Hindawi
Mathematical Problems in Engineering
Volume 2019, Article ID 5309852, 13 pages
https://doi.org/10.1155/2019/5309852

Research Article

Towards Optimization of Boosting Models for Formation Lithology Identification

Yunxin Xie,[1] **Chenyang Zhu**,[2] **Yue Lu**,[2] **and Zhengwei Zhu**[3]

[1]*School of Petroleum Engineering, Changzhou University, Changzhou 213100, China*
[2]*Electronics and Computer Science, University of Southampton, University Road, Southampton SO17 1BJ, UK*
[3]*School of Information Science and Engineering, Changzhou University, Changzhou 213100, China*

Correspondence should be addressed to Chenyang Zhu; cz4g16@soton.ac.uk

Received 6 June 2019; Accepted 24 July 2019; Published 14 August 2019

Academic Editor: Bogdan Smolka

Lithology identification is an indispensable part in geological research and petroleum engineering study. In recent years, several mathematical approaches have been used to improve the accuracy of lithology classification. Based on our earlier work that assessed machine learning models on formation lithology classification, we optimize the boosting approaches to improve the classification ability of our boosting models with the data collected from the Daniudi gas field and Hangjinqi gas field. Three boosting models, namely, AdaBoost, Gradient Tree Boosting, and eXtreme Gradient Boosting, are evaluated with 5-fold cross validation. Regularization is applied to the Gradient Tree Boosting and eXtreme Gradient Boosting to avoid overfitting. After adapting the hyperparameter tuning approach on each boosting model to optimize the parameter set, we use stacking to combine the three optimized models to improve the classification accuracy. Results suggest that the optimized stacked boosting model has better performance concerning the evaluation matrix such as precision, recall, and $f1$ score compared with the single optimized boosting model. Confusion matrix also shows that the stacked model has better performance in distinguishing sandstone classes.

1. Introduction

Well log data contain rich geological information, which is a synthesized reflection of formation lithology and physical properties. Therefore, geological interpretation of well log data and interpretation accuracy are crucial. Regular interpretation methods such as statistical approach have low accuracy and slow efficiency. A reliable way to understand subterranean lithology is obtaining core samples or cuttings from the reservoirs. However, it is expensive and there is always some depth uncertainty [1]. Hence, it is urgent to explore more effective, accurate, and economical methods to make better use of well log data. Because of the development of different logging tools such as wireline and logging while drilling, we can collect numerous data in the petroleum industry [2]. Computational techniques can help with the analyzing of the log data with high dimension. With the popularization and development of computer technology, sophisticated tasks such as lithology identification become more intelligent. In recent years, many researchers applied different algorithms on lithology identification and achieved great performance through updating and improving algorithms.

In recent years, there has been an increasing amount of literature on identifying lithology classes with machine learning approaches. Rogers et al. applied the back propagation approaches in neural network technique to obtain patterns to classify the lithology such as limestone, dolomite, and shale [3]. Rafik and Kamel compared the prediction accuracy of three approaches, namely, neural networks technique, generalized additive model, and alternating conditional expectations, to predict permeability using well logs [4]. Their results showed that the regression with alternating conditional expectations has the best performance. However, the neural network models are mainly applied to regression problems. The support vector machine models and random forest models have better performance in classification problems. Al-Anazi and Gates presented a support vector machine (SVM) classification approach to identify lithology in a variegated sandstone reservoir with

fuzzy logic [5]. Their result showed that the performance of SVM is better than neural network in lithology identification. Sebtosheikh and Salehi concluded that the prediction accuracy of an SVM classifier with normalized polynomial kernel function could be improved by using the tuning approach to obtain the optimum parameter set for the kernel function [6]. Besides SVM, several studies have used random forest models to classify lithology. Cracknell and Reading concluded that random forest models have better lithology classification performance compared with other machine learning models based on the geological and geochemical data [7]. Moreover, Ao et al. proposed a pruning random forest algorithm to select appropriate features on seismic interpretation, which improves the prediction accuracy and generalization of the model [8]. Also, Bestagini et al. demonstrated that a gradient boosting classifier could help with the feature augmentation and improve the facies classification accuracy based on wireline logging measurements [9].

Existing research that identifies lithology classes uses different lithology features. Thus, examining the performance of various machine learning models from existing studies is technically challenging. Our previous work analyzed the performance of five machine learning models with the same lithology feature set and concluded that ensemble methods perform better than classifiers such as support vector machine and neural network with a small feature set. Boosting approach is one of the ensemble methods. Moreover, our previous results showed that boosting method has a better capability in distinguishing sandstone classes compared with other classifiers [10]. Based on our previous work, this study conducts a meticulous evaluation over three boosting models, namely, AdaBoost, Gradient Tree Boosting (GTB), and eXtreme Gradient Boosting (XGBoost) with 5-fold cross validation. The models are optimized with regularization and stacking. Beyond the basic parameter set like the learning rate and the number of boosting stages, we investigate the impact of regularization parameters, namely, the fraction of samples to be applied for fitting the individual base learners and the number of features to be considered when determining the best split for individual base learners, on the performance of the model. Hyperparameter approach is then adopted to get the optimum values for the parameter set of each model. In the end, stacking is used to merge the three optimized boosting models. The performance of three boosting models and the stacked model is analyzed with accuracy scores and confusion matrix. Results show that the stacked model not only outperforms individual models in accuracy but also has better performance in distinguishing sandstone classes.

The paper is composed as follows. Section 2 provides a methodology for boosting models we used in this paper, together with the regularization and stacking optimization process. Section 3 describes the process that we train and optimize boosting models from the collected data. In Section 4, we presented the performance of individual models and the stacked model with accuracy matrix and confusion matrix. Section 5 summarizes the performance of the stacked model on classification accuracy and outlines future work.

2. Methodology

2.1. Boosting. With the enormously increased data, significantly improved algorithms, and powerful computing hardware, machine learning has excellent breakthroughs in many diverse areas such as recommendation systems [11], speech recognition [12], and natural language processing [13]. Machine learning is defined as statistical approaches to detect the data patterns that can be used to predict the properties of unseen data [14]. Supervised learning is one of the machine learning tasks that generate prediction models based on the labelled training data, which has two major learning divisions, namely, supervised regression and supervised classification. Our work performed the supervised classification on a labelled dataset with multiple classes from several logging wells. Boosting approach is adopted to solve the multiclass classification problem.

Boosting is one of the powerful machine learning procedures that produce accurate prediction rules by merging several fairly inaccurate rules [15]. Different from the bagging approach [16], boosting sequentially generates base models and combines the base models into the final accurate model. Boosting trains the weak classifiers on a weighted dataset sequentially, and the performance of the previous classifiers determines the weight of each data item [17]. Data items that are incorrectly predicted or misclassified by the preceding weak classifier would be assigned with a higher weight. Consider a two-class supervised classification problem with the training set $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, where each x_i belongs to the instance set X and each y_i belongs to the binary label set $Y = \{-1, +1\}$. Assuming the number of boosting iterations is M, boosting fits the model $f(x)$ which can be written as a weighted majority vote as $f(x) = \sum_{m=0}^{M} f_m(x) = \sum_{m=0}^{M} a_m h_m(x)$, where $h_m(x)$ denotes the sequentially built model and a_m denotes the learning rate for this mth model. The loss function $L(y, f(x))$ is used to tell how well the model fits the data by measuring the difference between the predicted values $f(x)$ and the true values y. Most of the boosting algorithms can be seen to find a_m and $h_m(x)$ to minimize the loss function $\sum_{i=1}^{n} L(y_i, f_{m-1}(x_i) + a_m h_m(x_i))$ at each iteration m.

