Hindawi
Mathematical Problems in Engineering
Volume 2022, Article ID 8513719, 17 pages
https://doi.org/10.1155/2022/8513719

Research Article

Hyperparameter Tuning of Machine Learning Algorithms Using Response Surface Methodology: A Case Study of ANN, SVM, and DBN

Warut Pannakkong,[1] **Kwanluck Thiwa-Anont**,[1] **Kasidit Singthong**,[1] **Parthana Parthanadee**,[2] **and Jirachai Buddhakulsomsiri**[1]

[1]*School of Manufacturing Systems and Mechanical Engineering, Sirindhorn International Institute of Technology, Thammasat University, Thailand*
[2]*Department of Agro-Industrial Technology, Faculty of Agro-Industry, Kasetsart University, Thailand*

Correspondence should be addressed to Jirachai Buddhakulsomsiri; jirachai@siit.tu.ac.th

Received 14 June 2021; Revised 29 November 2021; Accepted 27 December 2021; Published 17 January 2022

Academic Editor: Kuei-Hu Chang

This study applies response surface methodology (RSM) to the hyperparameter fine-tuning of three machine learning (ML) algorithms: artificial neural network (ANN), support vector machine (SVM), and deep belief network (DBN). The purpose is to demonstrate RSM effectiveness in maintaining ML algorithm performance while reducing the number of runs required to reach effective hyperparameter settings in comparison with the commonly used grid search (GS). The ML algorithms are applied to a case study dataset from a food producer in Thailand. The objective is to predict a raw material quality measured on a numerical scale. *K*-fold cross-validation is performed to ensure that the ML algorithm performance is robust to the data partitioning process in the training, validation, and testing sets. The mean absolute error (MAE) of the validation set is used as the prediction accuracy measurement. The reliability of the hyperparameter values from GS and RSM is evaluated using confirmation runs. Statistical analysis shows that (1) the prediction accuracy of the three ML algorithms tuned by GS and RSM is similar, (2) hyperparameter settings from GS are 80% reliable for ANN and DBN, and settings from RSM are 90% and 100% reliable for ANN and DBN, respectively, and (3) savings in the number of runs required by RSM over GS are 97.79%, 97.81%, and 80.69% for ANN, SVM, and DBN, respectively.

1. Introduction

Nowadays, machine learning (ML) algorithms have become an important part of various industries. ML algorithms provide a significant capability to perform or facilitate various tasks. In manufacturing, a company can gain some benefits from ML in terms of performance and efficiency in a wide variety of aspects. For example, ML can be used to reduce the labor cost, human error, and number of product defects or to increase the production rate. Another advantage is that ML algorithms can handle a large amount of input data for model training [1]. The main reason for ML algorithms' growing use is that ML algorithms can improve productivity and efficiency by automating them in the usage environment [1, 2]. Also, ML algorithms can learn from previous experience by discovering patterns in existing data and using those patterns to develop and/or improve their knowledge over time [3]. These benefits of ML algorithms can lead to business revenue and growth.

There are various tasks that ML algorithms can perform. Among them, the most common tasks are classification, clustering, and prediction [4]. Classification is supervised learning that involves predicting or labeling a data class. Training an ML algorithm for classification requires an input dataset with predefined classes. Clustering, an unsupervised machine learning task, involves grouping data into an

unknown number of clusters according to some common aspects. Prediction involves estimating some quantitative (numerical) label of future observations by using a mathematical or empirical model. Due to the highly competitive environment of today's business, an accurate prediction model is needed to support the decision-making process, which impacts the business performance. Prediction is used in a wide range of applications today. To select a prediction model, its accuracy is one of the most important criteria for a modeler. For ML algorithms, an important issue in constructing effective prediction models is that the hyperparameters of the algorithms must be appropriately set [5, 6].

Hyperparameter settings are a prespecified set of decisions for ML models that directly affect the training process and the prediction result, which reflects an ML's model performance [6, 7]. Model training is a learning process so that a model can recognize patterns in the training data and predict the output of new data based on these patterns. In addition to hyperparameter setting, model architecture, which indicates the complexity of the model, would also directly impact the execution time to train and test a model. Due to their impacts on the model performance and that the best set of values is unknown, hyperparameter setting has become an important and challenging problem in ML algorithm usage. In the literature, there are several methods for tuning hyperparameters. These methods are briefly discussed below.

Every hyperparameter has a wide range of values for a modeler to select. Without using an efficient tuning strategy, simple approaches are to use the default hyperparameter values suggested by an ML algorithm developer or the recommended values from other research studies in the literature. Also, a manual search can be performed, or values based on the modeler's experience can be used [5, 8]. While using default values and suggested values from previous studies require little effort for hyperparameter tuning, these values may not perform well on different sets of input data [5]. This is because every ML algorithm, when it is applied to a specific dataset, has its own best set of hyperparameter values that may change upon different or updated input data [6, 7, 9, 10]. Therefore, most researchers state that ML algorithms are more effective if their hyperparameters have been tuned [11].

In addition, using values based on a modeler's expertise requires the modeler to have some practical experience in using particular ML algorithms and that the modeler possesses enough knowledge of the application in consideration. In most cases, these requirements are difficult to fulfill. Therefore, efficient and systematic step-by-step tuning strategies that even an inexperienced modeler can follow are clearly more desirable than depending on the modeler's experience.

A sophisticated approach is to find optimal hyperparameter settings for the learning process of the ML algorithms using some optimization techniques. This approach treats hyperparameter setting as an optimization problem, where the objective function is to maximize the ML algorithm performance, i.e., minimizing the errors from ML algorithms, while treating hyperparameters as decision variables. Examples of these methods are genetic algorithm (GA) [11–16], particle swarm optimization (PSO) [17–19], Bayesian optimization [18, 20–22]. Although this approach can yield high ML algorithm performance, such optimization or search algorithms can be difficult to use and may not be available to inexperienced modelers.

Many recent studies propose various techniques to find the set of optimal hyperparameters. These include grid search (GS) [23], random search (RS) [8], or Latin hypercube sampling. Among them, GS is simple, easy to use, and requires little experience from the modeler [8, 11, 12]. With GS, all possible combinations of hyperparameter values are tested to find the best settings based on the specified lower and upper bounds of each hyperparameter, along with a specified step size that forms the hyperparameter space [11].

Since GS executes all possible combinations, many research articles regard this method as an exhaustive search. A weakness of GS is that the combination of hyperparameters increases multiplicatively as the number of hyperparameters increases. This makes GS extremely time-consuming and requires a high computational cost [11–13, 19, 24, 25]. In most situations with limited computational resources, GS may not be an appropriate technique for hyperparameter setting because of its computational requirements.

In addition to the inefficiency of GS, another issue of using GS is that for some ML algorithms, e.g., artificial neural network (ANN) and deep belief network (DBN), rerunning the algorithms with the same hyperparameter settings yields different prediction results. This implies that the best hyperparameter settings found from performing GS one time are likely different from the best hyperparameter settings from another round of GS. This uncertainty is caused by the nature of the ML algorithms. In this study, we refer to this issue as the reliability of hyperparameter setting. To obtain reliable hyperparameter settings, some studies implement k-fold cross-validation as a means to protect against randomness in the ML training process. Moreover, the k-fold cross-validation can also avoid the bias in data separation (i.e., training, validation, and test sets).

An alternative to GS is to perform a random search (RS), which randomly selects the values of hyperparameters [8, 18, 19]. Based on the principle that among a large number of hyperparameters, there are only a few important tunable ones, and RS has been shown to perform better than GS by testing some ML algorithms on some datasets [8]. However, applying this finding to other ML algorithms, applications, and datasets may be speculative.

To overcome the inefficiency of GS, designs of experiment (DOE) techniques have been proposed to find the optimal hyperparameter values of ML algorithms [7, 26, 27]. DOE is a mathematical and statistical approach, which is used to evaluate the effects of multiple experimental factors at the same time. DOE involves planning a series of experiments for hyperparameter testing, where each experiment is a set of experimental runs at different values of hyperparameters that should be tested together. After the experiments are run, a statistical analysis of experimental data is performed to estimate the effects of the

hyperparameters on the performance of the ML algorithms. In other words, an empirical model is constructed that relates the performance of ML algorithms, e.g., prediction errors (as response variable) to hyperparameters (as predictors of ML performance). Analyzing the results from an experiment can identify significant factors and their interactions, leading to subsequent experiments that should be run [28, 29]. Some important designs include factorial experiments for a small number of hyperparameters, fractional factorial experiments for a larger number of hyperparameters (up to 15), and response surface methodology (RSM) for numerical hyperparameters [7]. RSM is an efficient technique in DOE that can be used for optimizing the values of hyperparameters [30]. By using RSM, optimal (or near-optimal) settings can be found with a small number of runs [31]. This is because RSM can provide information on an effective search path that can move toward the optimal region [32].

In this study, we propose using RSM with k-fold cross-validation to search for optimal (or near-optimal) hyperparameters of ML algorithms. We hypothesize that RSM is similar to GS in terms of ML performance and can give reliable hyperparameter settings. It is also more efficient in terms of the number of experimental runs. Both GS and RSM are performed to find the set of optimal hyperparameters of the three ML prediction models, ANN, support vector machine (SVM), and DBN, using a dataset obtained from an industrial user. The ML model performance, including prediction accuracy and the number of runs, is then compared. The contributions of our study are that we demonstrate from extensive testing the following advantages of using RSM: (1) the approach is applicable to important ML algorithms with numerical hyperparameters, and their infrastructure is specified by the number of neurons in the hidden layer, which is also a numerical factor, (2) it is a step-by-step procedure that does not depend on the ML modeler's experience and knowledge in hyperparameter setting.

2. Problem Statement

This article demonstrates the process of hyperparameter tuning using RSM for ML algorithms for predicting a numerical response variable. The tuning process is a sequential experimentation, consisting of designing a set of initial experiments and experimental data analysis, leading to subsequent experiments. The process continues until an optimal (or near-optimal) set of hyperparameter values is obtained. The hyperparameter tuning process using RSM is also compared with grid search (GS) for the ML algorithm performance, a number of experimental runs, and computational time. Both techniques are applied with three different ML algorithms: an artificial neural network (ANN), support vector machine (SVM), and deep belief network (DBN). Descriptions of the algorithms are briefly discussed next.

