Comparative analysis of anomaly recognition methods in real time

Nikita V. Gololobov[1], Konstantin E. Izrailov[2,3], and Igor V. Kotenko[3*]
[1]Peter the Great St.Petersburg Polytechnic University
Saint-Petersburg, Russia
neptu133@gmail.com
[2]The Bonch-Bruevich Saint-Petersburg state university of telecommunications
Saint-Petersbur, Russia
konstantin.izrailvo@mail.ru
[3]St. Petersburg Federal Research Center of the Russian Academy of Sciences
Saint-Petersburg, Russia
ivkote@comsec.spb.ru

Abstract

The article discusses modern classes of algorithms used to detect anomalies in data streams: sliding window algorithm, metric algorithms, predictive-based algorithms, and algorithms based on hidden Markov models. During the research, it was possible to determine functional and efficiency criteria for assessing the class of algorithms and then comparing it with other considered classes. In addition, for each class of methods, strengths and weaknesses are given, the scope is described, and a generalized example of implementation in the form of pseudo code is given. The use of this approach makes it possible to cover entire groups of algorithms without reference to a specific implementation. The conclusions obtained as a result of the research can be applied solving problems of optimizing the process of detecting anomalies or increasing the efficiency of applied solutions used in these scenarios. The resulting calculations allow further development and optimization of methods in this area for unlabeled fixed data sets.

Keywords: anomaly, method, real time, algorithms, efficiency, predictions, sliding window, hidden Markov models

1 Introduction

Detecting anomalies is an important task in the field of information technology, since a huge number of modern processes operate on the basis of Big Data. Thus, outliers in this data can be accidental or malicious, which makes it necessary to detect (and neutralize) them. In particular, in machine learning problems, the purity and correctness of the data sets used to train the model are important criteria [1].

Often, in order to save time and resources, the model is trained with data sets that do not have a fixed size. With this approach, at each next moment of time, a new instance arrives at the input of the model, while the training time does not have a fixed value, and the data cannot be pre-labeled (for example, when identifying network insiders of information systems [2]).

Detecting anomalies in the data in real time, requires additional modification of the methods used for training on a fixed set [3, 4]. This is due to the complexity of the task and the lack of control over the incoming data for analysis.

Research Briefs on Information & Communication Technology Evolution (ReBICTE), Vol. 7, Article No. 10 (October 15, 2021)
DOI:*10.22667/ReBiCTE.2021.10.15.010*

*Corresponding author: St. Petersburg Federal Research Center of the Russian Academy of Sciences, 39, 14th Linija, St. Petersburg, Russia, Tel:+79217504307

To identify anomalies in real time, the following methods are used: sliding window algorithm, metric algorithms, prediction-based algorithms, algorithms based on hidden Markov models. To select a suitable algorithm, a comprehensive comparison of them according to various criteria is required. This issue is considered relevant when analyzing Big Data [5, 6, 7, 8].

In terms of a sequential unlimited data set, time series are considered, which in fact are analogs of instances from the sample. Modification and optimization of existing methods is conditioned by this very transition. Further each of the algorithms is described, indicating the areas of application, advantages and disadvantages, and also indicating its possible implementation (in the form of pseudo code).

2 Algorithm$_1$ – Sliding window algorithm

The classical basic method for detecting anomalies in the data stream is the sliding window algorithm, which is used for time series [9].

In the course of its work, the algorithm divides the sequence into a certain finite number of sub sequences – windows. The size of the window should be less than the length of the row itself. This allows you to capture anomalies while sliding. The search for anomaly itself is carried out in the course of sliding the window over the entire time series with a certain step, the size of which is less than the size of the window. The usage of this algorithm is shown in Figure 1.

