Distributed Deep Forest and its Application to Automatic Detection of Cash-Out Fraud

YA-LIN ZHANG and JUN ZHOU, Ant Financial Services Group, China
WENHAO ZHENG and JI FENG, National Key Lab for Novel Software Technology, Nanjing University, China
LONGFEI LI and ZIQI LIU, Ant Financial Services Group, China
MING LI, National Key Lab for Novel Software Technology, Nanjing University, China
ZHIQIANG ZHANG, CHAOCHAO CHEN, XIAOLONG LI, and YUAN (ALAN) QI, Ant Financial Services Group, China
ZHI-HUA ZHOU, National Key Lab for Novel Software Technology, Nanjing University, China

Internet companies are facing the need for handling large-scale machine learning applications on a daily basis and distributed implementation of machine learning algorithms which can handle extra-large-scale tasks with great performance is widely needed. Deep forest is a recently proposed deep learning framework which uses tree ensembles as its building blocks and it has achieved highly competitive results on various domains of tasks. However, it has not been tested on extremely large-scale tasks. In this work, based on our parameter server system, we developed the distributed version of deep forest. To meet the need for real-world tasks, many improvements are introduced to the original deep forest model, including MART (Multiple Additive Regression Tree) as base learners for efficiency and effectiveness consideration, the cost-based method for handling prevalent class-imbalanced data, MART based feature selection for high dimension data, and different evaluation metrics for automatically determining the cascade level. We tested the deep forest model on an extra-large-scale task, i.e., automatic detection of cash-out fraud, with more than 100 million training samples. Experimental results showed that the deep forest model has the best performance according to the evaluation metrics from different perspectives even with very little effort for parameter tuning. This model can block fraud transactions in a large amount of money each day. Even compared with the best-deployed model, the deep forest model can additionally bring a significant decrease in economic loss each day.

CCS Concepts: • **Computing methodologies** → **Distributed artificial intelligence**; **Machine learning algorithms**; **Distributed computing methodologies**;

Additional Key Words and Phrases: Deep forest, parameter server, large-scale machine learning

This research was partially supported by the National Key R&D Program of China (2018YFB1004300), the National Science Foundation of China (61751306), and the Collaborative Innovation Center of Novel Software Technology and Industrialization.

Authors' addresses: Y.-L. Zhang, J. Zhou, L. Li, Z. Liu, Z. Zhang, C. Chen, X. Li, and Y. (Alan) Qi, Ant Financial Services Group, China; emails: {lyn.zyl, jun.zhoujun, longyao.llf, ziqiliu, lingyao.zzq, chaochao.ccc, xl.li, yuan.qi}@antfin.com; W. Zheng, J. Feng, M. Li, and Z.-H. Zhou, National Key Lab for Novel Software Technology, Nanjing University, China; emails: {zhengwh, fengj, lim, zhouzh}@lamda.nju.edu.cn.

2157-6904/2019/09-ART55 $15.00
https://doi.org/10.1145/3342241

ACM Reference format:
Ya-Lin Zhang, Jun Zhou, Wenhao Zheng, Ji Feng, Longfei Li, Ziqi Liu, Ming Li, Zhiqiang Zhang, Chaochao Chen, Xiaolong Li, Yuan (Alan) Qi, and Zhi-Hua Zhou. 2019. Distributed Deep Forest and its Application to Automatic Detection of Cash-Out Fraud. *ACM Trans. Intell. Syst. Technol.* 10, 5, Article 55 (September 2019), 19 pages.
https://doi.org/10.1145/3342241

1 INTRODUCTION

Internet companies such as Ant Financial and Alibaba are facing the pressing need of developing algorithms for large-scale machine learning applications, such as recommendation system [12, 22, 38], advertising [28, 41], and fraud detection [7, 50], among which the detection of cash-out fraud for the financial services is a non-negligible task. Cash-out fraud is getting to be a severe threat for online credit financial firms like Ant Financial. The typical procedure of cash-out fraud can be described as the following: The shopper scans a seller-provided QR code with Alipay and pays to the seller using Ant Credit Pay (a credit service provided by Ant Financial; users can consume with the credit and pay back later), and then the shopper gets cash from the seller, and the seller can get a reward from the shopper for helping him to conduct this abnormal transaction. In this process, the shopper gets cash by consuming his credit but not money in Ant Credit Pay, and he is not aimed at buying something but to get the cash by consuming his credit. If he doesn't pay back later, this will result in an economic loss for the company.

Without a proper strategy to detect and prevent fraud behavior, a huge amount of money may be lost each day from cash-out fraud, making it a serious threat. To handle this problem, more and more companies are trying the techniques of machine learning to automatically detect the potential fraud transaction. Nowadays, machine learning based methods such as logistic regression (LR) [23] and multiple additive regression trees (MART) [17] are widely employed, and these methods have brought some improvement for this task. However, a more effective machine learning algorithm is always in strong demand, since this task is closely connected with the economy, and numerous transactions are conducted by Ant Credit Pay each day. Although the amount of fraudulent transactions is only a tiny percentage of the amount of all transactions, the fraudulent transactions will still lead to serious economic loss, and a small improvement with the machine learning model means that a large amount of cash-out fraud will be detected and will bring an obvious decrease in economic loss. Thus, the exploration and deployment of new machine learning approaches for this task is always an important issue for these companies.

Deep forest is a recently proposed approach which opens a new way of building deep models with non-differentiable components, especially tree ensembles [55, 56]. This new kind of deep model is shown to be able to achieve the best performance among all non-DNN methods and give competitive results with state-of-the-art DNN models in a variety of domain of tasks, including face recognition, image categorization, music classification, sentiment classification, and some low dimension data. In addition, the number of layers in the deep forest can be automatically determined according to the metric evaluated, making the model complexity adaptive with exact data (other than a pre-defined DNN structure). What's more, the deep forest approach has much fewer hyper-parameters to tune when comparing to DNN models. In fact, according to the paper, a default setting will produce highly competitive results across all these aforementioned different tasks, making it a good candidate for an off-the-shelf classifier. However, the effectiveness of this model has not been validated for extra-large-scale industrial tasks. Many features of industrial tasks, such as high dimension, class imbalance, and extremely large scale need to be taken into consideration

for the employment of this model in industrial tasks. The distributed version of the deep forest model is needed for handling industrial tasks.

In this work, based on the parameter server based system KunPeng [51, 52], we implemented the distributed version of the deep forest model with industrial standard, which is able to handle millions of high-dimensional data. To meet the need for industrial tasks, many improvements have been introduced for our developed deep forest model based on the original version of the deep forest model. To name a few, MART is employed as the base learner for the consideration of both efficiency and effectiveness, the cost-based strategy is applied for handling extra-imbalanced data, feature selection with MART is adopted for high-dimension data, and different evaluation metrics are provided for automatically determining of the layer amount in the cascade level.

We validate the performance of the deep forest model on the crucial task of automatic detection of cash-out fraud, which is with an extremely large-scale and severely class-imbalanced. The results show that the performance of deep forest is significantly better than all previously deployed methods with regard to different evaluation metrics. Deep forest is able to block fraud transactions in a large amount of money per day. What's more, the robustness of deep forest is also verified through the experiments.

Briefly speaking, the main contributions of this work can be concluded as follows:

- —We implement and deploy the first distributed version of the deep forest model based on the existing distributed system KunPeng.
- —Many improvements are brought based on the original version of the deep forest model, which includes MART as base learners for efficiency and effectiveness consideration, the cost-based method for handling prevalent class-imbalanced data, MART-based feature selection for high-dimension data, and different evaluation metrics for automatically determining the cascade level.
- —We validate the performance of the deep forest model on a crucial task named automatic detection of cash-out fraud, which is with extremely large-scale and class-imbalanced; the results show that the performance of deep forest is significantly better than all existing methods with regard to different evaluation metrics. In addition, the robustness of deep forest is verified through the experiments too.

