research publications on this issue [5]. The significance of bottlenecks in production lines, particularly their impact on output, has led many researchers to explore various solutions to mitigate the problems caused by these bottlenecks. A comprehensive review of theories, methods, and techniques for bottleneck identification and management in manufacturing systems in general is provided by Tang *et al.* [6] and a few years earlier by Ibidunmoye *et al.* [7]. They note that only by accurately identifying and predicting bottlenecks within a system is possible to avoid issues such as inefficient resource allocation and delivery delays or implementing measures to minimize these negative impacts. This is even more so under unstable production conditions, at which dynamic bottlenecks are likely to occur, making the identification and prediction of bottlenecks crucial. Similarly, in Kacar *et al.* [8], it is concluded that interactions between toolsets can be quite complex and that models focusing on a limited set of toolsets may give misleading estimates of system performance. Su *et al.* [9] deal with dynamic bottlenecks in their research and propose a dynamic bottleneck identification method (DBI-BS) that is based on effective buffers and machine states to identify bottlenecks accurately.

According to Fang *et al.* [10], bottleneck identification methods can be broadly categorized into two types: deterministic methods and statistical methods.

Deterministic methods for identifying bottlenecks are divided into simulation and analytical approaches. Simulation methods use real and historical data to simulate material flow and identify machines that hinder progress, such as the system proposed by Wedel *et al.* [11] using WIP data. Analytical methods are typically based on mathematical models such as Markov chains (e.g., Subramaniyan *et al.* [12]). Yan *et al.* [13] offer an analytical/empirical method for calculating Characteristic Curves (CCs) of throughput as a function of cycle time, indicating what is desirable in terms of system operation. However, the drawback of the analytical methods is that they rely on state matrices and specific formulas but often fail to account for the dynamic nature of real production lines and usually require stable conditions. In contrast, statistical methods are data-driven, utilizing historical data to generate models. Examples include algorithms by Li *et al.* [14] and Hopp and Spearman [15], which classify bottlenecks based on machine performance and availa-

bility. Cao *et al.* [16] integrate machine data with production variables using an analytical algorithm. Mönch and Zimmermann [17] present a computational study and results of a performance evaluation of a Shifting Bottleneck Heuristic (SBH) applied to complex multi-product job shops.

AI-based methods have also been attempted on the problem in recent years. Both supervised and unsupervised ML models are reported, e.g., the unsupervised model proposed by Subramaniyan *et al.* [18], to identify bottlenecks by analyzing activity durations through hierarchical cluster execution time-series and Dynamic Time Warping (DTW) image indices, classifying specific clusters as bottlenecks. Another example is the algorithm named Minerva by Thomas *et al.* [19] to address Job Shop Scheduling with jobs that are enqueued periodically. Minerva first finds the optimal resource scheduling for a target interval, based on a model-free reinforcement learning technique. Then, using an Artificial Neural Network (ANN) classifier, it identifies the constrained resources for each target interval. A relatively recent work by Subramaniyan *et al.* [20] proposes an active period-based data-driven algorithm to predict throughput bottlenecks in the production system from the large sets of machine data. To facilitate their prediction, an Auto-Regressive Integrated Moving Average (ARIMA) method is employed.

Unlike in [18], where unsupervised ML is proposed, we propose supervised ML models in this paper. For a survey of supervised ML approaches, the reader is referred to Muhammad and Yan [21], where they focus on the strengths of supervised learning.

In summary, many approaches to bottleneck identification have been proposed, with the more recent ones being based on AI/ML. Collectively, the literature on bottlenecks shows that a significant portion of it is dedicated to identifying bottlenecks, with fewer studies aimed at predicting or diagnosing the underlying causes of bottlenecks. In this paper, building on Subramaniyan *et al.* [20], we take their approach one step further and propose an application of supervised ML models for big data to the problem at hand, of bottlenecks' identification and prediction. We develop a dual-phased approach that leverages ML for this task in a complex semiconductor re-entrant production line, which is characterized by frequent bottleneck changes, and is responsive in real-time.

3. Method and Results

In this section, we present a Machine Learning (ML) based algorithm for identifying and predicting bottlenecks using real data from the production line. The methodology follows a widely accepted structure for developing ML solutions, comprising of five stages: 1) data collection; 2) pre-processing (including data verification and division into validation training sets); 3) comparison between several ML models and selecting the best model; 4) enhancement and refinement of the selected model; and 5) evaluation of the model on new data (*i.e.*, data collected in real-time, during production.)

The following sub-sections provide details on each step, including examples of

applying them in a real semiconductor production line, to demonstrate accurate, high-quality classification of bottlenecks.

3.1. Data Collection

As indicated earlier, the semiconductor production line consists of thousands of process steps, performed by various types of tools (machines) through a re-entrant flow and subject to all sorts of constraints, e.g., time limit constraints known as Critical Queue Times (CQTs). To manage this complexity, the production line is segmented, with each segment requiring specialized expertise and knowledge. For illustrative purposes, Figure 1 depicts a 12-day production line divided into three segments, each containing various tools, holding different amounts of WIP. Table 1 summarizes some of the input data retrieved from the Manufacturing Execution

