UDC 004.8 **doi:** https://doi.org/10.20998/2522-9052.2022.1.02

Viktor Chelak, Svitlana Gavrylenko

National Technical University "Kharkiv Polytechnic Institute, Kharkiv, Ukraine

METHOD OF COMPUTER SYSTEM STATE IDENTIFICATION BASED ON BOOSTING ENSEMBLE WITH SPECIAL PREPROCESSING PROCEDURE

Abstract. **The subject of the research** is methods of identifying the state of the Computer System. **The object of research** is the process of identifying the state of a computer system for information protection. **The aim of the research** is to develop the method for identifying the state of a computer system for information protection. This article is devoted to the development of method (boosting ensemble) to increase the accuracy of detecting anomalies in computer systems. **Methods used**: artificial intelligence methods, machine learning, decision tree methods, ensemble methods. **The results were obtained**: a method of computer system identification based on boosting ensemble with special preprocessing procedure is developed. The effectiveness of using machine learning technology to identify the state of a computer system has been studied. Experimental researches have confirmed the effectiveness of the proposed method, which makes it possible to recommend it for practical use in order to improve the accuracy of identifying the state of the computer system. **Conclusions.** According to the results of the research, ensemble classifier of computer system state identification based on boosting was proposed. It was found that the use of the proposed classifier makes it possible to reduce the variance to 10%. In addition, due to the optimization of the initial data, the efficiency of identifying the state of the computer was increased. Prospects for further research may be to develop an ensemble of fuzzy decision trees based on the proposed method, optimizing their software implementation.

Keywords: computer system state identification; data processing; decision tree boosting ensembles.

Introduction

In the modern world, the task of information security is one of the highest priority goals. It's all to blame for the losses that any technology area can incur. These losses can affect even the most distant areas from information technology. This task is becoming more difficult every year and is demanding on the accuracy and speed of performance of methods for detecting threats and their prevention. [1]

One of the subtasks of our scientific direction is the identification of the state of a computer system. A computer system can be defined by a large number of features: the workload of various components, the amount of processed memory per second, etc. If we add to this a huge number of processes that are performed in processor races, we get a gigantic set of data, factors and conflicting information that needs to be processed. In order to find our potential features, it is necessary to analyze our data and make optimization according to criteria, perform preliminary data processing. This approach will improve our training set to achieve low errors on machine learning methods.

The aim of the research is to develop method for identifying the state of a computer system using an algorithm for pre-processing the original data.

Analysis of related works. In [2], a comparative analysis was carried out on various applied problems using ensemble methods: Gradient Boosting, Extreme Gradient Boosting, etc. The results of this considered work were good enough for boosting algorithms. The work [3] presents a method aimed at solving applied problems in the field of medicine and diagnostics. However, while this model works perfectly in tasks related to signals and images, for non-uniform data (such as in the task of intrusion detection) it showed a high variance error. There are also interesting solutions to the problems of classifying web pages using boosting algorithms [4]. Unfortunately, the presented models work poorly on data with weak correlations, while only a small amount of characteristics of a computer system have strong correlations. Therefore, this is a rather high disadvantage for tasks that are considered by the field of information protection. The article [5] proposes a boosting method, specifies the optimization problem for a special procedure for updating the weights of the classifiers in boosting. The method is highly accurate, but consumes a lot of resources for recognition. Given that the problem of identifying the state of the system requires frequent analysis and monitoring in almost real time, the method does not fit the class of problems under consideration due to performance. A very specific one was proposed in [6] based on regression boosting algorithms. This method does an excellent job with tasks that work online (in real time). But it also has limitations (in order to protect against overfitting), which makes it is impossible to obtain high accuracy on the test sample.

As a result of the analysis of relative work, one can draw attention to the fact that each method focuses on a specific task and criteria to the detriment of other criteria. The class of boosting algorithms is able to solve the problem of classifying the state of a computer system in order to detect threats, but it needs modification.

Formulation of the problem. Build a model which is capable of making multiple classification of the state of a computer system. Basic requirements for such a model:

- The bias error must be 0%.
- The variance error must not exceed 10%.
- For cases of binary classification, the system output "-1" means the normal state of the computer system and "1" - an abnormal state. For multiple classifications, anything other than zero means a specific class of threat, anomaly, intrusion, malware family, etc.
- The speed of classifying the state of the constructed model should not exceed 1 second. Otherwise, the constructed model will not be able to process data in real time.

