Potential Off-Grid User Prediction System Based on Spark

LI Xuebing[1,3], SUN Ying[1,2], ZHUANG Fuzhen[1,2], HE Jia[1,2], ZHANG Zhao[1,2], ZHU Shijun[4], and HE Qing[1,2]

(1. Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China;
2. University of Chinese Academy of Sciences, Beijing 100049, China;
3. College of Information Science and Engineering, Yanshan University, Qinhuangdao, Hebei 066004, China;
4. ZTE Corporation, Shenzhen, Guangdong 518057, China)

Abstract: With the increasingly fierce competition among communication operators, it is more and more important to make an accurate prediction of potential off-grid users. To solve the above problem, it is inevitable to consider the effectiveness of learning algorithms, the efficiency of data processing, and other factors. Therefore, in this paper, we, from the practical application point of view, propose a potential customer off-grid prediction system based on Spark, including data pre-processing, feature selection, model building, and effective display. Furthermore, in the research of off-grid system, we use the Spark parallel framework to improve the gcForest algorithm which is a novel decision tree ensemble approach. The new parallel gcForest algorithm can be used to solve practical problems, such as the off-grid prediction problem. Experiments on two real-world datasets demonstrate that the proposed prediction system can handle large-scale data for the off-grid user prediction problem and the proposed parallel gcForest can achieve satisfying performance.

Keywords: data mining; off-grid prediction; Spark; parallel computing; deep forest

DOI: 10.12142/ZTECOM.201902005

http://kns.cnki.net/kcms/detail/34.1294.TN.20190621.1839.002.html, published online June 21, 2019

Manuscript received: 2018-03-13

1 Background

In recent years, the competition among communication operators is fiercer with the advent of the 4G era. The communication operators face new challenges in many ways, including fine management, precision marketing, products innovation, promotion of user terminals, retaining users, data analysis, and so on. How to win the competition? It plays a significant role that doing your best to retain old users while attracting new users through improving service quality and others. Therefore, it is an urgent need to solve the problem that the operators can make a good prediction of potential off-grid users.

This work is supported by ZTE Industry-Academia-Research Cooperation, the National Key Research and Development Program of China under Grant No. 2017YFB1002104, the National Natural Science Foundation of China under Grant Nos. U1836206, U1811461, and 61773361, and the Project of Youth Innovation Promotion Association CAS under Grant No. 2017146.

The past decade has witnessed the remarkable growth of Internet communication technology and sensor networks for perceiving and obtaining information. There is a huge amount of data which has great potential values in all walks of life, including the telecommunications industry. Big data is high-volume, high-velocity, and high-variety information assets that require new forms of processing to enable enhanced decision making, insight discovery, and process optimization [1]. The potential values of big data need advanced data mining techniques [2], [3] to discover and they have drawn much attention from the practitioners. Now there are many good algorithms to achieve latent useful information such as neural networks, cluster analysis, genetic algorithms, decision trees, decision rules, support vector machines, and more. Nevertheless, conventional approaches come across significant challenges when computing power and memory space are limited in big data era. So distributed technology is necessary to solve the problem of limited computing power and memory space. Apache Hadoop, which is an open-source software framework used for distributed stor-

age and processing of dataset of big data based on MapReduce programming model [4]–[7], becomes a good framework to solve the above problem. The main idea of the MapReduce model is to hide details of parallel execution and allow users to focus only on data processing strategies. MapReduce has brought revolution in computing model, and also provides a new processing platform for data mining environment. However, Hadoop processes data from disk which makes it inefficient for data mining applications that often require iteration. Spark is a more recent distributed framework that works with Hadoop and provides in-memory computation that allows iterative jobs to be processed much faster, so it is a more suitable base for data mining [8].

Previous works mainly focus on analyzing off-grid user problem from the angle of algorithms. GUI et al. studied the main technology of customer loss prediction and analyzed the advantages and disadvantages of the decision tree, neural network, and so on. Louis [9], Rosset [10] and Nash [11] have studied customer churn prediction as a classification problem and used Logistic Regression, Decision Tree and Bayes to establish a customer churn prediction model. However, the ultimate aim of algorithm research is to solve the practical problem. In this paper, we propose a complete solution—a potential customer off-grid prediction system based on Spark—to solve the problem. And the data mining techniques include data storage, data clearing, feature selection, training model, and result showing. Furthermore, we use the Spark parallel framework to improve the gcForest algorithm which is a state-of-the-art decision tree ensemble approach. The new parallel gcForest algorithm can be used to solve off-grid prediction problem. Moreover, the parallel gcForest can be used to solve more practical problems.

The rest of this paper is organized as follows. In section 2, we introduce techniques related to our proposal. In sections 3 and 4, we introduce the system architecture and parallel gcForest in detail, which are both under the Spark computational framework. In section 5, we analyze experimental results. Section 6 concludes this paper.

2 Related Technologies

In this section, we will introduce the mostly related techniques of prediction system, involving Spark computing framework, Hadoop Distributed File System (HDFS), MapReduce, and Hive.

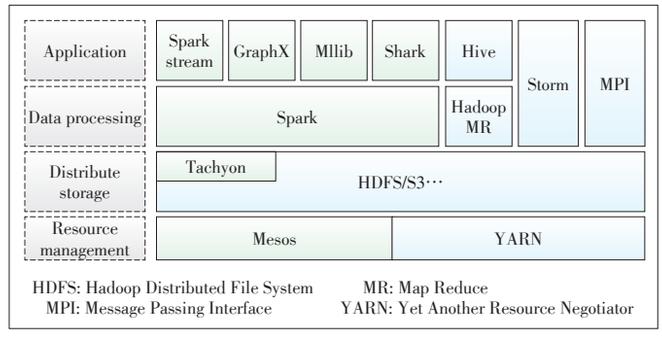
2.1 Apache Spark

Apache Spark is an open-source cluster computing framework, which runs programs in memory, developed at the University of California, Berkeley's AMPLab. As a MapReduce-like cluster computing engine, Spark also possesses good characteristics such as scalability and fault tolerance as MapReduce does. However, Apache Spark has its architectural foundation the resilient distributed dataset (RDD). This is a read-only multiset of data items distributed over a cluster of machines and users can explicitly cache an RDD in memory across machines and reuse it in multiple MapReduce-like parallel operations. Therefore, Spark can be well qualified to process iterative jobs, including PageRank algorithm, K-means algorithm, etc. Besides, it can outperform Hadoop by 10x in iterative machine learning jobs. Spark has strong fault-tolerant performance through a notion of lineage: If a partition of an RDD is lost, the RDD has enough information about how it was derived from other RDDs to be able to rebuild just that partition. With the help of the lineage, Spark recovers the lost data quickly and effectively. Spark shows great performance in processing iterative computation because it can reuse intermediate results and keep data in memory across multiple parallel operations.

Currently, Spark has devolved into a large data processing platform based on memory computing, and its architecture includes resource management, distributed storage, data processing, and application. Berkeley called the entire ecosystem of Spark as the Berkeley data analysis stack (BDAS). For resource management, Spark supports standalone (native Spark cluster), Hadoop YARN, or Apache Mesos. For distributed storage, Spark can interface with a wide variety, including Hadoop Distributed File System (HDFS), MapR File System (MapR-FS), Cassandra, and more. Furthermore, Spark has a distributed memory file system, Tachyon, which is used to cache data in memory more than 100 times faster than HDFS. Data processing engine is the Spark computing framework based RDDs, which are the foundation of the overall project. On the top of Spark, there are many distributed subprojects, such as Spark Streaming, GraphX, Spark Structured Query Language (SQL), MLlib, and more. In addition, as shown in **Fig. 1**, there is a very important advantage that Spark ecosystem is compatible with the Hadoop ecosystem.