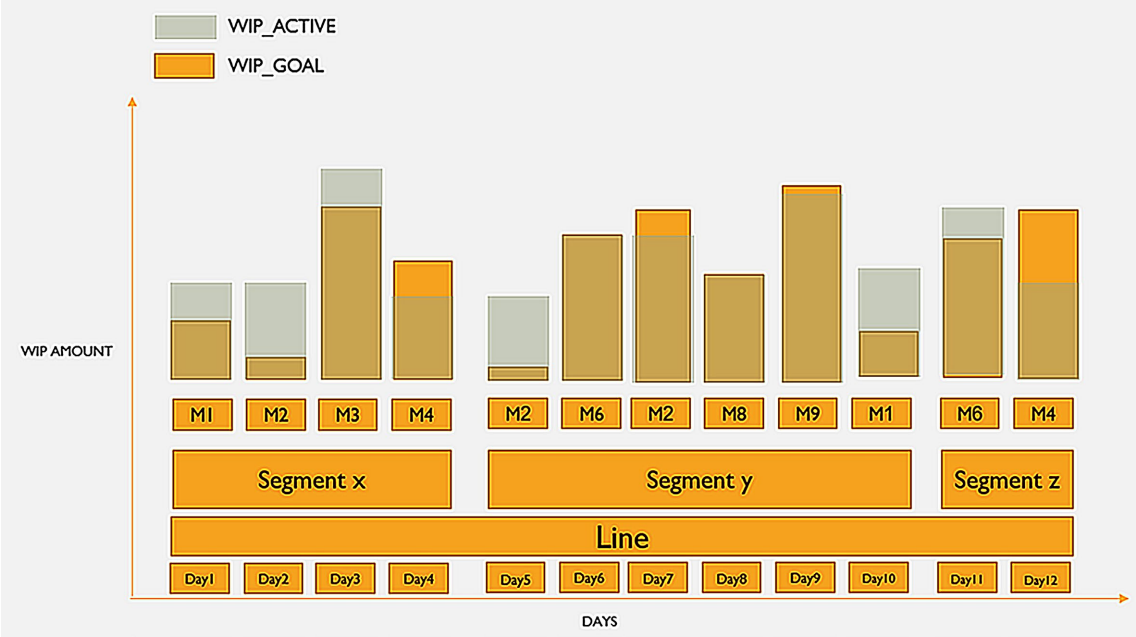

Figure 1. Illustration of a segmented production line.

Table 1. Input data from production.

Variable name	Explanation	Value
WEEK	ID of the week	Categorical
SHIFT DATE	Shift end and start date	Date
SHIFT NO.	ID number for the shift	Numeric
LAYER_NAME	The specific operation performed by a tool	Categorical
TOOL_LOOP	Is the tool part of a CQT (Y/N)	Categorical
TOOL_NAME	The name of the tool	Categorical
OPERATION NO.	The operation number performed by the tool	Numeric
WIP_ACTIVE	Actual amount of WIP at the tool (continuous throughout the shift)	Numeric

Continued

WIP_GOAL	Target amount of WIP at the tool	Numeric
CT_ACTUAL	Actual cycle time	Numeric
CT_GOAL	Target cycle time	Numeric
AVAILABILITY REQUIRED	The availability expected at each tool	Numeric
ACTUAL AVAILABILITY	The availability for each tool (continuous throughout the shift)	Numeric
OUTS GOAL	Required tool output	Numeric
OUTS ACTUAL	Actual output by tool	Numeric
UTILIZATION	The utilization of the tool (continuous throughout the shift)	Numeric

System (MES) for such a segmented production line. As can be seen from the table, there are 4 types of entries: time (indicated by week and shift); process information (operations in the process flow, tools performing the operations); process limitations (such as Critical Queue Times or CQTs); and operational information (e.g., WIP at each operation, cycle time at the operation vs. goal, output of each operation, etc.) Furthermore, a sample of the dataset is provided as a supplement to the paper in an EXCEL file. The file is comprised of 20 columns (or features), and 13,483 rows (or entries). The file can serve practitioners and researchers to further study the problem and other algorithms.

3.2. Pre-Processing Stage

The data processing and validation phase is crucial in developing an ML model, as it ensures the integrity and accuracy of the data used. Without proper validation, there is a risk of developing a model based on flawed or mis-representative data, echoing the adage "garbage in, garbage out". Since the ML model relies on data from various sources, data validation is crucial. "Exception" type data is cleaned to prevent biases in the model (e.g., abnormal shifts, defined as unplanned events such as production line stoppages, or planned events irrelevant to the analysis, such as machine experiments). Filtering these events is essential to avoid bias in the models. It is important to note that parameter values that were outliers in their distribution are not filtered out, as these outliers might indicate potential bottlenecks. As part of this stage, we have also utilized expert line operators to determine which toolsets were bottlenecks during each shift and classify them as such, so that the process of learning is supervised. Since different shifts had different operators with mixed opinions about which toolsets were bottlenecks during a shift, the classification, over 30 weeks of data, 14 shifts each week, has certainly captured well the variation and removed any randomness in the process. This was verified via applying the Cohen's kappa statistic, which measures inter-rater reliability for categorical (nominal) data, yielding a value of 0.73, whereas values of >0.6 generally indicate good agreement.

Feature engineering is another critical step in deciding which parameters to use and whether to create new ones based on existing ones to add meaningful insights.

For instance, parameters like WIP, output, cycle time, and availability, which are commonly used by line operators to identify bottlenecks, are manipulated to create new parameters, such as a new WIP parameter indicating whether the actual WIP exceeds the target WIP. It is noteworthy that "target WIP" is commonly used in semiconductor manufacturing for defining desirable WIP levels and distribution across the entire process flow. Target WIP is defined per operation in the process and the cumulative target WIP over all operations sums up to the desirable WIP in the manufacturing system. A bottleneck implies exceeding WIP_Goal at any operation, *i.e.*, the tool performing the operation becomes congested.

These new parameters (or features) help better capture underlying relationships between actual and target values in the model. Examples of such parameters are provided in Table 2. Then, correlated features are checked for, using the Spearman correlation test, which reveals anticipated correlations between variables such as CT_GOAL and CT_ACTUAL (0.89), or ACTUAL AVAILABILITY and AVAILABILITY REQUIRED and also between CT ACTUAL and WIP ACTIVE (0.75). These correlations were expected, given that more WIP typically leads to higher Cycle Times (CTs).

Finally, the data is divided into time-aware training, validation, and test sets to facilitate model development and evaluation. The training set is comprised of 70% of the data, the validation set contains 15% of the data, and the test set includes the remaining 15%. This division ensures that each subset maintains the same proportion of the two target values, "bottleneck" and "non-bottleneck", as seen in the overall dataset. A random split of the data is performed to eliminate any potential time or shift-related biases in the classification.

