An Improvement to Feature Selection of Random Forests on Spark

Ke Sun
School of Software,
Shanghai JiaoTong University,
Shanghai, 200240, China
slamke@sjtu.edu.cn

Wansheng Miao, Xin Zhang
China Aeronautical Radio
Electronics Research Institute,
Shanghai, 200233, China
miao_wansheng@careri.com

Ruonan Rao
School of Software,
Shanghai JiaoTong University,
Shanghai, 200240, China
rnrao@sjtu.edu.cn

Abstract—**The Random Forests algorithm belongs to the class of ensemble learning methods, which are common used in classification problem. In this paper, we studied the problem of adopting the Random Forests algorithm to learn raw data from real usage scenario. An improvement, which is stable, strict, high efficient, data-driven, problem independent and has no impact on algorithm performance, is proposed to investigate 2 actual issues of feature selection of the Random Forests algorithm. The first one is to eliminate noisy features, which are irrelevant to the classification. And the second one is to eliminate redundant features, which are highly relevant with other features, but useless. We implemented our improvement approach on Spark. Experiments are performed to evaluate our improvement and the results show that our approach has an ideal performance.**

Keywords—Random Forests; feature selection; feature importance; Spark

I. INTRODUCTION

This paper is primarily focused on Random Forests for feature selection. The main contribution of this paper is twofold: to propose a stable, strict, high efficient, problem independent and data-driven improvement to feature selection of Random Forests from feature importance and to provide an implementation of this strategy on Spark. The problem to solve is to eliminate the noisy and redundant features from raw data to be analyzed. The general strategy involves evaluating features using Random Forests score of importance of the feature. The strategy we propose does not depend on any pre-knowledge of the problem but depend on data-driven thresholds to make decisions.

Before stating the details, two main topics addressed in this paper are introduced shortly below.

A. Distributed Computing and Machine learning

Thousands of TBs data is produced every day. Industry and academia attach much more importance to these data, but it is impossible to use only one computer to analyze the massive data for speed and storage reasons. Traditional distributed computing techniques are very difficult for normal researchers to use. Big data occurs to solve this problem in time, which is a technique for normal researchers to conduct distributed computing and attracts great attention from industry and academia from the beginning. Various distributed computing frameworks and systems have been released and put into practice, such as MPI[1] , MapReduce[2] , GraphLab[3] and Spark[4]. In these systems, Spark is fast and easy to use for large-scale data processing, which has an advanced DAG execution engine that supports cyclic data flow and in-memory computing. Spark is suitable for iterative algorithms and interactive data mining tools.

Machine learning is a common technology that is adopted to analyze massive data to learn the implied knowledge, in which classification is the most important sub domain. Decision tree, Bayes and SVM are the three most common used classification algorithms. As accuracy requirements grow, single learning method can't satisfy researchers' requirements, more and more ensemble learning methods are introduced, such as Random Forests(abbreviated RF), bagged trees, boosted trees and boosted stumps. Rich Caruana[5] has studied the most used supervised learning methods and the result showed that RF was always one of the best algorithms to predict classification. RF has many advantages. First, over-fitting, which is a serious problem in many learning methods, is no longer to be worried about here. Second, RF can rank the features according to their importance. Third, RF has good ability to overcome the noise. Forth, RF is easy to parallel, which is very important for distributed computing.

B. Real Problems and Feature Selection

In e-commerce, medicine, transportation and other fields, scientists and engineers use multifarious machine learning algorithms to study these data in order to find the implied knowledge. For example, medical scientist may analyze the data library to predict patients' disease type. Actually, the raw data is not good enough to analyze. There is too much noise in raw data, including useless records and features. This phenomenon is inevitable. RF is an excellent heuristic algorithm of classification overcoming noise, because it takes a two-phrase randomly sampling method, which can defense a small amount of noise effectively. But too much noise may decrease RF's accuracy greatly, especially noisy features and redundant features. It is a worthwhile issue to improve RF algorithm using feature selection method to eliminate useless features.