- The training time is not strictly limited. However, it should consider the possibility of adding new data during the week. (behavioral models of zero-day vulnerabilities).
- Ability to update weights and new factors.
- The developed system should not affect more than 10% of the operating system resources. (RAM usage, CPU time, etc.).
- A special data preprocessing procedure is required to remove anomalies and noise in the training set.

The input data that are described in the training and test samples are presented in Table 1.

The input data were examined using statistical characteristics such as variance, mathematical expectation, standard deviation. A comparison was also made between the characteristics for the test sample and the training sample to balance the data. One of the main characteristics for machine learning, the correlation coefficients is presented in Table 2.

As a result, the input data in a large number of cases turned out to have a very low correlation (0.0, 0.1). The training set also contains highly correlated data obtained as a result of the simplest operations on some criteria. For example, the calculation of the total CPU workload is the usual sum of four other criteria.

Table 1 **– Lists of input data that are used to describe a computer system**

The code	Name of Input data	Description
A	C0 Percent Disk Read Time	System volume (Read only)
B	C0 Percent Disk Write Time	System volume (Write only)
C	C0 Percent Idle Time	System volume Idle time
D	C0 Percent Disk Time	System volume total time (100% - *C* or *A* + *B*)
E	D1 Percent Disk Read Time	Other volumes (Average, Read)
F	D1 Percent Disk Write Time	Other volumes (Average, Write)
G	D1 Percent Idle Time	Other volumes (Average, Idle)
H	D1 Percent Disk Time	Other volumes ("*E* + *F*")
I	Total Percent Disk Read Time	All volumes (Cumulative, Read)
J	Total Percent Disk Write Time	All volumes (Cumulative, Write)
K	Total Percent Idle Time	All volumes (Cumulative, Idle)
L	Total Percent Disk Time	All volumes (Cumulative, R+W)
M	CPU0 Percent Processor Time	First CPU core workload
N	CPU1 Percent Processor Time	Second CPU core workload
O	CPU2 Percent Processor Time	Third CPU core workload
P	CPU3 Percent Processor Time	Fourth CPU core workload
Q	CPU Total Percent Processor Time	Overall CPU workload
R	Ethernet Bytes Sent	Total number of bytes sent
S	Ethernet Bytes Received	Total number of bytes received
T	Free RAM Memory	Unused RAM
U	Calculated Percentage Usage	Difference between RAM size and *T*

Table 2 **– Correlation coefficients of input data**

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
A	-	0.00	-0.66	0.98	0.00	0.00	0.01	0.00	0.44	0.00	-0.55	-0.43	-0.04	-0.04	-0.04	-0.04
B	0.00	-	-0.69	0.20	0.00	0.02	-0.03	0.00	0.00	0.92	-0.58	0.09	-0.04	-0.03	-0.04	-0.04
C	-0.66	-0.69	-	-0.79	-0.01	-0.05	0.11	-0.01	-0.30	-0.65	0.88	-0.36	0.08	0.08	0.08	0.08
D	0.98	0.20	-0.79	-	0.00	0.00	0.00	0.00	0.43	0.19	-0.65	0.44	-0.04	-0.04	-0.05	-0.05
E	0.00	0.00	-0.01	0.00	-	0.10	-0.27	1.00	0.90	0.04	-0.14	0.90	0.00	-0.01	0.00	-0.01
F	0.00	0.02	-0.05	0.00	0.10	-	-0.68	0.14	0.08	0.42	-0.37	0.13	0.04	0.03	0.04	0.03
G	0.01	-0.03	0.11	0.00	-0.27	-0.68	-	-0.30	-0.24	-0.30	0.57	-0.27	-0.03	-0.02	-0.03	-0.04
H	0.00	0.00	-0.01	0.00	1.00	0.14	-0.30	-	0.90	0.05	-0.16	0.90	0.00	-0.01	0.00	-0.01
I	0.44	0.00	-0.30	0.43	0.90	0.08	-0.24	0.90	-	0.03	-0.37	0.99	-0.02	-0.03	-0.02	-0.02
J	0.00	0.92	-0.65	0.19	0.04	0.42	-0.30	0.05	0.03	-	-0.68	0.13	-0.02	-0.02	-0.02	-0.02
K	-0.55	-0.58	0.88	-0.65	-0.14	-0.37	0.57	-0.16	-0.37	-0.68	-	-0.43	0.05	0.05	0.05	0.05
L	-0.43	0.09	-0.36	0.44	0.90	0.13	-0.27	0.90	0.99	0.13	-0.43	-	-0.02	-0.03	-0.02	-0.03
M	-0.04	-0.04	0.08	-0.04	0.00	0.04	-0.03	0.00	-0.02	-0.02	0.05	-0.02	-	0.97	0.97	0.97
N	-0.04	-0.03	0.08	-0.04	-0.01	0.03	-0.02	-0.01	-0.03	-0.02	0.05	-0.03	0.97	-	0.97	0.97
O	-0.04	-0.04	0.08	-0.05	0.00	0.04	-0.03	0.00	-0.02	-0.02	0.05	-0.02	0.97	0.97	-	0.97
P	-0.04	-0.04	0.08	-0.05	-0.01	0.03	-0.04	-0.01	-0.02	-0.02	0.05	-0.03	0.97	0.97	0.97	-