2.1. Artificial Neural Network (ANN). Artificial neural network (ANN) was originated in 1943 by McCulloch and Pitts [33]. They proposed an ANN in terms of a mathematical model imitating the brain's neuron connections. A feed-forward neural network (FFNN), a well-known and widely used neural network, has a structure that usually consists of three types of layers: input, hidden, and output layers. The feed-forward term represents one-way direction information flow from the input layer to the output layer. Each node is connected with the other nodes in the different layers by weights.

The learning mechanism of FFNN is the back-propagation algorithm, as shown in Figure 1. The weights and the biases are changed by recommendations from the gradient descent algorithm. The number of cycles that the weights and biases are optimized is called the training cycles [34, 35].

The hyperparameters of FFNN are the numbers of input nodes, numbers of hidden layers and their respective hidden nodes, learning rate, and training cycles. A greater number of layers and nodes can capture more complex relationships between inputs and an output; however, this requires a time-consuming training process. In addition, more complex relationships need a smaller learning rate and greater training cycles, and vice versa. The duration of the training process is proportional to the training cycles.

2.2. Support Vector Machine (SVM). Support vector machine (SVM) was introduced by Vapnik in 1995 [36]. SVM is a supervised ML algorithm for classification and regression problems in various industries such as medicine, banking, beverages, and traffic flow [37–39] because it promises outstanding performance and avoids the overfitting problem that can occur in an ANN.

Originally, SVM could only handle linear problems as it constructs a linear hyperplane to classify the dataset objects. However, nonlinear problems can be analyzed by SVM after data transformation by a kernel function (transformation function). This function is used to transform the points of data to the required form (Figure 2). Nevertheless, the kernel function must be properly selected according to the type of dataset. Otherwise, SVM cannot provide an accurate prediction [40, 41].

The hyperparameters of SVM are complexity constant (C) and convergence epsilon (Conv). The complexity constant (C) represents the trade-off between minimization of the misclassified observations and margin maximization. The higher the value of C, the narrower the margin that causes less misclassification. Conversely, the lower the value of C, the wider the margin that leads to higher misclassification. The convergence epsilon (Conv) influences the precision on the Karush–Kuhn–Tucker (KKT) condition that is the stopping criteria of the hyperplane optimization. The lower the value of Conv, the more the precision of KKT condition, and the more the iterations of hyperplane optimization.

2.3. Deep Belief Network (DBN). A deep belief network (DBN) was created by Hinton in 2006 [42], which is a combination of an artificial neural network (ANN) and a restricted Boltzmann machine (RBM). DBN can be applied

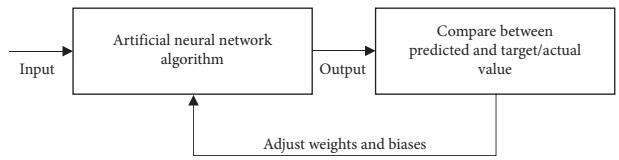

Figure 1: Backpropagation algorithm.

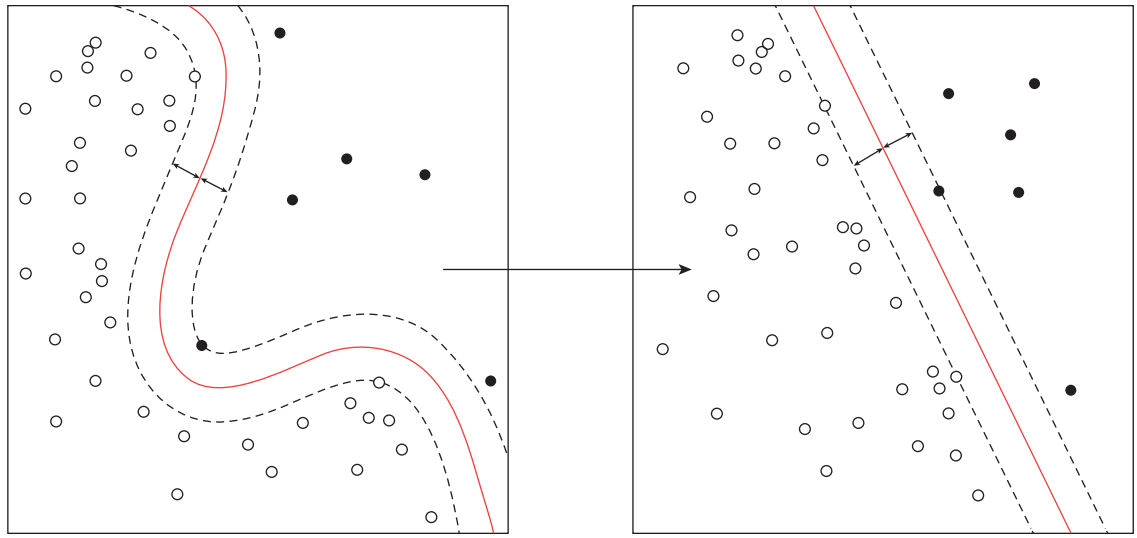

Figure 2: Diagram of (a) SVM with nonlinear hyperplane; (b) SVM with linear hyperplane (source: Alisneaky, 2011).

as a prediction model in various fields (i.e., prediction for reservoir landslide displacement and prediction for deformed coal) and some other applications such as image recognition, speech recognition, and clustering. [43, 44].

The structure of a DBN is similar to an ANN, but its initial weights are initialized by RBM in the pretraining stage before the fine-tuning stage (backpropagation algorithm). The weights from RBM can shorten the computational time in the fine-tuning stage because the computational time of the backpropagation algorithm depends on the initial values of connection weights which are randomly defined in a typical ANN.

The hyperparameters of RBM in the pretraining stage are the learning rate and training cycles of RBM. The hyperparameters (same as in ANN) consist of the number of input nodes, a number of hidden layers and their respective hidden nodes, learning rate, and training cycles of the backpropagation algorithm in the fine-tuning stage. Typically, the learning rate of RBM is larger than the backpropagation algorithm, but the number of training cycles of RBM is smaller than the backpropagation algorithm, as RBM uses a greedy algorithm [45]. The architectures of DBN and RBM are shown in Figure 3.

2.4. ML Algorithm Performance Measure. We evaluate and compare the model performance of the two methods for the best hyperparameter settings. This involves using response surface methodology (RSM) and grid search (GS) by comparing the accuracy of the prediction model. In this study, the mean absolute error (MAE) is a key factor to measure and compare the performance between these two approaches. The MAE can be expressed as follows:

$$\text{MAE} = \sum_{i=1}^{N} \frac{|yi - \widehat{y}i|}{N}, \tag{1}$$

where y_i and $\widehat{y}_i$ denote the actual value and predicted value of observation i, respectively, e_i denotes the prediction error of observation i, and N denotes the total number of observations in the data. The lower the MAE, the better the model performance.

2.5. Data Partitioning and Cross-Validation. In general, constructing an ML model involves the following steps: constructing an initial model, setting the hyperparameter values, and testing the model performance. To perform these steps, an input dataset is usually partitioned into three datasets. These datasets are the training set for initial model construction, validation set for the tuning hyperparameter, and testing set for evaluating the model performance.

In addition, k-fold cross-validation is applied for the data partitioning process. k-fold cross-validation is a widely used technique that is applied to building and testing models in machine learning. This technique randomly splits a dataset into k subsets of the same size. The value of k is specified by the modeler. Each subset takes turn to be in the training, validation, and testing sets. In other words, the training set contains $k-2$ subsets, and the validation and testing sets require one subset each. For k-fold cross-validation, an ML algorithm is constructed, validated, and tested k times.

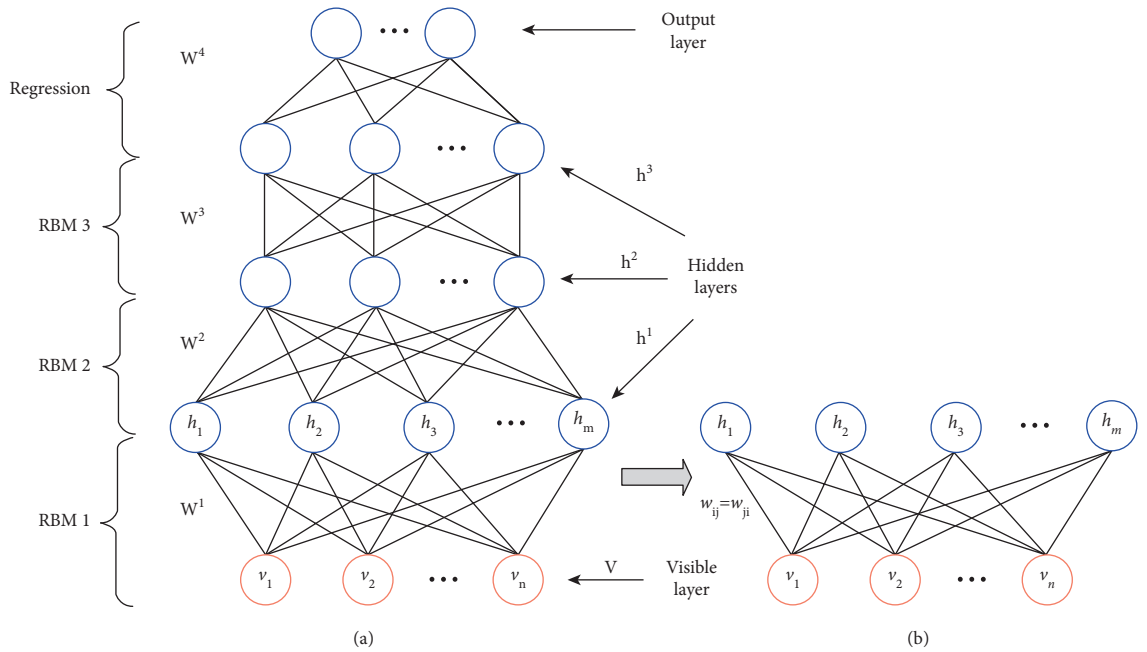

Figure 3: Diagram of (a) deep belief network (DBN) architecture and (b) restricted Boltzmann machines (RBM) (source: Shao et at., 2018).