2.2 Hadoop Distributed File System

HDFS is the system component of Hadoop, which is designed to store very large data sets reliably and to stream those data sets at high bandwidth to user applications [12]. Comparing with other existing distributed file systems, HDFS has



▲**Figure 1. Spark Ecosystem.**

some advantages as follow. On the one hand, HDFS architecture ensures that it could detect fault and automatically recover quickly from those faults. Therefore, HDFS is highly fault-tolerant. Moreover, it can be deployed on low-cost hardware. On the other hand, HDFS provides high throughput access to application data and is suitable for applications that have large datasets.

2.3 MapReduce

MapReduce [13] is such a programming model that provides an associated implementation for processing and generating large-scale data sets using Map( ) and Reduce( ) procedures. Specifically, users design a map function that takes a set of data and converts it into another set of data, where individual elements are broken down into <key, value> pairs and a reduce function takes the output from a map as input and combines those data <key, value> pairs into a smaller set of <key, value> pairs. Generally, the reduce job is performed after the map job.

The major advantage of MapReduce is that it processes and generates big data sets with a distributed, parallel algorithm on a computer cluster. We use a typical example to promote the understanding of MapRecuce idea. Consider the following pseudo-code for a task that counts the appearance of each word in a set of documents:

```
map(String key, String value):
// key: document name
// value: document contents
for each word w in value: output(w, 1);

reduce(String key, Iterator values):
// key: a word
// values: a list of counts
int sum count = 0;
for each v in values:
sum count += v;
```

Here, the map function counts each word which appears in the document. And the reduce function sums sum all of its input values get the total counts of a single word.

2.4 Hive

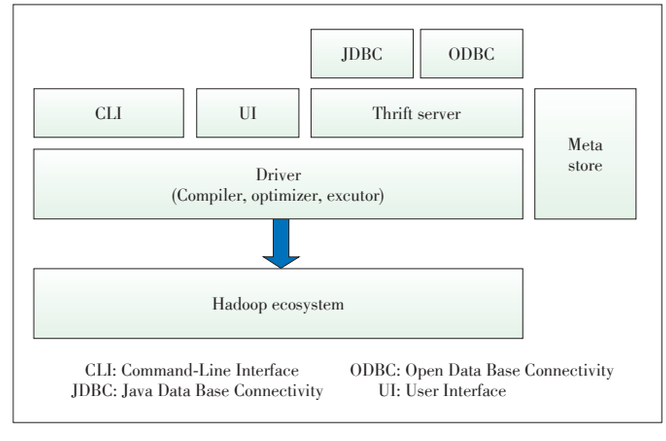
Hive is a data warehouse software project to facilitate reading, writing, and managing of large datasets in distributed storage using SQL and it builds on Hadoop. In general, data query, statistics, and other data operations on distributed data need to be implemented in the MapReduce Java Application Programming Interface (API) to execute SQL applications. This way is very inconvenient. To solve the problem, Hive provides the necessary SQL abstraction to integrate SQL-like queries into the underlying Java without the need to implement queries in the low-level Java API. Hive provides a SQL-like interface to handle data stored in a distributed file system that integrates with Hadoop. Since most data warehousing applications work with SQL-based querying languages, Hive defines SQL-like query language, called HQL (Hibernate Query Language), which allows users who are familiar with SQL to query data on Hadoop.

Hive supports the analysis of large-scale data stored in the HDFS file system. It provides HQL and converts queries to MapReduce and Spark jobs which can run in Hadoop YARN. Major components of the Hive architecture are shown in **Fig. 2**. Metastore stores metadata for each of the tables such as the name of the table, the partition of the table, and its properties. Metadata is highly crucial because it can help the driver to track the data. A driver plays a role of receiving the HQL statements, which likes a controller. And it creates sessions to start the execution of statement and monitors the progress of the execution. The compiler, optimizer, and executor complete HiveQL query statements from lexical analysis, syntax analysis, compilation, optimization, and query plan generation. The generated query plan is stored in the HDFS and then executed. The command-line interface (CLI), user interface (UI), and thrift server are user interfaces which mainly use for user interaction. Hive is simple to learn and use. It can quickly implement MapReduce statistics through SQL-like language and is very suitable for statistical analysis for the big data which stores distributed system.

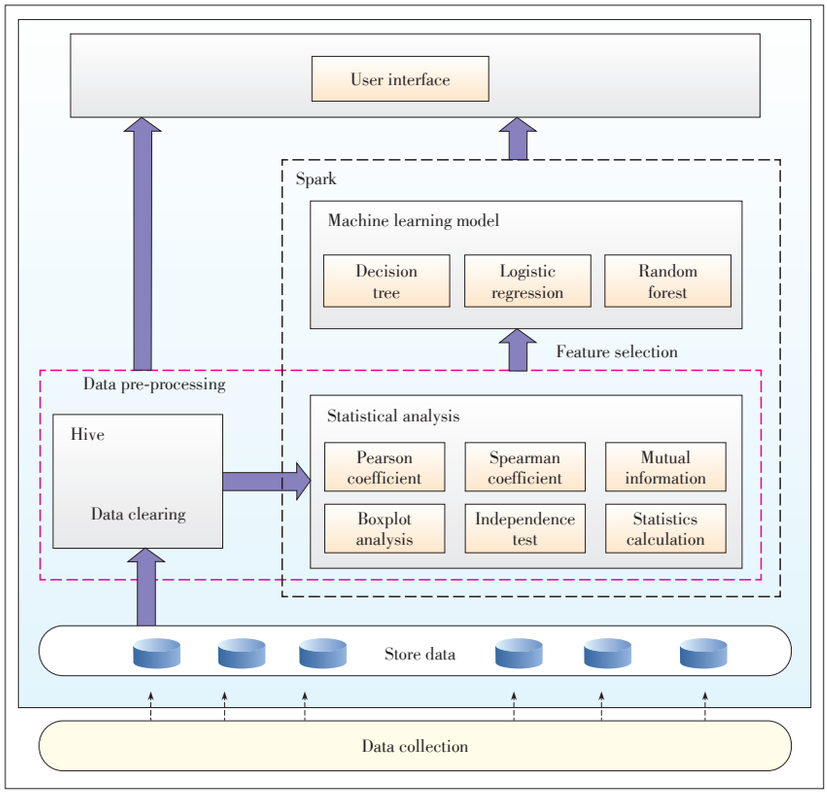
3 System Architecture

We present a complete solution to deal with the problem of off-grid user prediction in this paper, named the potential off-grid user prediction system. The whole system is built on Spark, a distributed computing framework and can be applied to large-scale data. The proposed system architecture is shown in **Fig. 3**.

In Fig. 3, the system includes four parts, i.e., data storage, data pre-processing, statistical analysis, training model and user interface. In the following subsections, we will detail the



▲**Figure 2. Hive architecture.**



▲Figure 3. Architecture of the potential off-grid user prediction system based on Spark.

structure of the whole system, including the data processing process, the algorithm, and the specific design of the system.

