Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 7842847, 13 pages
https://doi.org/10.1155/2020/7842847

Research Article

An Empirical Study for Adopting Machine Learning Approaches for Gas Pipeline Flow Prediction

Guoliang Shen,[1] Mufan Li,[2] Jiale Lin,[3] Jie Bao,[4] and Tao He[5]

[1]*Zhejiang Zhenergy Natural Gas Operation Co., Ltd., Hangzhou 310052, China*
[2]*East China Normal University, Shanghai 200062, China*
[3]*Shanghai Jiao Tong University, Shanghai 200240, China*
[4]*Shanghai Weidi Information Technology Co., Ltd., Shanghai 200050, China*
[5]*COWAROBOT Co., Ltd., and Anhui Province Key Laboratory of Multimodal Cognitive Computation, Shanghai, Anhui 230601, China*

Correspondence should be addressed to Tao He; tommie.he@cowarobot.com

Received 10 July 2020; Accepted 22 July 2020; Published 8 September 2020

Guest Editor: Jiayi Ma

As industrial control technology continues to develop, modern industrial control is undergoing a transformation from manual control to automatic control. In this paper, we show how to evaluate and build machine learning models to predict the flow rate of the gas pipeline accurately. Compared with traditional practice by experts or rules, machine learning models rely little on the expertise of special fields and extensive physical mechanism analysis. Specifically, we devised a method that can automate the process of choosing suitable machine learning algorithms and their hyperparameters by automatically testing different machine learning algorithms on given data. Our proposed methods are used in choosing the appropriate learning algorithm and hyperparameters to build the model of the flow rate of the gas pipeline. Based on this, the model can be further used for control of the gas pipeline system. The experiments conducted on real industrial data show the feasibility of building accurate models with machine learning algorithms. The merits of our approach include (1) little dependence on the expertise of special fields and domain knowledge-based analysis; (2) easy to implement than physical models; (3) more robust to environment changes; (4) requiring much fewer computation resources when it is compared with physical models that call for complex equation solving. Moreover, our experiments also show that some simple yet powerful learning algorithms may outperform industrial control problems than those complex algorithms.

1. Introduction

Machine learning has been playing an increasingly important role in industrial control. In particular, an accurate model used for estimating the state of the complex industry system is essential for automatic control. As shown in Figure 1, the flow rate model is a key part of the comprehensive analysis and control system of natural gas pipelines. Traditionally, industrial models are often built on physical mechanism analysis and industrial expertise, which are called physical models in this paper. Nevertheless, it is costly to build physical models that are based on extensive theoretical and experimental analysis. Some physical models require massive computational resources to calculate the results. In our problem, to build an accurate model to calculate the flow rate of gas pipelines, knowledge about hydromechanics is required. Moreover, calculating the flow rate of gas pipelines needs to solve many complicated flow equations, making it very difficult to control the pipeline system in real time. Adding more relevant factors into a physical model means much more analysis work, such as the shape of the pipeline in our problem. Therefore, some physical models omit some relevant factors to keep the model simple. As a result, they are not so accurate or robust to environmental changes. Building statistical models based on machine learning algorithms requires much less expertise in special fields, and one can automate the modeling process by computer. In particular, one can take more relevant

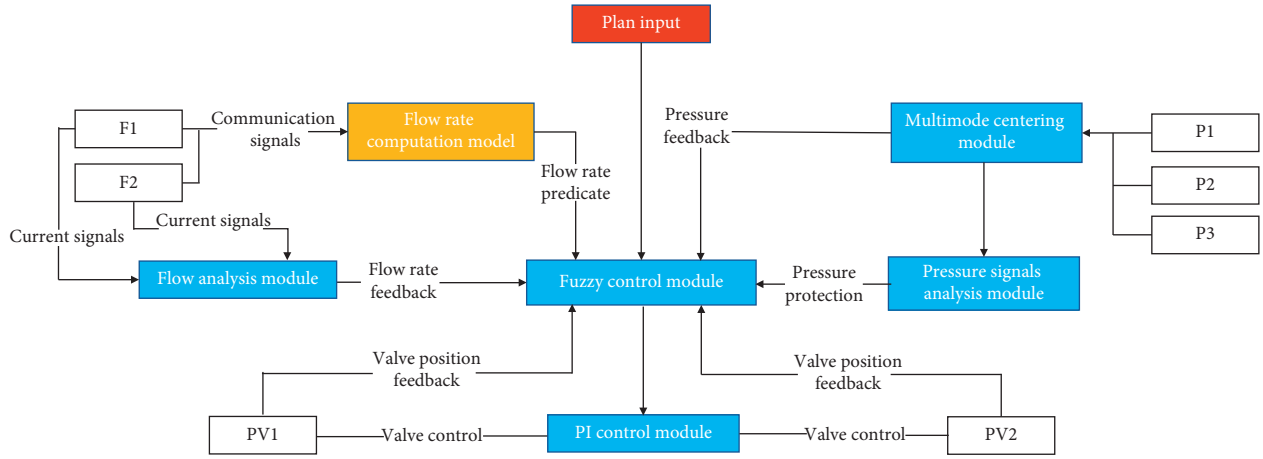


Figure 1: The structure of comprehensive analysis and control system of natural gas pipelines. The yellow label represents the flow rate computation model. The predictions of this module are fed to the fuzzy control module, and the fuzzy control module uses this information and other signals to control the natural gas pipeline system.

factors into consideration to build models that are more accurate and more adaptive to environmental changes.

There are plenty of existing machine learning algorithms, and choosing a suitable algorithm is essential to build an accurate and robust model. Network architecture search (NAS) [1, 2] can automatically design the neural network architecture and choose the proper hyperparameters, but NAS methods are restricted in the neural network algorithm. Although methods based on neural network and deep learning have shown excellent performance on some complex problems such as image recognition and game-playing [3, 4], these methods often cannot achieve satisfactory performance in some relatively simple problems. Our method is not restricted to neural networks so that one can consider the learning algorithms in a much wider spectrum.

Machine learning methods are widely used in industrial control problems. Work [5] proposed an SVM-based method to predict the draining time of oil pipeline. The GBDT algorithm is used to diagnose the gas sensor faults and predict the power load of the grid. Artificial neural network technology is also a popular method in industry control [6]. Unsupervised learning is often combined with supervised learning to produce a better performance [7, 8], and Yang et al. proposed a novel [9] unsupervised learning method based on the dual autoencoder network. The carefully selected training set can filter out anomalies, and the feature selection procedure can remove noisy features. Applying these two preprocessing methods can result in a more stable and powerful model. Papers [10, 11] used feature selection methods in industry control. Paper [12] introduced a technique that can perform training sample selection and feature selection simultaneously.

2. Materials and Methods

This paper is focused on an empirical study on applying popular machine learning methods to the flow prediction problem, which is seldom addressed by data-driven learning models in the literature. Note that in this application-oriented paper, we have not devised methodologically particularly new approach, instead we resort to a comprehensive study on the performance of the existing methods. While it is still worthy that we have adopted the GBDT stacking model and compare it with the baseline GBDT, as shown in Figure 2.

Specifically, linear regression, neural network, random forest, support vector machine, and K nearest neighbors are evaluated in this work. To select the proper algorithm and hyperparameters to model the flow rate of the gas pipeline, we split the flow rate data into training set and testing set. We use the training set to train different machine learning models and report their performance on the testing set.

2.1. Neural Network. As shown in Figure 3, we designed a neural network [13] models with several layers, each layer contains a linear transform operation and a batch normalization [14] operation and passes the output through an activation function. We test several different configurations to find a good neural network architecture.

2.1.1. Training of Neural Network Models. We split the training set to several minibatches to reduce the requirement of memory. And, we adopted Adam [15] as the optimizer, set the initial learning rate to 0.003, and reduce the learning rate by 0.5 every 5000 steps. We trained each model for a total of 500 epochs.

