Scholar Publishing Group
Machine Learning Theory and Practice
https://doi.org/10.38007/ML.2022.030201
ISSN 2790-0983 Vol. 3, Issue 2: 1-11

An Improved Random Forest Algorithm Based on Spark

Liming Zhang*
Jilin International Studies University, Changchun 130117, China
zhanglm574@sina.com
**corresponding author*

Keywords: Spark Distributed Platform, Random Forests, Optimization Algorithms, Parallelized Design

Abstract: Classification algorithms are an important branch of data mining, and are also of great importance in the era of BD. The random forest algorithm (RFA) is one of the classification algorithms and is widely used in various industries for its good classification performance. However, the performance of RFA is not so good when dealing with high-dimensional data and unbalanced data. The main objective of this paper is to improve the RFA based on Spark. In this paper, we read a large amount of relevant algorithm literature in terms of algorithm research, and gain a comprehensive understanding of what feature selection and unbalanced classification are, as well as what characteristics they have and how these problems should be solved. It then focuses on how some domestic and international scholars have solved these problems. This paper focuses on studying and analysing the strengths and weaknesses of the RFA, and makes relevant improvements to address the two weaknesses of the RFA. In order to solve the problems of the RFA in the field of feature selection and the field of unbalanced classification, the optimization is improved respectively, and the parallelized design of the optimized algorithm is finally implemented on Spark.

1. Introduction

With the rapid development of Internet technology and the increasing maturity of various applications and sensor technologies, huge amounts of data information can be accessed or accumulated by organisations in various fields. BD is valued by all industries. However, due to the nature of big data (BD), it is not possible to extract valuable insights directly from it, which is why extracting valuable insights from BD has become a popular research topic. Data mining techniques can be very good at extracting valuable information from data. There are many BD platforms available, and Spark is widely used for its advantage of fast iteration [1-2].

In a related study, Sridevi et al. proposed a novel IDS framework that can overcome these problems with the Boruta Feature Selection with Grid Search Random Forest (BFS-GSRF)

algorithm [3]. The performance of BFS-GSRF was compared with ML algorithms such as Linear Discriminant Analysis (LDA) and Classification and Regression Trees (CART). The proposed work was implemented and tested on the Network Security Laboratory - Discovery Knowledge Dataset (NSL-KDD). ERs show that the proposed model BFS-GSRF yields higher accuracy (i.e. 99%) in detecting attacks and it outperforms LDA, CART and other existing algorithms.An integrated random forest and gradient boosting machine learning algorithm (RFGBM) is presented by Amandeep et al. that takes into account object-oriented parameters such as cohesion, coupling, cyclic complexity errors, number of subgenerations and depth of the inheritance tree, and tested the reusability of the given software code [4]. In addition, the proposed algorithm was compared with other algorithms. Using RFGBM, performance metrics such as accuracy, error rate, relative absolute error and mean absolute error were improved. The algorithm also utilises the help of unsupervised filters to pre-process the data in order to eliminate missing values, thereby improving efficiency. The proposed algorithm outperforms the existing algorithms in terms of performance parameters.

This paper develops a study on the improvement of the RFA based on Spark. In the area of feature selection. The MMIC is used to score the features, and then the features are ranked according to their scores from highest to lowest and all features with high scores and some features with moderate scores (selected randomly) are selected to participate in the construction of the RFA. The final ERs demonstrate that the PM solves the problems encountered by the traditional RFA when dealing with high-dimensional data, and improves the accuracy and stability of the algorithm.

2. Design Research

2.1. Spark Distributed Platform

1) Spark Ecosystem

Spark provides a whole ecosystem, which includes.

(1) Spark Streaming.

(2) Spark ML lib and Spark ML.

(3) Spark SQL.

(4) Spark Graph X.

These four components make up the entire Spark ecosystem, and Berkeley Lab AMP refers to this complete set of components collectively as the Berkeley Data Analytics Stack, or BDAS for short; this set of components can be used in Spark without many restrictions [5-6].

2) Spark RDD

Resilient Distributed Dataset (RDD), a RDD [7-8]. Specifically, it is a distributed, fault-tolerant, scalable and partitioned data structure [9-10]. In addition, Spark provides a full set of supporting arithmetic for RDDs, allowing users to perform very rich logical computations [11-12].

