DOI: 10.15514/ISPRAS-2024-36(4)-8

Intelligent Algorithms for Detecting Attacks in the Web Environment

[1] M.A. Lapina, ORCID: 0000-0001-8117-9142 <mlapina@ncfu.ru>
[1] V.V. Movzalevskaya, ORCID: 0009-0007-7540-3110 <vitaliya1306@gmail.com>
[1] M.E. Tokmakova, ORCID: 0009-0000-2608-7712 <marinatokmakova175@mail.ru>
[1] M.G. Babenko, ORCID: 0000-0001-7066-0061 <mgbabenko@ncfu.ru>
[2] V.P. Kochin, ORCID: 0000-0001-7782-9536 <kochyn@bsu.by>

[1] North Caucasus Federal University,
1, Pushkina st., Stavropol, 355017, Russia.
[2] Belarusian State University,
4, Nezavisimosti ave., Minsk, 220030, Belarus.

Abstract. The article is devoted to the analysis of the use of machine learning algorithms to detect attacks using a custom web environment or the functionality of user applications. Learning with a teacher and clustering algorithms are considered. The dataset uses a sample of online shopping transactions collected by an e-commerce retailer. The dataset contains 39,221 transactions. To detect attacks in the web environment, the most optimal implementations of machine learning algorithms were selected after their review and comparative analysis. The most effective algorithm for detecting fraudulent transactions has been determined. We use the accuracy and running time of the algorithm as criteria. The accuracy of detecting fraudulent transactions for Random Forest, GB (Scikit-learn), GB (CatBoost) algorithms is 100%, and the KD-trees algorithm is 99,9%. The gradient boosting algorithm in the CatBoos implementation is 4,2 times faster than Random Forest, 2,4 times faster than GB Scikit-learn, 1,2 times faster than GB without using the cat_features parameter, 41,9 times faster than k-dimensional trees, 66,8 times faster than DBSCAN. The data obtained for each method is presented in the form of tables. Within the framework of this work, the parameters for evaluating the effectiveness of the algorithms under study are learning time indicators, as well as characteristics from the Confusion matrix and Classification Report for classification algorithms, and fowlkes_mallows_score, rand_score, adjusted_rand_score, Homogeneity, Completeness, V-measure for clustering algorithms.

Keywords: machine learning, web environment, consumer websites, cybersecurity, classification algorithms with a teacher, clustering.

For citation: Lapina M.A., Movzalevskaya V.V., Tokmakova M.E., Babenko M.G., Kochin V.P. Intelligent algorithms for detecting attacks in the web environment. Trudy ISP RAN/Proc. ISP RAS, vol. 36, issue 4, 2024. pp. 99-116. DOI: 10.15514/ISPRAS–2024–36(4)–8.

Интеллектуальные алгоритмы обнаружения атак в веб-среде

[1] М.А. Лапина, ORCID: 0000-0001-8117-9142 <mlapina@ncfu.ru>
[1] В.В. Мовзалевская, ORCID: 0009-0007-7540-3110 <vitaliya1306@gmail.com>
[1] М.Е. Токмакова, ORCID: 0009-0000-2608-7712 <marinatokmakova175@mail.ru>
[1] М.Г. Бабенко, ORCID: 0000-0001-7066-0061 <mgbabenko@ncfu.ru>
[2] В.П. Кочин, ORCID: 0000-0001-7782-9536 <kochyn@bsu.by>

[1] Северо-Кавказский федеральный университет,
355017, Россия, г. Ставрополь, ул. Пушкина, д. 1.
[2] Белорусский государственный университет,
220030, Белоруссия, г. Минск, пр-к. Независимости, д. 4

Аннотация. Статья посвящена анализу использования алгоритмов машинного обучения для обнаружения атак с использованием пользовательской веб-среды или функциональности пользовательских приложений. Рассматриваются алгоритмы обучения с преподавателем и кластеризации. В наборе данных используется выборка транзакций онлайн-покупок, собранная розничным продавцом электронной коммерции. Набор данных содержит 39 221 транзакцию. Для обнаружения атак в веб-среде были выбраны наиболее оптимальные реализации алгоритмов машинного обучения после их обзора и сравнительного анализа. Был определен и реализован наиболее эффективный по времени и качеству алгоритм для рассматриваемой выборки данных. Данные, полученные по каждому методу, представлены в виде таблиц. В рамках данной работы параметрами для оценки эффективности исследуемых алгоритмов являются показатели времени обучения, а также характеристики из матрицы путаницы и отчета о классификации для алгоритмов классификации, а также fowlkes_mallows_score, rand_score, adjusted_rand_score, Однородность, полнота, V-мера для алгоритмов кластеризации.