At the end of this step, the outcome is three high-quality data files suitable for training, validating, and testing the ML models. Table 3 provides a quantitative summary of the entire dataset used, reflecting its composition and distribution.

Table 2. New ratio indices created from the base indices.

Variable name	Explanation	Value
$\frac{\text{WIP\_ACTIVE}}{\text{WIP\_GOAL}}$	Actual amount of material in relation to the target	Integer
$\frac{\text{CT\_ACTIVE}}{\text{CT\_GOAL}}$	Actual cycle time relative to target	Float
$\frac{\text{ACTUAL\_AVAILABILITY}}{\text{REQUIRED\_AVAILABILITY}}$	Availability in practice in relation to the goal	Float
$\frac{\text{OUT\_ACTIVE}}{\text{OUT\_GOAL}}$	Actual amount of material in relation to the target	Float
$\frac{\text{WIP\_ACTIVE}}{\text{OUT\_ACTIVE}}$	The actual amount of material in relation to output	Float

Table 3. Summary of datasets (training, validation, and test).

Dataset	Total records	Bottleneck	Non-Bottleneck	%	Name
Models' comparison	123,939	6729	117,210	70%	Train set
Tuning models	26,558	1431	25,127	15%	Validation set
Test models	26,559	1470	25,089	15%	Test set

3.3. Comparison of ML Methods

In the previous step, high-quality data files are created, and parameters are selected for accurately classifying bottlenecks. The next step involves selecting and testing the best ML algorithm. Choosing the right model is critical for success, especially given the constraints of an unbalanced target variable and non-linear relationships. Moreover, the model must be explainable, allowing users to understand its decisions for WIP steering and scheduling around bottlenecks. Two algorithms were evaluated: Random Forest and Extreme Gradient Boosting (XGBoost). Both of them construct ensembles of decision trees but differ in how these trees are built and combined. The advantage of these algorithms of trees is that they are explainable, which is important for validation with production experts and managers, and yet they are also complex and dynamic in their ability to learn.

To determine the best model, each of them has been optimized and evaluated using the Synthetic Minority Over-sampling Technique with Edited Nearest Neighbors (SMOTENNs) and class weighting (using the class weight variable), to address the unbalanced dataset. An unbalanced dataset is one where the class proportions are skewed, with the majority class significantly outnumbering the minority class. For instance, in a binary classification problem with one hundred records, if one class has 90 observations and the other has 10, the dataset is considered unbalanced with a 90:10 ratio. This imbalance can lead to the "Accuracy Paradox", where accuracy metrics may appear excellent (e.g., 95%) but do not reflect the true performance of the model due to the skewed class distribution. To accurately compare the models and assess their performance, an appropriate metric has been devised based on Grandini *et al.* [22]. There, several metrics for dealing with unbalanced datasets are reviewed and the F1 score is recommended. The F1 score (see Equation (1)) is the harmonic mean of Recall (Equation (2)) and Precision (Equation (3)). Recall measures the percentage of correctly identified positive observations out of all actual positives, while Precision measures the percentage of correct positive predictions out of all positive predictions. The F1 score provides a balanced measure that considers both Precision and Recall. This approach ensures that our model evaluation accurately reflects the classification performance, avoiding bias towards the majority class.

$$\text{F1 score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{1}$$

$$\text{Recall} = \frac{\text{True positive}}{\text{True positive} + \text{Falce negative}} \quad (2)$$

$$\text{Precision} = \frac{\text{True positive}}{\text{True positive} + \text{Falce positive}} \quad (3)$$

In our evaluation of the models, we have also included a fourth measure, the Cohen's kappa value as described in Vieira *et al.* [23], which is a performance measure for feature selection.

As mentioned earlier, SMOTENN has been deployed to address the imbalance. This method focuses on manipulating the data by duplicating instances of the minority class ("bottleneck"), *i.e.*, creating synthetic records using the k-Nearest Neighbor (KNN) model from the minority class and then omitting overly similar records to maintain the distribution of each parameter. More on this can be found in Gavrylenko *et al.* [24].

Next, some information about the application of the two algorithms, namely Random Forest and Extreme Gradient Boosting (XGBoost), is provided.

Random Forest (RF) is a common ML algorithm that combines multiple decision trees to arrive at a single result. The trees are created by random sampling of parameters and the data (using the bagging method), such that each tree is not optimal (shallow), but when many diverse and different trees are connected together into a "forest of trees", an improved model emerges, which, based on majority vote, makes a balanced decision. Another advantage of the method is avoiding overfitting by using parameters that are used to control the learning process ("Hyperparameters"), such as maximum depth of trees, minimum number of observations for splitting, etc.

To reach a good classification with the RF model, the hyperparameters of this model were evaluated by performing several runs with different values (*i.e.*, cross-validation with a K of 10). The hyperparameters evaluated were the number of trees in each iteration, the accuracy index, the minimum samples for splitting, and the maximum random parameters for each tree. The best result was obtained with the following values:

Number of trees in each iteration = 40,

Accuracy index: entropy,

Minimum samples for splitting = 7,

Maximum parameters in each tree = 50% of all parameters = 3 parameters.

The second algorithm, Extreme Gradient Boosting (XGBoost), is a highly regarded ML algorithm that is considered state-of-the-art. "Gradient Boosting" refers to improving a single model by combining it with several other models, thus producing a robust, high-quality integrated model. The general idea is to enhance model performance by minimizing classification errors.

In Gradient Boosting, shallow decision trees are trained iteratively. Each iteration uses the error residuals from the previous model to build a better decision tree in the next iteration. The final classification is a weighted sum of all the tree classifications built during the iterations. As in the case of the RF model, in this model as

well, the hyperparameters of this model were evaluated in order to reach a good classification. Again, several runs with different values were performed with the following hyperparameters: the number of trees in each iteration; the learning rate (L), which is a critical parameter to avoid a situation of over-fitting; and the maximum random parameters for each tree. In addition, since this method is more sensitive to over-adjustment, the early stopping principle was applied as well. This principle stops the model from reaching the point of minimum error and, on the other hand, prevents a situation of over-fitting and thus improves the accuracy of the model. The results of the experimentation with the hyperparameters are depicted in Figure 2. As can be observed from the charts, the best result was obtained with the following values:

Number of trees in each iteration = 100,

Learning rate (L) = 0.1,

Maximum random parameters in each tree = 50% of all parameters = 3 parameters,

Early stopping = 35.

