EasyChair Preprint

№ 12525

IDRF: an Improved Dynamic Random Forest Approach for Blockchain Time Series Data Classification

Ahmed Faris Alsayyad, Mohamed Mabrouk,
Ahmed Al-Shammari and Mounir Zrigui

March 16, 2024

IDRF: An Improved Dynamic Random Forest Approach for Blockchain Time Series Data Classification

Ahmed Faris Alsayyad [1], Mohamed Mabrouk [1], Ahmed Al-Shammari [2] and Mounir Zrigui [1]

1Research Laboratory in Algebra, Numbers Theory and Intelligent Systems RLANTIS, University of Monastir, Monastir, Tunisia

2Department of Computer Science, College of Computer Science and Information Technology, University of Al-Qadisiyah, Al Diwaniyah, 58002, Iraq

alsayyadahmed1985@gmail.com

Abstract. Recently, blockchain time series data has been widely studied throughout the communities of machine learning and data mining. However, Blockchain time series data dynamic class maintenance is still challenging. Existing works on blockchain time series data classification have shown serious accuracy and class maintenance limitations. Therefore, this paper proposes a novel framework called Improved Dynamic Random Forest (IDRF). The proposed framework includes two components as follows: initial classification and class maintenance. For classification, the proposed approach generates an initial set of classes. When new blockchain data arrive, we further proposed an incremental classification approach for maintaining the existing classes dynamically. Experiments on a real world dataset called "Bitcoin Heist Ransom Ware Address " verify the efficiency and effectiveness of the proposed blockchain time series data classification and maintenance approaches in terms of accuracy, execution time and RMSE.

Keywords: Blockchain, Classification, Dynamic Random Forest, Time series data, Security

1 Introduction

A blockchain is a distributed database shared among a computer network's nodes [2]. They are most recognized for their critical function in cryptocurrency systems for keeping a secure and decentralized record of transactions, although their applications are not restricted to Bitcoin [12]. Emerging technologies such as big data and blockchain are becoming commonplace. These technologies alter how businesses operate [11]. Recent Trends on Sophisticated Flooding Attacks and Detection Methods Based on Multi-s Sensors Fusion Data for Cloud Computing Systems. Fusion: Practice and Applications [1]. Blockchain data analysis is a process that entails interpreting, categorizing, and monitoring blockchain transaction data to provide users with essential insights and aid in risk assessment [18]. Blockchain analytics has become the most promising data science application with these analytical capabilities. It is not very easy to analyze real-time data. However, blockchain technology enables businesses to analyze data in real-time, which aids in detecting abnormalities at an early stage. Collecting important insights and patterns from massive databases is known as data min-

ing. The process of obtaining insights from data stored on a blockchain network is known as blockchain data mining [17]. It has grown in importance in the crypto ecosystem because analyzing blockchain data may reveal significant information, such as transaction volumes and market patterns, allowing firms and individuals to make more educated investment decisions [5]. Detecting patterns in trade activity and forecasting market developments.

Time series analysis is a method of analyzing a set of data points gathered over a period of time [8]. Time series classification analyses several labelled classes of time series data and then predicts or classifies the class to which a new data set belongs using supervised machine learning[9]. There has been little work on classification approaches in a dynamic environment. Several studies have suggested a classification model for classifying data[7]. These models could be more efficient for classifying dynamic data because they require a long execution time to finalize the classification process and give low accuracy with high RMSE. This gives a limitation which does not apply to real web blockchain data [15]. Therefore, we introduce a dynamic classification approach for classifying and maintaining the blockchain time series data classification.

1.1 Contributions

–We propose the fastest random forest classification for time series blockchain data.
– We propose a new incremental maintenance approach, including insertion and updating operations.
– We validate the proposed classification approaches with experiments on a real-world dataset and demonstrate their efficiency and effectiveness.