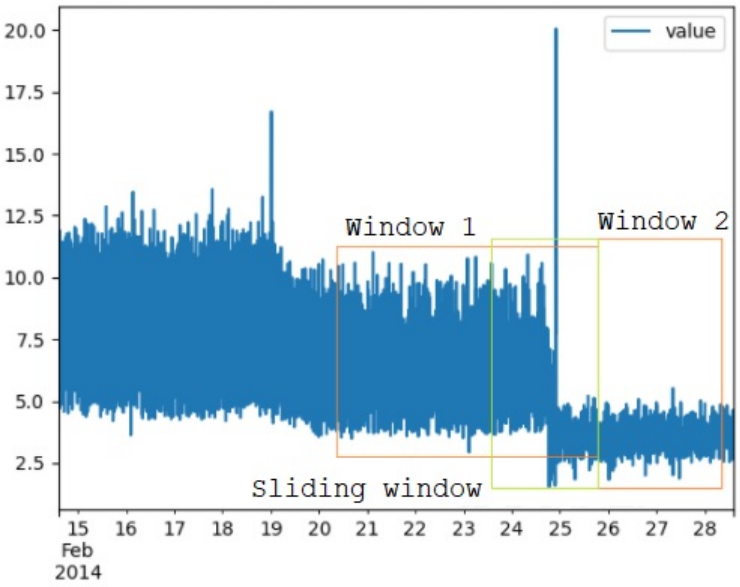

Figure 1: Example of applying the sliding window algorithm

2.1 Application area

The application of this method is possible in systems that do not require a complete set of data. In systems with unsupervised recognition mode, the use of the sliding window algorithm can lead to an increase in the rate of inaccurate detections. This is due to the analyzed sample – calculations will be performed only for instances located in the window area.

When using the sliding window algorithm in systems with preliminary model building, additional modifications to the algorithm are not required. At the recognition stage, each instance can be analyzed separately, which greatly reduces the number of errors of the first and second kind.

2.2 Method description

The anomaly score of the test time series is calculated by aggregating the anomaly scores of its windows. Formally, the sliding window algorithm can be described as follows:

1. Based on the data for training, straining, consisting of the sets S_1, S_2, ..., S_n, p windows of each time series S_i are extracted, where the number of windows is calculated as $\mid S_i \mid + m - 1$ when shifting a window of size m. Similarly, for target data S_test, consisting of data sets T_1, T_2, ..., T_n, each test time series T_i is divided by $\mid T_i \mid + m - 1$ window.

2. For each test window A (t_{ij}), the anomaly estimate is calculated using its similarity to the learning windows. This similarity function can be Euclidean, Manhattan, or any other distance appropriate to the set of context.

2.3 Method advantages

The most significant advantage is the versatility of the method. Window methods can capture all different kinds of anomalies: point, collective, and contextual. Since the entire time series is split into smaller subsequences, it becomes possible to define an instance as anomalous.

If the entire time series is anomalous, then all subsequences are also anomalous, therefore, window-based methods show high efficiency in conditions of anomalousness of the entire time series as a whole.

2.4 Disadvantages of the method

The disadvantage of sliding window methods is the high requirements for the accuracy of the selection of the window size for the explicit recording of the anomaly. Thus, if the window size is chosen less than the cycle length, the recognition accuracy will be insufficient to obtain satisfactory results of the algorithm operation. At the same time, if you specify a window size larger than the loop, the recognition efficiency will increase, but its efficiency in terms of performance will decrease.

Another drawback of methods based on the use of sliding windows is their algorithmic complexity, which is $O((nl)^2)$, where l is the average length of the time series, n is the number of time series in the data set. Most of the window-based methods are suggested for troubleshooting a chasm detection problem.

2.5 Implementation

When implementing methods, in the general case, both specific data sets and a complete array of data sets can be used as input parameters. The values of the number of training and test windows can also be calculated in advance. The implementation of the comparison function depends on the conditions for detecting the anomaly, as well as the number of classes of anomalies to be detected in Algorithm 1.

3 Algorithm$_2$ – Metric algorithms

The assumption underlying these methods is that anomalous time series differ from ”normal” ones, which can be captured by estimating a measure of proximity [10].