3.1 Data Pre-Processing

The data pre-processing plays an important role in data mining. The phrase "garbage in, garbage out" are particularly applicable to data mining and machine learning and also prove that data quality is very important for achieving good performance. If there is much irrelevant and redundant information present or noisy and unreliable, knowledge discovery during the training phase will be more difficult. In our system, the pre-processing of operation is divided into two parts, including data cleaning and statistical analysis with the purpose of feature selection.

3.1.1 Data Cleaning

Data cleaning[1] is the process of detecting and correcting (or removing) corrupted or inaccurate records from a database, involving incomplete, incorrect, inaccurate or unrelated parts of the data, and then replacing, modifying, or deleting the dirty or course data. In the system, we have the following ways to do simple data cleaning.

- Removing obvious useless feature. There are a few useless features in the user data provided by the operator, which seriously affect the final result performance. So we need to find and get rid of them. For example, a feature that has a large number of null values or the same values is an obvious useless feature.
- Data transformation. This function allows the mapping of the data from its given format into the format expected by the appropriate application. This is important for the training model. For instance, for the non-vector feature in the data, we need to convert it to a vector feature that can be computed. In addition, there are some features that need to be normalized to enhance the performance of training.
- Labelling data. When the system reads data from the HDFS and saves it to the hive, the data is not labeled. This function is to label the data according to the rules which is essential to the training of off-grid models.

Based on Hive, we can also do a lot of operations of processing data. The function-based Hive can continue to expand in our system.

3.1.2 Statistical Analysis

In our system, the major purpose of statistical analysis is to make feature selection better. The simple data clearing in the previous step has cleared the obvious useless features, and this step is further feature selection. As we all know, feature selection, or called variable selection, is the process of selecting a subset of relevant features (variables, predictors) for use in model construction. There are four reasons to do feature selection:

- Simplification of models to make them easier to interpret by researchers
- Reducing training times
- Removing redundancy and irrelevant features to avoid the curse of dimensionality
- Enhancing generalization by reducing overfitting and make a better model.

The analytical methods used in the system are divided into two categories: the correlation analysis of features and data distribution analysis. The correlation analysis of features is very important in the process of data pre-processing. The main objective of feature analysis is to eliminate extraneous and redundant features without losing important information. After acquiring the data in the machine learning task, it is usually necessary to do correlation analysis of feature and select feature correlation. There are four correlation analysis methods we used, involving the pearson correlation coefficient, spearman's rank correlation coefficient, mutual information and chi-squared test.

(1) Pearson correlation coefficient:

[1] https://en.wikipedia.org/wiki/Data cleansing

In statistics, the pearson correlation coefficient is a measure of the linear correlation between two variables X and Y by a value between 1 and -1. For example, 1 is total positive linear correlation, 0 is no linear correlation, and -1 is total negative linear correlation. The definition of pearson correlation coefficient is the covariance of the two variables. And it is always represented by ρ when applied to a population. The formula for ρ is Eq. (1).

$$\rho(X,Y)=\frac{\operatorname{cov}(X,Y)}{\sigma_X\sigma_Y}, \tag{1}$$

where cov is the covariance; σ_X is the standard deviation of X and σ_Y is the standard deviation of Y. When applied to a sample, it is commonly represented by the letter r and may be called the sample correlation coefficient. We can obtain a formula for r by substituting estimates of the co-variances and variances based on a sample into the formula above. So if we have one dataset $\{x_1,\cdots,x_n\}$ containing n values and another dataset $\{y_1,\cdots,y_n\}$ containing n values, that formula for r is:

$$r=\frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i-\bar{y})^2}}, \tag{2}$$

where n is the sample size; x_i and y_i are the single samples indexed with i; $\bar{x}=(1/n)\sum_{i=1}^{n}x_i$, which is used to calculate the sample mean. The way to calculate $\bar{y}$ is similar.

(2) Spearman's rank correlation coefficient:

In statistics, Spearman's rank correlation coefficient is a nonparametric measure of rank correlation, i.e., the statistical dependence between the rankings of two variables. It assesses how the relationship between two variables is described by monotone functions. The difference of pearson correlation and spearman correlation is that the former assesses linear relationships, while the later assesses monotonic relationships [14].

Spearman's rank correlation coefficient often be denoted as r_s. If we have two variables whose sizes are n and X_i,Y_i are their raw scores, X_i,Y_i are converted to ranks rgX_i,rgY_i. The formula for r_s is Eq. (3).

$$r_s=\frac{\operatorname{cov}(rg_X,rg_y)}{\sigma_{rg_X},\sigma_{rg_y}}, \tag{3}$$

where $\operatorname{cov}(rg_X,rg_y)$ is the covariance of the rank variables and $\sigma_{rg_X},\sigma_{rg_y}$ are the standard deviations of the rank variables.

(3) Mutual information:

In probability theory and information theory, the mutual information of two random variables is a measure of the mutual dependence between the two variables. It is one of many quantities that measures how much one random variable tells us about another. It is a dimensionless quantity with units of bits and can be thought of as the reduction in uncertainty about one random variable given knowledge of another. High mutual information indicates a large reduction in uncertainty; low mutual information indicates a small reduction; and zero mutual information between two random variables means the variables are independent. Formally, the mutual information of two discrete random variables X and Y can be defined as Eq. (4).

$$I(X;Y)=\sum_{y\in Y}\sum_{x\in X}p(x,y)\log\left(\frac{p(x,y)}{p(x)p(y)}\right), \tag{4}$$

where $p(x,y)$ is the joint probability function of X and Y; $p(x)$ and $p(y)$ are the marginal probability distribution functions of X and Y respectively. In the case of continuous random variables, the summation is replaced by a definite double integral:

$$I(X;Y)=\int_Y\int_X p(x,y)\log\left(\frac{p(x,y)}{p(x)p(y)}\right)dx\,dy, \tag{5}$$

where $p(x,y)$ is now the joint probability density function of X and Y, and $p(x)$ and $p(y)$ are the marginal probability density functions of X and Y respectively. If the log base 2 is used, the units of mutual information are bits.

(4) Chi-squared test:

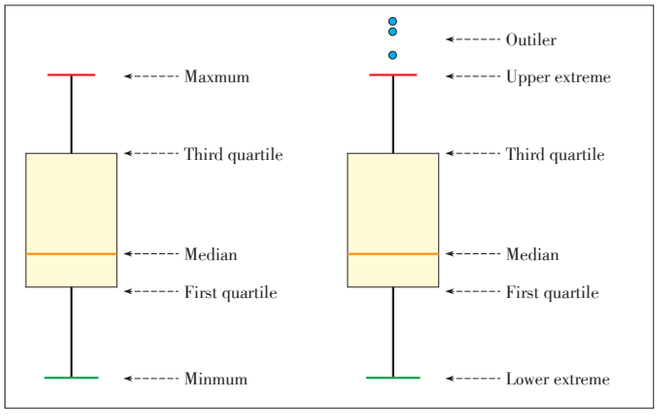
The chi-squared test is a common hypothesis testing method based on the χ^2 distribution. It is the deviation between the actual observed value and the theoretical inference value of the statistical sample. The deviation between the actual observation value and the theoretical inference value determines the size of the chi square value. The larger the chi square value is, the more it does not conform; the smaller the value of the value, the smaller the deviation, the more consistent. If two values are exactly equal, the chi square value is 0, and it indicates that the theoretical value is in full conformity.