The rest of the article is organized as follows: First, we give a detailed description of the whole system, in which the improvements of the specified deep forest model are addressed. Then, experiments on the task of automatic detection of cash-out fraud are presented and results from different perspectives are analyzed. Finally, we conclude this article and discuss future work.

2 SYSTEM OVERVIEW

In this section, we give a detailed description of the whole system. Since the system is based on our distributed system KunPeng, we will first briefly introduce the system. Then, we discuss the distributed MART since it is used as the base learner for the deep forest. After that, the specific deep forest model along with the improvements are addressed, which is important for the deployment of this model in an industrial standard. The following part is about distributed implementation and job scheduling. Finally, an easy-to-use graphical user interface is introduced.

2.1 KunPeng System

KunPeng [51, 52] is a parameter server based distributed learning system with parallel optimization algorithms developed to handle the large-scale problems that arise in the industrial community, and it is among the world's largest online learning systems which can process data at petabyte and models with billions of parameters. A parameter server system [13, 30, 47] is composed of two

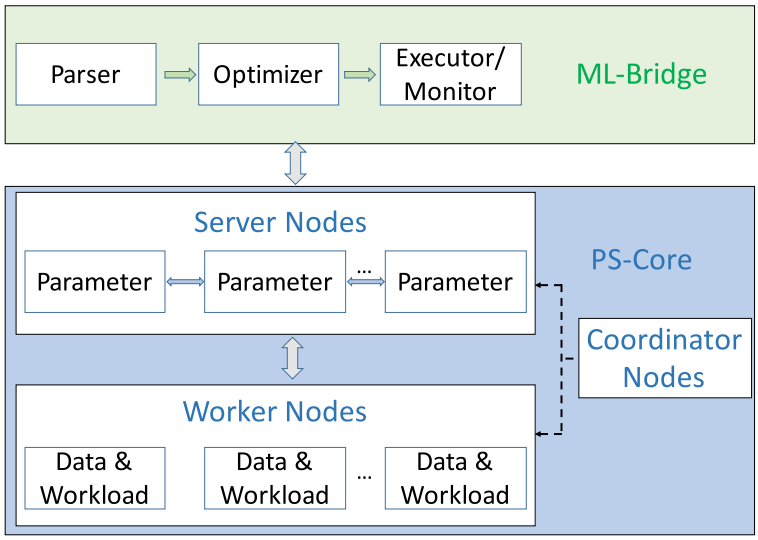

Fig. 1. The simplified architecture of KunPeng, including ML-Bridge and PS-Core.

main parts: the stateless workers, which perform the bulk of computation tasks of model training, and the stateful servers which maintain the parameters of the model. The huge model parameters are distributed on the servers and can be passed to the workers through network communication so that hundreds of billions of model parameters can be handled. In addition, the parameter server also provides a solution to the node failures in the clusters with the help of checkpoints.

Due to these advantages, KunPeng is developed as a production-level parameter server based distributed learning system. Briefly speaking, KunPeng is built with many optimizations: (1) a robust failover mechanism which guarantees the high success rate of large-scale jobs; (2) an efficient communication implementation for sparse data and general communication interfaces; and (3) a user-friendly C++ and Python SDKs [52].

Many popular algorithms are implemented, such as the Follow-the-Regularized-Leader Proximal (FTRL-Proximal) [36], Multiple Additive Regression Trees (MART) algorithm [17] and its extension LambdaMART [5], the Sparse Logistic Regression algorithm [33], Factorization Machines [31], Latent Dirichlet Allocation (LDA) algorithm [2], deep learning [21] framework based on CPU cluster, and so on. Based on this system, many other algorithms can be further developed to handle extremely large-scale tasks.

To make it more convenient to use, ML-Bridge, which is a practical machine learning (ML) pipeline, is provided upon the core part of KunPeng, so that the users can conveniently use the system by writing some simple scripts. The simplified architecture of the whole system is illustrated in Figure 1, with two main parts showing: the ML-Bridge and the PS-Core. The users only need to operate on the ML-Bridge layer.

2.2 Multiple Additive Regression Tree

In this section, we will briefly introduce MART, and address some important features of MART which are utilized in our implementation of the distributed deep forest framework. Then, the distributed implementation of MART in KunPeng is briefly explained. Since MART is already implemented in KunPeng with great efficiency and effectiveness, and significant improvements have been shown to be reached by making it as the basic learners, we will use it as the basic building

block for the distributed deep forest implementation, and any other kinds of building blocks can also be employed for the distributed version of the deep forest.

MART, which is also known as Gradient Boosting Decision Tree (GBDT) and Gradient Boosting Machine (GBM) [17], is a widely used machine learning algorithm in both academic and industrial fields, because of their high effectiveness and great interpretability [52]. In fact, most of the winning Kaggle competitions or data science projects always use an ensemble of MART or its variants [9] as the final model, due to its superior performance.

To give a quick glance at MART, we will first briefly introduce boosting decision tree [15]. Boosting decision tree constructs the model by additively fitting a tree model to the current residual. Let $\boldsymbol{x}_i$ denote the i-th instance and y_i is the corresponding label of this instance. At the t-th iteration, we have the prediction F_{t-1} from the first $(t-1)$-rounds, then the tree model f_t is learned to minimize the following objective:

$$L^t = \sum_{i=1}^{n} l(y_i, F_{t-1}(\boldsymbol{x}_i) + f_t(\boldsymbol{x}_i)) + \Omega(f_t), \tag{1}$$

in which l is the loss function and Ω is the regularization term which controls the complexity of the tree model f_t. Then, the prediction F_t from the t-round is

$$F_t(\boldsymbol{x}_i) = F_{t-1}(\boldsymbol{x}_i) + f_t(\boldsymbol{x}_i). \tag{2}$$

Similar to boosting decision tree, MART is also additively constructed with the philosophy of fitting the residual. However, in boosting decision tree, finding the best model f_t for an arbitrary loss function L may get to be computationally infeasible in many conditions. To handle this, MART [17] is proposed; by making an approximation to the real residual with the steepest-descent method, the so-called pseudo-residual is fitted. Furthermore, second-order approximation is widely explored to efficiently optimize the objective [18], and it has been implemented in most of the systems such as XGBoost [9] and lightGBM [25]. The objective can be shown as follows:

$$\hat{L}^t = \sum_{i=1}^{n} \left[l(y_i, F_{t-1}(\boldsymbol{x}_i)) + g_i f_t(\boldsymbol{x}_i) + \frac{1}{2} h_i f_t^2(\boldsymbol{x}_i) \right] + \Omega(f_t), \tag{3}$$

in which $g_i = \partial_{F_{t-1}} l(y_i, F_{t-1}(\boldsymbol{x}_i))$ and $h_i = \partial^2_{F_{t-1}} l(y_i, F_{t-1}(\boldsymbol{x}_i))$ are the first-order and second-order gradients on the loss function.

Many scalable systems, such as XGBoost [9] and lightGBM [25], are some well-known implementations for this model and its variants, with additional optimization of the speed and memory usage. Similarly, KunPeng-MART is the parameter server based implementation on the KunPeng system. Currently, many machine learning tasks in Ant Financial and Alibaba are using KunPeng-MART on a daily basis including many predictive tasks involving machine learning models during the double 11 online shopping festival and other daily online financial services [1].

There are two particular features for MART that are worth noticing and important for further use in our system. We will give a brief discussion below.