3. Hyperparameter Tuning with RSM

RSM is a design of experiments (DOE) technique that generates a small number of experimental runs to evaluate the causal effects of experimental factors on a response variable [46]. Experimental data are used to fit a regression model that relates the response variable to the experimental factors. Information from the regression model would guide the search process in subsequent experiments toward the region of factor values that optimize the response variable. For ML algorithms, the response variable is the ML performance measure, i.e., MAE in this study, and experimental factors are the hyperparameters of the ML algorithms. The procedure for using RSM for hyperparameter settings implemented in this study is shown in Figure 4.

In Step 1, a list of hyperparameters and their ranges is specified for each ML algorithm. Then, in Step 2, an experiment is designed to generate a number of experimental runs. Each run consists of a set of hyperparameter values that are used to construct the ML model in Step 3, where the prediction performance of the ML model is measured for the training, validation, and test sets. In Step 4, the regression model that relates the prediction performance to the hyperparameters is fitted. For fine-tuning, only the prediction performance of the validation set is used in this step. If the regression model contains only significant linear terms, then the tuning process has not reached the optimal region of hyperparameters. In this case, the path of steepest descent that would improve the ML model performance is from the coefficients of the variables representing the hyperparameters in the regression model. Subsequent experiments that move the hyperparameter values along this path are run until no further improvement can be obtained (then, go to Step 2). However, if the regression model is a second-order model, then the regression model is optimized to find a set of optimal hyperparameter values. Finally, some confirmation runs are performed to verify the ML model performance at the optimal settings.

In this study, the experimental design that is applied in Step 2 is the central composite design (CCD). The CCD is an efficient design commonly used for constructing a quadratic (second-order) model. Using the CCD, a small number of experiments are performed to optimize the response variable. A CCD generally contains three levels of each factor, as shown in Figure 5. The three types of design points are the factorial points (±1) or corner points, axial points (±α), and some replications of the center points (0).

From Figure 5, each factor in a CCD has five levels. Two levels are associated with factorial points, two levels with axial points, and one level at the center. A design space, bounded by factorial points and axial points, is the region of hyperparameters under study. The design space that contains a second-order model of the response variable is the region that has the minimum, maximum, or saddle point.

For each factor, factorial points are low and high values of interest for the factor. Axial points are design points that set all factors at the center, except for one factor. With axial points, a quadratic model of the response variable can be constructed, and the curvature effects of experimental factors can be estimated [30,47]. Setting axial points requires the modeler to set an alpha (α) value. This value is the distance between each axial point and the center point. In this study, the α value is specified as 1.682 for a three-factor experiment (ANN), and 1.414 and 2.378 for a two-factor experiment (SVM) and a six-factor experiment (DBN), respectively. The value of α, which mainly depends on the number of factors, can be found using the following equation:

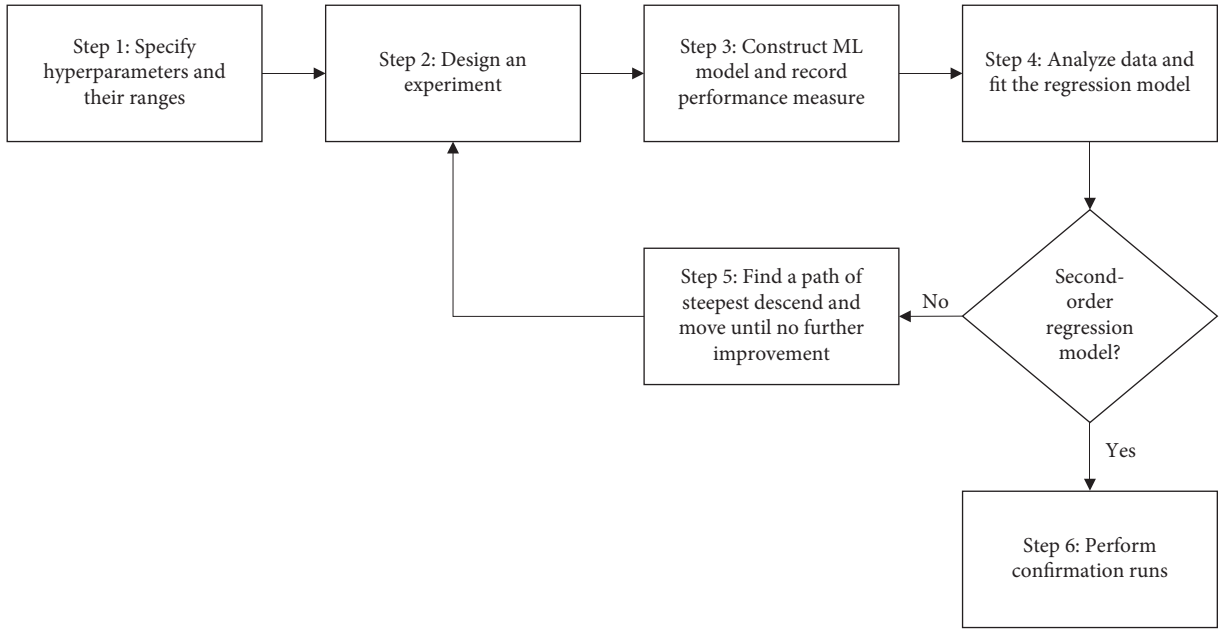

Figure 4: General procedure of RSM.

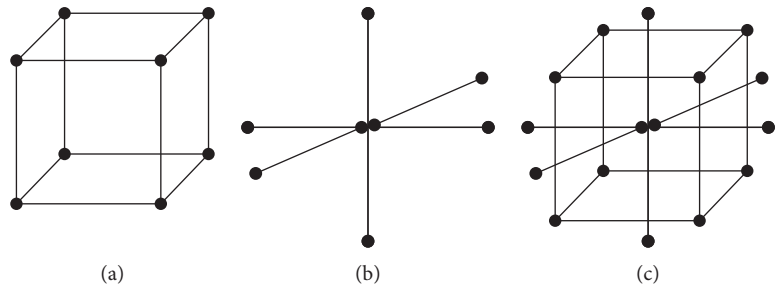

Figure 5: Central composite design (CCD) with three factors: (a) factorial points, (b) axial points and center points, and (c) central composite design.

$$\alpha = [\text{number of experimental runs}]^{1/4}. \quad (2)$$

For the center points, the value is set halfway between the low and high levels of each factor. Replications at the center point are usually performed to provide a measure of the variability of the prediction model, as well as to check for the curvature of the response surface.

In this study, the total number of experimental runs of each ML algorithm depends on the number of hyperparameters. For two-factor CCD (the number of hyperparameters $k = 2$ for SVM), there are $2^k = 4$ factorial points (full factorial), $2k = 4$ axial points, and usually five replications at the center point, for a total of 13 design points for each kernel function. For three-factor CCD ($k = 3$ for ANN), there are 20 design points: $2^k = 8$ factorial points (full factorial), $2^k = 6$ axial points, and six replications of center points. For six-factor half-fraction CCD ($k = 6$ for DBN), there are a total of 53 runs, which are $2^{k-1} = 32$ factorial points (half fraction of a full factorial), $2k = 12$ axial points, and nine center points. In comparison, the number of runs for two-factor, three-factor, and six-factor experiments by GS, with five levels in each factor, is $5^2 = 25$ runs, $5^3 = 125$ runs, and a prohibitive $5^6 = 15,625$ runs, respectively. It can be seen that the CCD offers significant savings in the number of experimental runs when compared to GS. Specifically, a general k-factor CCD contains 2^k factorial points, $2k$ axial points, and n replications at the center, for a total of $(2^k + 2k + n)$ runs, rather than the 5^k runs in GS.

4. Computational Experiment

4.1. Industrial Application and Dataset. A computational experiment is conducted on a dataset obtained from an industrial user, the largest producer and distributor in the canned fruit industry in Thailand. The data are collected at the raw material (RM) receiving operation from one of the main products. This operation is located at the beginning of the production line. The dataset contains observational data over a period of one year from over two thousand batches of RM.

To the user, an important characteristic of the packing medium, which must be prepared in advance, is that it has to match the degree of Brix of the incoming RM. The degree of Brix is, therefore, the dependent variable, of which an

accurate prediction would lead to significantly less packing medium preparation cost and lead time. A number of factors can be observed before the arriving RMs are put into production including RM harvest month, color, geographical variables, supplier, and source data. These factors are used as independent variables in the ML prediction models. The dataset is partitioned into 10 subsets for cross-validation purposes, where eight subsets are used as the training set and one subset each as the validation and test sets.

4.2. Hyperparameter Ranges for GS. First, the list of hyperparameters for the three ML algorithms is given in Table 1, along with their initial input ranges. All experimental runs are based on this information.

ANN hyperparameters include the number of hidden nodes (HN), number of training cycles (TC), and learning rate (LR). Using GS, the value of each hyperparameter starts at its lower bound and increases with a fixed step size until reaching the upper bound. With the given number of steps, the total number of experimental runs from GS with 10-fold cross-validation is $(10 \times 21 \times 21 \times 10) = 44{,}100$ runs.

The SVM contains two hyperparameters, which are the complexity constant and convergence epsilon. In addition, a user can specify the type of kernel function of SVM, so it is also treated as a factor in our experiment. Four types of kernel functions are considered, including ANOVA, dot, radial, and Epanechnikov. For SVM, the total number of experimental runs from GS with 10-fold cross-validation is $(102 \times 11 \times 4 \times 10) = 44{,}880$ runs.

For DBN, the hyperparameters include the number of hidden nodes (HN), number of training cycles during pretraining (n_i) and training (n), learning rate of pretraining and training ($\mathrm{LR}_{\mathrm{RBM}}$ and LR, respectively), and batch size (N_{b}). Since there are six hyperparameters, the number of steps for each hyperparameter is set to 3, and the total number of experimental runs from GS with 10-fold cross-validation is equal to $(3^6 \times 10) = 7{,}290$ runs.

4.3. GS Results. The 10-fold cross-validation requires that all experimental runs for the three ML algorithms are to be performed 10 times. For each fold, the best hyperparameter settings for each algorithm are the one that results in the minimum MAE. Tables 2–4 contain the best sets of hyperparameters from all 10-fold cross-validation results for ANN, SVM, and DBN, respectively.

