IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 13, 2020, accepted October 21, 2020, date of publication October 29, 2020, date of current version November 10, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3034566

Two Derivative Algorithms of Gradient Boosting Decision Tree for Silicon Content in Blast Furnace System Prediction

SHIHUA LUO AND TIANXIN CHEN
School of Statistics, Jiangxi University of Finance and Economics, Nanchang 330013, China
Corresponding author: Tianxin Chen (526971826@qq.com)

This work was supported in part by the Natural Science Foundation of China under Grant 61973145, and in part by the Foundation of the Education Department of Jiangxi Province under Grant GJJ180247.

ABSTRACT The background of the present study complies with silicon content prediction in hot metal in the blast furnace system. The blast furnace system is a highly complex industrial reactor in the conventional process. The system is subject to several problems (e.g., system automation, the thermal state of the blast furnace, and the life prediction of blast furnace) that should be addressed by professionals. To determine the prediction state of the heat in the blast furnace, the silicon content in the blast furnace molten iron commonly acts as a key indicator. Based on the assumption that the blast furnace system exhibits a stable state, the accuracy of hot metal silicon is analyzed by using a range of machine learning algorithms. In the present study, two derivative algorithms of gradient boosting decision tree are adopted to develop a strong boosting predictor based on the extreme gradient boosting (XGBoost) algorithm and the light gradient boosting machine (LightGBM) algorithm for prediction. Compared with the conventional algorithms (e.g., lasso, random forest, support vector machine and gradient boosting decision tree), the prediction by using the two boosting algorithms is capable of more effectively guiding and determining the state of the blast furnace. As revealed from experimentally simulated results, the mentioned two boosting algorithms exhibit better comprehensive prediction performance than the conventional algorithms on the datasets of two practical blast furnace systems, demonstrating that the R-square of the two blast furnaces in the training set is over 0.7. The mentioned two algorithms are of certain guiding significance for exploring blast furnace problems.

INDEX TERMS XGBoost, LightGBM, gradient boosting decision tree, silicon content, blast furnace system.

I. INTRODUCTION

The Blast furnace (BF) ironmaking process, an essential process in the iron and steel industry, refers to a complex physical and chemical process extensively applied for pig iron production. BF temperature refers to a vital index to reflect the operation state. The responsibility of a BF operator is to ensure the stable operation of the BF for high-quality and low-cost molten iron production. For this purpose, the BF temperature should be maintained in a reasonable range. Otherwise, the unexpected fluctuation of temperature will result in a series of problems (e.g., unnecessary heat loss, in-furnace cooling and abnormal nodulation). However, in practical productions, the thermal state of BF cannot be directly measured as impacted by the harsh conditions (e.g., high temperature and high pressure) [1]. Thus, BF temperature prediction and control have been overall a hot and difficult issue in this field [2], [3]. The content of silicon in molten iron refers to a vital and easy-to-detect index extensively adopted to measure the internal thermal state of a BF in smelting process control. Accordingly, the silicon content is required to be accurately predicted to help BF operators take corresponding actions.

The critical significance and difficulty of modeling BF smelting process are reflected by the aim to accurately predict the BF temperature and the relationship between temperature, coal injection amount, air volume, air temperature, as well as coke load. Since the beginning of the 21st century, studies on BF temperature prediction and control by adopting intelligent algorithms have also been leaping forward.

The associate editor coordinating the review of this manuscript and approving it for publication was Wai-Keung Fung.

Liu built a Bayesian network model for silicon content prediction in molten iron [4]. In 2005, Liu and Gong initially adopted fuzzy clustering algorithm to divide the samples. Then, they applied Bayesian network model to predict the silicon content in molten iron [5]. In 2005, Gao published one study on silicon content in BF molten iron in accordance with chaos theory on Acta Physica Sinica [6]. Subsequently, in his doctoral dissertation, he built a combination model by complying with chaos theory, i.e., the modified chaotic weighted local prediction model for silicon content prediction [7]. Gao *et al.* developed a chaotic prediction method based on Volterra expansion. By conducting the grid search for reconstruction parameters, the linear Volterra filter based on chaos overall outperformed the chaotic predictor based on Taylor expansion to predict the silicon content of BF molten iron [8]. Chen *et al.* developed a hybrid method, in which genetic algorithm evolutionary neural network was adopted for silicon content prediction in BF molten iron [9]. Wang mentioned the wavelet analysis method in her master thesis in 2005 [10], [11]. In 2014, she further proposed the model for silicon content prediction in molten iron by adopting the random forest algorithm [12], and the predicting effect was significantly improved. As an abstract mathematical model built by complying with the Artificial Neural Networks, such algorithm is capable of effectively establishing a nonlinear complex system and dealing with nonlinear data to a certain extent. The mentioned algorithm has now been extensively applied in researching the silicon content in BF molten iron [13]–[16]. In terms of the prediction model based on time series analysis, Henrik and Matias put forward a nonlinear time series model in 2002 [17]. In 2005, Uusi adopted a multivariate time series model for the analysis of the silicon content in molten iron [18]. Bhattacharya employed a partial least squares algorithm to conduct the silicon content prediction, and the observed input and output were split into the principal components [19]. In 2009, Liu and Zeng *et al.* carried out dimensionality reduction on variables by conducting principal component analysis, selected the parameters with a cumulative variance contribution rate over 85%, and then built a partial least squares model [20]. Zeng *et al.* exploited data-driven algorithms to establish multivariate time series models in 2011 and 2013 [21], [1]. Zhou *et al.* developed a nonlinear subspace identification method, i.e., Hammerstein model, by complying with least squares support vector machine (SVM) for silicon content prediction in BF molten iron [22]. Zeng *et al.* built a linear input and output model for the prediction of the silicon content in BF molten iron with the use of subspace identification method. They further adopted this model to achieve silicon content prediction and control in BF molten iron [21]. Jian applied the radial basis function (RBF) method and the SVM algorithm for silicon content classification and prediction in BF molten iron [23]–[25]. Moreover, he drew upon least squares SVM in 2008 to study the silicon content of BF molten iron in-depth [26]. Not before long, in 2011, Jian used the smooth support vector machine regression (SVR) method for silicon content prediction in molten iron [27]. Such method is capable of effectively determining the trend of the silicon content and further analyzing the variations in furnace temperature. Besides the direct regression prediction of silicon content in BF molten iron, SVMs can be adopted to predict the trend of silicon content in BF molten iron as well [28]–[30], [35]–[38]. To be specific, Gao *et al.* proposed an algorithm for the determination of the control limits of silicon content in BF molten iron with the use of fuzzy SVMs. They further converted the mentioned prediction problem into a multi-classification problem [28]. In 2020, with the use of Mimo-TS fuzzy model, Li *et al.* put forward the assessment of BF system [33]. Moreover, Luo *et al.* proposed the algorithm integrating AdaBoost and the support vector machine to achieve the aim of predicting the molten iron silicon content of BF [34]. The mentioned models have achieved satisfactory results in a range of aspects, whereas the presented algorithms and models still have their defects. For instance, neural network consumes a considerable amount of time in modeling, and it exhibits more sensitivity to the dimensionality of nonlinear data; chaotic model will cause poor prediction as impacted by time lag; time series model only exploits historical data of BF smelting and requires the stable conditions of a BF system; SVM model has an overly sophisticated process of parameter selection.