For the early stopping hyperparameter, the two charts on the left depict the error (nlogloss) as a function of the number of trees, for both the training set and the validation set. Based on this function, a value of 35 repetitions has been selected for early stopping to avoid overfitting to the data.

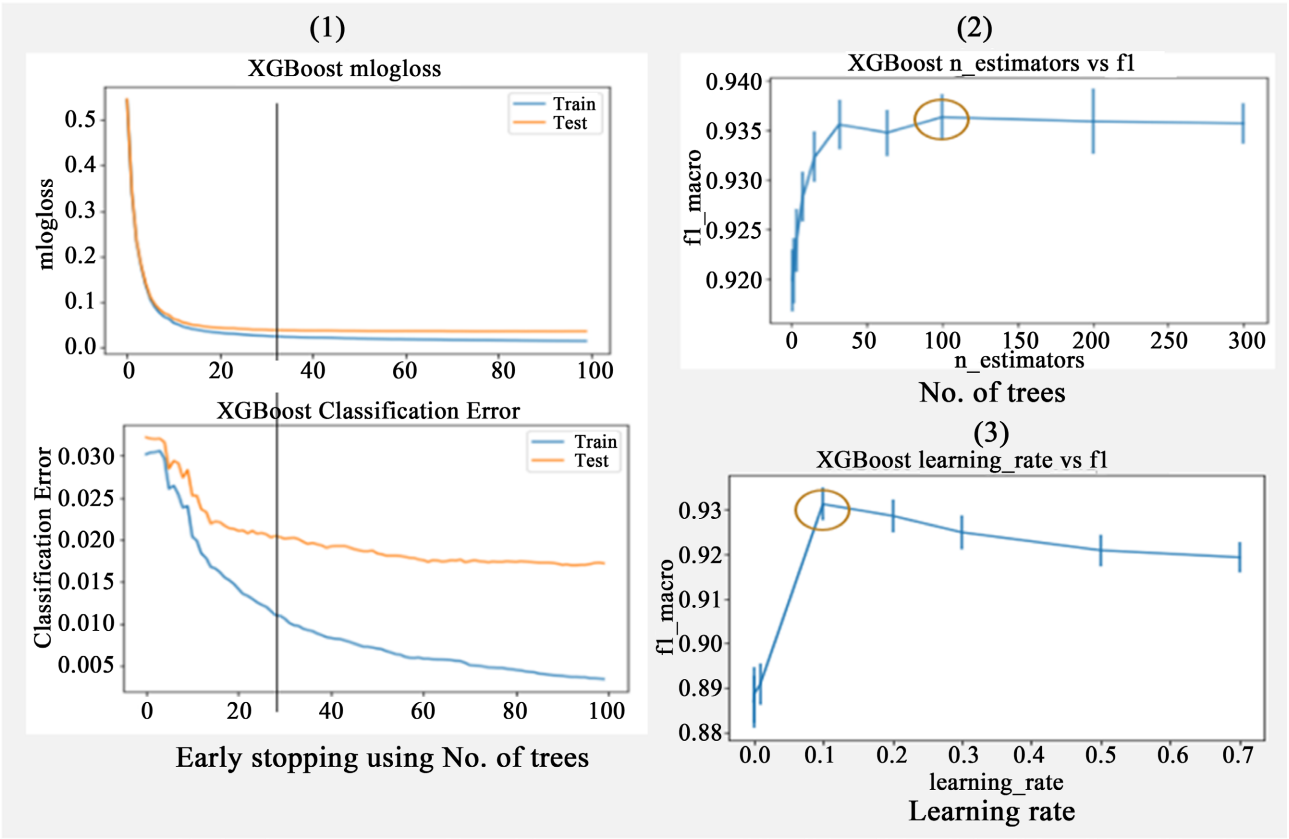

Figure 2. Experimentation with hyperparameters in XGBoost.

Table 4. F1 score on the test set for RF and XGBoost.

Method	Model	F1 score	p-value
Imbalance (using class weight)	RF	88.23%	0.0019
	XGBoost	89.33%	0.0005
Oversampling (using SMOTENN)	RF	80.96%	0.0320
	XGBoost	80.74%	0.0350

After optimizing both models for their hyperparameters, they were executed with these hyperparameter values on the validation set. Cross-validation with K = 30 was applied and the F1 score of each run was recorded. Lastly, T-test was used to check significance of the difference between the average results of the two models. Table 4 contains a summary of the comparison between the two models. As can be seen, the best model is XGBoost with class weight for imbalance, achieving an F1 score of 89.33% with a p-value of 0.0005, indicating that it is statistically significantly better than the others.

3.4. Model Enhancement and Refinement

After selecting the best model in the previous step and achieving a high accuracy of 89.33% on the F1 score, the next step is to further improve the model to better identify dynamic bottlenecks. Feature importance on second model (RF) was not checked, as the model accuracy was too low. Enhancing an existing model is achieved via the following actions: 1) obtaining additional data, 2) using and/or creating additional variables that can help the model learn better, and 3) fine-tuning the hyperparameter that controls the learning of the model.

In this case, the entire dataset has been reused and re-distributed into new training, validation, and test sets (the division was done randomly). In re-evaluating significant parameters for the model (feature importance) based on the new dataset, the parameter $\frac{\text{WIP\_ACTIVE}}{\text{WIP\_GOAL}}$ has now emerged as highly important (see Figure 3) and, as Figure 4 suggests, it is also highly correlated with the CT_Actual/CT_Goal.