2.2. Gradient Boosting Decision Tree. Random forest [16] models are those composed of many decision trees. In the decision tree algorithms, an algorithm determines which child node to go base on input attributes, and repeat this step until it reaches a leaf node and outputs the value stored in the leaf node as the prediction. However, the capability of a single decision tree is limited, and it is difficult for a single decision tree to capture complex relationships between features and labels. To solve this problem, we can compose

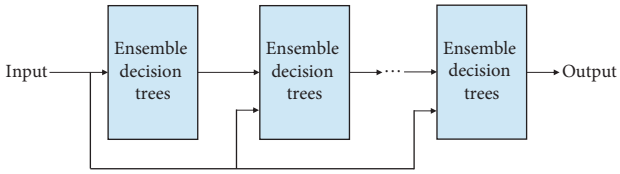


Figure 2: Stacking of ensemble decision trees.

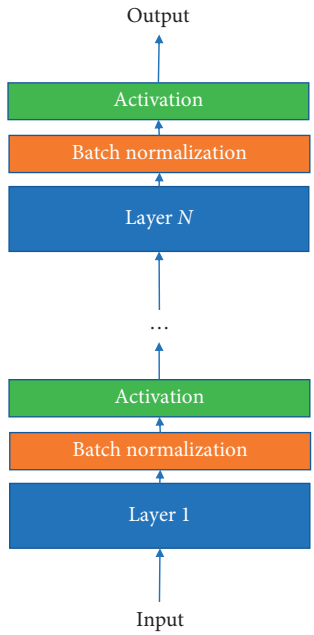


Figure 3: Structure of neural network.

several decision trees to learn complex relationships. These class algorithms are called random forest algorithms. Gradient Boosting Decision Tree (GBDT) [17, 18] is one kind of random forest algorithms. In GBDT algorithms, each new decision tree learns the residual error of all previous decision trees. When adding the M^{th} decision tree, we want to fit the parameters θ_m of this tree to satisfy the following condition:

$$\theta_m = \arg\min_{\theta_m} \sum_{i=1}^{N} L\left(y_i, f_m(x_i, \theta_m) + \sum_{k=1}^{m-1} f_k(x_i)\right), \quad (1)$$

where $f_m(x_i; \theta_m)$ is a function corresponding to the decision tree with parameters θ_m and $f_k(x_i)$ is the function determined by first k decision trees. $L(y_i, \hat{y}_i)$ is the loss function that is used to measure the performance of the model. The output of the model composed by M decision trees is

$$y = \sum_{k=1}^{M} f_k(x_i). \quad (2)$$

2.2.1. Stacking of Gradient Boosting Decision Trees. Although gradient boosting decision tree works well in many applications, it is not suitable for many other applications, such as image classification and speech recognition. This work points out that these drawbacks are because these ensemble decision trees are shallow models and cannot perform representation learning. They try to mitigate this problem by stacking several layers of ensemble decision trees. As illustrated in Figure 2, the first several layers work as feature transformers; instead of aggregating the results of each decision tree, the result of each decision tree is fed to the next layer as features.

2.3. Support Vector Regression. Support vector regression [19] is a regression developed from the support vector machine algorithm [20]. In support vector machine algorithms, we want to maximize the minimum distance of each data point from the hyperplane. But, in support vector regression algorithms, we want to minimize the maximum distance of each data point from the hyperplane. Figure 4 illustrates the difference and relationship between SVM and SVR algorithms.

2.4. K-Nearest Neighbor Regression. K-nearest neighbor regression [21] is a regression algorithm that predicts the result based on K neighbors' ground truths that are closest to the given data point. We can take the average of K nearest neighbors' ground truths as prediction or we can weigh the K nearest neighbors' ground truth by the distance between the data point and the neighbour.

2.5. Performance Measure. We use mean square error (MSE), coefficient of determination (R^2), and mean relative error (MRE) to measure the performance of our machine learning models.

The definition of mean squared error can be described by the following equation:

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2, \quad (3)$$

where N is the size of the dataset, y_i is the ground truth of i^{th} data item, and $\hat{y}_i$ is prediction given by the machine learning model [22].

The following equation describes the definition of the coefficient of determination (R^2):

$$R^2 = \frac{\text{var}(y) - (\hat{y}_i - y)^2}{\text{var}(y)}, \quad (4)$$

where $\text{var}(y)$ is the variance of labels and $\text{var}(\hat{y} - y)$ is the sum of the square of the error between predictions and labels. R^2 describes how much variance can be explained by the model. The definition of mean relative error (MRE) can be described by the following equation:

$$\text{MRE} = \frac{1}{N} \sum_{i=1}^{N} \left| \frac{\hat{y}_i - y_i}{y_i} \right|. \quad (5)$$

2.6. Input Data and Label. We obtained the data from the sensors deployed in our monitoring systems. The dataset contains the following data:

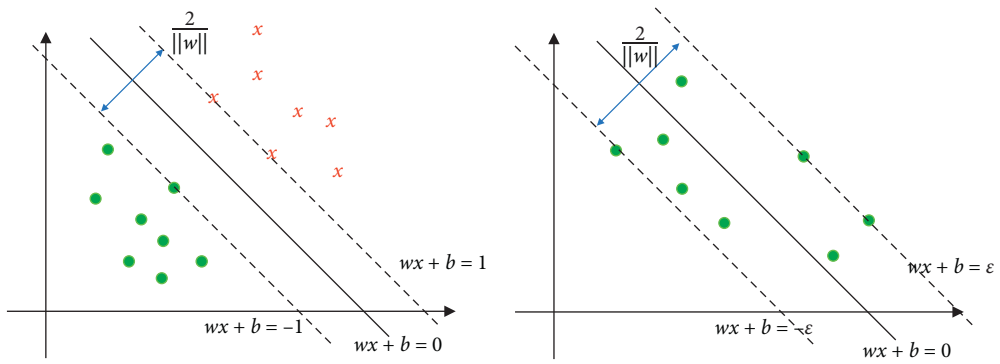


Figure 4: The difference and relationship between SVM and SVR.

(1) Generation Time: time the data being generated
(2) F_W: working flow rate
(3) F_S: standard flow rate
(4) PV: adjusting valve
(5) PT: compensatory pressure
(6) TE: compensatory temperature
(7) FT: flowmeter

The input features are F_W, PV, PT, TE, and FT, and the label is F_S. The values of standard flow rate (F_S) range from 1812.6 to 3172.5. The distribution of standard flow rate is given in Figure 5; from the figure, we can see that most standard flow rate values are around 30000, but there are also some values distributed around 15000.

2.7. Training Set and Testing Set Splitting. The dataset contains 109741 items in total, and we split the dataset to a training set which has 103741 items and a testing set with 6000 items.

2.8. Data Normalization. We subtract mean value from the dataset and divide the result by standard deviation to generate the normalized dataset. We compute the mean value and the standard deviation on the training set. The data normalization process can be described in the following equation:

$$\mu = \text{mean value of traning set},$$

$$\sigma = \text{standard deviation of training set},$$

$$x_i = \frac{x_i - \mu}{\sigma}, \quad \text{for each } x_i \text{ in traning set and testing set.} \tag{6}$$

2.9. Benchmark. We take the mean value of labels and the linear regression model as the benchmark and compare it with other models.

3. Result

3.1. Linear Models. Among all tested methods, linear models are the simplest models and have fewer parameters than other models. Models with fewer parameters are less prone to overfitting but may be incapable of modeling complicated relationships between input and labels. We tested several different linear models with parameter regularization. Lasso regression [23] is the linear regression with L1 regularization on parameters, and ridge regression [24] is the linear regression with L2 regularization on parameters. The accuracies of linear models are worse than other methods except SVR, but linear models have the merit of minimum computation resource requirement. The experiments with different regularization strength suggest that the linear models are simple enough, and adding additional regularization hurts the performance of linear models in this problem. The results are shown in Figure 6 and Table 1.