1. Preprocessing, decision trees and boosting ensemble for identifying the state of the computer system

A special data preprocessing procedure can be represented in the form of 4 parts:

1. Processing conflicting information.
2. Handling missing values.
3. Handling anomalous values (strong outliers).
4. Noise handling (weak outliers).

Conflicting information means two or more samples that are completely coinciding in all criteria and

at the same time have different output classes. No such samples were found in the training set used in this study. However, it is necessary to take into account the fact that the data is obtained in real time and the likelihood of collisions is quite possible.

The way to deal with such data is quite simple - to estimate the probabilities. If conflicting information tends in most cases (80% or more) to one of the classes, delete the data that does not include this sample in this class. If the data does not have such a single class or, in the worst case, is distributed across classes, such data must be deleted.

In the process of monitoring the indicators of a computer system, a failure may occur, due to which some criteria in the sample are lost. An example would be hard disk load values. At the time of the request for information, the system did not have access to it, which is why an empty string or NaN was returned. In the case of training, such data is initially not allowed in the training set. In any case, the system must correctly process the input data in the classification mode. It is necessary to store and calculate the average value of each criterion, and in case of its absence, supply the average value to the model instead of an empty datum. This approach will help to avoid a strong distortion of statistical characteristics (the mathematical expectation will remain the same).

Strong and weak outliers can be found and removed from the training set using machine learning methods such as One-class support vector machines, DBSCAN, etc. The proposed system uses several methods at once to detect anomalies. In addition to those already listed, a standard deviation (coefficient is 3.0) is used to detect strong outliers or anomalies. The standard deviation is calculated for each criterion and only if many criteria are outside the specified range - the sample is considered abnormal and removed. As a result of such cleaning, about 15-25% of the data will be eliminated.

In addition to preprocessing the samples, the proposed solution analyzes the criteria themselves for optimization. The criteria are not considered in case a large number of high absolute values of the correlation coefficients (0.8 or higher). Criteria for which the variance value is close to zero will also be excluded, as such criteria are insignificant. The data is also being scaled to the normalized range (0, 1). If such normalization results in a low variance, the data are scaled in the range from 0 to 10n, where n starts at 1 and is increased until the variance has an acceptable value or the criteria is detected as insignificant and deleted.

Any ensemble has an ordinary element. To solve the problem of identifying the state of a computer system, decision trees are used as an ordinary element. Decision tree is a special case of binary trees and has nodes of two types: leaf or result node and internal or decision-making node. The leaf node is a kind of an exit point from the tree, the result of the model's work, its response. The decision-making node tests one of the sample criteria for equality or exceeding a threshold. When the test succeeds, such a node directs the sample to the next layer (right branch), which can be either the next decisive element or the resulting one. Otherwise the sample is directed to the left branch, which can also be a node of any type. An example of a tree that stands for an ensemble is shown in Fig. 1. It should also be noted that, in contrast to standard decision trees, the proposed method has a weighted accuracy estimate (1). This allows such a tree to be used as a boosting element.

$$E = \sum_{i=1}^{N} w_i [y_i \neq t_i], \quad (1)$$

where w_i – weight of i-th training sample; y_i – actual classification result for i-th training sample; t_i – required classification result for i-th training sample; N – number of samples; $[y_i \neq t_i]$ – indicator function (2).

$$[y_i \neq t_i] = \begin{cases} 1, y_i \neq t_i; \\ 0, y_i = t_i. \end{cases} \quad (2)$$

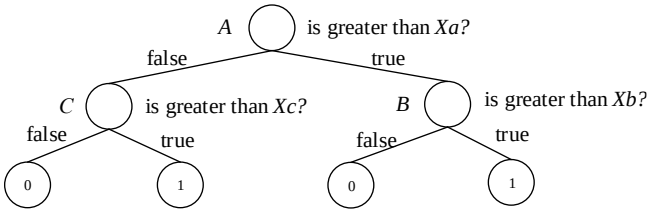

Fig. 1. Example of decision tree

The algorithm for constructing a decision tree can be described by the following actions:

1. Pick a criterion.
2. Find the minimum and maximum values in the training sample. (left and right border)
3. Find the average value between the maximum and minimum values and assign it to the threshold point of the criterion.
4. If the error on both branches is not high, go to step 6.
5. Determine in which branch the error is greater. Find the average between the threshold and the border of the side of the branch and assign it to the threshold. Assign the old value of the threshold to the opposite border. Go to step 4.
6. Repeat steps 1-5 for all criteria.
7. Choose the best criterion (with the minimum value of the error function).
8. Recursively repeat steps 1-7 for the branch with larger error (unless the error on both branches is zero). Do the same for the other branch, unless its error is zero.
9. If the current element is the root element and the error function is zero, the deci-sion tree is built.

The procedure for building a decision tree is rather complicated, because it pro-vides for work with heterogeneous data. In addition, such trees can be controlled using a fitness function and parameters such as maximum tree depth, programmable stop condition, etc.

The structure of the proposed boosting algorithm is shown in Fig. 2. The main idea behind boosting is the principle of building such an ensemble. The construction of classifiers occurs sequentially. Initially, equal weighting factors are assigned to all samples. Then, for

samples that the tree has not learned to recognize correctly or has spent too many decisive elements, the weighting factors increase, while for other samples, they decrease.

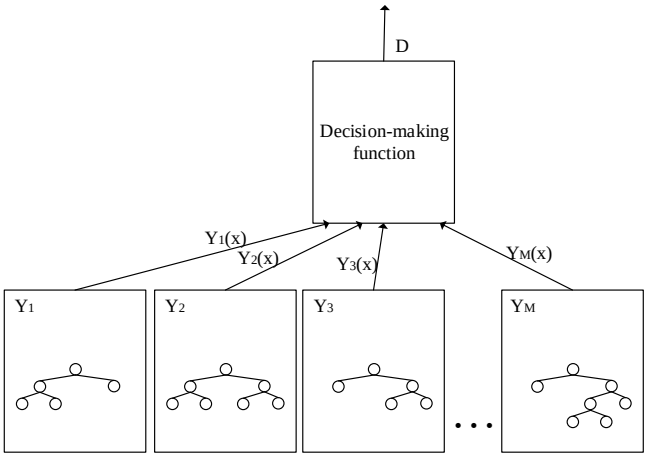

Fig. 2. The structure of the constructed model of the boosting ensemble

Each tree in boosting has its own weight coefficient, which depends on the error rate (3):

$$e_r = E \Big/ \sum_{i=1}^{N} w_i \,, \tag{3}$$

where e_r – error rate of *r*-th classifier; E – weighted accuracy estimate.

Expression for classifier weight (4) is used for the final decision-making based on the decisions of all elements of the ensemble.

$$\varsigma_r = \ln\left(\left(1-e_r\right)/e_r\right), \tag{4}$$

where ς_r – weight of *r*-th classifier.

To build the next tree, the weights of the training sample are adjusted (5):

$$w_i^{r+1} = w_i^r e^{\varsigma_r [y_i \neq t_i]}, \tag{5}$$

where w_i^{r+1} – weight of *i*-th training sample for next decision tree; w_i^r – weight of *i*-th training sample for current decision tree.

Ensemble decision making for the case of binary (6) and multiple (7) classification is presented below.

$$D = sign(\sum_{r=1}^{M} \varsigma_r y_r(x)), \tag{6}$$

where D – decision of boosting ensemble (complex result of ensemble); M – number of learners in boosting; $y_r(x)$ – result of *r*-th decision tree (classifier); x – vector of input values.

$$D_k = \sum_{r=1}^{M} \varsigma_r \left[y_r(x)=k \right]; D = k,\ D_k = \max_k \left(D_k\right), \tag{7}$$

where D_k – weighted decision for *k*-th class; $[y_r(x) = k]$ – indicator function that checks the equality of the solution of the *r*-th classifier and a specific class *k*; k – class number.

As can be seen from expression (6), in the case of a binary classification for the normal and anomalous state of a computer system, one can make a label for classes - 1 and 1, respectively, and using the sign function to obtain the final answer. If this is a multiple classification (7) – for each class, the sum of the weight coefficients of the classifiers, which recognize the input data as this class, is considered. The resulting decision in this case is determined by the plurality principle (the class with the highest sum of weights is the result) [7].