Algorithm 1: Sliding window to identify anomalies in the data

```
Input:
    DatasetCollection – Collection of datasets
    DatasetCount – Number of datasets in collection
    WindowSize – Size of sliding window
    TimeRows – Time rows from testing dataset
    TimeRowsCount – Number of time rows from testing dataset
Output:
    AnomalyList – List of found anomalies
1  begin
2      List<data> Anomalies;
3
       // Step 1:  Calculate train and test window count
4      int TrainWindowCount = DatasetCount + WindowSize - 1;
5      int TestWindowCount = TimeRowsCount + WindowSize - 1;
6
       // Step 2:  Iterate over all possible wi1ndows
7      while TrainWindowCount != 0 do
8          while TestWindowCount != 0 do
               // Step 3:  Calculate diff and add to anomaly list
9              data Anomaly = DetectAnomalyWithDiff(DatasetCollection .GetNextValue(),
                 TimeRows.GetNextValue());
10
11             if Anomaly != 0 then
12                 Anomalies.Append(Anomaly);
13             end
14             TrainWindowCount = TrainWindowCount - 1;
15         end
16         TestWindowCount = TestWindowCount - 1;
17     end
18
19     return Anomalies;
20 end
```

3.1 Application area

Metric algorithm-based methods use pairwise proximity between series to train and recognize anomalies, using a distance kernel or similarity kernel to compute an anomaly estimate of the test time series.

When different types of anomalies are detected, similarity-based methods can show unsatisfactory results, in a situation where a small number of anomalous instances are obscured by “normal” ones. Nevertheless, metric methods can be used to identify collective anomalies and anomalous time series with greater efficiency due to the peculiarities of the functioning of algorithms based on them.

3.2 Method description

The estimate of the target time series anomaly in relation to the series from the training set is calculated using the following techniques:

1. The indicator of anomaly is the distance from the test time series to its k-th nearest neighbor in the training data set.

2. The training time series can be grouped into a certain finite number of clusters, after which it becomes possible to calculate the cluster centroids. In this case, an indicator of the anomaly is the distance between the time series of tests to its closest cluster center.

3.3 Method advantages

This group of methods has a number of advantages that result from the dynamic use of the presence of similarity for two time series. Thus, the transformation matrix can be represented as sparse matrices, thereby reducing the spatial complexity of the algorithms.

In the course of their work, DTW-based algorithms use the optimal path for transforming the time series, which significantly increases the efficiency of detecting anomalous time series. In addition, due to the use of optimal alignment, algorithms from the metric class can be used in combination with other methods.

3.4 Disadvantages of the methods

One of the most obvious drawbacks is the inapplicability of metric methods for series with potential redundancy of matches. As a result, the analysis of such series using DTW can make the algorithm computationally complex and lead to distortion of the actual distance between time series.

Another disadvantage of metric methods is the requirement for time series to be synchronous. Otherwise, attributes such as Euclidean distance and DTW may be out of phase and, in fact, different despite the initial similarity.

3.5 Implementation

Typically, when using metric algorithms, the k-nearest neighbors algorithm is applied. After obtaining the distances, the greatest distance from the center is calculated, which is the right border for detecting anomalies. The left border can be set depending on the conditions under which the detection is performed in Algorithm 2.

4 Algorithm$_3$ – Prediction based algorithms

Anomaly detection methods using predictive models are usually based on searching for individual observations, such as outliers [11, 12].

4.1 Application area

The idea behind these methods is to assume that ”normal” time series are generated from a statistical process, while anomalous time series do not correspond to the parameters of this process. Thus, the general traversal is based on studying the parameters of the observed statistical process from a set of normal time series for training, and then assessing the probability that the target time series has the same or similar characteristics as the series for training.