3.2. GBDT. We chose mean squared error as the loss function and tested several different learning rates and maximum numbers of leaves in each decision tree. Moreover, we tested the GBDT models with several different maximum numbers of leaf nodes in each decision tree, and the results are given in Figure 7 and Table 2. The more the leaves in each decision tree, the more sophisticated functions between input and output can be learnt by GBDT models. In this application, we found that GBDT models with more leaves yields a better accuracy, but the improvement is negligible when then leaves number larger than 10000.

We also tested how different learning rates influence the performance of GBDT models. The learning rate controls how many residual errors to be eliminated when adding a new decision tree to the GBDT model. The results given in Figure 8 and Table 3 suggest that setting the learning rate too low or too high will hurt the accuracy of GBDT models.

3.2.1. Stacking of GBDT. We also tested the performance of stacked GBDT models with different learning rates. The results given in Figure 9 and Table 4 do not show an improvement when compared with normal GBDT models as

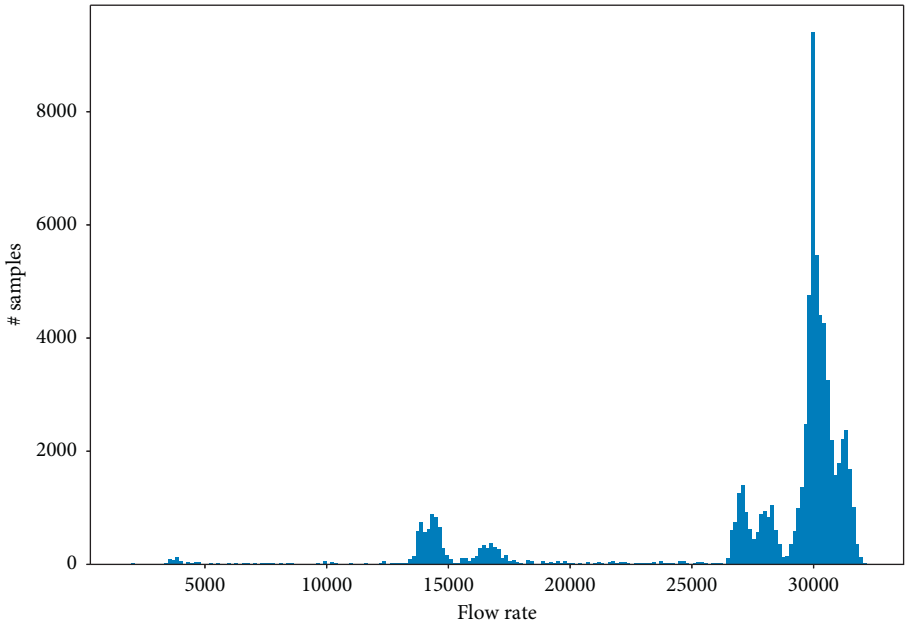


Figure 5: The distribution of standard flow rates.

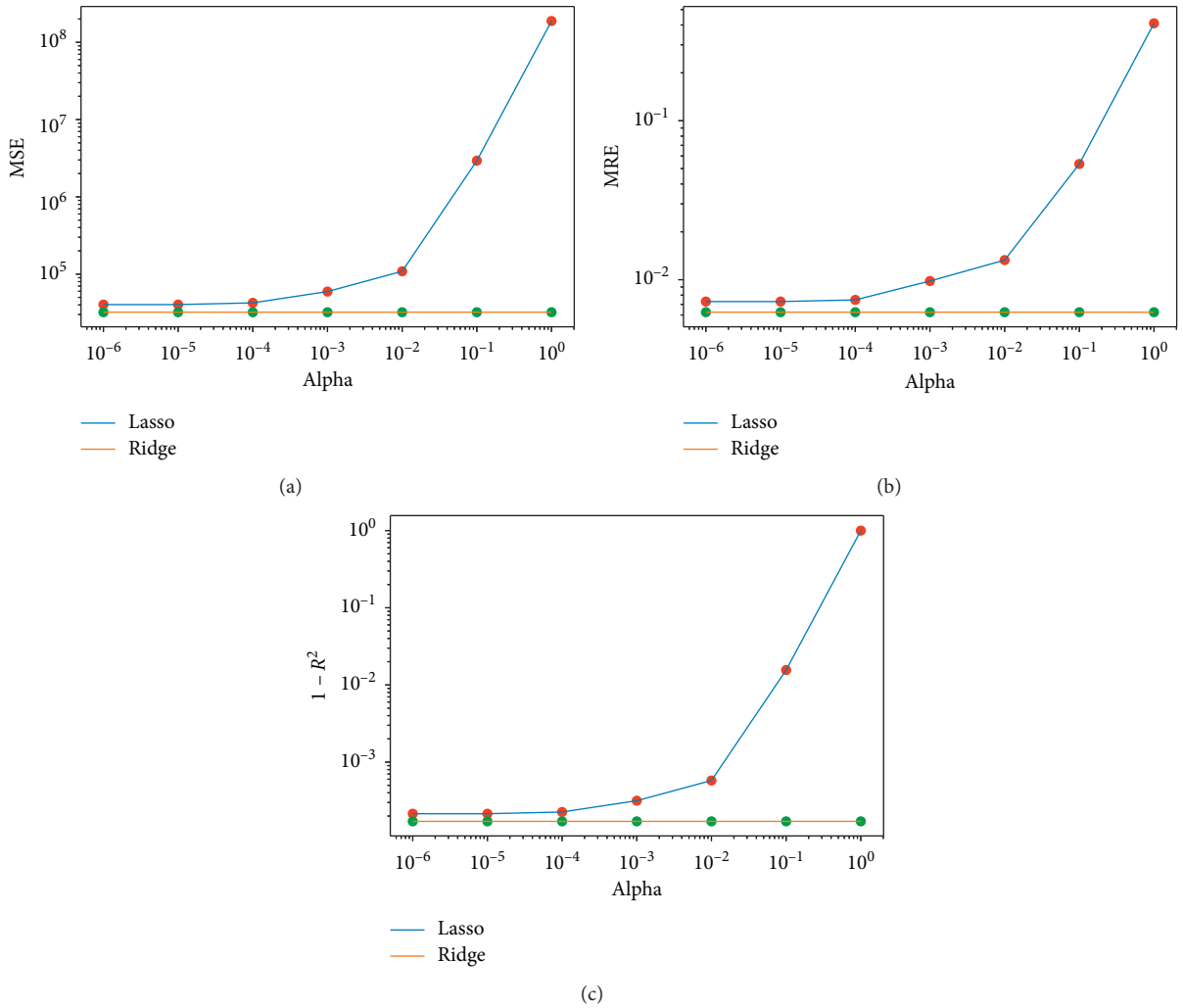


Figure 6: Relation between penalty and performance. Alpha is the strength of regularization. The regularization (a) MSE, (b) MRE, and (c) $1 - R^2$.

Table 1: Sensitivity tests using different regularization methods and penalty strength.

	Alpha	1×10^{-6}	1×10^{-5}	0.0001	0.001	0.01	0.1	1.0
Lasso	MSE	4×10^{4}	4×10^{4}	4.2×10^{4}	6×10^{4}	1.1×10^{5}	2.9×10^{6}	1.9×10^{8}
	MRE	7.2×10^{-3}	7.2×10^{-3}	7.4×10^{-3}	9.8×10^{-3}	1.3×10^{-2}	5.3×10^{-2}	4.0×10^{-1}
	$1-R^2$	2.1×10^{-4}	2.1×10^{-4}	2.2×10^{-4}	3.1×10^{-4}	5.7×10^{-4}	1.5×10^{-2}	9.9×10^{-1}
Ridge	MSE	3.2×10^{4}	3.2×10^{4}	3.2×10^{4}	3.2×10^{4}	3.2×10^{4}	3.2×10^{4}	3.2×10^{4}
	MRE	6.2×10^{-3}	6.2×10^{-3}	6.2×10^{-3}	6.2×10^{-3}	6.2×10^{-3}	6.2×10^{-3}	6.2×10^{-3}
	$1-R^2$	1.7×10^{-4}	1.7×10^{-4}	1.7×10^{-4}	1.7×10^{-4}	1.7×10^{-4}	1.7×10^{-4}	1.7×10^{-4}