In general, feature selection methods come in 3 types: filter, wrapper and embedded methods[6]. A filter method does not depend on any models, but only uses the characteristics of the data, while a wrapper method estimates feature importance based on a given model evaluation. The embedded method incorporates the feature selection in the model building process, such that feature importance is guiding the modeling process.

Filter methods can be used as preprocessing to remove some spurious features, but feature importance cannot be assessed since no models are built. Hence a filter method cannot estimate the impact of a feature's absence and it is not competent. Wrapper methods are constrained by the given model operated on, because such methods are just like a black box simulation. Embedded methods are beneficial since models are built with partial knowledge of feature importance. In this paper, we study this type of feature selection method.

Usually, there are two different objectives for feature selection: to select a subset of features with the minimum possible generalization error and to select a smallest possible subset with a given discrimination capability[7]. In this paper, we concentrate on the first objective.

Various feature selection methods of RF have been proposed to eliminate useless features and reduce data's dimension. The strategy in most methods is based on the quantifiable feature importance introduced by the RF models. Other strategies include counting the times of a feature chosen to build the decision trees and calculating other ratios. In this paper, we take the most used strategy. Under this strategy, the common used methods are to calculate the feature importance iteratively and eliminate one or more features of lowest importance per iteration until the size of remaining features decreasing to a given number. These methods are not strict, universal or efficient. The size of remaining features and the number of features eliminated per iteration must be given before the procedure. They are not problem independent. In this paper, we analyze the problem of using RF to learn raw data and propose our improvement to solve the problem, and evaluate our approach in the end. Our approach is a stable, strict, high efficient, problem independent and data-driven improvement to feature selection of Random Forests. During the procedure, our approach is mainly focused on eliminating the following type of features:

- Noisy features
- Redundant features

The rest of the paper is organized as follows. In Section 2, we describe some related work of the problem we studied. We then make a brief introduction to RF and analyze the problem of using RF to learning raw data and propose our improvement in Section 3. In Section 4, we make a brief introduction to Spark and state the implementation of our approach on Spark. Then we conduct experiments and evaluate our approach. Finally, a conclusion is made.

II. Related work

RF is a hot topic in machine learning domain. Much research has been done by researchers, including feature selection on RF and parallel implementation of RF. Simon Bernard[8] introduced a Sequential Forward Selection method to select sub-forests of RF to improve the performance of RF and studied the Correlation, Strength ratio of trees in forests. Pablo M.Granitto[7] adopted a recursive feature elimination method with RF for PTR-MS analysis of agroindustrial products and achieved ideal results compared to other methods. S.Stijven and W.Minnebo[9] studied the characteristics of RF and Symbolic Regression on feature selection from feature importance. J.Assuncao[10] introduced an efficient implementation of RF on MapReduce framework. H.Bostrom[11] conducted various experiments and studied the effect of the size of trees to the performance of RF. The result from [11] showed that the size of trees in forests would reduce the performance of algorithm when it was beyond a threshold. Bingguo Li[12] introduced an efficient implementation of RF on MapReduce framework with improvement of the memory overuse issue and made a comparison with others' implementation. Simon Bernard[13] also introduced a Sequential Forward Selection method and a Sequential Backward Selection to select sub-forests of RF to improve the performance of RF. B.Panda[14] introduced an efficient implementation of RF on MapReduce with "level-wise" training, which increased the speed of RF deeply.

III. Analysis and improvement

In this section, we make a brief introduction to RF firstly. Then, we analyze the two issues introduced in Section 1 separately.

A. Random Forests

RF is a popular and very efficient algorithm based on aggregation ideas, introduced by Breiman[15]. In detail, RF belongs to the family of ensemble methods, which uses randomization to produce a diverse pool of individual classifiers as for Boosting. RF can be defined as a generic principle of classifier combination that uses L tree-structured base classifiers $\{h(x, \Theta k), k = 1,\ldots, L\}$ where Θk is a family of independent identically distributed random vectors, and x is an input data. The particularity of this kind of ensemble is that each decision tree is built from a random vector of parameters. A Random Forest can be built for example by randomly sampling a feature subset for each decision tree and by randomly sampling a training data subset for each decision tree[8].