Algorithm 2: Metric Algorithm for Detecting Anomalies in Data

```
Input:
    Dataset – Testing dataset
Output:
    Anomaly – Anomaly value from testing dataset
1  begin
2      Dictionary<value, distance> Neighbours;
3      int MaximumDistace = 0;
4      int CurrentDistance;
5      data anomaly;
6
       // Step 1:  Get k-nearest neighbors as dictionary
7      CalculateKNN(Dataset, Neighbours);
8
       // Step 2:  Iterate over dictionary to get maximum distance
9      foreach distance in Neighbours.GetDistances() do
10         CurrentDistance = distance;
11
12         if CurrentDistance > MaximumDistance then
13             MaximumDistace = CurrentDistance;
14         end
15     end
16
       // Step 3:  Get anomaly as most distant value
17     Anomaly = GetValueByDistance(MaximumDistance);
18
19     return Anomaly;
20 end
```

4.2 Method description

The operation of prediction-based algorithms consists of the following stages:

1. The model is trained on n consecutive instances to predict the value of $n+1$ instances following it.

2. The model trained as a result of the first stage is applied on the target data set: for all instances, starting from $n+1$, its value is predicted based on the values of previous observations. In this case, there is a forecast error corresponding to the observation, and depending on the difference between the actual and the predicted value of the instance.

The methods differ in the forecasting models used and are traditionally classified as follows:

1. Time series models: moving average (MA), autoregressive model (AR), ARMA, ARIMA, Kalman filters and others. The input data of such models are all time series and the length of the known values - n. The n value is also used to denote the order of the model. These models differ in the type of filters used for forecasting.

2. General regression (excluding time series): linear regression, Gaussian process regression, support vector machine. For this list of models, the input data are non-preprocessing subsequences of a fixed length m - the length of the history. Various kernel functions such as polynomial, RBF and sigmoid can be used as a display function.

4.3 Method advantages

Since the methods described above assume that the sets are generated from a statistical process, provided this statement is true, the prediction-based algorithms perform satisfactorily.

The use of this class of methods implies the possibility of using a dynamic length history. This is applicable in situations where $n+1$ instance values cannot be predicted with a high degree of confidence.

Prediction-based methods can identify anomalies for each observation in a time series. Consequently, these methods can recognize all types of anomalies: point, collective and contextual anomalies.

4.4 Disadvantages of the methods

Like sliding window methods, the length of the history is important when determining anomaly. Thus, if the length of history m is chosen less than the value of the cycle length, the efficiency of the method will be unsatisfactory.

In order to improve the quality of the algorithm, it is recommended to use a history length greater than the cycle length. Nevertheless, if the value of m is slightly larger than the loop length, the complexity of the algorithm increases due to the increase in the dimension of the data.

In addition, due to the sparse nature of high-dimensional data, it must also be considered that specimens located closer in less-dimensional spaces will be located much further in a higher-dimensional space due to its sparseness.

4.5 Implementation

Depending on the prediction model used, implementations may differ. However, in general terms, prediction-based algorithms are in two parts, one of which analyzes and the other calculates the expected value. If the received new value differs greatly from the expected one, it is marked as anomalous in Algorithm 3.

5 Algorithm$_4$ – Hidden Markov Model Algorithms

A group of algorithms based on hidden Markov models are used to model sequences and then detect anomalies in similar sequences.

5.1 Application area

Hidden Markov Model (HMM) is a statistical model that simulates the operation of a process similar to a Markov process with unknown parameters. The task of such a model is to find unknown parameters based on the observed instances and the fixed behavior of the model. The HMM is also widely used to detect anomalies in time series.

This group of methods is based on the assumption that the observed time series is an indirect observation underlying the target time series. It is assumed that the process that creates the hidden time series is Markov, but the observed process that creates the original time series may not be Markov. By analogy, a ”normal” time series can be modeled using the HMM, but an abnormal time series cannot be.

