Reports in Mechanical Engineering
Vol. 6, No. 1, 2025, pp. 27-45
ISSN: 2683-5894, DOI: https://doi.org/10.31181/rme444

Evaluating the Performance of Serial Production Lines Using Tree-Based Machine Learning Methods

Feras Mabrouke[1], Basel Al-Sahili[1], Ramiz Assaf[1*], Abdalmuttaleb Al-Sartawi[2], Salem Aljazzar[3], Siraj Zahran[3], Mohammad Kanan[3*]

[1]Department of Industrial and Mechanical Engineering, Faculty of Engineering, An-Najah National University, Nablus, Palestine
[2]Accounting, Finance and Banking Department, Ahlia University, Bahrain
[3]Industrial Engineering Department, Jeddah College of Engineering, University of Business and Technology, Jeddah 21448, Kingdom of Saudi Arabia

Article Info

Article history:

Received December 19, 2024
Revised March 18, 2025
Accepted May 21, 2025

Keywords:

Serial Production Line,
Tree-based Models,
Machine Learning,
Multioutput Regression,
Simulation, Synchronous,
Buffered Systems,
Operational Efficiency,
Performance Evaluation

ABSTRACT

This study investigates the efficiency of serial production lines using advanced tree-based machine learning algorithms, including Random Forest (RF), XGBoost, LightGBM, and CatBoost. By analyzing various performance indicators such as throughput, machine reliability, and processing time, we construct predictive models to evaluate production line behavior. The comparative results demonstrate that LightGBM outperforms the other models in prediction accuracy and computational speed. These findings suggest that machine learning techniques, particularly ensemble tree models, can significantly enhance decision-making and operational performance in manufacturing systems.

Corresponding Author:

Ramiz Assaf
Department of Industrial and Mechanical Engineering, Faculty of Engineering, An-Najah National University, Nablus, Palestine
Email: ramizassaf@najah.edu

1. Introduction

Serial production lines form the backbone of modern manufacturing industries, facilitating the efficient mass production of goods through a sequence of interconnected processes. The design and operation of serial production lines are essential to achieving optimal productivity, cost-efficiency, and product quality in industries (Hon, 2005),as they represent a tightly coupled sequence of processing stations where workpieces are processed sequentially, this sequential nature introduces complexities due to the interdependence of stations and the potential for blocking and starving (Liu et al., 1996). In an increasingly competitive global market, the ability to optimize serial production lines has become a key differentiator for manufacturers. Inefficiencies in these systems can lead to bottlenecks, excessive idle times, and resource wastage, directly impacting production costs and delivery timelines. Conversely, well-optimized production lines can significantly enhance throughput, minimize downtime, and reduce operational costs, providing a competitive edge to manufacturers. Traditional methods for evaluating serial production lines have included analytical approaches, which often involve simplifying assumptions and can struggle with real-world complexities, particularly for systems with more than two machines (Kim et al., 2024). Most of these analytical models assume idealized machine behavior, often ignoring critical interactions such as buffer blockage and starvation, which significantly affect system performance in practical settings. Our approach addresses this limitation by modeling machine states along with buffer dynamics, capturing the real-world effects of upstream and downstream

dependencies. This research focuses on utilizing machine learning algorithms to analyze and optimize serial production lines by addressing these complexities. By building multioutput regression using tree-based models. These models are powerful tools for analyzing structured data due to their ability to model complex, nonlinear relationships by using decision Trees as their basis (Hu & Li, 2022). This work aims to predict key metrics namely, average throughput, throughput variability, and buffer levels under different production scenarios. These predictions can help identify potential bottlenecks, enhance decision-making, and improve system reliability. The tree-based regression models take the parameters of the production system as inputs and compute scalar values representing key performance metrics, such as average throughput, throughput variability, and buffer levels. Since each model has unique strengths in handling structured data and capturing nonlinear relationships, this work employs and compares several algorithms: Random Forest (RF), Extra Trees (ET), Gradient boosting decision tree (GBDT), XG Boost, and Light GBM. These models are chosen for their proven performance in predictive tasks and their ability to handle the complexities of serial production line evaluation. This article focuses on evaluating synchronous serial production lines, where machine efficiency is represented as a general Markovian system (Assaf et al., 2014) and the system have a finite buffer size. Tree-based regression models are developed and trained for a fixed number of machines within the production line. The training and testing datasets are generated using discrete-event simulations conducted under steady-state conditions. The production system parameters for these simulations are randomly sampled from predefined ranges to ensure a comprehensive analysis of system behavior. The main contributions of this article are as follows:

1. Development of a robust framework for evaluating synchronous serial production lines, incorporating machine efficiency and finite buffer size.
2. Implementation and comparison of advanced tree-based regression models to evaluate the performance of serial production line.
3. Establishing a methodology that can be extended to evaluate production lines with varying configurations and parameters, ensuring applicability across a wide range of manufacturing systems.
4. Publishing the data sets used for training and testing the models to enable future researchers to validate and extend the findings of this study.

2. Related Work

The evaluation of serial production lines is crucial for optimizing performance, identifying bottlenecks, and improving both quantity and quality of production. Two prominent methods used for this purpose are decomposition-based methods and aggregation-based methods. Below, provide a summarized overview of these approaches and their applications in analyzing and improving serial production lines (Wang et al., 2014). Decomposition-based methods simplify the analysis of long production lines by dividing them into smaller segments, typically consisting of two machines and a buffer in between (Le Bihan & Dallery, 2000). These smaller segments are analyzed using Markov chains to model the Work-In-Progress (WIP) level and the operational states (up or down) of the machines. The insights gained from these short lines are then used to evaluate the performance of the entire production line (Wang et al., 2014). Key developments in decomposition-based methods include the work of Gershwin and Berman (Gershwin & Berman, 1981), who developed probability balance equations for two-machine short lines, providing closed-form solutions for steady-state probabilities and enabling the evaluation of average production rate and WIP levels. Tolio et al. extended the analysis to lines with multiple failure modes (Colledani & Tolio, 2005), deriving analytical performance evaluations. Tan and Gershwin proposed a general Markovian model for continuous-flow systems with multiple quality-related states (Tan & Gershwin, 2011). Building on this foundation, Gershwin introduced the first decomposition-based algorithm for evaluating long lines with identical machine processing times and finite buffer sizes (Gershwin, 1987). Aggregation-based methods evaluate production lines by recursively combining segments either backward or forward until convergence. This approach has been proven effective for analyzing and improving production line performance (Wang et al., 2014). Notable advancements in aggregation-based methods include Yishu Bai's development of a novel algorithm designed to evaluate production lines with unreliable machines and limited buffer capacities (Bai et al., 2021). This algorithm improves upon traditional methods by more accurately modeling machine-buffer interactions, demonstrating consistent convergence in numerical evaluations, and achieving greater precision in key performance metrics. Additionally. Kuo et al. identified bottlenecks by evaluating machine blockage and starvation probabilities without calculating production rate sensitivity (Kuo et al., 1996). Chiang et al. proposed new techniques to identify downtime, uptime, and cycle-time bottlenecks through blockage and starvation probabilities (Chiang et al., 2000). The Equivalent Machine Method (EMM) offers another analytical approach, simplifying production lines by replacing entire subsystems with equivalent machines that

