Yamashiro, Hirochika; Nonaka, Hirofumi

Article

Estimation of processing time using machine learning and real factory data for optimization of parallel machine scheduling problem

Operations Research Perspectives

Provided in Cooperation with:
Elsevier

Suggested Citation: Yamashiro, Hirochika; Nonaka, Hirofumi (2021) : Estimation of processing time using machine learning and real factory data for optimization of parallel machine scheduling problem, Operations Research Perspectives, ISSN 2214-7160, Elsevier, Amsterdam, Vol. 8, pp. 1-9, https://doi.org/10.1016/j.orp.2021.100196

Operations Research Perspectives 8 (2021) 100196

Contents lists available at ScienceDirect

Operations Research Perspectives

journal homepage: www.elsevier.com/locate/orp

Estimation of processing time using machine learning and real factory data for optimization of parallel machine scheduling problem

Hirochika Yamashiro *, Hirofumi Nonaka

Nagaoka University of Technology, Japan

ARTICLE INFO

Keywords:
Machine learning
Gaussian process regression
Gradient boosted decision trees
Artificial neural networks
Identical parallel machine scheduling
Operations research

ABSTRACT

Traditionally, mathematical optimization methods have been applied in manufacturing industries where production scheduling is one of the most important problems and is being actively researched. Extant studies assume that processing times are known or follow a simple distribution. However, the actual processing time in a factory is often unknown and likely follows a complex distribution. Therefore, in this study, we consider estimating the processing time using a machine-learning model. Although there are studies that use machine learning for scheduling optimization itself, it should be noted that the purpose of this study is to estimate an unknown processing time. Using machine-learning models, one can estimate processing times that follow an unknown and complex distribution while further improving the schedule using the computed importance variable. Based on the above, we propose a system for estimating the processing time using machine-learning models when the processing time follows a complex distribution in actual factory data. The advantages of the proposed system are its versatility and applicability to a real-world factory where the processing times are often unknown. The proposed method was evaluated using process information with the processing time for each manufacturing sample provided by research partner companies. The Light gradient-boosted machine (LightGBM) algorithm and Ridge performed the best with MAPE and RMSE. The optimization of parallel machine scheduling using estimated processing time by our method resulted in an average reduction of approximately 30% for the makespan. On the other hands, the results of probabilistic sampling methods which are Kernel Density Estimation, Gamma distribution, and Normal Distribution have shown poorer performance than ML approaches. In addition, machine-learning models can be used to deduce variables that affect the estimation of processing times, and in this study, we demonstrated an example of feature importance computed from experimental data. In addition, machine-learning models can be used to deduce variables that affect the estimation of processing times, and in this study, we demonstrated an example of feature importance computed from experimental data.

1. Introduction

Traditionally, mathematical optimization methods have been applied in the manufacturing industries where production scheduling is one of the most important problems and is being actively researched.

Li et al. [1] proposed a scheduling method for additive manufacturing machines, known as metal 3D printing, using a mixed-integer linear programming (MILP) model. In their study, the authors used a self-created dataset for evaluation. Many other studies have also used datasets with predefined processing times [2–7]. However, processing times in a real factory are not always known in advance.

This problem can be solved in two ways. The first is to sample the processing time from a simple probability distribution, such as a uniform or normal distribution instead of using a known processing time. The second is a machine learning-based prediction method.

Many studies have been conducted using the first, where the distributions were uniform, for the evaluation of scheduling algorithms [8–12]. For example, Lin et al. [13] proposed a method for solving the parallel machine scheduling problem using heuristic rules and Mixed-Integer Linear Programming (MILP) methods. The processing time was sampled from a uniform distribution and the method was then evaluated for each job and machine.

In contrast, Shen et al. [14] sampled the processing time from a normal distribution to evaluate their proposed method. Their method works on the assumption that the jobs follow a normal distribution and allows bias in the processing time. Several other studies [2,15] have also evaluated models by sampling the processing time from a normal distribution.

* Corresponding author.
E-mail addresses: s173358@stn.nagaokaut.ac.jp (H. Yamashiro), nonaka@kjs.nagaokaut.ac.jp (H. Nonaka).

https://doi.org/10.1016/j.orp.2021.100196
Received 4 March 2021; Received in revised form 9 July 2021; Accepted 12 July 2021
Available online 16 July 2021

Table 1
A comparison between this and previous studies on analyzability when processing time is unknown.

Paper	Analyzability when the processing time is unknown
[1]-[7]	×
[8]-[15]	O
Our	O

Table 2
Comparison of the analyzability of processing time when it does not follow a simple distribution.

Paper	Analyzability when processing times do not follow a normal or uniform distribution
[1]-[15]	×
Our	O

These distributions can be used for evaluation and for sampling the known processing time. However, the distribution is not always uniform at actual manufacturing sites because of the irregularity in processing times. There are also cases where it is difficult to assume that the processing time follows a simple normal probability distribution. The distribution of actual factory processing times is often unknown or asymmetrical. Therefore, the above-mentioned sampling methods are inherently problematic.

To address this issue, in this study we investigate the prediction of processing times from process information using a machine learning model. Specifically, product information such as product materials and the number of products manufactured are considered as the input, and a regression model is constructed with the processing time as the objective variable. Machine learning models have been used in various estimation problems in real factories [16–20] and may be useful in estimating processing time in this study.

In the scheduling optimization field, some studies have used machine learning for scheduling optimization itself, rather than estimating the unknown processing time, which is the objective of this study. For example, [21] proposed an automated guided vehicles (AGVs) real-time scheduling method using deep Q-learning to minimize the makespan and delay rates. In another study, [22] proposed a method that uses multiple supervised learning methods and an ensemble technique to predict the average tardiness and select the best dispatching rule from multiple dispatching rules. In addition, several other studies have used machine learning to solve scheduling optimization [23–25]. In the above study, machine learning is used for scheduling optimization. However, in this study, instead of proposing a method for scheduling optimization using machine learning methods, we use machine learning to estimate the processing time required for scheduling optimization.

Tables 1 and 2 show a comparison between the previous studies and the proposed methods.

Based on the aforementioned findings, this research proposes a system to estimate the processing time and perform machine scheduling in cases where the processing time follows a complex distribution, similar to a real-world factory scenario.

In this paper, Section 2 presents an overall diagram of the proposed system, the actual machine learning model to be used, and the formulation of the machine scheduling problem to be handled. In Sections 3 and 4, we apply the proposed system to real-world factory data to show the accuracy of the machine learning model and the rate at which job completion time is reduced. In Section 5, we verify whether the distribution of actual processing time using the Kolmogorov–Smirnov (KS) test follows a normal distribution and show the necessity of using a machine learning model in a real factory. We also discuss the accuracy of the machine learning model and the scheduling results. Finally, in Section 6, the conclusions of this study and future work are discussed.