RDDs have the following characteristics.

(1) It can be sliced and diced. i.e. there is a list of slices, similar to MapReduce, which also requires distributed data to perform the relevant computations [13-14].

(2) Fault-tolerant. An RDD is very fault-tolerant and will recompute tasks for that node based on its own data source.

(3) memory-based (elastic). The default location for RDDs is memory, but Spark will also store them on disk if memory resources are insufficient.

(4) Rich data sources. RDDs can be created in a variety of ways, i.e. from files in HDFS or tables in Hive, or from user-defined collections.

(5) Dependencies. The main dependencies are narrow and wide dependencies.

(6) Cacheability: RDDs cache data that needs to be reused, which greatly improves

computational performance.

(7) Directed acyclic graphs (DAGs). A specific set of DAG operations is generated after certain arithmetic operations are performed on an RDD, which also gives the notion of a loop that essentially stores the arithmetic operations between RDDs. When an error occurs somewhere, it recovers the RDD according to Lineage [13-14].

3) Spark runtime framework

There are three node roles: the driver node, the cluster manager, and the executor node.

The driver node is responsible for central coordination and scheduling of individual worker nodes. The executor node, which is the node that actually performs the tasks, also has two responsibilities, one is to run the tasks assigned to it by the driver node and return the execution results, and the other is to maintain the distributed in-memory storage. The cluster manager, on the other hand, is a role that is separate from the drive in order to manage a large-scale cluster, and it performs the initiation, management, backup and monitoring of the two node roles above [15-16].

2.2. Random Forest

DTs have features such as simple models that are easy to implement, but are prone to overfitting and the inability to guarantee the achievement of global optimality when performing classification, as shown in Figure 1, to avoid phenomena such as overfitting [17-18].

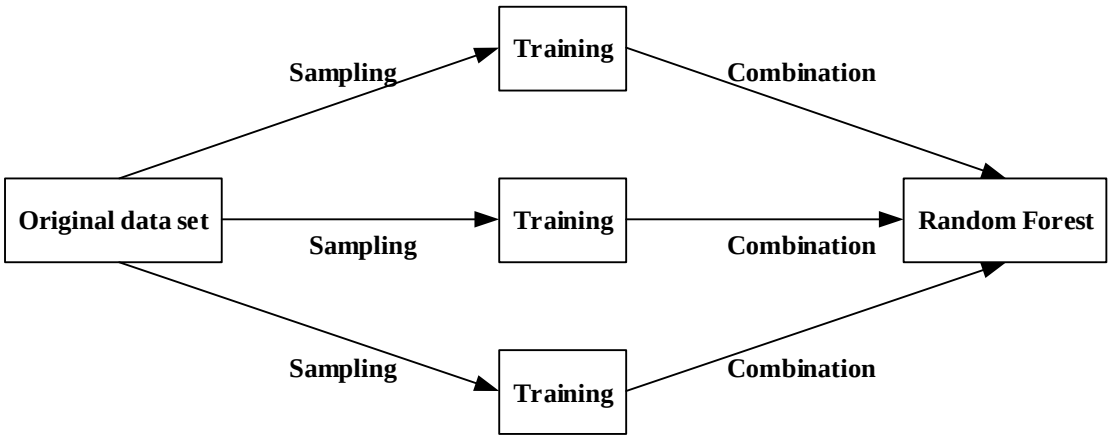

Figure 1. The core idea of the RFA

The first step is to generate multiple independent DTs, and the second step is to vote on the classification results and generate the final results.

Step1: Sample the training data with put-back sampling to obtain N subsets of samples.

Step2: After obtaining the sample subsets, the feature subset selection is performed.

Step3: Train a single DT on the basis of the sample subset and feature subset of the data.

Step 4: The trained DTs are combined to classify the data by voting.

The algorithm flow is shown in Figure 2.