Ключевые слова: машинное обучение, веб-среда, потребительские веб-сайты, кибербезопасность, алгоритмы классификации с преподавателем, кластеризация.

Для цитирования: Лапина М.А., Мовзалевская В.В., Токмакова М.Е., Бабенко М.Г., Кочин В.П. Интеллектуальные алгоритмы обнаружения атак в веб-среде. Труды ИСП РАН том 36, вып. 4, 2024 г., стр. 99–116 (на английском языке). DOI: 10.15514/ISPRAS–2024–36(4)–8.

1. Introduction

The urgent need to ensure information security on the Internet is easily explained by several factors, including the massive unification of heterogeneous and distributed systems, the presence of large amounts of confidential information in end systems maintained by corporations and government agencies, the easy distribution of automated malicious software by attackers, and the ease with which computer crimes can be committed anonymously.

With the development of web technologies and a significant increase in the volume of information, this problem is only getting worse. Attackers often exploit vulnerabilities in the system or web applications to upload a malicious file or malicious code to a web server. This is often used as a so-called backdoor for working with and managing a web server, because such a file can provide attackers with remote access to the server management interface, including executing commands, manipulating files and connecting to a database. Therefore, an accurate determination of whether files stored on a web server are malicious is of great importance for the security of the web server.

With the constant operation of web browsers, the security and privacy of users may be at risk, because browser vulnerabilities can lead to unprotected use [1]. Important user data, such as login, can be collected and used for profiling, which raises serious privacy concerns.

In this regard, the development of new, more advanced access control mechanisms and the optimization of existing ones is very relevant [1].

Machine learning algorithms allow you to detect and prevent attacks, which significantly increases the effectiveness of site protection [2].

At its core, machine learning can be represented as the process of inferring algorithms for predicting unknown data using previously collected information.

As part of the study of the effectiveness of several of the most widely used machine learning models, several algorithms have been implemented and analyzed, classification and clustering methods have been considered. In this regard, it is necessary to define a training sample. The article uses a sample from the work [2]. The paper does not provide a complete description of all the algorithms, as such information is too extensive. A detailed description of each method can be found in the book [3].

The article presents the following sections: section 1 – introduction; section 2 – relevance; section 3 – theoretical description of models; section 4 – modeling; section 5 – analysis of the results; section 6 – conclusions.

2. Relevance

The security issues of the web environment and user applications are very relevant, since web applications are accessible over the network and very often contain vulnerabilities, which is why they regularly become targets of cyber-attacks [4]. In today's digital environment, the protection of web space plays an important role in ensuring the integrity of user data and confidentiality, as well as in the safe and continuous use of web applications. Unfortunately, the rapid spread of digital technologies is always accompanied by the same rapid spread of threats and attacks in the web environment, which constantly requires the creation of new security methods, as well as the improvement of existing ones [5].

To date, there are no universal means of protection, much less those that could identify new types of threats in the web environment. Machine learning algorithms can be used to prevent attacks. They allow you to automate the process of detecting and preventing attacks.

The Bayesian networks. A Bayesian network is a graphical model that encodes probabilistic relationships between variables of interest [6, 7]. These networks are a combination of two different mathematical fields: graph theory and probability theory [8].

The Bayesian network is used to study cause-and-effect relationships and, therefore, to gain an understanding of the problem area and predict the consequences of intervention. Because the model has both causal and probabilistic semantics, it is ideally suited to represent a combination of prior knowledge and data.

Artificial neural network. The neural network includes many neurons used for processing and analyzing information, its structure is like the nervous system of a living organism [9, 10]. Neural networks consist of basic blocks like neurons. These blocks interact with each other based on connections, the strength of which can be changed because of the learning process or modification of the algorithm [11].

The use of neural networks demonstrates some of the best results in classification problems with limited input parameters, in such tasks, this method is much more effective than other machine learning methods [12].

The support vector machine. This method refers to learning algorithms with a teacher and is often used in solving problems related to detecting attacks in a web environment.

The method of reference vectors is based on linear classification [13-15]. It is one of the classic machine learning methods that can help solve big data classification problems. This method is especially effective in multi-domain applications in a big data environment [16]. However, the support vector machine is mathematically complex and computationally expensive.