Given that a WIP-related parameter has now shown high importance, several new parameters that pertain to the flow and not to a specific tool were added to the model to enhance bottleneck identification. These new parameters include the calculated mean, median, maximum, minimum, and standard deviation of the baseline parameters. From these, the absolute delta between each tool and the segment indices was devised, per the following equations (shown here for WIP_ACTIVE only):

$$\text{mean} = \left| \frac{\sum_1^N \text{WIP\_ACTIVE}_m}{N} - \frac{\text{WIP}_{\text{ACTIVE}_m}}{\text{WIP}_{\text{GOAL}_m}} \right| \mid \forall m \in \text{segment} \tag{4}$$

$$\text{mod} = \left| \text{Mod}\left(\text{WIP}_{\text{ACTIVE}}\right)_{\text{segment}} - \frac{\text{WIP}_{\text{ACTIVE}_m}}{\text{WIP}_{\text{GOAL}_m}} \right| \mid \forall m \in \text{segment} \tag{5}$$

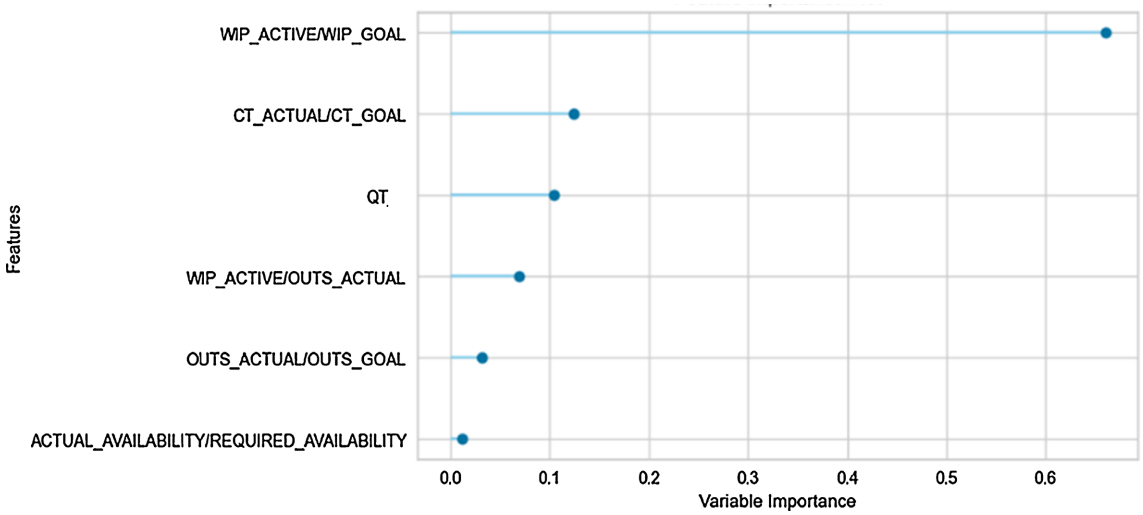

Figure 3. Feature importance of the parameters for XGBoost training-set.

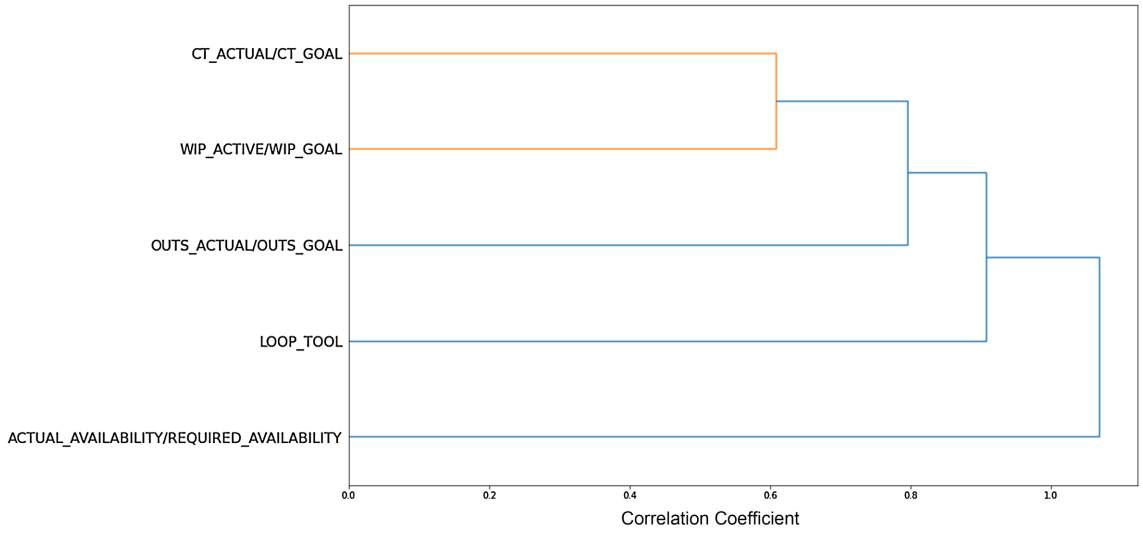

Figure 4. Spearman linkage dendrogram showing parameters with similar correlation by using hierarchical clustering.

$$\max = \left| \max\left(\mathrm{WIP}_{\mathrm{ACTIVE}}\right)_{\mathrm{segment}} - \frac{\mathrm{WIP}_{\mathrm{ACTIVE}_m}}{\mathrm{WIP}_{\mathrm{GOAL}_m}} \right| \mid \forall m \in \mathrm{segment} \tag{6}$$

$$\min = \left| \min\left(\mathrm{WIP}_{\mathrm{ACTIVE}}\right)_{\mathrm{segment}} - \frac{\mathrm{WIP}_{\mathrm{ACTIVE}_m}}{\mathrm{WIP}_{\mathrm{GOAL}_m}} \right| \mid \forall m \in \mathrm{segment} \tag{7}$$

$$\mathrm{std} = \sqrt{\frac{\sum_1^N \left(\frac{\mathrm{WIP}_{\mathrm{ACTIVE}_m}}{\mathrm{WIP}_{\mathrm{GOAL}_m}} \right) - \sum_1^N \mathrm{WIP}_m}{\mathrm{N}}} \mid \forall m \in \mathrm{segment} \tag{8}$$

where m is a tool in the segment and N is the number of tools in the segment.