2. Method

In this section, we describe a machine learning method for estimating the processing time and a method for machine scheduling optimization. In this research, we use machine learning models to find the variables required for scheduling a machine in an actual factory and then perform scheduling. The advantage of the proposed system is its versatility and applicability to a real-world factory where the processing times are often unknown.

An overview of the system is shown in Fig. 1. We first measure the data from the work site and store them in a database for machine learning. Next, using the stored process information as input, a machine learning model is used to estimate the processing time. Finally, we perform scheduling optimization for a factory where the processing times are unknown.

2.1. Machine learning

In this study, we used four typical machine learning models often used for machine learning to estimate the processing time. In addition, we used multiple linear regression as the baseline method. Of the following models, the most accurate provided the estimated processing time used for scheduling.

1. LightGBM
2. Gaussian Process
3. Ridge Regression
4. Artificial Neural Network

2.1.1. LightGBM [26–30]

LightGBM is a novel gradient boosting decision tree (GBDT) algorithm proposed by Ke et al. [31]. A summary of the LightGBM learning method is shown in Fig. 2. In LightGBM, a histogram of the input data is sampled, and the data are weighted and input to simple decision trees. Each decision tree is trained sequentially, and the weights of the input data reflect the previous training results.

This model has several innovations that allow more computational efficiency than the conventional GBDT algorithm. The first is a sampling method called gradient-based one-side sampling, which, to limit computational complexity, reduces the amount of data needed for training by using samples rather than all of the input data. In addition, it is sampled for the distribution of the data does to remain unchanged. It is known that in this method, if the number of data points is sufficiently large, the approximation of the distribution of the data will be accurate. The second, called exclusive feature bundle (EFB), is a method that reduces feature dimensionality. The features of real-world data are often sparse and exclusive. We use EFB to combine multiple features into one, thus reducing the amount of calculation. The best way to summarize features is to consider a reduced version of the graph-coloring problem and solve it using a greedy algorithm. Fig. 3 shows a simple example.

Given the supervised training set $X = \{(x_i, y_i)\}_{i=1}^{n}$, LightGBM aims to find an approximation $\hat{f}(x)$ to a certain function $f^*(x)$ that minimizes the expected value of a specific loss function $L(y, f(x))$ as follows:

$$\hat{f} = \arg\min_{f} E_{y,X} L(y, f(x)). \tag{1}$$

This model was used as a processing time prediction method because of its short training time and high estimation accuracy compared to general machine learning models.

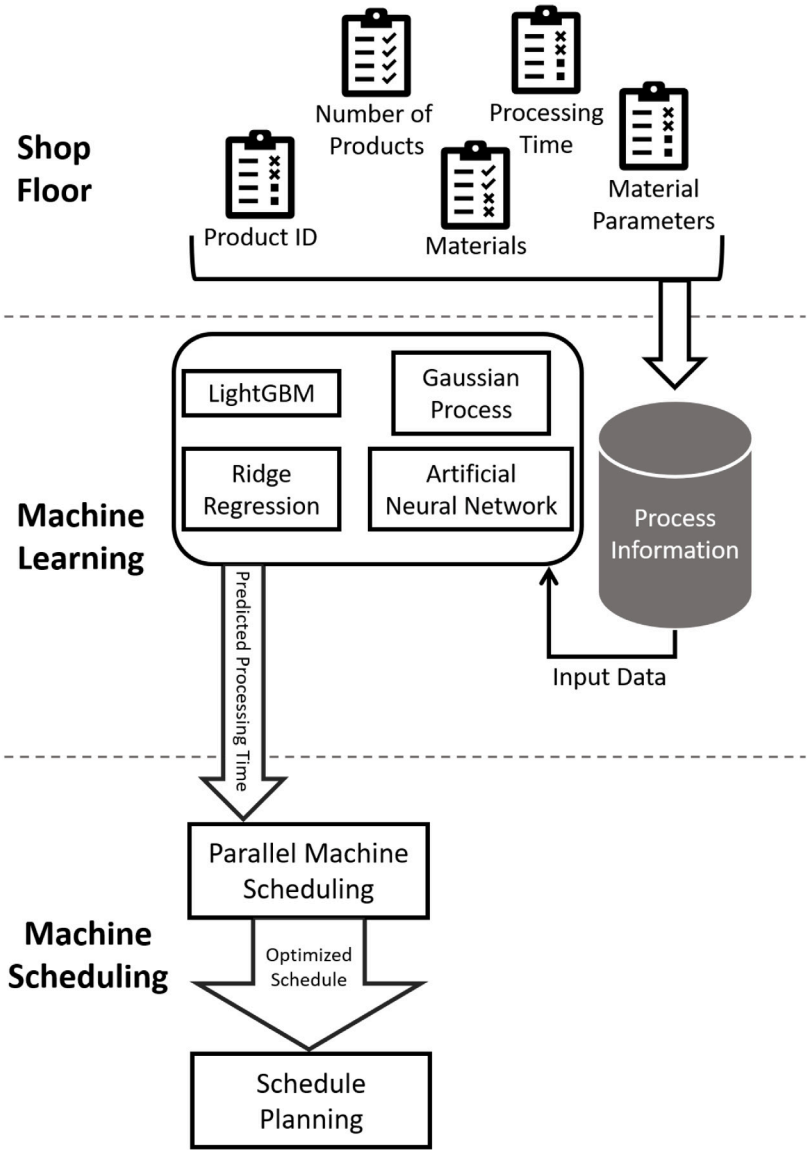

Fig. 1. Overview of the proposed system.

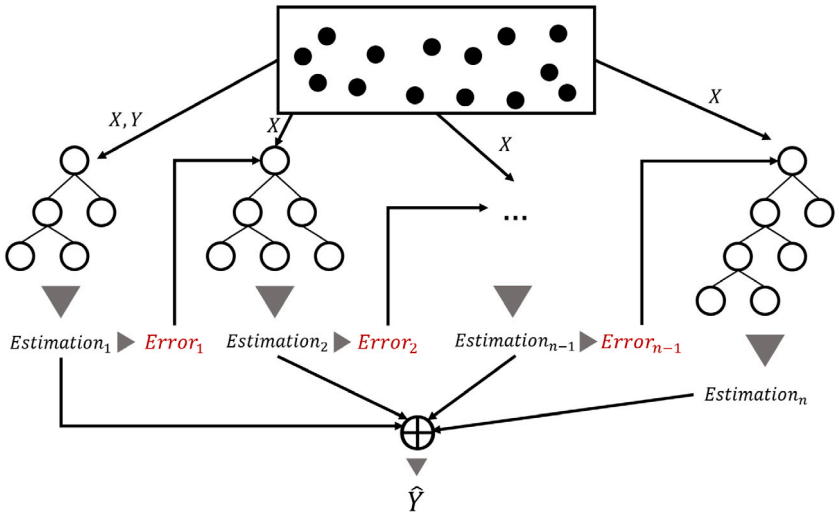

Fig. 2. Summary of the LightGBM learning method.