Common attacks in the web environment. In this part of the study, the five most common attacks committed in 2020-2021 are presented and described [4]. The most common attacks include the following: attacks on clients (98%), data leakage (91%) and unauthorized access to the application (84%).

If we talk about the distribution of attacks by industry, the most popular for 2022 are: the public sector of the economy (30%), financial technology (23%), education (16%) [5] (Fig. 1).

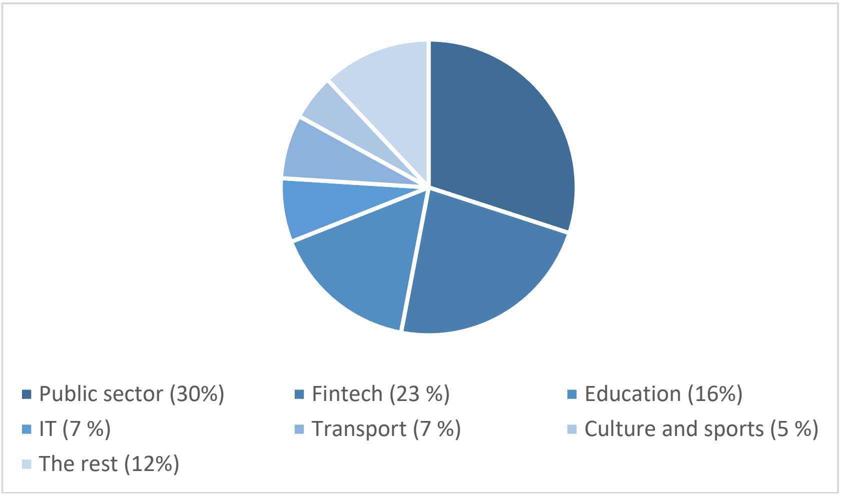

Fig. 1. Distribution of web attacks by industry.

Attacks on customers. These attacks are the most popular in the statistics given, they exploit weaknesses in applications running on a computer controlled directly by a user, often referred to as a client [17]. In 84% of the studied applications, threats of unauthorized access to the personal accounts of users, including administrators, were identified. In 72% of web applications, an attacker can gain access to functionality or content that should not be available to him, for example, to view the personal accounts of other users.

Attackers can harm users or compromise them using information and data extracted in various ways, for example, from cookies.

The most deplorable consequences of the actions of intruders include the disclosure of important confidential information, obtaining unauthorized access to application codes and local network resources, which leads to the spread of malicious actions on the infrastructure.

Data leak. Data leaks are the second most pressing security threat in web application research. The results of the security analysis showed that more than three quarters of web applications were exposed to the disclosure of user IDs. Personal data was disclosed in 60% of applications, and user credentials in 47%, which is 13 and 16 percent more, respectively, than in 2019. Personal and credentials are desirable targets for intruders, which is confirmed by the data of the final analysis of current cyber threats in 2021 [18].

A data leak is the intentional or unintentional disclosure of confidential information to unauthorized persons. Data leakage poses a serious threat to organizations, including significant reputational damage and financial losses.

As the volume of data grows exponentially and data leaks occur more frequently than ever before, data loss detection and prevention has become one of the most pressing security concerns for enterprises. Despite a lot of research on protecting confidential information from leakage, this remains an urgent research problem.

Unauthorized access to the application. Unauthorized access to the application (UAA) was detected in 84% of web applications. This is an attack in which an attacker gains access to an application without the permission or consent of the owner. This can happen in a variety of ways, including password hacking, exploiting security vulnerabilities, and authentication.

Unauthorized access to the application may result in viewing, changing, or deleting confidential information, disrupting the operation of the application, and gaining full control over the system.

To prevent UAA, it is necessary to take measures to protect applications and systems, such as regular software updates, the use of complex passwords and two-factor authentication, as well as monitoring user activity and intrusion detection.

Denial of service. Distributed Denial of Service (DDoS) attacks consist of streams of packets from various sources. These streams consume some key resource, making it inaccessible to legitimate users.

The interaction of distributed machines generating attack streams makes tracking and mitigation a very difficult task. Some protection mechanisms focus on detecting an attack near a computer, characterizing it, and filtering attack packets. Although the detection accuracy of these mechanisms is high, traffic is usually so aggregated that it is difficult to distinguish legitimate packets from attack packets. More importantly, the volume of the attack may be more than the system can withstand.

DDoS attacks can be used to conceal other network attacks. When a website is under attack, an internal and external group of information security specialists usually focuses on closing the ports of these sites, clearing traffic, and resuming its operation.