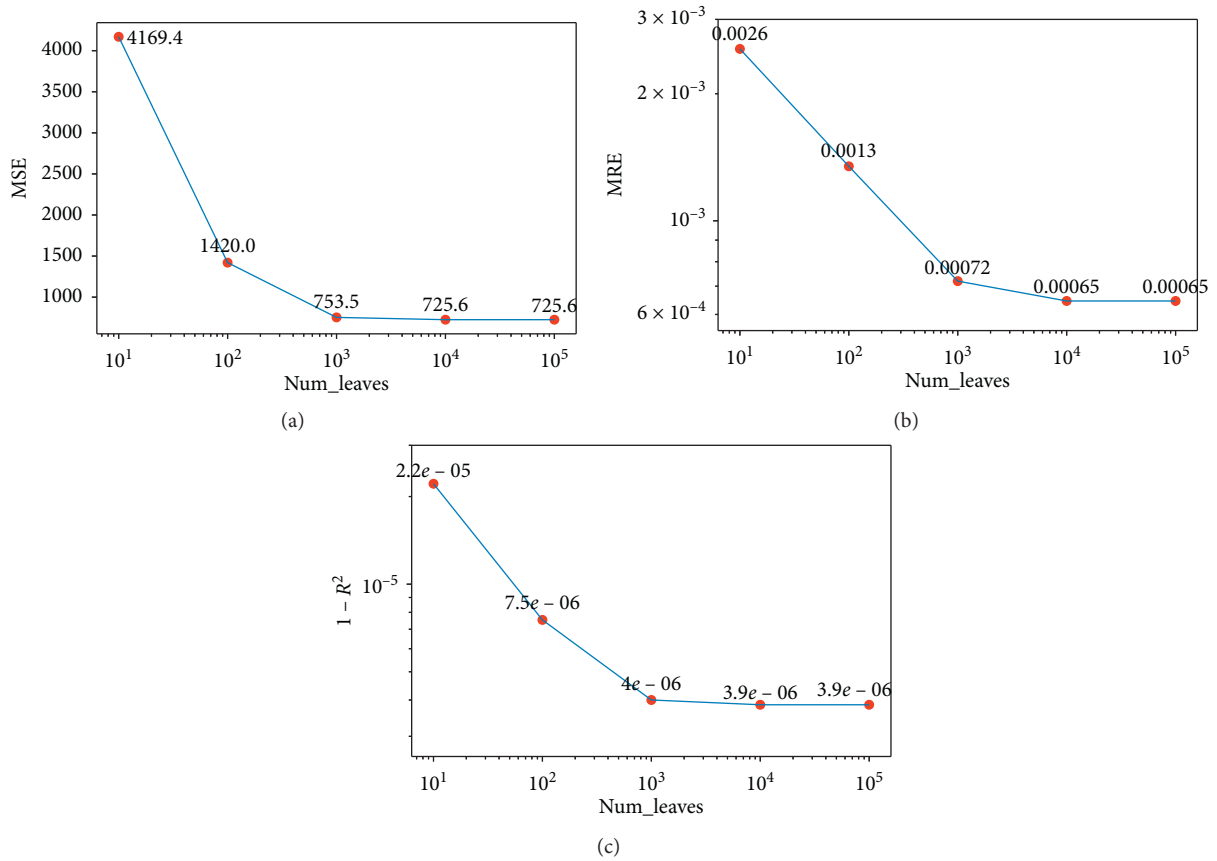


Figure 7: Sensitivity between performance measure and the maximum number of leaves in each decision tree. As shown in these figures, we can get performance gain as the maximum number of leaves increase, but the performance gain is negligible when the maximum number of leaves is greater than 10000. (a) MSE, (b) MRE, (c) $1-R^2$.

Table 2: Sensitivity tests given different maximum number of leaves in each decision tree.

Num leaves	10	100	1000	10000	100000
MSE	4.2×10^{3}	1.4×10^{3}	7.5×10^{2}	7.3×10^{2}	7.3×10^{2}
MRE	0.00255392	0.00134632	0.000719617	0.000645802	0.000645802
$1-R^2$	2.2×10^{-5}	7.5×10^{-6}	3.9×10^{-6}	3.8×10^{-6}	3.8×10^{-6}

we expected. This may be the result of overfitting of stacked GBDT models which have much more parameters than normal GBDT models.

3.3. KNN. We conducted a series of experiments to study the influence of the number of selected neighbors and the different averaging methods when calculating the prediction based on the nearest neighbors' ground truth. The results are shown in Figure 10 and Table 5. From the result, we can find that the accuracy is getting worse as the number of neighbors increases when simply averaging the labels of each selected neighbor. We can get rid of this problem when using distance between selected neighbors and input as weight of each label of selected neighbors.

3.4. SVR. We tested three kernel functions to find out which one is most suitable when applying the SVR algorithm to this

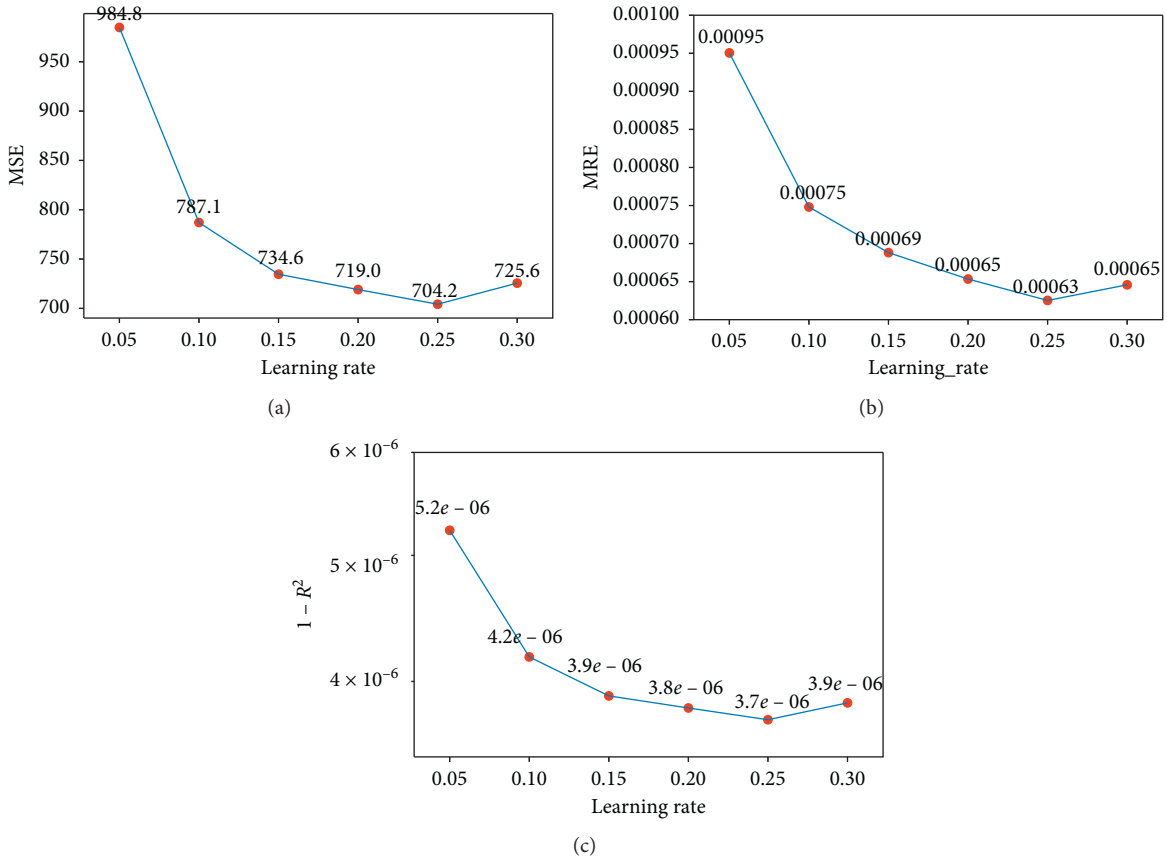


Figure 8: Relation between learning rate and performance. The performance becomes better as the learning rate increase at first, but the performance gets worse after the learning rate greater than 0.25. This suggests that 0.25 is a proper learning rate. (a) MSE, (b) MRE, (c) $1 - R^2$.