1.2 Paper Organization

The rest of the paper is organized as follows: The related work is presented in Section 2. which is followed by the proposed solution in Section 3. Section 4 presents the experimental results, and finally, Section 5 concludes the paper.

2 Related Work

This section highlights the proposed studies on Time Series Data Classification (TSC). TSC problems are becoming prevalent in statistics and machine learning [10]. Much research addresses some problems associated with time series data classification . Pfisterer et al, 2019 [13] Suggested Practical data analysis approaches employ additive models and the capability of combining practical basis representations with machine learning methods such as SVMs or classification trees. Functional Data Analysis (FDA) is a closely related field that relates to a set of challenges. Data analysis is constantly indexed over a particular region. Both domains aim to handle analogous issues while employing diverse methodologies; for example, classification or

regression tasks involving functional variables are a good illustration. The primary objective of this study is to provide non-specialist practitioners with a comprehensive evaluation of the most suitable method(s) to use for analyzing diverse time series data sets. This will be achieved through the execution of various analyses on a broad range of data sets, followed by a meticulous examination of the empirical outcomes. In addition, the authors provide the R software framework for doing functional data analysis in the context of supervised learning. This framework integrates machine learning algorithms and several linear statistical approaches. This makes it simple to combine such techniques with machine-learning toolkits.

Similarly, Zhou and Bian, 2019 [19] present an approach for identifying text emotion in Chinese literature based on BiGRU-Attention. Instead of a bidirectional extended short-term memory network (biLSTM), the method constructs a hidden layer with a bidirectional gated recurrent unit (biGRU). Then, it adds an attention model to input the outcome of each time point in the hidden layer to the fully connected layer, yielding a probability vector. The weights applied to each hidden layer result using the probability vector are then added to produce the result vector. The experimental results demonstrate that the model mentioned in this paper is more accurate and effective in text classification.

Cabello et al., 2020 [6] The proposal of STSF (Supervised Time Series Forest TSC) emerged as a means to address the question of whether all instances are used in the same context. The findings provided by STSF are presented in a manner that is easily comprehensible, while also improving the efficiency of categorization through the analysis of a subset of the original time series. This improved efficiency is achieved by leveraging the tree-based structure of STSF. The STSF methodology employs a top-down approach to identify pertinent subseries within three distinct time series representations. This process is conducted prior to the training of any tree classifier. The determination of subseries relevance is based on feature ranking metrics. The results obtained from experiments conducted on extensive real-world datasets demonstrate that the STSF method attains comparable levels of accuracy to state-of-the-art TSC techniques. Moreover, the STSF method exhibits significantly higher efficiency, enabling the prolonged utilization of TSC.

However, these studies ignore the incremental updates. As a result, these methods are impractical in dynamic environments. Furthermore, the proposed TSC algorithms are inefficient due to the high computing costs. Finally, these methods need to calculate the number of sensitive parameters to complete the classification process.

3 CLASSIFICATION APPROACHES

This section explains the technical details of the classification approaches and proposed approach for blockchain time series data classification. Section 3.1 discusses the baseline approach (Random Forest). Section 3.2 discusses the Dynamic Random Forest (DRF), and finally, the proposed Improved Dynamic Random Forest (IDRF) is discussed in Section 3.3.

3.1 Baseline Approach : Random Forest Tree

A Random Forest (RF) Algorithm is a supervised machine-learning method widely used in Machine Learning for Classification and Regression applications. The more trees in a Random Forest algorithm, the greater its accuracy and problem-solving capability.

Random Forest is a classifier that uses the average of many decision trees on different subsets of a given dataset to increase its predicting performance [16]. An empirical study of automated privacy requirements classification in issue reports. Automated Software Engineering, It is built on the idea of ensemble learning, which integrates numerous classifiers to solve a complicated issue and enhance the model's performance. The Random Forest Algorithm is explained in detail in the following steps:

Step 1: Choose random samples from a data or training set.