The performance of a RF model related to the quality of each tree. Generally, out-of-bag (abbreviated OOB) samples are used to quantify the quality of individual tree. The OOB samples are the subset of dataset that are not used for building the current tree. Moreover, OOB samples are used to estimate the prediction error and then to evaluate feature importance. In this paper, we will concentrate on the prediction accuracy of RF focusing on OOB error rate.

The quantification of the feature importance (abbreviated FI) is an important issue in many applied problems complementing feature selection by interpretation issues[8]. In RF framework, the most widely used score of importance of a given feature is the increasing in mean of the misclassification rate of a tree in the forest when the corresponding values of this features are randomly permuted in the OOB samples. More important features give more increasing in mean of the misclassification rate.

Almost all of the common used machine learning libraries have their implementation of RF, such as mahout, R and GraphLab. Unfortunately, MLlib has not given its RF so far, which is the machine learning library on Spark.

B. Noisy Features

Noisy features may be drawn for all kinds of reasons, but they have one common aspect, which is low contribution to algorithm's accuracy and low importance. Feature importance (abbreviated FI) is a crucial issue for ranking features and RF has the feature to compute the FI. Taking FI to make feature selection is a feasible approach. In the following of this section, we will propose a three-phrase method to make feature selection:

Phrase 1: Preliminary feature elimination of small importance:

- T-test for binary classification and ANOVA for multi-class classification.

Phrase 2: Feature selection using RFE-RF:

- Recursive feature elimination with RF according to Gaussian distribution;
- Order the remaining features in decreasing order of average of FI;

Phrase 3: Optimal sub-RF selection using SFS:

- Sequential Forward Selection to select the optimal sub-RF.

First, a preliminary feature selection is conducted for the fact that there may be some features having no concern with the classification problem which is analyzed. For binary classification problem, T-test is used and for multi-class classification problem, Analysis of variance (abbreviated ANOVA) is used. The *p-value* can be set to a rational value to take the pre-selection, for example 0.05.

Second, recursive feature elimination (abbreviated RFE) with RF is conducted. FI is a crucial issue for ranking features and RF has the feature to compute the FI, but FI from a randomly built forests cannot be used directly to make feature selection for the randomicity of RF. Significant features may have a lower FI in one time occasionally and it is incorrect to eliminate these features. In order to avoid eliminating important features, the threshold can be set to a very low value. But a low threshold cannot guarantee full elimination of noisy features. However, the most unimportant features may always have the lowest FI. Then it is necessary to build the forests, calculate FI and eliminate features recursively.

Recursive feature elimination algorithm is introduced by Guyon[16] which provides good performance with moderate computation efforts and the most popular method using RFE is using a linear Support Vector Machine (abbreviated SVM-RFE) to eliminate features. Pablo M.Granitto[7] used RF with RFE (abbreviated RF-RFE) for PTR-MS analysis of agroindustrial products and made comparison with SVM-RFE, which showed that RF-RFE outperformed SVM-RFE.

A rational threshold is important when using RFE to eliminate features. Some researchers take the approach to eliminate only one feature once iteration until the size of features reduced to a given constant, but the constant is hard to be determined. Actually, when analyzing huge amounts of data from real research, we can make a hypothesis that the FI of the feature fits Gaussian distribution[17], whose notation is $N(\mu,\sigma 2)$. The μ value is the mean or expectation of the distribution and the σ value is its standard deviation. About 95% of the values lie within two standard deviations.

$$P(\mu-2\sigma\leq X\leq\mu+2\sigma)=95.4\% \qquad (1)$$

Thus, we take the method to eliminate the feature whose FI is lower than μ-2σ once iteration until no features to be eliminated. Moreover, the threshold can be set to μ-$n*\sigma$ according to the real problem. After this step, irrelevant features can be eliminated, but there may also be some remaining irrelevant features. The third phrase is conducted next. Before the third phrase, the remaining features are ordered in decreasing order of average FI of past iterations. If the iteration is too small, more runs are executed to compute a rational average of FI.