Additionally, as part of this step, parameters that have shown similar behavior to other parameters based on hierarchical clustering and Spearman's correlation analysis were dropped, so the fine-tuning of parameters works both ways. Follow-

ing the above changes, re-evaluation of the feature importance was performed (see Figure 5), highlighting the importance of the Mod(WIP_ACTIVE) parameter which has been squared to increase its relative impact (see Equation (9)), based on feedback from production experts, and capture those tools deviating significantly from the segment's average values and more likely to become bottlenecks.

$$\left(\mathrm{Mod}\left(\mathrm{WIP}_{\mathrm{ACTIVE}}\right)-\frac{\mathrm{WIP}_{\mathrm{ACTIVE}_m}}{\mathrm{WIP}_{\mathrm{GOAL}_m}}\right)^2 \mid \forall m \in \text{segment} \tag{9}$$

Lastly, a model re-run was performed to re-optimize the hyperparameters and check the F1 score and the accuracy metrics of the refined model. This is summarized in Table 5 for using the training, validation, and test sets, respectively. As highlighted in Table 5, the F1 score has improved through the model enhancement and refinement step from 89.33% at the beginning of this step to 95.58% with the refined model.

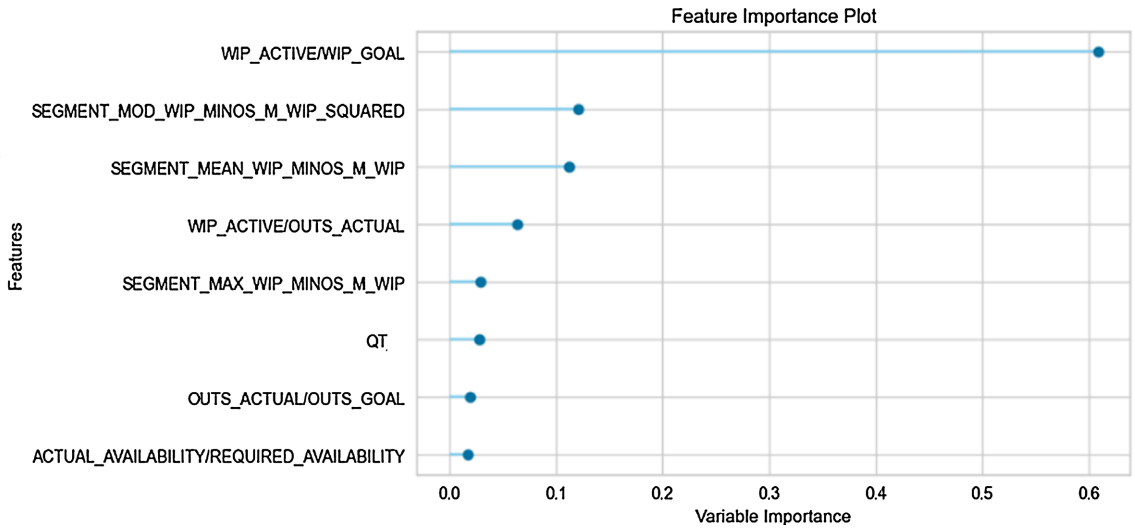

Figure 5. Feature importance of the parameters for XGBoost post-refinement.

Table 5. F1 score, recall, precision, and kappa on the test set for RF and XGBoost.

Metric	Training	Validation	Test
F1	96.84%	95.97%	95.58%
Recall	94.56%	93.58%	94.44%
Precision	99.43%	98.68%	97.21%
Kappa	93.68%	91.94%	91.58%
Std			0.0049

3.5. Model Evaluation

The last step in the methodology for developing an ML solution is the evaluation of the model on new data, particularly data collected in real-time during production. To that end, real-time data from a semiconductor production facility over a

period of 100 production shifts of 12 hours each, containing 177,228 records in total, was used.

The outcome of the model addresses the key question of what the current and future bottlenecks are, and how they can be improved. In comparing the outcome of the model with the existing processes and systems for identifying bottlenecks by production personnel, several observations can be outlined, and they are discussed next.

The first observation is that while production personnel classify bottlenecks at the toolset level (*i.e.*, tool group), the ML model classifies bottlenecks at the individual tool level, reflecting the (known but ignored) fact that not all tools within the same toolset are identical. For example, certain tools may be restricted from certain operations owing to yield concern. Such a restriction is dynamic and may change over time.

The second observation is that the production staff spends much of its time during the shift reacting to changes in the production line and their implications, such as changes in availability or changes in incoming WIP to toolsets. Conversely, the ML model automates this process and not only frees up production resources for exception management but also does a better job in predicting the impacts of the changes through the shift and subsequent shifts.

Another important observation is with respect to existing literature to date on bottleneck identification and management. Most of the work to date has been focused on machine uptime and downtime (availability data). In contrast, the ML model at hand clearly shows that this is insufficient. Additional data regarding the process flow, the WIP and WIP flow through the segments, the capacity (e.g., required availability) and more are helpful in better predicting bottlenecks and behaviors over time. Specifically, in our comparison, we have observed that the cycle time parameter is not important for classifying bottlenecks, while the WIP parameter is. This is a surprising observation that can be explained by the nature of these two parameters. While WIP indicates real-time status at the tool and operation level, cycle time is a lagging indicator that is measured only after a lot has completed processing. In other words, real-time data is more important than past data.

Lastly, an observation regarding the enhancement and refinement of the ML model based on real data can be made. That is, the additional parameters that were added to the model have contributed significantly to its outcome, particularly the parameters that compare the delta to segment averages. All in all, after a sufficient period of deploying the ML solution in parallel to the existing solution, a 4% improvement in production line productivity and output has been observed. This was measured over a sufficiently large time window of three months (13 weeks) before and after the implementation of the new model, while no major changes occurred in the operation (*i.e.*, similar volumes and product mixture).