Step 2: This algorithm will generate a decision tree for each training set.

Step 3: The choice tree will be averaged for voting.

Step 4: Finally, choose the prediction result with the most votes as the final forecast result.

Each tree has distinct traits, variations, and qualities that distinguish it from other trees. Not all trees are created equal. Immune to the curse of dimensionality: Because a tree is a conceptual construct, it does not need to be evaluated for features. As a result, the feature space is decreased. Parallelization: Because each tree is built independently from diverse data and characteristics, we can fully use the CPU to build random forests. Splitting the training and testing: We do not need to separate the data for train and testing in a Random Forest since The decision tree's ability to perceive the input is limited to only 30%. Stability: The result is determined by the Bagging technique, wherein it is determined by means of a majority vote or average. Hyper-parameters are used in random forests to improve model performance and predictive power or speed up the model. To improve predictive power, the following hyper-parameters are used:

n_ estimators: The number of trees the algorithm constructed before averaging the products.

Max_features: The amount of features used by random forest before contemplating splitting a node.

Mini_sample_leaf: Counts the number of leaves needed to separate an internal node.

To boost the model's speed, the following hyper-parameters are used:

N_jobs: Tells the engine how many processors it can utilize. If the value is 1, only one processor can be used; if the value is -1, there is no limit.

Random state: Controls the sample's unpredictability. If the model has a defined value of random state and has been given the same hyper-parameters and training data, it will always generate the same outcomes [14].

Definition 1. " A random forest is defined as a classifier containing of a group of tree-structured classifiers $\{h(x, \Theta k), k = 1, ..., L\}$, where $\{\Theta k\}$ are random vectors

that are distributed individually, with each tree h(x, Θk) casts a unit vote for the most common class at input x [3]".

```
Data : D={d1,d2,…,dn}, Precondition: A training set S := (x1, y1), . . . ,(xn, yn), features F, No. of trees in forest B
  Results : C={c1,c2,..cn}
    function Random_Forest(S , F)
    H ← ∅ ;
        for i ∈ 1, . . . , B do
      S (i)  ← A bootstrap sample from S;
   hi ← Randomized Tree Learn(S (i) , F);
         H ← H ∪ {hi};
        end for
       return H
    end function
  function Randomized Tree Learn(S , F) At each node:
    f  very small subset of F;
    Split on best feature in f;
    return The learned tree;
  end function
```

Algorithm 1: Baseline Approach

Algorithm description: Algorithm 1 presents the steps of classification of blockchain time series data using a baseline (Random Forest) algorithm. In lines 1-2, we prepare the training and testing data and create the function random forest with two variables, S and F. In lines 3-4, we create an indicator which refers to a sub-tree with 0 value. In lines 5-6, we make a loop from 1 to the number of trees created (B). In lines 7-8, we assign strap elements from the training set (S). In lines 9-10, we create a function for a sub-new tree of the learning set and increment the new sub-tree. In lines 11-12, we assign a feature and choose the best subset of features by splitting it. Finally, we return the best-learned tree and close the function in lines 13-14.

3.2 Dynamic Random Forest (DRF)

Dynamic Random Forest (DRF) is an adaptive tree-based classification approach. This approach is proposed by Bernard et al., 2012[4]. The key concept is to The objective is to perform direct tree induction in a manner that maximizes the dependence of each tree on the present trees in the ensemble. The achievement of this task is facilitated by the implementation of resampling strategies on the training data[20]. These approaches involve the application of boosting methods and the integration of other randomization procedures often utilized in traditional Random Forest (RF) systems. Regarding accuracy, the DRF outperforms the typical static RF induction approach significantly. The Dynamic Random Forest (DRF) encompasses a series of three primary procedures: the resampling of training data, the integration of boosting methods,

and the incorporation of supplementary randomization mechanisms commonly employed in conventional Random Forest (RF) approaches. The mechanism of DRF is to prevent the introduction of trees that might reduce the performance of the forest by forcing the algorithm to create only trees that match the already produced ensemble. The equation of DRF is explained in the following.

$$y = c(x,y) = \frac{1}{|h_{oob}|} \times \sum_{hi \in h_{oob}} I(h_i(x_i) \quad (1)$$

Where I (.) is the indicator function; x represents an input data point, and y represents its actual class; hi(x) is a similar formulation. for h(x, Θi), which represents the i-th classifier output; and hoob stands for the collection of x out-of-bag trees, i.e. the trees for which x is an out-of-bag instance.