Fig. 3. Example of exclusive feature bundling.

2.1.2. Gaussian process [32–36]

A Gaussian process [37] is a collection of a finite number of random variables that have a Gaussian distribution. It is specified by the mean and covariance functions. Gaussian process regression is a probabilistic regression technique that employs a Bayesian methodology to derive a non-linear model. It assumes that the output of the regression model is dependent on the latent function $f(X)$ and the Gaussian noise, ϵ, with a zero mean. The output of the regression model is given by Eq. (2).

$$y = f(X) + \epsilon. \tag{2}$$

where the function $f(\cdot) : \mathbb{R}^p \mapsto \mathbb{R}$. The latent function follows a Gaussian distribution with a mean and variance given by

$$\mu = \mathbf{K}(X_{tr}, X_{te})^T \mathbf{k}(X_{tr} + \sigma^2 I)^{-1} y \tag{3}$$

$$\sigma = \mathbf{k}(X_{te}) - \mathbf{K}(X_{tr}, X_{te})^T \mathbf{k}(X_{tr} + \sigma^2 I)^{-1} \mathbf{K}(X_{tr}, X_{te}). \tag{4}$$

where $\mathbf{K}(X_{tr}, X_{te})$ is the covariance between the training inputs and test point, and $\mathbf{k}(X_{tr})$ and $\mathbf{k}(X_{te})$ are the autocovariances of the training and test points, respectively. The covariance function reflects the presumptions about the latent function $f(X)$, and therefore has a very important purpose. In general, this covariance function is called a kernel function, and the combination of various kernel functions can greatly change the accuracy of the model.

2.1.3. Ridge regression [38–41]

Ridge regression is a model in which the $L2$ norm is introduced as a regularization term in linear regression and the weights are trained to minimize the following objective functions:

$$(\mathbf{y} - \mathbf{X}\boldsymbol{w})^T (\mathbf{y} - \mathbf{X}\boldsymbol{w}) + \alpha \|\boldsymbol{w}\|_2. \tag{5}$$

where α is a parameter that expresses the strength of the regularization. This model is less prone to overfitting than simple linear regression models.

2.1.4. Artificial neural network

An artificial neural network (ANN) is a powerful and widely used machine learning model capable of combining several non-linear functions to capture non-linear relationships between input data and a label [42–47]. The ANN consists of a feed-forward phase and a back-propagation phase. In the feed-forward phase, input data are propagated forward through the layers of the ANN and configure the loss function with the target value at the output layer. During the back-propagation phase, the derivative of the loss function with respect to each weight is calculated in the reverse direction, and then optimization techniques are used to find the optimal weight. From among the many neural network models, we used Residual neural network (ResNet) [48] in this research. A ResNet represents a mass of layers with a structure that adds the input to the output double or triple layers ahead. Using this structure, we can create a model that is robust against gradient loss and accuracy saturation even when the layers are deep.

2.2. Machine scheduling

The factory data used in this research are treated as identical parallel machine scheduling problems because all jobs can be processed by all machines and the processing time is not machine dependent.

In this study, to simplify the evaluation, we solve a parallel machine scheduling problem where the processing times of the work machine are identical, which is a machine scheduling problem where the objective function is to minimize the total processing time (makespan).

We now describe the variables used in the optimization and definitions of the expressions. J represents the number of jobs used for optimization, and there are n jobs. Similarly, M represents the number of machines used for optimization, and there are m machines. p_{ij} is the processing time of job j on machine i.

The following notations are introduced:

J_j job $j, j = 1, 2, \ldots, n$

M_i machine $i, i = 1, 2, \ldots, m$

p_{ij} processing time of job j on machine i

x_{ij} is a decision variable that is set to 1 if job j is assigned to machine i and 0 if otherwise. We formulate the problem as follows:

$$\min \quad \sum_{i=1}^{m} \sum_{j=1}^{n} p_{ij} x_{ij} \tag{6}$$

$$\text{s.t.} \quad \sum_{i=1}^{m} x_{ij} = 1 \quad \forall j = 1, 2, \ldots, n \tag{7}$$

$$x_{ij} \in \{0, 1\} \quad \forall i = 1, 2, \ldots, m \quad \forall j = 1, 2, \ldots, n. \tag{8}$$

Eq. (6) is an objective function and represents the time it takes to complete all jobs. Eq. (7) denotes the assignment of a given job to only one machine. In this study, the Gurobi optimizer was used to perform the calculations.

Table 3
Number of elements per item.

Columns	Types of variables	Number of categories
Number of processes	Natural number	–
Machine code	Category	18
Work completion flag	Category	2
Quality level	Category	5
Material A	Category	184
Parameter A	Category	63
Parameter B	Category	101
Parameter C	Category	21
Parameter D	Category	2
Parameter E	Category	38
Actual processing time	Natural number	–

3. Experiments

To evaluate the proposed method, we used process information with known processing times compiled for each production sample provided by the research partners. The process information used in this study included 22537 data points for one year from July 2018 to June 2019. The operator manually pressed the start and end buttons to measure the processing time. The number of elements in the preprocessed data items is summarized in Table 3. With these items as inputs, we constructed a machine learning model to predict the processing time and solve a parallel machine scheduling problem using the estimated processing time. The mean absolute percentage error (MAPE) is used as an evaluation index for the machine learning model. The MAPE value is given by the following formula:

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right| * 100. \tag{9}$$

where A_t represents the measured value and F_t represents the predicted value. MAPE calculates the absolute value of the error rate of the predicted value and uses it as an evaluation index by calculating the average of the error rates.

The process information for 10 days is extracted and used as test data. The rest of the data were divided into training data and validation data at a 9 to 1 ratio using the product ID and were evaluated using the K-Fold method. The reason for using the product ID to split the training and test data was to reproduce the situation of estimating the processing time of an unknown product on the current sample when used in a real factory. The program was run on a server with Intel Xeon W-2123 3.60 GHz processors, 128 GB RAM, and a 64-bit Ubuntu 16.04 operating system.

The machine learning models to be compared in this study were subjected to hyperparameter tuning. The hyperparameter tuning was performed using Optuna, a parameter tuning framework.