In the table, the MAEs of the validation dataset of all 10 folds are given. The reported hyperparameter values for each fold represent the best setting among 4,410 experimental runs. The best values of HN and TC can vary from the lower bound to the upper bound, i.e., HN from 2 to 9 and TC from 10 to 1,000 (most are 500 or more). LR is the only hyperparameter where the best values are clustered around the lower bound, from 0.005 to 0.02, with the most frequent value at 0.005. The results suggest that the best and reliable setting of LR is relatively low, while the best values of the HN and TC are inconclusive.

It is important to note that each value of MAE (from cross-validation) is based on one experimental run at their respective settings. Also, rerunning an ANN at the same hyperparameter setting gives different MAE results due to the nature of ANNs. Therefore, to test the reliability of the best settings found from the 10-fold cross-validation, an ANN with each of these settings is rerun as confirmation. The number of confirmation runs is selected such that the 95% confidence interval of the average MAE of the confirmation runs is within 5% of the mean, which results in eight confirmation runs.

After performing the confirmation runs, the average MAE of the validation set and its SD are reported for each hyperparameter setting. It was found that the average MAEs from confirmation runs are not as good as the MAEs value obtained from cross-validation GS due to the natural variability of the ML algorithm performance. As seen in Table 2, the minimum value of MAE (0.95) from the 1st fold of cross-validation is different from the average of the MAE (1.03) from eight confirmation runs. To properly evaluate the reliability of the hyperparameters from GS, we perform "prediction interval (PI) coverage" in this study. The PI coverage uses the half-width of the 95% PI. We define the performance of GS as reliable if the PI coverage is satisfied, i.e., the MAE from one run (cross-validation) is within the half-width of the 95% PI from the average MAE of the confirmation runs. Finally, the PI coverage of the hyperparameter settings found from GS is satisfied in 8 out of 10 folds. This indicates reasonable (80%) reliability of the ANN hyperparameter values found from GS.

Table 3 provides the best hyperparameter settings from 10-fold cross-validation of SVM. In 7 out of 10 folds, setting the kernel function as ANOVA and a low value of the complexity constant (−1 or 1) is reliable, whereas the best coverage values tend to vary. SVM is an ML algorithm based on statistical learning, where performing a rerun of SVM at the same hyperparameter setting would give the same prediction value, i.e., the same prediction performance. Therefore, there is no need to do the confirmation runs.

Table 4 provides DBN results in the same fashion as Table 2. Most of the best hyperparameter settings are an HN of 10 (high value), n_i and n are either 500 or 1000 (medium or high value), $\mathrm{LR}_{\mathrm{RBM}}$ and LR are either 0.05 or 0.10 (medium or high value), and the batch size is 1,025 (medium value). In 8 out of 10 folds, the PI coverage is satisfied, which indicates 80% reliability of hyperparameter settings of DBN found from GS.

As mentioned earlier, due to GS multiplicative nature, the required number of runs is large especially when k-fold cross-validation is used. For an ANN with three hyperparameters, the 10-fold cross-validation in our testing requires 44,100 runs. The GS search for the ANN is reasonable because each hyperparameter is searched in over 10 steps. However, for a DBN with six hyperparameters, the same level of detial for GS search is prohibited, i.e., more than 10_6 runs for each fold. The same level of detail for GS search is prohibited, i.e., more than 10^6 runs for each fold. That means the search space of GS for DBN is not well explored. In other words, GS may be ineffective for an ML algorithm with many hyperparameters due to computational resource limitations.

Table 1: List of ANN hyperparameters and their ranges.

ML	Hyperparameter	Description	Lower	Upper	No. of steps
ANN	HN	No. of hidden nodes	1	10	10
	TC	No. of training cycles	10	1000	21
	LR	Learning rate	0.0001	0.1	21
SVM	C	Complexity constant	−1	100	102
	Conv	Convergence epsilon	0.001	0.1	11
DBN	HN	No. of hidden nodes	1	10	3
	n_i	No. of pretraining cycles	10	1000	3
	n	No. of training cycles	10	1000	3
	LR_{RBM}	Pretraining learning rate	0.0001	0.1	3
	LR	Learning rate	0.0001	0.1	3
	N_b	Batch size	350	1700	3

Table 2: Best hyperparameter setting and the test performance of ANN.

	Grid search				Confirmation runs			PI coverage
Fold	HN	TC	LR	MAE (validation)	Avg MAE (validation)	SD of MAE (validation)	95% PI half-width	
1	**7**	**604**	**0.005095**	0.9513	1.0543	0.0605	0.1517	Yes
2	6	802	0.005095	1.0333	1.1461	0.0571	0.1432	Yes
3	9	208	0.005095	0.9637	1.0327	0.032	0.0803	Yes
4	9	852	0.005095	0.9743	1.0706	0.0458	0.1149	Yes
5	5	1000	0.005095	0.9621	1.0731	0.0188	0.0472	No
6	3	10	0.01009	0.9569	1.0265	0.0441	0.1106	Yes
7	9	654	0.01009	1.1191	1.1956	0.0431	0.1081	Yes
8	6	505	0.02008	1.0143	1.2014	0.0713	0.1788	No
9	7	951	0.005095	0.9894	1.0938	0.0538	0.1349	Yes
10	2	604	0.015085	1.0633	1.2126	0.0606	0.152	Yes

Bold values indicate the best set of hyperparameters from Grid Search that leads to the minimal MAE (validation).

Table 3: Best hyperparameter setting and the test performance of SVM.

Fold	Kernel function	C	Conv	MAE (validation)
1	*ANOVA*	1	0.0109	1.0668
2	*Radial*	3	0.0307	1.1237
3	*ANOVA*	2	0.0703	1.052
4	*Radial*	75	0.001	1.0492
5	*ANOVA*	1	0.1	1.061
6	*Dot*	−1	0.001	0.9971
7	*ANOVA*	1	0.0406	1.1648
8	*ANOVA*	−1	0.0901	1.105
9	*ANOVA*	−1	0.0208	1.0693
10	*ANOVA*	1	0.0802	1.1319

Table 4: Best hyperparameter setting and the test performance of DBN.

	GS						Confirmation runs			
Fold	HN	n_i	n	LR_{RBM}	LR	N_b	MAE (validation)	Avg MAE (validation)	SD of MAE (validation)	95% PI half-width	PI coverage
1	10	1000	505	0.1	0.05005	1025	0.987	1.0369	0.0318	0.0832	Yes
2	10	1000	10	0.05005	0.1	1025	1.0906	1.119	0.0383	0.1002	Yes
3	10	1000	1000	0.05005	0.05005	350	0.9718	1.01	0.0219	0.0573	Yes
4	10	505	505	0.1	0.05005	1025	1.0092	1.0647	0.0352	0.0921	Yes
5	10	505	1000	0.1	0.05005	1700	0.9978	1.0462	0.0456	0.1193	Yes
6	10	1000	10	0.1	0.1	1025	0.9313	1.0063	0.0246	0.0644	No
7	10	505	1000	0.1	0.1	1025	1.112	1.1737	0.038	0.0994	Yes
8	10	505	505	0.1	0.05005	1025	1.0361	1.0736	0.0333	0.0871	Yes
9	10	505	1000	0.1	0.1	1025	0.9893	1.0675	0.0277	0.0725	No
10	5.5	1000	505	0.05005	0.05005	350	1.0446	1.1091	0.0251	0.0657	Yes

4.4. RSM Results

4.4.1. Artificial Neural Network. From the initial ranges of hyperparameters, listed in Table 1, the search process begins at the low values of these hyperparameters, i.e., near the origin point of the hyperparameter space. The lower bound and upper bound are specified for the axial points of the CCD. Since HN and TC are integers, the values of the hyperparameters at factorial points are rounded to the nearest integer.

Experiment 1. Identifying the steepest descent direction

A three-factor CCD with 20 design points is performed for each data fold. Each row in the table represents an experimental run, where point type of 1, −1, and 0 indicates that the experiment is a factorial point, axial point, and center point, respectively. Note that, this CCD is constructed by specifying the axial points for each hyperparameter as follows: 1 and 3 for HN, 10 and 100 for TC, and 0.0001 and 0.001 for LR. Also, the values of HN of some design points are rounded to the nearest integer since HN cannot be a fractional number. The MAEs of the validation set of all 10 folds of data are shown in Table 5.

Analysis of variance (ANOVA) is performed on each data fold separately. The steps in the data analysis of the second fold are described as an example. First, the regression model begins as a full-quadratic model, which contains the main effects, squared terms, and two-factor interaction terms. Then, insignificant terms are eliminated from the regression model, one at a time, until the final model is obtained. The regression model fitting process is conducted, while maintaining the hierarchy of the model, i.e., keeping lower-order term(s) in the presence of a significant higher-order term.

The ANOVA from the final model of the second data fold is shown in Table 6. The linear term and the quadratic term of TC are significant, while the HN and LR effects are not. This may be caused by the ranges of HN and LR being too narrow. Further analysis as shown in Figure 6 suggests that a higher value of HN could reduce the MAE, and a high value of LR may already be effective. Therefore, the search direction of subsequent experiments needs to increase HN and LR, while fixing TC. A similar analysis is performed on the other nine data folds, which results in nine additional ANOVA tables and regression models. After examining the results of all 10 folds, four patterns of hyperparameter effects on the MAE of the validation set are summarized in Table 7, along with a suggested search direction.

We denote a quadratic effect of a hyperparameter as convex if the value of the hyperparameter that is associated with a low value of MAE occurs in the middle of the input range. A concave pattern indicates the opposite hyperparameter effect where low values of MAE occur at the boundaries of the input range. In addition, there are two linear trends, a negative trend and a positive trend, which point to a low value of MAE occurring at the upper bound and lower bound, respectively. These patterns also suggest the search directions. For a convex pattern and linear positive trend, the hyperparameter values remain unchanged at the value that gives the minimum MAE and at the lower bound, respectively, in subsequent experiments. However, for a concave pattern and linear negative trend, the search direction is to increase the hyperparameter value. The search direction is defined by the coded coefficients of the final regression model. For example, the final regression model from the second fold of data is as follows:

$$\begin{aligned}\text{MAE} = {} & 1.1343 - 0.0006\,HN - 0.0085\,TC - 0.0016\,LR \\ & + 0.0050\,TC2.\end{aligned} \tag{3}$$

Based on the coded coefficients in the model, every coded unit increase in HN is associated with (-0.0000/-0.0016) = 2.6667 coded units increase in LR. The 2.6667 coded units have to be transformed to an original scale. Recall that in the original scale, LR = 0.00028 is coded as −1 at the factorial point, and LR = 0.00055 is coded as 0 at the center points. In other words, a coded unit increase is equivalent to (0.00055–0.00028) = 0.00027 units in LR. Therefore, in the steepest descent direction, an increase of 2.6667 coded units becomes a $(0.00027 \times 2.6667) = 0.000714$ increase in LR.