Data : D={d1,d2,…,dn} ,D new={d new _1,d new _2,…,d new _n } , T is a training set (xi , yi) ,N is a training instances in T ,M the feature number, M is the feature number, L is the forest tree number
Results: C={c_1,c_2,..c_n} ,C new ={c_{new_1}, c_{new_2},…, c_{new_n} }
for all $x_i \in T$ **do**
$D_l(x_i) \leftarrow 1 / N$;
end for
for l from 1 to L **do**
$T_l \leftarrow$ a bootstrap sample;
$T_l \leftarrow T_l$ weighted with D_l;
tree ← Random Tree(T_l);
forest ← forest S tree;
Z ← 0;
for all $x_i \in T$ **do**
if $_{oo}$BTrees (x_i) ≠ ∅ **then**
D l +1(xi) ← W(c(xi , yi));
else
$D l_{+1}$ (x_i) ← D_l (x_i) ;
end if
Z ← Z + D_{l+1}(x_i) ;
end for
for all xi ∈ T **do**
D_{l+1}(x_i) ← D_{l+1}(x_i) / Z ,
end for
end for
return forest

Algorithm 2: Dynamic Random Forest (DRF)

Algorithm description: Algorithm 2 presents the steps of classification of blockchain time series data using the Dynamic Random Forest (DRF) algorithm. In lines 1-2,

prepare the training and testing data and create the function random forest with four variables N, M, L, and W. In lines 3-4-5, assign an instance for each element in the training set. In line 6, make a loop from 1 to the number of trees created (L). In line 7, taking a sample training instance randomly from tree. In line 8, put the new training instances with a weighted random tree. In lines 9-10, create a new random tree with features. In line 11, make an indicator with 0 value. In lines 12-17, for each element in the training set, check the value not equal to 0; if it is true, the new element is replaced with a new weighted value. In lines 18-19, adjust the Z value with the new tree input. In lines 20-21, creating a new tree with new features. However, the DRF approach involves only insertion operations to update the generated classes. Therefore, there is a need to improve the DRF approach to be able to classify web data and maintain the generated classes frequently.

3.3 Proposed Improved Dynamic Random Forest (IDRF)

Figure 1 shows the proposed framework called" Improved Dynamic Random Forest (IDRF)".