Table 3: The details of each performance measure of GBDT models when using the different learning rates.

Learning rate	0.05	0.1	0.15	0.2	0.25
MSE	9.8×10^2	7.9×10^2	7.3×10^2	7.2×10^2	7×10^2
MRE	0.000950502	0.000748058	0.000688089	0.000653405	0.000625291
$1 - R^2$	5.227×10^{-6}	$4.178 \times 10^{-6}0{-}6$	3.899×10^{-6}	3.817×10^{-6}	3.738×10^{-6}

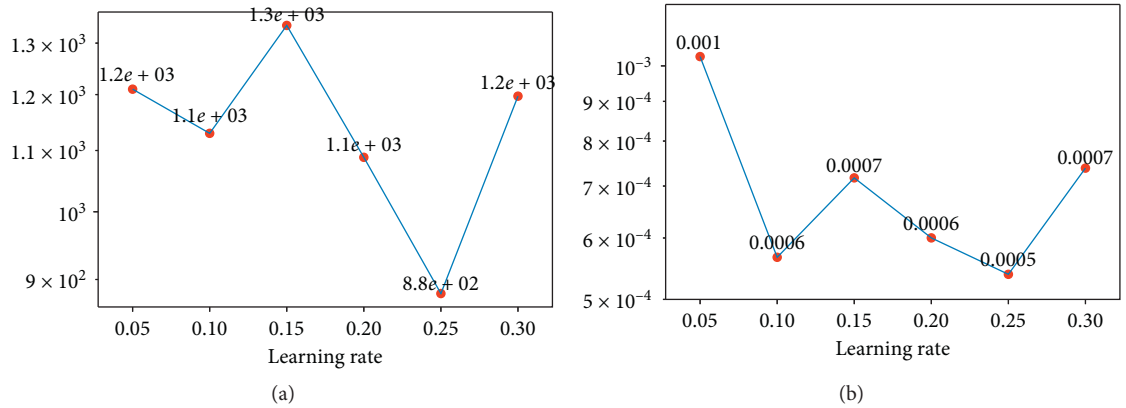
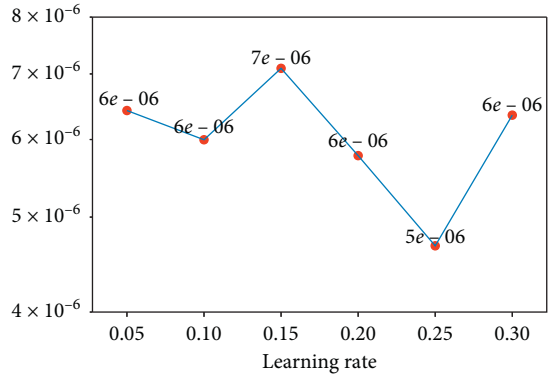


Figure 9: Continued.

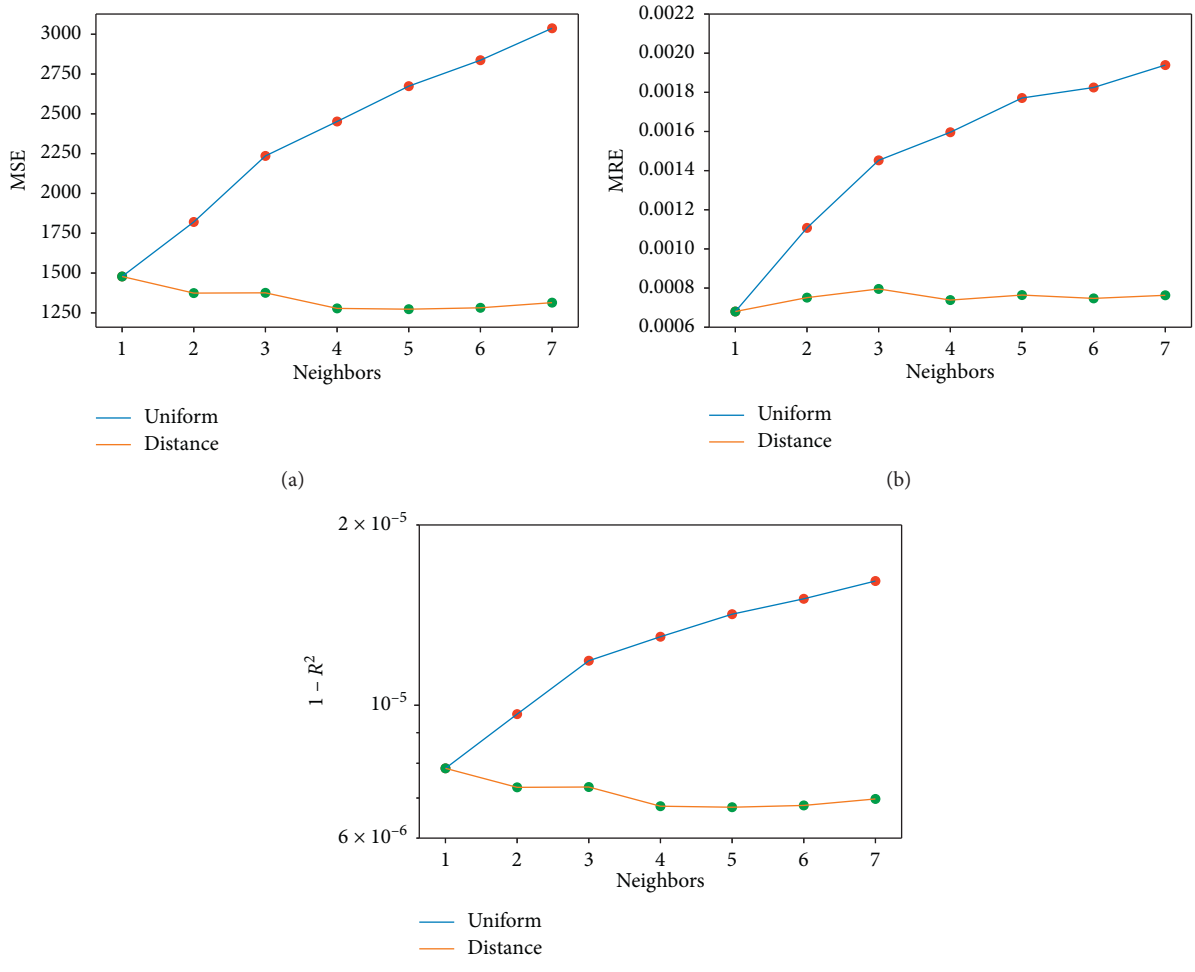


(c)

Figure 9: Relation between learning rate and performance of stacked GBDT models. (a) MSE, (b) MRE, (c) $1 - R^2$.

Table 4: The details of each performance measure of stacked GBDT models when using the different learning rates.

Learning rate	0.05	0.1	0.15	0.2	0.25
MSE	1.21×10^3	1.13×10^3	1.34×10^3	1.09×10^3	8.8×10^2
MRE	0.001027	0.000566	0.000717	0.000600	0.000538
$1 - R^2$	6.423×10^{-6}	5.996×10^{-6}	7.093×10^{-6}	5.777×10^{-6}	4.674×10^{-6}



(a)

(b)

(c)

Figure 10: Performance variation gives different neighbor numbers and averaging methods. The results suggest that, when taking the distance between the given data point and neighbors as weight, the performance gets better as the number of neighbors increases. However, the performance gets worse as the number of neighbors increase when using equal weights for each neighbor. These results suggest that weight based on distance is a better choice for this problem. (a) MSE, (b) MRE, (c) $1 - R^2$.