Table 4 shows the tuned hyperparameter of the LightGBM. In this study, three important parameters were tuned. *num_leaves* is the main parameter to control the complexity of the tree model. *min_data_in_leaf* is a very important parameter to prevent overfitting in a leaf-wise tree. *max_depth* is a parameter that limits the depth of the tree, thus suppressing overfitting. The range of values for the parameters to be tuned is as follows:

number of leaves $2 \leq num\_leaves \leq 128$

minimum data in leaf $1 \leq min\_data\_in\_leaf \leq 40$

max depth $0 \leq max\_depth \leq 50.$

Eq. (10) represents the tuned kernel function. In the Gaussian process, the kernel function is a parameter. In this study, we experimented with RBF, Matern32, and Matern52 kernels, and found that the $RBFkernel$ attained the best MAPE value.

Table 4
Tuned LightGBM parameters.

Parameter	Parameter value
objective	regression
boosting_type	gbdt
metric	mape
num_leaves	83
min_data_in_leaf	21
max_depth	2

Table 5
Tuned network parameters.

Parameter	Parameter value
hidden_layers	15
optimizer	SGD
batch_size	992
learning_rate	0.000248
activation_function	ReLU

$$k(\mathbf{x}, \mathbf{x}') = ConstantKernel * RBFKernel + WhiteKernel. \tag{10}$$

$Constantkernel$ represents a constant kernel. The $RBFkernel$ is an abbreviation for the radial basis function kernel and is a general kernel that is used for non-linear objective variables. The $RBFkernel$ is expressed as follows:

$$k(x_i, x_j) = \exp\left(-\frac{d(x_i, x_j)^2}{2l^2}\right). \tag{11}$$

where l is a parameter. $WhiteKernel$ is a kernel that considers the observation noise, which is the variation in the target variable during observation. In this study, we introduce the variation in processing time for the same material due to the difference in number of performance, assuming that the observed noise is the variation in processing time. The value of $ConstantKernel$ is set to 1 for each kernel.

We tuned the parameter α in the Ridge regression and found that $\alpha = 7.146$ was optimal. The range of values for the parameter to be tuned is as follows:

α $0.000001 \leq \alpha \leq 100000.$

Table 5 shows the parameters of the tuned network. In this study, we constructed a network with fifteen hidden layers, consisting of five Residual Blocks stacked on top of each other, each consisting of three layers. In addition, as in general ResNet, the activation function is input after the batch normalization layer. The range of values for the parameters to be tuned is as follows:

optimizer Adam, SGD, Adagrad, Adadelta and RMSprop

learning rate $0.000001 \leq learning\_rate \leq 10$

batch size $100 \leq batch\_size \leq 1000.$

The first step is to estimate the processing time for the extracted test data and perform scheduling optimization. The optimization procedure optimizes it using the number of jobs, the number of machines per day, and the estimated processing times. To evaluate the optimal schedule, we compare the makespan of the current schedule with the makespan of the optimal schedule, calculate the reduction rate of the makespan, and use the average value of the two. Eq. (12) is a method for calculating the makespan reduction rates, where $CT_{current}$ is the makespan of the current schedule and $CT_{optimized}$ is the optimized makespan. An example of calculating the makespan shortening rate is shown in Fig. 4.

$$\frac{CT_{current} - CT_{optimized}}{CT_{current}}. \tag{12}$$

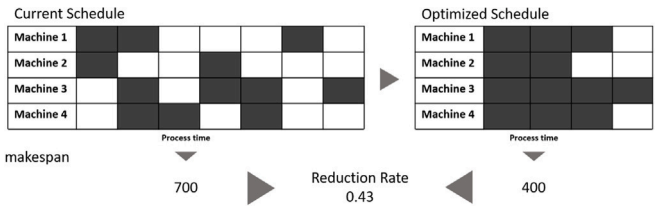

Fig. 4. An example of calculating the makespan.

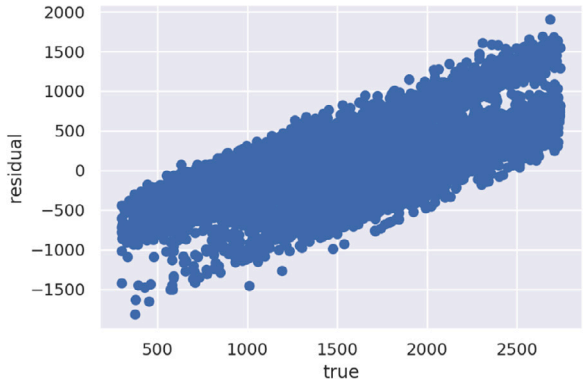

Fig. 5. A residual plot of LightGBM.

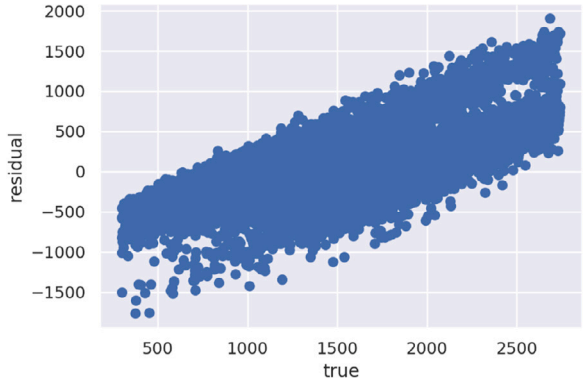

Fig. 6. A residual plot of Ridge Regression.

4. Results

The data were divided by the K-Fold method, and the mean MAPE and RMSE were calculated. The mean MAPE and RMSE values for each machine learning model are shown in Table 6. In addition, we performed a Welch's t-test to confirm the significance of the difference between the output of each method. Consequently, the null hypothesis on MAPE and RMSE can be rejected at the five percent significance level for the difference between LightGBM and GP. Conversely, the null hypothesis on MAPE and RMSE could not be rejected at the five percent significance level for the difference between LightGBM and Ridge Regression. Therefore, LightGBM and Ridge Regression were selected as the best-performing methods. To complement our analysis, we further conducted a residual plot for the top three methods based on the MAPE to analyze the distribution residuals. Figs. 5–7 show the residual plots for the three methods. As a result, errors are evenly distributed which means techniques do not tend to under/overestimate the time.

Because LightGBM was the most accurate of the machine learning models in terms of the mean MAPE values, the predicted processing time of LightGBM was used to optimize the parallel machine scheduling. The mean reduction rate of the makespan for the optimal schedule

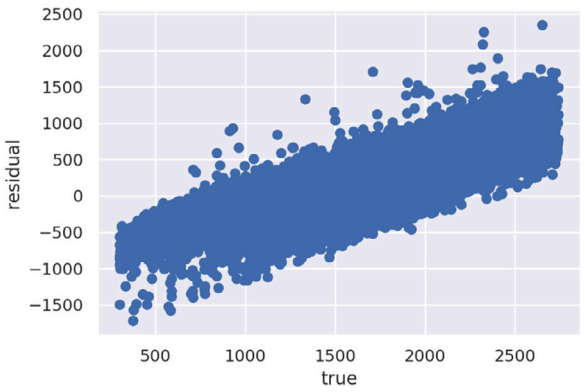

Fig. 7. A residual plot of Gaussian process.