preserve input-output behavior. While these classical methods are useful for benchmarking, their reliance on simplifying assumptions often makes them unsuitable for complex, large-scale, or highly variable systems. Evaluating the average throughput of a production line is essential for understanding and optimizing its efficiency. Throughput reflects the number of finished products generated over a specific period, providing a direct measure of the system's productivity (Tan & Gershwin, 2009). By analyzing throughput, managers can identify bottlenecks that limit production capacity, assess the impact of reliability and processing times, and implement strategies to streamline operations. For instance, Gershwin highlights that understanding throughput dynamics allows for the development of decomposition algorithms to evaluate and improve production performance (Gershwin, 1987), particularly in lines with limited buffers and variable processing times. Tracking the standard deviation of a production line's periodic output provides insights into its efficiency and consistency, it helps identify trends, assess the impact of process changes, and benchmark performance across different lines or facilities. A decrease in standard deviation following process improvements indicates enhanced consistency and effectiveness, while an increase suggests potential issues requiring further investigation. Additionally, comparing standard deviations across production lines allows businesses to identify best practices and target areas for improvement, ultimately driving overall productivity and efficiency. In a production line, machines are connected in series, separated by buffers, and parts are processed sequentially. Buffers are essential for reducing idle time caused by machine failures and variable processing times, thereby minimizing starving and blocking. However, buffer allocation is often constrained by physical and budgetary limitations, requiring efficient allocation to optimize production line performance (Demir & Koyuncuoğlu, 2021). Evaluating buffer size and capacity is critical for mitigating variability, ensuring smooth production flow, minimizing delays, and enhancing throughput. Proper sizing balances the trade-off between performance and cost, as excessive buffers raise inventory costs, while insufficient buffers lead to frequent disruptions (Li et al., 2009). Effective evaluation is key to achieving optimal efficiency and system reliability. The most relevant work related to our study is the research conducted by kim S, where a neural network model was developed and trained using synthetic data generated through simulations of serial production lines (Kim et al., 2024). Their model effectively approximated the average throughput, with machine efficiency modeled using a Weibull distribution characterized by high variability in machine uptime and downtime. In contrast, our research adopts an exponential distribution for machine efficiency. A key aspect of our study involves comparing the performance of tree-based regression models with their neural network approach to highlight the differences and evaluate the effectiveness of the two methodologies. Recent literature highlights the potential of AI-driven optimization and hybrid modeling in manufacturing. For example, (Samee et al., 2022) discusses metaheuristic-based optimization for complex industrial tasks, while (El-Kenawy et al., 2022) focuses on machine learning for operational efficiency modeling. Hybrid frameworks combining machine learning with simulation, as seen in, (El-kenawy et al., 2021) have shown strong performance in predictive maintenance and control. More recently, (Elshabrawy, 2025) and (Khaled & Singla, 2025) explore multi-objective optimization and AI-based predictive frameworks for enhancing production system efficiency. Boosting and bagging are two ensemble learning techniques commonly used to improve the performance of tree-based models. Bagging (Bootstrap Aggregating) focuses on reducing variance by training multiple decision trees independently on different bootstrap samples of the data and averaging their predictions. Models like Extra trees and Random Forest leverage bagging to enhance robustness and prevent overfitting. In contrast, boosting aims to reduce bias by sequentially building trees where each subsequent tree corrects the errors of its predecessor. Models like Gradient Boosting, Light GBM, and XG Boost utilize boosting to create a strong predictive model by combining weak learners in a weighted manner. While bagging excels in reducing overfitting, boosting is more effective in handling underfitting, but may be prone to overfitting if not carefully regularized (Dey, 2024; P. Bruce et al., 2020). Tree-based models have demonstrated significant potential across various industries, offering interpretable and efficient solutions for response surface modeling. These models frequently outperform traditional methods such as linear regression and support vector machines in capturing complex relationships and nonlinear interaction (Dasari et al., 2015), six different tree-based models were applied in a study to Hardgrove Grindability Index (HGI) (Chen et al., 2025), a critical metric for assessing coal grindability, with implications for improving production efficiency and economic profitability in the coal industry. Similarly, another study employed three tree-based models to analyze the elevated temperatures on the tensile strength of composite materials (Machello et al., 2024). In the manufacturing sector, tree-based models have been widely adopted for predictive tasks. For example, a Random Forest model was developed to diagnose the causes of variations in a production line (Wong & Chu, 2021). Another study employed Random Forest model to predict tool wear (Wu et al., 2017), while Light Gradient Boosting was utilized to forecast setup time in electronic production lines (Chen et al., 2024). These applications underscore the ability of tree-based models in addressing diverse challenges in industrial and manufacturing environments.

3. Serial Production Line Description

3.1 Overview of the Serial Production Line

A serial production line consists of M machines and M − 1 buffers, as shown in figure. 1, where machines are connected through buffers. Each machine operates independently and is modeled using a single-failure-mode approach, represented by a Markov chain matrix that characterizes the machine's uptime and downtime, following a geometric distribution. Buffers serve as storage for work-in-progress (WIP) items between machines and have finite capacities. All machines in the production line are assumed to operate synchronously, meaning they share identical cycle times. A more detailed description of the system can be found in (Assaf et al., 2014).

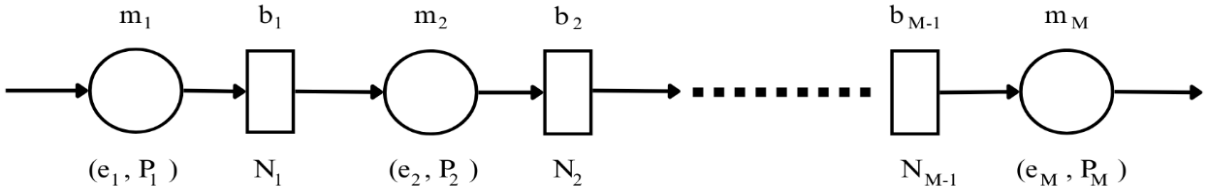


Figure 1: Serial production line with M machines (circles) and M − 1 buffers (rectangles).

System Description:

1. Number of Machines: M machines are included in the system.
2. Number of Buffers: M−1buffers are present between the machines, each with a finite capacity (N).
3. Processing Cycle Time: The system operates with a constant processing cycle time, scaled so that one cycle equals one time unit.
4. Machine States: Each machine can be in one of two states: Operational (Up): The machine is functioning normally, Failure (Down): The machine is non-operational.
5. Transition Probability Matrix (Markov chain): the transitions between the two states of a machine are represented using a probability matrix (P) which follows a geometric distribution, a visual representation of the transitions between statis shown in figure. 2. The matrix is defined as: $P = \begin{bmatrix} p_{U,U} & p_{U,D} \\ p_{D,U} & p_{D,D} \end{bmatrix}$

In the matrix $P_{U,U}$ represent the Probability of remaining in an up state, $P_{U,D}$ is the Probability of transitioning from up to down state, $P_{D,U}$ Probability of recovery from a down state to up, and $P_{D,D}$ is the Probability of remaining in a down state.

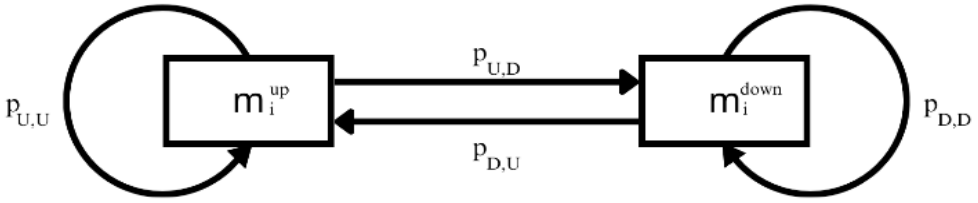


Figure 2: Transitions Between Statis

6. Machine Efficiency: Each machine in the system has a unique efficiency calculated using equation 1:

$$e = \frac{p_{D,U}}{p_{D,U} + p_{U,D}}$$

Therefore, a serial line with M machines have 6M -1 parameters. We denote this parameter set as an input.
Input X = (P_1 P_2 … P_M e_1 e_2 … e_M N_1 N_2 … N_{M-1}).

The evaluation of serial production lines focuses on three primary metrics. The first is average throughput, which measures the mean number of items produced per unit of time and serves as an indicator of overall production efficiency. The second is the standard deviation of throughput, which quantifies the variability in the number of items produced per unit of time, providing insight into the consistency and reliability of the production process. Lastly, buffer levels represent the average number of items stored in each buffer, highlighting the system's storage utilization and the flow balance between machines.

3.2 Data Generating

The data for the models are obtained through discrete-event simulations of various serial production lines.

The parameter of these lines is selected from the following ranges: M ∈ {2, 3, 4}, N ∈ [2, 50], e ∈ [0.65, 0.99], and $P_{D,\,U}$ ∈ [0.1, 0.4]. A total of 30 000 production lines is simulated for generating the training data (10 000 lines, respectively, for each M = 2, 3, 4). The detailed procedure for determining line parameters for each M is as follows:

1) Randomly select the efficiency $\mathbf{e_i}$ from the range [0.65, 0.99], i = 1, 2, …, M.
2) Randomly select the buffer size N_j from the range [2, 50], j=1, 2, …, M-1.
3) Randomly select the $P_{D,U,i}$ from the range [0.1, 0.4]
4) Calculate $P_{D,D,i}$ from $1 = P_{D,D,i} + P_{D,U,i}$
5) Calculate $P_{U,D,i}$ from equation 1.
6) Calculate $P_{U,U,i}$ from $1 = P_{U,D,i} + P_{U,U,i}$

The simulation procedure begins by initializing the system parameters, including the number of machines, buffer capacities, transition probabilities, and machine efficiency. The production line is then simulated for each set of parameters, during which machine states, buffer levels, and transitions are recorded. Throughput metrics and buffer utilization are calculated, and results are aggregated across multiple replicates to ensure statistically significant outputs. For each selected production line, ten discrete-event simulations are performed. The output variables for a given parameter set are obtained by averaging the results from these ten simulations. Each simulation starts with empty buffers, and the throughput gradually increases from zero during a transient period. To ensure accurate steady-state measurements, the transient period is excluded when calculating the output variables.

4. Methods

4.1 Multioutput Regression

Multioutput regression is a type of supervised machine learning designed to predict multiple target variables simultaneously. Unlike traditional regression, which focuses on predicting a single target variable based on input features, multioutput regression extends this capability to handle multiple numerical targets at once. This approach is particularly useful in real-world scenarios where multiple dependent variables need to be modeled together, especially when they are interrelated. In this research, we implement a multioutput regression model using the Scikit-learn library ("MultiOutputRegressor, Scikit-learn," 2024). In Scikit-learn, multioutput regression is achieved by leveraging wrappers that extend single-output regression models to handle multiple targets as illustrated in figure. 3. The wrapper trains a separate regression model for each target variable while sharing the same set of input features. This approach allows any regression model in Scikit-learn to be adapted for multioutput tasks, enabling flexibility and consistency across different types of models.