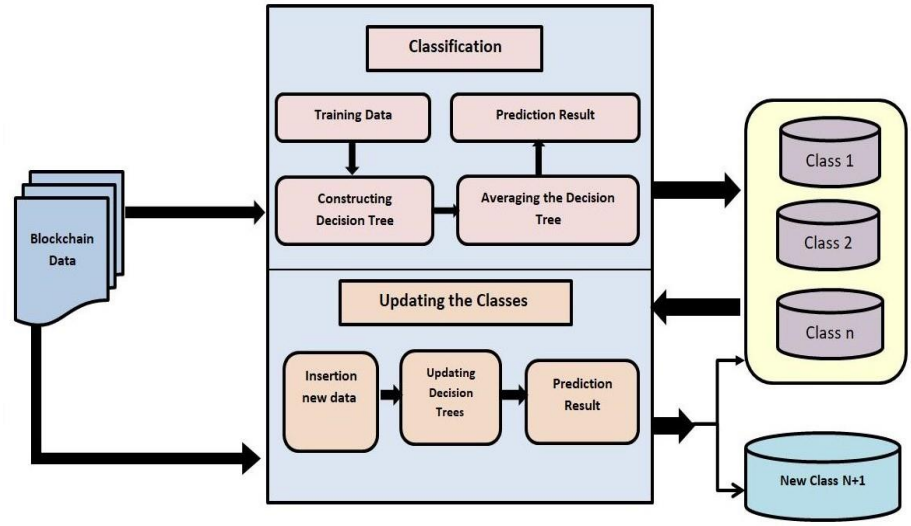

Fig. 1. IDRF framework

Improved Dynamic Random Forest (IDRF) approach for blockchain time series data classification. It is based on a sequential technique that generates an ensemble of random trees by making each one reliant on the one before. The adaptive resampling process of boosting inspired this dynamic component of the DRF method. The DRF method combines the same concept with the randomization techniques used in "classical" RF induction algorithms.

The first phase involves four main stages that are stated as follows:

Pre-processing: Data pre-processing converts data into a format that can be handled more readily and efficiently in data mining, machine learning, and other data science operations. The approaches are often utilized in the early phases. With many dimensions or features per observation, principal component analysis (PCA) is used to analyze massive datasets, improve data interpretation while retaining the most information, and enable multidimensional data visualization. Formally, PCA is a statistical method for lowering a dataset's dimensionality.

Training data: training data is an essential step in time series data classification to train the IDRF approach to get more accurate predictions.

Constructing Decision Tree: A decision tree is generated on a full dataset utilizing all the attributes of interest. It picks random observations and particular features to build numerous decision trees and then averages the results. For classification problems, the random forest output is the class chosen by the majority of trees.

Prediction result: The best possible result is obtained by selecting the majority of the best value. In this stage, we use many decision trees to get less over-fitting, higher accuracy, and approximate missing values.

The second phase involves three main stages that are stated as follows:

Insertion new blockchain data: The first stage in the updating the class is data insertion. New incoming data will be added to the framework to be processed incrementally as shown in figure 1.

Updating decision trees: new decision trees are built. Increasing the number of trees will improve the performance of the IDRF approach. With incremental updating, historical data instances do not need to be reprocessed to update existing classes. This might be helpful in cases where the initial data was too big to handle. Therefore, the features of the data change over time, or the complete data is not accessible when the tree is updated.

Prediction results: The best result is reached by choosing the majority of the best value. At this level, we reduce over-fitting, increasing accuracy and estimating missing data.

Data : D={d1,d2,…,dn}, D new={d new _1,d new _2,…,d new _n } , T represents training set (xi , yi) ,N represents number of training instances in T , M numbers of features , L numbers of trees in the forest to be built ,W(c(x,y)) it's a weighting function inversely proportional to c(x, y)
Results: C={c1,c2,..cn}, C new ={c new_1, c new_ 2,…, c new_ n }

```
from sklearn.decomposition import PCA
pca = PCA(n_components = 2);
X_train = pca.fit_transform(X_train());
```

```
X_test = pca.transform(X_test);
explained_variance = pca.explained_variance_ratio;
   for all xi ∈ T do
    for l from 1to L do
     Tl← a bootstrap sample;
          tree ← Random Tree(Tl) ;
     forest ← forest S tree ;
     Dl ( xi ) ← l / N ;
     Z ← 0 ;
      for all xi ∈ T do
        if OoBTrees( xi ) ≠ Ø then
          D l+1(xi) ←W(c(xi , yi);
        else
          D l+1(xi) ← Dl(xi) ;
        end if
       Z  ← Z  +  Dl+1(xi)  ;
      end for
       for all  xi ∈ T do
        D l+1( xi ) ←  D l+1( xi ) / Z;
        Update  D l+1( xi );
       end for
    end for
    return forest
```

Algorithm 3: Improved Dynamic Random Forest (IDRF)