Third, Sequential Forward Selection (abbreviated SFS) is taken on the remaining features to generate an optimal sub-RF, which has the lowest OOB error rate. SFS was introduced on RF by Bernard[8], which showed that the accuracy of RF reached the optimal point only with a smaller subset of forests. We build the nested models starting from the one with only the most important feature and ending with the one involving all the important features in the sequence from Phrase 2. The model leading to the smallest OOB error is selected and the features in the model are selected.

It is expected that the OOB error is non-decreasing as soon as all the "true" features have been selected. Thus, the OOB error may be nearly constant or slightly increasing during the SFS stage. So we can set a threshold to stop the Phrase 3 for pruning when the OOB error is not increasing beyond it for several times continuously.

C. Redundant features

It is common that raw data may include some redundant features, which are highly relevant with other features in data set. For example, birthday and age features in some problems. The redundant features are nearly useless and even jeopardize the accuracy of RF. Sometimes, the researcher may only want to study the real "true" features without redundant ones. Feature selection to eliminate redundant features is also a worthwhile issue.

After noisy features are eliminated from raw data, the accuracy of RF will be increased obviously and we can get a subset of features, denoted by S. A feature is not redundant only if the error gain exceeds a threshold. In this subset of features, we perform a sequential features evaluation. The reason is that the OOB error rate decrease must be significantly greater than the average variation obtained by adding noisy features.

The procedure is that we build a sequence of RF models each of which is eliminating only one feature in S. The OOB error rates are calculated and the OOB error gains are also calculated for every feature. In the subsection above, the optimal OOB error rate is computed and the optimal subset of features are got. The remaining features out of optimal subset after pre-selection are regarded as noisy features, denoted by *N*. SFS procedure can be performed from the optimal model to the model with all the features after pre-selection. During the procedure, the average variation of OOB error rate gains, denoted by *t*, can be calculated using the following formula:

$$t = \frac{1}{|\mathrm{N}|}\sum_{k=1}^{|N|}|\mathrm{OOBerr(k+1)} - \mathrm{OOBerr(k)}| \qquad (2)$$

In order to get a precise result, a cross validation can be conducted.

IV. Implementation

After our analysis, our approaches were implemented on Spark. In this section, a brief introduction to Spark is made firstly and then the implementation details will be stated next.

A. Spark

Spark is a fast and general cluster distributed computing system for Big Data. It is easy to use, which provides high-level APIs in Scala, Java, and Python. It has an optimized engine that supports general computation graphs for data analysis. It also supports a rich set of higher-level tools including Spark SQL for SQL and structured data processing, MLLib for machine learning, GraphX for graph processing, and Spark Streaming for streaming computing.

Spark introduces Resilient Distributed Datasets (abbreviated RDDs), a distributed memory abstraction that lets programmers perform in-memory computations on large clusters in a fault-tolerant manner. Thus, Spark is suitable for iterative algorithms and interactive data mining tools. To achieve fault tolerance efficiently, RDDs provide a restricted form of shared memory, based on coarse-grained transformations rather than fine-grained updates to shared state.

RDD, the key to Spark' high speed, is a read-only, partitioned collection of records. It can only be created through deterministic operations on either data in stable storage, such as local disk and HDFS, or from other RDDs. An RDD has enough information about how it was derived from other datasets (its lineage) to compute its partitions from data in stable storage. Thus, there is no need to materialize RDDs at all times. This is a powerful property: in essence, a program cannot reference an RDD that it cannot reconstruct after a failure.

There are 2 kind of operations allowed on RDDs, which are transformation and action. Programmers start by defining one or more RDDs through transformations, such as map and filter, on data in stable storage. After RDDs are created, they can be used in actions, which are operations that return a value to the application or export data to a storage system. In addition, programmers can call a persist method to indicate which RDDs they want to reuse in future operations, which can speed up the computation greatly.

It is easy to use Spark. Developers write a driver program that connects to a cluster of workers. The driver defines one or more RDDs and invokes actions on them. Spark code on the driver also tracks the RDDs' lineage. The workers are long-lived processes that can store RDD partitions in RAM across operations and do distributed computations. A worker is an executor, which does the real task and the driver controls the workers. But commutations between driver and workers, job schedule and failure handling are handled by Spark, which frees developers to true concerns.