To address the problems in the above models, the present study proposes two derivative algorithms based on gradient boosting decision tree (GBDT), i.e., Extreme Gradient Boosting (XGBoost) algorithm and Light Gradient Boosting Machine (LightGBM) algorithm, for silicon content prediction in BF molten iron [31], [32], [39]–[47]. The core idea of GBDT is to iterate multiple decision trees by altering the sample label, and finally form a strong learner. Thus, the predicted value is close to the actual data. The XGBoost algorithm follows the GBDT algorithm to add regularization terms to prevent the model from overfitting. Moreover, the LightGBM algorithm is based on the GBDT algorithm to simplify the model and expedite the calculation by gradient-based one-side sampling and exclusive feature bundling. The main contribution of this study is to propose two potential alternatives to predict the internal thermal state of BF, i.e., XGBoost predictor and LightGBM predictor. Furthermore, the predicted results can fundamentally guide the operators to judge the thermal state of BF.

The innovation of the present study mainly embodies two points. First, XGBoost algorithm and LightGBM algorithm are new algorithms proposed in recent five years, demonstrating that the former was proposed in 2016, the latter was proposed in 2017. The mentioned two methods are relatively new. Secondly, XGBoost algorithm and LightGBM algorithm are initially adopted to analyze the silicon content of hot metal from industrial data of BF. Based on these two points of innovation, it is necessary and feasible to select these two algorithms to study the state of BF.

The rest of the present study is organized as follows. In Section 2, the theoretical models of GBDT algorithm,

XGBoost algorithm and LightGBM algorithm are presented. In Section 3, an empirical assessment is conducted on the real datasets of two BFs to reveal the advantages of the two derivative algorithms. Lastly, the conclusions are drawn in Section 4.

II. REVIEW OF RELATED METHODS

A. EXTREME GRADIENT BOOSTING (XGBoost)

1) INTRODUCTION OF XGBoost

XGBoost algorithm, a derivative algorithm of GBDT algorithm, was proposed by Chen Tianqi in his doctoral dissertation [27]. Since GBDT algorithm is easy to cause overfitting, the idea of XGBoost algorithm is to introduce regularization terms into the original GBDT algorithm. In comparison with other algorithms, XGBoost has been comprehensively optimized, as reflected by the significantly improved training speed and accuracy. The algorithm steps are clarified below:

Suppose x_i is the input of the i_{th} sample and y_i is the output value of the i_{th} sample, $\hat{y}_i^{(t)}$ is the predicted value of by t-time. If the initial prediction value of the model is assumed as 0, the t models obtained by t-times integrated learning are expressed by:

$$\begin{aligned}\hat{y}_i^{(0)} &= 0\\ \hat{y}_i^{(1)} &= \hat{y}_i^{(0)} + f_1(x_i)\\ \hat{y}_i^{(2)} &= \hat{y}_i^{(1)} + f_2(x_i)\\ &\cdots\\ \hat{y}_i^{(t)} &= \hat{y}_i^{(t-1)} + f_t(x_i)\end{aligned} \tag{1}$$

where f denotes the weak learner. The loss function of XGBoost model consists of two parts, i.e., the empirical loss of the model and the regularization term, as expressed below:

$$loss = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \Omega(f_i) \tag{2}$$

where $\sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)})$ is the empirical loss of the model. The regularization term is $\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} {w_j}^2$, where T denotes the number of leaf nodes in the model, w_j represents the value on each leaf node. γ, λ are the corresponding coefficients. By rewriting the objective function of Eq. (2), it yields:

$$loss = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \sum_{i=1}^{t-1} \Omega(f_i) + \Omega(f_t) \tag{3}$$

where $\sum_{i=1}^{t-1} \Omega(f_i)$ is the regular term of the preceding t-1 term, $\Omega(f_t)$ stands for the regular term of t term. By taking $\hat{y}_i^{(t-1)} + f_t(x_i)$ in the error function as a whole, the second-order Taylor expansion is conducted at $\hat{y}_i^{(t-1)}$ to solve the whole value. Then, the objective function becomes:

$$loss = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)) + \sum_{i=1}^{t-1} \Omega(f_i) + \Omega(f_t) \tag{4}$$

where $g_i = \partial_{\hat{y}_i}^{(t-1)} l(y_i, \hat{y}_i^{(t-1)})$, $h_i = \partial_{\hat{y}_i}{}^{2(t-1)} l(y_i, \hat{y}_i^{(t-1)})$. g_i is the first derivative of the loss function and h_i is the second derivative of the loss function. The constant terms in the objective function are deleted to yield:

$$loss \approx \sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t) \tag{5}$$

Furthermore, each data point x_i will eventually fall on a leaf node. Besides, for the data points falling on the same leaf node, the output will be identical. If j leaf nodes are yielded in total, and the corresponding output of each leaf node is w_j, and the objective function is rewritten as:

$$loss \approx \sum_{i=1}^{n} [g_i w_j + \frac{1}{2} h_i w_j^2] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} {w_j}^2 \tag{6}$$

The sample set on each leaf node j is defined as I_j:

$$I_j = \{i \,|q(x_i) == j\} \tag{7}$$

By transforming the sample accumulation operation to the leaf node operation, the final objective function is defined as:

$$\begin{aligned} loss &\approx \sum_{j=1}^{T} [(\sum_{i \in I_j} g_i) w_j + \frac{1}{2} (\sum_{i \in I_j} h_i + \lambda) w_j^2] + \gamma T\\ &= \sum_{j=1}^{T} [G_j w_j + \frac{1}{2} (H_j + \lambda) w_j^2] + \gamma T \end{aligned} \tag{8}$$

where $G_j = \sum_{i \in I_j} g_i$, $H_j = \sum_{i \in I_j} h_i$. G_j represents the sum of the first derivatives on all leaf nodes, while H_j stands for the second derivatives on all leaf nodes. If the structure of the tree is determined, to minimize the objective function, the derivative can be 0, the optimal w can be acquired, and the final loss can be obtained by substituting w into the objective function:

$$\begin{aligned} w_j^* &= -\frac{G_j}{H_j + \lambda}\\ loss^* &= -\frac{1}{2} \sum_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} + \gamma T \end{aligned} \tag{9}$$

With the objective function, to split a leaf node, a simple decision can be employed as the judgment criterion of whether the splitting can be conducted, i.e., whether the sum of the objective functions on both sides after the splitting can be greater than that without the splitting. The information acquired before and after the splitting is expressed as:

$$Gain = \frac{1}{2}[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{(H_L + H_R) + \lambda}] - \gamma \tag{10}$$

where G_L and H_L denote the G value and H value of the left subtree, respectively, and G_R and H_R represent the G value and H value of the right subtree, respectively. The larger the gain value, the greater the loss value will be. Thus, when a leaf node is split, the gain values corresponding to all candidate features are calculated, and the feature representing the maximum gain value is taken for splitting.

Algorithm 1: Exact Greedy Algorithm for Split Finding

Input: I, instance set of current node
Input: d, feature dimension
$gain \leftarrow 0$
$G \leftarrow \sum_{i \in I} g_i$, $H \leftarrow \sum_{i \in I} h_i$
for $k = 1$ *to* m **do**
  $G_L \leftarrow 0$, $H_L \leftarrow 0$
  for j *in sorted*(I, *by* $\mathbf{x}_{jk}$) **do**
    $G_L \leftarrow G_L + g_j$, $H_L \leftarrow H_L + h_j$
    $G_R \leftarrow G - G_L$, $H_R \leftarrow H - H_L$
    $score \leftarrow \max(score, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda})$
  end
end
Output: Split with max score

FIGURE 1. Accurate Algorithm.