2.1.1. AdaBoost. The AdaBoost algorithm, articulated by Freund and Schapire in 1995, is one of the most useful boosting algorithms for classification problems [18]. Algorithm 1 provides the algorithm for AdaBoost. Considering a two-class classification problem, each data point in the training set $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ is provided with an weighting parameter w_n for each x_i, which is set to $1/n$ initially. At each stage of the training process, AdaBoost trains a new classifier using the data with different weighting parameters that are altered based on the performance of the previous classifier. Those misclassified data points get greater weights when trained in the next stage training. Finally, when the number of base classifiers reaches the number of iterations, these base classifiers are combined to form a strong classifier based on the weight of each base classifier a_m and the corresponding base classifier $h_m(x)$.

(1) Given $(x_1, y_1), \ldots, (x_n, y_n)$, where $x_i \in X$, $y_i \in Y = \{-1, +1\}$; the number of iterations: M
(2) Initialize equal weights for each data item $w_i = 1/N$, $i = 1, 2, \ldots, N$
(3) For $m = 1$ to M:
  (a) Fit a classifier $h_m(x)$ to training data by minimizing the weighted error function $J_m = \sum_{i=1}^{N} w_n^{(m)} I(h_m(x_i) \neq y_i)$
  (b) Compute $\text{err}_m = \sum_{i=1}^{N} w_i^{(m)} I(h_m(x_i) \neq y_i) / \sum_{i=1}^{N} w_i^{(m)}$ and evaluate $\alpha_m = \ln\{(1 - \text{err}_m)/(\text{err}_m)\}$
  (c) Update the weight for each data item $w_n^{(m+1)} = w_n^{(m)} \exp\{\alpha_m I(h_m(x_i) \neq y_i)\}$
(4) Construct final model $h(x)$ with a_m and weak model $h_m(x)$: $h(x) = \text{sign}(\sum_{m=1}^{M} \alpha_m h_m(x))$.

Algorithm 1: AdaBoost algorithm, modified from [17].

(1) Given $(x_1, y_1), \ldots, (x_n, y_n)$. The number of iterations: M
(2) Initialize $f_0(x) = a_0 = \mathbf{argmin} \sum_{i=1}^{n} L(y_i, a_0)$
(3) For $m = 1$ to M:
  (a) Compute $\forall i \cdot i \in 1 \ldots n \Longrightarrow z_{im} = -[\partial L(y_i, f(x_i))/\partial f(x_i)]_{f(x)=f_{m-1}(x)}$
  (b) Fit base model $h_m(x)$, $h_m(x) = \sum_{j=1}^{J_m} b_{jm} I(x \in R_{jm})$
  (c) Calculate $a_m h_m(x)$ to minimize the loss function $a_m h_m(x) = \mathbf{argmin} \sum_{i=1}^{n} L(y_i, f_{m-1}(x_i) + a_m h_m(x_i))$
  (d) Update current model $f_m(x)$ with previous model $f_m(x)$ and the constrained $\rho a_m h_m(x)$, $f_m(x) = f_{m-1}(x) + \rho a_m h_m(x)$ $(x \in R_{jm})$
(4) Calculate the final boosting model $f(x) = \sum_{m=0}^{M} f_m(x)$.

Algorithm 2: Gradient Tree Boosting algorithm, modified from [20, 21].

Basic AdaBoost models are used to handle the binary classification problems. Hastie et al. proposed a new algorithm named stagewise additive modelling using a multiclass exponential loss function (SAMME) that extends the original AdaBoost algorithm to support multiclass classification [19]. SAMME is similar to AdaBoost except for the computation for learning rate a_m. In SAMME, a_m is calculated as equation (1), where K stands for the number of classes in multiclass classification. So, the misclassified data points would gain more weight during the training process in SAMME than AdaBoost. Also in SAMME, the final model is also calculated differently with equation (2), which combines the weak classifiers and predicts the final class with the most got class k. In this case study, we applied SAMME in AdaBoost algorithm to perform the multiclass lithology classification task:

$$a_m = \ln\left\{\frac{1 - \text{err}_m}{\text{err}_m}\right\} + \ln(K - 1), \tag{1}$$

$$h(x) = \mathbf{argmax} \sum_{m=1}^{M} a_m I\left(h_m(x) = k\right). \tag{2}$$

2.1.2. Gradient Tree Boosting. Gradient Tree Boosting (GTB) is another stepwise boosting approach that builds the final accurate model by adding base models sequentially. The base models of each training stage are trained to best reduce the loss function. Friedman proposed the gradient boosting method [20] and the refined generalized boosting model that uses regression tree as the base model [21]. The GTB training algorithm is presented in Algorithm 2. The model is initialized with a value a_0. A gradient descent process is applied at each training stage m to minimize the loss function $\sum_{i=1}^{n} L(y_i, f_{m-1}(x_i) + a_m h_m(x_i))$. Assume there are M iterations. For each training stage, first-order Taylor expansion of loss function is evaluated and z_{im} is calculated to get the direction to minimize $a_m h_m(x)$. In GTB, regression tree models are taken as base learner as it can simplify the decision-making process [22]. The regression tree model selects the feature with the highest information gain as the root node. Then, the root node splits and adds features with the next best information gain as its child node. This splitting and adding process is then repeated recursively for the new grandchild nodes. The input space is partitioned into J_m joint regions $R_{1m}, R_{2m}, \ldots, R_{jm}$ with predicted constant values $b_{1m}, b_{2m}, \ldots, b_{jm}$. The base learner $h_m(x)$ is the sum of these predicted values. Then, $a_m h_m(x)$ is calculated to minimize the loss function. At last, the new model $f_m(x)$ is updated with the sum of previous model $f(m-1)$ and $a_m h_m(x)$. However, a large number of iterations lead to poor generalized models. To solve this problem, Friedman's algorithm uses a shrinkage parameter ρ on the calculated model $a_m h_m(x)$ to constrain the learning rate of the training process [21]. The training process for the boosting model is slow and requires more boosting iterations to train the model.

2.1.3. eXtreme Tree Boosting. GTB algorithm adopts first-order Taylor expansion of loss function for approximate tree learning, while Chen and Guestrin proposed a scalable tree boosting algorithm eXtreme Tree Boosting (XGBoost) by using higher-order approximation to learn a better tree structure [23]. The XGBoost algorithm works as Algorithm 3. For each iteration m in boosting steps, it calculates the multiplier z_{im} and g_{im} with first-order Taylor expansion

(1) Given $(x_1, y_1), \ldots, (x_n, y_n)$. The number of iterations: M
(2) Initialize $f_0(x) = a_0 = \mathbf{argmin}\sum_{i=1}^{n} L(y_i, a_0)$
(3) For $m = 1$ to M:
  (a) Compute $z_{im} = -[\partial L(y_i, f(x_i))/\partial f(x_i)]_{f(x)=f_{m-1}(x)}$, for $i = 1, \ldots, n$, and $g_{im} = -[\partial^2 L(y_i, f(x_i))/\partial f\ (x_i)^2]_{f(x)=f_{m-1}(x)}$, for $i = 1, \ldots, n$
  (b) Fit base model $h_m(x)$, $h_m(x) = \sum_{j=1}^{J_m} b_{jm} I(x \in R_{jm})$.
  (c) Determine the leaf weight for the learnt structure with z_{im} and g_{im}
  (d) Update current model $f_m(x)$ with previous model $f_m(x)$ and the constrained $\rho a_m h_m(x)$, $f_m(x) = f_{m-1}(x) + \rho a_m h_m(x)(x \in R_{jm})$
(4) Calculate the final boosting model $f(x) = \sum_{m=0}^{M} f_m(x)$.