As we mentioned above, the analytical methods for feature selection are divided into two categories: correlation analysis of features and data distribution analysis. In our system, we mainly use box plot to do data distribution analysis.

(5) Box plot:

The box plot is a method for graphically depicting groups of numerical data through their quartiles. Box plots are useful for identifying outliers and for comparing distributions. Box plots may also have lines extending vertically from the boxes indicating variability outside the upper and lower quartiles, hence the terms box-and-whisker plot. Outliers may be plotted as individual points which are out of upper extreme or lower extreme. The values of upper extreme and upper extreme are calculated by the interquartile range (IQR)[2]. **Fig. 4** shows two kinds of boxplots. The first one is a generic example of box plot with the maximum, third quartile, median, first quartile, and minimum values labeled. And the second one is that outliers are plotted as individual dots and are in-line with whiskers. The relative

[2] https://en.wikipedia.org/wiki/Interquartile range



▲**Figure 4. Two kinds of box plots.**

vertical spacing between the labels reflects the values of the variable in proportion. Furthermore, in some situations, two or more box plots can be placed side-by-side on a coordinate plane to show how a phenomenon or scenario evolves with time, which is plotted along the independent-variable or x-axis. In our system, the box plot plays an important role in data distribution analysis and feature selection.

3.2 Machine Learning Algorithm

In practical applications, our prediction system based Spark integrates three machine learning algorithms to solve the problem of off-grid user prediction, including logistic regression, decision tree, and random forest. The system also supports the extension to integrate new machine learning methods. Next, we will introduce the three algorithms and the principle of parallelization based Spark.

(1) Logistic Regression

Logistic regression is one of the most commonly-used prediction methods. It is used with data in which there is a binary outcome variable, or where the outcome takes the form of a binomial proportion. Like linear regression, one estimates the relationship between predictor variables and an outcome variable. In logistic regression, however, one estimates the probability that the outcome variable assumes a certain value, rather than estimating the value itself. It uses a logistic function, as shown in Eq. (6), to describe the probabilities of outcomes.

$$f(z)=\frac{1}{1+e^{-z}}. \tag{6}$$

If y is the possibility that sample x is a positive example as well as parameter is w and b, we can gain Eqs. (7) and (8).

$$p(y=1|x)=\frac{e^{w^{\mathrm{T}}x+b}}{e^{w^{\mathrm{T}}x+b}+1}, \tag{7}$$

$$p(y=0|x)=\frac{1}{e^{w^{\mathrm{T}}x+b}+1}. \tag{8}$$

Then we'll estimate w and b with maximum likelihood method. For a given dataset $\{(x_i,y_i)\}_m^{i=1}$, we need to maximize log-likelihood, as shown in Eq. (9), which is the objective function of the model. Then we could use optimization methods in convex optimization theory, such as gradient descent method and Newton method, to get his optimal solution.

$$l(w,b)=\sum_{i=1}^{m}\ln p(y_i|x_i;w,b). \tag{9}$$

(2) Decision Tree

Decision trees and their ensembles are popular methods for the machine learning tasks of classification and regression. Decision trees are widely used since they are easy to interpret, handle categorical features, extend to the multiclass classification setting, do not require feature scaling, and are able to capture nonlinearities and feature interactions. Tree ensemble algorithms such as random forests and boosting are among the top performers for classification and regression tasks. Our system supports decision trees for binary and multiclass classification and for regression, using both continuous and categorical features. The implementation partitions data by rows, allowing distributed training with millions of instances.

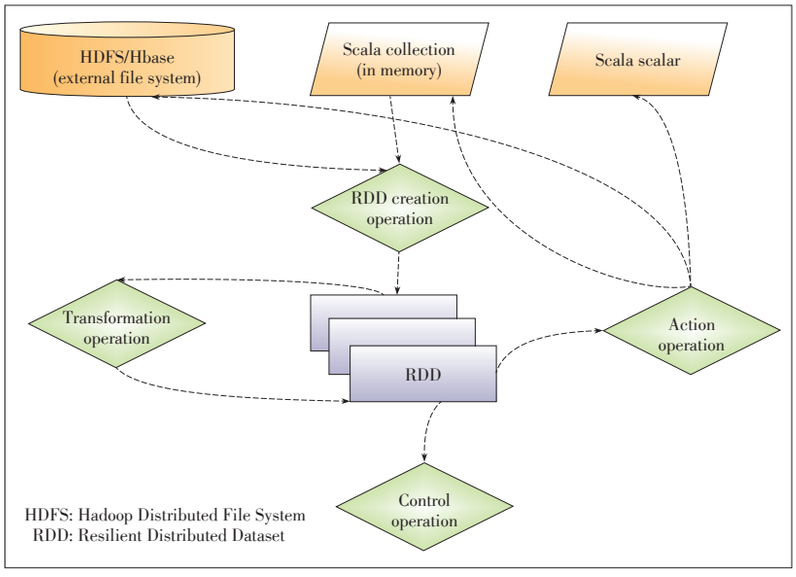
(3) Random Forest

Random forests [15] are ensembles of decision trees, which are one of the most successful machine learning models for classification and regression. They combine many decision trees in order to reduce the risk of overfitting. Like decision trees, random forests handle categorical features, extend to the multiclass classification setting, do not require feature scaling, and are able to capture nonlinearities and feature interactions. Random forests train a set of decision trees separately, so the training can be done in parallel. The algorithm injects randomness into the training process so that each decision tree is a bit different. Combining the predictions from each tree reduces the variance of the predictions, improving the performance on test data. In training model, the randomness injected into the training process includes two methods, including subsampling the original dataset on each iteration to get a different training set (a.k.a. bootstrapping) and considering different random subsets of features to split on at each tree node. Then a random forest will aggregate the predictions from its set of decision trees to make a prediction on a new instance.

(4) Parallelization

Parallel implementation of the machine learning algorithm mentioned above is based on Spark. For a spark data mining program, generally, the relationship between the RDD and operations is shown in **Fig. 5**. It is necessary to complete a job through creation operation, transformation operation, control operation, and action operation. Of course, there are multiple jobs in a Spark application. In Fig. 5, these concepts are important in spark programming.

- RDD: The main abstraction Spark provides is a RDD, which is a collection of elements partitioned across the nodes



▲Figure 5. Spark program flowchart.

of the cluster that can be operated on in parallel.

- Creation operation: RDD initial creation is the responsibility of the SparkContex, and the memory of the collection or external file system is the input source.
- Control operation: RDD can be saved in disk or memory for later reuse.
- Action operation: Since Spark is lazy computing, doing action operation on any RDD will trigger a spark job run and produce a result.

According to the above basic knowledge of Spark parallelization, all the algorithms that we integrate in our system can achieve the parallelization based on Spark. We will take the parallel gcForest algorithm as an example to introduce the implementation of the parallel algorithm in Section 4.