Table 6
MAPE value and RMSE for each model.

Models	MAPE	RMSE
LightGBM	21.960	358.255
Ridge regression	22.515	361.122
Gaussian process	22.896	369.993
Artificial neural network	36.353	526.234
KDE	47.647	732.424
Gamma	50.673	727.504
Multiple regression	444469.903	47848190.368
Multiple regression with lasso	23.286	359.123

was 29.468%. Table 7 summarizes the makespan of the current schedule, the makespan of the optimized schedule using the actual processing time, and the makespan of the optimized schedule using the processing time predicted by the machine learning model.

5. Discussion

We tested whether the processing time of the data used in this study follows a normal distribution using the Kolmogorov–Smirnov (KS) test for the null hypothesis that the two distributions to be compared are the same. In this study, the KS test was performed using the distribution of processing times in the real data and the normal distribution created from the mean and variance of processing times. A KS test resulted in a p-value of zero, rejecting the null hypothesis. This indicates that the actual factory data used in this study does not follow the normal distribution because the processing times are biased by differences in worker inputs and other similar factors despite the products being identical. Fig. 8 shows the histogram of the real data and the histogram of the normal distribution created from the real data.

We also sampled the processing time using a normal distribution and evaluated the sampled processing time by MAPE and RMSE using the evaluation data. As parameters of the normal distribution, we used the mean and variance of the actual processing time. The results show that the MAPE value is approximately 0.5, which is inferior to the performance of machine learning models such as LightGBM and Ridge regression. This is because the KS test rejects normality and unlike machine learning models, the sampling process does not use features, resulting in large local errors. Conversely, since the histogram clearly shows that the data distribution is not Gaussian, sampling with a gamma distribution may be considered. We performed the KS test and MAPE/RMSE evaluation as well as the normal distribution. The parameters of the gamma distribution, α and β, were sampled using the NUTS method, which is the MCMC sampling method, and the EAP estimator was obtained. The results Fig. 10 showed that the KS test rejected the null hypothesis and did not follow the gamma distribution. The MAPE was also in the low 0.4 range, indicating that the performance was inferior to machine learning.

Table 7
Comparison of the current schedule makespan and the optimized schedule makespan.

Current schedule	Schedule optimized by actual processing time	Schedule optimized by LightGBM predicted processing time	Schedule optimized by Ridge predicted processing time
27833	23529	23146	24258
45722	21941	25462	17410
32668	16460	17447	30822
32616	31465	32410	24605
26300	21108	23355	26556
35448	30432	26777	22628
45746	19090	17998	18715
28154	22091	21050	20895
24538	20950	19063	19290
28807	18437	16292	16313

Table 8
Training time for each machine learning model.

Machine learning model	Training time
LightGBM	19.593
Ridge regression	12.062
Gaussian process	3649.024
Artificial neural network	93500.670

The kernel density estimation (KDE) is a non-parametric method of estimating the probability density function of a random variable. KDE is a basic data smoothing problem that makes inferences about a population based on a finite sample of data. The KDE method is to approximate the distribution of training data. Fig. Fig. 9 shows the histogram of the real data and the histogram of the predicted values from KDE. We compared the results of the KS test and MAPE/RMSE as well as the normal and gamma distributions. The results show that the KS test fails to reject the null hypothesis, indicating that the KDE method is capable of generating an adequate distribution. However, the MAPE value was in the low 0.4 range. This is because, unlike machine learning models, KDE does not use features, which leads to large local errors. In conclusion, the results show that a method that uses probability sampling is inferior to the machine learning method in terms of MAPE/RMSE. This is mainly because it does not take into account the features, which leads to local errors.

We performed variable selection using Lasso regression for multiple regression. The Lasso regression is one of the linear regression models that introduce the $L1$ regularization. In this model, we minimize the following objective function:

$$\|y - \mathbf{X}\boldsymbol{w}\|_2^2 + \lambda\|\boldsymbol{w}\|_0. \tag{13}$$

where λ denotes the parameter that controls the $L1$ term. When $\lambda = 0$, the equation is the same as in ordinary multiple regression analysis. One of the characteristics of lasso regression is that the weights of some variables may become zero by performing $L1$ regularization. In this study, we use this property to filter the variables.

From the results of Table 7, our proposed system were able to reduce the makespan by approximately 30% compared to the current schedule by using LightGBM/Ridge regression to estimate the processing time from the process information and scheduling with the estimated values. The reason for this is thought to be that with the current schedule, the processing is concentrated on a specific machine, and the scheduling optimization has allocated the processing to other machines.

In addition, the MAPE and RMSE value of ANN is low compared to the computation time, it is better to use LightGBM or Ridge regression. Table 8 shows the training time for each machine learning model.

In addition, machine learning models can be used to deduce variables that affect the estimation of processing time. We show an example of feature importance computed from experimental data using SHapley Additive exPlanations. In this study, the feature importance was calculated using LightGBM, which had the best mean MAPE value. Fig. 11 shows the importance of the item "Material A". In "Material A", it can be interpreted that the material that tends to increase the value of the processing time is used in part 25.

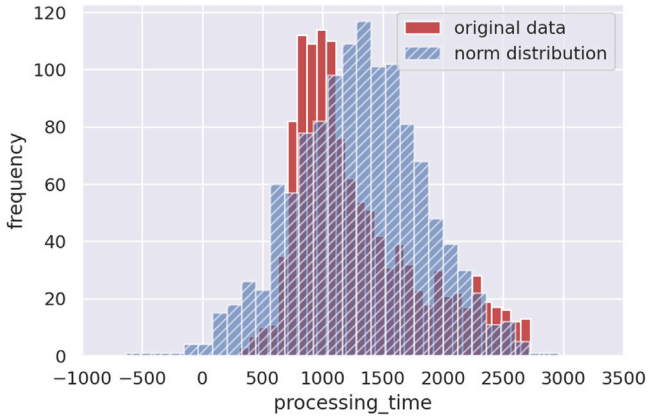

Fig. 8. The histogram of the real data and the normal distribution.

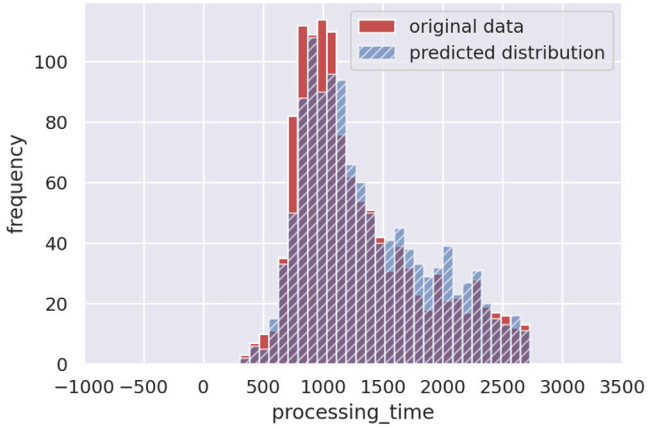

Fig. 9. The histogram of the real data and the predicted distribution by KDE.