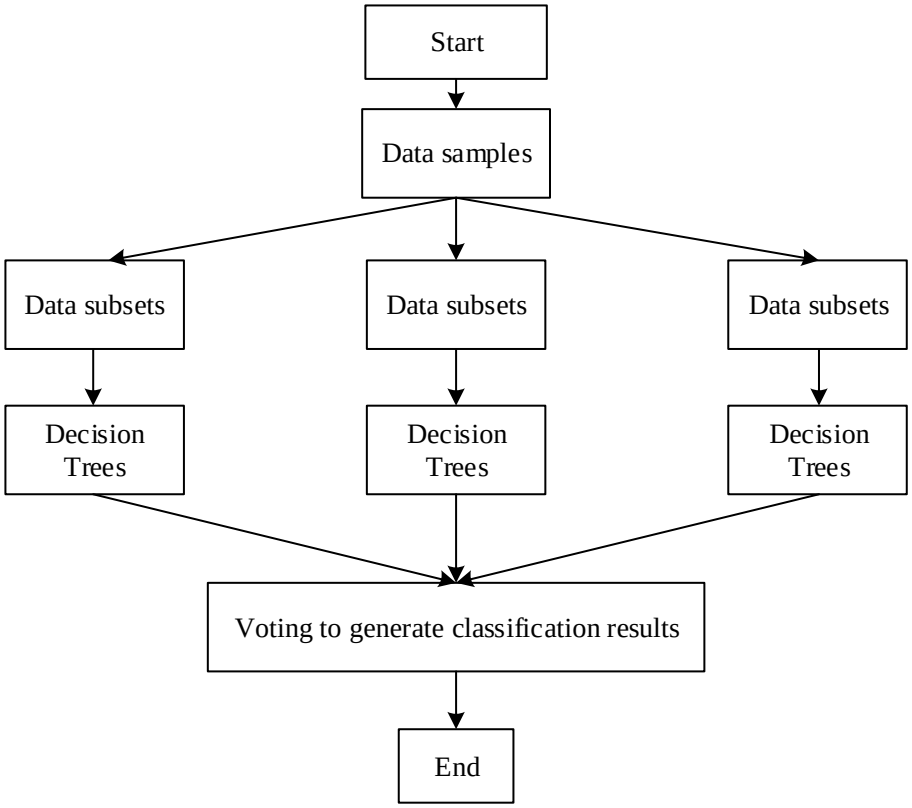

Figure 2. Flow chart of the RFA

The RFA mainly solves the problem of overfitting and local optimality when classifying a single DT [19-20]. The disadvantages are as follows.

Firstly, high-dimensional data containing redundant features can affect the classification accuracy of the RFA.

Second, it is not possible to distinguish the classification accuracy of single DTs, so the classification accuracy of the random forest model is somewhat affected by DTs with weak classification accuracy.

3. Experimental Study

3.1. The Process of Building the Spark Distributed Platform

1) Change the hostname of each host by modifying /etc/hostname and also modifying the /etc/hosts file so that each machine can communicate by hostname. You can test this by pinging the hostname.

2) Since Spark clusters communicate via ssh, configure ssh login.

3) Spark is based on Scala, which is a JVM language. Next, install the JDK, and in subsequent steps you will use some of the JDK's tools that the JRE does not have. There are two ways to install the JDK, either by downloading the binary executable and configuring it manually, or by using Oracle's installation package. In practice, manual configuration does not have a high success rate, so use the installation package and configure the executable location in /etc/profile. Regarding the choice of JAVA version, it is recommended to choose JAVA 7.

4) To install the Scala language environment, it is recommended to use version 2.10.5. Higher

versions have not been tested in Spark support and may have problems. Download the executable from the official web site, unzip it to a suitable location and configure the location of the executable in /etc/profile.

5) The platform uses Hadoop Distributed File System (Hadoop) for the file curing layer, so you need to install Hadoop. Hadoop is successfully installed.

(6) Spark needs to be compiled locally after downloading the source code, you can use Maven or Sbt build tool, Maven is an automatic build tool for Java, while Sbt is a special automatic build tool for Scala, there are still some problems. If you have problems with dependencies in the middle of a build, run the dependency check command mvn-U dependency:tree a few times.

After installing and successfully starting the service you can easily view the cluster in action using your browser.

At this point, the Spark distributed platform environment is complete, and it is important to configure several Spark configuration files. In Spark-env.sh, you need to configure the cluster's environment information and important variables such as node memory, as shown in Table 1.