Algorithm 2: Approximate Algorithm for Split Finding

for $k = 1$ *to* m **do**
  Propose $S_k = \{s_{k1}, s_{k2}, \cdots s_{kl}\}$ by percentiles on feature k.
  Proposal can be done per tree (global), or per split(local).
end
for $k = 1$ *to* m **do**
  $G_{kv} \leftarrow= \sum_{j \in \{j | s_{k,v} \geq \mathbf{x}_{jk} > s_{k,v-1}\}} g_j$
  $H_{kv} \leftarrow= \sum_{j \in \{j | s_{k,v} \geq \mathbf{x}_{jk} > s_{k,v-1}\}} h_j$
end
Follow same step as in previous section to find max score only among proposed splits.

FIGURE 2. Approximate Algorithm.

2) TREE NODE SPLITTING METHOD

Two types of splitting methods can be adopted for splitting the tree nodes, i.e., accurate algorithm and approximate algorithm. The algorithm steps are illustrated in Fig. 1 and 2, respectively.

Method 1: Accurate Algorithm

All possible splitting points of all features are traversed, the gain value is calculated, and the feature corresponding to the maximal gain value is selected for splitting.

Method 2: Approximate Algorithm

For each feature, only fractiles are considered ion to simplify the calculation.

3) XGBoost ADVANTAGES

The advantages of the XGBoost algorithm are as summarized below:

[1] Column subsampling. Using the method of the random forest for reference, column sampling is supported, which can reduce overfitting while simplifying the calculation;

[2] XGBoost supports the automatic processing of missing values. XGBoost can learn the splitting direction automatically for the samples with missing feature values;

[3] XGBoost supports parallelism. The parallelism of XGBoost refers to feature granularity. When calculating the gain value of features, the execution is performed in parallel, whereas it remains serial building in the process of tree building;

[4] The regularization term is substituted to XGBoost algorithm to reduce the complexity of the model, and the final model is less likely to cause overfitting;

TABLE 1. Memory consumption of the three algorithms for a range of datasets (unit: GB).

Data	XGBoost	XGBoost-hist	LightGBM
Higgs	4.853GB	3.784GB	0.868GB
Yahoo-LTR	1.907GB	1.468GB	0.831GB
MS-LTR	5.469GB	3.654GB	0.886GB
Expo	1.553GB	1.393GB	0.543GB
Allstate	6.237GB	4.990GB	1.027GB

TABLE 2. Time consumption of the three algorithms for a range of datasets (unit: s).

Data	XGBoost	XGBoost-hist	LightGBM
Higgs	3794.34s	551.90s	238.51s
Yahoo-LTR	674.32s	265.30s	150.19s
MS-LTR	1251.27s	385.20s	215.32s
Expo	1607.35s	588.25s	138.50s
Allstate	2867.22s	1355.71s	348.08s

[5] XGBoost-based learners support cart, linear regression and logistic regression;

[6] XGBoost supports user-defined loss function (the second-order derivative of the loss function is required).

4) XGBoost DEFECTS

The XGBoost algorithm also has defects, which largely consist of:

[1] XGBoost follows a pre-sorting strategy. Before iterating, it pre-sorts the features of nodes and traverses to select the optimal splitting point. In the presence of a large scale of data, such greedy splitting method is time-consuming.

[2] XGBoost complies with the level-wise strategy to generate a decision tree, and splits the leaves of the same layer simultaneously, as an attempt to perform multi-thread optimization. Accordingly, overfitting can be largely avoided, whereas many leaf nodes achieve a low splitting gain, and no further splitting is required, which results in unnecessary consumption.

To remedy the defects brought by XGBoost algorithm, another derivative algorithm of GBDT, i.e., LightGBM algorithm, is further proposed.

B. LIGHT GRADIENT BOOSTING MACHINE (LightGBM)

1) INTRODUCTION OF LightGBM

LightGBM refers to another derivative algorithm of GBDT, and it was proposed by Microsoft in 2017 [28]. Compared with XGBoost algorithm, LightGBM algorithm has the characteristics of fast training speed and low memory consumption. Table 1 and 2 list the comparison of memory and training time between XGBoost, XGBoost-hist (the XGBoost with gradient histogram) and LightGBM for a range of datasets:

Conventional boosting algorithms (e.g., GBDT and XGBoost) already have quite a good efficiency, whereas they are suggested to be not capable of satisfying the requirements for efficiency and scalability in the environment of large-scale and high-dimensional data. This is primarily explained as the conventional boosting algorithms are required to scan

Algorithm 2: Gradient-based One-Side Sampling

Input: I: training data, d: iterations
Input: a: sampling ratio of large gradient data
Input: b: sampling ratio of small gradient data
Input: $loss$: loss function, L: weak learner
models $\leftarrow$ {}, fact $\leftarrow \frac{1-a}{b}$
topN $\leftarrow$ a $\times$ len(I) , randN $\leftarrow$ b $\times$ len(I)
for $i = 1$ ***to*** d **do**
  preds $\leftarrow$ models.predict(I)
  g $\leftarrow$ $loss$(I, preds), w $\leftarrow$ {1,1,...}
  sorted $\leftarrow$ GetSortedIndices(abs(g))
  topSet $\leftarrow$ sorted[1:topN]
  randSet $\leftarrow$ RandomPick(sorted[topN:len(I)], randN)
  usedSet $\leftarrow$ topSet + randSet
  w[randSet] $\times =$ fact ▷ Assign weight $fact$ to the small gradient data.
  newModel $\leftarrow$ L(I[usedSet], $-$ g[usedSet], w[usedSet])
  models.append(newModel)

FIGURE 3. Gradient-based One-Side Sampling algorithm.

all the sample points for the respective feature to select the optimal splitting point, which consumes considerable time. To reduce the time consumed in processing large samples of high-dimensional data, LightGBM uses the following two strategies. One is gradient-based one-side sampling (GOSS), carrying out the calculation of the gradient by sampling the samples, instead of employing all the sample points. The other refers to exclusive feature bundling (EFB) which determines the optimal splitting point by bundling certain features jointly to the dimension reduction of the features rather than scanning all features, which reduce the cost for finding the optimal splitting point. This significantly reduces the time complexity of processing samples, while keeping or even increasing the accuracy as verified by considerable experiments that use LightGBM on certain datasets. In brief, the GBDT with the use of the GOSS and EFB algorithms is termed as LightGBM. The following sections will introduce the mentioned two algorithms.

2) INTRODUCTION OF GRADIENT-BASED ONE-SIDE SAMPLING ALGORITHM (GOSS)

The gradient size of the GBDT algorithm is capable of indicating the weight of the sample. The smaller the gradient, the better the model fitting effect will achieve. The Gradient-based One-Side Sampling (GOSS) algorithm employs this information for sampling, thereby reducing considerable samples exhibiting small gradients. Thus, only the samples with high gradients need to be considered in the further calculation, which significantly reduces the amount of calculation. The GOSS algorithm retains the samples exhibiting large gradients and randomly conducts sampling with small gradients. To keep the data distribution of the samples not changed, a constant is introduced for the samples exhibiting small gradients to balance when calculating the gain. The specific algorithm steps are illustrated in Fig. 3.

The specific steps of the GOSS algorithm are elucidated as follows:

Input: training data, iteration steps *d,* the sampling rate of large-gradient data is *a*, sampling rate of small-gradient data is expressed as *b*, loss function and type of weak learner (the decision tree on the whole);

Output: trained strong learner;

[1] Sort the sample points in descending order in accordance with their absolute values;

[2] Select the first $a\times100\%$ samples of the sorted results to yield a subset of large-gradient sample points;

[3] For the remaining sample set *(1-a)*$\times$*100%* samples, randomly select $b\times$*(1-a)*$\times$*100%* sample points to yield a set of small-gradient sample points;

[4] Conduct the integration of the large-gradient sample with the sampled small-gradient samples;

[5] Multiply the small-gradient samples through a weight coefficient;

[6] Use the mentioned sampled samples to learn a new weak learner;

[7] Repeat Steps [1] to [6] continuously until the specified number of iterations or convergence is reached.