Algorithm description: Algorithm 3 presents the steps of classification of blockchain time series data using the proposed Improved Dynamic Random Forest (DRF) algorithm and, in lines 1-2, prepares the training and testing data. In lines 3-7, recall the PCA function for each element in the training and testing set. In lines 8-9, take a sample training instances randomly from a generated tree. In line 10, put the new training instances with a weighted random tree. In lines 11-12, create a new random tree with features. In line 13, make an indicator with 0 value. In lines 14-19, for each element in the training set, check that the value is not equal to 0; if it is true, the new element is replaced with a new weighted value. In lines 20-21, adjust the Z value with the new input of the tree (insertion). In lines 22-23, it creates a new tree with new features; in line 24, it updates the generated class incrementally.

4 Experimental Results

The experimental results of the proposed IDRF approach demonstrated in this section. Several evaluation aspects are conducted in this paper to verify the efficiency and effectiveness of the proposed classification approaches as well as compare it with traditional and existing classification approaches. The experiments are implemented in Python programming language and executed by a processor Intel(R) Core(TM) i5-6300U CPU @ 2.40GHz.

4.1 Dataset Description

We conducted experiments on a real dataset. We use the Bitcoin Heist Ransom Ware Address dataset. The format of the dataset is. (.csv) file and the total size is 224 MB. The dataset is classified into four subsets: Small, Medium, Large, and Very large. The first subset contains records.
It is available online at the following link: https://doi.org/10.24432/C5BG8V. The experimented dataset includes the whole Bitcoin transaction history from January 2009 to December 2018. It collected daily network transactions and built the Bitcoin graph for 24 hours. The dataset includes ten attributes that are stated as follows: address, weight, count, looped year, day, length, income, neighbors, label

4.2 Root Main Square Error (RMSE)

The Root Main Square Error (RMSE) is one of the most prominent measures for evaluating the quality of model predictions. It is extremely helpful to have a single number to evaluate a model's performance. It can be measured during training, cross-validation, or monitoring after deployment. The equation of RMSE is described as follow:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} \|y(i) - \hat{y}(i)\|^2}{N}}, \qquad (2)$$

Where N is the number of data points, y(i) is the i-th measurement, and y^(i) is its corresponding prediction.

Figure 2 shows the RMSE of the proposed classification approaches (RF, DRF, and IDRF). The IDRF has the lowest RMSE value (less prediction errors), whereas the RF has the most excellent RMSE value across all subsets. The findings show that the DRF has a lower RMSE value than the RF approach and is more significant than the IDRF approach (RF>DRF>IDRF), indicating that the IDRF is the most effective classification approach.

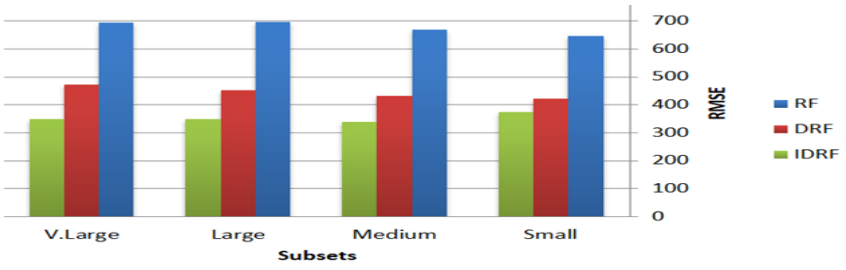

Fig. 2. RMSE of the proposed classification approaches (RF, DRF, and IDRF)

4.3 Accuracy

Classification accuracy is the ratio of accurate predictions to the total number of input samples. Figure 3 shows the accuracy of the proposed classification approaches, including Random Forest (RF), Dynamic Random Forest (DRF), and Improved Dynamic Random Forest (IDRF). We can observe that the IDRF has outperformed RF and DRF in all subsets of the experimented dataset. The main reason behind this performance is using the Principal Component Analysis (PCA) with IDRF approach for preprocessing blockchain data. Obviously, it influences by increasing the data quality and reducing the computational cost.