Algorithm 3: XGBoost algorithm, modified from [23].

and higher-order Taylor expansion and uses the multiplier to calculate the leaf weights that determine the weak tree model structure. Then, the weak tree models $f_m(x)$ are combined to generate the final boosting model $f(x)$.

2.2. Regularization. The task of machine learning is to fit an accuracy model to describe a pattern for both training data or untrained data. If a trained model is excessively complex and too fit to the training data, the overfitting occurs so that the trained model has poor predictive performance. Regularization technique is used to control the model complexity and improve the generalization of the model, which is a trade-off between training loss and model complexity. Regularization can be seen to minimize the loss function $\sum_{i=1}^{n} L(y_i, f(x_i) + ah(x_i)) + C * \Omega(h(x_i))$ where $\Omega(h(x_i))$ denotes the complexity of the model and C denotes the penalization parameter on model complexity. Lasso regularization ($L1$) and Ridge regularization ($L2$) are the most commonly used regularization techniques in supervised learning. $L1$ regularization can be seen to minimize the loss function $\sum_{i=1}^{n} L(y_i, f(x_i) + ah(x_i)) + C * \sum \min\|w\|$, where the penalization C is applied on the absolute value of the weights [24]. $L2$ regularization is used minimize the loss function $\sum_{i=1}^{n} L(y_i, f(x_i) + ah(x_i)) + C * \sum \min\|w\|^2$ [25]. The penalization C determines how much to penalize the weights to make the model simpler and avoid overfitting. In $L1$ regularization, different penalization parameters can be tried on different independent subsample of data to get the optimal value. Chen and Guestrin have used several regularization techniques to control the complexity of the tree model, namely, regularization on the basis function expansion, regularization on the individual tree model, and randomization, based on $L2$ regularization [23]. Chen's approach tried to minimize the objective function $\sum_{i=1}^{n} L(y_i, f(x_i) + ah(x_i)) + \Omega(h(x_i))$ in each iteration, and they used the second-order approximation to quickly optimize the objective in general setting [26]. In Chen's approach, they expand $\Omega(h(x_i))$ with equation (3). Here, γT denotes the number of leaves in the tree and $(1/2)\lambda\|w\|^2$ denotes the $L2$ norm of leaf scores. By expanding $\Omega(h(x_i))$, the optimal solution is to minimize objective in equation (4), which can be rewritten to equation (5) with a second-order approximation [23]. Then, the optimal leaf weight w_j^* of leaf j can be computed by equation (6). In this case study, we applied $L1$ regularization with subsampling on the GTB model and $L2$ regularization on the XGBoost model:

$$\Omega(h(x_i)) = \gamma T + \frac{1}{2}\lambda\|w\|^2, \tag{3}$$

$$L = \sum_{i=1}^{n} L(y_i, f_{m-1}(x_i) + a_m h_m(x_i)) + \gamma T + \frac{1}{2}\lambda\|w\|^2, \tag{4}$$

$$L \simeq \sum_{j=1}^{T}\left[\left(\sum_{i\in I_j} z_{im}\right)w_j + \frac{1}{2}\left(\sum_{i\in I_j} g_{im}\right)w_j^2\right] + \gamma T, \tag{5}$$

$$w_j^* = -\frac{\sum_{i\in I_j} z_{im}}{\sum_{i\in I_j} g_{im}}. \tag{6}$$

2.3. Stacked Generalization. Ensemble methods are learning strategies that take a weighted vote over the predictions of a collection of constructed classifiers to classify new examples [27]. These stacked classifiers should be accurate and diverse to make the ensemble of the classifiers with better performance than any of the individual classifiers [28]. The stacking method works as shown in Figure 1, which implements hard and soft voting for the classifiers. In hard voting, the final class label is predicted as the label with the majority vote by the classification models, while soft voting averages the class probabilities to predict the final class. In this case study, we applied soft voting with equal weights for the AdaBoost classifier, Gradient Tree Boosting classifier, and XGBoost classifier.

2.4. Evaluation. Overfitting and underfitting are two critical issues for learning models. Overfitting occurs when the model is too complicated and fits the training data too well, which has high variance and low bias. Underfitting, on the contrary, has a high bias but low variance because the model is too simple. Evaluating the accuracy of classifiers is helpful to predict the result of unseen data [29]. Cross validation is

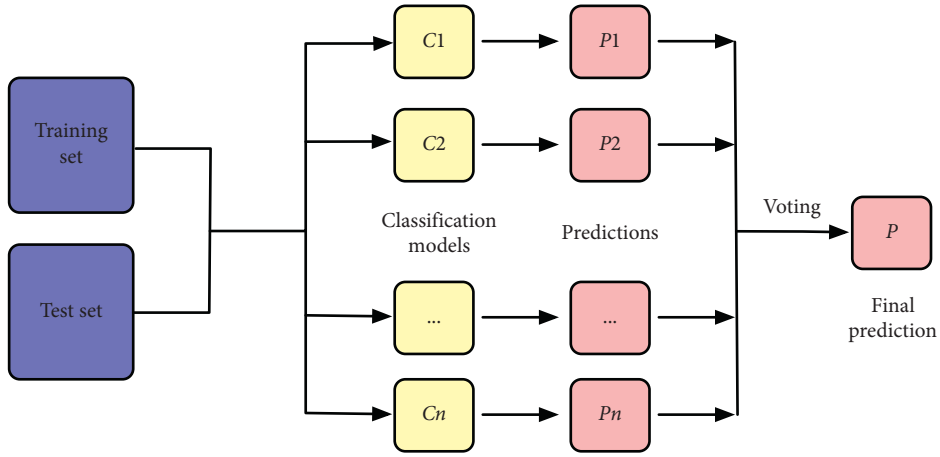

Figure 1: Ensemble method working process.

one of the evaluation approaches with resampling procedure on a limited number of data. A k-fold cross validation splits the dataset D into k subsets with approximate equal size randomly. For each iteration $i \in \{1, 2, \ldots, k\}$, the classifier is trained on data D/D_i and tested on D_i. The accuracy of the model is estimated by averaging the k results.

There are several evaluation metrics to measure the performance of a multilabel classification model, namely, accuracy score, precision, recall, and $f1$ score. Accuracy score measures the portion of the classes that are correctly predicted. Precision defines the ratio of positive classes that are predicted by the model correctly to all predicted positive classes. Recall defines the ratio of positive classes that are predicted by the model correctly to all actual positive classes. The $f1$ score is a balanced average of the precision and recall, which can be presented as

$$f1 = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}. \quad (7)$$

In this case study, a generalizable multiclass classification model is built to identify lithology classes from different areas. We use 5-fold cross validation with accuracy score and $f1$ score as the evaluation matrix to control the model complexity and improve the model generalization.

3. Case Study

3.1. Dataset. Ordos Basin, which is located in the central-west of China, is a large multicycle cratonic basin. The proved reserves of natural gas, coal-bed gas, and coal in Ordos Basin are ranking first in China. The relevant research data are collected from the Carboniferous and Permian clastic strata, which are the main gas-bearing continental strata in Ordos Basin. The major lithologies of these formations are sandstone, mudstone, and coal. Also, a small amount of carbonate rocks can be found. The data used in this study are obtained from two gas fields, namely, the Daniudi gas field (DGF) and Hangjinqi gas field (HGF), which are two productive gas fields in north Ordos Basin. A large amount of log data and core analysis reports of the corresponding depth of twelve wells are used to identify lithologies.