Based on the above-mentioned results, it can be concluded that the analysis of explainable machine learning models can be useful for making decisions in actual factories. Specifically, it can be used in integration with productivity improvement methods such as value stream mapping (VSM), which can contribute to further schedule improvement.

6. Conclusion

In this study, we proposed a system for estimating the processing time and scheduling machines when the processing times are complexly distributed in real-world factory data. The advantages of our system are

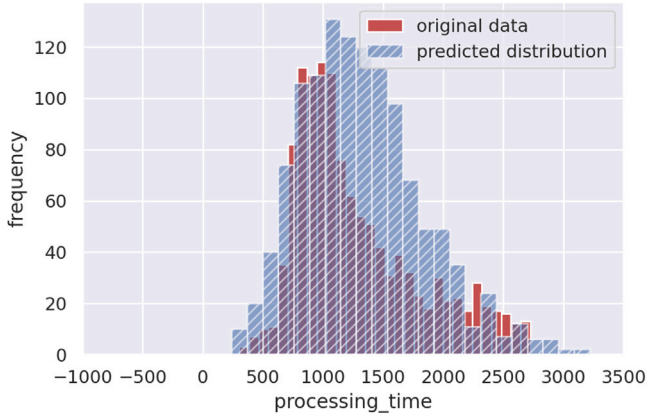

Fig. 10. The histogram of the real data and the predicted gamma distribution.

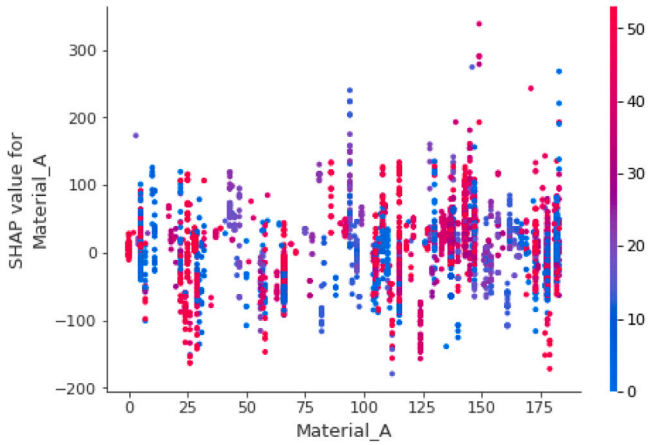

Fig. 11. Importance of item "Material A".

that scheduling optimization is possible for jobs with complex distributions with unknown processing times, and that process information can be analyzed by the machine learning model used in the learning process.

To verify the proposed method, we predicted the processing times using actual factory data and solved a parallel machine scheduling problem using the predicted processing times to perform scheduling optimization. Four machine learning models were used to estimate the processing time. LightGBM had the best accuracy with a MAPE value of approximately 22%. Parallel machine scheduling optimization, using the estimated LightGBM processing time reduced the makespan by approximately 30%. Unlike previous studies that assumed a normal distribution for processing time, the factory processing time used in this study did not follow a normal distribution. This demonstrates the need to estimate the processing time using a machine learning model.

In this study, the experiment was only conducted at one factory; therefore, it is necessary to verify whether the proposed method can be applied to other factories. However, the factory where the experiment was conducted has a large number of data, and the results are considered to be reliable.

The effectiveness of applying a machine learning technique to a real factory for scheduling optimization will depend on the amount of input data used. Therefore, problems such as scalability and generalizability warrant further investigation. Several studies have used the computation of the feature importance of the trained model to perform such analyses in real factories [49–52]. Therefore, we believe that such analyses with an explainable machine learning model would be useful to support decision-making in real factories and will lead to further improvements in scheduling. In particular, after identifying items that significantly affect the processing time and by focusing on them, the processing time can be reduced. By analyzing the work related to those items, productivity methods such as VSM [53–55], and also scheduling could be improved. Furthermore, the reduced processing time can be used to further reduce the schedule. In addition, a framework for improving production efficiency [56–58] can also be applied to factories where the processing times are not known.

In this paper, we focused on methods for offline scheduling, but in the future, we may apply these methods to online scheduling problems. In such cases, selecting a method to minimize the makespan by using a causal verification model such as A/B test would be possible.

In the experiments of this study, we assumed that the working time of a job is constant for all machines. However, in the future, it may be possible to deal with the case where the working time of a job varies from machine to machine.

CRediT authorship contribution statement

Hirochika Yamashiro: Conception and design of study, Acquisition of data, Analysis and/or interpretation of data, Writing – original draft. **Hirofumi Nonaka:** Conception and design of study, Acquisition of data, Analysis and/or interpretation of data, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

All authors approved the version of the manuscript to be published.

References

[1] Li Q, Kucukkoc I, Zhang DZ. Production planning in additive manufacturing and 3D printing. Comput Oper Res 2017;83:157–72. http://dx.doi.org/10.1016/j.cor.2017.01.013.

[2] González-Neira EM, Urrego-Torres AM, Cruz-Riveros AM, Henao-García C, Montoya-Torres JR, Molina-Sánchez LP, et al. Robust solutions in multi-objective stochastic permutation flow shop problem. Comput Ind Eng 2019;137:106026. http://dx.doi.org/10.1016/j.cie.2019.106026.

[3] Chergui A, Hadj-Hamou K, Vignat F. Production scheduling and nesting in additive manufacturing. Comput Ind Eng 2018;126:292–301. http://dx.doi.org/10.1016/j.cie.2018.09.048.

[4] Fera M, Fruggiero F, Lambiase A, Macchiaroli R, Todisco V. A modified genetic algorithm for time and cost optimization of an additive manufacturing single-machine scheduling. Int J Ind Eng Comput 2018;9(4):423–38. http://dx.doi.org/10.5267/j.ijiec.2018.1.001.

[5] Kucukkoc I. MILP models to minimise makespan in additive manufacturing machine scheduling problems. Comput Oper Res 2019;105:58–67. http://dx.doi.org/10.1016/j.cor.2019.01.006.

[6] Min L, Cheng W. A genetic algorithm for minimizing the makespan in the case of scheduling identical parallel machines. Artif Intell Eng 1999;13(4):399–403. http://dx.doi.org/10.1016/S0954-1810(99)00021-7.