Table 1. Spark configuration information

Spark-env.sh
export JAVA_HOME=/usr/lib/jvm/jdk1.7.0_79 export SCALA_HOME=/usr/lib/scala/scala-2.10.5 export HADOOP_HOME=/home/hadoop/hadoop-2.6.3 export HADOOP_CONF_DIR=$HADOOP_HOME/etc/Hadoop export LD_LIBRARY_PATH=$HADOOP_HOME/lib/native SPARK_MASTER_IP=192.168.8.101 SPARK_WORKER_MEMORY=7g SPARK_EXECUTOR_INSTANCES=1 SPARK_EXECUTOR_MEMORY=5g

In addition, you need to change the memory size of the Driver, otherwise you will get an error later in the program due to lack of space. The default size is 1GB. Modify "Spark.driver.memory" in Spark-defaults.sh, the memory size of Driver in this experimental environment is 2GB.

Finally, pay special attention to the firewall settings on the host, set the ports and protocols allowed to pass through Spark and HDFS in iptable. When using integrated development environments such as Intelli JIDEA to debug programs, you need to turn off the system's firewall and other firewall software, otherwise this software will prevent the Spark protocol from communicating properly.

3.2. The Random Forest CP

The random FCCP (forest classifier construction process) is divided into two main phases, the algorithm specific construction phase and the prediction phase.

In the construction phase, for the training set D={(Xi,Yj), the general CP of the random FC is as follows.

1) Sampling the training set D for k times

The training set D is randomly sampled k times with put-back and the corresponding k training subsets are generated.

$$Dtrain = \{D_1, D_2, D_3 \ldots D_k\} \quad (1)$$

In addition, each sample has some uncollected data, called out-of-box (OOB) data. If k samples are taken, k sets of OOB data can be obtained.

$$DOOB = \{OOB_1, OOB_2, OOB_3 ... OOB_k\} \quad (2)$$

These OOB data are typically used to validate the accuracy of the model.

2) Constructing each DT

A DT is constructed for each subset to be trained, with each classification tree following the ID3, C4.5 or CART model. In addition, each classification tree is trained by randomly selecting m features from M features into the nodes of the classification tree, where m<<M.

3) Generating a random FC.

The final random FC is made up of K DTs. A particular choice is made as follows.

$$H(X) = \sum_{i=1}^{k} h_i(x) \quad (3)$$

Where X is the input space, hi(x) is the DT, and x is the corresponding sub-input space for each decision tree (DT).

In the prediction stage, the prediction result of the RFA is voted on by all the classification trees.

The relevant codes are listed in Table 2 below.

Table 2. Pseudo-code of the RFA

The pseudo-code for the RFA is as follows.
Input: training data set D, number of sample subsets n, number of DTs a.
Output: random forest f(x).
1. for t = 1, 2, 3 .... , T.
(i) Select m random data points from the original dataset to obtain a training set Dt. (ii) Using the training set Dt, train a DT (can be ID3, C4.5 or CART classification tree), in the training process of each DT, the division rule for each node is to select k features randomly from all the features first, and then select the best division point from these k features to do the division of the DT subtree.
2. A DT is pooled into a forest, and the final classification result is voted on by these a DTs.

3.3. Parallelising the Design of Optimisation Algorithms

The traditional RFA hardly performs well in the face of large data. Therefore, in order to optimize the RFA to be able to handle large amounts of data, this chapter will carry out the parallelization design of the improved algorithm on the platform on Spark in terms of both data and tasks.

1) Data parallelisation

In Spark, there are two basic data structures, RDD and Data Frame, where RDD is a data structure with an execution unit and Data Frame is a data structure with an execution unit.

Data Frame provides detailed information about the structure of the data, how many features are in the dataset, and what type of features are in each dataset. It is also more efficient and faster to use Data Frame than RDD in Spark. So in Spark, Data Frame is used instead of traditional RDD.

2) Task Parallelisation

The parallelisation of the optimisation algorithm is designed in two main phases, the first of which is the calculation of the MICV (value) of each feature and the ranking of the features; the second phase is the construction of the RFA.