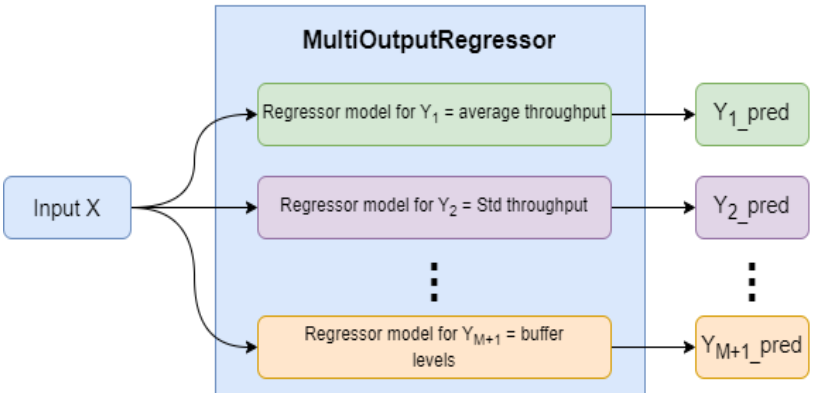


Figure 3: Multioutput Regressor

4.2 Tree-based Machine Learning Models

Tree-based models are widely used due to their simplicity, interpretability, and ability to handle both numerical and categorical data. Decision trees, initially developed by Leo Breiman and others in 1984, are effective methods for classification and regression tasks. One of their key advantages is interpretability, as the tree structure provides a clear visualization of decision-making processes. Additionally, decision trees are resilient to missing values and less sensitive to outliers compared to other machine learning algorithms. However, advanced tree-based models, improve performance by combining multiple decision trees using techniques like bagging or boosting. While these models, are highly effective, they are often considered "black-box" methods due to their complexity. As a result, they require careful hyperparameter tuning to optimize performance and reduce the risk of overfitting (P. Bruce et al., 2020).

4.2.1 Random Forest (RF)

The random forest algorithm, proposed by L. Breiman in 2001 (Breiman, 2001) is an ensemble learning method that builds multiple decision trees during training and combines their predictions to improve accuracy and robustness. It works by using bagging technique, where different subsets of the training data are randomly sampled with replacement for each tree. Additionally, at each split in the tree, only a random subset of features is considered to ensure diversity among the trees. The final prediction is made by averaging the outputs for regression or voting for classification. This approach reduces overfitting, improves generalization, and handles high-dimensional data effectively, making it a versatile and reliable model (Biau & Scornet, 2016; Chen & Ishwaran, 2012).

4.2.2 Extra Trees (ET)

Extra Trees, or Extremely Randomized Trees, is an ensemble method similar to Random Forest but introduces more randomness during tree construction. While Random Forest selects the best split from a subset of features, Extra Trees splits randomly within that subset, reducing the reliance on the data's fine details (Galelli & Castelletti, 2013). This increased randomness results in simpler trees and often improves generalization. Like Random Forest, it aggregates predictions from multiple trees through averaging for regression or voting for classification. Extra Trees is computationally faster than Random Forest because it doesn't compute the optimal split, making it suitable for large datasets with high-dimensional features (Geurts et al., 2006; Wehenkel et al., 2006).

4.2.3 Gradient Boosting Decision Tree (GBDT)

Gradient Boosting is an iterative ensemble method that builds decision trees sequentially, where each tree corrects the errors of its predecessor. It minimizes a loss function, such as mean squared error, by training each tree to predict the residuals from the previous iteration. Unlike Random Forest, Gradient Boosting emphasizes the contribution of poorly predicted instances, making it highly effective for capturing complex patterns. While it achieves high accuracy, it can be sensitive to overfitting and requires careful hyperparameter tuning. Gradient Boosting works well for structured data and is widely used in competitive machine learning tasks (Li et al., 2018; Natekin & Knoll, 2013).

4.2.4 XGBoost

XGBoost (Extreme Gradient Boosting) is an optimized implementation of Gradient Boosting that introduces enhancements to improve speed, scalability, and accuracy (Kavlakoglu & Russi, 2024). It uses techniques such as regularization to reduce overfitting, parallel computation for faster training, and efficient handling of missing data. XGBoost builds trees iteratively, with each tree focusing on minimizing the residual errors of the previous trees. It also supports advanced features like column sampling, monotonic constraints, and early stopping during training. These features make XGBoost a powerful and flexible tool for both regression and classification tasks, widely favored in machine learning competitions and real-world applications (Chen & Guestrin, 2016).

4.2.5 LightGBM

LightGBM (Light Gradient Boosting Machine) is a highly efficient gradient boosting decision tree algorithm developed by Microsoft for machine learning tasks (Ke et al., 2017),which is another implementation of Gradient Boosting designed for efficiency and scalability on large datasets. Unlike traditional Gradient Boosting, LightGBM grows tree leaf-wise, splitting the leaf with the highest loss reduction, resulting in deeper and more efficient trees. It also uses histogram-based binning, reducing memory usage and speeding up computation. LightGBM supports categorical features directly and handles sparse data efficiently. While it excels in speed and accuracy, it can overfit on small datasets without proper tuning. LightGBM is widely used in applications requiring fast training and prediction times, such as real-time systems and large-scale datasets (NA, 2024).

4.3 Hyperparameter Optimization and Evaluation Metrics

For hyperparameter optimization, we will use Bayesian Optimization (BO), a probabilistic model-based method that efficiently tunes hyperparameters in machine learning models. Unlike traditional methods like grid search or random search, BO builds a surrogate model, typically using Gaussian Processes, to approximate the objective function. It balances exploration, by testing new hyperparameter combinations, and exploitation, by focusing on promising areas of the hyperparameter space. This method is particularly effective for optimizing expensive black-box functions (Nguyen, 2019), often achieving better results with fewer evaluations compared to default settings or random search (Galuzzi et al., 2020). BO-based approaches provide an efficient and reliable solution for hyperparameter tuning across various machine learning tasks (Wu et al., 2019). In this study, hyperparameter optimization was performed using Bayesian Optimization with the Scikit-Optimize library, aiming to minimize the Mean Absolute Percentage Error (MAPE) during cross-validation. The hyperparameter search space was defined

based on prior experience and relevant literature (Chen et al., 2025) , as shown in Table 1, A total of 50 iterations were conducted for each model, with each iteration evaluated using 5-fold cross-validation to ensure result reliability and stability. The accuracy of the models is evaluated by calculating the mean absolute percentage error (MAPE) and the coefficient of determination (r^2) between the predicted outputs obtained from the tree-based models and the simulated outputs using the following formula:

$$MAPE = \frac{1}{n} \times \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \quad , \quad r^2 = \frac{(y_i - \hat{y}_i)^2}{(y_i - \bar{y})^2}$$

In the formulas, n represents the total number of samples, y_i is the i-th actual value, $\hat{y}_i$ is the i-th predicted value and $\bar{y}$ is the mean for the samples data.

4.4 Model Construction

The construction process of the tree-based models for the generated data is as follows, with multioutput regression applied to each model to predict multiple output variables simultaneously (see figure 4.):

1. For each production line, the dataset is randomly divided into a training set and a test set in a 4:1 ratio.
2. The hyperparameter search space is defined for five tree-based machine learning models. The mean absolute percentage error (MAPE) from cross-validation is used as the fitness metric. To ensure a balance between training time and model evaluation stability, 5-fold cross-validation is employed. In each iteration, the training set is divided into five equal parts: four are used for training, and one is used for validation, with each fold serving as the validation set in turn. The average MAPE across the five folds is calculated as the fitness value. Bayesian Optimization, implemented via Scikit-Optimize, is used to perform 50 iterations to identify the optimal hyperparameter combination for each model.
3. Each machine learning model is trained using its optimized hyperparameters with multioutput regression applied to predict all target variables simultaneously. Predictions are made on both the training and test sets, and evaluation metrics are computed. The predictive performance of the models is compared to determine the most effective model for predicting the output variables.