Implementation of Operating system commands. OS command injection is a vulnerability in websites that allows an attacker to inject malicious code or commands into the system through the user interface or using scripts that are processed by the operating system. This entails obtaining confidential data by an attacker, as well as spreading attacks to other systems.

3. Theoretical description of the models

This section provides the mathematical construction of algorithms, description, and analysis of their implementations, as well as justification for choosing the most optimal one for the data sample used.

3.1 Random Forest

Random forests are formed as simple ensembles of several decision trees, usually containing tens to thousands of such trees.

After training each individual decision tree, general predictions of random forests are made by choosing the statistical mode of forecasts of individual trees for classification trees (that is, each tree "votes") and the average statistical value of forecasts of individual trees for regression trees. However, the increased complexity of random forests makes it difficult to analyze predictions compared to single decision trees. When building a random forest tree, the following happens:

- A subset of training objects is randomly selected from the entire dataset. This subset may contain duplicate objects.
- A subset of features is randomly selected (usually the square root of the total number of features). This reduces the correlation between the trees in the ensemble and improves their diversity.
- A decision tree is built based on the selected subset of data and features. When building a tree, it is necessary to use an information criterion (for example, the entropy criterion), which allows you to choose the best feature for dividing data at each available level of the tree.
- Repeat steps 1-3 for each tree in the ensemble.

In general, the algorithm for constructing a random forest consisting of N trees can be represented as follows:

For each n = 1, ..., N do

- Generate a selection of X_n bootstraps.
- Build a decision tree b_n from a sample of X_n.

To solve the classification problem, a majority vote is used, and for the regression problem, an average one is used. The final classifier looks like this:

$$a(x) = \frac{1}{N}\sum_{i=1}^{N} \quad b_i(x), \#(1)$$

in classification tasks, it is recommended to take:

$$m = \sqrt{n}, \#(2)$$

and in regression tasks:

$$m = \frac{n}{3}, \#(3)$$

where n is the number of features.

3.2 Gradient boosting

Gradient boosting (GB) is an optimization algorithm used to minimize errors in a machine learning model.

This method is based on a vector called a gradient, which represents the largest increase in the function in its direction, its coordinates are partial derivatives of the function [19]. In other words, if the function f(x, y, z) is given, then the gradient is calculated using the formula:

$$gradf = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right). \#(4)$$

The idea of the gradient approach is to iteratively adjust the model parameters in the direction of the negative gradient of the loss function (which represents the error) to reduce the error and find the optimal parameters that give the best prediction results. Therefore, this method can be used to obtain the smallest error value or to find weights when training neural networks. Weights are values that indicate important information when training neural networks using a «teacher» [20].

Because at each iteration of the algorithm, the model parameters are updated in the direction opposite to the gradient of the loss function. The size of the step that the algorithm takes in this direction is determined by the learning rate. The key parameter is the optimal learning rate of the model, this is since too large a step can lead to skipping the minimum, and too small can affect the optimization process, slowing it down.

The formula for updating the parameter θ at each iteration is as follows:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta), \#(5)$$

where η is the learning rate, $\nabla_\theta J(\theta)$ is the gradient of the loss function J.

There are three main implementations of GB: LightGBM, XGBoost and CatBoost, below are the main characteristics of these implementations (Table 1).

Two implementations were chosen for the study: CatBoost and Scikit-learn. The implementation of CatBoost, because it provides high performance and prevents overfitting, it is very simple, but at the same time it is not inferior in efficiency to LightGBM and XGBoost (Table 1), also this implementation is stated as the fastest and optimized [21], therefore, the study compares it with Scikit-learn, the most frequently implemented.

3.3 k-dimensional trees

A k-dimensional tree (KD-tree) is a binary tree that stores data in a format optimized for multidimensional spatial analysis. The formation of a KD-tree can be considered as a preliminary stage of classification algorithms using the k-nearest neighbor (k-NN) method, but this technique can also be considered an independent clustering algorithm. The algorithm creates a tree structure, and clusters are stored in leaves.

A typical algorithm for the formation of KD-trees is given below. For each node that differs from the sheet, the following actions are performed:

- Choosing the dimension for separation.
- Select the separation point.
- Division of the subspace according to the selected dimension and the separation point.
- Termination of subspace separation when the current subspace contains fewer elements than a certain number of sample elements for a separate subspace.

Table 1. Results for different diode insertion strategies.