The presented algorithm can significantly reduce the model learning rate while keeping the accuracy of the learner without altering the data distribution.

The above description indicates that when $a = 0$, the GOSS algorithm degenerates into a random sampling algorithm; when $a = 1$, the GOSS algorithm becomes an algorithm that takes all the samples. In a large number of cases, the accuracy of the model trained by the GOSS algorithm is higher than that by the random sampling algorithm. Besides, sampling will also increase the diversity of weak learners, thereby potentially improving the generalization ability of the trained model.

3) INTRODUCTION OF EXCLUSIVE FEATURE BUNDLING ALGORITHM (EFB)

When LightGBM algorithm is being implemented, both data sampling and feature sampling are carried out, thereby further decreasing the training speed of the model. However, this type of feature sampling is a range of from the general feature sampling, which bundles exclusive features together to reduce the feature dimension. The main idea is that in practical applications, high-dimensional data is often sparse data (e.g., one-hot encoding), which makes it possible to design an almost lossless method to reduce the number of effective features. It is noteworthy that many features in the sparse feature space are mutually exclusive (for instance, non-zero values rarely occur simultaneously). This allows us to safely bundle mutually exclusive features together to form one feature, thereby reducing feature dimensions.

Since the features are divided into smaller numbers of mutually exclusive bundles, this is an NP-hard problem, that is, it is not likely to find an accurate solution in polynomial time. For the mentioned reason, an approximate solution is adopted here, i.e., there are a few sample points between

Algorithm 3: Greedy Bundling

Input: F: features, K: max conflict count
Construct graph G
searchOrder $\leftarrow G$.sortByDegree()
bundles $\leftarrow \{\}$, bundlesConflict $\leftarrow \{\}$
for i ***in*** *searchOrder* **do**
  needNew $\leftarrow$ True
  for $j = 1$ ***to*** *len(bundles)* **do**
    cnt $\leftarrow$ ConflictCnt(bundles[j],F[i])
    if *cnt + bundlesConflict[i]* $\leq K$ **then**
      bundles[j].add(F[i]), needNew $\leftarrow$ False
      break
  if *needNew* **then**
    Add $F[i]$ as a new bundle to $bundles$
Output: $bundles$

Algorithm 4: Merge Exclusive Features

Input: $numData$: number of data
Input: F: One bundle of exclusive features
binRanges $\leftarrow \{0\}$, totalBin $\leftarrow 0$
for f ***in*** F **do**
  totalBin += f.numBin
  binRanges.append(totalBin)
newBin $\leftarrow$ new Bin(numData)
for $i = 1$ ***to*** $numData$ **do**
  newBin[i] $\leftarrow 0$
  for $j = 1$ ***to*** *len(F)* **do**
    if $F[j].bin[i] \neq 0$ **then**
      newBin[i] $\leftarrow$ F[j].bin[i] + binRanges[j]
Output: $newBin$, $binRanges$

FIGURE 4. Exclusive Feature Bundling algorithm.

features not mutually exclusive (e.g., there are some corresponding sample points that are not non-zero values simultaneously). Allowing a small part of conflict can obtain a smaller number of feature bundles, thereby further enhancing the effectiveness of the calculation. It can be theoretically proved that by allowing a small part of conflict, the accuracy of the model is affected. This refers to the maximum conflict rate of the respective bundling. Accordingly, when it has a limited choice, it is capable of achieving a good balance between accuracy and efficiency. The algorithm steps of the Exclusive Feature Bundling (EFB) algorithm are illustrated in Fig. 4.

There are two issues to be addressed. In Fig.4, Algorithm 3 is to determine which features should be bundled together, while Algorithm 4 is how to construct the bundle. The steps of the EFB algorithm are elucidated as follows:

Input: feature F, maximum conflict number K, graph G;

Output: feature bundles;

[1] A graph with weights on its edges is constructed, whose weights correspond to the total conflicts between features;

[2] Sort features in descending order by their degree in the graph;

[3] Check the respective feature in the ordered list and assign it to the existing bundling with minor conflicts (controlled by γ), or create a novel bundling.

The time complexity of the presented algorithm is $O(\#features^2)$, and it can be processed only once before undergoing model training. Such complexity is acceptable when the feature dimension is not significantly large. However, this method is particularly inefficient when the sample dimension is high. Thus, this study proposes another more efficient algorithm: sorting by counting the non-zero value, which is similar to sorting by degree, since more non-zero values cause higher conflict probability commonly. This only alters the sorting strategy of the mentioned algorithm, so it only modifies the sorting by the degree to the sorting by counting the non-zero value, and the other remains unchanged.

4) INTRODUCTION OF HISTOGRAM

LightGBM adopts the histogram algorithm for the bundling of exclusive features. The basic idea of histogram algorithm is

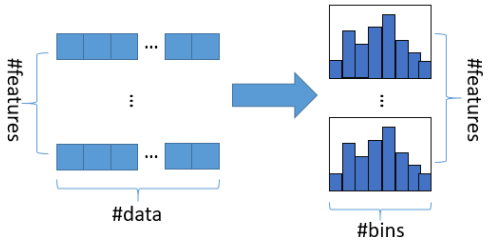

FIGURE 5. Histogram algorithm.

to discretize successive eigenvalues into k integers and build a histogram of width k. When traversing the data, the discretized value acts as an index to accumulate the statistics in the histogram. After traversing the data once, the histogram accumulates the required statistics and subsequently traverses again to seek out the optimal splitting point.

Given that histogram-based algorithm stores discrete bins instead of continuous feature values, a feature bundle can be constructed by letting mutually exclusive features reside in a range of bins. The mentioned process can be achieved by increasing the offset of the original value of the feature.

Indeed, the histogram algorithm remains not perfect. Since the features are discretized, the splitting point is not significantly accurate, probably affecting the result. But the results on a range of datasets show that the discretized splitting points slightly impact the final accuracy, and the accuracy may increase. The reason is that the decision tree is originally a weak model, and it is not overly important whether the splitting point is accurate; moreover, a not-so-accurate splitting point exerts the regularizing effect, capable of effectively preventing overfitting; even if the training error of a single tree is a slightly larger than that of an accurately-split algorithm, it causes no significant difference under the framework of gradient boosting. The idea of histogram algorithm is illustrated in Fig. 5.

The histogram algorithm exhibits the advantages below:

[1] It down-regulates the calculation amount of splitting the gain: the pre-sorted algorithm is applied by default in XGBoost, requiring the calculation amount of $O(\#data)$, while the histogram algorithm only requires the calculation amount of $O(\#bins)$, and $O(\#bins)$ is significantly smaller than $O(\#data)$.

FIGURE 6. Histogram accelerating. (Note: the feature number of the right leaf node is equal to the feature number of the root node minus the feature number of the left leaf node).

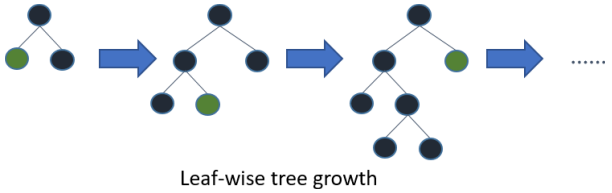

FIGURE 7. Growth strategy of leaf-wise decision tree.