2. Analysis of results and comparison with standard models

A software prototype of the proposed model was developed. Fig. 3 shows a part of the tree, which is an element of the boosting ensemble. In order to check the effectiveness of the approach, it is necessary to compare not only the proposed method with the classical ones. But also make a comparison with data preprocessing (optimization of criteria and removal of inconsistent data, filling in the gaps in information, etc.) and without. The main characteristics of the methods are bias error value, variance error value, prediction speed and time spent on training.

As part of the comparison, we use not simple standard models of machine learning, but optimal ones (with the best parameters to maximize results). Such methods are: Optimized Decision Tree, Linear Discriminant, Quadratic Discriminant, Logistic Regression, Optimized SVM, Optimized Gaussian SVM, Optimized K-Nearest Neighbors, Weighted KNN, Random Forest, Ensemble of Sub-space KNN and Optimized AdaBoost.

Fig. 4 shows a comparative histogram for bias error. As you can see from the histogram, the special preprocessing procedure made a positive effect on the bias error not only for the proposed method, but also for most of the others, due to the removal of anomalies in the data. The quadratic discriminant method diverges on both normal and optimized data. And also the methods of linear discriminant and optimal AdaBoost were able to converge on sets with preprocessing. The proposed method (the rightmost bars in Fig. 4) received the minimum error on the training sample without preprocessing (only 1 sample was not recognized correctly) and after preprocessing it showed an error of 0%, which meets the requirements (the sample was removed, and of all the data it was the most anomalous sample).

Fig. 5 shows a comparison of the methods for variance error. The variance error is the absolute value of the difference between the bias error and the error on the test dataset (which were not used in training). The worst result was shown by an ensemble of KNN subspaces (over 50%). The rest of the methods have clearly improved the situation with preprocessing. Only optimized KNN, weighted KNN, logistic regression and random forest got errors less than 10%. However, some of them had a non-zero bias error - which is also critical for the system of identifying the state of a computer system. The proposed method showed low errors after preprocessing (0.17%).

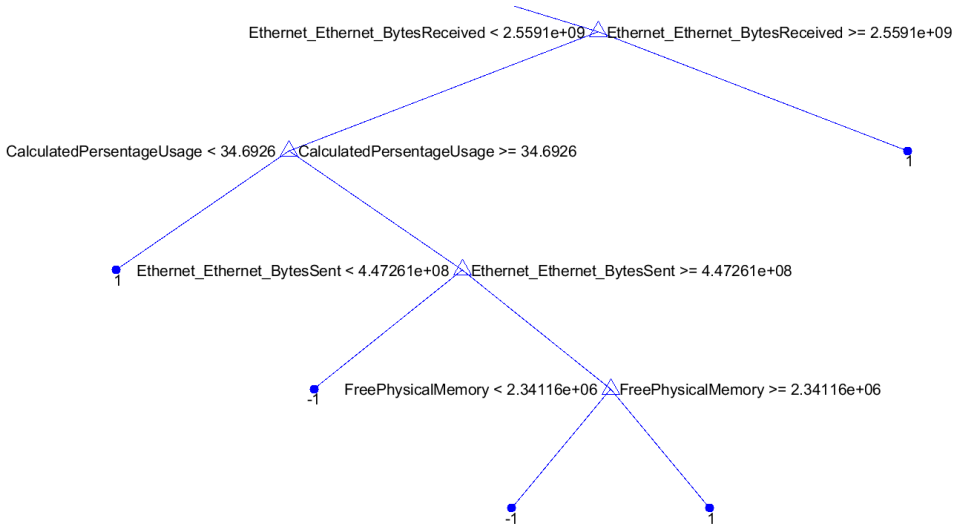

Fig. 3. Part of the built decision tree for the boosting ensemble (binary classification)

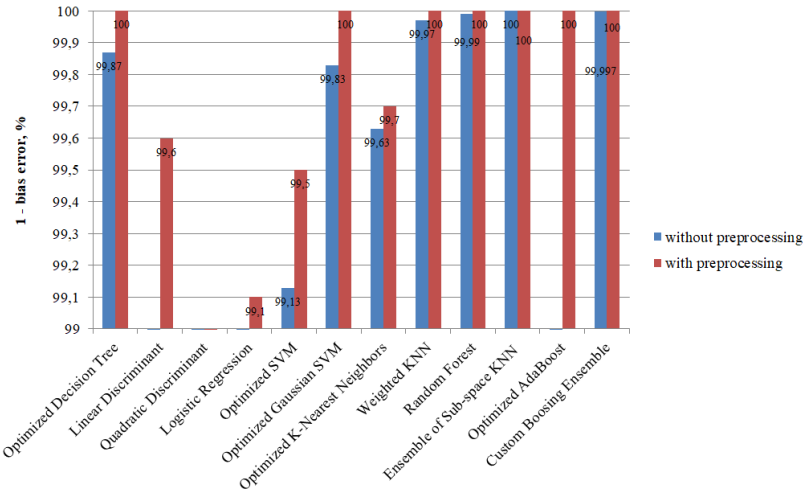

Fig. 4. Accuracy comparison on training data (100% - bias error)