In this study, based on the grounds of the sandstone grain diameter classification of the oil industry standard of China, five sandstone types are classified in the light of different grain size ranges. The lithologies are divided into five types, namely, sandstone, siltstone (S), mudstone (M), coal (C), and carbonate rock (CR). Based on the grain diameters of the sandstones, we categorized the sandstone classes into pebbly sandstone (PS), coarse sandstone (CS), medium sandstone (MS), and fine sandstone (FS). Seven well log parameters are collected from twelve wells of the two gas fields, which provide information of characteristics of the rock or sediment in a borehole. Table 1 shows the log parameters with their descriptions. Thus, we classify eight lithology classes in the DGF area and seven lithology classes in the HGF area during model selection and evaluation.

3.2. Data Processing. We collected 915 and 1,238 log readings from several wells in the DGF area and HGF area, respectively. Seven logging parameters mentioned in Section 3.1 are extracted to build the training and test dataset. With the collected data, we created two datasets, one with 915 instances of data with seven features and the other with 1,238 instances of data with seven features. We used the data from the DGF to train and optimize the boosting models; then, the performance of the optimized model was evaluated by using the datasets from both areas. During the training process, the data were randomly split into two datasets, which takes 80% as the training set and 20% as the test set.

3.3. Multiclass Classification Model Training. In this case study, we evaluated the performance of unoptimized boosting models, namely, AdaBoost, Gradient Tree Boosting, and XGBoost. Then, we optimized the Gradient Tree Boosting approach and XGBoost with the regularization technique. In the end, we optimized the three classifiers by applying soft voting with equal weights on them. It is essential for supervised models to obtain appropriate parameter set to be trained to predict the unseen data. This case study used tuning as a parameter selection process to improve the model classification accuracy. The tuning process uses a performance

Table 1: Logging parameters with descriptions.

Logging parameter (unit)	Abbreviation	Descriptions
Gamma-ray (API)	GR	Measure naturally occurring gamma radiation
Acoustic (μs/m)	AC	Measure capacity to transmit seismic waves
Density (g/cm^3)	DEN	Provide formation's bulk density
Compensated neutron (%)	CNL	Measure thermal and epithermal neutron
Deep lateral (Ωm)	LLD	Measure deep formation resistivity
Shallow lateral (Ωm)	LLS	Measure shallow formation resistivity
Caliper (cm)	CAL	Measure the diameter and shape of the formation

matrix to evaluate the same model with various parameter sets and selects the parameter set that has the best classification accuracy. Grid search approach was adopted with the accuracy score to optimize the parameter set. We iterate through all the combinations of the parameter set to get the best one. As mentioned in Section 2, the boosting model requires two parameters to be tuned, namely, the number of iterations M and the learning rate a_m. In regression tree models, the parameter of a tree needs to be tuned, namely, the minimum number of samples required at a leaf node, maximum depth of the individual tree, and the minimum number of samples required to split an internal node. In addition, the parameter of regularization also needs to be tuned, namely, the parameters that constraint and penalize the complexities of individual tress and subsampling of row and column. Table 2 provides the parameters that require tuning in the AdaBoost classifier, GTB classifier, and XGBoost classifier, including the parameters for regularization of models.

The grid search will train the data through all the permutation of the parameter set. It takes a lot of time and calculation power to train the data if the parameter set is large. Narrowing down the range of parameter set is helpful to improve the training efficiency and accuracy. In this case study, we adopted the validation curve and test set deviance to determine the range of the parameter set.

Validation scores show whether the trained model is overfitting or underfitting for the parameter values. When the model is underfitting, the validation score increases with the training score and vice versa. We used 5-fold cross validation to calculate the validation curve for AdaBoost and XGBoost to determine the proper parameter search range. Figures 2 and 3 show the validation curve for parameters that need to be tuned for the AdaBoost classifier and XGBoost classifier, respectively. Results indicate that both the tree model parameters as well as the regularization parameter require tuning as the validation curve does not change dramatically. Table 2 shows the determined search range of parameter of AdaBoost and XGBoost.

Test set deviance was also applied to determine the search range for tree model parameters and regularization parameters based on the boosting iterations. Figure 4 presents the test set deviance for the GTB model with different parameter sets. We used the cross-entropy loss to measure the performance of a model over various stages of the boosting iteration. Results show that the test deviance converges within 200 boosting iterations for the parameter set. We determined the search range for boosting iterations is within 100–200. The fourth figure shows that a smaller learning rate will get a better result. Besides, the tree model parameters require tuning as test deviance are almost the same for different parameter values. We determined the search range for GTB in Table 2.

The tuning process iterates through all the combinations of the parameter set within the search range. Each time, the training data are trained with the specific parameter set with 5-fold cross validation. And the optimum parameter set that has the best cross validation score was selected to train the data from the DGF and HGF area. Table 2 presents the optimum parameter set for the three classification models.

3.4. Prediction Evaluation. To prove that the optimized boosting model works on general lithology classification problems, we evaluate the model with lithology datasets from two different areas. Besides, random seeds are also examined in the model to promise the generalization of the optimized boosting model. For the two datasets from two areas, we trained the data ten times and calculated the matrix of precision, recall, and $f1$ score to assess the performance of AdaBoost, GTB, XGBoost, and the stacked model. The data are split into the training set and test set randomly with a random seed in each training iteration. The training data are trained on four models with the optimum parameter set. Test data are then used to calculate the performance matrix. After ten training iterations, average scores of precision, recall, and $f1$ score are calculated to analyze the performance of different models. Beyond the performance matrix, confusion matrix of each model is calculated over the two areas. We used the confusion matrix to present which lithology class is challenging to identify and which kind of lithology classes are difficult to be distinguished.

4. Results and Analysis

This section analyzes the classification results of the three classifiers as well as the stacked model. Initially, the grid search result of the best parameter value set for the four classifiers is presented. Next, the evaluation matrix that contains precision, recall, and $f1$ score is assessed over the models trained with the best hyperparameter in the AdaBoost classifier, GTB classifier, and XGBoost classifier. Evaluation is also provided in the stacked model that has soft voting over the three classifiers with the best parameter set. Finally, we develop confusion matrices in two areas by comparing the predicted labels from four classifiers with true labels from the test set, which presents the classification performance of each classifier on each lithology class.

Table 2: Tuned parameters for boosting models with search range and optimum value.