First, severe class-imbalance data is often encountered in various tasks including fraud detection [6], anomaly detection [7], and medical diagnosis [26]. Take two-class classification as an example; the number of some class of data may be seriously less than that of the other class of data. If we simply use the common strategy without special design, the performance may be pretty unsatisfactory [24]. To handle the class-imbalance problem, two different solutions are always employed, i.e., the cost-based methods [57] and the sampling-based methods [35].

Note that the cost-based strategy [57] can be naturally embedded to MART, and thus in our implementation, we use cost-based strategy to deal with the class-imbalance problem. Concretely, the weight is assigned to each sample, and higher weights will be set to the class with less amount

(which are always the ones with larger cost if they are misclassified) while smaller weights are set to the class with more amount (which are always the ones with smaller cost if they are misclassified). Thus, the goal in Equation (3) can be modified as follows:

$$\hat{L}_w^t = \sum_{i=1}^{n} w_i[l(y_i, F_{t-1}(\boldsymbol{x}_i)) + g_i f_t(\boldsymbol{x}_i) + \frac{1}{2} h_i f_t^2(\boldsymbol{x}_i)] + \Omega(f_t), \tag{4}$$

in which w_i is the importance weight associated with instance $\boldsymbol{x}_i$.

Second, extremely high-dimensional data is ubiquitous in industrial tasks, and feature selection [11] is always a necessary and important step in the whole pipeline. Fortunately, the estimates of feature importance can be calculated by MART and feature selection can be performed [48]. Generally speaking, the importance score of each attribute indicates the importance of it when constructing a tree. The more frequently a feature is used to make a decision, the more important it is. Concretely, for every single tree, the importance of an attribute j is calculated by

$$\hat{I}_j^2(T) = \sum_{l=1}^{L-1} \hat{i}_l^2 \mathbb{1}(v_l = j), \tag{5}$$

in which L is the number of leaf nodes (and $L-1$ is the number of non-terminal nodes), v_l is the corresponding feature associated with node l, $\hat{i}_l^2$ is the corresponding empirical improvement in square-error from the splitting, and $\mathbb{1}$ is the indicator function. Then, as shown in Equation (6), the global importance $\hat{I}_j^2$ of an attribute j is calculated by averaging the importance value $\hat{I}_j^2(T_m)$ that the feature obtained from each single tree [17].

$$\hat{I}_j^2 = \frac{1}{M} \sum_{m=1}^{M} \hat{I}_j^2(T_m). \tag{6}$$

To handle the industrial tasks, the distributed version of the MART model is implemented in our distributed system KunPeng, which is named KunPeng-MART. Currently, many machine learning tasks in Ant Financial and Alibaba are using KunPeng-MART on a daily basis. There are many challenges encountered when developing distributed MART, such as the storage problem and computation and communication cost [52]. To meet the need for extremely huge storage, a data parallelization mechanism is employed in KunPeng-MART. To be specific, each worker only stores a subset of the whole data for each feature, and the main workflow for splitting a node is as follows: (1) each worker calculates the local weighted quantile sketch with the data stored on it; (2) each worker pushes the local weighted quantile sketch to servers, and the servers merge them up to a globally weighted quantile sketch, and find the splitting value; (3) each worker pulls the splitting value from servers and splits samples to two nodes. Another key challenge is that the computation and communication cost of the split-finding algorithm may become very high. To handle this, the communication schema of KunPeng is employed to reduce the cost of merging local sketches, and this really speeds up the whole process. We will use Kunpeng-MART as the building block for the distributed version of the deep forest, and employ the aforementioned features of MART when handling our tasks.

2.3 The Specified Deep Forest Structure

In this section, we start with a brief introduction of the original deep forest model, and our specified version of the deep forest model is introduced. We bring many improvements to the original deep forest model for the consideration of better performance for real industrial tasks, which will be explained in detail below.

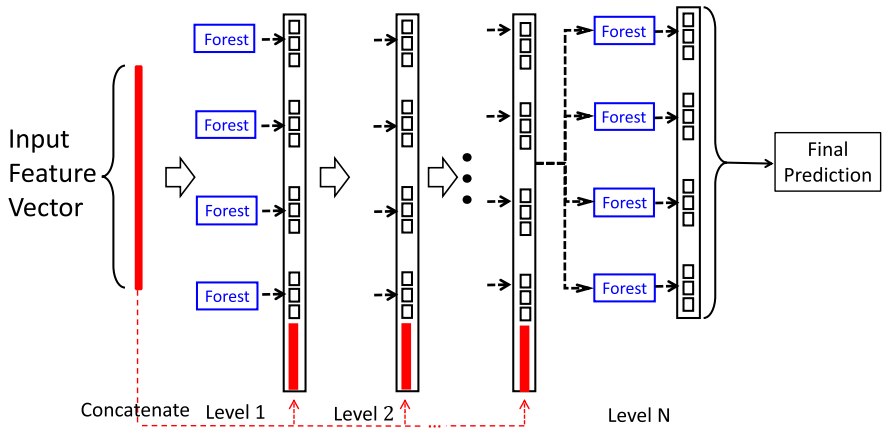

Fig. 2. The cascade module of the original deep forest model [55]. Suppose each layer consists of four forests and there are three classes to predict; thus, each forest will output a three-dimensional vector, and each layer will output a 12-dimensional vector, which is then concatenated with the original feature vector as the new representation, and input to the forest in the next layer.

Deep forest [37, 55, 56] is a recently proposed deep learning approach which uses tree ensembles [53] as its building blocks in each layer. The original version consists of two modules, i.e., the fine-grained module and the cascading module. As discussed in the original paper [55], when there are spatial or sequential feature relationships, the multi-grained scanning process helps improve performance apparently. In our task, the data is not spacial or sequential, so the fine-grained module is removed to achieve better efficiency.

We focus on the cascading module for our implemented system. As shown in Figure 2, in the cascading module of the original deep forest model, a multi-layer structure is built. Each layer can be regarded as an ensemble of ensemble, which consists of several base learners, and each base learner is an ensemble of decision tree forests, i.e., a random forest [4] or a completely random tree forest [32]. Here, different types of forests are used for the purpose of improving diversity [42] so that the overfitting problem can be alleviated.

The model is built layer by layer. Each layer receives the processed feature information outputted from its preceding layer, as well as the original input features of the data (the red part in Figure 2), and concatenate them together as the input features for the forests in this layer (the first layer uses only the original feature), then the processing results of all these forests in this layer are outputted to the next level. The processing result of one level is the combination of the class vectors generated by its forests. For example, if m base learners are trained in each layer for a k-class task, each forest will output a k-dimensional vector, and the output of each layer is a $(m * k)$-dimensional vector for each sample, and this vector will be concatenated with the original feature of this sample as the input feature for the base learners in the next layer. The method of passing the output of one layer's base learners as input to the next layer is related to stacking [3, 46]. As suggested in [45, 53], K-fold cross-validation is conducted in each layer to reduce the risk of overfitting. After expanding a new layer, the performance of the whole cascade can be estimated on a validation set, and the cascading process will terminate when the accuracy on validation set stops increasing; thus, the number of cascade layers is automatically determined in the deep forest.

To meet the need for the applications in Ant Financial, the original version of deep forest may not be enough, and we develop our specific version of the deep forest pipeline, which is shown in Figure 3. There are some challenges that should be taken into consideration when building a statistical model for real-world tasks.