[2] It further expedites the model training by histogram subtraction. According to the binary tree, the histogram of the leaf node can be obtained by exploiting the subtraction of the histogram of the leaf node's parent and neighboring nodes. Thus, it is only required to yield a histogram for a leaf node (its #data smaller than its neighbors) to develop the histogram of neighboring nodes through the subtraction of the histogram, and the cost *O(#bins)* is very small. Histogram accelerating can be demonstrated by Fig. 6.

Fig.6 shows the histogram acceleration, which indicates that the feature of the root node minus the feature of the left leaf node is equal to the feature of the right node. Meanwhile, when constructing the histogram of the leaf node, it can be constructed by subtracting the histogram of parent node from that of adjacent leaf node, to reduce half of the calculation. In the actual operation process, the leaf nodes with small histogram can be calculated first, and the leaf nodes with large histogram can be obtained by using histogram difference.

5) GROWTH STRATEGY OF LEAF-WISE DECISION TREE

Overall, most decision tree learning algorithms are capable of growing the tree by complying with the level-wise strategy, recording the leaves that split the identical layer at a time, and treat the leaves of the identical layer indiscriminately. Numerous leaves achieve a low splitting gain, and there is no need to split them to avoid unnecessary costs. LightGBM adopts a leaf-wise strategy to grow trees. It means that the decision tree does not need to be segmented by each root node. Each time from all the current leaves, the strategy identifies the one exhibiting the maximum splitting gain and performs the splitting process subsequently. Accordingly, compared with Level-wise strategy, Leaf-wise strategy is capable of reducing more errors and achieving better accuracy when the number of splits is the same. However, when the sample size is small, leaf-wise strategy may cause overfitting. Thus, LightGBM can exploit the additional parameter max_depth to limit the tree depth and avoid overfitting. Furthermore, the leaf-wise strategy is illustrated in Fig. 7.

6) LightGBM ADVANTAGES

Compared with XGBoost algorithm, LightGBM algorithm largely exhibits the following two advantages:

(1) Less memory consumption

[1] XGBoost should record the index of the feature value and the statistical value of the corresponding sample after presorting. As opposed to the mentioned, LightGBM adopts the histogram algorithm to convert the feature value to the bin value without the need to record the feature to the sample index, thereby reducing the spatial complexity from *O(#data)* to *O(#bins)* and further significantly reducing the memory consumption;

[2] LightGBM employs the histogram algorithm to alter the feature value as the bins value, thereby decreasing the memory consumption;

[3] LightGBM follows an EFB algorithm during training to down-regulate the number of features and memory consumption.

(2) Faster speed

[1] LightGBM follows the histogram algorithm to transform the way of traversing the samples into traversing the histograms, which significantly reduces the time complexity;

[2] LightGBM employs a GOSS algorithm to remove the samples exhibiting small gradients during the training process, which significantly reduces the calculation amount;

[3] LightGBM adopts a growth strategy based on leaf-wise algorithm to construct a tree, which reduces considerable unnecessary calculations;

[4] LightGBM draws upon optimized feature parallel and data parallel methods to speed up the calculations. When the scale of data is significantly large, it can adopt the strategy of voting parallel as well;

[5] LightGBM has also optimized the cache, which increases the hit rate of Cache hit.

III. EMPIRICAL ANALYSIS

In the present section, experiments are performed to compare the proposed method with six conventional algorithms (i.e., Lasso, Random Forest (RF), Support Vector Machine (SVM), Gradient Boosting Decision Tree (GBDT), XGBoost and LightGBM) on two real-world (Baotou & Nanchang Iron and Steel Co) BF datasets. In this study, all experiments are performed in Jupyter Notebook for Python 3.6 environment by employing a single computer with 3.4 GHz Intel Core i7 processors and 64 GB of RAM. All the machine learning algorithms mentioned in the present study are conducted by virtue of sklearn.

A. ASSESSMENT CRITERIA

Overall, the performance of the algorithm model can be revealed by the differences between the factory actual data and the predicted value, thereby demonstrating that the smaller the gap, the better the predicting effect of the model will achieve. To address the issue of target attribute regression of the samples, the R-square and mean square errors (MSE)

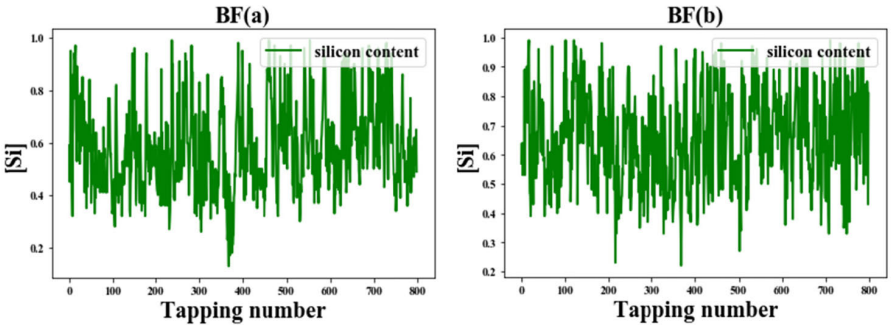

FIGURE 8. Silicon content in hot metal.

have been universally employed to embody the advantages and disadvantages of the model, as expressed below:

$$R^2 = 1 - \frac{RSS}{TSS} \tag{11}$$

where $TSS = \sum_{i=1}^{m} (y_i - \bar{y})^2$ denotes the sum of squares of samples; $RSS = \sum_{i=1}^{m} (\hat{y}_i - y_i)^2$ indicates the sum of squares of residuals.

Overall, R-square indicates the fitting effect exerted by the model. The larger the value of R-square, the higher the fitting effect of the model will achieve. Theoretically, the optimal value of R-square reaches 1. If the predicted value is a random value, R-square may be negative. Besides, if the predicted value is overall the expected value, R-square is 0.

$$MSE = \frac{1}{m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2 \tag{12}$$

where $\hat{y}_i$ stands for the predicted value, y_i represents the actual value. In addition, MSE reflects the fitting degree of the model, indicating the difference between the practical value of the target attribute and the predicted value of the model. The smaller the MSE, i.e., close to 0, the higher the fitting effect of the model will exhibit.

B. SILICON CONTENT PREDICTION IN BLAST FURNACE SYSTEM

In the present section, the performances of the proposed XGBoost and LightGBM algorithms are assessed on the real datasets sampled from two BFs, i.e., BF at Baotou Iron and Steel Group Company of China and BF at Nanchang Iron and Steel Group Company of China, respectively labeled as BF(a) and BF(b). To be specific, BF(a) is a medium-sized BF with an inner volume of 2500 m^3, while BF(b) refers to a small-sized BF with an inner volume of 750 m^3. For the two real BFs, the empirical study takes 7 research variables into account, i.e., air volume, air temperature, air pressure, material speed, top pressure, air permeability as well as oxygen content. The research variables are measured with the use of the data acquisition module, followed by being preprocessed by the data processing module of BF Expert System, and used as the model inputs. The silicon content in molten iron reveals the sample label, as outputted to be a sample.

A total of 800 pieces of data is obtained from the two BFs. The sampling interval is 1.5 h for BF(a) and 1 h for BF(b). Fig. 8 illustrates the data of silicon content obtained from the two BFs. In the empirical analysis, the dimensions of a range of variables are different. Since large-scale variables more significantly impact the model than small-scale variables, the importance of some large-scale variables may be overestimated. To tackle down the mentioned problem, the input data are processed to have the identical dimension according to:

$$x_i^j = \frac{u_i^j - \mu(u_i^j)}{\delta(u_i^j)}, \quad i = 1, 2, \cdots n; \; j = 1, 2, \cdots m \tag{13}$$

where u_i^j denotes the j^{th} input parameter of the i^{th} sampled data; $\mu(u_i^j)$ represents the mean of the j^{th} parameter; $\delta(u_i^j)$ indicates the standard deviation of the j^{th} parameter.