Experiment 2. Searching along the steepest descent direction

Search results along the steepest descent direction are given in Figure 7. From the results, an HN of 8 and LR of 0.0117 seem to be effective. Therefore, these settings become a new center point for the next experiment.

The process of searching along the path of steepest descent is repeated for each data fold. This process stops when the best hyperparameter values on this path are found. These settings become the center point for further experiments.

Experiment 3. Finding optimal setting

For the second data fold, another CCD is conducted. The results are shown in Figure 8.

The regression model fitting process is performed based on the MAE results and hyperparameter values in Figure 8. The ANOVA of the final model is shown in Table 8. The linear terms and quadratic terms of HN and TC are significant, while the LR effects are not significant. The final regression model is then optimized with respect to HN and TC. The best hyperparameter settings from the second data fold are HN = 8, TC = 194, and LR = 0.005095 with the expected value of MAE equal to 1.0900. Similar to the GS, eight confirmation runs are performed.

Experiment 3 is repeated for the other nine data folds. A summary of the results is shown in Table 9. The table contains the best values of the hyperparameters for all 10 folds, the validation MAE at these settings from the experiment, and the results from the confirmation run. In the last column, PI coverage indicates whether or not the 95% prediction intervals of MAE contain the validation MAE from the RSM experiments. The results show that in 9 out of 10 folds, the 95% PIs contain the validation MAE. This indicates a 90% reliability of the hyperparameter values obtained from RSM.

Table 5: Design and experiment results of Experiment 1.

Point type	Hyperparameter			MAE (validation)									
	HN	TC	LR	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Fold 6	Fold 7	Fold 8	Fold 9	Fold 10
1	1	28	0.00028	1.1332	1.1465	1.1254	1.0984	1.1071	1.0267	1.2692	1.1173	1.0874	1.1636
1	3	28	0.00028	1.1146	1.1502	1.1334	1.0932	1.0968	1.0202	1.2843	1.1294	1.0979	1.2028
1	1	82	0.00028	1.1271	1.1358	1.094	1.0736	1.0964	0.9994	1.2469	1.1184	1.064	1.1396
1	3	82	0.00028	1.0968	1.1301	1.079	1.076	1.0865	1.0024	1.2377	1.1203	1.0673	1.1515
1	1	28	0.00082	1.1943	1.1355	1.1634	1.0719	1.1013	1.0267	1.2563	1.1192	1.0741	1.1534
1	3	28	0.00082	1.1126	1.1306	1.0735	1.0753	1.0891	1.0202	1.2627	1.1166	1.0633	1.1422
1	1	82	0.00082	1.1785	1.1306	1.1399	1.0815	1.1001	0.9994	1.2433	1.1187	1.0757	1.1393
1	3	82	0.00082	1.1002	1.1338	1.0409	1.0936	1.0576	1.0024	1.2282	1.0857	1.0849	1.1382
−1	1	55	0.00055	1.1568	1.1344	1.1177	1.0728	1.0998	1.0091	1.2498	1.1186	1.0715	1.1446
−1	3	55	0.00055	1.0994	1.128	1.0669	1.0766	1.0821	1.0107	1.2496	1.1102	1.0669	1.1451
−1	2	10	0.00055	1.1334	1.1778	1.1729	1.1257	1.1336	1.0387	1.2884	1.164	1.1121	1.2212
−1	2	100	0.00055	1.0783	1.1275	1.0786	1.1046	1.0868	0.978	1.2232	1.1127	1.0793	1.1793
−1	2	55	0.0001	1.0783	1.1275	1.0786	1.1046	1.0868	0.978	1.2232	1.1127	1.0793	1.1793
−1	2	55	0.001	1.0852	1.1335	1.0873	1.1112	1.0974	0.9773	1.2438	1.1451	1.0833	1.2123
0	2	55	0.00055	1.0925	1.1483	1.1793	1.0804	1.0969	1.0005	1.2511	1.2337	1.1578	1.1438
0	2	55	0.00055	1.0925	1.1303	1.1046	1.1078	1.0865	0.9727	1.2771	1.2183	1.0833	1.1793
0	2	55	0.00055	1.0882	1.1433	1.1025	1.0948	1.0702	0.9904	1.2396	1.1337	1.0675	1.1806
0	2	55	0.00055	1.0947	1.1305	1.0741	1.0839	1.0886	1.0111	1.2367	1.1917	1.0742	1.1373
0	2	55	0.00055	1.2047	1.129	1.107	1.0889	1.1036	0.9756	1.2324	1.109	1.0673	1.143
0	2	55	0.00055	1.0777	1.1529	1.0663	1.09	1.096	0.9768	1.2828	1.1203	1.0822	1.1441

Table 6: Analysis of variance of the second fold.

Source	Df	Adj SS	Adj MS	F-value	P value
Model	4	0.001418	0.000355	3.56	0.031
Linear	3	0.001043	0.000348	3.49	0.042
HN	1	0.00001	0.00001	0.1	0.754
TC	1	0.000998	0.000998	10.02	0.006
LR	1	0.000035	0.000035	0.35	0.562
Square	1	0.000375	0.000375	3.77	0.071
TC × TC	1	0.000375	0.000375	3.77	0.071
Error	15	0.001494	0.0001		
Lack-of-fit	10	0.000947	0.000095	0.87	0.606
Pure error	5	0.000547	0.000109		
Total	19	0.002912			

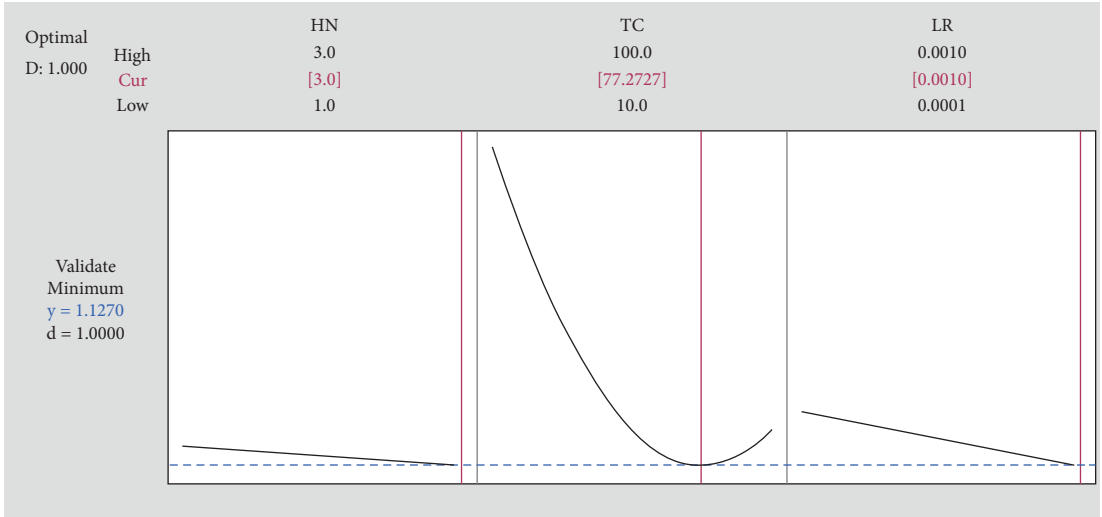

Figure 6: The response optimizer from the second fold.

Table 7: Patterns of hyperparameter effects and search direction.

Fold	Hidden nodes	Training cycles	Learning rate	Search direction
1	Convex	Linear (−)	Linear (+)	Fix HC, TC at minimum MAE and increase TC
2	Linear (−)	Convex	Linear (-)	Increase HN, LR together and fix TC at minimum MAE
3	Convex	Linear (−)	Linear (−)	Fix HN at minimum MAE and increase TC and LR together
4	Concave	Convex	Convex	Increase HN and fix TC
5	Linear (−)	Convex	Linear (−)	Increase HN and LR together
6	Convex	Convex	Linear (−)	Increase LR
7	Linear (−)	Convex	Convex	Increase HN
8	Linear (−)	Linear (−)	Convex	Increase HN and TC
9	Linear (−)	Concave	Linear (+)	Increase HN
10	Concave	Convex	Convex	Increase HN

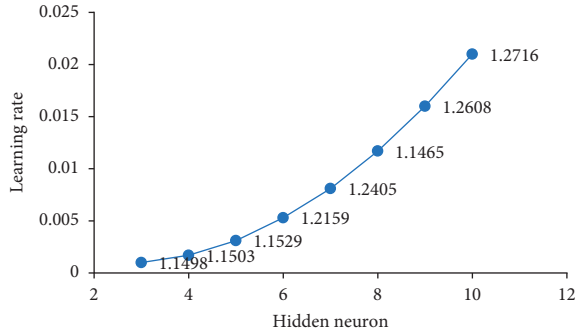

Figure 7: The search direction from the second fold.