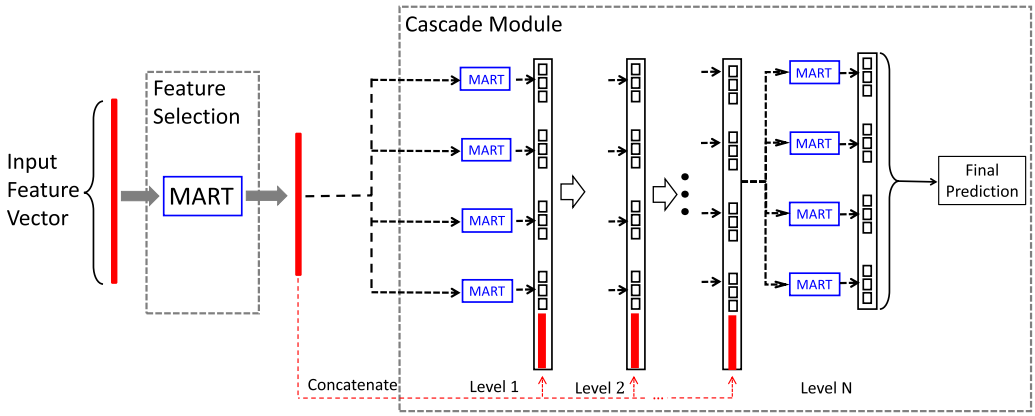

Fig. 3. The pipeline of the specified deep forest structure, which includes feature selection module and cascade module. MARTs are used as the base learners.

Firstly, the raw training data is often in high-dimensional space; usually, thousands or even more raw features are used to represent a single entity or a transaction, in which many irrelevant attributes may exist. This can make the training process really time-consuming and resource -consuming. In addition, when deploying a model for real-time prediction, it is not economically efficient and unnecessary to calculate every attribute for each prediction. To handle this problem, the raw training data is first trained on a MART for feature importance evaluation. Then, based on the feature importance score, feature selection is conducted for the training set. The feature dimension can be greatly reduced through this procedure, while the performance will not dramatically degrade so that the filtered features can be used for further model training.

Secondly, the data we are facing is often extremely class imbalanced; the positive samples may be much rarer, compared with negative samples. Sometimes, the number of negative samples can be as much as 10,000 times the number of positive ones. Therefore, a mechanism which can effectively handle such a situation is needed in order to get a reasonable result. To deal with this problem, the cost-based strategy is employed in each base learner so that the class-imbalance problem is addressed. Concretely, samples from different classes will be allocated with different weights, while the positive ones (which are also the minority) are always with bigger weights, meaning that the misclassification of them will lead to a larger loss.

Thirdly, for the extremely large-scale tasks in industrial settings, both the data size and the features size can be huge, and the running time may be pretty lengthy, so efficiency and effectiveness are both important. To meet this, all base learners in the original deep forest (i.e., random forests and completely random tree forests) are replaced with the MART model implemented in our Kunpeng system, which is shown to be able to provide excellent performance with consideration of both efficiency and effectiveness. What's more, the supplement of [55] also shows that a significant improvement can be reached by using MART as the base learners.

Finally, for many tasks in the industrial scenario, the evaluation metric may be specific, so the original applied metric accuracy in the deep forest cannot meet all of the need (e.g., accuracy may not be a good metric for the severely imbalanced data). To handle this problem, we provide more metrics (such as AUC [16] and F1-Score [39]) for the automatic growing of the cascading structure. What's more, the specifically designed evaluation metrics can also be introduced for personalized demand.

Furthermore, note that we use MART as the base learner of deep forest since it is already implemented in KunPeng, and better efficiency and effectiveness can be obtained in this way, while

in the original paper, random forest [4] and completely random tree forest [19, 32] are used as the base learner to provide better diversity (which is crucial for ensemble methods) [27, 43, 53]; when replacing all base learners with MART, the diversity is damaged to some extent, so some strategies are applied in MART to alleviate this problem, including instance sampling, setting different number of trees for each forest, and setting different depth for trees in different forests.

It should be noticed from the above description, each base learner can be trained in a distributed fashion, and all the base learners within a layer can also be trained in parallel, making the whole process easy to be implemented in a distributed fashion. The next section will briefly explain how to build such a model via a parameter server based distributed learning system.

2.4 Distributed Implementation and Job Scheduling

In industrial scenarios, we are always confronted with data with tremendous size and high dimension, which means that the distributed version of the algorithm is needed. On the other hand, the deep forest model has been proved to be effective in a range of different tasks; however, the performance of this model has not been verified in extremely large-scale tasks. In this section, we introduce the distributed framework and job scheduling method for the development of the distributed version of the deep forest model.

The distributed deep forest framework is built upon the widely used parameter server based system KunPeng in Alibaba and Ant Financial. Based on the KunPeng architecture, the distributed version of deep forest contains three different kinds of nodes: (1) the worker nodes, which execute the heavy computing tasks; (2) the server nodes, which maintain the globally shared parameters; and (3) the coordinator nodes, which coordinate the worker and server nodes, and perform job scheduling for the whole task.

To deploy the deep forest algorithm in a distributed manner, a key problem is how to do job scheduling for the whole pipeline. In each layer, the process of deep forest consists of the following sub-jobs: (1) data preparation process, which splits the whole dataset into different training and valid fold, since k-fold cross-validation is needed to reduce the risk of over-fitting; (2) model training process, which trains different base learners based on the split training data parallelly; (3) prediction process, which makes prediction on the split valid data parallelly; (4) combination and concatenation process, which combines the output of different base learners from the previous level and concatenates these outputs with the original features, and this processed data will be used as the input feature for the subsequent level.

To perform efficient job scheduling, the directed acyclic graph (DAG) [44] is employed in our system. A directed acyclic graph is a finite directed graph with no directed cycles. As shown in Figure 4, we regard each aforementioned process as a node in the graph, and only the corresponding processes are connected. The pre-conditions of one node are the input of that node. Only when all the pre-conditions of one node are satisfied, that node could be executed. What's more, each node is executed separately, which means that the failure of one node will not influence other nodes. Each node is dependent on its inputs, which means that if and only if all the pre-conditions of the node are finished, that node is allowed to execute. In this way, the node will not wait for the performing of irrelevant nodes, and the waiting time will be significantly shortened since each node only needs to wait for the corresponding per-nodes finishing executing. For example, once the splitting of the k-fold training data is finished, the corresponding model on this fold can be called to do training, rather than waiting until all of the data are prepared, and the prediction on the related valid data will be called to execute if the training is finished. In addition, this design provides a better solution for failover. For example, if some node is crashed for some reason, we will only need to rerun from this node instead of running the whole algorithm from the beginning since its pre-conditions are finished successfully.

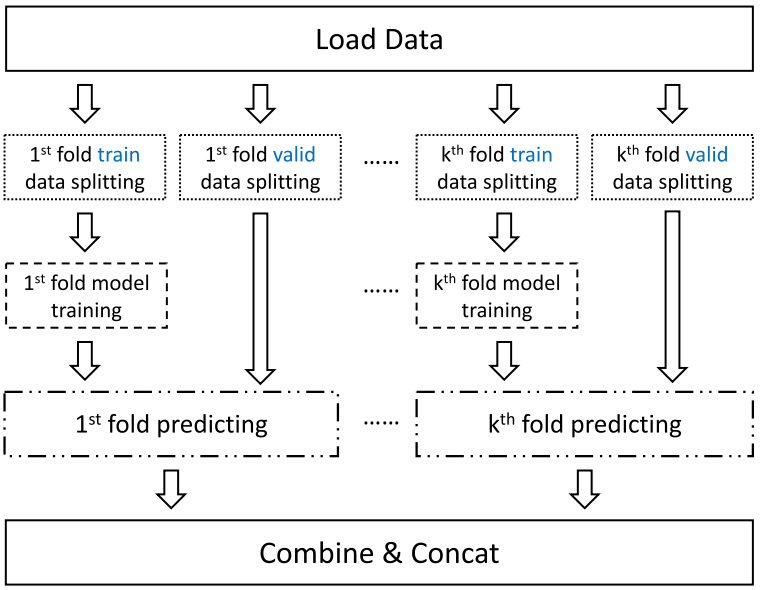

Fig. 4. Job scheduling with DAG; each rectangle represents a process, and only the corresponding processes are connected with each other.