Parameter	LightGBM	XGBoost	CatBoost	Scikit-learn
The main purpose	Optimized for working with categorical data	Focused on efficiency and productivity	It is focused on speed when working with large amounts of data	A simple and effective way to implement (classical implementation)
Categorical data	Built-in processing without pre-coding	Pre-coding is required	It has optimizations for categorical features	Pre-coding is required
Learning rate	High, with GPU support	High, with GPU support	Very high	High
Preventing over-training	Uses a variety of regularization strategies	Supports L1 and L2 regularization	Uses mechanisms such as EFB	Uses different methods to prevent overfitting
Big Data	Optimized to work efficiently with large datasets	It may be ineffective on very large datasets	Optimized to work with large amounts of data with low memory requirements	It can handle large amounts of data, but there are limitations related to the amount of memory
Programming languages	Support for major languages, including Python, R, Java	Extensive language support, including Python, R, Java, Scala	Supports Python, R, Java and other languages	Supports Python
The complexity of the models	Generates more complex models with retraining control	Allows you to adjust the complexity of the model through hyperparameters	Builds lightweight models with a histogram approach	Uses regularization to reduce complexity
Interpretability	Provides good interpretability	Provides an average level of interpretability	Interpretability can be difficult due to optimizations	Provides good interpretability

The result of this procedure is a binary search tree in feature subspaces, while combining all subspaces of leaf nodes forms a complete feature space. When creating KD-tree models to search for nearest neighbors, a binary tree with a split space must be saved as an addition to the training data points. Moreover, additional data on which sample items belong to specific leaf nodes should also be stored in the model. Thus, even more space is required to store such a model, which makes it less economical (in terms of memory resource consumption) than the original k-NN models.

In an inhomogeneous KD-tree:

$$H_i(t) = (x_1, x_2, \dots, x_{i-1}, t, x_{i+1}, \dots, x_k), \#(6)$$

for $1 \leq i \leq k$, parallel to the axis *(k-1)* of the dimensional hyperplane at point t. For the root, you need to divide the points through the hyperplane $H_1(t)$ into two, if possible, equally large sets of points and write t to the root, to the left of this, all points with $x_1<t$ are saved, on the right are those with $x_1>t$.

For the left subtree, you need to divide the points again into a new «split plane» $H_2(t)$, and t is stored in the inner node. To the left of this, all points with $x_2<t$ are saved. This continues recursively over all spaces. Then everything starts again from the first space until each point can be clearly identified through the hyperplane.

KD-trees are usually not suitable for high-dimensional data. For feature spaces with high dimensionality, the efficiency of KD-trees is comparable to the efficiency of linear search by simple iteration. Nevertheless, KD-trees are very convenient for quickly searching for nearest neighbors with an average time complexity of *O(log n)*.

3.4 DBSCAN

DBSCAN (DensityBased Spatial Clustering of Applications with Noise) is a density–based spatial clustering algorithm for applications with noise. It increases clusters according to density-based connectivity analysis. Density-based clustering algorithms are used to detect clusters in datasets of arbitrary shape and large size. These algorithms are usually grouped as dense regions of points in the data space, separated by low-density regions.

The key idea of DBSCAN is that for each cluster object of a neighborhood of a given radius ε, there must be at least a minimum number of Pts_{min} objects, which means that the power of the neighborhood must exceed a certain threshold. The neighborhood of an arbitrary point p is defined by:

$$N_{\varepsilon} = q \in D / dist(p, q) < \varepsilon, \#(7)$$

where *D* is the database of objects. If the neighborhood of a point *P* contains at least the minimum number of points, then this point is called the main point. The main point is defined as:

$$N_{\varepsilon}(P) > Pts_{min}, \#(8)$$

where ε and Pts_{min} are user-defined parameters, which mean the radius of the neighborhood and the minimum number of points in the neighborhood of the base point, respectively.

4. Modeling

This section describes the implementation, training, and testing of the machine learning algorithms described above.

The dataset uses a sample of online shopping transactions collected by an e-commerce retailer. The dataset contains 39,221 transactions, each of which contains 5 properties or characteristics that can be used to describe the transaction: accountAgeDays (age of the account from which the payment was made), numItems (number of purchased items), localTime (local time of purchase), PaymentMethod (payment method), paymentMethodAgeDays (number of days since the addition of this payment method), as well as a binary label that determines whether the transaction is fraudulent: the label «1» indicates a fraudulent transaction, The label «0» is assigned to the legal transaction (label column, Table 2). The dataset is taken from [2]. The task of the trained model is to accurately determine by the properties of the transaction whether it is fraudulent or not.