3.3 System Detailed Design

The ultimate goal of a software system is to solve practical problems. Next, we will introduce the detailed design of our system through unified modeling language (UML) class diagrams. In software engineering, a class diagram in UML is a type of static structure diagram that describes the structure of a system by showing the system's classes, attributes, operations (or methods), and the relationships among objects. The class diagram is an important part of object-oriented programming and used for detailed modelling translating the models into programming code. The detailed design of the proposed system at the programming level is shown in **Fig. 6**.

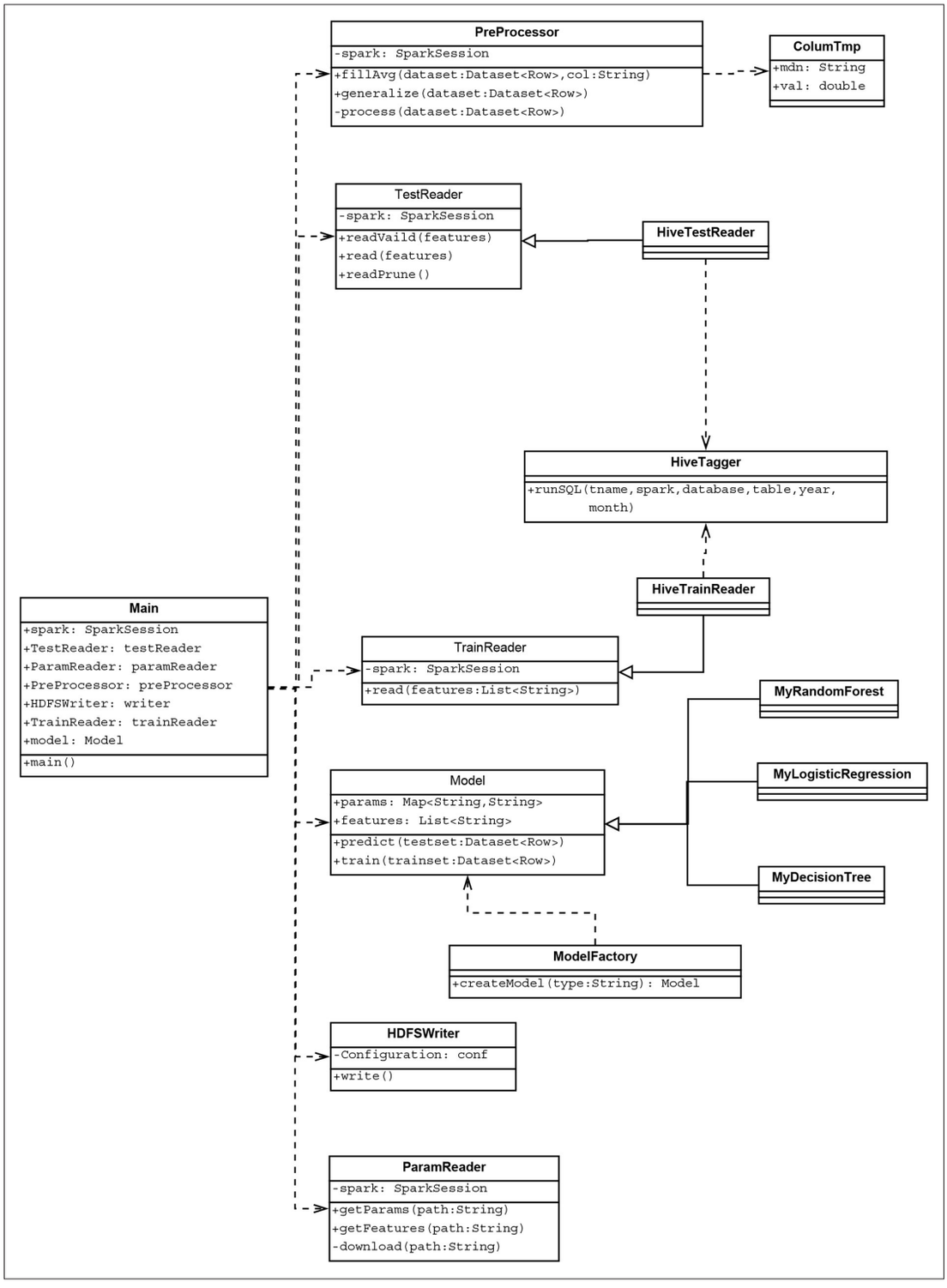
4 Parallel Deep Forest

4.1 Deep Forest

In this paper, in addition to a potential off-grid user prediction system, we also improve the deep forest algorithm based on Spark to solve the problem of off-grid prediction. The deep forest algorithm, called gcForest [16], is a novel decision tree ensemble approach which could compete to deep neural networks in many tasks. As shown in **Fig. 7**, there are two crucial components in the gcForest. The main one is cascade forest structure which capacitates gcForest to do representation learning. The other one is multi-grained scanning structure and it is useful to enhance the representational learning ability of cascade forest structure.

Cascade forest is the core of the whole algorithm structure. As we all know, deep neural networks [17], [18] always depend on the layer-by-layer processing of feature to make the networks have representation learning ability. Similar to neural networks, gcForest also has its own ability to do representation relying on cascade forest. Each layer of the cascade forest is composed of multiple random forests [15]. And every forest would produce an estimate of class distribution. Then all estimated class distribution will form a class vector, which combines with the original feature vector to form new vector as input to the next layer. As we all know, the diversity is essential to ensemble model [19]. Therefore, in order to encourage diversity in cascade forest, gcForest use two kinds of random forests, one is random forest, and the other one is completely-random tree forest [20]. As illustrated in Fig. 7, there is an example of binary classification task which is solved by gcForest. In cascade structure, each layer is made up of four forests, including two random forests and two completely-random tree forests. Each of the four forests will produce a two-dimensional vector, and finally one layer gains eight-dimensional vector which will combine with original feature vector to pass to next layer. In addition, we conduct cross validation on each forest to reduce overfitting. Furthermore, after building a new level, the performance of this level will compare to the performance of the previous level, which is estimated on validation. The training procedure will automatically stop if there is no significant performance gain. Thus, it is an important advantage that gcForest adaptively decides its model complexity by terminating training.

Multi-grained scanning plays a crucial role in the gcForest. It can enhance the performance of cascade forest. It is well known that handling feature relationships is very important in deep neural networks. For example, using spatial relationships among the raw pixels in convolution networks is effective to solve image problem. Inspired by this recognition, multi-grained scanning is to make use of the feature relationships by sliding windows which are used to scan the raw features. The window size we can set is multiple values. As illustrated in Fig. 7, there are raw feature of 400-dim and two windows sizes, 50-dim and 100-dim, respectively. Then the instances extract-