In the experiment, the two sampled datasets are split into two separate parts, i.e., training set and testing set. Lastly, 640 data samples are used for training, and the rest are adopted for testing. Since BF data are time series data, cross validation method is not applied in the empirical analysis.

When predicting the silicon content, the proposed predictor has three key steps. The basic steps are elucidated below:

[1] The silicon content prediction is conducted by using six classical algorithms, i.e., Lasso, Random Forest (RF), Support Vector Machine (SVM), Gradient Boosting Decision Tree (GBDT), XGBoost and LightGBM.

[2] The R-square and MSE of the results achieved by the six algorithms are calculated to compare their performances. The R-square is close to 1, the MSE is close to 0, demonstrating that the prediction of the model is more accurate.

[3] The running time of a range of algorithm models is determined to reflect the efficiency of the model. The shorter the operation time, the higher computational efficiency the model will exhibit. Accordingly, the corresponding model is more applied in artificial intelligence data simulation.

To select model parameters, in the two BF models, Lasso, Random Forest (RF), Support Vector Machine (SVM) and Gradient Boosting Decision Tree (GBDT) act as the default parameters. The two boosting algorithms derived from GBDT (XGBoost and LightGBM) select parameters exhibiting high accuracy for prediction. The calculation parameters of two

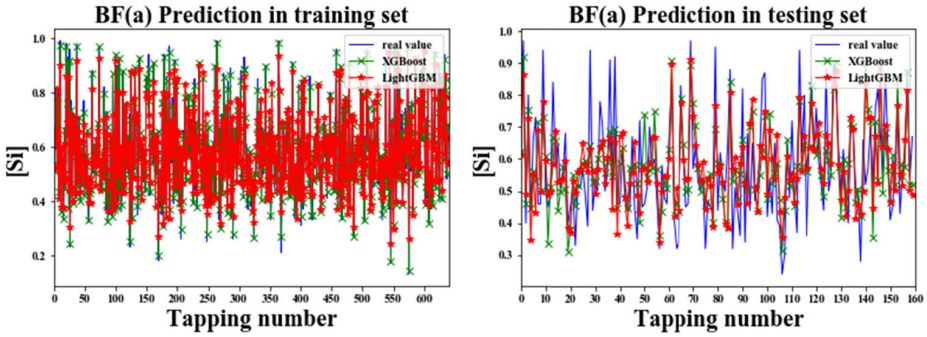

FIGURE 9. Predicted results of BF(a).

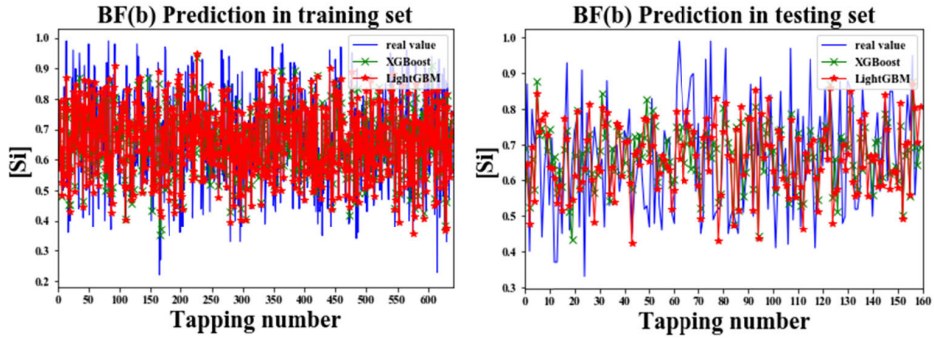

FIGURE 10. Predicted results of BF(b).

blast furnace models for silicon content prediction are briefed below:

BF(a):

[1] lasso: L_1 regularity coefficient was 0.05.

[2] random forest (RF): The maximal number of decision trees allowed reaches 100; the number of feature selection refers to the square root of all features; the maximal depth of the tree reaches 5.

[3] support vector machine (SVM): The default refers to RBF kernel function, and the penalty coefficient C reaches 1; the degree of the polynomial kernel function is 3; the epsilon in the epsilon SVM model reaches 0.1; the independent term in the kernel function refers to 0.

[4] gradient boosting decision tree (GBDT): Alpha coefficient is 0.9, and the loss function is LS; there are maximally 100 decision trees allowed; the maximum depth of the tree is 3; the learning rate is 0.1.

[5] XGBoost: There are maximally 100 decision trees allowed; the maximum depth of the tree is 6, and the learning rate reaches 0.2.

[6] LightGBM: There are maximally 100 decision trees allowed; the maximum depth of the tree is 6, and the learning rate reaches 0.2.

BF(b):

[1] lasso: L_1 regularity coefficient is 1.

[2] random forest (RF): The maximal number of decision trees allowed refers to 100; the number of feature selection indicates the square root of all features; the maximal depth of the tree reaches 5.

TABLE 3. The predicted results of BF(a).

Algorithm	R-Square (training set)	R-Square (testing set)	MSE (training set)	MSE (testing set)	Time (s)
Lasso	0.096	0.079	0.025	0.026	1.48
RF	0.516	0.417	0.014	0.017	0.147
SVM	0.475	0.405	0.015	0.017	0.018
GBDT	0.678	0.402	0.009	0.017	0.045
XGBoost	0.993	0.421	0.0001	0.016	0.74
LightGBM	0.866	0.422	0.004	0.016	0.08

[3] support vector machine (SVM): The default refers to RBF kernel function, and the penalty coefficient C reaches 1; the degree of the polynomial kernel function is 3; the epsilon in the epsilon SVM model reaches 0.1; the independent term in the kernel function is 0.

[4] gradient boosting decision tree (GBDT): Alpha coefficient is 0.9, and the loss function is LS; there are maximally 100 decision trees allowed; the maximum depth of the tree reaches 3; the learning rate is 0.1.

[5] XGBoost: The maximum number of decision trees allowed is 100; the maximum depth of the tree is 4, and the learning rate is 0.1.

[6] LightGBM: There are maximally 100 decision trees allowed; the maximum depth of the tree is 6, and the learning rate is 0.2.

Table 3 and 4 list the simulation results of two BFs by the algorithms. Fig. 9 and 10 illustrate the predicted results achieved by XGBoost and LightGBM on the two BFs.

As can be seen from the above tables and figures, for the two BF systems, the two algorithms derived from GBDT (i.e.,

TABLE 4. Table 4. The predicted results of BF(b).