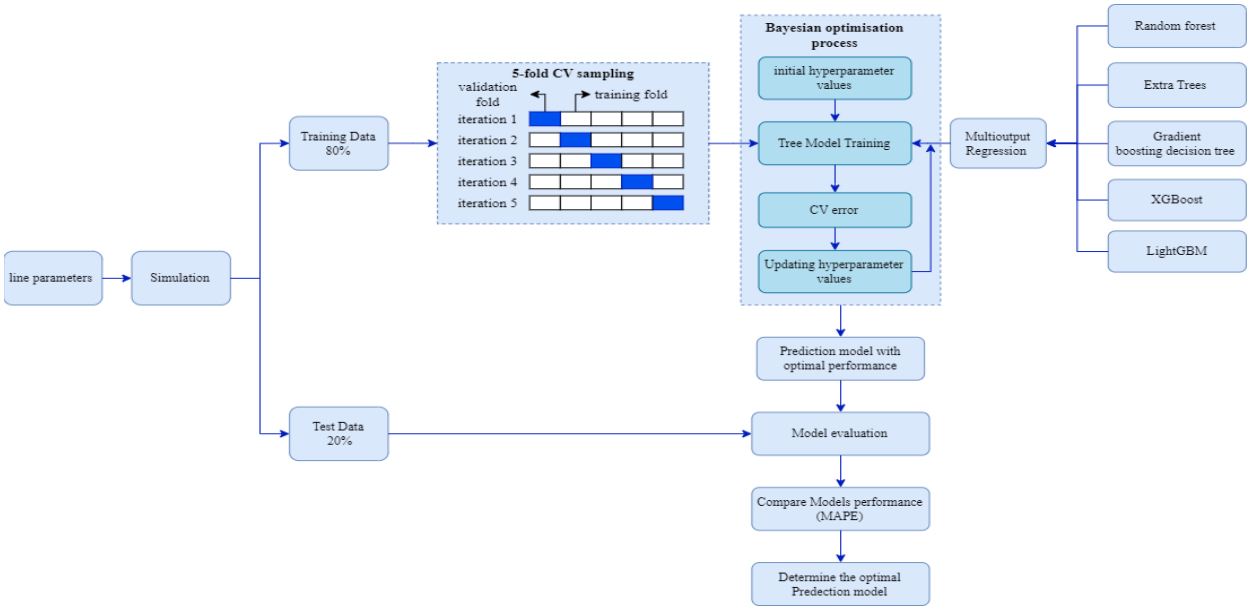


Figure 4: Models Construction Process

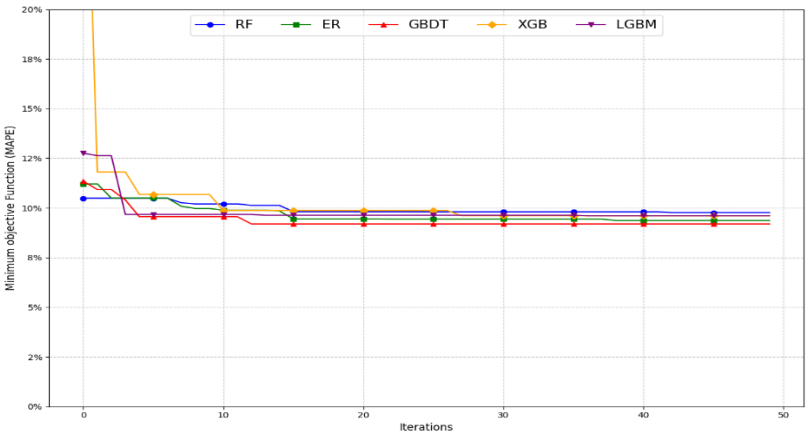
5. Result and Discussions

5.1 ML Models Comparisons

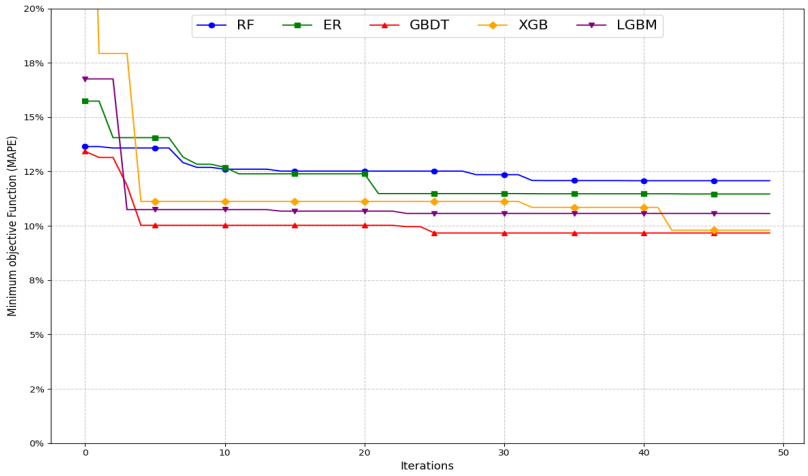
5.1.1 Prediction Models Results

As mentioned earlier, the hyperparameters of the machine learning algorithms were first needed to be tuned in order to obtain optimal prediction models. figure. 5 shows the minimum values of the objective functions, which imply

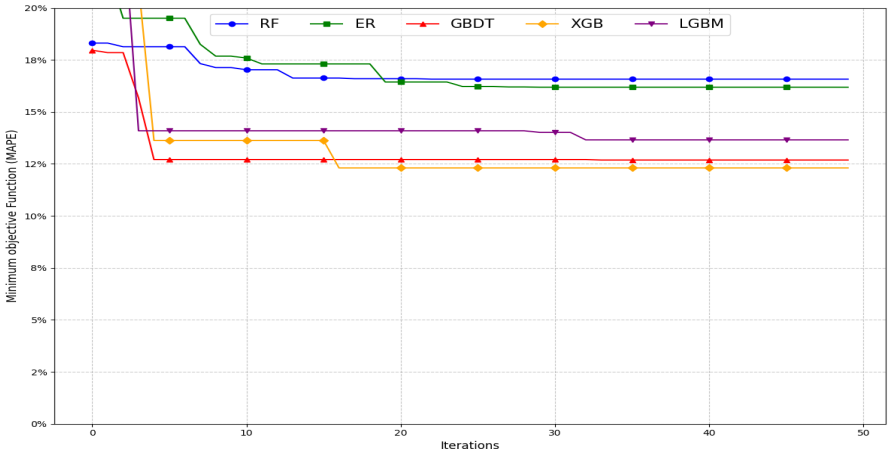
the loss in the cross-validated fit of RF, GBDT, ET, XGBOOST, LightGMB models during 50 steps of Bayesian optimization.



(a)



(b)



(c)

Figure 5: Hyperparameter Tuning Process of the Machine Learning Models: (a) M = 2, (b) M = 3, (c) M = 4.

Best points for each model for each system represent the set of values in the hyperparameter space by which the objective function converges to a minimum value. It is seen that the minimum CV error ($MAPE_{MOR}$) for most of the

models for each system is after 30 iterations, we noticed that the Rf and ET have the highest error and the GBTD, XGBOOST, and LightGMB have the lowest error with not a very significant difference. The list of hyperparameters with their optimal values obtained at the best points is presented in Table 1, for each prediction model for each system. In Table 2, The n_estimators parameter defines the number of trees in the ensemble, balancing model complexity and computational cost. Max_depth controls the maximum depth of each tree, preventing overfitting by limiting the growth of the trees. The min_samples_split parameter specifies the minimum number of samples required to split an internal node, while min_samples_leaf ensures that each leaf node contains at least a minimum number of samples, aiding in regularization. For boosting algorithms, the learning_rate governs the contribution of each tree to the overall model, with smaller values typically requiring more trees to achieve optimal performance. The subsample parameter determines the fraction of training samples used for fitting each tree, introducing randomness to reduce overfitting. Additionally, colsample_bytree defines the fraction of features sampled for each tree, promoting diversity among trees. Parameters such as min_child_weight and min_child_samples further control tree growth by specifying the minimum sum of instance weights and minimum number of samples in child nodes, respectively, ensuring the model is robust to noise and small datasets.

Table 1: Hyperparameter Search Ranges for ML Models.

M	Hyperparameter	Range	ML Model				
			RF	GBDT	ET	XGBoost	LightGMB
2	n_estimators	50-500	500	433	500	285	491
	max_depth	1-50	13	30	15	50	45
	min_samples_split	2-10	2	10	2	-	-
	min_samples_leaf	1-10	1	10	1	-	-
	learning_rate	0.01-0.5	-	0.021	-	0.18	0.062
	subsample	0.5-1	-	0.52	-	1	0.92
	colsample_bytree	0.5-1	-	-	-	1	0.94
	min_child_weight	1-10	-	-	-	9	10
	min_child_samples	1-20	-	-	-	-	1
3	n_estimators	50-500	500	500	500	463	500
	max_depth	1-50	31	23	50	39	10
	min_samples_split	2-10	2	10	2	-	-
	min_samples_leaf	1-10	1	10	1	-	-
	learning_rate	0.01-0.5	-	0.012	-	0.055	0.071
	subsample	0.5-1	-	0.5	-	0.735	0.5
	colsample_bytree	0.5-1	-	-	-	1	1
	min_child_weight	1-10	-	-	-	9	7
	min_child_samples	1-20	-	-	-	-	6
4	n_estimators	50-500	500	249	408	347	481
	max_depth	1-50	36	20	45	22	37
	min_samples_split	2-10	2	7	2	-	-
	min_samples_leaf	1-10	1	8	1	-	-
	learning_rate	0.01-0.5	-	0.054	-	0.039	0.079
	subsample	0.5-1	-	0.614	-	0.651	0.771
	colsample_bytree	0.5-1	-	-	-	1	1
	min_child_weight	1-10	-	-	-	10	5
	min_child_samples	1-20	-	-	-	-	6

In order to further analyze the multioutput regression performance of the tree-based models, two metric values, mean absolute percentage error (MAPE) and goodness of fit (r^2) were used in this study. The formulations of these metrices are presented as:

$$MAPE_{MOR} = \frac{MAPE_{avg-throughput} + MAPE_{std-throughput} + \sum_{j=1}^{M-1} MAPE_j}{M+1}$$

$$r^2_{MOR} = \frac{r^2_{avg-throughput} + r^2_{std-throughput} + \sum_{j=1}^{M-1} r_j^2}{M+1}$$

Figure 6. Shows the MAPE (test) for the five different tree-based models. The yellow bars represent the MAPE when using the RF model, the blue bars correspond to the ET model, and the orange, green and grey bars represent

the results from the GBDT, XGBOOST and LightGMB models, respectively. Across all models, it is observed that the MAPE increases with an increasing number of machines in the production line. The GBDT, XGBOOST and LightGMB model demonstrates the best performance among the considered models, while the RF and ET model exhibits the poorest.