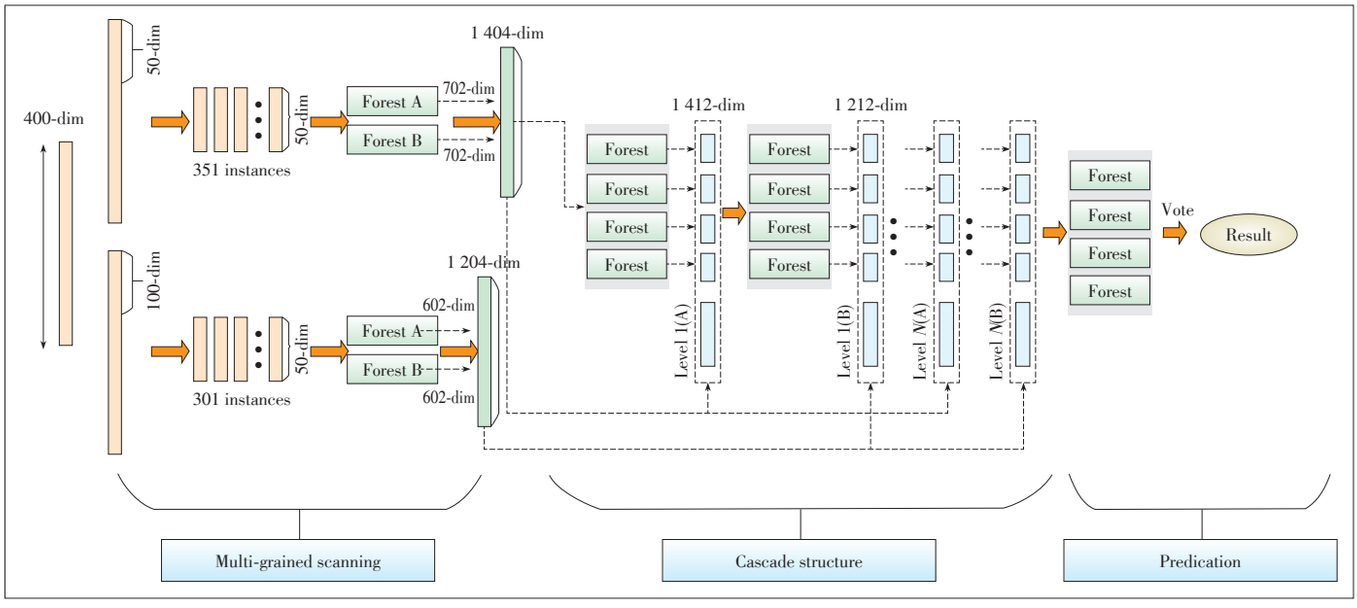


◀**Figure 6.**
The unified modeling language (UML) class diagram of the proposed potential off-grid user prediction system.

ed from sliding windows are produced to form new representation vectors through random forest finally.

4.2 Parallel gcForest

There is no denying that gcForest is state-of-the-art ensemble algorithm based decision tree. However, there are also certain limitations; for example, the algorithm becomes particularly slow when the volume of data is large. In practical application, it is an unavoidable problem to deal with large-scale data. Therefore, in our paper, we implement the parallelization of gc-

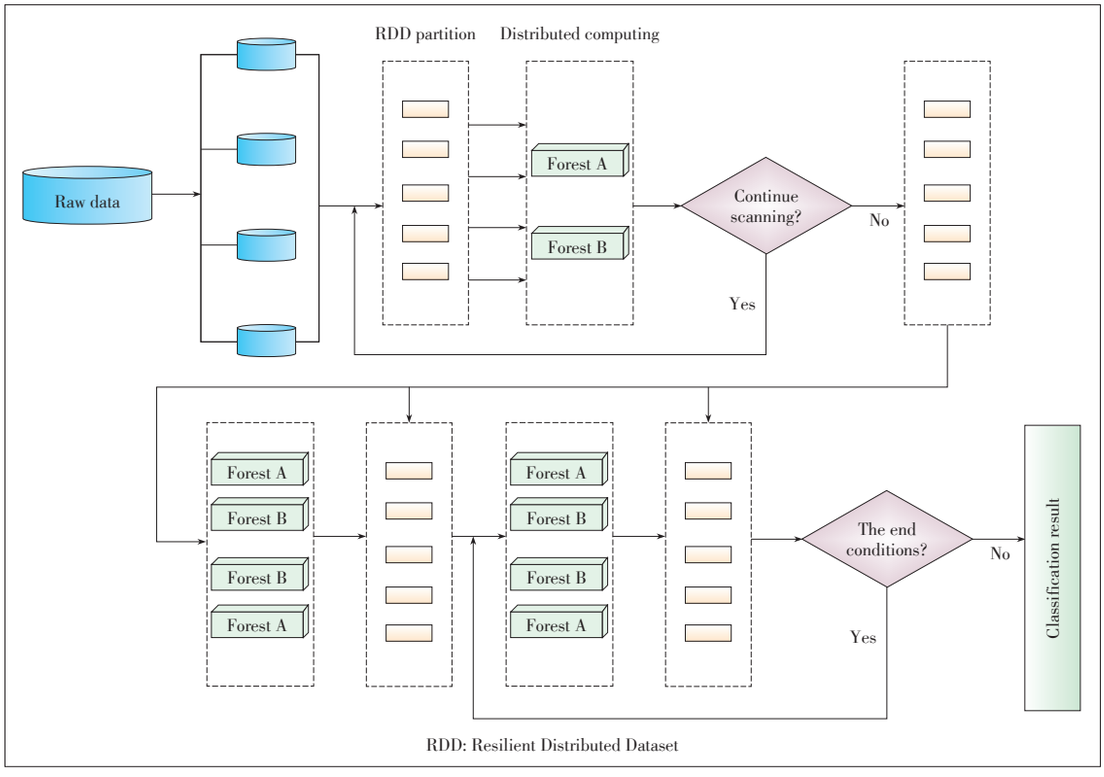


▲**Figure 7. The gcForest architecture. Suppose there are two classes to predict, two sliding windows are used and original features are 400-dim.**

Forest algorithm based on spark parallel framework, as shown in **Fig. 8**.

As mentioned above, the structure of gcForest algorithm is divided into two parts, including cascading structure, which capacitates gcForest to do representation learning, and multi-grained scanning structure, which is useful to enhance performance. Similarly, the algorithm is divided into two modules to achieve the multi-grained scanning structure and cascading structure. In the realization of parallelization, there are mainly the following innovations:

- The index-based scanning algorithm. In multi-grained scanning structure, the gcForest uses sliding windows to produce a large number sub-instances by scanning the raw features. The result of this is that more storage space is needed.



◀**Figure 8. The Parallel gcForest Algorithm.**

To solve the problem, we put forward an index-based scanning algorithm. When scanning, we generate a unique ID for each generated sample, and then we record the sample information, such as the original sample number and the beginning and the end of the scan, which can find newly generated instances from the original sample. When we build a decision tree in a random forest, we produce samples based on recorded information. After building the decision tree, the sample is deleted and only the model is kept. By doing this, we save a lot of storage space.

- Random sampling to construct random forest. This method is also to solve the newly generated samples and requires a lot of storage space problems in multi-grained scanning structure. Before training the stochastic forest model, we sampled the newly generated distances randomly. In other words, we do not use all the newly generated distances.
- Parallelization of Random Forest algorithm. In cascading structure of gcForest, the main part of each layer is the random forest model, involving random forest and completely-random tree forest. These two kinds of random forest models are easy to be implemented based on spark framework. As shown in **Fig. 9**, the construction of decision tree in forest is distributed evenly to each node of the cluster, and all trees grow in parallel with different nodes in the cluster. When choosing the optimal partition of the node of the decision tree, the sub-partitioning strategy is used. Each node records the corresponding partition in the data partition of the nodes, summarizes the data, and then finds the best partition by traversing. In addition, the growth of the decision tree in the forest is trained by the method of layer by layer.