Algorithm 3: A prediction-based algorithm for identifying anomalies in the data

Input:
Dataset – Testing dataset ValueCount – Number of values to analyze before prediction
BorderDiff – Maximum diff to mark value as anomaly

Output:
Anomalies – Anomaly list value from testing dataset

```
1  begin
2      int ValuesStored = 0;
3      List<data> Values;;
4      List<data> Anomalies;;
5      data LastPredicted = Null;
6
       // Step 1:  Iterate over values in dataset
7      foreach Value in Dataset do
           // Step 2:  Iterate over dictionary to get maximum distance
8          Values.add(Dataset);
9          ValuesStored = ValueStore + 1;
10
11         if ValuesStored == ValueCount then
               // Step 3:  Get predicted value and clear Values list for future
12             LastPredicted = PredictNextValue(Values);
13             ValuesStored = 0;
14             Values.Clear();
15             continues;
16         end
17
18         if LastPredicted != Null then
               // Step 4:  Compare predicted and next value
19             Anomalies.append(GetValueIfAnomaly(LastPredicted, Value, BorderDiff);
20         end
21     end
22
23     return Anomalies;
24 end
```

5.2 Method description

The classical HMM-based anomaly detection method is formally described as follows: The time series used for training, $O_{train} = O_1, ..., O_n$ are considered as a sequence of indirect observations of the HMM model. To determine the parameters of the HMM with maximizing the probability P (O_{train} — λ), a method called the Baum-Welch overestimation procedure is used. At the testing stage, taking into account the unknown time series $O_{test} = O_1, ..., O_n$, for which the probability P(O_{test} — λ) is calculated using the trained model. Of all analyzed time series, the anomalous are the series with the minimum value P (O_{test} — λ).

5.3 Method advantages

The advantage of methods based on hidden Markov models is the possibility of introducing domain-specific knowledge about the problem under consideration into the model.

In addition, there is no need for these traits to be statistically independent from each other, as would be the case with the generative model. Finally, instead of simple transition probabilities, arbitrary functions can be used for pairs of adjacent hidden states, which together makes it possible to optimize the algorithms for searching for anomalies for the conditions in which the analysis is performed.

5.4 Disadvantages of the methods

The main disadvantage of methods based on hidden Markov models is the limited scope of their application. In the absence of the main Markov process, these methods may not miss even obvious anomalies.

HMM-based methods create a Markov model for the time series, which ultimately allows an assessment of the anomalous behavior of each instance in the time series. Nevertheless, if the assumption entered in the method is violated, none of the types of anomalies will be detected by this method.

5.5 Implementation

Method implementations can differ to some extent in the set of used parameters. When solving problems of identifying anomalous time series in a set of time series described by a distribution, standard parameters are used: the distribution of the initial state, the probability of a state transition, and others. Often, for optimization purposes, a segment HMM is used, in which the probability distribution over the duration of each latent state is used as an additional feature in Algorithm 4.

6 Functional comparison

Let us further compare the described algorithms in terms of their functionality. As opportunities (i.e. comparison criteria), let us single out the Top-5 most demanded for algorithms with the following purpose:

1. $Criterion_1$ – dependence of performance on previous experience, i.e. the need for preliminary "idle" work with the test dataset, even after training, in order to determine its statistical parameters;
2. $Criterion_2$ – adaptation to new data without additional adjustment, i.e. the ability to use algorithms without making changes with different data sets;
3. $Criterion_3$ – high requirements for the initial parameters of the algorithm, such as the purity of the data set (no noise) or the need for partial marking of the data sets used for training;
4. $Criterion_4$ – requirements for the statistical dependence of time series, i.e. taking into account the distribution of instances in the dataset when analyzing time series;
5. $Criterion_5$ – the ability to detect collective anomalies, i.e. not only single, but also independent sequential group anomalies.

The final criterion comparison with the point summation is presented in Table 1. To summarize, the following designations are used: "+" – 1 point, "-" – 0 points.