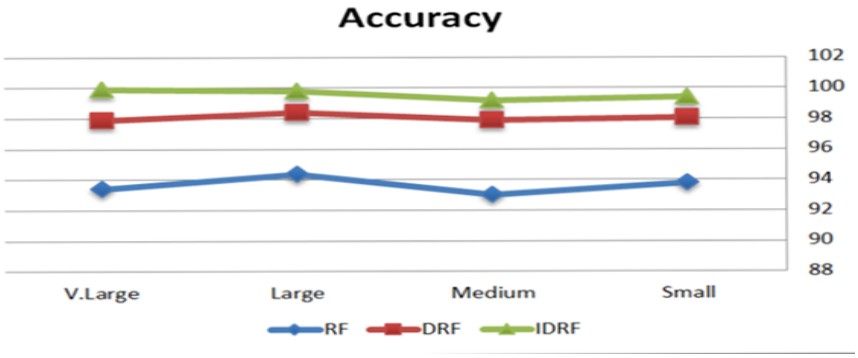

Fig. 3. Accuracy of the proposed classification approaches

4.4 Execution Time

The execution time is an essential metric to measure the efficiency and the performance of. The length of time needed for a model to run depends on a variety of factors, including the size of the dataset and the complexity of the model. Figure 4 shows the execution time of blockchain time series data classification using DRF and IDRF approaches. We observe that the IDRF requires less running time in comparison with the DRF approach in the entire subsets. Specifically, the IDRF maintains only frequently updated (active) classes to avoid unnecessary computations in both the insertion of new data and updating the existing classes.

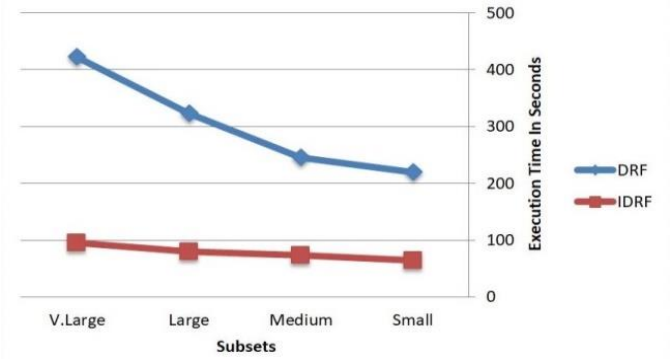

Fig. 4. Execution time of blockchain time series data classification using DRF and IDRF approaches

5 CONCLUSION

This paper introduces an effective framework for time series blockchain data classification. The proposed framework includes two main components: (1) initial classification and (2) class maintenance. For classification, we propose a fast and effective classification approach called Improved Dynamic Random Forest (IDRF) for classifying blockchain time series data. For the class maintenance, IDRF dynamically maintains the set of current classes by inserting new data and updating the existing classes. Verified experimentally, the proposed approaches outperformed other classification approaches regarding accuracy, Root Main Square Errors (RMSE), and processing time. The development in the classification and maintenance of time series blockchain data can improve the performance of blockchain applications. For future work, further improvements will be considered in the maintenance approach to be more effective in dynamic situations.

REFERENCES

1. Alhammadi, N. A. M., Mabrouk, M., & Zrigui, M.: Recent Trends on Sophisticated types of Flooding Attacks and Detection Methods based on Multi Sensors Fusion Data for Cloud Computing Systems. Fusion: Practice and Applications, 11(1), 37-7(2023).
2. Ali, A. A. M. A., Mabrouk, M., & Zrigui, M. A Review: Blockchain Technology Applications in the Field of Higher Education. Journal of Hunan University Natural Sciences, 49(10) . (2022).
3. Amlak, G. M. H., & Al-Saedi, K. H. K. Data Mining Techniques for Cloud Privacy Preservation. International Journal of Intelligent Systems and Applications in Engineering, 11(6s), 246-256(2023).
4. Bernard, S., Adam, S., & Heutte, L.: Dynamic random forests Pattern Recognition Letters, 33(12), 1580-1586(2012).
5. Brunner, C., Knirsch, F., & Engel, D. SPROOF: A Platform for Issuing and Verifying Documents in a Public Blockchain. In ICISSP (pp. 15-25) (2019).