Boosting model	Tuned parameters	Search range	Optimum value
AdaBoost	Learning rate	0.1–0.9	0.4
	Number of iterations	50–300	200
GTB	Learning rate	$1e-5$–1	0.3
	The minimum number of samples obliged at a leaf node	5–20	20
	Maximum depth of the individual tree	5–20	20
	The number of boosting steps	100–200	200
	The minimum number of samples obliged to split an internal node	10–50	25
	Subsample	0.6–1	0.7
XGBoost	Learning rate	0–0.3	0.3
	Minimum loss reduction to split	0.1–0.5	0.2
	$L1$ regularization term on terms	$1e-5$–$1e-2$	$1e-4$
	The minimum number of samples obliged at a leaf node	5–50	20
	Maximum depth of individual tree	1–9	6
	Number of boosting steps	300–900	900
	Ratio of columns when constructing trees	50–100	60
	Subsample	0.4–1	0.7
	Minimum sum of instance weight	1–10	2

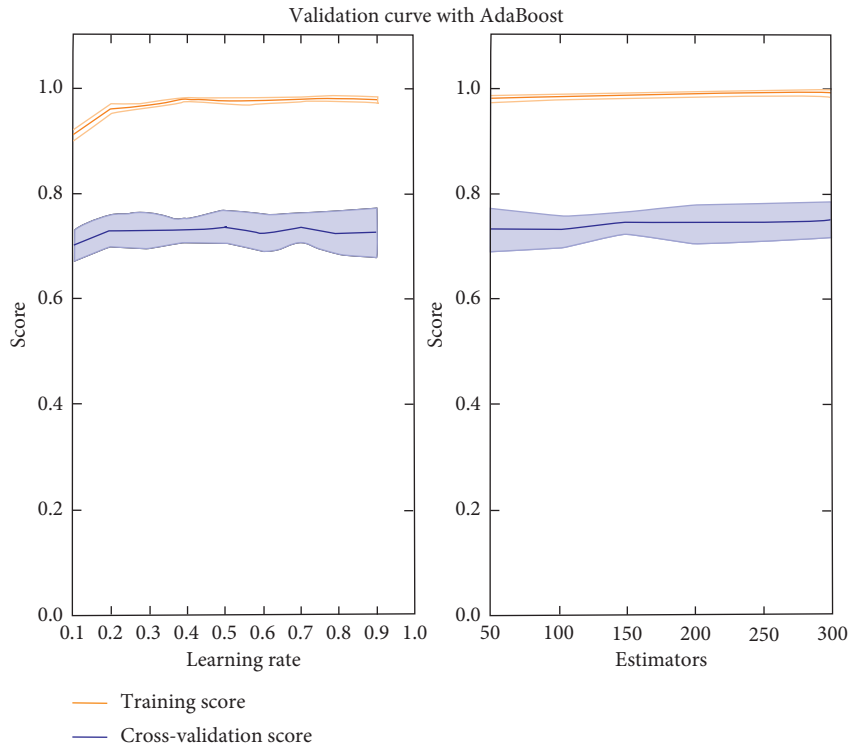

Figure 2: Validation curve of learning rate and boosting iterations for the AdaBoost classifier.

4.1. Tuning Process. Table 2 shows the optimum parameter set for the three classification models. Results show that the performance of three boosting models is profoundly affected by the learning rate that shrinks the contribution of each base model $h_m(x)$ within the boosting model. A small value of learning rate leads to underfitting as the penalty imposed on the base models is small, but if there is no shrinkage used in the boosting stages, a huge variance will occur. The number of boosting iterations is also a significant parameter for boosting approaches. Results show that 200–300 boosting iterations are enough to train the model with the specified boosting model. More boosting iterations will make the model better fit the training data at the expense of increased large variance. Regarding parameters related to the tree structure, smaller value of the depth of the tree models will cause underfitting for the model. Other parameters will not cause the model losing classification accuracy.

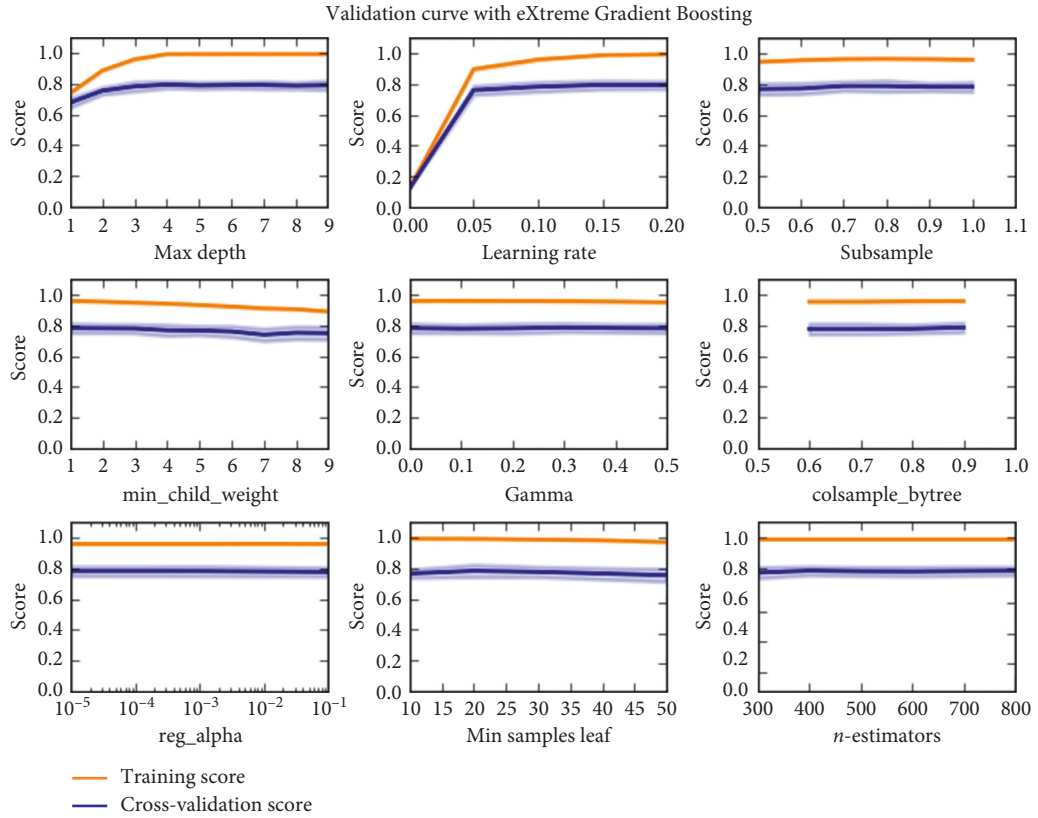

Figure 3: Validation curve of parameters for the XGBoost classifier.