Below are 3 random rows from the selected dataset (Table 2).

Table 2. Model training time.

line number	accountAgeDays	numItems	localTime	paymentMethod	paymentMethodAgeDays	lable
37813	64	1	4,748314	creditcard	0,0	0
15585	2000	1	4,524580	creditcard	0,0	0
24030	653	1	4,748314	creditcard	0,0	0

The values in the columns accountAgeDays, numItems, localTime, payment Method Age Days are integers (accountAgeDays, numItems) and non-integers (localTime, paymentMethodAgeDays). The PaymentMethod column contains the payment method as a string, which can take the following values: «creditcard», «paypal», «storecredit».

In machine learning, there are quite a few different metrics that allow you to determine the accuracy and efficiency of a trained model. Within the framework of this study, the parameters for evaluating the effectiveness of the algorithms under study are learning time indicators (Section 5), as well as characteristics from the Confusion matrix and Classification Report for classification algorithms.

Support is the number of actual occurrences of the class in the dataset.

4.1 Random Forest

First, you need to import the Random Forest Classifier and create a model. The main input parameters of the classifier are the number of trees in the forest (n_estimators), the maximum depth (max_depth) and the number of functions that should be considered when searching for the best separation (max_features). By default, these characteristics are 100, None, and sqrt (n_features), respectively, where n_features is the number of functions in the data.

The results of the model trained using the random forest algorithm are presented in Table 8 (for the methods of random forest, GB (Scikit-learn), GB (CatBoost), a general table is given, since their results turned out to be identical).

The algorithm identified all classes without errors, that is, it identified 190 fraudulent transactions and 12753 legal ones. There are no objects identified as falsely negative or falsely positive. The precision, recall and F1-Score values for each class are close to their best value.

An estimate of the operating time of the model obtained during experiments on the same data is presented in Table 7 (Section 5). The average operating time of the model is 0,08 sec.

4.2 Gradient boosting

One of the implementations of the GB algorithm, which is considered in the study, is the implementation of Scikit-learn. One of the undoubted advantages of the chosen implementation is ease of use and accessibility [22].

First, you need to import the GradientboostingClassifier and create a model. The main input parameters of the classifier are the number of boosting stages to perform (n_estimators), maximum depth (max_depth). GB is usually resistant to overfitting, therefore, with large values of the n_estimators parameter, the algorithm shows better results. By default, these characteristics are 100 and 3, respectively.

The results of the model trained using the GB algorithm in the implementation of Scikit-learn are presented in Table 2 (for the methods of random forest, GB (Scikit-learn), GB (CatBoost), a general table is given, since their results turned out to be identical).

The algorithm identified all classes without errors, that is, it identified 190 fraudulent transactions and 12753 legal ones. There are no objects identified as falsely negative or falsely positive. The precision, recall and F1-Score values for each class are close to their best value.

An estimate of the operating time of the model obtained during experiments on the same data is presented in Table 7 (Section 5). The average operating time of the model is 0.046629 sec.

CatBoost is another implementation of the GB algorithm. The main parameters used by the algorithm in model training and prediction are the index of the first used tree (ntree_start), the index of the first unused tree (ntree_end) and the list of categorical features (cat_features).

The results of the model trained using the GB algorithm for the implementation of CatBoost are presented in Table 8 (for the methods of random forest, GB (Scikit-learn), GB (CatBoost), a general table is given, since their results turned out to be identical).

The algorithm identified all classes without errors, that is, it identified 190 fraudulent transactions and 12753 legal ones. There are no objects identified as falsely negative or falsely positive. The precision, recall and F1-Score values for each class are close to their best value.

An estimate of the operating time of the model obtained during experiments on the same data is presented in Table 7 (Section 5). The average operating time of the model is 0,024148 sec.

By default, categorical features are not considered in the process of training a model using CatBoost, however, one of the key features of this implementation is precisely the ability to train a model without first preparing categorical data.

Below is the importance of each feature in the process of training the model without preliminary data preparation (Table 3) and with it, i.e. using the cat_features parameter (Table 4).

Table 3. The importance of features in training a model without the cat_features parameter.

line number	Feature Id	Importances
0	accountAgeDays	78,826096
1	localTime	6,487206
2	numItems	4,092288
3	paymentMethodAgeDays	3,228341
4	paymentMethoz_storecredit	3,123910
5	paymentMethod_creditcard	2,912722
6	paymentMethod_paypai	1,329436

Table 4. The importance of features in training a model with the cat_features parameter.

line number	Feature Id	Importances
0	accountAgeDays	84,247069
1	paymentMethodAgeDays	8,237712
2	localTime	3,882424
3	numItems	3,632795
4	paymentMethod	0,000000

According to the data from the tables above, it can be seen that the importance of features in the learning process for models with the cat_features parameter and without this parameter is quite different.