B. Implementation Detail

- Overall

In our forests, CART decision tree[18] is chosen. Both continuous and categorical features are supported. Gini[18] and Entropy[18] are 2 common coefficients to measure the node impurity of the homogeneity of the labels. And both have their merits and drawbacks. Thus, we implemented both of them.

The driver passes random seeds to different trees and the executors of trees take the seeds to randomly sample features to speed up.

- Training

Training the RF model is the most important stage. Two common methods have been introduced before. First, independent trees are built per machine only using local data. Second, distributed trees are built over all dataset. Each node builds a subset of forests in the first method. It is fast for parallel, but accuracy decrease is inevitable. The second method can bring an exact solution with no accuracy decrease, but its more data transfers bring low speed.

In our implementation, a "level-wise" strategy is adopted to reduce total data transfers, which is similar to Google's PLANET[14] and adopted by MLlib. The top few levels of trees see overall dataset and the deep levels see only local data. Fewer and fewer training records are seen as tree level is descendant. Once the number of training samples get small enough, for example smaller than a threshold, sub-trees are trained locally by shuffling sub-tree training samples to matching executors and cache the working set only.

- Prediction

Every single tree votes to predict when using RF to make classification. The common method is that every single tree has equivalent weight and the majority wins in the end. Sometimes, this method cannot obtain a good result for some low-quality trees' interference. However, weighted vote strategy is also introduced in our implementation, which takes OOB error as a tree's weight.

V. Evaluation

We experimented our implementation using 6 high dimensional classification datasets from UCI[19] and analyzed the results finally. The datasets are from 6 different fields. Accuracy and scalability are 2 main aspects that will be studied. During the experiments, each dataset is divided into a training subset and a testing subset, with respectively 2/3 of the samples used for training and the other 1/3 used for testing. The number of trees in the forest, donated by *mtree*, is fixed to 200. And the sample proportion is 66% for each tree. The number of features randomly chosen, donated by *mfeature* is fixed to$\sqrt{m}$, which is a default value common used in literature and *m* is the size of all features. All single CART trees are trained to a maximum depth of 30.

A. *Accuracy*

Experiments are conducted on a cluster with 10 slaves (every node with 4GB memory). Before the experiments, *0.5*m* of noisy features and *0.2*m* of redundant features were added to original datasets to make a rational evaluation. Two other algorithms are also adopted in the experiments to make a comprehensive comparison, which are Decision Tree of MLlib and pure RF algorithm. Table I shows the accuracy of Decision Tree of MLlib, pure RF and our approach(abbreviated IRF) on all the 6 datasets. Table II shows the size of features selected by our approach and the size of features in original dataset and how many useless features are eliminated from original dataset and adjunction.

TABLE I. ACCURACY OF 3 DIFFERENT MODELS

Dataset	**Classification model**		
	Decision Tree	***pure RF***	***IRF***
1	51.34%	55.34%	59.34%
2	83.79%	85.47%	90.32%
3	73.22%	78.44%	81.72%
4	85.41%	87.21%	88.21%
5	64.91%	69.01%	68.01%
6	68.62%	72.41%	78.32%

TABLE II. THE SIZE OF FEATURES OF CLASSIFICATION PROCEDURE

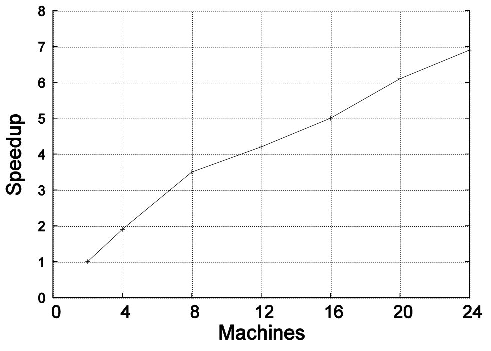
Dataset	**Size of Features**			
	Original	***Adjunction***	***Selected***	***Eliminated***
1	617	431	213	835
2	128	89	67	150
3	56	39	33	62
4	12	8	7	13
5	24	16	11	29
6	27	18	23	22

The result shows that our approach gives an excellent performance on classification. Our approach improves RF's accuracy and it has an ideal ability to select features. It is a stable, strict, high efficient and data-driven improvement to feature selection of Random Forests.