[7] Wu X, Che A. A memetic differential evolution algorithm for energy-efficient parallel machine scheduling. Omega 2019;82:155–65. http://dx.doi.org/10.1016/j.omega.2018.01.001.

[8] Fanjul-Peyro L, Ruiz R, Perea F. Reformulations and an exact algorithm for unrelated parallel machine scheduling problems with setup times. Comput Oper Res 2019;101:173–82. http://dx.doi.org/10.1016/j.cor.2018.07.007.

[9] Wang S, Wang X, Yu J, Ma S, Liu M. Bi-objective identical parallel machine scheduling to minimize total energy consumption and makespan. J Cleaner Prod 2018;193:424–40. http://dx.doi.org/10.1016/j.jclepro.2018.05.056.

[10] Zhang G, Xing K, Cao F. Discrete differential evolution algorithm for distributed blocking flowshop scheduling with makespan criterion. Eng Appl Artif Intell 2018;76:96–107. http://dx.doi.org/10.1016/j.engappai.2018.09.005.

[11] Goren S, Sabuncuoglu I. Optimization of schedule robustness and stability under random machine breakdowns and processing time variability. IIE Trans 2009;42(3):203–20. http://dx.doi.org/10.1080/07408170903171035.

[12] Tang D, Dai M, Salido MA, Giret A. Energy-efficient dynamic scheduling for a flexible flow shop using an improved particle swarm optimization. Comput Ind 2016;81:82–95. http://dx.doi.org/10.1016/j.compind.2015.10.001.

[13] Lin S-W, Ying K-C. Uniform parallel-machine scheduling for minimizing total resource consumption with a bounded makespan. IEEE Access 2017;5:15791–9. http://dx.doi.org/10.1109/ACCESS.2017.2735538.

[14] Shen J, Zhu Y. A parallel-machine scheduling problem with periodic maintenance under uncertainty. J Ambient Intell Humaniz Comput 2019;10(8):3171–9. http://dx.doi.org/10.1007/s12652-018-1032-8.

[15] Framinan JM, Fernandez-Viagas V, Perez-Gonzalez P. Using real-time information to reschedule jobs in a flowshop with variable processing times. Comput Ind Eng 2019;129:113–25. http://dx.doi.org/10.1016/j.cie.2019.01.036.

[16] Carvalho TP, Soares FA, Vita R, Francisco RdP, Basto JP, Alcalá SG. A systematic literature review of machine learning methods applied to predictive maintenance. Comput Ind Eng 2019;137:106024. http://dx.doi.org/10.1016/j.cie.2019.106024.

[17] Kang Z, Catal C, Tekinerdogan B. Machine learning applications in production lines: A systematic literature review. Comput Ind Eng 2020;149:106773. http://dx.doi.org/10.1016/j.cie.2020.106773.

[18] Khalifa RM, Yacout S, Bassetto S. Developing machine-learning regression model with Logical Analysis of Data (LAD). Comput Ind Eng 2020;106947. http://dx.doi.org/10.1016/j.cie.2020.106947.

[19] Morariu C, Morariu O, Răileanu S, Borangiu T. Machine learning for predictive scheduling and resource allocation in large scale manufacturing systems. Comput Ind 2020;120:103244. http://dx.doi.org/10.1016/j.compind.2020.103244.

[20] Dalzochio J, Kunst R, Pignaton E, Binotto A, Sanyal S, Favilla J, et al. Machine learning and reasoning for predictive maintenance in industry 4.0: Current status and challenges. Comput Ind 2020;123:103298. http://dx.doi.org/10.1016/j.compind.2020.103298.

[21] Hu H, Jia X, He Q, Fu S, Liu K. Deep reinforcement learning based AGVs real-time scheduling with mixed rule for flexible shop floor in industry 4.0. Comput Ind Eng 2020;149:106749. http://dx.doi.org/10.1016/j.cie.2020.106749.

[22] Priore P, Ponte B, Puente J, Gómez A. Learning-based scheduling of flexible manufacturing systems using ensemble methods. Comput Ind Eng 2018;126:282–91. http://dx.doi.org/10.1016/j.cie.2018.09.034.

[23] Jong W-R, Chen H-T, Lin Y-H, Chen Y-W, Li T-C. The multi-layered job-shop automatic scheduling system of mould manufacturing for Industry 3.5. Comput Ind Eng 2020;149:106797. http://dx.doi.org/10.1016/j.cie.2020.106797.

[24] Waschneck B, Reichstaller A, Belzner L, Altenmüller T, Bauernhansl T, Knapp A, et al. Optimization of global production scheduling with deep reinforcement learning. Proc CIRP 2018;72(1):1264–9. http://dx.doi.org/10.1016/j.procir.2018.03.212.

[25] Zhou Z, Li F, Zhu H, Xie H, Abawajy JH, Chowdhury MU. An improved genetic algorithm using greedy strategy toward task scheduling optimization in cloud environments. Neural Comput Appl 2020;32(6):1531–41. http://dx.doi.org/10.1007/s00521-019-04119-7.

[26] Fan J, Ma X, Wu L, Zhang F, Yu X, Zeng W. Light Gradient Boosting Machine: An efficient soft computing model for estimating daily reference evapotranspiration with local and external meteorological data. Agricult Water Manag 2019;225:105758. http://dx.doi.org/10.1016/j.agwat.2019.105758.

[27] Ma X, Sha J, Wang D, Yu Y, Yang Q, Niu X. Study on a prediction of P2P network loan default based on the machine learning LightGBM and XGboost algorithms according to different high dimensional data cleaning. Electron Commer Res Appl 2018;31:24–39. http://dx.doi.org/10.1016/j.elerap.2018.08.002.

[28] Sun X, Liu M, Sima Z. A novel cryptocurrency price trend forecasting model based on LightGBM. Finance Res Lett 2020;32:101084. http://dx.doi.org/10.1016/j.frl.2018.12.032.

[29] Zhao Q, Ye Z, Su Y, Ouyang D. Predicting complexation performance between cyclodextrins and guest molecules by integrated machine learning and molecular modeling techniques. Acta Pharm Sinica B 2019;9(6):1241–52. http://dx.doi.org/10.1016/j.apsb.2019.04.004.

[30] Song Y, Jiao X, Qiao Y, Liu X, Qiang Y, Liu Z, et al. Prediction of double-high biochemical indicators based on lightgbm and XGBoost. In: Proceedings of the 2019 international conference on artificial intelligence and computer science. 2019, p. 189–93. http://dx.doi.org/10.1145/3349341.3349400.

[31] Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, et al. Lightgbm: A highly efficient gradient boosting decision tree. In: Advances in neural information processing systems. 2017, p. 3146–54.

[32] Lv Y, Yang X, Zhang G. Durability of phase-change-material module and its relieving effect on battery deterioration during long-term cycles. Appl Therm Eng 2020;115747. http://dx.doi.org/10.1016/j.applthermaleng.2020.115747.