Table 5: Performance changes under different neighbors numbers and averaging methods.

Weighed	Neighbors	7	6	5	4	3	2	1
Uniform	MSE	3.00×10^{3}	2.80×10^{3}	2.70×10^{3}	2.50×10^{3}	2.20×10^{3}	1.80×10^{3}	1.50×10^{3}
	MRE	1.93×10^{-3}	1.82×10^{-3}	1.77×10^{-3}	1.59×10^{-3}	1.45×10^{-3}	1.10×10^{-3}	6.80×10^{-4}
	$1-R^2$	1.61×10^{-5}	1.50×10^{-5}	1.41×10^{-5}	1.30×10^{-5}	1.18×10^{-5}	9.66×10^{-6}	7.84×10^{-6}
Distance	MSE	1.30×10^{3}	1.30×10^{3}	1.30×10^{3}	1.30×10^{3}	1.4×10^{3}	1.40×10^{3}	1.50×10^{3}
	MRE	7.63×10^{-4}	7.47×10^{-4}	7.64×10^{-4}	7.38×10^{-4}	7.95×10^{-4}	7.51×10^{-4}	6.80×10^{-4}
	$1-R^2$	6.97×10^{-6}	6.80×10^{-6}	6.75×10^{-6}	6.78×10^{-6}	7.30×10^{-6}	7.29×10^{-6}	7.84×10^{-6}

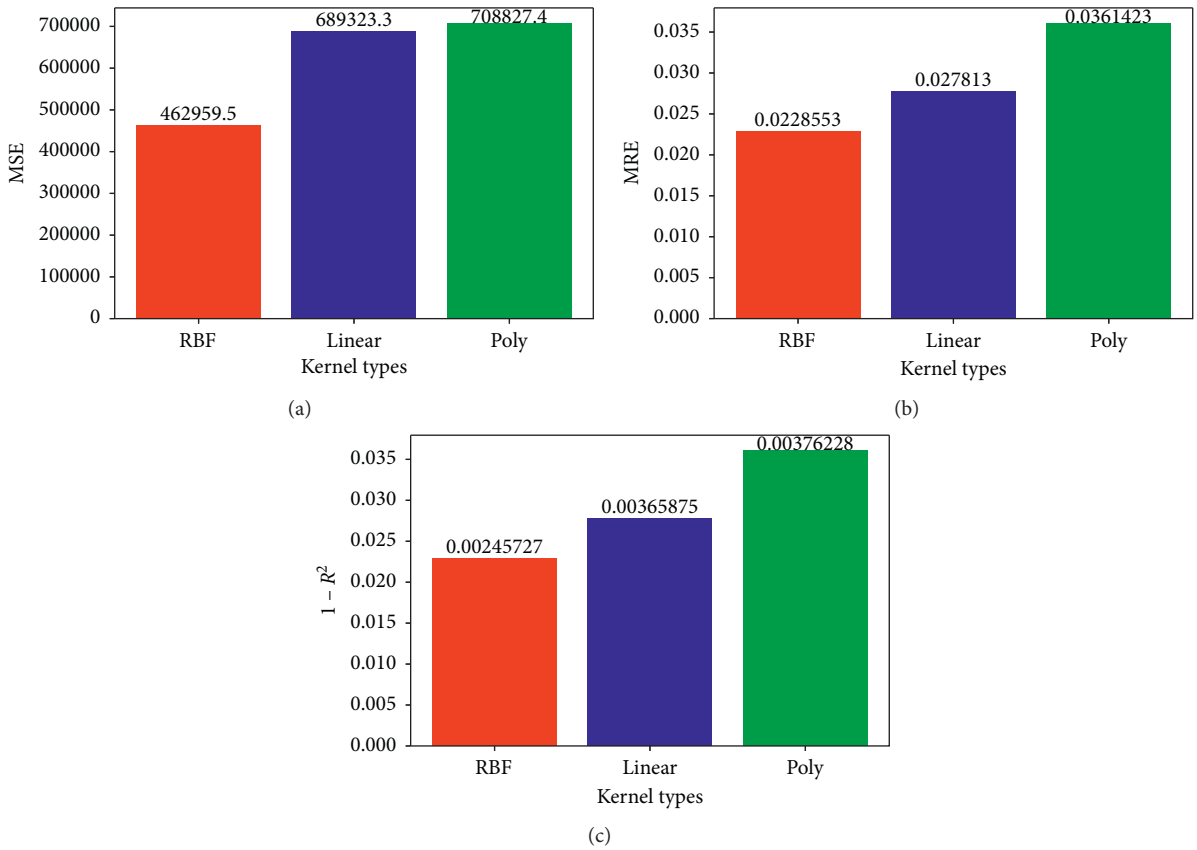


Figure 11: Performance variation over SVR models by using different kernel functions. The result shows that the RBF kernel performs better than the others. (a) MSE, (b) MRE, (c) $1-R^2$.

problem. The results given in Figure 11 and Table 6 are even worse than linear models. SVR models with linear kernels are just linear models whose optimization object is different from the aforementioned liner models. The object of the SVR algorithm is to minimize the maximum divergence between ground truth and predicated value, and we do not adopt this criterion when evaluating our models.

Table 6: Details of the performance of each SVR model when using different kernel functions.

Kernel	RBF	Linear	Poly
MSE	4.6×10^{5}	6.9×10^{5}	7.1×10^{5}
MRE	2.286×10^{-2}	2.781×10^{-2}	3.614×10^{-2}
$1-R^2$	2.457×10^{-3}	3.659×10^{-3}	3.762×10^{-3}

3.5. Neural Network. We conducted several experiments to investigate how the performance of neural network models changes when using different numbers of layers. The results are shown in Figure 12 and Table 7. In our experiments, we found increasing the number of layers of neural network will reduce the error, but the error gets larger after adding too many layers to neural network models.

We also conducted several experiments to investigate how the performance of neural network models relies on the number of units in each layer. The results are shown in Figure 13 and Table 8. We found adding more units in each layer of neural network model reduces the average absolute error consistently, but the average relative error decreases first and increases when too many units are added.