5 Experiment

5.1 Datasets

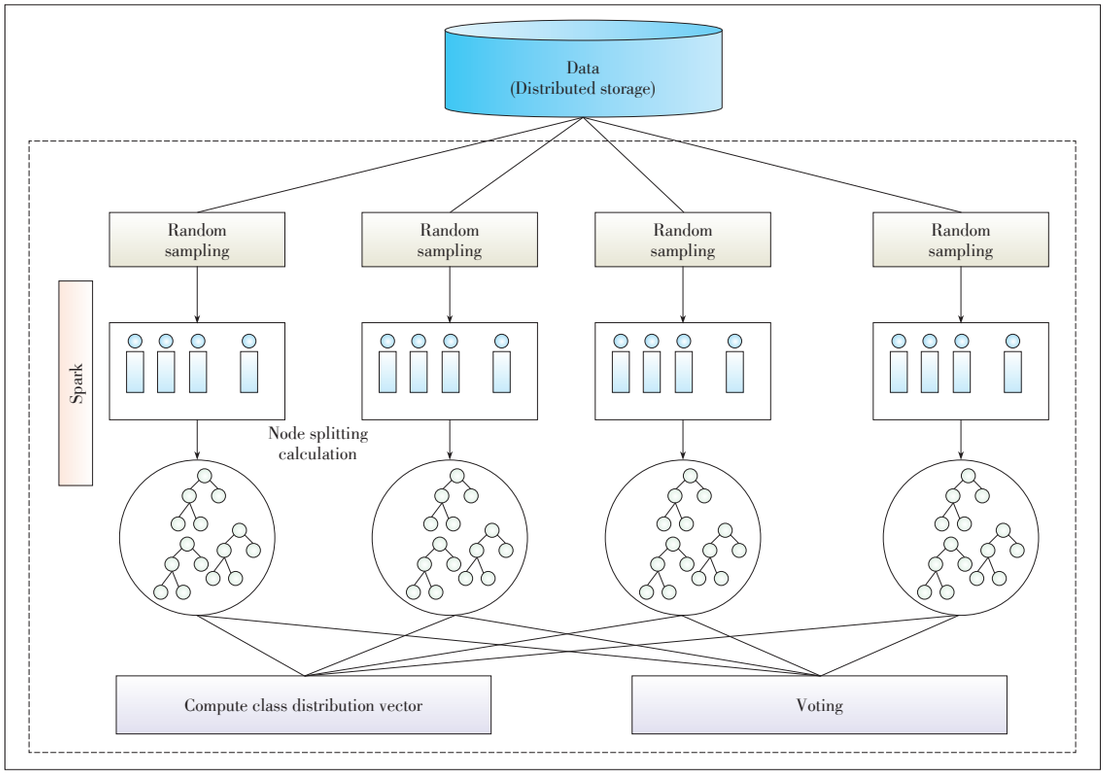
In order to validate the effectiveness of our system and the parallel gcForest algorithm, we conducted experiments on two real-world datasets. The first dataset is from a phone company which is used for churn management. The second dataset is operation data of China Telecom users. The information of experimental data is shown in **Table 1**.

(1) Churn Management Dateset:

This dataset contains summarized data records for each customer for a phone company. Our goal is to build a model so that this company can predict potential churners. There are 21 variables to represent the customers' record. This is a common set of off-grid data and the meaning of few variables as shown in **Table 2**.

(2) Telecom Users Dataset:

The Telecom user dataset is provided by telecom operators, which records the behavior of Telecom users with 131 dimensions. In addition, this dataset is raw data, not processed or simply processed, and the data scale is very large. The purpose of our system is to process these dirty data step by model and practical application. The data size is shown in Table 1. A few features are shown in **Table 3** to facilitate the understanding of



◀**Figure 9. Structure of the parallel random forest algorithm.**

▼Table 1. Experiment data

Dataset	Dimension	Training size	Test size
Churn management dateset	21	2 000	1 033
Telecom users dataset	131	1 million+	1 million+

▼Table 2. Features in the churn management dataset

Feature	Meaning
Account_length	How long this person has been in this plan
International_plan	This person has international plan=1, otherwise plan=0
Voice_mail_plan	This person has voice mail plan=1, otherwise plan=0
Number_vmail_message	The number of voice mails

▼Table 3. Some features in the Telecom users dataset

Feature	Meaning
LatestServiceTime	Last service time before collecting information
CallerCount	The number of calling in a month
DataDuration	Time spent in the data business in a month
SmsSendTimesChange	The number of SMS messages sent in a month

telecom users dataset.

5.2 Experimental Results

We mainly conducted a system functional test. In this paper, we use precision, recall, accuracy and F1-score to evaluate the experiment results which are commonly used metrics in machine learning. The results of our experiment are shown in **Tables 4** and **5**.

The churn management dataset is a dataset that has been cleaned and is often used for research on the problem of off-grid. Therefore, we used it to validate that our system and our parallel gcForest are effective. In this experiment, we firstly made feature selection by using the methods in our system, involving correlation calculation and box plots analysis. According to the results of the analysis, we finally selected 12 features as follow: account length, area code, international plan, voice mail plan, number vmail messages, total day minutes, total day charge, total eve minutes, total international minutes, total international calls, total international charge, and number customer service calls. After feature selection, we use the decision tree, logistic regression, and random forest to deal with the prediction problem. In particular, the first three algorithms used selected features and some strategies to train model in the experiment; for example, we used threshold-moving to reduce the bad impact of class-imbalance. Besides, the parallel gcForest use all features of the dataset to do end-to-end training, thereby reducing the use of rules in training. The reason is that gcForest can automatically filter useless features. The final results are shown in Table 4. The result of the decision tree algorithm is the best, followed by the parallel gcForest. The result of logical regression is the worst. The experimental results show that our system and parallel gcForest are effective.

▼Table 4. Experimental results based on the churn management dataset

Algorithm	Recall	Precision	Accuracy	F1
Decision tree	0.7907	1.0	0.9913	0.8831
Logistic regression	0.9535	0.4409	0.9477	0.6029
Random forest	0.7442	0.8205	0.9825	0.7805
Parallel gcForest	0.8140	0.8333	0.9855	0.8235

▼Table 5. Experimental results based on the Telecom user dataset

Training data	Test data	Recall	Precision	Accuracy	F1
September	October	0.7895	0.6023	0.9233	0.6833
September	April	0.7186	0.6682	0.9173	0.6925
October	September	0.7147	0.6723	0.9197	0.6926
October	April	0.7239	0.5310	0.9331	0.6126
April	September	0.7545	0.5305	0.9330	0.6125
April	October	0.7844	0.6052	0.9238	0.6833

The Telecom user dataset is the raw data produced in real life in recent years. This dataset is very large and of poor quality. The role of our system is to use this data to solve the problem of off-grid prediction. This dataset is mainly to verify the practicality and effectiveness of our system. We selected three months of user data, including June 2016, September 2016, and April 2017. Moreover, the size of every month's data is millions. These data are read from the HDFS and we get a better result to predict whether the user will become an off-grid user through data pre-processing, feature selection, training model, and a series of operations. In the experiment, we use k-fold cross validation to select the final algorithm to train model. The advantage of the system is that it can choose a better model. As shown in Table 5, the best performance is achieved by logical regression.