For the first stage, the main objective is to calculate the MICV of the features, which is mainly dependent on the maximum mutual information coefficient (MMIC). Therefore, the main design idea of the first stage is HDFS+Spark-based MMIC method.

For the second stage, the main objective is the construction of the RFA. So the main design idea of the second stage is HDFS + Spark-based implementation of the RFA.

For the overall parallelisation of the optimisation algorithm the design ideas are as follows.

First, the training set is read from the database and stored in HDFS, forming several blocks of data. The MICVs for each attribute are calculated separately.

The attribute intervals were then divided into high, medium and low categories based on the MICVs of the attributes being between high and low.

A training subset was made available for each compartment by sampling in separate pockets.

Each training subset was constructed as a tree according to the MT presented in this chapter.

Finally, the trees are pooled to generate the forest.

3) The specific steps of the optimised RFA are as follows.

Step1: Randomly have put-back to draw K sample subsets.

Step2: Calculate the importance of each feature according to the Relief-F algorithm.

Step3: Perform feature subspace selection according to the feature subspace generation strategy proposed in this paper.

Step4: DT training is carried out, and the node splitting attribute in the DT training process is selected by the Gini coefficient. The Gini coefficient is calculated as shown in equation (4).

$$Gini(D,A)=\frac{|D_1|}{|D|}Gini(D_1)+\frac{|D_2|}{|D|}Gini(D_2) \tag{4}$$

where

$$Gini=\sum_{i=1}^{k}P_i(1-P_i)=1-\sum_{i=1}^{k}P_i^2 \tag{5}$$

Step5: Repeat Step2-4 until all DTs are trained to obtain the random forest model

Step6: Vote to get the final output.

4. Experiment Analysis

4.1. Algorithm Improvement Experiments

After the classifier has completed the relevant task, the evaluation of the classification effect is an important part of the experiment. The confusion matrix is a very important way in the evaluation of classification effects. Taking the binary classification as an example, the confusion matrix is shown in Table 3.

Table 3. Confusion matrix

True/Predicted	0	1
0	TP	FP
1	FN	TN

For the evaluation of classifiers evaluation indicators are mainly composed of the following, which are calculated as follows.

(1) Correctness rate. The correctness rate is a common metric when performing classifier evaluation, and is calculated as shown in equation (6).

$$accuracy=\frac{TP+TN}{TP+FP+FN+TN} \tag{6}$$

(2) Error rate. The error rate represents the proportion of classification results that are misclassified and is calculated as shown in equation (7).

$$error=\frac{FP+FN}{TP+FP+FN+TN} \tag{6}$$

(3) Recall rate. The recall rate represents the proportion of samples in category 0 that are

correctly classified, and is calculated as shown in equation (8).

$$recall = \frac{TP}{TP + FN} \tag{6}$$

Classification accuracy was chosen as the evaluation metric for the improved algorithm.

In this paper, we use the datasets in UCI for comparison and validation, and the feature dimensions of each dataset are shown in Table 4.

Table 4. Dimensionality of features in the dataset

Data set number	Name of dataset	Dimension of dataset features
Dataset 1	Wine	13
Dataset 2	Breast	32
Dataset 3	Sonar	60
Dataset 4	Musk	168

4.2. Validation and Analysis

A comparison of the classification accuracy before and after the improvement of the RFA is in Table 5.

Table 5. Comparison of accuracy rates before and after improvement

Accuracy	Pre-improvement algorithm	Post-improvement algorithm
Dataset 1	0.87	0.89
Dataset 2	0.85	0.87
Dataset 3	0.81	0.83
Dataset 4	0.75	0.79