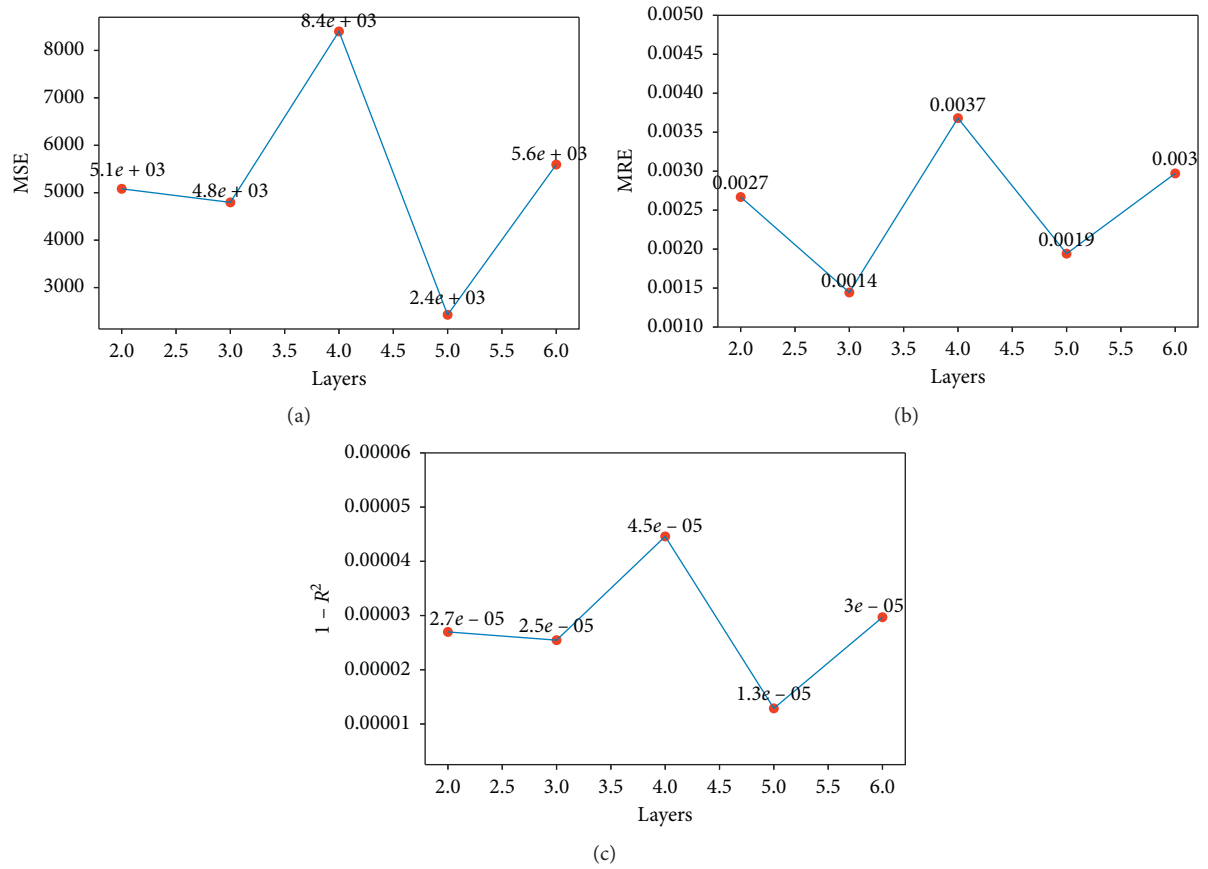


Figure 12: Performance variation as the number of network layers changes. (a) MSE, (b) MRE, (c) $1 - R^2$.

Table 7: Sensitivity tests for the number of network layers.

Layers	2	3	4	5	6
MSE	5.1×10^3	4.8×10^3	8.4×10^3	2.4×10^3	5.6×10^3
MRE	2.699×10^{-3}	1.444×10^{-3}	3.680×10^{-3}	1.941×10^{-3}	2.971×10^{-3}
$1 - R^2$	2.698×10^{-5}	2.546×10^{-5}	4.459×10^{-5}	1.288×10^{-5}	2.969×10^{-5}

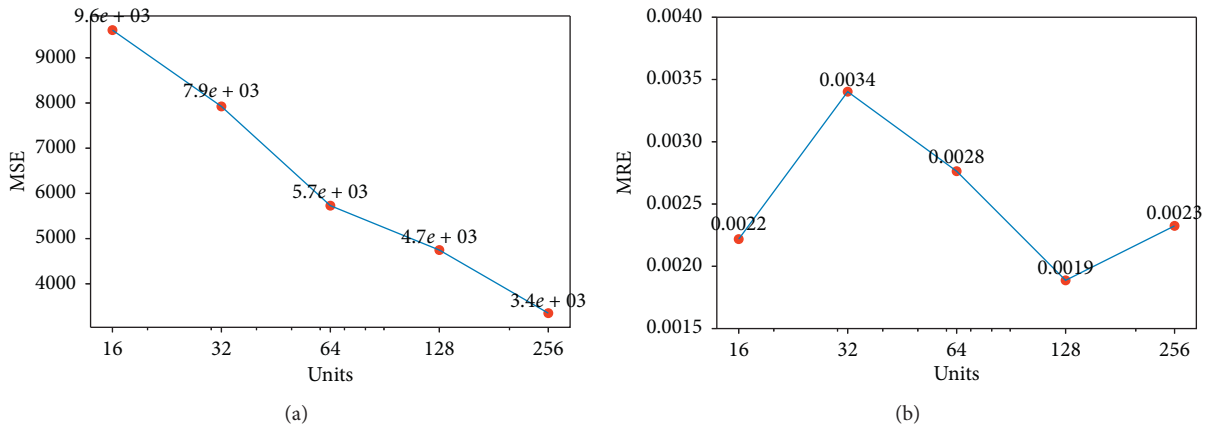
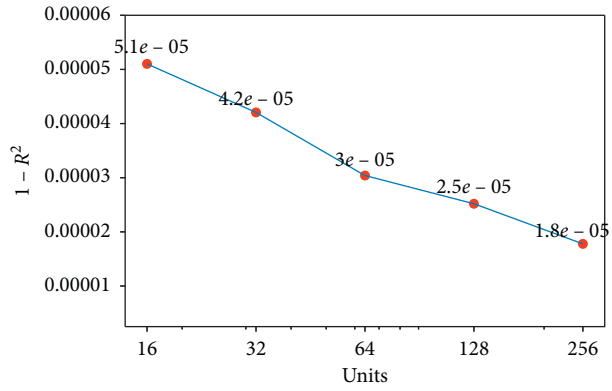


Figure 13: Continued.



(c)

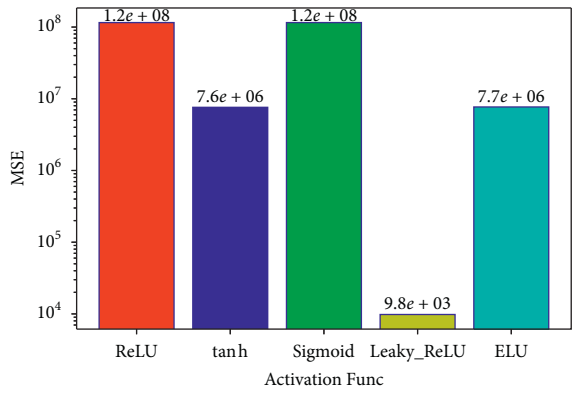
Figure 13: Sensitivity tests for the performance of neural network models and the number of units in each layer. This result suggests that a bigger model may have a better performance. (a) MSE, (b) MRE, (c) $1 - R^2$.

Table 8: Sensitivity tests of neural network models and the number of units in each layer.

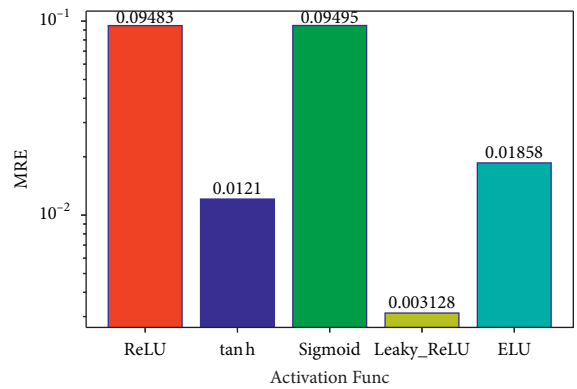
Hidden units	16	32	64	128	256
MSE	9.6×10^3	7.9×10^3	5.7×10^3	4.7×10^3	3.4×10^3
MRE	2.218×10^{-3}	3.402×10^{-3}	2.764×10^{-3}	1.887×10^{-3}	2.325×10^{-3}
$1 - R^2$	5.102×10^{-5}	4.206×10^{-5}	3.040×10^{-5}	2.519×10^{-5}	1.778×10^{-5}

Table 9: Details of the performance when using different activation functions.

Activation	ReLU	tan h	Sigmoid	Leaky_ ReLU	ELU
MSE	1.2×10^8	7.6×10^6	1.2×10^8	9.8×10^3	7.7×10^6
MRE	9.483×10^{-2}	1.210×10^{-2}	9.495×10^{-2}	3.128×10^{-3}	1.858×10^{-2}
$1 - R^2$	6.130×10^{-1}	4.035×10^{-2}	6.131×10^{-1}	5.208×10^{-5}	4.081×10^{-2}

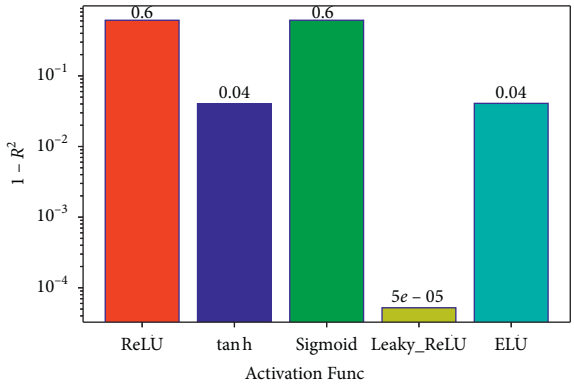


(a)



(b)

Figure 14: Continued.