The experiment results show that, in the data of academic research, the performance of our system is better than the performance in real-world data. The results are normal because the data in the actual application are dirty and the data scale is very large. Table 5 shows that our system could solve the problem of off-grid user prediction problem effectively. In addition, Table 4 shows that the proposed Parallel gcForest is effective.

6 Conclusions

In this paper, we develop a parallel off-grid prediction system, which includes many machine learning algorithms for data mining. Built on the Spark computing framework, the whole system can be applied to large data to solve the problem of off-grid user prediction and used for practical applications. Furthermore, we also use the Spark parallel framework to speed up the gcForest algorithm. The new parallel algorithm enables

gcForest to solve practical problems, such as off-grid prediction problem and other problems of data mining.

References

[1] WARD J S, BARKER A. Undefined by Data: a Survey of Big Data Definitions [EB/OL]. (2013). https://arXiv preprint arXiv:1309.5821
[2] HAN L X, ONG H Y. Parallel Data Intensive Applications Using MapReduce: A Data Mining Case Study in Biomedical Sciences [J]. Cluster Computing, 2015, 18 (1): 403–418. DOI: 10.1007/s10586-014-0405-9
[3] LU P, DONG Z J, LUO S M, et al. A Parallel Platform for Web Text Mining [J]. ZTE Communications, 2013, 11(3): 56–61. DOI: 10.3969/j.issn.1673-5188.2013.03.010
[4] PAGANO F, PARODI G, ZUNINO R. Parallel Implementation of Associative Memories for Image Classification [J]. Parallel Computing, 1993, 19(6): 667–684. DOI: 10.1016/0167-8191(93)90014-c
[5] CHU C-T, KIM S K, LIN Y-A, et al. Map-Reduce for Machine Learning on Multicore [C]//19th International Conference on Neural Information Processing Systems. Vancouver, Canada, 2006: 281–288, 2007.
[6] DEAN J, GHEMAWAT S. Mapreduce: Simplified Data Processing on Large Clusters. Communications of the ACM, 51(1): 107–113, 2008.
[7] LEE K H, LEE Y, CHOI H, et al. Parallel Data Processing with MapReduce: a Survey [J]. ACM SIGMOD Record, 2012, 40(4): 11– 20. DOI: 10.1145/2094114.2094118
[8] KOLIOPOULOS A-K, YIAPANIS P, TEKINER F, et al. A Parallel Distributed Weka Framework for Big Data Mining Using Spark [C]//IEEE International Congress on Big Data. New York, USA, 2015. DOI 10.1109/BigDataCongress.2015.12
[9] COX L A. Data Mining and Causal Modeling of Customer Behaviors [J]. Telecommunication Systems, 2002, 21(2/3/4): 349–381. DOI: 10.1023/A:1020911018130
[10] ROSSET S, NEUMANN E. Integrating Customer Value Considerations into Predictive Modeling [C]//Third IEEE International Conference on Data Mining, Melbourne, USA, 2003: 283–290. DOI: 10.1109/ICDM.2003.1250931
[11] NATH S V, BEHARA R S. Customer Churn Analysis in the Wireless Industry: A Data Mining Approach [C]//Annual Meeting of the Decision Sciences Institute. China, 2003: 505-510
[12] SHVACHKO K, KUANG H R, RADIA S, et al. The Hadoop Distributed File System [C]//IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), Incline Village, USA, 2010: 1–10. DOI: 10.1109/MSST.2010.5496972
[13] DEAN J, GHEMAWAT S. Mapreduce: Simplified Data Processing on Large Clusters [J]. Communications of the ACM, 2008, 51(1): 107–113. ACM, 2008
[14] MYERS J L, WELL A D. Research Design and Statistical Analysis [M]. 2nd ed. Mahwah, USA: Lawrence Erlbaum Associates, 2010
[15] BREIMAN L. Random Forests [J]. Machine Learning, 2001, 45(1): 5–32. DOI: 10.1023/A:1010933404324
[16] FENG J, ZHOU Z-H. Deep Forest: Towards an Alternative to Deep Neural Networks [C]//Twenty-Sixth International Joint Conference on Artificial Intelligence. Melbourne, Australia, 2017: 3553–3559
[17] BA L J, CARUANA R. Do Deep Nets Really Need to be Deep? [C]//Advances in Neural Information Processing Systems. Red Hook, USA: Curran Associates, 2013: 2654–2662
[18] KRIZHEVSKY A, SUTSKEVER I, HINTON G E. ImageNet Classification with Deep Convolutional Neural Networks [C]//International Conference on Neural Information Processing Systems. Doha, Qatar, 2012: 1097–1105
[19] ZHOU Z H. Ensemble Methods: Foundations and Algorithms [M]. London, UK: Taylor & Francis, 2012
[20] LIU F T, KAI M T, ZHOU Z H. Isolation Forest [C]//Eighth IEEE International Conference on Data Mining. Miami, USA, 2009: 413–422

Biographies

LI Xuebing received the M.S. degree from the College of Information Science and Engineering, Yanshan University, China. His research interests include machine learning and data mining. He is currently a recommended system engineer at Baidu.

SUN Ying is a master candidate at the Institute of Computing Technology, Chinese Academy of Sciences, China. She received the B.S. degree from Beijing Institute of Technology, China in 2017. Her research interests include machine learning and data mining. She has published two research papers in SIGKDD.

ZHUANG Fuzhen (zhuangfuzhen@ict.ac.cn) received the B.S. degree in computer science from Chongqing University, China in 2006, and the Ph.D. degree in computer software and theory from the University of Chinese Academy of Sciences, China in 2011. He is currently an associate professor at the Institute of Computing Technology, Chinese Academy of Sciences, China. His research interests include machine learning, data mining, transfer learning, multi-task learning and recommendation systems. He has published around 60 papers in various journals and conferences, such as *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Cybernetics*, *IEEE Transactions on Neural Network and Learning System*, KDD, IJCAI, AAAI, ICDE, and WWW.

HE Jia is a Ph.D. candidate at the Institute of Computing Technology, Chinese Academy of Sciences, China. Her research interests include machine learning, Bayesian nonparametric learning, and multi-view learning. She has published several papers in some relevant research conferences, such as IJCAI, ICDM, ECML, and CIKM.

ZHANG Zhao is a Ph.D. candidate in the Institute of Computing Technology, Chinese Academy of Sciences, China. He received the B.S. degree from Beijing Institute of Technology, China in 2015. His research interests include machine learning, data mining, and relational learning. He has published several papers in some relevant research conferences and journals, such as EMNLP, CIKM, and information systems.

ZHU Shijun received the B.E. degree in management science and engineering from University of Science and Technology of China (USTC) in 2003. Working with the Wireless Big Data R&D Center of ZTE Corporation, he is responsible for the development of smart optimization and planning sytem of wireless networks.

HE Qing is a professor at the Institute of Computing Technology, Chinese Academy of Science (CAS), and he is also a professor at University of Chinese Academy of Sciences, China. He received the B.S. degree from Hebei Normal University, China in 1985, and the M.S. degree from Zhengzhou University, China in 1987, both in mathematics. He received the Ph.D. degree in fuzzy mathematics and artificial intelligence in 2000 from Beijing Normal University, China. He was with Hebei University of Science and Technology from 1987 to 1997. He is currently a doctoral tutor at the Institute of Computing and Technology, CAS. His interests include data mining, machine learning, classification, and fuzzy clustering.