2.5 Graphical User Interface

Data scientists in Ant Financial and Alibaba are facing hundreds of different machine learning tasks each day. Numerous new tasks are created and each task is different by its own nature. Therefore, how to efficiently build and evaluate a model is critical for productivity. In order to solve such problem, in Ant Financial, the Platform of Artificial Intelligence (PAI for short)[1] is developed, which decouples the algorithms from different algorithm engines, for example, KunPeng, MaxCompute, MPI, and so on, and provides a uniform graphical user interface (GUI) for data scientists to process data, invoke multiple machine learning algorithms, create task pipeline at cloud, and so on.

As noted earlier, the deep forest algorithm is robust enough to handle different domains of tasks, making it one of the best choices when facing a new task. The parameter server based implementation of base leaner enables the model to handle even extremely large-scale real-world problems, and we have implemented a deep forest module in the PAI platform; the data scientists are able to create a deep forest model within a browser. That is, with only a few clicks of the mouse, the deep forest model is ready to train on massive training data and ready for deployment.

The demo of an overall GUI of the process with a deep forest model is illustrated in Figure 5. Each node represents an atomic operation, which includes loading the data, building the model, making predictions, and performing the evaluation. All the details of the deep forest model are encapsulated into a single node; the only thing needed to specify is which base learner to use, how many base learners per layer, and the detailed configuration of each base learner, as shown in Figure 6. The default base learners are MART, as introduced before.

The arrowed line indicates the sequential dependency and data flow from one to the other. By only a few drag-and-clicks, the user is able to create the deep forest model within minutes, and the evaluation results will be analyzed once the training is finished. If a deep forest is determined to be deployed, the trained model can be directly inputted to the data awaiting assessment.

[1] pai.alipay.com.

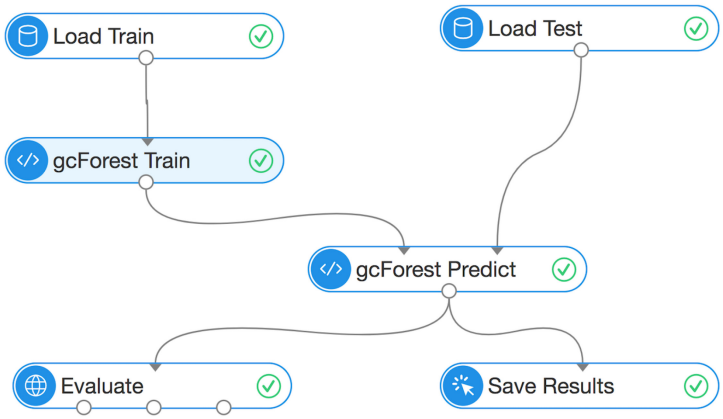

Fig. 5. The overall GUI of deep forest on PAI; each node represents an atomic operation.

```
{'mart': { 'objective': 'binary:logistic', 'treeCount': '10', 'maxDepth': '5', 'shrinkage': '0.3', 'sampleRatio': '1'},
{'mart': { 'objective': 'binary:logistic', 'treeCount': '11', 'maxDepth': '5', 'shrinkage': '0.3', 'sampleRatio': '1'},
{'mart': { 'objective': 'binary:logistic', 'treeCount': '12', 'maxDepth': '5', 'shrinkage': '0.3', 'sampleRatio': '1'},
{'mart': { 'objective': 'binary:logistic', 'treeCount': '13', 'maxDepth': '5', 'shrinkage': '0.3', 'sampleRatio': '1'}
```

Fig. 6. A simple example of the configuration for the training process.

3 APPLICATION

In this section, we validate the effectiveness of the deep forest model on one important application for Ant Financial, namely, the automatic detection of cash-out fraud. We will first give a detailed introduction to the task of cash-out fraud and a brief introduction of the data preparation process, then the empirical results are shown and analyzed from different perspectives. What's more, the robustness of the deep forest model is validated.

3.1 Task Description

Cash-out, which means pursuing cash gains using illegal or insincere means, is a troublesome problem in the scenario of the credit card, public accumulation funds, and credit financial companies like Ant Financial. Similar to the credit card, Ant Credit Pay is a credit production provided by Alipay. Alipay provides each user with a certain amount of credit, and the user can consume with the credit but not money to buy anything using Alipay and pay back later, which is pretty similar to the use of a credit card. However, there are always some people who try to pursue illegal gain from these credit productions, and this becomes a serious threat for companies like Ant Financial. Thus, the automatic detection of the cash-out fraud for Ant Credit Pay and the corresponding treatment strategy for the detected potential cash-out transactions are pretty crucial for risk control in these companies.

The cash-out fraud by Ant Credit Pay always follows the following process: the shopper makes a transaction to the seller with Alipay by scanning a QR code, and pays the seller using Ant Credit Pay. In this way, the credit but not the money of the shopper is consumed and the shopper can get cash from the seller. The threat is that the shopper will not pay back the money later, which will result in an economic loss for the company. And also, similar activities like cash-out fraud may

Table 1. The Number of the Training and Test Instances (Including the Number of Positive Instances, Negative Instances, and All Instances)

	# Pos. Ins.	# Neg. Ins.	# All Ins.
Train	171,784	131,235,963	131,407,704
Test	66,221	52,423,308	52,489,529

be a potential threat to the credit system. Without a proper strategy to detect them, millions of CNY may lose each day. What we need to do is to provide the system with the ability of detecting the potential hazard of cash-out when a QR code is scanned for a transaction, and the system can automatically disable the consuming of the credit in Ant Credit Pay for this payment if the transaction is detected to be a highly possible cash-out fraud, so that the shopper can only consume with money but not the credit; thus, the economic loss can be avoided.

3.2 Data Preparation

We formulate this task as a binary classification problem to predict if a transaction is a potential fraud transaction, and collect the original features from four different aspects, i.e., the seller features, which describe the identity information of the seller (such as the transaction amount information); the buyer features, which describe the identity information of the buyer (such as the gender and age information); the transaction features, which describe the information of the exact transaction (such as the time of the transaction), and the history features, which describe the information of both seller and buyer (such as the history transaction amount and other statistics in a period). Altogether, more than 5,000 dimension features are collected when each transaction is happening, with both numerical and categorical features included, so this problem is a hybrid modeling problem with huge feature size. What's more, we need to address that this task is naturally class-imbalanced, since the amount of fraudulent transactions is only a tiny percentage of the amount of all transactions.

To construct the training and test data, we sampled the training data from data collected from O2O (Online to Offline) transactions using Ant Credit Pay during some successive months, and the sampled transactions of the same scenario during the next few months are used as test data. The statistical information of the data is shown in Table 1. As we can see, this task is with extremely large-scale (with more than 131 millions of samples for training) and is severely class-imbalanced (the number of negative instances is more than 700 times the number of positive instances in the training set).

As we discussed above, the original collected features are with the size up to 5,000, among which many irrelevant attributes may be included. If we simply use all these raw features, the whole procedure may get to be too time-consuming and resource-consuming, and it will really reduce the efficiency when deployed as a real-time service, making it a bad choice. On the other hand, according to the business experience, the size of the finally used features can be greatly reduced without significant degradation of the performance. To bypass this obstructor, as shown in Figure 3, before training the final models, feature importance is firstly calculated and feature selection is performed with the help of MART.