Furthermore, as shown in **Figure 6**, a significant reduction of 83% in cycle time has been recorded at specific operations post the ML implementation, partially thanks to improved response to operation openness (qualification) following the

ML classification. The ML classification has also demonstrated outstanding capability to dismiss fault predictions of bottlenecks by human (production personnel) assessment, as can be seen in Table 6. The ML model has accurately predicted 18% of non-bottlenecks, which the human mis-classified as bottlenecks (39 cases out of a total of 219 cases).

The implementation of the model, at the shop floor control and execution, has been gradual since it introduced a big change in the mode of operation. First, it was introduced only as a bottleneck identification and prediction recommendation system, via a user interface for the operators throughout the shift. Operators could have overridden the recommendation of the model, and the model would use the corrected classification as part of its learning. This phase has helped remove fear and gain trust and confidence in the model's ability to accurately predict bottlenecks. Next, an iterative process between operators and system developers took place to refine the parameters and improve the accuracy. These two phases took several months to complete. In the last phase of the implementation, the model was embedded in the shop floor Manufacturing Execution System (MES) and its recommendations became decisions for execution. In this phase, the model is executed once a shift, at the beginning of the shift, based on real-time data available at the end of the previous shift.

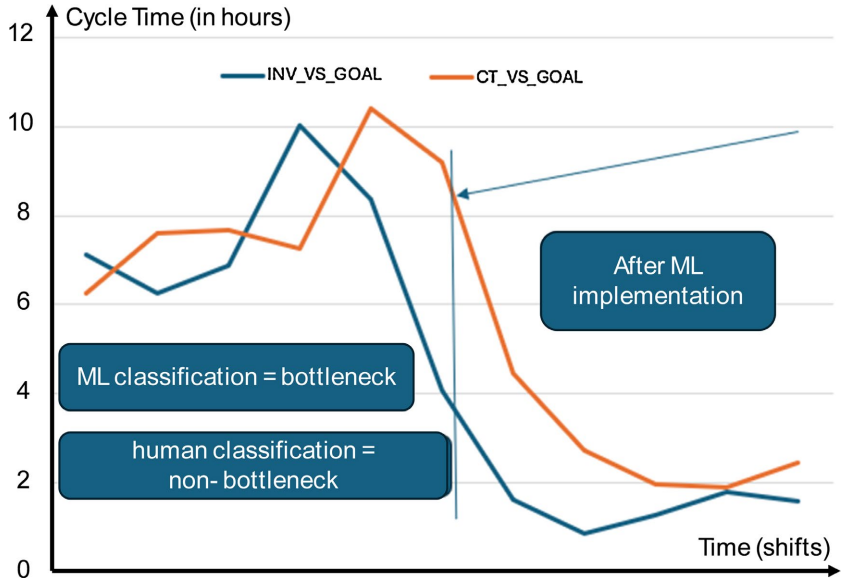

Figure 6. CT reduction after ML implementation.

Table 6. ML model bottleneck prediction accuracy.

		Model	
		Non-bottleneck	Bottleneck
Human	Non-bottleneck	180	93
	Bottleneck	39	86

4. Summary

Detecting bottlenecks in real-time in a semiconductor production line is a challenging task. Nonetheless, it is likely the most important task in a production line since bottlenecks dictate the rate of production and output. In this paper, we have developed a dual-phased Machine Learning (ML) model for this. The first phase involves evaluating the most relevant production parameters for the prediction, along with their predefined target values. In the second phase, these parameters are further enhanced and refined to improve the accuracy of the prediction via the introduction of segment-based parameters, which are devised from the baseline parameters. We have shown that the ML solution with the XGBoost algorithm achieves 95% accuracy on real-time data from a production line facility and improves manufacturing productivity by 4% over the long run. This result is enabled by a precise classification of the dynamically shifting bottlenecks within the production line relative to other existing methods in practice. Furthermore, on top of the ML model's productivity improvement, it is also less time-consuming (production personnel labor savings), faster, and more understandable.

With respect to the two ML algorithms evaluated, although they have had similar F1 scores for accuracy initially (RF at 88.23% and XGBoost at 89.33%), the XGBoost also performs in a manner that is considered "smarter", building trees which minimize the error in each iteration, while the RF model builds trees as different as possible and then classifies them according to the "majority opinion" rule.

This work has several important contributions at the theoretical and practical levels. At the theoretical level, we have devised a set of new parameters to be used for identifying bottlenecks. These parameters have not been proposed to date, and our work shows that they are important to accurately detect bottlenecks. At the practical level, we have demonstrated the benefit of employing ML for a foundational task in production lines. This paves the way for future similar customized solutions using ML for other production systems and settings.

As typically in the case of ML implementation, further work can be done in several directions. First, in terms of the inputs (features), there may be other parameter manipulations that can further improve the prediction. Second, to test the ML solution in different production settings and environments. And third, to leverage the ML solution for automating the decision-making process, which affects bottlenecks, e.g., the timing for planned maintenance of the tools.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Ehm, H. and Ponsignon, T. (2012) Future Research Directions for Mastering End-to-End Semiconductor Supply Chains. 2012 *IEEE International Conference on Automation Science and Engineering* (*CASE*), Seoul, 20-24 August 2012, 641-645.

https://doi.org/10.1109/coase.2012.6386323

[2] Schmenner, R.W. and Swink, M.L. (1998) On Theory in Operations Management. *Journal of Operations Management*, **17**, 97-113. https://doi.org/10.1016/s0272-6963(98)00028-x

[3] Alavian, P., Eun, Y., Meerkov, S.M. and Zhang, L. (2019) Smart Production Systems: Automating Decision-Making in Manufacturing Environment. *International Journal of Production Research*, **58**, 828-845. https://doi.org/10.1080/00207543.2019.1600765

[4] Goldratt, E.M. (1990) Theory of Constraints. North River, 159 p.