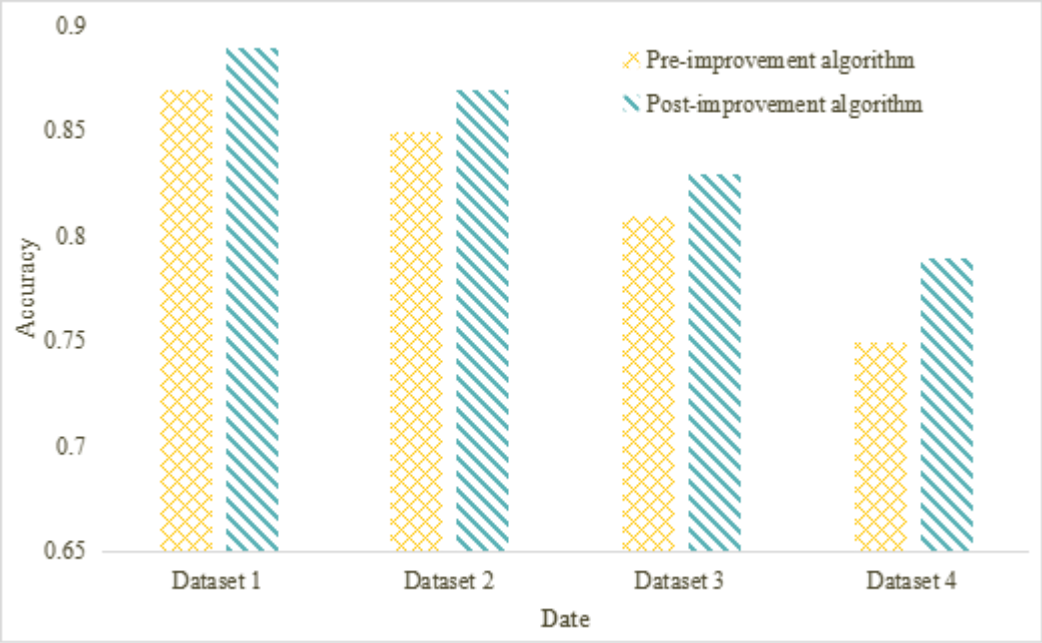

Figure 3. Comparison analysis of accuracy rate before and after improvement

As shown in Figure 3, the experimental results (ER) show that the optimized RFA proposed in this paper has a more obvious improvement in the classification accuracy compared with the RFA before optimization. At the same time, with the increase of the dimension of the data set features,

the improvement of the classification accuracy of the algorithm proposed in this paper is more obvious. Therefore, the optimized RFA in this paper has better results on datasets with higher feature dimensions.

The algorithm is then validated in a Spark cluster environment. The data set is an artificial test data set and the validation criterion is the speedup ratio, which is used to describe the reduction of the algorithm's running time. The variation of the algorithm's speedup ratio is shown in Table 6.

Table 6. Algorithm parallel acceleration ratios

Calculate the number of nodes/Acceleration ratio	Dataset 5	Dataset 6
1	1	1
2	1.6	1.8
3	1.9	2.2
4	2.6	3