The results of the model trained using the GB algorithm when implementing CatBoost using the cat_features parameter is presented in Table 8 (for the methods of random forest, GB (Scikit-learn), GB (CatBoost), a general table is given, since their results turned out to be identical).

The algorithm identified all classes without errors, that is, it identified 190 fraudulent transactions and 12753 legal ones. There are no objects identified as falsely negative or falsely positive. The precision, recall and F1-Score values for each class are close to their best value.

The accuracy of the algorithm with the cat_features parameter has not changed compared to the algorithm without this parameter and remains the same high.

An estimate of the operating time of the model obtained during experiments on the same data is presented in Table 7 (Section 5). the operating time of the model has improved slightly compared to the model with preprocessing of categorical features, the only exception is the «best operating time», which has deteriorated by several thousandths of a second.

The results of the model trained using the GB algorithm are presented in Table 5.

Table 5. Classification report for random forest, GB (Scikit-learn), GB (CatBoost) methods.

Parameter	correct	error	accuracy	macro avg	weighted avg
Precision	1	1	–	1	1
Recall	1	1	–	1	1
F1-Score	1	1	1	1	1
Support	12753	190	12943	12943	12943

As can be seen from the above data, the algorithm correctly identified all 190 fraudulent transactions and all 12753 legal transactions. There are no falsely negative and falsely positive objects. The precision, recall and F1-Score values for each class show their best value.

4.3 k-dimensional

The KD-trees algorithm is most often used as part of the (preliminary stage) k-NN algorithm. The study implemented a classifier based on the k-NN algorithm using KD-trees.

The main parameters of the algorithm are the number of points at which the transition to iteration occurs (leaf_size), the metric for calculating the distance (metric), as well as the number of neighbors that the algorithm considers during operation (n_neighbors), and the algorithm by which the k-NN are searched (algorithm).

By default, leaf_size = 40, metric = «minkowski», n_neighbors = 5, algorithm = «auto». The algorithm parameter will take the value «kd_tree» because it is the KD-trees that will be the basis of the algorithm.

The results of the model trained using the k-NN algorithm based on KD-trees are presented in Table 6.

Table 6. Classification Report for the k-NN method based on KD-trees.

Parameter	correct	error	accuracy	macro avg	weighted avg
Precision	1	1	–	1	1
Recall	1	0,99	–	1	1
F1-Score	1	1	1	1	1
Support	12753	190	12943	12943	12943

As can be seen from the above data, the algorithm correctly identified 189 out of 190 fraudulent transactions and all 12753 legal transactions. One object is identified as falsely negative, there are no false positive objects. The precision, recall and F1-Score values for each class are close to their best value.

An estimate of the operating time of the model obtained during experiments on the same data is presented in Table 7 (Section 5). The average operating time of the model is 0,798228 seconds.

4.4 DBSCAN

The DBSCAN algorithm treats clusters as high-density areas in divided low-density areas. Because of this rather general view, the detected clusters can have any shape. The central component of DBSCAN is the concept of core samples, i.e., samples located in high density areas. Thus, a cluster is a set of core samples, each of which is close to each other, and a set of non-core samples that are close to the core sample but are not core samples themselves. This algorithm has two input parameters that should be selected based on the dataset: \epsilon – the maximum distance between two samples so that they are considered neighbors; min_samples – the number of samples in the vicinity of the point so that it is considered the base/central. These two parameters formally define «density». Higher min_samples or lower \epsilon indicate a higher density required to form a cluster. During the study, various values of the above parameters were sorted out.

A cluster is a set of core samples that can be constructed by recursively taking a core sample, searching for all its neighbors that are core samples, searching for all their neighbors that are core samples, etc. The cluster also has a set of non-core samples that are neighbors of the main sample in the cluster, but are not the main samples. Intuitively, these samples are located on the periphery of the cluster.

To assess the accuracy and efficiency of the model, the indicators of operating time, FMI, Homogeneity, number of clusters, degree of noise (Noise points) are used.

The operating time of the DBSCAN algorithm is shown below (Fig. 2). We can see that as the eps parameter increases, the running time increases, while the min_samples parameter does not significantly affect the running time of the algorithm.