Concretely, we first run a MART model with all collected features and then feature importance scores are calculated with the obtained model using Equation (6). Based on the obtained feature importance scores, feature selection is then performed. We evaluate the performance with different selected feature size. Empirical results show that with the first 300 selected features (which are with the higher feature importance scores), the retrained model can already achieve the competitive

Table 2. The Results Using the Common Metrics, i.e., AUC, F1-Score, and KS-Score

	AUC	F1-Score	KS-Score
LR	0.9887	0.4334	0.8956
DNN	0.9722	0.3861	0.8551
MART	0.9957	0.5201	0.9424
gcForest	**0.9970**	**0.5440**	**0.9480**

performance with the model trained using the whole features, which also validates the redundancy of the raw features. Thus, the subsequently used features are filtered to be the 300 dimensions with higher importance scores and the whole efficiency is greatly improved.

3.3 Empirical Results

In this section, empirical results are shown from different perspectives and a detailed discussion is provided below.

3.3.1 Experiment Setting. Since the data is on an extremely large scale, the experiments are conducted with the distributed learning system KunPeng. To validate the effectiveness of the deep forest algorithm on this extremely large dataset, many KunPeng based algorithms are compared, including logistic regression (LR) [33], deep neural network (DNN) [21], and MART [17], in which LR and MART are the previously and currently deployed machine learning based methods, which have achieved some success during the past period. Since the problem is a hybrid modeling problem, we believe that DNN may not be a good choice for this task, and experiments are conducted with DNN to validate the effectiveness. Note that since the data is severely imbalanced, the cost-based strategy is employed in these models, i.e., instances from the positive and negative classes are allocated with different weights. We need to address that, for MART, 600 trees are used to get better performance (and a larger number of trees will not further improve the performance), while in the deep forest, each MART is with at most 200 trees, but not the same number of 600. For DNN, following the procedure explained in [55], we examine a variety of architectures with a validation set, and pick the one with the best performance, and retrain the whole network to get the final result. Specifically, the activation function is selected from ReLU and sigmoid, the unit number of each hidden layer is selected from 100, 300, 500, 1,000, and the dropout rate is selected from 0.25 and 0.5.

3.3.2 Common Metrics. We first evaluate the performance with the widely used metrics for binary classification tasks in Ant Financial, including the AUC (Area Under the ROC Curve), the F1 score, and the KS (Kolmogorov-Smirnov) score. These metrics will not directly reflect the economic influence by each model, but even a slight improvement at the third decimal place for these metrics is significant since this improvement can bring millions of decrease of economic loss, according to the business experience, so we first give the result and analysis from these metrics.

The results are shown in Table 2; as we can see, the deep forest method (gcForest for short) performs much better than all other existing methods. The advantage is especially noticeable for F1-Score since the AUC and KS-Score are already pretty high for other methods. MART performs as the second best, validated by its effectiveness. We need to address that the MART model is fine-tuned, and 600 trees are needed to reach this performance (and a larger number of trees will not further improve the performance), which may be the upper limit that can be reached by this model. However, for deep forest, only with 200 trees for each MART, and with slight tuning, the

Table 3. The Results Using the Specified Metrics, i.e., the Recall of the Positive Samples (the Potential Cash-Out Cases) Under a Given Interrupt Rate

	1/10,000	1/1,000	1/100
LR	0.3708	0.5603	0.8762
DNN	0.3165	0.4991	0.8471
MART	0.4661	0.6716	0.9358
gcForest	**0.4880**	**0.6950**	**0.9470**

Here, 1/100 means that 1/100 of all transactions are interrupted.

performance is already much better than the best baseline, and as we will show later, even with only 50 trees (the default setting) for each MART in deep forest, the performance is still better than the fine-tuned MART model. Here, DNN performs pretty unsatisfactorily, which verified the weakness of DNN for handling the hybrid modeling problem (while DNN is more suitable for the continuous modeling tasks like image recognition). LR performs unsatisfactorily too when comparing to MART and deep forest.

3.3.3 Specified Metrics. Besides the common metrics, as a real-world task, many specified metrics are always needed for analysis, with the consideration of practical use. For our task, one important metric is the recall of the positive samples (the potential cash-out cases) under a given interrupt (by disabling Ant Credit Pay for such transactions) rate. For example, if we can stand that 1/10,000 of all transactions to be interrupted, we will definitely hope that the selected cases contain as many positive samples as possible. When deploying these models, a threshold is always determined so that some ratio of the transaction will be interrupted. Since the volume of transaction is tremendous, a small improvement under this metric will be associated with a large number of transactions and a large amount of money can be saved for the company.

In Table 3, the recalls under different interrupt rates (e.g., 1/100 means that 1/100 of all transactions are interrupted) are provided. As we can see from the results, LR and DNN perform pretty unsatisfactorily; the MART brings great improvement from these methods. At any level of interrupt rate, the deep forest model can always capture most potential cash-out transactions, which means that it will provide the best performance when deployed. The improvements can be bigger than 2% even compared with the fine-tuned MART under the interrupt rate of 1/10,000 and 1/1,000, which is the currently deployed model for this task. The improvements for these metrics are much significant according to the business specialist, and these metrics are much more relevant to the real economic influence when the model is deployed.

3.3.4 PR Curve. Furthermore, according to the business experience, a PR (Precision-Recall) curve is always a good choice to provide a visual comparison, and the decision is always made based on this metric, so the improvement in this metric is more approved. A PR curve shows the relationship between precision and recall for each possible cut-off. Generally speaking, the curve with the bigger covering area is always regarded as a better one when comparing to the one with a smaller coverage area.

We draw the PR curve of different tested methods in Figure 7. The result is much cleare; the PR curve of deep forest covers those of all other methods, meaning that the performance of deep forest is much better than other methods, validating the effectiveness of the deep forest model. One interesting result is that the MART turns out to behave unsatisfactorily in the high score part, making it somewhat unsatisfactory for deployment.

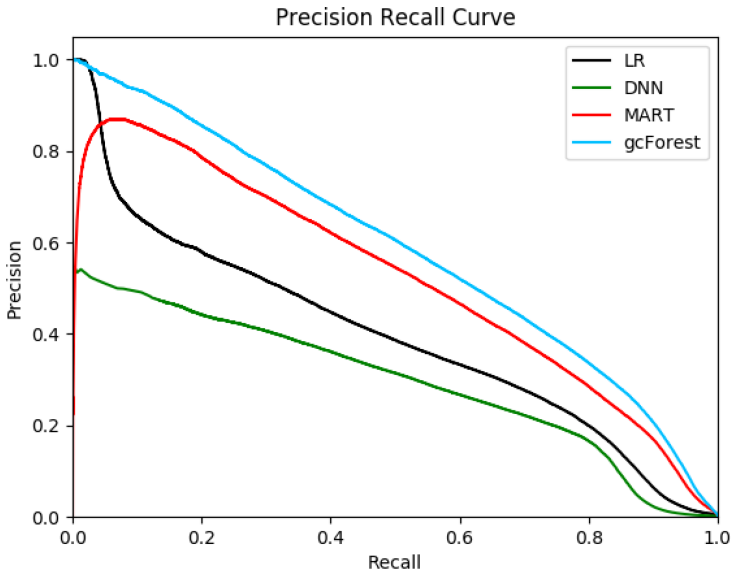

Fig. 7. The PR curve of LR, DNN, MART, and gcForest.