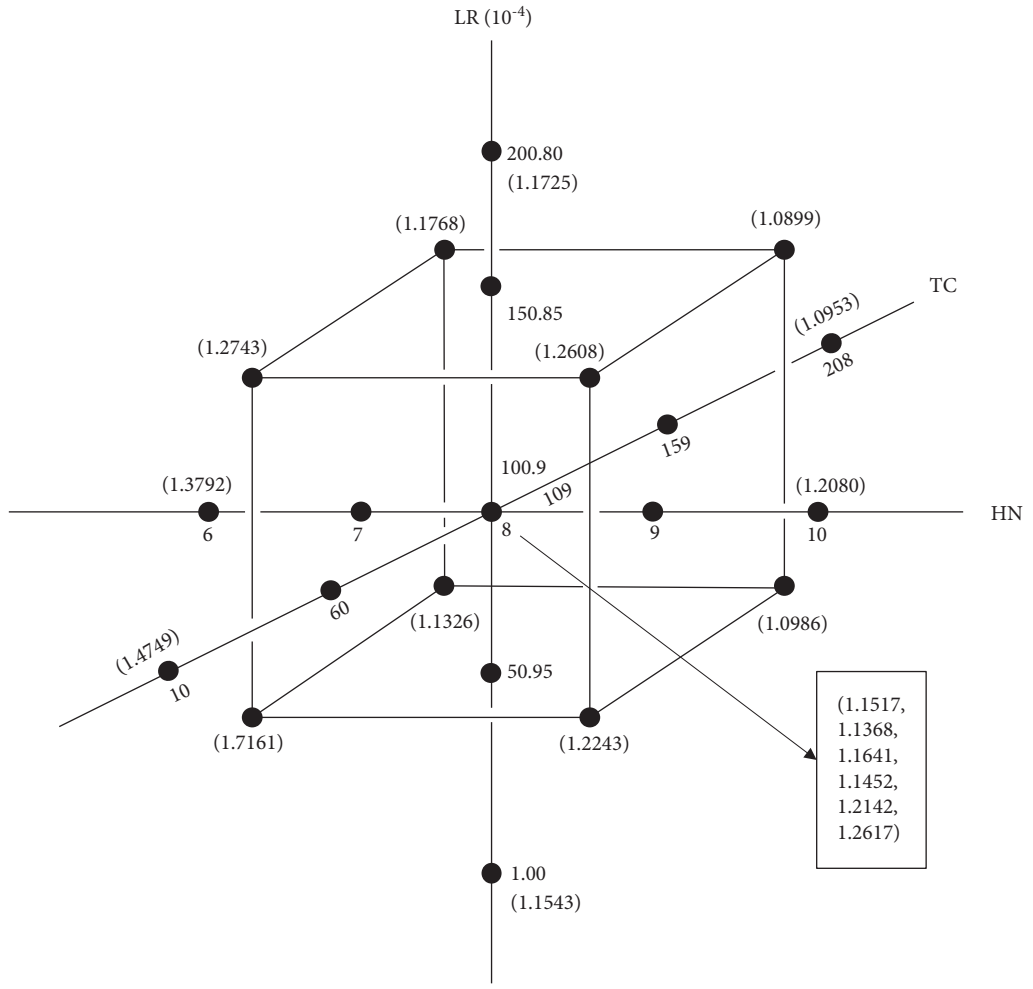

Figure 8: Design and experiment results of Experiment 3.

Table 8: ANOVA of the final model of the second data fold.

Source	Df	Adj SS	Adj MS	F-value	P value
Model	4	0.12862	0.032155	10.62	<0.001
HN	1	0.01148	0.011481	3.79	0.070
TC	1	0.08953	0.089534	29.56	<0.001
HN × HN	1	0.01762	0.017617	5.82	0.029
TC × TC	1	0.01483	0.014835	4.9	0.043
Error	15	0.04543	0.003028		
Lack-of-fit	10	0.03347	0.003347	1.4	0.373
Pure error	5	0.01195	0.002391		
Total	19	0.17405			

Table 9: Best hyperparameter setting and the validation performance of ANN using RSM.

	RSM				Confirmation runs			
Fold	HN	TC	LR	MAE (validation)	Avg MA (validation)E	SD of MAE (validation)	95% PI half-width	PI coverage
1	4	802	0.005095	1.012	1.2255	0.0567	0.1422	No
2	8	194	0.005095	1.087	1.1364	0.0503	0.1262	Yes
3	4	951	0.005095	1.034	1.0781	0.0263	0.066	Yes
4	5	258	0.005095	1.095	1.1179	0.0518	0.1299	Yes
5	10	60	0.005095	1.035	1.1535	0.0611	0.1532	Yes
6	**9**	**10**	**0.005095**	**0.98**	1.0818	0.0533	0.1337	Yes
7	5	109	0.005095	1.169	1.2794	0.0675	0.1693	Yes
8	8	456	0.005095	1.062	1.198	0.0643	0.1613	Yes
9	7	208	0.005095	1.051	1.1091	0.0348	0.0873	Yes
10	4	109	0.005095	1.113	1.2119	0.0579	0.1452	Yes

Bold values indicate that this is the combination of hyperparameters that give the minimal average MAE (validation) among all folds.

Among different folds, HN varies from 4 to 10, and TC varies from 10 to 951. The best LR is the same for all folds, which is 0.005095. Although the best settings of HN and TC are inconclusive, it is noticeable that there is a mild trend between HN and TC, i.e., lower HN tends to be associated with higher TC and vice versa.

4.4.2. Support Vector Machine (SVM). Similar to an ANN, the search starts at low values of the initial range of hyperparameter (see Table 1). Since the type of kernel function is a categorical factor, the first experiment is conducted separately for the four types of kernel functions.

Experiment 4. Identifying the steepest descent direction

Using a two-factor CCD, a data fold has 13 experimental runs for each kernel function. The total number of initial experimental runs to find paths of steepest descent is equal to $13 \times 10 \times 4 = 520$ runs. After running all experiments, the results in Table 10 reveal that *dot* and *radial* kernel functions are more effective than *ANOVA* and *Epanechnikov* functions, which, therefore, are not further considered.

The same process of RSM described in the previous section is conducted with SVM. Patterns of hyperparameter effects on MAE for *dot* and *radial* functions are summarized in Table 11, along with the data fold(s) associated with each pattern in the last two columns.

For subsequent experiments, search along the steepest descent direction.

The process of searching along the steepest descent directions to find the appropriate number of steps to move and to find the optimal settings for all 10 folds is performed in the same manner as that of the ANN. Detailed results are omitted to save space. A summary of the optimal hyperparameter settings for all 10 folds is provided in Table 12. Overall, the values of C for dot and radial functions are relatively low, while Conv is inconclusive. As mentioned previously, rerunning SVM at the same settings would give the same prediction value. Therefore, confirmation runs are not performed.

After applying the previous steps to all 10 folds, the best hyperparameters for *dot* and *radial* kernel functions are shown in Table 12. The minimum value of MAE is 1.053 for both kernel functions. For the dot function, the best hyperparameter settings are equal to 9 and 0.0307 for the C and Conv, respectively. For the radial function, C and Conv are 6 and 0.001, respectively.

4.4.3. Deep Belief Network (DBN). The initial ranges of hyperparameters in the first experiment for DBN are listed in Table 1. These ranges are also chosen to start at low values. The lower and upper bounds in Table 11 are values for the axial points. Since HN, n, and n_i are integers, their values at factorial points are rounded to the nearest integers.

Experiment 5. : A six-factor half-fraction CCD with 53 design points is performed for each data fold. For 10 folds, a total of $53 \times 10 = 530$ runs are conducted to find the paths of the steepest descent. As an example, ANOVA of the final model from the 2nd data fold is given in Table 13. The obtained regression model provides the steepest descent direction to increase LR together with n, while holding the other

Table 10: Experiment 4 results for the four kernel functions.

Kernel function	Average MAE from the validation set									
	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Fold 6	Fold 7	Fold 8	Fold 9	Fold 10
ANOVA	1.1964	1.2969	1.1607	1.1763	1.2061	1.2283	1.3041	1.2912	1.2432	1.249
Dot	**1.1021**	**1.1573**	**1.1056**	1.1024	1.1803	**1.0865**	1.32	1.2016	1.2016	1.1701
Radial	1.2124	1.2124	1.1136	**1.0758**	**1.1215**	1.1565	**1.258**	**1.1821**	**1.1479**	**1.1373**
Epanechnikov	1.1963	1.1885	1.1272	1.0887	1.1485	1.1271	1.2944	1.2157	1.1494	1.2193

Bold values indicate the minimal average MAE for each data fold (Fold 1 to Fold 10).

Table 11: Patterns of hyperparameter effects and search direction for *dot* and *radial* kernel function.

C	Conv	Search direction	*dot*	*radial*
Linear (−)	Linear (−)	Increase C and Conv together	Fold: 1, 5, 6, 10	Fold: 1
Concave	Linear (−)	Fix C at minimum MAE and increase Conv	Fold: 2, 7, 9	
Linear (+)	Linear (−)	Fix C at minimum MAE and increase Conv	Fold: 3, 8	
Linear (+)	Linear (+)	Fix C and Conv at minimum MAE	Fold: 4	
Convex	Linear (+)	Fix C and Conv at minimum MAE		Fold: 2
Linear (+)	Linear (-)	Fix C at lower bound and increase Conv		Fold: 3
Concave	Linear (+)	Fix C and Conv at minimum MAE		Fold: 4, 5, 9
Linear (−)	Linear (+)	Increase C and fix Conv at minimum MAE		Fold: 7
Concave	Linear (−)	Fix C at lower bound and increase Conv		Fold: 8, 10

Table 12: Best hyperparameter settings for *dot* and *radial* kernel functions using RSM.

Fold	*Dot*			*Radial*		
	C	Convergence	MAE (validation)	C	Convergence	MAE (validation)
1	12	0.001	1.1680	12	0.0109	1.1455
2	−1	0.1	1.1868	3	0.001	1.1237
3	−1	0.0406	1.1102	−1	0.0703	1.1575
4	-1	0.001	1.1694	**6**	**0.001**	**1.0529**
5	11	0.0109	1.1209	9	0.001	1.1137
6	**9**	**0.0307**	**1.0528**	17	0.0208	1.0892
7	−1	0.1	1.2577	11	0.001	1.2452
8	−1	0.0505	1.1234	3	0.0109	1.1604
9	−1	0.0901	1.0765	9	0.001	1.0988
10	10	0.0406	1.1923	7	0.1	1.1827

hyperparameters fixed at their boundaries or at the point where MAE is minimized if the squared term is significant.

The RSM process is conducted with DBN on 10 data folds. Patterns of hyperparameter effects on MAE and the search directions are summarized in Table 14. Moving along these steepest descent directions toward the optimal region gives the center points for the subsequent experiments. Then, running another CCD around these center points leads to the final regression models that yield the optimal settings for all 10 data folds. The best hyperparameter settings obtained from the final regression models and the validation MAE at these settings for DBN using RSM are provided in Table 15. The results from confirmation runs, including the average MAE, SD, half-width of 95% PI, and coverage of PI, are also shown. The PI coverages of the DBN hyperparameter settings from RSM indicate 100% reliability based on our testing.