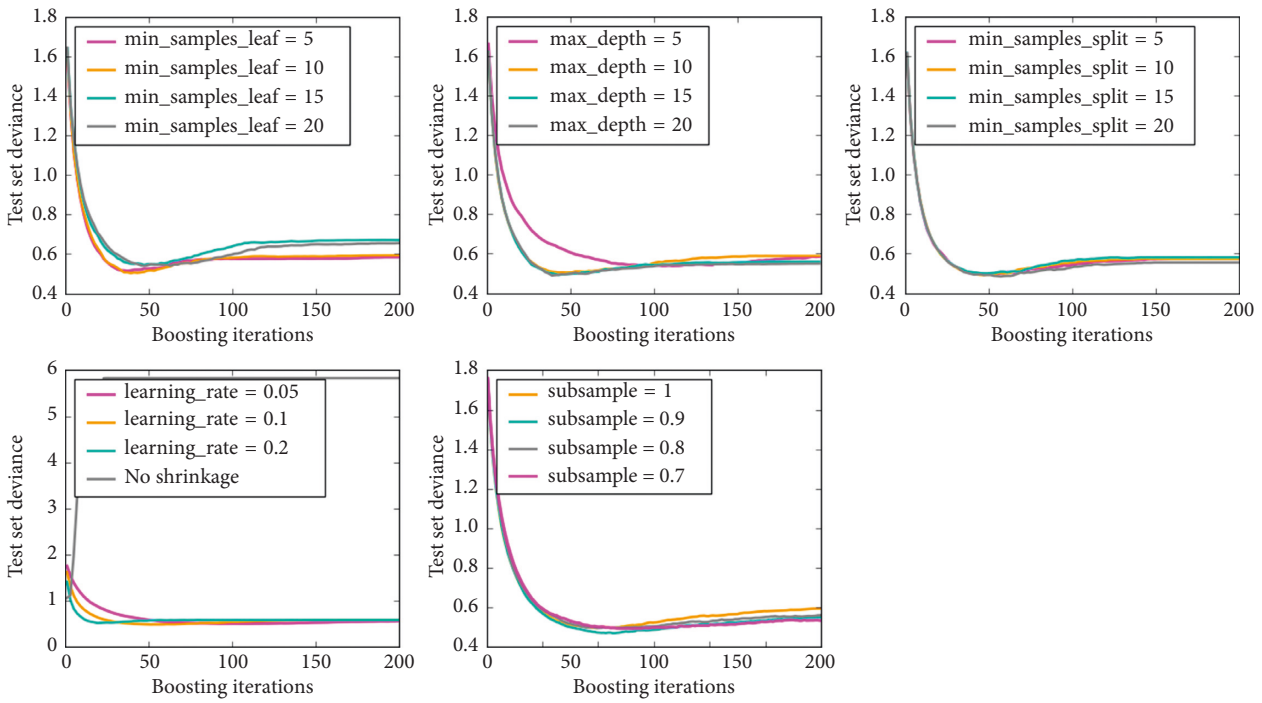

Figure 4: Test set deviance for the Gradient Tree Boosting classifier.

4.2. Accuracy Matrix. The performances of the four boosting models are evaluated over the training set with the 5-fold cross validation approach, which takes precision, recall, and $f1$ score as the evaluation matrix. Tables 3 and 4 show the evaluation matrix of each boosting model for each lithology class in the DGF and HGF, respectively. The results of the

Table 3: Performance matrix of 5-fold cross validation over the AdaBoost classifier, Gradient Tree Boosting classifier, XGBoost classifier, and stacked classifier in the Daniudi gas field.

	Precision	Recall	$f1$ score
AdaBoost			
C	0.996	0.935	0.964
CR	0.96	0.901	0.925
CS	0.629	0.585	0.603
FS	0.788	0.78	0.783
M	0.859	0.872	0.863
MS	0.779	0.835	0.805
PS	0.82	0.807	0.811
S	0.786	0.791	0.777
Avg	0.825	0.818	0.819
XGBoost			
C	0.991	0.938	0.963
CR	0.941	0.957	0.947
CS	0.592	0.629	0.601
FS	0.758	0.782	0.767
M	0.855	0.862	0.857
MS	0.788	0.759	0.771
PS	0.793	0.796	0.791
S	0.762	0.74	0.74
Avg	0.81	0.801	0.802
Gradient Tree Boosting			
C	0.969	0.961	0.964
CR	0.93	0.924	0.925
CS	0.614	0.605	0.602
FS	0.812	0.779	0.793
M	0.881	0.861	0.869
MS	0.781	0.815	0.793
PS	0.84	0.832	0.833
S	0.801	0.832	0.808
Avg	0.826	0.824	0.823
Stacking result			
C	0.99	0.962	0.974
CR	0.94	0.949	0.943
CS	0.64	0.668	0.648
FS	0.819	0.79	0.802
M	0.888	0.874	0.879
MS	0.801	0.809	0.803
PS	0.839	0.82	0.826
S	0.798	0.858	0.819
Avg	0.841	0.833	0.832

Table 4: Performance matrix of 5-fold cross validation over the AdaBoost classifier, Gradient Tree Boosting classifier, XGBoost classifier, and stacked classifier in the Hangjinqi gas field.

	Precision	Recall	$f1$ score
AdaBoost			
C	0.9	0.675	0.741
CS	0.836	0.81	0.821
FS	0.827	0.823	0.824
M	0.909	0.922	0.914
MS	0.858	0.817	0.835
PS	0.846	0.901	0.872
S	0.859	0.716	0.772
Avg	0.859	0.857	0.856
XGBoost			
C	0.825	0.725	0.759
CS	0.822	0.792	0.806
FS	0.796	0.816	0.802
M	0.899	0.919	0.907
MS	0.853	0.761	0.803
PS	0.839	0.886	0.862
S	0.819	0.805	0.805
Avg	0.846	0.845	0.842
Gradient Tree Boosting			
C	0.9	0.7	0.76
CS	0.832	0.793	0.812
FS	0.819	0.826	0.82
M	0.919	0.915	0.916
MS	0.854	0.823	0.836
PS	0.839	0.895	0.865
S	0.843	0.755	0.787
Avg	0.857	0.851	0.85
Stacking result			
C	0.9	0.75	0.801
CS	0.856	0.795	0.823
FS	0.826	0.835	0.827
M	0.919	0.924	0.919
MS	0.863	0.831	0.847
PS	0.847	0.904	0.874
S	0.901	0.818	0.852
Avg	0.868	0.867	0.864

performance matrix show that the prediction accuracy in the HGF area is higher than that in the DGF area. This is because that HGF has fewer classes to classifier, thus the prediction accuracy would be higher. Another important finding is that the optimized stacked boosting model achieves best classification accuracy compared with the other three individual boosting models in both areas. Tables 3 and 4 compare the average prediction accuracy of three boosting models and the stacked model. As shown in the tables, the $f1$ scores of optimized stacked boosting model are 83.2% and 86.4%, respectively, which are higher than all the individual boosting models. This proves that the performance of the stacked boosting model is better in accuracy and generalization. Moreover, Tables 3 and 4 present the abilities of four models to distinguish different lithology classes. Looking at the prediction accuracy for C and M, it is apparent that all boosting models have a higher classification accuracy for coal class and mudstone class compared to those of other classes. The accuracy scores indicate that there exist some identical lithology properties to identify coal and mudstone classes. Our previous research shows that the ensemble methods are applicable to identify lithology classes given the number of logging data is limited. The gradient tree boosting model performs better in classifying the four sandstone classes than the SVM classifier, neural network classifier, and random forest classifier [10]. In this case study, the average precision scores of GTB classifier on the sandstone classes in DGF and HGF are 76.1% and 83.%, respectively, while the average precision scores of the optimized stacked boosting model are 77.5% and 84.2%, respectively. These results show that the optimized boosting model has better performance in classifying sandstone classes than the GTB classifiers.