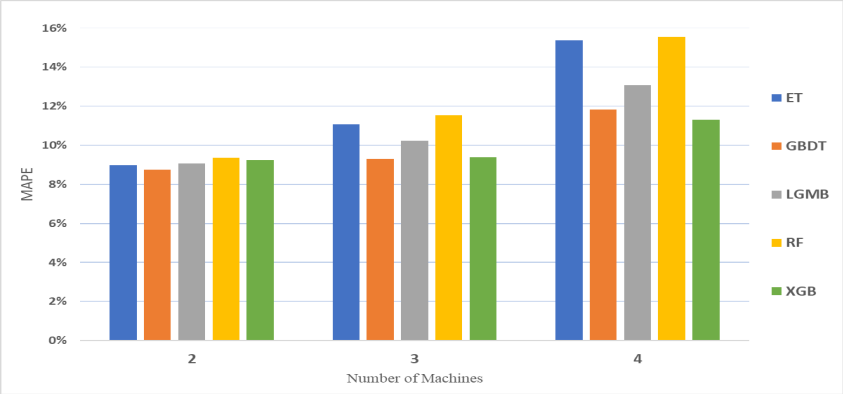


Figure 5: Multioutput Regressors MAPE (Test) With Five Different Tree-Based Models

Table 2 presents a comparison of the prediction performance of tree-based models, evaluated on training and test data across three multioutput regression tasks (2-machine system, 3-machine system, and 4-machine system). The average results across all systems were considered for better comparison. For the training data, ET achieves the highest r^2 score of 0.987 with a MAPE of 0.012, indicating strong fitting to the training set. The other models show slightly lower r^2 values, ranging from 0.942 (LightGBM) to 0.978 (GBDT), with MAPE values between 0.029 and 0.059. These results suggest a good balance between accuracy and generalization during training. On the test data, ET, XGBoost, and GBDT achieve the highest r^2 values of 0.826, 0.826, and 0.825, respectively, with moderate MAPE values ranging from 0.100 to 0.118. RF has the lowest r^2 (0.822) and the highest test MAPE (0.121), suggesting slightly less accurate predictions compared to the other models. LightGBM and GBDT exhibit similar performance with r^2 values of 0.825 and MAPE values of 0.100 and 0.108, respectively. Overall, the boosting models (GBDT, LightGBM, and XGBoost) perform competitively on test data, achieving relatively lower errors while maintaining good generalization. However, all models show some degree of overfitting, as indicated by the gap between training and test performance. Further analysis is needed to assess individual output performance across different regression tasks to better understand the observed discrepancies in the metrics.

Table 2: (a) Multioutput Regressors Performance Metric Values of Different Tree-Based Models

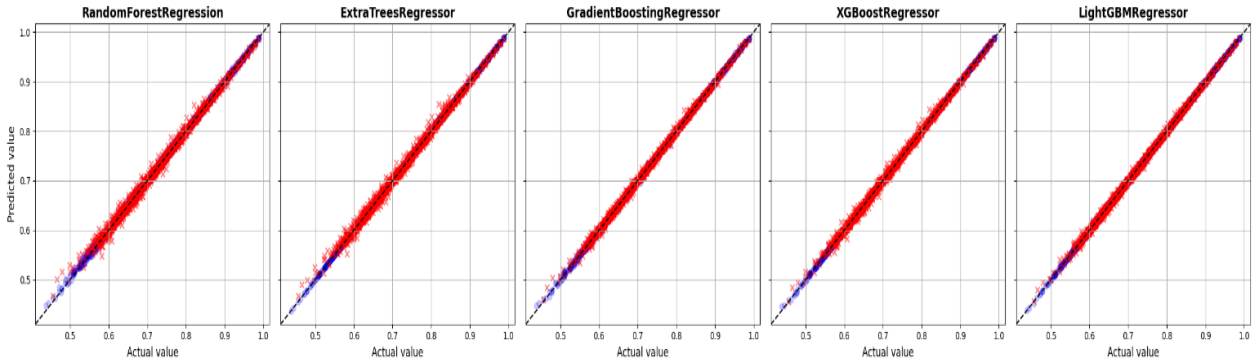
M	ML Model	Training Data		Test Data	
		MAPE	r^2	MAPE	r^2
2	ET	0.036	0.961	0.090	0.809
	GBDT	0.035	0.967	0.087	0.796
	LightGMB	0.063	0.911	0.091	0.804
	RF	0.053	0.934	0.094	0.806
	XGBoost	0.049	0.924	0.093	0.795
3	ET	0	1	0.111	0.832
	GBDT	0.040	0.968	0.093	0.834
	LightGMB	0.055	0.950	0.102	0.829
	RF	0.043	0.977	0.115	0.827
	XGBoost	0.029	0.962	0.094	0.833
4	ET	0	1	0.154	0.837
	GBDT	0.013	0.999	0.118	0.844
	LightGMB	0.060	0.964	0.131	0.843
	RF	0.058	0.977	0.155	0.833
	XGBoost	0.031	0.972	0.113	0.849
Average Result Across All Systems	ET	0.012	0.987	0.118	0.826
	GBDT	0.029	0.978	0.100	0.825
	LightGMB	0.059	0.942	0.108	0.825
	RF	0.051	0.963	0.121	0.822
	XGBoost	0.036	0.953	0.100	0.826

5.1.2 Average Throughput Result

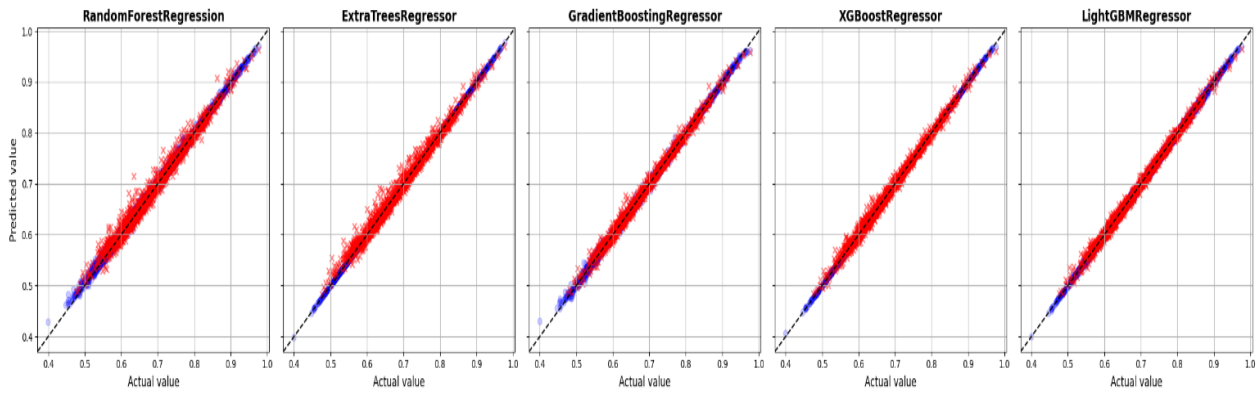
From figure. 7, regressions scatter plot, there are no significant differences in the performance of various models on the training (blue O) and test (red X) sets. The simulation versus prediction results for different models in figure. 7 shows that machine learning predictions are mostly located at a very small range of the Average Throughput simulation. Comparing the dispersion of pairs of data for each model in figure. 7 shows that all models are very good at predicting the Average Throughput on the test and training data sets. Table 3. presents the results of the average throughput prediction using tree-based models, evaluated on both training and test datasets. All models achieved perfect $r^2 = 1$ on the training data, indicating a complete fit to the training set. However, this raises concerns about overfitting, as the models capture the training data without errors. On the test data, GBDT, LightGBM, and XGBoost demonstrate the highest r^2 scores of 0.993, 0.994, and 0.993, respectively, with low MAPE values of 0.0065, 0.0061, and 0.0067. This indicates strong generalization and high accuracy in predicting average throughput. ET also performs well, achieving an r^2 of 0.9836 and a slightly higher MAPE of 0.01, showing competitive performance but slightly lower predictive accuracy than the boosting-based models. RF, while still performing well with an r^2 of 0.9824 and MAPE of 0.01, lags slightly behind the other models, indicating slightly less precise predictions. Overall, GBDT, LightGBM, and XGBoost stands as the best-performing models for predicting Average Throughput, offering a balance between low error rates and high generalization. ET provides strong but slightly less accurate results, while RF shows the highest error among the models, suggesting room for improvement.

Table 3: Average Throughput Performance Metric Values of Different Tree-Based Models

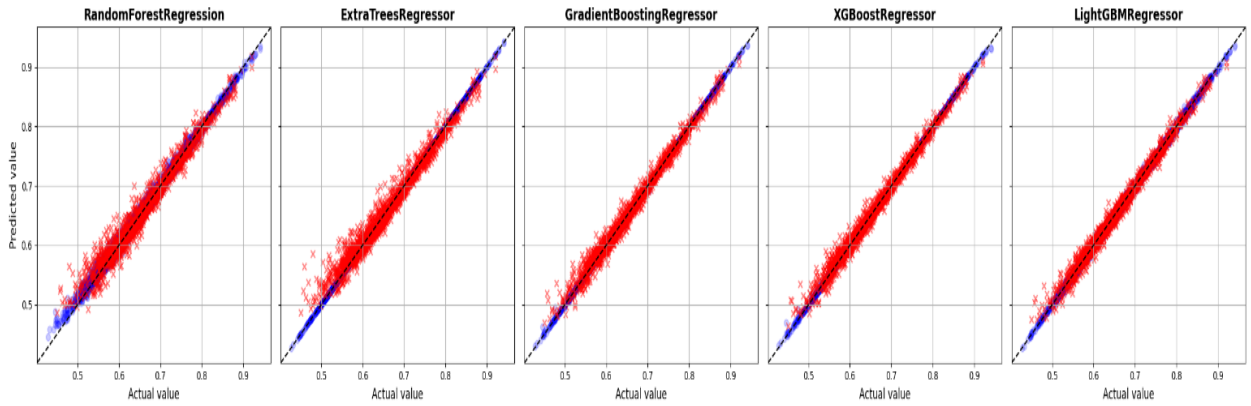
M	ML Model	Training Data		Test Data	
		MAPE	r^2	MAPE	r^2
2	ET	0.0011	1	0.0052	0.996
	GBDT	0.0012	1	0.0034	0.998
	LightGMB	0.0018	1	0.0032	0.998
	RF	0.0022	1	0.0056	0.996
	XGBoost	0	1	0.0041	0.998
3	ET	0	1	0.0098	0.986
	GBDT	0.0024	1	0.0062	0.994
	LightGMB	0.0027	1	0.0061	0.995
	RF	0.0039	1	0.0104	0.984
	XGBoost	0	1	0.0066	0.993
4	ET	0	1	0.0153	0.967
	GBDT	0.0007	1	0.0098	0.986
	LightGMB	0.0036	1	0.0089	0.989
	RF	0.0056	1	0.0153	0.966
	XGBoost	0	1	0.0094	0.987
Average Result Across All Systems	ET	0	1	0.01	0.983
	GBDT	0.0014	1	0.0065	0.993
	LightGMB	0.0027	1	0.0061	0.994
	RF	0.0039	1	0.01	0.982
	XGBoost	0	1	0.0067	0.993