[5] Mahmoodi, E., Fathi, M. and Ghobakhloo, M. (2022) The Impact of Industry 4.0 on Bottleneck Analysis in Production and Manufacturing: Current Trends and Future Perspectives. *Computers & Industrial Engineering*, **174**, Article ID: 108801. https://doi.org/10.1016/j.cie.2022.108801

[6] Tang, J., Dai, Z., Jiang, W., Wu, X., Zhuravkov, M.A., Xue, Z., *et al.* (2024) A Comprehensive Review of Theories, Methods, and Techniques for Bottleneck Identification and Management in Manufacturing Systems. *Applied Sciences*, **14**, Article 7712. https://doi.org/10.3390/app14177712

[7] Ibidunmoye, O., Hernández-Rodriguez, F. and Elmroth, E. (2015) Performance Anomaly Detection and Bottleneck Identification. *ACM Computing Surveys*, **48**, 1-35. https://doi.org/10.1145/2791120

[8] Kacar, N.B., Munch, L. and Uzsoy, R. (2018) Problem Reduction Approaches for Production Planning Using Clearing Functions. 2018 *IEEE* 14*th International Conference on Automation Science and Engineering* (*CASE*), Munich, 20-24 August 2018, 931-938. https://doi.org/10.1109/coase.2018.8560429

[9] Su, X., Lu, J., Chen, C., Yu, J. and Ji, W. (2022) Dynamic Bottleneck Identification of Manufacturing Resources in Complex Manufacturing System. *Applied Sciences*, **12**, Article 4195. https://doi.org/10.3390/app12094195

[10] Fang, W., Guo, Y., Liao, W., Huang, S., Yang, N. and Liu, J. (2020) A Parallel Gated Recurrent Units (P-Grus) Network for the Shifting Lateness Bottleneck Prediction in Make-to-Order Production System. *Computers & Industrial Engineering*, **140**, Article ID: 106246. https://doi.org/10.1016/j.cie.2019.106246

[11] Wedel, M., Noessler, P. and Metternich, J. (2016) Development of Bottleneck Detection Methods Allowing for an Effective Fault Repair Prioritization in Machining Lines of the Automobile Industry. *Production Engineering*, **10**, 329-336. https://doi.org/10.1007/s11740-016-0672-9

[12] Subramaniyan, M., Skoogh, A., Salomonsson, H., Bangalore, P. and Bokrantz, J. (2018) A Data-Driven Algorithm to Predict Throughput Bottlenecks in a Production System Based on Active Periods of the Machines. *Computers & Industrial Engineering*, **125**, 533-544. https://doi.org/10.1016/j.cie.2018.04.024

[13] Yan, C., Monch, L. and Meerkov, S.M. (2019) Characteristic Curves and Cycle Time Control of Re-Entrant Lines. *IEEE Transactions on Semiconductor Manufacturing*, **32**, 140-153. https://doi.org/10.1109/tsm.2019.2908721

[14] Li, L., Chang, Q. and Ni, J. (2009) Data Driven Bottleneck Detection of Manufacturing Systems. *International Journal of Production Research*, **47**, 5019-5036. https://doi.org/10.1080/00207540701881860

[15] Hopp, W.J. and Spearman, M.L. (2011) Factory Physics: Foundations of Manufacturing Management. Waveland Press.

[16] Cao, Z., Deng, J., Liu, M. and Wang, Y. (2012) Bottleneck Prediction Method Based on

Improved Adaptive Network-Based Fuzzy Inference System (ANFIS) in Semiconductor Manufacturing System. *Chinese Journal of Chemical Engineering*, **20**, 1081-1088. https://doi.org/10.1016/s1004-9541(12)60590-4

[17] Mönch, L. and Zimmermann, J. (2010) A Computational Study of a Shifting Bottleneck Heuristic for Multi-Product Complex Job Shops. *Production Planning & Control*, **22**, 25-40. https://doi.org/10.1080/09537287.2010.490015

[18] Subramaniyan, M., Skoogh, A., Muhammad, A.S., Bokrantz, J., Johansson, B. and Roser, C. (2020) A Generic Hierarchical Clustering Approach for Detecting Bottlenecks in Manufacturing. *Journal of Manufacturing Systems*, **55**, 143-158. https://doi.org/10.1016/j.jmsy.2020.02.011

[19] Thomas, T.E., Koo, J., Chaterji, S. and Bagchi, S. (2018) Minerva: A Reinforcement Learning-Based Technique for Optimal Scheduling and Bottleneck Detection in Distributed Factory Operations. 2018 10*th International Conference on Communication Systems & Networks* (*COMSNETS*), Bengaluru, 3-7 January 2018, 129-136. https://doi.org/10.1109/comsnets.2018.8328189

[20] Subramaniyan, M., Skoogh, A., Bokrantz, J., Sheikh, M.A., Thürer, M. and Chang, Q. (2021) Artificial Intelligence for Throughput Bottleneck Analysis—State-of-the-Art and Future Directions. *Journal of Manufacturing Systems*, **60**, 734-751. https://doi.org/10.1016/j.jmsy.2021.07.021

[21] Muhammad, I. and Yan, Z. (2015) Supervised Machine Learning Approaches: A Survey. *ICTACT Journal on Soft Computing*, **5**, 946-952.

[22] Grandini, M., Bagli, E. and Visani, G. (2020) Metrics for Multi-Class Classification: An Overview. arXiv: 2008.05756.

[23] Vieira, S.M., Kaymak, U. and Sousa, J.M.C. (2010) Cohen's Kappa Coefficient as a Performance Measure for Feature Selection. *International Conference on Fuzzy Systems*, Barcelona, 18-23 July 2010, 1-8. https://doi.org/10.1109/fuzzy.2010.5584447

[24] Gavrylenko, S., Vladislav, Z. and Khatsko, N. (2023) Methods for Improving the Quality of Classification on Imbalanced Data. 2023 *IEEE* 4*th KhPI Week on Advanced Technology* (*KhPIWeek*), Kharkiv, 2-6 October 2023, 1-5. https://doi.org/10.1109/khpiweek61412.2023.10312879