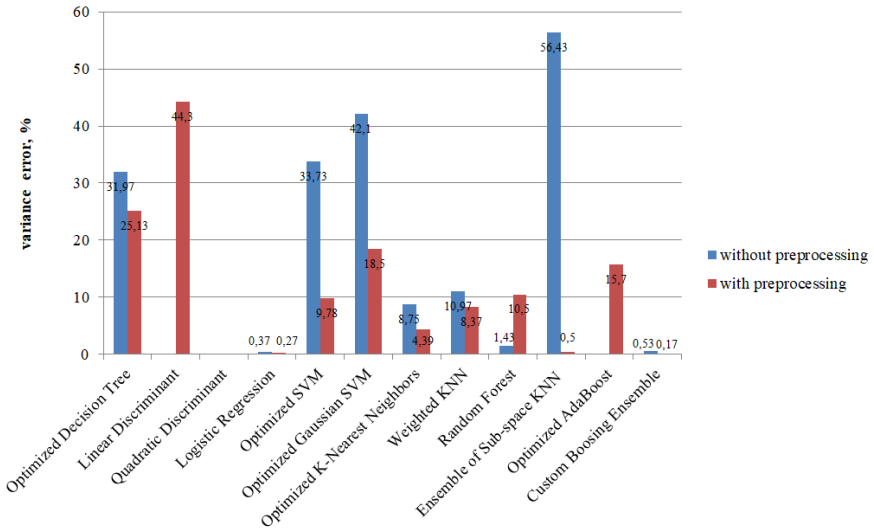

Fig. 5. Variance error comparison

Comparing the performance (Figure 6), it can be seen that this method is inferior to the standard methods in prediction rate. However, the requirement for performance (prediction rate is more than 1 sample per second) and resource costs are met. This prediction speed is sufficient to carry out identification of the state of a computer system in real time. You should pay attention to two main points. First, the data presented in Fig. 6 are estimates, and are not accurate. Secondly, the proposed method has a limitation on the use of processor time, when other methods could use the processor at maximum load.

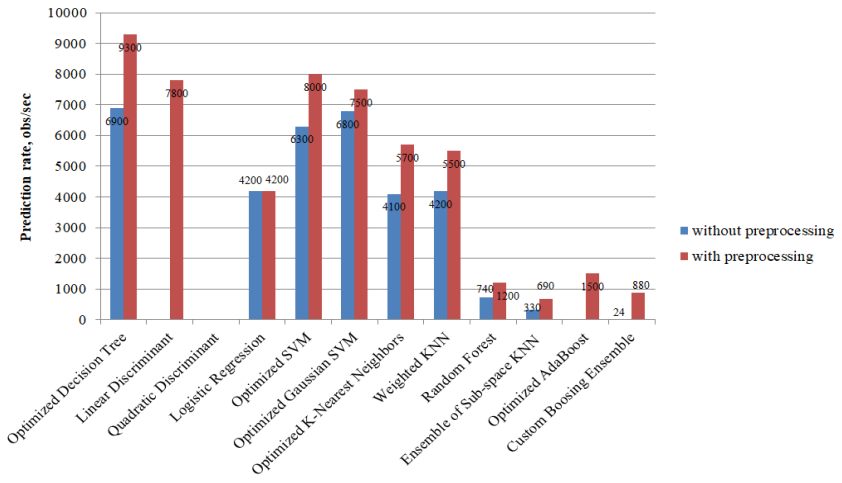

Fig. 6. Estimated prediction rate comparison

The last comparison was made for an insignificant characteristic - the training time (Figure 7). It should be noted that in this work there is no comparative analysis of the time for additional training. (in these situations, the training time is much shorter).

Unlike the prediction speed, the training time is an exact value. By using the preprocessing procedure, which reduced the amount of data and the load on the model building method, it was possible to accelerate training by ~ 25 times.