(a)



(b)



(c)

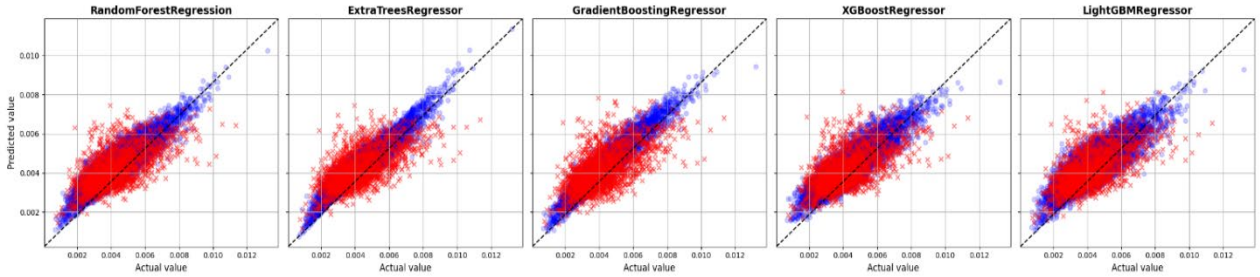
Figure 7: Scatter Plot Regression of Average Throughput Results: (a) M = 2, (b) M = 3, (c) M = 4.

5.1.3 Throughput Standard Deviation Results

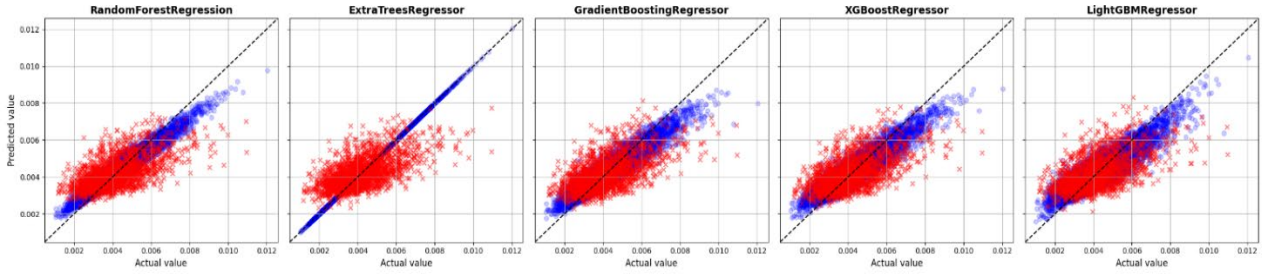
Table 4. summarizes the performance of tree-based models in predicting the Standard Deviation of Throughput, revealing significant shortcomings in the models' accuracy and generalization. Unlike the performance observed in predicting the average throughput, all models struggle with the standard deviation prediction, both in terms of the mean absolute percentage error and the coefficient of determination. For the training data, ET achieves the highest r^2 score of 0.961 and the lowest MAPE of 0.032, indicating superior performance relative to the other models during training. However, its performance significantly deteriorates on the test data, where r^2 drops to 0.379 and MAPE increases to 0.224, suggesting poor generalization. Similarly, boosting-based models like GBDT, LightGBM, and XGBoost exhibit consistently lower r^2 scores, ranging from 0.345 to 0.354, with MAPE values between 0.227 and 0.228 on the test data. RF also struggles, with r^2=0.371 and MAPE = 0.224, reflecting minimal predictive accuracy. These results indicate that none of the models effectively capture the variability in throughput. The poor predictions for the standard deviation of throughput significantly impact the overall multioutput regression results, as this target variable introduces substantial errors. While the models performed well in predicting average throughput, their inability to generalize for standard deviation highlights a major limitation. This discrepancy suggests that the models struggle to learn the underlying patterns of throughput variability, which is critical in this context. Addressing this issue may require feature engineering, model regularization, or alternative modeling approaches to improve generalization. From figure. 8, the regression scatter plot further illustrates these issues. It is evident that there are significant differences in the models' performance on the training (blue O) and test (red X) sets. The simulation versus prediction results show that machine learning predictions for standard deviation of throughput are widely dispersed and fail to align closely with the simulated values. Comparing the dispersion of data points for each model in figure. 8 underscores that all models perform poorly in predicting standard deviation throughput on both the training and test datasets. These findings emphasize the need for more advanced modeling techniques approaches to improve the predictions of standard deviation of throughput and, consequently, improve the performance of the multioutput regression models as a whole.

Table 4: Standard Deviation Throughput Performance Metric Values of Different Tree-Based Models

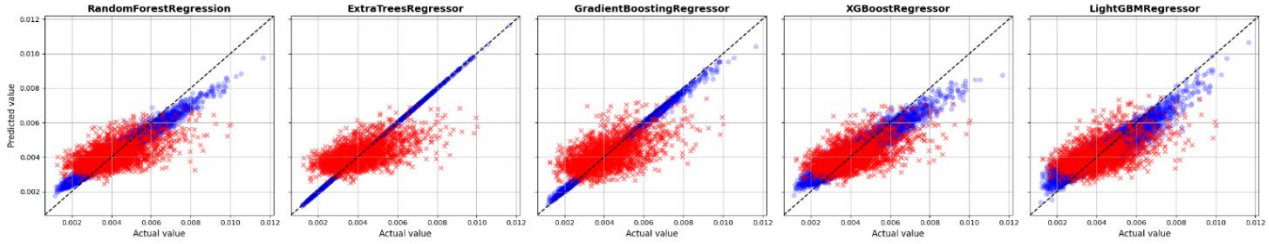
M	ML Model	Training Data		Test Data	
		MAPE	r^2	MAPE	r^2
2	ET	0.097	0.882	0.218	0.430
	GBDT	0.090	0.901	0.224	0.391
	LightGMB	0.154	0.733	0.219	0.415
	RF	0.134	0.804	0.218	0.426
	XGBoost	0.146	0.773	0.226	0.391
3	ET	0	1	0.225	0.376
	GBDT	0.096	0.875	0.227	0.364
	LightGMB	0.126	0.804	0.230	0.337
	RF	0.083	0.914	0.227	0.362
	XGBoost	0.111	0.850	0.229	0.357
4	ET	0	1	0.227	0.331
	GBDT	0.019	0.994	0.232	0.293
	LightGMB	0.118	0.827	0.233	0.284
	RF	0.086	0.905	0.227	0.326
	XGBoost	0.104	0.863	0.226	0.315
Average Result Across All Systems	ET	0.032	0.961	0.224	0.379
	GBDT	0.068	0.923	0.228	0.349
	LightGMB	0.133	0.788	0.227	0.345
	RF	0.101	0.875	0.224	0.371
	XGBoost	0.120	0.829	0.227	0.354



(a)



(b)



(c)

Figure 8: Scatter Plot Regression of Standard Deviation Throughput Results: (a) M = 2, (b) M = 3, (c) M = 4.

5.1.4 Average Buffer Level Results

Since the number of buffers keep increasing with the number of machines meaning the 2-machine system have 1 buffer, the 3-machine system have 2 buffers, and the 4-machine system have 3 buffers. We are going to calculate the $MAPE_{Buffer}$ by taking the average value of the $MAPE_j$ across all buffers in each system. The following equation represents it. where j represents the number of buffers j = 1, 2, … M – 1.

$$MAPE_{Buffer} = \frac{\sum_{j=1}^{M-1} MAPE_j}{M-1} \quad , \qquad R^2_{Buffer} = \frac{\sum_{j=1}^{M-1} R_j^2}{M-1}$$

The results in Table 5 present the performance of tree-based models in predicting the average buffer level, for the training data, all models achieve a perfect r^2 score of 1.000, indicating that they perfectly fit the training data, though this may also suggest overfitting. The MAPE values for training range from 0.003 (ET) to 0.046 (LightGBM), indicating that the models are highly accurate on the training set. However, on the test data, the models show some variance in performance. GBDT demonstrates the best generalisation with the lowest MAPE of 0.073 and the highest R2 of 0.99, closely followed by XGBoost (R2 = 0.99, MAPE = 0.079). ET achieves a strong r^2 of 0.982 and a moderate MAPE of 0.108, but its performance slightly lags behind the boosting models. LightGBM (r^2 = 0.99, MAPE = 0.092) and RF (r^2 = 0.978, MAPE = 0.116) also perform reasonably well, though their higher test errors indicate reduced predictive accuracy. The results highlight that while the models perform well overall in predicting the average buffer level, the increasing number of buffers introduces complexity, making the task more challenging. Boosting-based models (GBDT, LightGBM and XGBoost) consistently demonstrate better generalization compared to other methods, as reflected by their lower test MAPE and higher r^2 values. These findings highlight the advantage of boosting algorithms in handling more complex prediction tasks, particularly when there is an increasing variability due to additional buffers in the system.