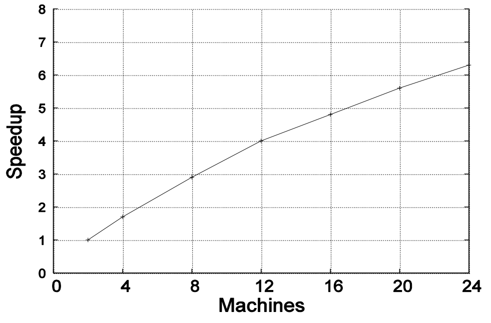
B. *Scalability*

Experiments are conducted from a cluster with 2 slaves (every node with 4GB memory) to a cluster with 24 slaves of the same variety. Datasets range from 10K to 100M points in size. Fig. 1 shows the speedup of our experiments of all the 6 datasets. In the figures blow, the x-axis is the number of machines and the y-axis is the speedup.

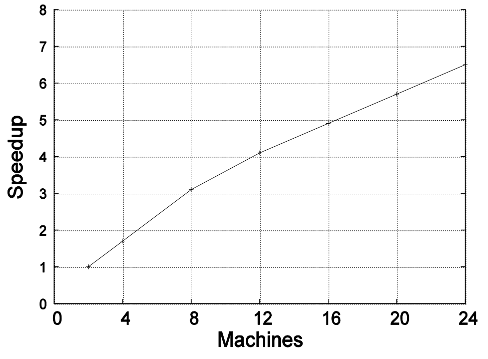
The results from Fig. 1 show that our implementation is scalable, practical and the speedup is acceptable. Our method is a practical method on Spark.



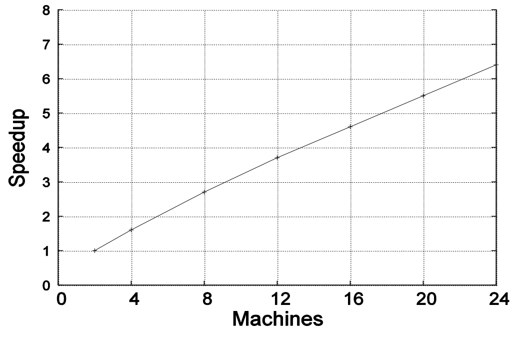
a. Dataset1



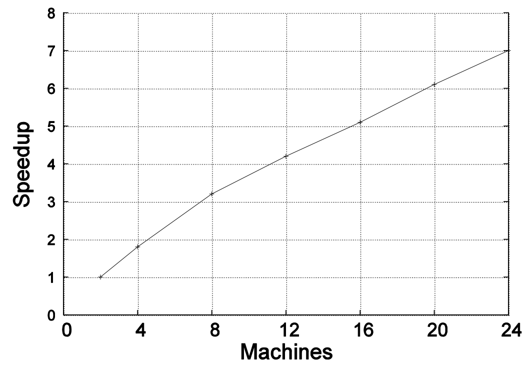
b. Dataset2



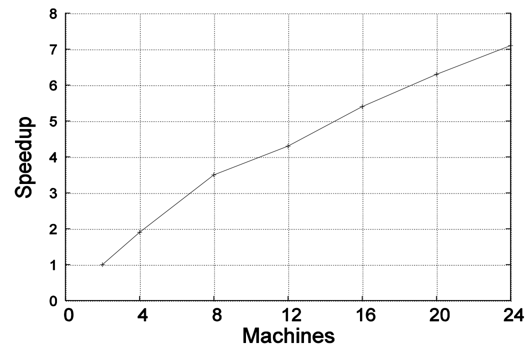
c. Dataset 3



d. Dataset 4



e. Dataset 5



f. Dataset 6

Fig. 1. The Speedup of IRF

VI. Conclusions

In this paper, we have introduced our improvement to feature selection of Random Forests from feature importance. Our approach is a stable, strict, high efficient, problem independent and data-driven improvement compared with other methods. Two common problems from real usage scenario are studied and solved by our approach, which are noisy features and redundant features. For noisy features, a three-phrase method has been proposed, including preliminary feature elimination, feature selection using RFE-RF and optimal sub-RF selection using SFS. For redundant features, a traversal method has been taken to eliminate redundant features. Our methods are based on rational mathematical theories in order to avoid random errors. We implemented our approach on Spark and made an evaluation using datasets from UCI. The evaluation showed that our approach was excellent and had an ideal performance.