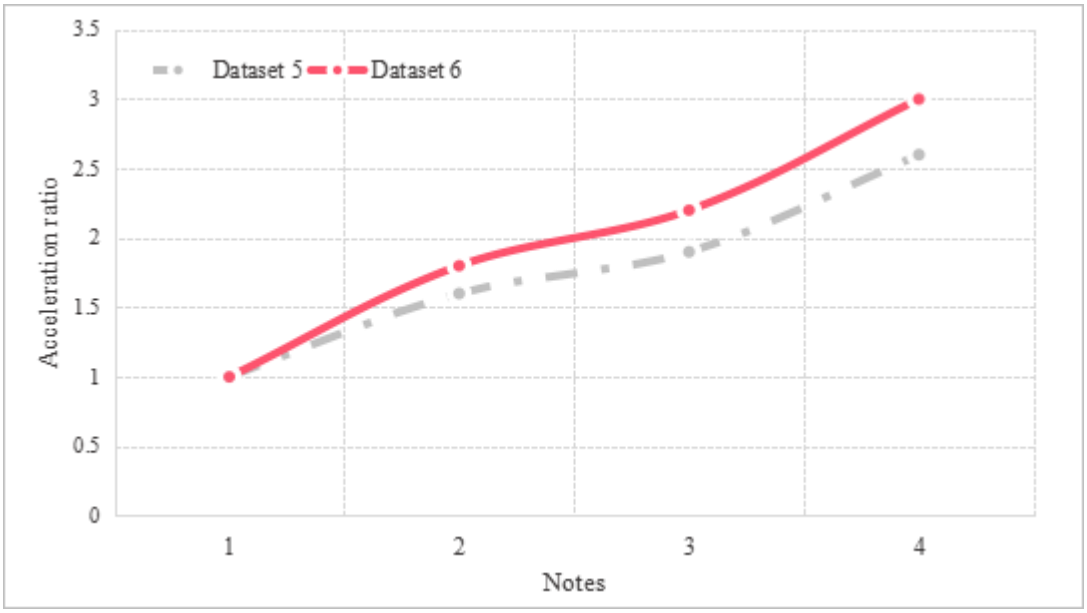

Figure 4. Algorithm parallel acceleration ratio curve

As can be seen in Figure 4, the algorithm speed-up ratio increases as the number of working nodes in the Spark cluster increases. It shows that the algorithm has good parallelisation. As the amount of data in different datasets, dataset 6 > dataset 5, the algorithm has a better speed-up ratio when the amount of data increases. Therefore, parallelisation of the optimisation algorithm based on Spark can be effective for BD analysis processing.

5. Conclusion

In this paper, two optimization methods are proposed for feature selection and unbalanced classification: in the feature selection area, the MMIC is used to score the features, then the features are ranked according to their scores from highest to lowest and all features with high scores and some features with medium scores (randomly selected) are selected to participate in the construction of the RFA. The final ERs demonstrate that the proposed method (PM) is a good solution to the problems encountered by traditional RFAs when dealing with high-dimensional data. In the field of unbalanced classification, the parallelization design of the optimization algorithm based on Spark is completed and the final ERs prove that the PM can solve the problems

encountered by the traditional RFA in the field of unbalanced classification well and achieve better results. Finally, according to the specific situation, the above two improved methods are used in the field of intrusion detection, which can solve some problems in the field of intrusion detection well and achieve good detection accuracy and detection speed.

Funding

This article is not supported by any foundation.

Data Availability

Data sharing is not applicable to this article as no new data were created or analysed in this study.

Conflict of Interest

The author states that this article has no conflict of interest.

References

[1] Bruno Henrique Meyer, Aurora Trinidad Ramirez Pozo, Wagner M. Nunan Zola: Improving Barnes-Hut t-SNE Algorithm in Modern GPU Architectures with Random Forest KNN and Simulated Wide-Warp. ACM J. Emerg. Technol. Comput. Syst. 17(4): 53:1-53:26 (2021). https://doi.org/10.1145/3447779

[2] Fahimeh Motamedi, Horacio Pérez Sánchez, Alireza Mehridehnavi, Afshin Fassihi, Fahimeh Ghasemi: Accelerating BD Analysis through LASSO-RFA in QSAR Studies. Bioinform. 38(2): 469-475 (2022). https://doi.org/10.1093/bioinformatics/btab659

[3] Sridevi Subbiah, Kalaiarasi Sonai Muthu Anbananthen, Saranya Thangaraj, Subarmaniam Kannan, Deisy Chelliah: Intrusion detection technique in wireless sensor network using grid search random forest with Boruta feature selection algorithm. J. Commun. Networks 24(2): 264-273 (2022). https://doi.org/10.23919/JCN.2022.000002

[4] Amandeep Kaur Sandhu, Ranbir Singh Batth: Software reuse analytics using integrated random forest and gradient boosting machine learning algorithm. Softw. Pract. Exp. 51(4): 735-747 (2021). https://doi.org/10.1002/spe.2921

[5] Pezhman Gholamnezhad, Ali Broumandnia, Vahid Seydi: An inverse model-based multiobjective estimation of distribution algorithm using Random-Forest variable importance methods. Comput. Intell. 38(3): 1018-1056 (2022). https://doi.org/10.1111/coin.12315

[6] Valeria D'Amato, Rita Laura D'Ecclesia, Susanna Levantesi: ESG score prediction through RFA. Comput. Manag. Sci. 19(2): 347-373 (2022). https://doi.org/10.1007/s10287-021-00419-3

[7] Josalin Jemima J., D. Nelson Jayakumar, S. Charles Raja, Venkatesh P.: Proposing a Hybrid Genetic Algorithm based Parsimonious Random Forest Regression (H-GAPRFR) technique for solar irradiance forecasting with feature selection and parameter optimization. Earth Sci. Informatics 15(3): 1925-1942 (2022). https://doi.org/10.1007/s12145-022-00839-y