Table 5: Average Buffer Level Across Each System Performance Metric Values of Different Tree-Based Models

M	ML Model			Test Data	
		MAPE	r^2	MAPE	r^2
2	ET	0.009	1	0.046	0.997
	GBDT	0.013	1	0.034	0.998
	LightGMB	0.033	1	0.050	0.998
	RF	0.023	1	0.057	0.995
	XGBoost	0.000	1	0.048	0.996
3	ET	0	1	0.104	0.984
	GBDT	0.030	1	0.069	0.992
	LightGMB	0.045	1	0.087	0.991
	RF	0.042	1	0.111	0.981
	XGBoost	0.002	1	0.070	0.992
4	ET	0	1	0.175	0.963
	GBDT	0.015	1	0.116	0.979
	LightGMB	0.059	1	0.138	0.980
	RF	0.067	1	0.178	0.959
	XGBoost	0.016	1	0.110	0.981
Average Result Across All Systems	ET	0	1	0.108	0.982
	GBDT	0.019	1	0.073	0.990
	LightGMB	0.046	1	0.092	0.990
	RF	0.044	1	0.116	0.978
	XGBoost	0	1	0.076	0.990

5.1.5 Statistical Hypothesis Test

To evaluate the predictive performance of various machine learning models across systems of differing complexity, the Wilcoxon Signed-Rank Test was employed to compare the predicted outputs against the actual observed values for each output individually. This non-parametric statistical test was selected because it does not assume normal distribution of the data and is particularly well-suited for comparing paired, continuous outcomes making it ideal for evaluating model performance across multiple regression targets. Moreover, it is robust against outliers and can handle asymmetric error distributions (Trawiński et al., 2012), which are often present in multi-output regression tasks involving throughput and buffer level predictions. The models were tested on systems with 2, 3, and

4 machines, and performance was assessed using three key metrics average throughput, standard deviation of throughput, and average buffer level. An "Average Result across all Outputs" score was computed for holistic comparison as shown in Table 6. Among all tested models, LightGBM consistently outperformed others, achieving the highest average results across outputs in systems with M = 2 (0.33), M = 3 (0.51), and M = 4 (0.34), with an overall average of 0.39 across all systems. XGBoost followed with a mean score of 0.25, while Random Forest (RF) and Extra Trees (ET) yielded minimal predictive capability, particularly in systems with M = 3, where both showed near-zero performance across all metrics. The statistical significance obtained from the Wilcoxon Signed-Rank Test confirmed that the differences between predicted and actual values were non-random for most configurations, particularly for models with higher aggregated scores such as LightGBM and GBDT. These findings suggest that LightGBM offers superior generalization performance and robustness across different system sizes, particularly in capturing complex throughput and buffer dynamics. Conversely, ET and RF models appear inadequate for capturing the nuances of such multi-output regression tasks, especially as system complexity increases.

Table 6: P Values Result for Different Models (95% Confidence Level)

M	ML Model	Avg Throughput	Std Throughput	Avg buffer level	Average Result Across all Outputs
2	ET	0.00	0.16	0.00	0.05
	GBDT	0.15	0.57	0.18	0.30
	LightGMB	0.18	0.37	0.45	0.33
	RF	0.00	0.17	0.65	0.27
	XGBoost	0.68	0.34	0.19	0.40
3	ET	0.00	0.00	0.00	0.00
	GBDT	0.49	0.00	0.01	0.17
	LightGMB	0.83	0.00	0.69	0.51
	RF	0.00	0.00	0.00	0.00
	XGBoost	0.18	0.00	0.01	0.06
4	ET	0.00	0.00	0.22	0.07
	GBDT	0.23	0.00	0.33	0.19
	LightGMB	0.60	0.00	0.41	0.34
	RF	0.00	0.00	0.14	0.05
	XGBoost	0.48	0.00	0.33	0.27
Average Result Across All Systems	ET	0.00	0.05	0.07	0.04
	GBDT	0.29	0.19	0.17	0.22
	LightGMB	0.54	0.12	0.52	0.39
	RF	0.00	0.06	0.26	0.11
	XGBoost	0.45	0.11	0.18	0.25

5.2 Comparison with Other Models

When comparing the performance of the average throughput prediction model for the 2-machine system in this study with kim S, similar research that utilized neural network architectures, several key differences and observations arise, (Kim et al., 2024) as shown in Figure. 9 The primary difference lies in the dataset size and complexity. The other research used a significantly larger dataset, comprising 70,000 rows per system, compared to this study's dataset of 10,000 rows per system. However, the complexity of their data was also higher, as they represented machine efficiency using a Weibull distribution with a coefficient of variation greater than one, capturing greater variability and randomness. In contrast, our dataset defines machine efficiency using an exponential distribution, which is comparatively simpler. These differences in dataset size and complexity largely balance each other, with the larger dataset in the other research compensating for its added complexity, making the comparison relevant. Figure 9 shows the MAPE result on the test data where the blue bars represent the result for the neural network models and the orange bars represent the result for the tree-based models. Note that only the LSTM (MAPE = 0.24%) outperformed the tree-based models in terms of predictive accuracy. Tree-based models such as GBDT (MAPE = 0.34%), LightGBM (MAPE = 0.32%), and XGBoost (MAPE = 0.41%) achieved comparable results, trailing the LSTM by a narrow margin. However, the other neural network models, including GRU (MAPE = 0.58%), RNN (MAPE = 0.66%), and MLP (MAPE = 2.18%), exhibited worse performance than the tree-based models, with much higher MAPE values. Traditional models such as ET (MAPE = 0.52%) and RF (MAPE = 0.56%) also outperformed the GRU, RNN, and

MLP. These findings indicate that while LSTM's ability to handle sequential data makes it highly effective for this task, other neural network architectures may struggle with similar datasets. Tree-based models in our study not only provide competitive performance, but outperform the majority of neural network models except for LSTM, highlighting their suitability for this application

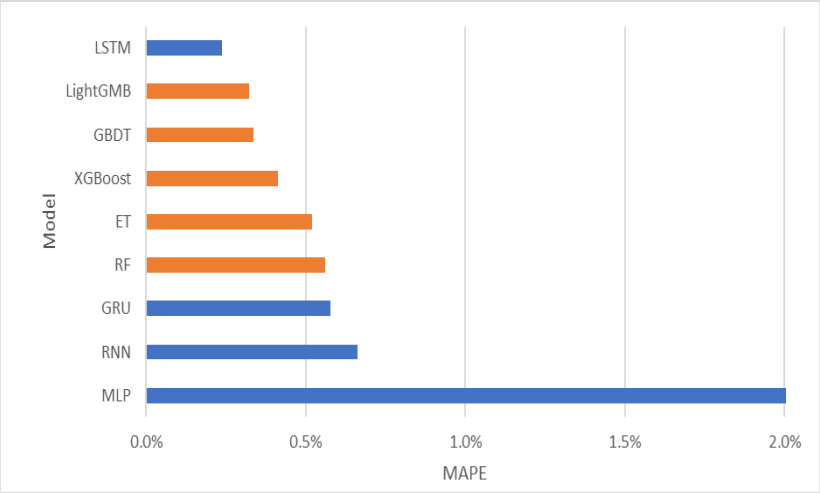


Figure 6: Average Throughput Results Tree-Based Models' Comparison with Neural Networks Models

6. Conclusion

This research demonstrates the practical value of tree-based machine learning models in evaluating the performance of serial production lines. Among the models tested, LightGBM showed superior accuracy and efficiency, followed closely by CatBoost and XGBoost. These models effectively captured complex production dynamics and provided reliable predictions for decision-making in manufacturing environments. The study confirms the feasibility of integrating intelligent predictive tools into production planning and monitoring systems. Future work could explore the application of deep learning methods and real-time data integration to further enhance predictive accuracy and adaptability in dynamic industrial settings.

References

Assaf, R., Colledani, M., & Matta, A. (2014). Analytical evaluation of the output variability in production systems with general Markovian structure. *OR spectrum*, *36*(3), 799-835. https://doi.org/10.1007/s00291-013-0343-6

Bai, Y., Tu, J., Yang, M., Zhang, L., & Denno, P. (2021). A new aggregation algorithm for performance metric calculation in serial production lines with exponential machines: design, accuracy and robustness. *International Journal of Production Research*, *59*(13), 4072-4089. https://doi.org/10.1080/00207543.2020.1757777

Biau, G., & Scornet, E. (2016). A random forest guided tour. *Test*, *25*(2), 197-227. https://doi.org/10.1007/s11749-016-0481-7

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5-32. https://doi.org/10.1023/A:1010933404324

Chen, S., Li, D., Gan, X., & Chiong, R. (2024). An Optimised Light Gradient Boosting Machine Model for Setup Time Prediction in Electronic Production Lines. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (pp. 647-650). https://doi.org/10.1145/3638530.3654428

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785-794). https://doi.org/10.1145/2939672.2939785

Chen, X., & Ishwaran, H. (2012). Random forests for genomic data analysis. *Genomics*, *99*(6), 323-329. https://doi.org/10.1016/j.ygeno.2012.04.003

Chen, Y., Khandelwal, M., Onifade, M., Zhou, J., Lawal, A. I., Bada, S. O., & Genc, B. (2025). Predicting the

hardgrove grindability index using interpretable decision tree-based machine learning models. *Fuel*, *384*, 133953. https://doi.org/10.1016/j.fuel.2024.133953