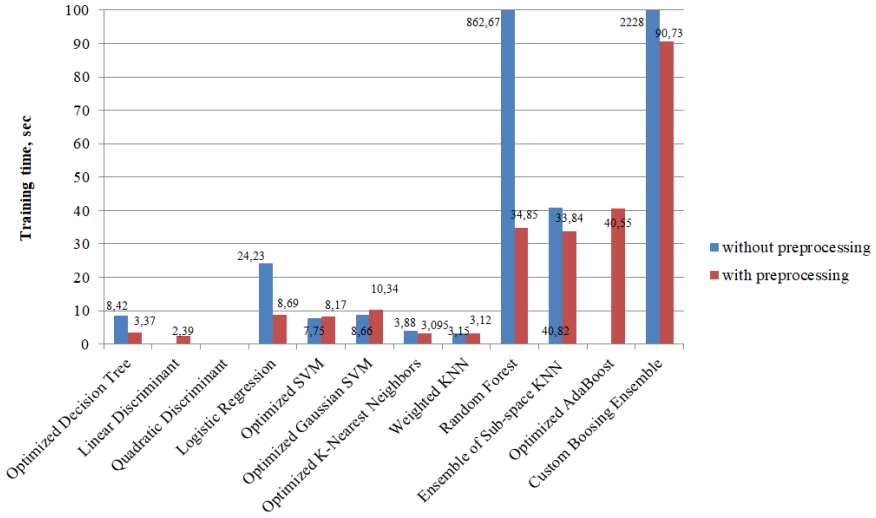

Fig. 7. Training time comparison

Conclusion

According to the results of the research, ensemble classifier of computer system state identification based on boosting was proposed. It was found that the use of the proposed classifier makes it possible to reduce the variance to 10%. In addition, due to the optimization of the initial data, the efficiency of identifying the state of the computer was increased. The practical significance lies in the fact that the developed method is implemented in software and researched during the solution of the real problem of identifying the state of a computer system. The experiments confirmed the efficiency of the proposed method, which makes it possible to recommend it for practical use as an express method of analysis of the state of the computer system. Prospects for further research may be to develop an ensemble of fuzzy decision trees based on the proposed method, optimize its software implementation and improve the quality of classification.

REFERENCES

1. (2020), “Inter American Development Bank and Organization of American States”, *Cybersecurity Risks, Progress, and the Way Forward in Latin America and the Caribbean*, pp. 19-38, doi: http://dx.doi.org/10.18235/0002513.
2. Kumkar, P., Madan, I., Kale, A., Khanvilkar, O. and Khan ,A. (2018), “Comparison of Ensemble Methods for Real Estate Appraisal”, *3rd International Conference on Inventive Computation Technologies* (ICICT), Coimbatore, India, pp. 297-300, doi: http://dx.doi.org/10.1109/ICICT43934.2018.9034449.
3. Lee, S., Son, C., Albertini, M. K. and Fernandes, H. C. (2020), “Multi-Phases and Various Feature Extraction and Selection Methodology for Ensemble Gradient Boosting in Estimating Respiratory Rate”, *IEEE Access*, Vol. 8, pp. 125648-125658, doi: http://dx.doi.org/10.1109/ACCESS.2020.3007524.

4. Dutta, J., Kim, Y. W. and Dominic, D. (2020), “Comparison of Gradient Boosting and Extreme Boosting Ensemble Methods for Webpage Classification”, *Fifth International Conference on Research in Computational Intelligence and Communication Networks* (ICRCICN), Bangalore, India, pp. 77-82. doi: http://dx.doi.org/10.1109/ICRCICN50933.2020.9296176.
5. Li, Z. and Wang, D. (2020), “Classification on Point-cloud of Shoe-last Curvature using Weight-updated Boosting based Ensemble Learning”, *IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, Chongqing, China, , pp. 2073-2077, doi: http://dx.doi.org/10.1109/ITAIC49862.2020.9338947.
6. Mirza, H. (2018), “Online boosting algorithm for regression with additive and multiplicative updates”, *26th Signal Processing and Communications Applications Conference (SIU)*, Izmir, pp. 1-4, doi: http://dx.doi.org/10.1109/SIU.2018.8404455.
7. Chelak, V. and Gavrylenko, S. (2021), “Development of Computer State Identification Method Based on Boosting Ensemble”, *Fifth International Scientific and Technical Conference "Computer and Information Systems And Technologies"*, Kharkiv, pp.53-54, doi: http://dx.doi.org/10.30837/csitic52021232208.