4.5. Comparison between GS and RSM. ANOVA is performed to compare the confirmation MAE between GS and RSM (denoted as Method) for ANN, SVM, and DBN (denoted as ML) while treating data folds as blocks (see Table 16). The interaction between method and fold is mildly significant (P value = 0.094), which indicates that the average MAE of validation set between GS and RSM is statistically different in some data folds. In Table 17, further analysis using Tukey's multiple comparisons shows that the difference is significant only for the first data fold, i.e., the hyperparameters from GS give statistically better results (on average) than those of RSM. For the other nine data folds, the hyperparameter settings from GS and RSM perform statistically the same.

To investigate the reliability of using the best hyperparameters from RSM to predict unforeseen data, the three ML models with the best hyperparameters from GS and RSM are applied to predict the results in the testing dataset. Then, ANOVA is conducted to compare the average MAE of the testing set between GS and RSM (see Table 18). The ANOVA shows that there is no significant difference in the average MAE between the two hyperparameter tuning methods (i.e., GS and RSM). Therefore, this implies that the hyperparameter setting from RSM is reliable in the testing set as well.

Table 13: Analysis of variance of the second fold.

Source	Df	Adj SS	Adj MS	F-value	P value
Model	8	363.18	45.397	14.5	<0.001
HN	1	0.446	0.446	0.14	0.708
LR_{RBM}	1	0.652	0.652	0.21	0.650
LR	1	137.73	137.73	43.98	<0.001
N	1	14.968	14.968	4.78	0.034
n_i	1	153.99	153.99	49.17	<0.001
N_b	1	28.911	28.911	9.23	0.004
HN × HN	1	11.969	11.969	3.82	0.057
LR × n	1	14.51	14.51	4.63	0.037
Error	44	137.8	3.132		
Lack-of-fit	36	111.06	3.085	0.92	0.603
Pure error	8	26.738	3.342		
Total	52	500.98			

Table 14: Patterns of hyperparameter effects and search direction for DBN.

Fold	HN	LR_{RBM}	n	n_i	LR	N_b	Search direction
1	Linear (−)	Concave	Linear (−)	Linear (+)	Linear (−)	Linear (+)	Increase HN, N, LR, and fix LR_{RBM}, n_i, $N_b$at min. MAE
2	Convex	Linear (+)	Linear (−)	Linear (+)	Linear (−)	Linear (+)	Increase N, LR, and fix HN, LR_{RBM}, n_i, $N_b$at min. MAE
3	Linear (−)	Linear (−)	Linear (−)	Linear (+)	Linear (−)	Linear (+)	Increase HN, LR_{RBM}, N, LR, and fix n_i, $N_b$at min. MAE
4	Linear (−)	Linear (−)	Linear (−)	Linear (−)	Linear (−)	Linear (+)	Increase HN, LR_{RBM}, N, n_i, LR, and fix, $N_b$at min. MAE
5	Linear (+)	Linear (−)	Linear (−)	Linear (−)	Linear (−)	Linear (+)	Increase LR_{RBM}, N, n_i, LR, and fix HN, $N_b$at min. MAE
6	Linear (−)	Linear (+)	Linear (−)	Linear (−)	Linear (−)	Linear (+)	Increase HN, N, n_i, LR, and fix LR_{RBM}, $N_b$at min. MAE
8	Linear (−)	Convex	Linear (−)	Convex	Linear (−)	Concave	Increase HN, N, LR, and fix LR_{RBM}, n_i, $N_b$at min. MAE
9	Linear (−)	Linear (−)	Linear (−)	Linear (+)	Linear (−)	Linear (+)	Increase HN, LR_{RBM}, N, LR, and fix n_i, $N_b$at min. MAE
10	Linear (+)	Linear (+)	Linear (−)	Linear (+)	Linear (−)	Linear (+)	Increase N, LR, and fix HN, LR_{RBM}, n_i, $N_b$at min. MAE

Table 15: Best hyperparameter setting and the validation performance of DBN using RSM.

	Cross-validation							Confirmation runs		
Fold	HN	n_i	n	LR_{RBM}	LR	N_b	MAE	Avg MAE	SD (MAE)	95% PI half-width	PI coverage
1	7	389	10	0.0001	0.0069	1025	1.081	1.097	0.032	0.109	Yes
2	3	530	10	0.001	0.0107	1025	1.166	1.168	0.009	0.03	Yes
3	8	575	10	0.001	0.0067	350	1.073	1.083	0.011	0.036	Yes
4	3	254	10	0.001	0.0068	1025	1.083	1.096	0.013	0.042	Yes
5	3	209	10	0.001	0.0073	1700	1.087	1.093	0.009	0.031	Yes
6	**5**	**209**	**10**	**0.001**	**0.008**	**1025**	**1.000**	0.998	0.013	0.045	Yes
7	4	434	10	0.001	0.0094	1025	1.223	1.232	0.018	0.061	Yes
8	7	344	10	0.001	0.0071	1025	1.12	1.14	0.035	0.118	Yes
9	6	620	10	0.001	0.0101	1025	1.131	1.089	0.028	0.095	Yes
10	6	620	10	0.001	0.0101	350	1.299	1.203	0.064	0.215	Yes

Bold values emphasize that this is the combination of hyperparameters that results in the minimal average MAE (0.998).

Table 16: ANOVA of confirmation MAE of validation set by hyperparameter tuning methods.

Source	Df	Adj SS	Adj MS	F-value	P value
Fold	9	0.173434	0.01927	34.08	<0.001
ML	2	0.019289	0.009645	17.06	<0.001
Method	1	0.033192	0.033192	58.7	<0.001
Fold × method	9	0.009315	0.001035	1.83	0.094
Error	38	0.021486	0.000565		
Total	59	0.256716			

4.6. Discussion. Grid search (GS) is widely used in several studies as a common hyperparameter tuning method. As shown in section 4.3, implementing GS in ANN, SVM, and DBN requires many experiments, i.e., 44,100 runs, 44,880 runs, and 7,290 runs, respectively. The results in section 4.5 indicate that on average, the prediction performance of hyperparameter settings obtained from GS and RSM is statistically the same in all three ML models, while RSM requires a smaller number of runs. Specifically, with RSM, the number of runs for ANN, SVM, and DBN is 976 runs

Table 17: Tukey's multiple comparison of the confirmation MAE between GS and RSM.

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Fold 6	Fold 7	Fold 8	Fold 9	Fold 10
GS	A	A	A	A	A	A	A	A	A	A
RSM	B	A	A	A	A	A	B	A	A	A

Note: Means that do not share the same letter are significantly different.

Table 18: ANOVA on MAE of testing set by hyperparameter tuning methods: GS and RSM.

Source	Df	Adj SS	Adj MS	F-value	P value
Fold	9	0.1723	0.0191	10.47	<0.001
ML	2	0.0367	0.0183	10.03	<0.001
Tuning method	1	0.0044	0.0044	2.41	0.127
Error	47	0.085969	0.001829		
Total	59	0.299374			

(97.79% saving over GS), 984 runs (97.81% saving), and 1,408 runs (80.69% saving), respectively. In other words, the benefit from using RSM has been demonstrated that it can significantly reduce the number of experiments to determine the suitable hyperparameter setting while maintaining the prediction performance.

Finally, it is worth noting that RSM, by nature, involves sequential experimentation. That is, RSM requires a modeler to run some experiments and perform data analysis to determine whether the subsequent experiment is needed. If it is the case, then findings from one experiment will guide the search in a subsequent experiment. A limitation of RSM is that this search process is not automated or has not been integrated as a hyperparameter tuning process in standard ML software.

5. Conclusion

Hyperparameter tuning is an essential process in ML algorithms. To obtain proper hyperparameter settings, GS is the common approach in many applications due to its simplicity to implement but with a drawback of many experimental runs. In this study, we propose a hyperparameter tuning process using the RSM approach. RSM is compared with GS on three ML algorithms: ANN, SVM, and DBN. A 10-fold cross-validation is performed to make sure that the ML performance is robust to the data partitioning process. The dataset used in the computational test is from an industrial user in the food industry. The purpose of the analysis is to predict the quality of the raw material in the production process. The MAE is considered a key measure. The MAE from the validation set is used to evaluate the prediction performance between the two hyperparameter tuning processes.

The comparison results indicate that the widely used GS method can give hyperparameter settings that are 80% reliable for both ANN and DBN, whereas the hyperparameter tuning process by RSM can give 90% and 100% reliability for ANN and DBN, respectively. In addition, the prediction performance from GS and RSM is directly compared using statistical analysis. The results show that the hyperparameter settings obtained from RSM can give statistically similar performance to those from GS in 9 out of 10 data folds and 7 out of 10 data folds for ANN and DBN, respectively. Although the performance from GS is slightly better than RSM, the savings in the number of runs from GS are 97.79%, 97.81%, and 80.6% for ANN, SVM, and DBN, respectively. Therefore, this demonstrates the effectiveness of using RSM as the hyperparameter tuning process for the three ML algorithms tested in this study.

We have demonstrated the robustness of the proposed method's effectiveness by testing it with three machine learning models having different characteristics: ANN with numerical hyperparameters, SVM with a mix between numerical hyperparameter and categorical (kernel function), and DBN with many numerical hyperparameters. The fundamental result is that the proposed method can be applied to ML algorithms with various characteristics for predicting a numerical response. This fundamental result also indicates the limitation of RSM that it cannot be applied to other types of responses such as binary and categorical responses found in classification problems. Directions for future research are to test the RSM tuning process in other ML algorithms and/or in other industrial applications to see whether our findings can be useful for other ML algorithms as well as other industries. [48].

Data Availability

Restrictions apply to the availability of these data. Data used in the computational study were obtained from an industrial user and are subject to a nondisclosure agreement. The data and their description may be made available from the corresponding author with the permission of the data provider.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this study.

Acknowledgments

This research was supported by the Thailand Research Fund - Master Research Grants (no. MRG-WI515S025), Kasetsart University Research and Development Institute (no. V-T(D) 118.52), the SIIT Young Researcher Grant (no. SIIT 2019-

YRG-WP01), and the Center of Excellence in Logistics and Supply Chain System Engineering and Technology (COE LogEn), Sirindhorn International Institute of Technology (SIIT), Thammasat University.