Chiang, S.-Y., Kuo, C.-T., & Meerkov, S. M. (2000). DT-bottlenecks in serial production lines: theory and application. *IEEE Transactions on Robotics and Automation*, *16*(5), 567-580. https://doi.org/10.1109/70.880806

Colledani, M., & Tolio, T. (2005). A decomposition method to support the configuration/reconfiguration of production systems. *CIRP annals*, *54*(1), 441-444. https://doi.org/10.1016/S0007-8506(07)60140-1

Dasari, S. K., Lavesson, N., Andersson, P., & Persson, M. (2015). Tree-based response surface analysis. In *International workshop on machine learning, optimization and big data* (pp. 118-129). Springer. https://doi.org/10.1007/978-3-319-27926-8_11

Demir, L., & Koyuncuoğlu, M. U. (2021). The impact of the optimal buffer configuration on production line efficiency: A VNS-based solution approach. *Expert Systems with Applications*, *172*, 114631. https://doi.org/10.1016/j.eswa.2021.114631

Dey, R. (2024). Bagging v/s Boosting. In. Medium. https://medium.com/@roshmitadey/bagging-v-s-boosting-be765c970fd1

El-kenawy, E.-S. M., Abutarboush, H. F., Mohamed, A. W., & Ibrahim, A. (2021). Advance artificial intelligence technique for designing double T-shaped monopole antenna. *Computers, Materials & Continua*, *69*(3). https://doi.org/10.32604/cmc.2021.019114

El-Kenawy, E.-S. M., Zerouali, B., Bailek, N., Bouchouich, K., Hassan, M. A., Almorox, J., Kuriqi, A., Eid, M., & Ibrahim , A. (2022). Improved weighted ensemble learning for predicting the daily reference evapotranspiration under the semi-arid climate conditions. *Environmental Science and Pollution Research*, *29*(54), 81279-81299. https://doi.org/10.1007/s11356-022-21410-8

Elshabrawy, M. (2025). A review on waste management techniques for sustainable energy production. *Metaheuristic Optim. Rev.*, *3*(2), 47-58. https://www.americaspg.com/article/pdf/3493

Galelli, S., & Castelletti, A. (2013). Assessing the predictive capability of randomized tree-based ensembles in streamflow modelling. *Hydrology and Earth System Sciences*, *17*(7), 2669-2684. https://doi.org/10.5194/hess-17-2669-2013, 2013

Galuzzi, B. G., Giordani, I., Candelieri, A., Perego, R., & Archetti, F. (2020). Hyperparameter optimization for recommender systems through Bayesian optimization. *Computational Management Science*, *17*(4), 495-515. https://doi.org/10.1007/s10287-020-00376-3

Gershwin, S. B. (1987). An efficient decomposition method for the approximate evaluation of tandem queues with finite storage space and blocking. *Operations research*, *35*(2), 291-305. https://doi.org/10.1287/opre.35.2.291

Gershwin, S. B., & Berman, O. (1981). Analysis of transfer lines consisting of two unreliable machines with random processing times and finite storage buffers. *AIIE transactions*, *13*(1), 2-11. https://doi.org/10.1080/05695558108974530

Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine learning*, *63*(1), 3-42. https://doi.org/10.1007/s10994-006-6226-1

Hon, K. (2005). Performance and evaluation of manufacturing systems. *CIRP annals*, *54*(2), 139-154. https://doi.org/10.1016/S0007-8506(07)60023-7

Hu, L., & Li, L. (2022). Using tree-based machine learning for health studies: literature review and case series. *International journal of environmental research and public health*, *19*(23), 16080. https://doi.org/10.3390/ijerph192316080

Kavlakoglu, E., & Russi, E. (2024). *What is XGBoost?* IBM. https://www.ibm.com/think/topics/xgboost

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, *30*. https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf

Khaled, K., & Singla, M. (2025). Predictive analysis of groundwater resources using random forest regression. *J. Artif.*

Intell. Metaheuristics, *9*(1), 11-19. https://www.researchgate.net/publication/388403901_Predictive_Analysis_of_Groundwater_Resources_Using_Random_Forest_Regression

Kim, S., Won, Y., Park, K. J., & Eun, Y. (2024). Throughput Approximation by Neural Network for Serial Production Lines With High Up/Downtime Variability. *IEEE Transactions on Industrial Informatics*, *20*(3), 4227-4235. https://doi.org/10.1109/TII.2023.3321026

Kuo, C., Lim, J., & Meerkov, S. (1996). Bottlenecks in serial production lines: A system-theoretic approach. *Arbor*, *1050*, 48109-42122. https://doi.org/10.1155/S1024123X96000348

Le Bihan, H., & Dallery, Y. (2000). A robust decomposition method for the analysis of production lines with unreliable machines and finite buffers. *Annals of Operations Research*, *93*(1), 265-297. https://doi.org/10.1023/A:1018996428429

Li, J., E. Blumenfeld, D., Huang, N., & M. Alden, J. (2009). Throughput analysis of production systems: recent advances and future topics. *International Journal of Production Research*, *47*(14), 3823-3851. https://doi.org/10.1080/00207540701829752

Li, X., Li, W., & Xu, Y. (2018). Human age prediction based on DNA methylation using a gradient boosting regressor. *Genes*, *9*(9), 424. https://doi.org/10.3390/genes9090424

Liu, C.-M., Su, S.-F., & Lin, C.-L. (1996). Predictive models for performance evaluation of serial production lines. *International Journal of Production Research*, *34*(5), 1279-1291. https://doi.org/10.1080/00207549608904965

Machello, C., Baghaei, K. A., Bazli, M., Hadigheh, A., Rajabipour, A., Arashpour, M., Rad, H. M., & Hassanli, R. (2024). Tree-based machine learning approach to modelling tensile strength retention of Fibre Reinforced Polymer composites exposed to elevated temperatures. *Composites Part B: Engineering*, *270*, 111132. https://doi.org/10.1016/j.compositesb.2023.111132

MultiOutputRegressor, Scikit-learn. (2024). https://scikit-learn.org/stable/modules/generated/sklearn.multioutput.MultiOutputRegressor.html

NA. (2024). *LightGMB Features*. Microsoft Corporation. https://lightgbm.readthedocs.io/en/stable/Features.html

Natekin, A., & Knoll, A. (2013). Gradient boosting machines, a tutorial. *Frontiers in neurorobotics*, *7*, 21. https://doi.org/10.3389/fnbot.2013.00021

Nguyen, V. (2019). Bayesian optimization for accelerating hyper-parameter tuning. In *2019 IEEE second international conference on artificial intelligence and knowledge engineering (AIKE)* (pp. 302-305). IEEE. https://doi.org/10.1109/AIKE.2019.00060

P. Bruce, A. Bruce, & P. Gedeck. (2020). Bagging and boosting. In *Practical Statistics for Data Scientists* (SECOND ed.). O'Reilly Media, Chapter 6. https://www.oreilly.com/library/view/practical-statistics-for/9781492072935/

Samee, N. A., El-Kenawy, E.-S. M., Atteia, G., Jamjoom, M. M., Ibrahim, A., Abdelhamid, A. A., El-Attar, N. E., Gaber, T., Slowik, A., & Shams, M. Y. (2022). Metaheuristic Optimization Through Deep Learning Classification of COVID-19 in Chest X-Ray Images. *Computers, Materials & Continua*, *73*(2). https://doi.org/10.32604/cmc.2022.031147

Tan, B., & Gershwin, S. B. (2009). Analysis of a general Markovian two-stage continuous-flow production system with a finite buffer. *International Journal of Production Economics*, *120*(2), 327-339. https://doi.org/10.1016/j.ijpe.2008.05.022

Tan, B., & Gershwin, S. B. (2011). Modelling and analysis of Markovian continuous flow systems with a finite buffer. *Annals of Operations Research*, *182*(1), 5-30. https://doi.org/10.1007/s10479-009-0612-6

Trawiński, B., Smętek, M., Telec, Z., & Lasota, T. (2012). Nonparametric statistical analysis for multiple comparison of machine learning regression algorithms. https://doi.org/10.2478/v10006-012-0064-z

Wang, Z., Mu, H., & You, J. (2014). Analysis of a serial production line with single part-type and multiple parallel-machine workstations. *IFAC Proceedings Volumes*, *47*(2), 306-313. https://doi.org/10.3182/20140514-3-FR-4046.00033

Wehenkel, L., Ernst, D., & Geurts, P. (2006). Ensembles of extremely randomized trees and some generic applications. In *Robust methods for power system state estimation and load forecasting*. https://orbi.uliege.be/bitstream/2268/13447/1/robust-trees.pdf

Wong, Q. Y., & Chu, Y. B. (2021). A mobile production monitoring system based on internet of thing (IoT) and random forest classification. *International Journal of Electrical and Electronic Engineering & Telecommunications*, *10*(4), 243-250. https://doi.org/10.18178/ijeetc.10.4.243-250

Wu, D., Jennings, C., Terpenny, J., Gao, R. X., & Kumara, S. (2017). A comparative study on machine learning algorithms for smart manufacturing: tool wear prediction using random forests. *Journal of Manufacturing Science and Engineering*, *139*(7), 071018. https://doi.org/10.1115/1.4036350

Wu, J., Chen, X.-Y., Zhang, H., Xiong, L.-D., Lei, H., & Deng, S.-H. (2019). Hyperparameter optimization for machine learning models based on Bayesian optimization. *Journal of Electronic Science and Technology*, *17*(1), 26-40. https://doi.org/10.11989/JEST.1674-862X.80904120