Received (Надійшла) 22.11.2021
Accepted for publication (Прийнята до друку) 26.01.2022

ВІДОМОСТІ ПРО АВТОРІВ / ABOUT THE AUTHORS

Челак Віктор Володимирович – аспірант кафедри «Обчислювальна техніка та програмування», Національний технічний університет “Харківський політехнічний інститут”, Харків, Україна;
Victor Chelak – PhD Student of Department of "Computer Engineering and Programming", National Technical University «Kharkiv Polytechnic Institute», Kharkiv, Ukraine;
e-mail: victor.chelak@gmail.com, Orcid ID: https:// orcid.org/0000-0001-8810-3394.

Гавриленко Світлана Юріївна – доктор технічних наук, професорка, професорка кафедри обчислювальної техніки та програмування, Національний технічний університет “Харківський політехнічний інститут”, Харків, Україна;
Svitlana Gavrylenko – Doctor of Technical Sciences, Professor, Professor of Department of "Computer Engineering and Programming", National Technical University «Kharkiv Polytechnic Institute», Kharkiv, Ukraine;
e-mail: gavrilenko08@gmail.com, Orcid ID: https://orcid.org/ 0000-0002-6919-0055.

Метод ідентифікації стану комп'ютерної системи на основі boosting ансамбля з спеціальною процедурою передобробки даних

В. В. Челак, С. Ю. Гавриленко

Анотація. Предметом дослідження є методи визначення стану комп'ютерної системи**. Об'єктом** дослідження є процес ідентифікації стану комп'ютерної системи для захисту інформації. **Метою** дослідження є розробка методу ідентифікації стану комп'ютерної системи для захисту інформації. Ця стаття присвячена розробці методу (*boosting* ансамбль) для підвищення точності виявлення аномалій в комп'ютерних системах. **Методи, що використовуються**: методи штучного інтелекту, машинне навчання, методи дерев рішень, ансамблеві методи. **Отримано результати**: розроблено метод ідентифікації комп'ютерних систем на основі бустингового ансамблю зі спеціальною процедурою передобробки. Отримано оцінку ефективності використання методів машинного навчання визначення стану комп'ютерної системи. Експериментальні дослідження підтвердили ефективність запропонованого методу, що дає змогу рекомендувати його для практичного використання з метою підвищення точності ідентифікації стану комп'ютерної системи. **Висновки**. За результатами дослідження запропоновано ансамблевий класифікатор ідентифікації стану комп'ютерної системи на основі бустингу. Встановлено, що використання запропонованого класифікатора дозволяє знизити помилку *variance* до 10%. Крім того, за рахунок оптимізації вихідних даних підвищено швидкість ідентифікації стану комп'ютерної системи. Перспективами подальших досліджень може бути розробка ансамблю нечітких дерев рішень на основі запропонованого методу, оптимізація їхньої програмної реалізації.

Ключові слова: ідентифікація стану комп'ютерної системи, опрацювання даних, boosting ансамблі дерев рішень.

Метод идентификации состояния компьютерной системы на основе boosting ансамбля с специальной процедурой предобработки данных

В. В. Челак, С. Ю. Гавриленко

Аннотация. **Предметом** исследования являются методы определения состояния компьютерной системы. **Объектом** исследования является процесс идентификации состояния компьютерной системы для защиты информации. **Целью** исследования является разработка метода идентификации состояния компьютерной системы для защиты информации. Данная статья посвящена разработке метода (*boosting* ансамбль) для повышения точности обнаружения аномалий в компьютерных системах. **Используемые методы**: методы искусственного интеллекта, машинное обучение, методы деревьев решений, ансамблевые методы. **Получены результаты**: разработан метод идентификации компьютерных систем на основе бустингового ансамбля со специальной процедурой предобработки данных. Получена оценка эффективности использования методов машинного обучения для определения состояния компьютерной системы. Экспериментальные исследования подтвердили эффективность предложенного метода, что позволяет рекомендовать его для практического использования с целью повышения точности определения состояния компьютерной системы. **Выводы**. По результатам исследования предложен ансамблевый классификатор идентификации состояния компьютерной системы на основе бустинга. Установлено, что использование предложенного классификатора позволяет снизить ошибку *variance* до 10 %. Кроме того, за счет оптимизации исходных данных повышена скорость определения состояния компьютерной системы. Перспективами дальнейших исследований может быть разработка ансамбля нечетких деревьев решений на основе предложенного метода, оптимизация их программной реализации.

Ключевые слова: идентификация состояния компьютерной системы, обработка данных, *boostin*g ансамбли деревьев решений.