6. Cabello, N., Naghizade, E., Qi, J., & Kulik, L.: Fast and accurate time series classification through supervised interval search. In 2020 IEEE International Conference on Data Mining (ICDM) (pp. 948-953). IEEE. (2020, November).
7. Gunawan, D. (2020). Classification of privacy preserving data mining algorithms: a review. Jurnal Elektronika dan Telekomunikasi, 20(2), 36-46.
8. Hazar, M. J., Abid Muslam Abid Ali, A., Zrigui, S., Maraoui, M., Mabrouk, M., & Zrigui, M.. Educational Videos Recommendation System Based on Topic Modeling. In International Conference on Computational Collective Intelligence (pp. 363-376). Springer, Cham. (2023).
9. Hewage, U., Pears, R., & Naeem, M. A.: Optimizing the trade-off between classification accuracy and data privacy in the area of data stream mining. Int J Artif Intell, 1(1), 147-167 (2022).
10. Mohasseb, A., Aziz, B., Jung, J., & Lee, J.: Predicting CyberSecurity Incidents using Machine Learning Algorithms: A Case Study of Korean SMEs. In ICISSP (pp. 230-237) (2019, February).
11. Naseriparsa, M., Al-Shammari, A., Sheng, M., Zhang, Y., & Zhou, R. RSMOTE: improving classification performance over imbalanced medical datasets. Health information science and systems, 8, 1-13(2020).
12. Niranjan, A., Nitish, A., Deepa Shenoy, P., & Venugopal, K. R.: Security in data mining-a comprehensive survey. Global Journal of Computer Science and Technology, 16(5) (2016).
13. Pfisterer, F., Beggel, L., Sun, X., Scheipl, F., & Bischl, B.: Benchmarking time series classification--Functional data vs machine learning approaches. arXiv preprint arXiv:1911.07511.3(2019).
14. Pramod, D. (2023). Privacy-preserving techniques in recommender systems: state-of-the-art review and future research agenda. Data Technologies and Applications, 57(1), 32-55.
15. Reza, K. J., Islam, M. Z., & Estivill-Castro, V. : Privacy Preservation of Social Network Users Against Attribute Inference Attacks via Malicious Data Mining. In ICISSP (pp. 412-420) . (2019, February).
16. Sangaroonsilp, P., Choetkiertikul, M., Dam, H. K., & Ghose, A.: An empirical study of automated privacy requirements classification in issue reports. Automated Software Engineering, 30(2), 20(2023).
17. Selvarajan, S., Srivastava, G., Khadidos, A. O., Khadidos, A. O., Baza, M., Alshehri, A., & Lin, J. C. W.. An artificial intelligence lightweight blockchain security model for security and privacy in IIoT systems. Journal of Cloud Computing, 12(1), 38(2023).
18. Zhang, C.:Design of a cryptographic simulation system for classification privacy based on blockchain technology. In International Conference on Cyber Security, Artificial Intelligence, and Digital Economy (CSAIDE 2022) (Vol. 12330, pp. 100-105). SPIE(2022, August).
19. Zhou, L., & Bian, X.: Improved text sentiment classification method based on BiGRU-Attention. In journal of physics: conference series (Vol. 1345, No. 3, p. 032097). IOP Publishing. (2019, November).
20. Choi, S. H., Shin, J. M., & Choi, Y. H.: Dynamic nonparametric random forest using covariance. Security and Communication Networks, 2019, 1-12(2019).