Supplementary Materials

Table 9. Best hyperparameter setting and the validation performance of ANN using RSM. (*Supplementary Materials*)

References

[1] K. Y. Ngiam and I. W. Khor, "Big data and machine learning algorithms for health-care delivery," *The Lancet Oncology*, vol. 20, no. 5, pp. e262–e273, 2019.

[2] A. L'heureux, K. Grolinger, H. F. Elyamany, and M. A. Capretz, "Machine learning with big data: challenges and approaches," *IEEE Access*, vol. 5, pp. 7776–7797, 2017.

[3] G. D. Magoulas and A. Prentza, *Machine Learning in Medical Applications. Advanced Course on Artificial Intelligence*, Springer, New York, NY, USA, 1999.

[4] K. Kourou, T. P. Exarchos, K. P. Exarchos, M. V. Karamouzis, and D. I. Fotiadis, "Machine learning applications in cancer prognosis and prediction," *Computational and Structural Biotechnology Journal*, vol. 13, pp. 8–17, 2015.

[5] P. Probst, M. N. Wright, and A. L. Boulesteix, "Hyperparameters and tuning strategies for random forest," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 9, p. e1301, 2019.

[6] F. Hutter, L. Kotthoff, and J. Vanschoren, *Automated Machine Learning: Methods, Systems, Challenges*, Springer Nature, London, UK, 2019.

[7] G. A. Lujan-Moreno, P. R. Howard, O. G. Rojas, and D. C. Montgomery, "Design of experiments and response surface methodology to tune machine learning hyperparameters, with a random forest case-study," *Expert Systems with Applications*, vol. 109, pp. 195–205, 2018.

[8] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, pp. 281–305, 2012.

[9] P. Probst, A. L. Boulesteix, and B. T. Bischl, "Importance of hyperparameters of machine learning algorithms," *Journal of Machine Learning Research*, vol. 20, pp. 1–32, 2019.

[10] D. Yogatama and G. Mann, "Efficient transfer learning method for automatic hyperparameter tuning," in *Proceedings of the International Conference*, Reykjavik, Iceland, April 2014.

[11] I. Syarif, A. Prugel-Bennett, and G. Wills, "SVM parameter optimization using grid search and genetic algorithm to improve classification performance," *Telkomnika*, vol. 14, no. 4, pp. 1502–1509, 2016.

[12] A. k. Al-Fugara, M. Ahmadlou, A. R. Al-Shabeeb, S. AlAyyash, H. Al-Amoush, and R. Al-Adamat, "Spatial mapping of groundwater springs potentiality using grid search-based and genetic algorithm-based support vector regression," *Geocarto International*, vol. 8, pp. 1–20, 2020.

[13] L. Zhou, K. K. Lai, and L. Yu, "Credit scoring using support vector machines with direct search for parameters selection," *Soft Computing*, vol. 13, no. 2, pp. 149–155, 2009.

[14] A. S. Wicaksono and A. A. Supianto, "Hyper parameter optimization using genetic algorithm on machine learning methods for online news popularity prediction," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 12, pp. 263–267, 2018.

[15] L. Zhou, K. K. Lai, and J. Yen, "Bankruptcy prediction using SVM models with a new approach to combine features selection and parameter optimisation," *International Journal of Systems Science*, vol. 45, no. 3, pp. 241–253, 2014.

[16] J. H. Han, D. J. Choi, S. U. Park, and S. K. Hong, "Hyperparameter optimization using a genetic algorithm considering verification time in a convolutional neural network," *Journal of Electrical Engineering & Technology*, vol. 15, no. 2, pp. 721–726, 2020.

[17] P. R. Lorenzo, J. Nalepa, M. Kawulok, L. S. Ramos, and J. R. Pastor, "Particle swarm optimization for hyper-parameter selection in deep neural networks," in *Proceedings of the genetic and evolutionary computation conference*, pp. 481–488, Berlin, Germany, July, 2017.

[18] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Proceedings of the Twentyfourth International Conference on Neural Information Processing Systems*, vol. 24, pp. 2546–2554, Granada, Spain, December 2011.

[19] M. A. Amirabadi, M. H. Kahaei, S. A. Nezamalhosseini, and V. T. Vakili, "Deep Learning for channel estimation in FSO communication system," *Optics Communications*, vol. 459, Article ID 124989, 2020.

[20] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," *Advances in Neural Information Processing Systems*, vol. 25, pp. 2951–2959, 2012.

[21] W. M. Czarnecki, S. Podlewska, and A. J. Bojarski, "Robust optimization of SVM hyperparameters in the classification of bioactive compounds," *Journal of Cheminformatics*, vol. 7, no. 1, pp. 1–15, 2015.

[22] G. N. Kouziokas, "SVM kernel based on particle swarm optimized vector and Bayesian optimized SVM in atmospheric particulate matter forecasting," *Applied Soft Computing*, vol. 93, Article ID 106410, 2020.

[23] D. C. Montgomery, *Design and Analysis of Experiments*, John Wiley & Sons, Hoboken, NJ, USA, 2017.

[24] F. Budiman, "SVM-RBF parameters testing optimization using cross validation and grid search to improve multiclass classification," *Scientific Visualization*, vol. 11, no. 1, pp. 80–90, 2019.

[25] H. Kaneko and K. Funatsu, "Fast optimization of hyperparameters for support vector regression models with highly predictive ability," *Chemometrics and Intelligent Laboratory Systems*, vol. 142, pp. 64–69, 2015.

[26] N. Asadi and H. Zilouei, "Optimization of organosolv pretreatment of rice straw for enhanced biohydrogen production using Enterobacter aerogenes," *Bioresource Technology*, vol. 227, pp. 335–344, 2017.

[27] L. Yu, X. Yao, S. Wang, and K. K. Lai, "Credit risk evaluation using a weighted least squares SVM classifier with design of experiment for parameter selection," *Expert Systems with Applications*, vol. 38, no. 12, pp. 15392–15399, 2011.

[28] I. H. Boyacı, "A new approach for determination of enzyme kinetic constants using response surface methodology," *Biochemical Engineering Journal*, vol. 25, pp. 55–62, 2005.

[29] A. Y. Aydar, "Utilization of response surface methodology in optimization of extraction of plant materials," *Statistical Approaches with Emphasis on Design of Experiments Applied to Chemical Processes*, Intech Open, Londan, U.K, pp. 157–169, 2018.

[30] A. Asghar, A. A. Abdul Raman, and W. M. Daud, "A comparison of central composite design and Taguchi method for

optimizing Fenton process," *TheScientificWorldJOURNAL*, vol. 2014, Article ID 869120, 14 pages, 2014.

[31] J. U. Ani, U. C. Okoro, L. E. Aneke et al., "Application of response surface methodology for optimization of dissolved solids adsorption by activated coal," *Applied Water Science*, vol. 9, no. 3, p. 60, 2019.

[32] C.-F. Mandenius and A. Brundin, "Bioprocess optimization using design-of-experiments methodology," *Biotechnology Progress*, vol. 24, no. 6, pp. 1191–1203, 2008.

[33] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[34] D. Svozil, V. Kvasnicka, and J. Pospichal, "Introduction to multi-layer feed-forward neural networks," *Chemometrics and Intelligent Laboratory Systems*, vol. 39, no. 1, pp. 43–62, 1997.

[35] E. M. Johansson, F. U. Dowla, and D. M. Goodman, "Backpropagation learning for multilayer feed-forward neural networks using the conjugate gradient method," *International Journal of Neural Systems*, vol. 2, no. 4, pp. 291–301, 1991.

[36] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[37] S.-q. Zhang and K.-P. Lin, "Short-term traffic flow forecasting based on data-driven model," *Mathematics*, vol. 8, no. 2, p. 152, 2020.

[38] G. Battineni, N. Chintalapudi, and F. Amenta, "Machine learning in medicine: performance calculation of dementia prediction by support vector machines (SVM)," *Informatics in Medicine Unlocked*, vol. 16, Article ID 100200, 2019.

[39] N. L. da Costa, L. A. G. Llobodanin, M. D. de Lima, I. A. Castro, and R. Barbosa, "Geographical recognition of Syrah wines by combining feature selection with Extreme Learning Machine," *Measurement*, vol. 120, pp. 92–99, 2018.

[40] H. Bhavsar and M. H. Panchal, "A review on support vector machine for data classification," *International Journal of Advanced Research in Computer Engineering & Technology*, vol. 1, no. 10, 2012.

[41] C.-Y. Yeh, C.-W. Huang, and S.-J. Lee, "A multiple-kernel support vector regression approach for stock market price forecasting," *Expert Systems with Applications*, vol. 38, no. 3, pp. 2177–2186, 2011.

[42] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[43] H. Li, Q. Xu, Y. He, X. Fan, and S. Li, "Modeling and predicting reservoir landslide displacement with deep belief network and EWMA control charts: a case study in Three Gorges Reservoir," *Landslides*, vol. 17, no. 3, pp. 693–707, 2020.

[44] X. Wang, T. Chen, and H. Xu, "Thickness distribution prediction for tectonically deformed coal with a deep belief network: a case study," *Energies*, vol. 13, no. 5, p. 1169, 2020.

[45] L. De Marchi and L. Mitchell, *Hands-On Neural Networks: Learn How to Build and Train Your First Neural Network Model Using Python*, Packt Publishing Ltd, Birmingham, UK, 2019.

[46] W. A. Jensen, "Confirmation runs in design of experiments," *Journal of Quality Technology*, vol. 48, no. 2, pp. 162–177, 2016.

[47] S. K. Behera, H. Meena, S. Chakraborty, and B. C. Meikap, "Application of response surface methodology (RSM) for optimization of leaching parameters for ash reduction from low-grade coal," *International Journal of Mining Science and Technology*, vol. 28, no. 4, pp. 621–629, 2018.

[48] M. S. Saad, A. M. Nor, M. E. Baharudin, M. Z. Zakaria, and A. F. Aiman, "Optimization of surface roughness in FDM 3D printer using response surface methodology, particle swarm optimization, and symbiotic organism search algorithms," *International Journal of Advanced Manufacturing Technology*, vol. 105, no. 12, pp. 5121–5137, 2019.