According to the criteria-based comparison of the functional capabilities of the algorithms (see Table 1), the following conclusions can be drawn. First, $Algorithm_3$ (5 points) has the highest compliance with

Algorithm 4: A prediction-based algorithm for identifying anomalies in the data

```
Input:
    Dataset – Testing data set with states
    InitialStateDistribution – Distribution of values in data set
    StateChangeProbability – Probability of changing process state
Output:
    Anomalies – Anomaly list value from testing data set
1  begin
2      List<data> Anomalies;;
3      data data AnomalyPath;
4
       // Step 1:  Iterate over states to set paths
5      foreach State in Dataset do
6          PathProbability = GetPathByStateChangeProbability(State, StateChangeProbability);
7
           // Step 2:  Check if path is possible in terms of distribution
8          if IsPathPossible(PathProbability) == True then
9              continue;
10         else
               // Step 3:  Add impossible by distribution path to anomalies
11             AnomalyPath = GetPath(PathProbability);
12             Anomalies.Add(AnomalyPath);
13         end
14     end
15
16     return Anomalies;
17 end
```

Algorithm name	$Creiterion_1$	$Creiterion_2$	$Creiterion_3$	$Creiterion_4$	$Creiterion_5$	**Points**
$Algorithm_1$	–	+	--	+	--	2
$Algorithm_2$	+	+	+	--	+	4
$Algorithm_3$	+	+	--	--	+	3
$Algorithm_4$	+	–	--	+	+	3
Total points	3	3	1	2	3	12

Table 1: Criteria comparison of algorithms capabilities

all the criteria. Secondly, in contrast to it, $Algorithm_1$ has the least compliance with all criteria (2 points). And, thirdly, the combination of all algorithms on average satisfies each of the criteria approximately equally (3 points, with the exception of 2 points for $Criterion_4$).

7 Efficiency comparison

Let us compare the described algorithms from the standpoint of their efficiency [13], by which we mean the classic set of the following three indicators (i.e. comparison criteria):

- Potency – the ability of the algorithm to obtain the correct result (analogue of F-measure);

- Operativeness – the speed of the algorithm when receiving the result;
- Resource Efficiency – the amount of resources (human, hardware and software, etc.) required for the algorithm to obtain a result.

The final criterion comparison is presented in Table 2. The cells of the table indicate the scores according to the following values of the indicators: 3 – high, 2 – medium, 1 – low. For simplicity, we will assume that the indicators are of equal importance in the integral efficiency and their scores can be summed up.

Algorithm name	Potency	Operativeness	Resource Efficiency	Points
$Algorithm_1$	2	2	3	7
$Algorithm_2$	2	1	1	4
$Algorithm_3$	1	3	2	6
$Algorithm_4$	3	3	1	7
Total points	8	9	7	24

Table 2: Criteria comparison of algorithms efficiency

Based on the comparison of the performance indicators of the algorithms (see Table 2), the following conclusions can be drawn. First, $Algorithm_1$ and $Algorithm_4$ have the highest efficiency (7 points each). Secondly, in contrast to it, $Algorithm_2$ (4 points) has the least efficiency. And, thirdly, the combination of all algorithms will give the following total scores for each of the performance indicators: Potency – 8. Operativeness – 9 and Resource Efficiency – 7. Thus, the indicators for the complex algorithm have similar values. This conclusion is especially interesting, since separately the algorithms had a large difference in the values of indicators – except for $Algorithm_1$, in all the others there was an indicator with both the lowest value (i.e. 1) and the highest (i.e. 3).

8 Conclusion

In the course of the research, the analysis of the effectiveness of the most common methods of searching for anomalies in data presented in the form of time series was carried out. Based on the results of the work carried out, we can talk about the high efficiency of both the sliding window algorithm and the algorithms based on hidden Markov models.

The scientific novelty of the research carried out lies in the aggregation of knowledge and comparison of algorithms within a wide range of problems of identifying data anomalies in real time. The results obtained in this way can be applied both in applied (for example in the field of information security [14, 15, 16, 17]) and scientific directions.

Optimizing existing systems used to detect anomalies can greatly improve their performance [18]. On the other hand, on the basis of the analysis performed, the methods themselves can be optimized, as well as combined methods can be created that make it possible to level the existing costs according to the principle of complementarity.