(c)

Figure 14: Sensitivity tests using different activation functions. The leaky ReLU function outperforms others by a notable margin. (a) MSE, (b) MRE, (c) $1 - R^2$.

Table 10: The best performance of each method that we tried.

Measure	Mean	Linear regression	SVR	KNN	Neural network	GBDT
MSE	1.88×10^8	32133.29	4.6×10^5	1273.00	2426.29	704.2363
MRE	4.01×10^{-1}	6.24×10^{-3}	2.286×10^{-3}	7.642×10^{-4}	1.94×10^{-2}	6.252×10^{-4}
$1 - R^2$	1	1.705×10^{-4}	2.45×10^{-3}	6.75×10^{-6}	1.28×10^{-5}	3.7×10^{-6}

We tested several neural network models with different activation functions. The results are shown in Figure 14 and Table 9. An interesting phenomenon found in this set of experiments is that neural models using Leaky ReLU as activation function performs way much better than other activations.

4. Discussion

We give a comparison of different models on Table 10. From the table, we can conclude that the GBDT algorithm yields the best result among all methods tried. By comparing the performance of different hyperparameter settings of GBDT models, we can discover that a carefully selected hyperparameter setting can improve the performance significantly. This procedure is time-consuming if done manually, so we automated this procedure by testing different preset hyperparameters. The result of stacked GBDT models is very close to GBDT models, but stacked GBDT models are much complex and time-consuming than simple GBDT models, so a simple GBDT model is a better choice in this problem. KNN regression is a simple yet powerful method on this problem, and its performance is better than the neural network models we tried on this problem. This result suggests that neural network models may not be a wise choice for simple table datasets. The results of linear models and SVR models show that the relation between input and output in this problem cannot be grasped by linear models. The SVR models yield the worst performance among all tested methods, even with kernels which can map the input features into higher dimensions. When using the neural network algorithm, the leaky ReLU activation function is recommended. This activation function outperforms other activation functions by a big margin.

5. Conclusion and Future Work

In this paper, we have presented a comprehensive empirical study on the performance of different popular machine learning models for the task of the flow rate of gas pipeline prediction. For future work, we are going to explore the adoption of temporal point process [25–29] for relevant learning tasks in gas pipeline system for its dynamic nature. We will also explore the structure information [30–32] to improve flow prediction from the graph computing perspective.

Data Availability

The dataset and code can be downloaded at https://github.com/programokey/GasPipeline/.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported by the National Nature Science Foundation of China (NSFC U1609220 and 61672231).

References

[1] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," 2016, https://arxiv.org/abs/1611.02167.

[2] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," 2016, https://arxiv.org/abs/1611.01578.

[3] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks,"

IEEE Transactions on Pattern Analysis and Machine, vol. 39, no. 6, pp. 1137–1149, 2017.

[4] D. Silver, A. Huang, C. J. Maddison et al., "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[5] J. Jian, W. Zang, L. Zhou et al., "Wavelet support vector machine-based prediction for emptying time of mobile pipeline," *Oil & Gas Storage and Transportation*, vol. 32, no. 5, pp. 63–67, 2013.

[6] X. Ma, Y. Duan, M. Liu et al., "Prediction of pressure drop of coke water slurry flowing in pipeline by PSO-BP neural network," *Proceedings of the Chinese Society of Electrical Engineering*, vol. 32, no. 5, pp. 54–60, 2012.

[7] W. Shang, J. Cui, C. Song, J. Zhao, and P. Zeng, "Research on industrial control anomaly detection based on FCM and SVM," in *Proceedings of the 2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE)*, New York, NY, USA, August 2018.

[8] Y. Xiaojian, Z. Yue, and Q. Jinghui, "Research on abnormal detection of chemical storage tank based on FCM-ANN," *Computer Applications and Software*, vol. 2, no. 38, 2017.

[9] X. Yang, C. Deng, F. Zheng, J. Yan, and W. Liu, "Deep spectral clustering using dual autoencoder network," in *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4066–4075, Long Beach, CA, USA, June 2019.

[10] Z. Yong, F. Zongde, W. Kanwei et al., "Application of neighborhood rough sets in features selection of wheelsets tread defect images," *Computer Measurement & Control*, vol. 11, 2008.

[11] A. Mohamed, M. S. Hamdi, and S. Tahar, "Self-organizing map-based feature visualization and selection for defect depth estimation in oil and gas pipelines," in *Proceedings of the 2015 19th International Conference on Information Visualisation*, pp. 235–240, Barcelona, Spain, July 2015.

[12] C. Li, X. Wang, W. Dong, J. Yan, Q. Liu, and H. Zha, "Joint active learning with feature selection via cur matrix decomposition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 6, pp. 1382–1396, 2018.

[13] F. Rosenblatt, *Principles of Neurodynamics. Perceptrons and the Theory of Brain Mechanisms*, Cornell Aeronautical Lab Inc Buffalo, Niagara Falls, NY, USA, 1961.

[14] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," 2015, https://arxiv.org/abs/1502.03167.

[15] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2017, https://arxiv.org/abs/1711.05101.

[16] T. K. Ho, "Random decision forests," in *Proceedings of the 3rd International Conference on Document Analysis and Recognition*, vol. 1, pp. 278–282, Montreal, Canada, August 1995.

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

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

[19] H. Drucker, C. J. C. Burges, L. Kaufman et al., "Support vector regression machines," *Advances in Neural Information Processing Systems*, pp. 155–161, 1997.

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

[21] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[22] S. A. Glantz, B. K. Slinker, and T. B. Neilands, *Primer of Applied Regression and Analysis of Variance*, McGraw-Hill, New York, NY, USA, 1990.

[23] F. Santosa and W. W. Symes, "Linear inversion of band-limited reflection seismograms," *SIAM Journal on Scientific and Statistical Computing*, vol. 7, no. 4, pp. 1307–1330, 1986.

[24] M. Gruber, *Improving Efficiency by Shrinkage: The James--Stein and Ridge Regression Estimators*, Routledge, Abingdon, UK, 2017.

[25] S. Xiao, M. Farajtabar, X. Ye et al., "Wasserstein learning of deep generative point process models," *Advances in Neural Information Processing Systems*, pp. 3247–3257, 2017.

[26] S. Xiao, J. Yan, X. Yang et al., "Modeling the intensity function of point process via recurrent neural networks," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, San Francisco, CA, USA, February 2017.

[27] S. Xiao, H. Xu, J. Yan et al., "Learning conditional generative models for temporal point processes," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, New Orleans, LA, USA, February 2018.

[28] S. Xiao, J. Yan, M. Farajtabar, L. Song, X. Yang, and H. Zha, "Learning time series associated event sequences with recurrent point process networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 10, pp. 3124–3136, 2019.

[29] J. Yan, X. Liu, L. Shi et al., "Improving maximum likelihood estimation of temporal point process via discriminative and adversarial learning," in *Proceedings of the IJCAI*, pp. 2948–2954, Stockholm, Sweden, July 2018.

[30] J. Yan, C. Li, Y. Li, and G. Cao, "Adaptive discrete hypergraph matching," *IEEE Transactions on Cybernetics*, vol. 48, no. 2, pp. 765–779, 2017.

[31] J. Yan, M. Cho, H. Zha, X. Yang, and S. M. Chu, "Multi-graph matching via affinity optimization with graduated consistency regularization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 6, pp. 1228–1242, 2015.

[32] J. Yan, J. Wang, H. Zha, X. Yang, and S. Chu, "Consistency-driven alternating optimization for multigraph matching: a unified approach," *IEEE Transactions on Image Processing*, vol. 24, no. 3, pp. 994–1009, 2015.