[33] Kumari SA, Srinivasan S. Ash fouling monitoring and soot-blow optimization for reheater in thermal power plant. Appl Therm Eng 2019;149:62–72. http://dx.doi.org/10.1016/j.applthermaleng.2018.12.031.

[34] Zhu L, Chen J, Chen C-I. Prognostics of tool failing behavior based on autoassociative Gaussian process regression for semiconductor manufacturing. In: 2020 IEEE international conference on industrial technology (ICIT). IEEE; 2020, p. 316–21. http://dx.doi.org/10.1109/ICIT45562.2020.9067286.

[35] Faghihpirayesh R, Imbiriba T, Yarossi M, Tunik E, Brooks D, Erdoğmuş D. Motor cortex mapping using active gaussian processes. In: Proceedings of the 13th ACM international conference on pervasive technologies related to assistive environments. 2020, p. 1–7. http://dx.doi.org/10.1145/3389189.3389202.

[36] Raissi M, Perdikaris P, Karniadakis GE. Machine learning of linear differential equations using Gaussian processes. J Comput Phys 2017;348:683–93. http://dx.doi.org/10.1016/j.jcp.2017.07.050.

[37] Williams CK, Rasmussen CE. Gaussian processes for regression. In: Advances in neural information processing systems. 1996, p. 514–20.

[38] Li K-C, et al. Asymptotic optimality of C_L and generalized cross-validation in ridge regression with application to spline smoothing. Ann Statist 1986;14(3):1101–12. http://dx.doi.org/10.1214/aos/1176350052.

[39] Pasha G, Shah M. Application of ridge regression to multicollinear data. J Res (Science) 2004;15(1):97–106.

[40] An S, Liu W, Venkatesh S. Face recognition using kernel ridge regression. In: 2007 IEEE conference on computer vision and pattern recognition. IEEE; 2007, p. 1–7. http://dx.doi.org/10.1109/CVPR.2007.383105.

[41] Ogutu JO, Schulz-Streeck T, Piepho H-P. Genomic selection using regularized linear regression models: ridge regression, lasso, elastic net and their extensions. In: BMC proceedings, vol. 6. Springer; 2012, p. S10. http://dx.doi.org/10.1186/1753-6561-6-S2-S10.

[42] Son H, Hyun C, Phan D, Hwang HJ. Data analytic approach for bankruptcy prediction. Expert Syst Appl 2019;138:112816. http://dx.doi.org/10.1016/j.eswa.2019.07.033.

[43] Lee BC, Brooks DM, de Supinski BR, Schulz M, Singh K, McKee SA. Methods of inference and learning for performance modeling of parallel applications. In: Proceedings of the 12th ACM SIGPLAN symposium on principles and practice of parallel programming. 2007, p. 249–58. http://dx.doi.org/10.1145/1229428.1229479.

[44] Mellit A, Pavan AM. A 24-h forecast of solar irradiance using artificial neural network: Application for performance prediction of a grid-connected PV plant at trieste, Italy. Sol Energy 2010;84(5):807–21. http://dx.doi.org/10.1016/j.solener.2010.02.006.

[45] Agatonovic-Kustrin S, Beresford R. Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research. J Pharm Biomed Anal 2000;22(5):717–27. http://dx.doi.org/10.1016/S0731-7085(99)00272-1.

[46] Çepelioğullar Ö, Mutlu İ, Yaman S, Haykiri-Acma H. A study to predict pyrolytic behaviors of refuse-derived fuel (RDF): Artificial neural network application. J Anal Appl Pyrolysis 2016;122:84–94. http://dx.doi.org/10.1016/j.jaap.2016.10.013.

[47] Benali L, Notton G, Fouilloy A, Voyant C, Dizene R. Solar radiation forecasting using artificial neural network and random forest methods: Application to normal beam, horizontal diffuse and global components. Renew Energy 2019;132:871–84. http://dx.doi.org/10.1016/j.renene.2018.08.044.

[48] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016, p. 770–8. http://dx.doi.org/10.1109/CVPR.2016.90.

[49] Wen S, Buyukada M, Evrendilek F, Liu J. Uncertainty and sensitivity analyses of co-combustion/pyrolysis of textile dyeing sludge and incense sticks: Regression and machine-learning models. Renew Energy 2020;151:463–74. http://dx.doi.org/10.1016/j.renene.2019.11.038.

[50] Zhang J, Wang P, Gao RX. Deep learning-based tensile strength prediction in fused deposition modeling. Comput Ind 2019;107:11–21. http://dx.doi.org/10.1016/j.compind.2019.01.011.

[51] Li F, Wu J, Dong F, Lin J, Sun G, Chen H, et al. Ensemble machine learning systems for the estimation of steel quality control. In: 2018 IEEE international conference on big data (big data). IEEE; 2018, p. 2245–52. http://dx.doi.org/10.1109/BigData.2018.8622583.

[52] Yang H, Park M, Cho M, Song M, Kim S. A system architecture for manufacturing process analysis based on big data and process mining techniques. In: 2014 IEEE international conference on big data (big data). IEEE; 2014, p. 1024–9. http://dx.doi.org/10.1109/BigData.2014.7004336.

[53] Hines P, Rich N. The seven value stream mapping tools. Int J Oper Prod Manag 1997. http://dx.doi.org/10.1108/01443579710157989.

[54] Wang P, Wu P, Chi H-L, Li X. Adopting lean thinking in virtual reality-based personalized operation training using value stream mapping. Autom Constr 2020;119:103355. http://dx.doi.org/10.1016/j.autcon.2020.103355.

[55] Heravi G, Firoozi M. Production process improvement of buildings' prefabricated steel frames using value stream mapping. Int J Adv Manuf Technol 2017;89(9–12):3307–21. http://dx.doi.org/10.1007/s00170-016-9306-9.

[56] Leusin ME, Frazzon EM, Uriona Maldonado M, Kück M, Freitag M. Solving the job-shop scheduling problem in the industry 4.0 era. Technologies 2018;6(4):107. http://dx.doi.org/10.3390/technologies6040107.

[57] Dias LS, Pattison RC, Tsay C, Baldea M, Ierapetritou MG. A simulation-based optimization framework for integrating scheduling and model predictive control, and its application to air separation units. Comput Chem Eng 2018;113:139–51. http://dx.doi.org/10.1016/j.compchemeng.2018.03.009.

[58] Negri E, Ardakani HD, Cattaneo L, Singh J, Macchi M, Lee J. A digital twin-based scheduling framework including equipment health index and genetic algorithms. IFAC-PapersOnLine 2019;52(10):43–8. http://dx.doi.org/10.1016/j.ifacol.2019.10.024.