Further research in this direction can be aimed at conducting a similar analysis for algorithms for detecting data anomalies in fixed unlabeled datasets. An alternative direction for further research may be the search for ways to integrate methods for detecting anomalies in real time into the machine learning process [19] at the stage of additional training of the model, which can potentially increase the level of its protection against known attacks [20].

Acknowledgments

The reported study was partially funded by the budget project 0073-2019-0002.

References

[1] J. Ding, X. Li, and V. Gudivada. Augmentation and evaluation of training data for deep learning. In *Proc. of the 2017 IEEE International Conference on Big Data (BigData'17), Boston, MA, USA*, pages 2603–2611. IEEE, December 2017.

[2] I. Kotenko, A. Krasov, I. Ushakov, and K. Izrailov. An approach for stego-insider detection based on a hybrid nosql database. *Journal of Sensor and Actuator Networks*, 10(2), 2021.

[3] P. Zhou and S. Abbaszadeh. Towards real-time machine learning for anomaly detection. In *Proc. ot eh 2020 IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC'20), Boston, MA, USA*, pages 1–3. IEEE, October-November 2020.

[4] D. Wu, Z. Sun, Y. Zhu, L. Tian, H. Zhu, P. Xiong, Z. Cao, M. Wang, Y. Zheng, C. Xiong, H. Jiang, K. Tsoi, X. Niu, W. Mao, C. Feng, X. Zha, G. Deng, and W. Luk. Custom machine learning architectures: towards realtime anomaly detection for flight testing. In *Proc. of the 2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW'18), Vancouver, BC, Canada*, pages 1323–1330. IEEE, May 2018.

[5] Z. Hasani. Robust anomaly detection algorithms for real-time big data: Comparison of algorithms. In *Proc. of the 6th Mediterranean Conference on Embedded Computing (MECO'17), Bar, Montenegro*, pages 1–6. IEEE, June 2017.

[6] A. Huck, M. Guillaume, G. Oller, and M. Grizonnet. Comparisonof local anomaly detection algorithms based on statistical hypothesis tests. In *Proc. of the 3rd Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS'11), Lisbon, Portugal*, pages 1–4. IEEE, June 2011.

[7] A. Kunjumon, A. Madhu, and J. Kizhakkethottam. Comparison of anomaly detection techniques in networks. In *Proc. of the 2015 International Conference on Soft-Computing and Networks Security (ICSNS'15), Coimbatore, India*, pages 1–3. IEEE, February 2015.

[8] G. Baldi, Y. Diaz, T. Dimitrakos, F. Martinelli, C. Michailidou, P. Mori, O. Osliak, and A. Saracino. Session-dependent usage control for big data. *Journal of Internet Services and Information Security (JISIS)*, 10(3):76–92, August 2020.

[9] M. Khan, K. Ferens, and W. Kinsner. A cognitive multifractal approach to characterize complexity of non-stationary and malicious dns data traffic using adaptive sliding window. In *Proc. of the 14th IEEE International Conference on Cognitive Informatics Cognitive Computing (ICCI*CC'15), Beijing, China*, pages 76–83. IEEE, July 2015.

[10] B. Tu, X. Yang, X. Ou, G. Zhang, J. Li, and A. Plaza. Ensemble entropy metric for hyperspectral anomaly detection. *IEEE Transactions on Geoscience and Remote Sensing*, pages 1–17, 2021.

[11] C. Wang, Y. Yao, and H. Yao. Video anomaly detection method based on future frame prediction and attention mechanism. In *Proc. of the IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC'21), NV, USA*, pages 0405–0407. IEEE, January 2021.

[12] X. Li, T. Zhang, K. Li, and Y. Liu. Spacecraft telemetry data anomaly detection based on multi-objective optimization interval prediction. In *Proc. of the 2019 Prognostics and System Health Management Conference (PHM-Qingdao'19), Qingdao, China*, pages 1–8. IEEE, October 2019.