We will further investigate the strategy to simplify the procedure of feature selection of Random Forests from feature importance and improve our implementation on Spark with much higher speed. Moreover, we will adopt our approach in real usage scenario to evaluate it.

Acknowledgment

The authors wish to thank the colleagues from Software Department of China National Aeronautical Radio Electronics Research Institute and the colleagues from distributed computing lab, school of software, Shanghai JiaoTong University. This work was supported in part by a grant from Software Department of China National Aeronautical Radio Electronics Research Institute.

References

[1] Open MPI. http://www.open-mpi.org/.

[2] Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters[J]. Communications of the ACM, 2008, 51(1): 107-113.

[3] Low Y, Gonzalez J, Kyrola A, et al. Graphlab: A new framework for parallel machine learning[J]. arXiv preprint arXiv:1006.4990, 2010.

[4] Zaharia M, Chowdhury M, Das T, et al. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing[C]//Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation. USENIX Association, 2012: 2-2.

[5] Caruana R, Niculescu-Mizil A. An empirical comparison of supervised learning algorithms[C]//Proceedings of the 23rd international conference on Machine learning. ACM, 2006: 161-168.

[6] Guyon I, Elisseeff A. An introduction to variable and feature selection[J]. The Journal of Machine Learning Research, 2003, 3: 1157-1182.

[7] Granitto P M, Furlanello C, Biasioli F, et al. Recursive feature elimination with random forest for PTR-MS analysis of agroindustrial products[J]. Chemometrics and Intelligent Laboratory Systems, 2006, 83(2): 83-90.

[8] Bernard S, Heutte L, Adam S. Towards a better understanding of random forests through the study of strength and correlation[M]//Emerging Intelligent Computing Technology and Applications. With Aspects of Artificial Intelligence. Springer Berlin Heidelberg, 2009: 536-545.

[9] Stijven S, Minnebo W, Vladislavleva K. Separating the wheat from the chaff: on feature selection and feature importance in regression random forests and symbolic regression[C]//Proceedings of the 13th annual conference companion on Genetic and evolutionary computation. ACM, 2011: 623-630.

[10] Assunçao J, Fernandes P, Lopes L, et al. Distributed Stochastic Aware Random Forests--Efficient Data Mining for Big Data[C]//Big Data (BigData Congress), 2013 IEEE International Congress on. IEEE, 2013: 425-426.

[11] Boström H. Concurrent Learning of Large-Scale Random Forests[C]//SCAI. 2011: 20-29.

[12] Li B, Chen X, Li M J, et al. Scalable random forests for massive data[M]//Advances in Knowledge Discovery and Data Mining. Springer Berlin Heidelberg, 2012: 135-146.

[13] Bernard S, Heutte L, Adam S. On the selection of decision trees in random forests[C]//Neural Networks, 2009. IJCNN 2009. International Joint Conference on. IEEE, 2009: 302-307.

[14] Panda B, Herbach J S, Basu S, et al. Planet: massively parallel learning of tree ensembles with mapreduce[J]. Proceedings of the VLDB Endowment, 2009, 2(2): 1426-1437.

[15] Breiman L. Random forests[J]. Machine learning, 2001, 45(1): 5-32.

[16] Guyon I, Weston J, Barnhill S, et al. Gene selection for cancer classification using support vector machines[J]. Machine learning, 2002, 46(1-3): 389-422.

[17] Gaussian-distribution. http://en.wikipedia.org/wiki/Normal_distribution.

[18] CART.http://www.statsoft.com/Textbook/Classification-and-Regression-Trees.

[19] UCI.https://archive.ics.uci.edu/ml/datasets.html.