Algorithm	R-Square (training set)	R-Square (testing set)	MSE (training set)	MSE (testing set)	Time (s)
Lasso	0	−0.0006	0.026	0.024	0.027
RF	0.282	0.124	0.019	0.021	0.156
SVM	0.246	0.199	0.019	0.019	0.019
GBDT	0.573	0.138	0.011	0.02	0.051
XGBoost	0.723	0.137	0.007	0.02	0.130
LightGBM	0.784	0.026	0.006	0.022	0.109

XGBoost & LightGBM), have shown significantly better performance than other algorithms in terms of the fitting effect of the training set. According to the two BF models, the empirical analysis also got different conclusions. For BF(a), it is noteworthy that the R-square of the training set of the mentioned two algorithms exceeds 0.85, and for the XGBoost algorithm, the R-square even reaches up to 0.993. The MSEs of the mentioned two algorithms are both less than 0.005, of which 0.0001 for XGBoost algorithm. As revealed from the comparison between the two algorithms, each exhibits its own merits. For the training set of BF(a), XGBoost algorithm exhibits better predicting effect than LightGBM. Moreover, the mentioned two algorithms exhibit high performance in the testing set of BF(a), indicating the R-square in both algorithms exceeds 0.4 on the testing set. Notably, the proposed algorithms exhibit better performance in terms of the medium-sized BF than on the small-size BF, revealing that the size of the BF will impact the regression performance. Thus, it is difficult to assess the silicon content in the molten iron of small-sized BF. For BF(b), the prediction accuracy of training set of BF(b) is more significant than that of XGBoost. The prediction performance of the two algorithms is not ideal on the testing set, i.e., even worse than the SVM algorithm. Furthermore, the R-squares of the results by all algorithms in the BF(b) are low on the whole, indicating that the performance on the BF(b) dataset is relatively poor. As indicated by the time consumed, SVM algorithm outperforms the other algorithms. Besides, for the mentioned two BF models, it can be discovered that the time consumed by LightGBM is lower than that by XGBoost, which further proves the rationality of the proposed algorithm. Besides, the two derivative algorithms proposed in the present study consume more time than some other algorithms. Finally, it should be noted that the tapping interval of BF is usually 1-1.5h. Accordingly, the proposed two derivative algorithms remain sufficiently effective for the existing task of silicon content prediction in BF system.

However, silicon content prediction in molten iron by the mentioned algorithm model pertains to the point estimation in statistics. Furthermore, interval estimation can also be applied for silicon content prediction in BF molten iron. The range of silicon content y in molten iron of BF is assessed as:

$$y \in (\bar{y} - \frac{\sigma}{\sqrt{n}} U_{\frac{\alpha}{2}}, \bar{y} + \frac{\sigma}{\sqrt{n}} U_{\frac{\alpha}{2}}) \tag{14}$$

where $\bar{y}$ denotes the mean value of silicon content of all samples; n represents sample capacity; σ refers to the standard deviation of all samples; α shows the confidence level. In practical research, $\alpha = 0.05$, reflecting 95% confidence level. Since the silicon content in molten iron complies with the normal distribution, $U_{\frac{\alpha}{2}} = 1.96$. For BF(a) and BF(b), the range of silicon content in molten iron estimated by interval are (0.253,0.910) and (0.349,0.973), respectively.

IV. CONCLUSION

In this study, XGBoost and LightGBM algorithms based on GBDT algorithm are proposed for silicon content prediction in molten iron of two BF systems, which is formulated as a regression problem. To be specific, XGBoost algorithm introduces regular terms to the GBDT algorithm to prevent the overfitting phenomenon, while LightGBM algorithm is based on the GBDT algorithm as well, and it simplifies and expedites the calculation by adopting GOSS and EFB strategies. According to the experimentally achieved results on two real BF datasets, the good performance of the mentioned proposed two methods is verified. In comparison with the classical algorithms of Lasso, Random Forest (RF), Support Vector Machine (SVM) and Gradient Boosting Decision Tree (GBDT), the proposed predictor has been significantly optimized as indicated by the R-square and MSE. The predicted results provide guidance for controlling the complex BF iron-making process. It is noteworthy that it is adopted to control the temperature of BF, so BF exhibits a stable state. The original angle of the present study is novel, both XGBoost algorithm and LightGBM algorithm are popular machine learning algorithms over the past five years. It is employed initially to study in BF system, which is of certain guiding significance.

REFERENCES

[1] H. Saxen, C. Gao, and Z. Gao, "Data-driven time discrete models for dynamic prediction of the hot metal silicon content in the blast furnace—A review," *IEEE Trans. Ind. Informat.*, vol. 9, no. 4, pp. 2213–2225, Nov. 2013.

[2] S. Ueda, S. Natsui, H. Nogami, J.-I. Yagi, and T. Ariyama, "Recent progress and future perspective on mathematical modeling of blast furnace," *ISIJ Int.*, vol. 50, no. 7, pp. 914–923, 2010.

[3] X. G. Liu and F. Liu, *Optimization and Intelligent Control System of Blast Furnace Ironmaking Process*. Beijing, China: Metallurgical Industry Press, 2003.

[4] X. Liu, "Study on silicon predictive control model of blast furnace temperature based on Bayesian network," Ph.D. dissertation, School Math., Zhejiang Univ., Hangzhou, China, Tech. Rep., 2004.

[5] X. Liu and S. Gong, "Application of fuzzy Bayesian network to prediction of silicon content in molten iron of blast furnace," *Metall. Automat.*, vol. 29, no. 5, pp. 30–32, 2005.

[6] C. Gao and Z. Zhou, "Chaos analysis of blast furnace smelting process," *Acta Physica Sinica*, vol. 54, no. 4, pp. 1490–1494, 2009.

[7] C. Gao, "Chaos dynamics of blast furnace ironmaking process," Ph.D. dissertation, School Math., Zhejiang Univ., Hangzhou, China, 2004.

[8] C. Gao, L. Jian, X. Liu, J. Chen, and Y. Sun, "Data-driven modeling based on volterra series for multidimensional blast furnace system," *IEEE Trans. Neural Netw.*, vol. 22, no. 12, pp. 2272–2283, Dec. 2011.

[9] W. Chen, B.-X. Wang, and H.-L. Han, "Prediction and control for silicon content in pig iron of blast furnace by integrating artificial neural network with genetic algorithm," *Ironmaking Steelmaking*, vol. 37, no. 6, pp. 458–463, Aug. 2013.

[10] W. H. Wang, "Study on prediction model of blast furnace temperature based on wavelet analysis theory," Ph.D. dissertation, Zhejiang Univ., Hangzhou, China, 2005.