[8] María Guadalupe Bedolla-Ibarra, María del Cármen Cabrera-Hernández, Marco Antonio Aceves-Fernández, Saúl Tovar-Arriaga: Classification of attention levels using a RFA optimized with Particle Swarm Optimization. Evol. Syst. 13(5): 687-702 (2022). https://doi.org/10.1007/s12530-022-09444-2

[9] Mirna Nachouki, Mahmoud Abou Naaj: Predicting Student Performance to Improve Academic Advising Using the RFA. Int. J. Distance Educ. Technol. 20(1): 1-17 (2022). https://doi.org/10.4018/IJDET.296702

[10] Saeed Samadianfard, Katayoun Kargar, Sadra Shadkani, Sajjad Hashemi, Akram Abbaspour, Mir Jafar Sadegh Safari: Hybrid models for suspended sediment prediction: optimized random forest and multi-layer perceptron through genetic algorithm and stochastic gradient descent methods. Neural Comput. Appl. 34(4): 3033-3051 (2022). https://doi.org/10.1007/s00521-021-06550-1

[11] Hafiz Syed Mohsin Abbas, Zahid Hussain Qaisar, Xiaodong Xu, Chunxia Sun: Nexus of E-government, cybersecurity and corruption on public service (PSS) sustainability in Asian economies using fixed-effect and RFA. Online Inf. Rev. 46(4): 754-770 (2022). https://doi.org/10.1108/OIR-02-2021-0069

[12] C. Venkata Narasimhulu: An automatic feature selection and classification framework for analyzing ultrasound kidney images using dragonfly algorithm and random FC. IET Image Process. 15(9): 2080-2096 (2021). https://doi.org/10.1049/ipr2.12179

[13] Musavir Hassan, Muheet Ahmed Butt, Majid Zaman: An Ensemble RFA for Privacy Preserving Distributed Medical Data Mining. Int. J. E Health Medical Commun. 12(6): 1-23 (2021). https://doi.org/10.4018/IJEHMC.20211101.oa8

[14] Sam Goundar, Akashdeep Bhardwaj: Property Valuation Using Linear Regression and RFA. Int. J. Syst. Dyn. Appl. 10(4): 1-16 (2021). https://doi.org/10.4018/IJSDA.20211001.oa13

[15] Alankrita Aggarwal, Kanwalvir Singh Dhindsa, P. K. Suri: Performance-Aware Approach for Software Risk Management Using RFA. Int. J. Softw. Innov. 9(1): 12-19 (2021). https://doi.org/10.4018/IJSI.2021010102

[16] Shenbagarajan Anantharajan, Shenbagalakshmi Gunasekaran: Automated brain tumor detection and classification using weighted fuzzy clustering algorithm, deep auto encoder with barnacle mating algorithm and random FC techniques. Int. J. Imaging Syst. Technol. 31(4): 1970-1988 (2021). https://doi.org/10.1002/ima.22582

[17] Bruno Henrique Meyer, Aurora Trinidad Ramirez Pozo, Wagner M. Nunan Zola: Improving Barnes-Hut t-SNE Algorithm in Modern GPU Architectures with Random Forest KNN and Simulated Wide-Warp. ACM J. Emerg. Technol. Comput. Syst. 17(4): 53:1-53:26 (2021). https://doi.org/10.1145/3447779

[18] Teer Ba: Performance analysis of sports training based on RFA and infrared motion capture. J. Intell. Fuzzy Syst. 40(4): 6853-6863 (2021). https://doi.org/10.3233/JIFS-189517

[19] S. Ramalingam, K. Baskaran: An efficient data prediction model using hybrid Harris Hawk Optimization with RFA in wireless sensor network. J. Intell. Fuzzy Syst. 40(3): 5171-5195 (2021). https://doi.org/10.3233/JIFS-201921

[20] François Bienvenu, Jean-Jil Duchamps, Félix Foutel-Rodier: The Moran forest. Random Struct. Algorithms 59(2): 155-188 (2021). https://doi.org/10.1002/rsa.20997