4.3. Confusion Matrix. Figures 5 and 6 show the confusion matrix that compares the class labels predicted by the four

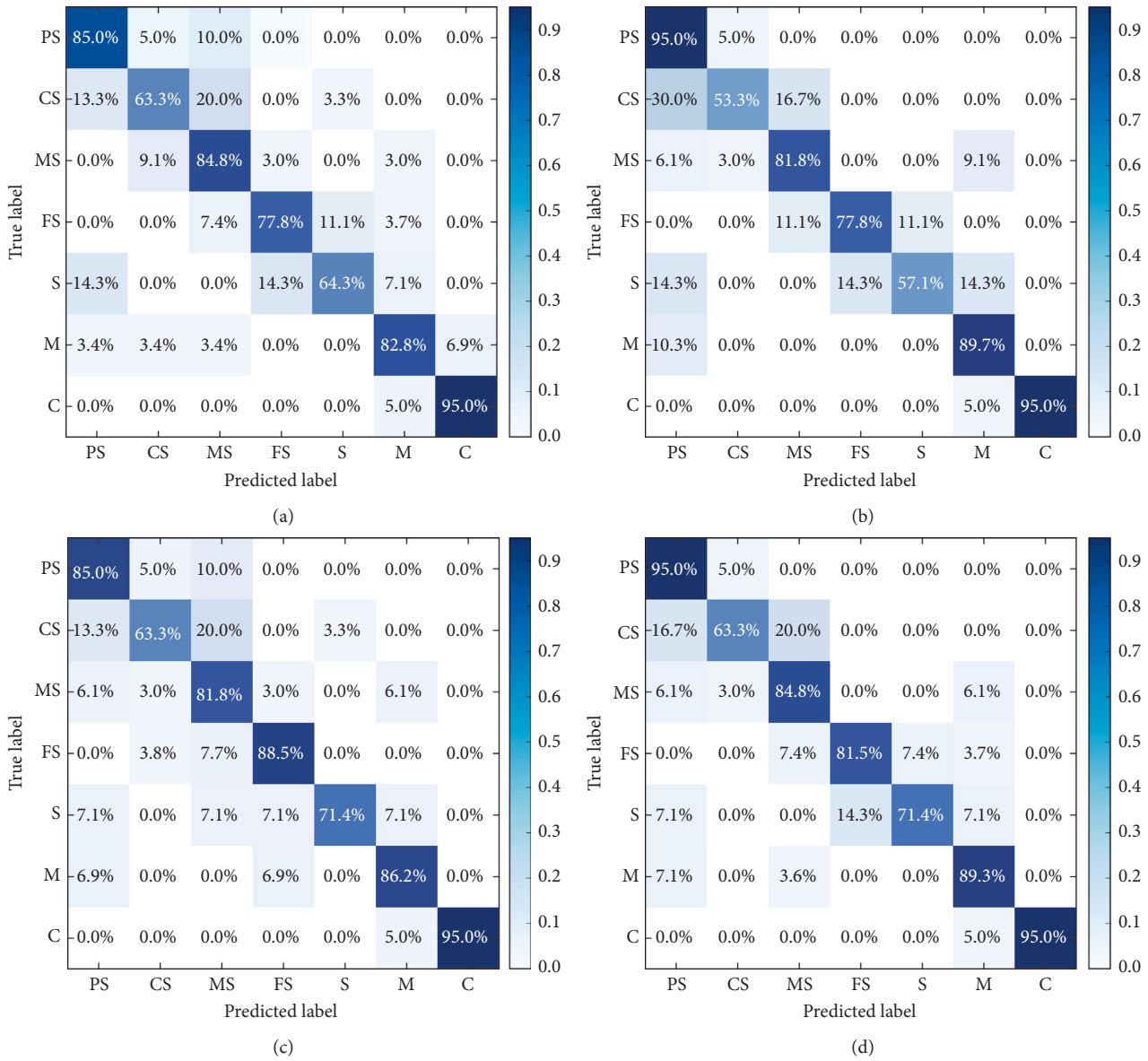

Figure 5: Confusion matrix of (a) AdaBoost model, (b) Gradient Tree Boosting model, (c) XGBoost model, and (d) stacked model on the DGF test dataset.

trained models and true labels on the DGF and HGF test dataset. To eliminate the effect of CR in the confusion matrix, we remove the CR classification result in the DGF area. The diagonal of the matrix presents the percentage of lithology classes that are correctly classified. The rest of the matrix shows the percentage of corresponding classes that are misclassified to other classes. It is evident in the matrix that only a small amount of C and M classes would be misclassified to other classes. In both areas, the classification accuracy for M and C classes is over 95.0% and 85.0%, respectively. However, the four sandstone classes are easy to be misclassified with each other. All the models have difficulty in distinguishing the sandstone classes. For example, with the GTB model in the DGF area, 30% of CS class would be misclassified to PS class and 16.7% of CS class would be misclassified to MS class. And this is the case for all the four models. It is likely that the grain diameters of sandstone are challenging to ascertain in some circumstances. And it is possible that human error is involved in the labelling work that labels the sandstone classes. However, the stacked model has the best classification accuracy in classifying sandstone classes. In the DGF area, the average classification accuracy for AdaBoost, GTB, and XGBoost are 77.7%, 77.22%, and 79.65%, respectively. However, the classification accuracy for the stacked model is 81.15%. The same result applies to the HGF area. The average classification accuracy for AdaBoost, GTB, and XGBoost is 80.1%, 81.75%, and 81.73%, respectively. And the classification accuracy for the stacked model is 82.73%. As shown in Figure 5, three individual models have different classification performances in different sandstone classes. The GTB model can classify the PS class with 95% of accuracy while the other models only

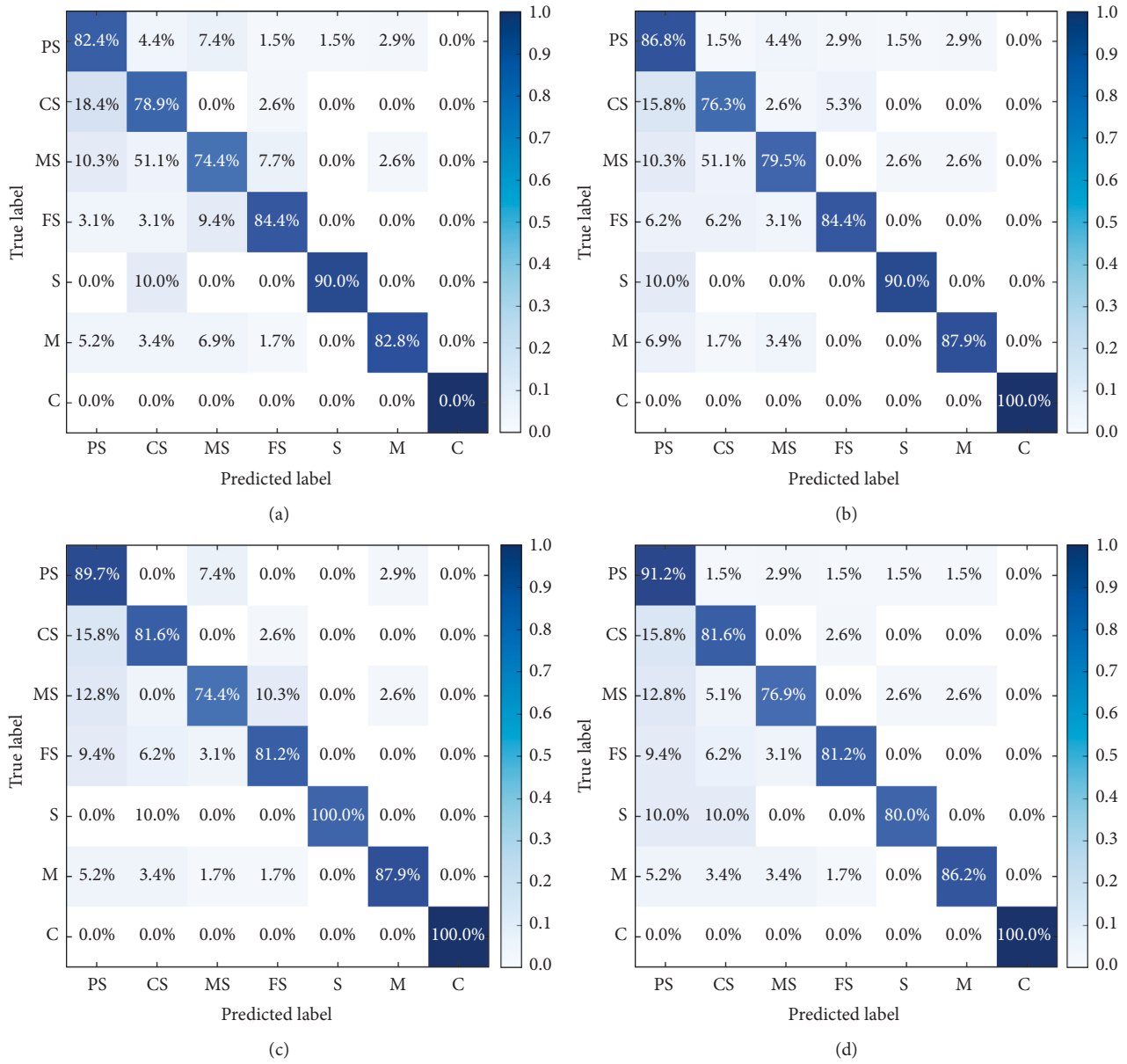

Figure 6: Confusion matrix of (a) AdaBoost model, (b) Gradient Tree Boosting model, (c) XGBoost model, and (d) stacked model on the HGF test dataset.