Table 4. The Results Using the Common Metrics, i.e., AUC, F1-Score, and KS-Score

	AUC	F1-Score	KS-Score
MART	0.9957	0.5201	0.9424
gcForest-d	0.9962	0.5247	0.9444
gcForest-t	**0.9970**	**0.5440**	**0.9480**

3.3.5 Economic Benefit. Until now, we have provided analysis from different perspectives, and all of these results validate the effectiveness of the deep forest model on this extremely large-scale task. Furthermore, when deployed, the deep forest model can block cash-out fraud transactions in a large amount of money every day (the detail is business confidential). Even compared with the best-deployed model (MART with 600 trees), the deep forest model (200 trees for each MART) can still additionally bring a significant decrease of economic loss each month, making it a better choice for this task.

3.4 Robustness Analysis

In this section, we give a brief discussion on the robustness of the deep forest model.

According to the original paper of deep forest [55], a default setting of the deep forest model will produce highly competitive results in a range of tasks, which means that less effort on parameter tuning is needed when we use this model. To validate the robustness of parameter setting for he deep forest model in a large-scale setting, we compare performances among the deep forest model with default setting (using four MARTs as base learner, each with only 50 trees), the slightly-tuned deep forest model (by only changing the number of trees to 200 in each MART), and the best tuned model of the MART (with 600 trees).

The corresponding results of the tested models are shown in Table 4, Table 5, and Figure 8 (gcForest-d for the default setting of deep forest and gcForest-t for the slightly tuned deep forest).

Table 5. The Results Using the Specified Metrics, i.e., the Recall of the Positive Samples (the Potential Cash-Out Cases) Under a Given Interrupt Rate

	1/10,000	1/1,000	1/100
MART	0.4661	0.6716	0.9358
gcForest-d	0.4703	0.6775	0.9397
gcForest-t	**0.4880**	**0.6950**	**0.9470**

Here, 1/100 means that 1/100 of all transactions are interrupted.

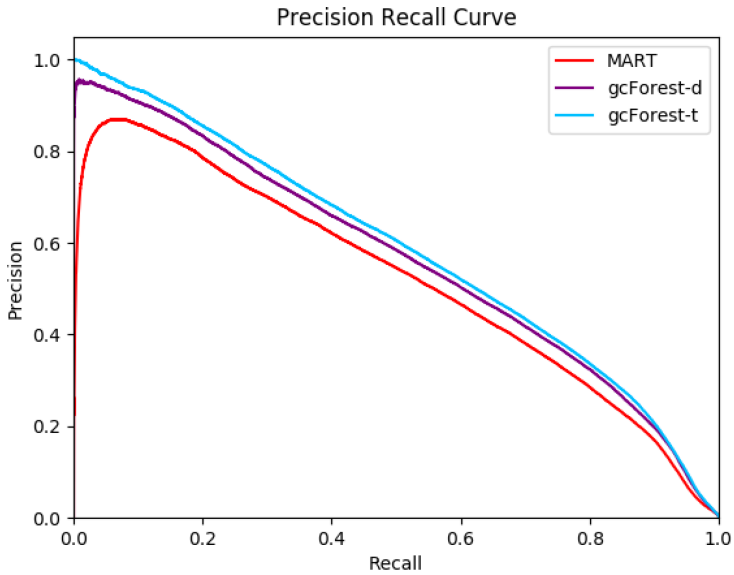

Fig. 8. The PR curve of deep forest with default parameters (gcForest-d), slightly tuned deep forest (gcForest-t), and fine-tuned MART (MART).

As we can see, even without any tuning, the performance of the default setting of the deep forest model gcForest-d (with only 50 trees for each MART) is still shown to be much better than the fine-tuned MART model (with up to 600 trees), which is consistent with the claim from the original paper that the default setting of deep forest can produce a highly competitive performance. Note that even with four forests in each layer, the number of trees of the deep forest is still much smaller than that used in the fine-tuned MART, leading to a great resource saving. On the other hand, a slight tuning of the deep forest model gcForest-t (by only changing the number of trees to 200) has already led to much better performance. We believe that a fine-tuning of the deep forest model will lead to a much more excellent result.

4 CONCLUSION AND FUTURE WORK

In this article, we introduce the distributed version of the deep forest model, which is developed and deployed based on the parameter server system KunPeng. To meet the need for real-world applications, many improvements are introduced, which include MART as the building block with consideration of both efficiency and effectiveness, the cost-based strategy for handling extra-imbalanced

data, MART for feature selection, and different evaluation metrics for automatically determining the layer amount. Experiments on the task of automatic detection of cash-out fraud are performed and results are analyzed from different perspectives. All results show that the deep forest model can provide a highly competitive performance, and a significant decrease in economic loss can be reached with this model even when compared with the best-deployed model.

There are still many directions which can be further explored in the future. Currently, most of the tasks that we are facing are supervised two-class classification problems, and the developed system along with the experiment in this article is mainly focused on this setting. However, other problems, such as regression and multi-class classification tasks, can be further tested to validate the effectiveness of the deep forest model under an extremely large scale. What's more, other widely studied settings, such as semi-supervised learning [8], multi-label learning [49], multi-instance learning [14], and learning under label noise [34] can also be probed. For example, many real-world problems are in multi-label or multi-instance format, while the tree-based model has been successfully adapted to these problems [10, 29, 40], and we believe that with a proper strategy, the deep forest model may be naturally employed to these problems with better performance. What's more, the robustness under label noise is a non-negligible issue in reality. Previous study [20] has shown that many tree-based methods are robust to symmetric label noise under large sample size, and we think that the deep forest model has the potential to be robust for this problem. In addition, the importance reweighting framework is shown to be helpful in the presence of label noise [34]. We will consider improving the robustness of deep forest under label noise setting with proper instance reweighting techniques for industrial tasks, and we will further validate the robustness of the deep forest model with more experiments on proper real-world tasks. Furthermore, it has been deemed as the holy grail challenge for the artificial intelligence community to combine machine learning and logical reasoning to work together [54], and compared to the combination of neural network and logic reasoning, combining the deep forest model with logic reasoning may be more convenient to leverage domain knowledge.

ACKNOWLEDGMENTS

The authors want to thank the reviewers for their helpful comments and suggestions.

REFERENCES

[1] 2016. MIT Technology Review. Retireved from https://www.technologyreview.com/s/602850/big-data-game-changer-alibabas-double-11-event-raises-the-bar-for-online-sales/.
[2] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research* 3, Jan (2003), 993–1022.
[3] Leo Breiman. 1996. Stacked regressions. *Machine Learning* 24, 1 (1996), 49–64.
[4] Leo Breiman. 2001. Random forests. *Machine Learning* 45, 1 (2001), 5–32.
[5] Christopher J. C. Burges. 2010. From Ranknet to Lambdarank to Lambdamart: An overview. *Learning* 11, 23–581 (2010), 81.
[6] Philip K. Chan, Wei Fan, Andreas L. Prodromidis, and Salvatore J. Stolfo. 1999. Distributed data mining in credit card fraud detection. *IEEE Intelligent Systems and Their Applications* 14, 6 (1999), 67–74.
[7] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)* 41, 3 (2009), 15.
[8] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. 2009. Semi-supervised learning. *IEEE Transactions on Neural Networks* 20, 3 (2009), 542–542.
[9] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 785–794.
[10] Amanda Clare and Ross D. King. 2001. Knowledge discovery in multi-label phenotype data. In *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 42–53.
[11] Manoranjan Dash and Huan Liu. 1997. Feature selection for classification. *Intelligent Data Analysis* 1, 3 (1997), 131–156.

[12] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, et al. 2010. The YouTube video recommendation system. In *Proceedings of the 4th ACM Conference on Recommender Systems*. ACM, 293–296.

[13] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, et al. 2012. Large scale distributed deep networks. In *Advances in Neural Information Processing Systems*. 1223–1231.

[14] Thomas G. Dietterich, Richard H. Lathrop, and Tomás Lozano-Pérez. 1997. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence* 89, 1–2 (1997), 31–71.