[13] I. Kholod, A. Shorov, and S. Gorlatch. Efficient distribution and processing of data for parallelizing data mining in mobile clouds. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, 11(1):2–17, March 2020.

[14] M. Buinevich, K. Izrailov, and G. Ganov. Intellectual method of programs interactions visualization for information security audit of the operating system. In *Proc. of the 12th Majorov International Conference on Software Engineering and Computer Systems (MICSECS'20), Saint Petersburg, Russia*, volume 1, pages 1–12. CEUR Workshop Proceedings, December 2020.

[15] M. Buinevich, K. Izrailov, I. Kotenko, and P. Kurta. Method and algorithms of visual audit of program interaction. *Journal of Internet Services and Information Security (JISIS)*, 11(1):16–43, February 2021.

[16] A. Fedorchenko, I. Kotenko, and A. Chechulin. Integrated repository of security information for network security evaluation. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, 6(2):41–57, June 2015.

[17] F. Valenza and M. Cheminod. An optimized firewall anomaly resolution. *Journal of Internet Services and Information Security (JISIS)*, 10(1):22–37, February 2020.

[18] F. Rubin, S. Paulo, W. Santos, R. Oliveira, F. Rossi, and T. Ferreto. Evaluating energy and thermal efficiency of anomaly detection algorithms in edge devices. In *Proc. of the 2020 International Conference on Information Networking (ICOIN'20), Barcelona, Spain*, pages 208–213. IEEE, January 2020.

[19] C. Johnson, B. Khadka, R. B. Basnet, and T. Doleck. Towards detecting and classifying malicious urls using deep learning. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, 11(4):31–48, December 2020.

[20] V. Fedorov, E. Balenko, N. Gololobov, and K. Izrailov. Testing platform invoke as a tool for shellcode injection in windows applications. In *Proc. of the 2021 International Conference on Automatics and Energy (ICAE'21), Vladivostok, Russia*, volume 2096 of *Journal of Physics: Conference Series*, page 012048. IOP Publishing, October 2021.

Author Biography

Nikita Gololobov graduated from Bonch-Bruevich St. Petersburg State University of Telecommunications, Infocommunication networks and systems Department in 2020. Currently, he is studying for a master's degree in St. Petersburg State Polytechnic University. His scientific interests include information security and methods of vulnearability searching, malware analysis, UEFI, data science and machine learning algorithms.

Konstantin Izrailov graduated from St. Petersburg State Polytechnic University, Physical and Mechanical Department in 1996. He received PhD degree from the Bonch-Bruevich St. Petersburg State University of Telecommunications in 2017. Currently, he is an associate professor at the St. Petersburg State University of Telecommunications and Senior Researcher of the St. Petersburg Federal Research Center of the Russian Academy of Sciences. He has about 70 published articles and 5 patents. He is also the Deputy Editor-in-Chief of the university journal. His scientific interests include information security, search of vulnerabilities in machine code, reverse engineering, telecommunication devices, machine learning, intelligent algorithms, Internet of things, UEFIBIOS, Smart City.

Igor Kotenko graduated with honors from St. Petersburg Academy of Space Engineering and St. Petersburg Signal Academy. He obtained the Ph.D. degree in 1990 and the National degree of Doctor of Engineering Science in 1999. He is Professor of computer science and Head of the Laboratory of Computer Security Problems of the St. Petersburg Federal Research Center of the Russian Academy of Sciences. He is the author of more than 500 refereed publications, including 14 textbooks and monographs. Igor Kotenko has a high experience in the research on computer network security and participated in several projects on developing new security technologies. For example, he was a project leader in the research projects from the US Air Force research department, via its EOARD (European Office of Aerospace Research and Development) branch, EU FP7 and FP6 Projects, HP, Intel, F-Secure, etc. The research results of Igor Kotenko were tested and implemented in more than 50 Russian research and development projects.