[11] X. G. Liu and W. H. Wang, "Application of wavelet analysis to improve the prediction of silicon content in molten iron," *Steel*, no. 8, pp. 15–17 and 37, 2005.
[12] W. H. Wang, X. G. Liu, and X. Y. Liu, "Prediction model of silicon content in molten iron based on stochastic forest algorithm," *Metall. Automat.*, vol. 38, no. 5, pp. 33–38, 2014.
[13] J. G. Li and X. L. Min, "Neural network prediction model of silicon content in molten iron of blast furnace," *J. Hebei Inst. Technol.*, vol. 24, no. 3, pp. 17-23, 2002.
[14] Y. T. Wang and J. C. Zhou, "Prediction method of silicon content in molten iron of blast furnace based on neural network model," *Steel*, vol. 35, no. 3, pp. 66–71, 1999.
[15] W. Chen, B.-X. Wang, and H.-L. Han, "Prediction and control for silicon content in pig iron of blast furnace by integrating artificial neural network with genetic algorithm," *Ironmaking Steelmaking*, vol. 37, no. 6, pp. 458–463, Aug. 2010.
[16] B. Yao, T. Yang, and X. Ning, "An improved artificial neural network model for predicting silicon content of blast furance hot metal," *J. Univ. Sci. Technol. Beijing*, vol. 7, no. 4, pp. 269–276, 2000.
[17] W. Matias and S. Henrik, "Application of nonlinear time series analysis to the prediction of silicon content of pig iron," *ISIJ Int.*, vol. 42, no. 3, pp. 316–318, 2002.
[18] H. Uusi, "Prediction of hot metal silicon in blast furance by multivariate time serial modeling," *Tech. Rep.-Lab Product Process Des.*, vol. 45, no. 12, pp. 385–390, 2005.
[19] T. Bhattacharya, "Prediction of silicon content in blast furnace hot metal using partial least squares (PLS)," *ISIJ Int.*, vol. 45, no. 12, pp. 1943–1945, 2005.
[20] J. S. Zeng, X. G. Liu, and S. H. Luo, "Application of principal component regression and partial least squares method in blast furnace smelting," *J. Zhejiang Univ.*, vol. 36, no. 1, pp. 33–36, 2009.
[21] J. Cai, J. Zeng, and S. Luo, "A state space model for monitoring of the dynamic blast furnace system," *ISIJ Int.*, vol. 52, no. 12, pp. 2194–2199, 2012.
[22] P. Zhou, H. Song, H. Wang, and T. Chai, "Data-driven nonlinear subspace modeling for prediction and control of molten iron quality indices in blast furnace ironmaking," *IEEE Trans. Control Syst. Technol.*, vol. 25, no. 5, pp. 1761–1774, Sep. 2017.
[23] L. Jian and X. G. Liu, "Application of support vector machine in prediction of silicon content in molten iron," *Metall. Automat.*, no. 3, pp. 33–36, 2005.
[24] L. Jian, "Application of support vector machine in temperature prediction of blast furnace," Ph.D. dissertation, Zhejiang Univ., Hangzhou, China, 2006.
[25] L. Jian and S. H. Gong, "Multiclass classification of silicon content in molten iron based on support vector machine," *J. Zhejiang Univ.*, no. 3, pp. 282–285, 2007.
[26] L. Jian, C. Gao, L. Li, and J. Zeng, "Application of least squares support vector machines to predict the silicon content in blast furnace hot metal," *ISIJ Int.*, vol. 48, no. 11, pp. 1659–1661, 2008.
[27] L. Jian, C. Gao, and Z. Xia, "A sliding-window smooth support vector regression model for nonlinear blast furnace system," *steel Res. Int.*, vol. 82, no. 3, pp. 169–179, Mar. 2011.
[28] C. Gao, Q. Ge, and L. Jian, "Rule extraction from fuzzy-based blast furnace SVM multiclassifier for decision-making," *IEEE Trans. Fuzzy Syst.*, vol. 22, no. 3, pp. 586–596, Jun. 2014.
[29] C. Gao, L. Jian, and S. Luo, "Modeling of the thermal state change of blast furnace hearth with support vector machines," *IEEE Trans. Ind. Electron.*, vol. 59, no. 2, pp. 1134–1145, Feb. 2012.
[30] L. Jian, C. Gao, and Z. Xia, "Constructing multiple kernel learning framework for blast furnace automation," *IEEE Trans. Autom. Sci. Eng.*, vol. 9, no. 4, pp. 763–777, Oct. 2012.
[31] T. Q. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd SIGKDD Conf. Knowl. Discovery Data Mining*, 2016, pp. 785–794.
[32] G. Ke, Q. Meng, and T. Finley, "LightGBM: A highly efficient gradient boosting decision tree," Microsoft, Albuquerque, NM, USA, Tech. Rep., 2017.
[33] J. Li, C. Hua, Y. Yang, and X. Guan, "A novel MIMO T-S fuzzy modeling for prediction of blast furnace molten iron quality with missing outputs," *IEEE Trans. Fuzzy Syst.*, early access, Apr. 1, 2020, doi: 10.1109/TFUZZ.2020.2983667.
[34] S. Luo, Z. Dai, T. Chen, H. Chen, and L. Jian, "A weighted SVM ensemble predictor based on AdaBoost for blast furnace ironmaking process," *Appl. Intell.*, vol. 50, pp. 1997–2008, Feb. 2020.
[35] J. Zhao, Y. Xu, and H. Fujita, "An improved non-parallel universum support vector machine and its safe sample screening rule," *Knowl.-Based Syst.*, vol. 170, pp. 79–88, Apr. 2019.
[36] Y. Zhang, M. Ni, C. Zhang, S. Liang, S. Fang, R. Li, and Z. Tan, "Research and application of AdaBoost algorithm based on SVM," in *Proc. IEEE 8th Joint Int. Inf. Technol. Artif. Intell. Conf. (ITAIC)*, May 2019, pp. 662–666.
[37] S. Sun, X. Xie, and C. Dong, "Multiview learning with generalized eigenvalue proximal support vector machines," *IEEE Trans. Cybern.*, vol. 49, no. 2, pp. 688–697, Feb. 2019.
[38] J. Sun, H. Li, H. Fujita, B. Fu, and W. Ai, "Class-imbalanced dynamic financial distress prediction based on AdaBoost-SVM ensemble combined with SMOTE and time weighting," *Inf. Fusion*, vol. 54, pp. 128–144, Feb. 2020.
[39] J. Deng and W. Yi, "Supervised learning based online tracking filters: An XGBoost implementation," Tech. Rep., 2020.
[40] L. Guang-yu and H. Geng, "The behavior analysis and achievement prediction research of college students based on XGBoost gradient lifting decision tree algorithm," in *Proc. 7th Int. Conf.*, 2019, pp. 289–294.
[41] J. Ma, Z. Yu, Y. Qu, J. Xu, and Y. Cao, "Application of the XGBoost machine learning method in PM2.5 prediction: A case study of shanghai," *Aerosol Air Qual. Res.*, vol. 20, no. 1, pp. 128–138, 2020.
[42] X. Zhang, T. Deng, and G. Jia, "Nuclear spin-spin coupling constants prediction based on XGBoost and LightGBM algorithms," *Mol. Phys.*, vol. 118, no. 14, pp. 1–10, 2019.
[43] J. Nobre and R. F. Neves, "Combining principal component analysis, discrete wavelet transform and XGBoost to trade in the financial markets," *Expert Syst. Appl.*, vol. 125, pp. 181–194, Jul. 2019.
[44] Y. Wang and Y. Guo, "Forecasting method of stock market volatility in time series data based on mixed model of ARIMA and XGBoost," *China Commun.*, vol. 17, no. 3, pp. 205–221, Mar. 2020.
[45] Z. Zhan, Z. You, and Y. Zhou, "An efficient LightGBM model to predict protein self-interacting using Chebyshev moments and Bi-gram," in *Intelligent Computing Theories and Application*. Cham, Switzerland: Springer, 2019.
[46] J. Zhang, D. Mucs, and U. Norinder, "LightGBM: An effective and scalable algorithm for prediction of chemical toxicity–application to the Tox21 and mutagenicity data sets," *J. Chem. Inf. Model.*, vol. 59, no. 10, pp. 4150–4158, 2019.
[47] Y. Ju, G. Sun, Q. Chen, M. Zhang, H. Zhu, and M. U. Rehman, "A model combining convolutional neural network and LightGBM algorithm for ultra-short-term wind power forecasting," *IEEE Access*, vol. 7, pp. 28309–28318, 2019.

SHIHUA LUO received the B.S. degree and the M.S. degree in mathematics from Jiangxi Normal University, China, in 1998 and 2001, respectively, and the Ph.D. degree in operational research and cybernetics from Zhejiang University, China, in 2007. He is currently a Professor with the Jiangxi University of Finance and Economics. His research interests include modeling and optimization of complex systems, especially on the extraction of non-linear characteristics of chemical process and financial process.

TIANXIN CHEN received the B.S. degree in statistics from the Lanzhou University of Finance and Economics, China, in 2016, and the M.S. degree in statistics from the Jiangxi University of Finance and Economics, in 2019, where he is currently pursuing the Ph.D. degree. His research interests include multivariate statistical analysis, data mining, and machine learning algorithms.