[15] Harris Drucker and Corinna Cortes. 1996. Boosting decision trees. In *Advances in Neural Information Processing Systems*. 479–485.

[16] Tom Fawcett. 2006. An introduction to ROC analysis. *Pattern Recognition Letters* 27, 8 (2006), 861–874.

[17] Jerome Friedman. 2001. Greedy function approximation: A gradient boosting machine. *Annals of Statistics* (2001), 1189–1232.

[18] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. 2000. Additive logistic regression: A statistical view of boosting. *Annals of Statistics* 28, 2 (2000), 337–407.

[19] Pierre Geurts, Damien Ernst, and Louis Wehenkel. 2006. Extremely randomized trees. *Machine Learning* 63, 1 (2006), 3–42.

[20] Aritra Ghosh, Naresh Manwani, and P. S. Sastry. 2017. On the robustness of decision tree learning under label noise. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 685–697.

[21] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. 2016. *Deep Learning*. Vol. 1. MIT Press, Cambridge.

[22] Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, and Joaquin Quiñonero Candela. 2014. Practical lessons from predicting clicks on ads at Facebook. In *Proceedings of the 8th International Workshop on Data Mining for Online Advertising*. ACM, 1–9.

[23] David W. Hosmer, Jr., Stanley Lemeshow, and Rodney X. Sturdivant. 2013. *Applied Logistic Regression*. Vol. 398. John Wiley & Sons.

[24] Nathalie Japkowicz and Shaju Stephen. 2002. The class imbalance problem: A systematic study. *Intelligent Data Analysis* 6, 5 (2002), 429–449.

[25] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*. 3149–3157.

[26] Igor Kononenko. 2001. Machine learning for medical diagnosis: History, state of the art and perspective. *Artificial Intelligence in Medicine* 23, 1 (2001), 89–109.

[27] Ludmila I. Kuncheva and Christopher J. Whitaker. 2003. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning* 51, 2 (2003), 181–207.

[28] Anísio Lacerda, Marco Cristo, Marcos André Gonçalves, Weiguo Fan, Nivio Ziviani, and Berthier Ribeiro-Neto. 2006. Learning to advertise. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 549–556.

[29] Christian Leistner, Amir Saffari, and Horst Bischof. 2010. MIForests: Multiple-instance learning with randomized trees. In *European Conference on Computer Vision*. Springer, 29–42.

[30] Mu Li, David G. Andersen, Jun Woo Park, Alexander J. Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J. Shekita, and Bor-Yiing Su. 2014. Scaling distributed machine learning with the parameter server. In *Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation*, Vol. 14. 583–598.

[31] Mu Li, Ziqi Liu, Alexander J. Smola, and Yu-Xiang Wang. 2016. Difacto: Distributed factorization machines. In *Proceedings of the 9th ACM International Conference on Web Search and Data Mining*. ACM, 377–386.

[32] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation forest. In *Proceedings of the 8th IEEE International Conference on Data Mining*. IEEE, 413–422.

[33] Jun Liu, Jianhui Chen, and Jieping Ye. 2009. Large-scale sparse logistic regression. In *Proceedings of the 15th ACM SIGKDD Iternational Conference on Knowledge Discovery and Data Mining*. ACM, 547–556.

[34] Tongliang Liu and Dacheng Tao. 2016. Classification with noisy labels by importance reweighting. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38, 3 (2016), 447–461.

[35] Xu-Ying Liu, Jianxin Wu, and Zhi-Hua Zhou. 2009. Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 39, 2 (2009), 539–550.

[36] H. Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, Sharat Chikkerur, Dan Liu, Martin Wattenberg, Arnar Mar Hrafnkelsson, Tom Boulos, and Jeremy Kubica. 2013. Ad click prediction: A view from the trenches. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1222–1230.

[37] Ming Pang, Kai-Ming Ting, Peng Zhao, and Zhi-Hua Zhou. 2018. Improving deep forest by confidence screening. In *2018 IEEE International Conference on Data Mining*. IEEE, 1194–1199.
[38] Michael J. Pazzani and Daniel Billsus. 2007. Content-based recommendation systems. In *The Adaptive Web*. Springer, 325–341.
[39] David Martin Powers. 2011. *Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness and Correlation*, Vol. 2. 37–63.
[40] Yashoteja Prabhu and Manik Varma. 2014. Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 263–272.
[41] Berthier A. Ribeiro-Neto, Marco Cristo, Paulo Braz Golgher, and Edleno Silva de Moura. 2005. Impedance coupling in content-targeted advertising. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 496–503.
[42] Tao Sun and Zhi-Hua Zhou. 2018. Structural diversity for decision tree ensemble learning. *Frontiers of Computer Science* 12, 3 (2018), 560–570.
[43] E. Ke Tang, Ponnuthurai N. Suganthan, and Xin Yao. 2006. An analysis of diversity measures. *Machine Learning* 65, 1 (2006), 247–271.
[44] Krishnaiyan Thulasiraman and Madisetti N. S. Swamy. 2011. *Graphs: Theory and Algorithms*. John Wiley & Sons.
[45] Kai Ming Ting and Ian H. Witten. 1999. Issues in stacked generalization. *Journal of Artificial Intelligence Research* 10 (1999), 271–289.
[46] David H. Wolpert. 1992. Stacked generalization. *Neural Networks* 5, 2 (1992), 241–259.
[47] Eric P. Xing, Qirong Ho, Wei Dai, Jin Kyu Kim, Jinliang Wei, Seunghak Lee, Xun Zheng, Pengtao Xie, Abhimanu Kumar, and Yaoliang Yu. 2015. Petuum: A new platform for distributed machine learning on big data. *IEEE Transactions on Big Data* 1, 2 (2015), 49–67.
[48] Zhixiang Xu, Gao Huang, Kilian Q. Weinberger, and Alice X. Zheng. 2014. Gradient boosted feature selection. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 522–531.
[49] Min-Ling Zhang and Zhi-Hua Zhou. 2014. A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering* 26, 8 (2014), 1819–1837.
[50] Ya-Lin Zhang, Longfei Li, Jun Zhou, Xiaolong Li, and Zhi-Hua Zhou. 2018. Anomaly detection with partially observed anomalies. In *Companion Proceedings of The Web Conference 2018*. International World Wide Web Conferences Steering Committee, 639–646.
[51] Jun Zhou, Qing Cui, Xiaolong Li, Peilin Zhao, Shenquan Qu, and Jun Huang. 2017. PSMART: Parameter server based multiple additive regression trees system. In *Companion Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 879–880.
[52] Jun Zhou, Xiaolong Li, Peilin Zhao, Chaochao Chen, Longfei Li, Xinxing Yang, Qing Cui, Jin Yu, Xu Chen, Yi Ding, and Yuan (Alan) Qi. 2017. KunPeng: Parameter server based distributed learning systems and its applications in Alibaba and Ant Financial. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1693–1702.
[53] Zhi-Hua Zhou. 2012. *Ensemble Methods: Foundations and Algorithms*. CRC Press.
[54] Zhi-Hua Zhou. 2019. Abductive learning: Towards bridging machine learning and logical reasoning. *Science China Information Sciences* 62, 7 (2019), 076101.
[55] Zhi-Hua Zhou and Ji Feng. 2017. Deep forest: Towards an alternative to deep neural networks. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. 3553–3559.
[56] Zhi-Hua Zhou and Ji Feng. 2019. Deep forest. *National Science Review* 6, 1 (2019), 74–86.
[57] Zhi-Hua Zhou and Xu-Ying Liu. 2006. Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Transactions on Knowledge and Data Engineering* 18, 1 (2006), 63–77.

Received February 2019; revised April 2019; accepted June 2019