achieve 85%. Also, the AdaBoost model can classify the MS class with 84.8% while other models can only achieve 81.8%. The stacked model can balance the advantages of three individual models and construct the model with the best performance. It achieves the 95% of accuracy for PS class and 84.8% for MS class. The same rule applies to the HGF area. Overall, the stacked model has better classification accuracy in sandstone classes compared with the individual models.

5. Conclusion

In this study, well logging data acquired from several wells from two gas fields in the Ordos Basin were used to train the boosting models for the multiclass lithology classification problem. The performance of three individual boosting models and their stacked model was evaluated through classification accuracy, precision, recall, $f1$ score, and confusion matrix. During the training process, the grid search approach was adopted to optimize the parameter set for boosting models. We used the test deviance approach and validation curves to narrow down the search range of the parameter set. In order to prove the generalization of the result, the experiment was performed ten times in two different areas. Each time, all the data items were mixed and then split into training sets and test sets randomly, where training sets are used to train the boosting model and stacked boosting model with 5-fold cross validation and evaluation matrix. Testing sets are used to evaluate the performance of each model with the confusion matrix.

Results show that the stacked boosting model performs the best for lithology classification compared with other individual boosting models when the number of lithology characteristics is limited. The test deviance result also shows that 200–300 might be enough for the number of boosting iterations for GTB. A large number of boosting iterations will cause overfitting in AdaBoost and GTB. Moreover, the stacked boosting model could avoid overfitting by applying regularization on GTB and XGBoost. Our previous work shows that the difficulty for multiclass lithology classification is the sandstone classification. The confusion matrix shows that the optimized stacked boosting model has better performance in distinguishing sandstone classes, while the GTB and AdaBoost classifiers have relatively lower classification accuracy. The optimized stacked boosting model has a relatively high classification accuracy, which could assist in lithology identification and improve work efficiency. However, as the ensemble method requires the trained boosting models for the soft voting, it is relatively slow to construct. As the boosting approach trains the final model step by step, more work can be done to develop a pretrained model as the base model for the multiclass lithology classification problem. Also, more data with more features could be collected to improve the classification accuracy for sandstone classes.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (grant no. 61772090) and China Scholarship Council (CSC) (grant CSC no. 201708060147). The authors would like to thank the Huabei branch of Sinopec Group for providing the resources used in this study.

References

[1] O. Serra and H. T. Abbott, "The contribution of logging data to sedimentology and stratigraphy," *Society of Petroleum Engineers Journal*, vol. 22, no. 1, pp. 117–131, 1982.

[2] F. A. Anifowose, "Ensemble machine learning: the latest development in computational intelligence for petroleum reservoir characterization," in *Proceedings of the SPE Saudi Arabia Section Technical Symposium and Exhibition*, Society of Petroleum Engineers, Al-Khobar, Saudi Arabia, May 2013.

[3] S. J. Rogers, J. H. Fang, C. L. Karr, and D. A. Stanley, "Determination of lithology from well logs using a neural network (1)," *AAPG Bulletin*, vol. 76, no. 5, pp. 731–739, 1992.

[4] B. Rafik and B. Kamel, "Prediction of permeability and porosity from well log data using the nonparametric regression with multivariate analysis and neural network, Hassi R'Mel Field, Algeria," *Egyptian Journal of Petroleum*, vol. 26, no. 3, pp. 763–778, 2017.

[5] A. Al-Anazi and I. D. Gates, "On the capability of support vector machines to classify lithology from well logs," *Natural Resources Research*, vol. 19, no. 2, pp. 125–139, 2010.

[6] M. A. Sebtosheikh and A. Salehi, "Lithology prediction by support vector classifiers using inverted seismic attributes data and petrophysical logs as a new approach and investigation of training data set size effect on its performance in a heterogeneous carbonate reservoir," *Journal of Petroleum Science and Engineering*, vol. 134, pp. 143–149, 2015.

[7] M. J. Cracknell and A. M. Reading, "Machine learning for lithology classification and uncertainty mapping," in *Proceedings of the AGU Fall Meeting Abstracts*, San Francisco, CA, USA, December 2012.

[8] Y. Ao, H. Li, L. Zhu, S. Ali, and Z. Yang, "Identifying channel sand-body from multiple seismic attributes with an improved random forest algorithm," *Journal of Petroleum Science and Engineering*, vol. 173, pp. 781–792, 2019.

[9] P. Bestagini, V. Lipari, and S. Tubaro, "A machine learning approach to facies classification using well logs," in *Proceedings of the SEG Technical Program Expanded Abstracts*, pp. 2137–2142, Society of Exploration Geophysicists, Houston, TX, USA, September 2017.

[10] Y. Xie, C. Zhu, W. Zhou, Z. Li, X. Liu, and M. Tu, "Evaluation of machine learning methods for formation lithology identification: a comparison of tuning processes and model performances," *Journal of Petroleum Science and Engineering*, vol. 160, pp. 182–193, 2018.

[11] M. J. Pazzani and D. Billsus, "Content-based recommendation systems," in *The Adaptive Web*, pp. 325–341, Springer Verlag, Berlin, Germany, 2007.

[12] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 6645–6649, IEEE, Vancouver, BC, Canada, May 2013.

[13] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, pp. 2493–2537, 2011.

[14] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*, MIT press, London, England, 2012.

[15] R. E. Schapire, "The boosting approach to machine learning: an overview," in *Nonlinear Estimation and Classification*, pp. 149–171, Springer, New York, NY, USA, 2003.

[16] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[17] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, New York, NY, USA, 2006.

[18] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[19] T. Hastie, S. Rosset, J. Zhu, and H. Zou, "Multi-class adaboost," *Statistics and Its Interface*, vol. 2, no. 3, pp. 349–360, 2009.

[20] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *The Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.

[21] J. H. Friedman, "Stochastic gradient boosting," *Computational Statistics & Data Analysis*, vol. 38, no. 4, pp. 367–378, 2002.

[22] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 21, no. 3, pp. 660–674, 1991.

[23] T. Chen and C. Guestrin, "Xgboost: a scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, ACM, San Francisco, USA, August 2016.

[24] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[25] A. E. Hoerl and R. W. Kennard, "Ridge regression: biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

[26] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)," *The Annals of Statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[27] T. G. Dietterich, "Ensemble methods in machine learning," in *Proceedings of the International Workshop on Multiple Classifier Systems*, pp. 1–15, Springer, Berlin, Germany, June 2000.

[28] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 10, pp. 993–1001, 1990.

[29] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proceediongs of the International Joint Conference on Artificial Intelligence (IJCAI)*, vol. 14, pp. 1137–1145, Montreal, Canada, August 1995.