The values of the FMI parameter are shown below (Fig. 3). We can see that when the eps parameter is increased, the FMI values increase. The min_samples parameter has no significant effect.

Below are the indicators of the value of the homogeneity parameter (Fig. 4). We can see that fraudulent transactions clearly fall into a separate class only with small eps values. The min_samples parameter has no significant effect.

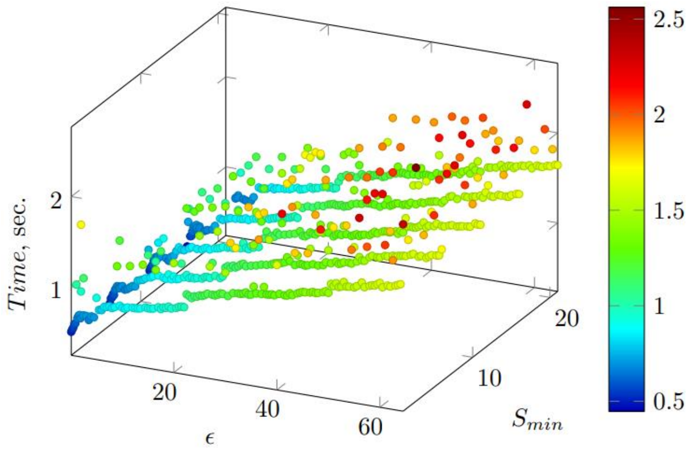

Fig. 2. The operating time of the DBSCAN algorithm.

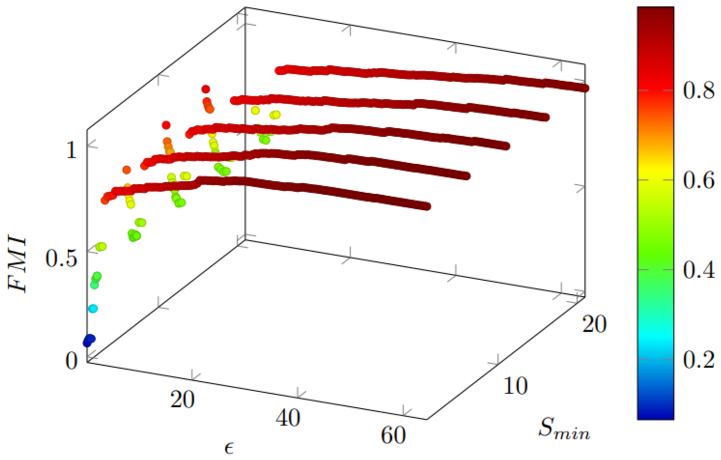

Fig. 3. FMI for the DBSCAN algorithm.

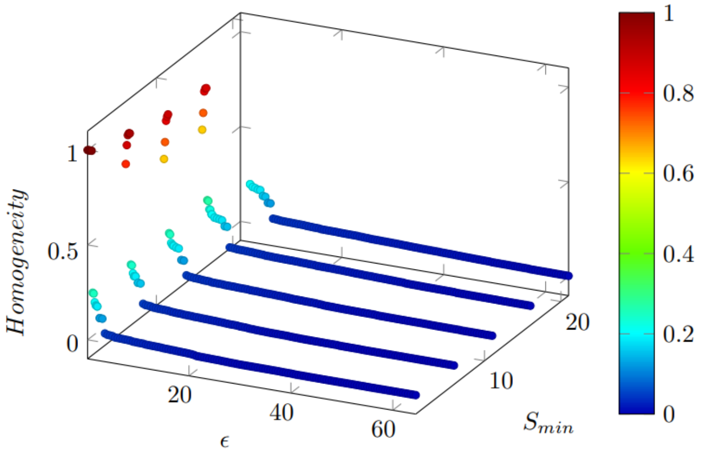

Fig. 4. Homogeneity for the DBSCAN algorithm.

The clusters of the DBSCAN algorithm are shown below (Fig. 5). We can see that with small values of min_samples, the number of clusters can vary significantly depending on eps. However, with an increase in the min_samples parameter, the number of clusters fluctuates significantly less.

The noise indicators are shown below (Fig. 6). We can see that with an increase in the eps parameter, the number of noise points decreases significantly. However, with small eps values, almost all points are identified as noise.

An estimate of the operating time of the model obtained during experiments on the same data is presented in Table 7 (Section 5). The average running time of the model is 1,2715 sec., which is an order of magnitude slower than classification algorithms, for comparison, the running time of the slowest classification algorithm (KD-trees) is 0,798228 sec.

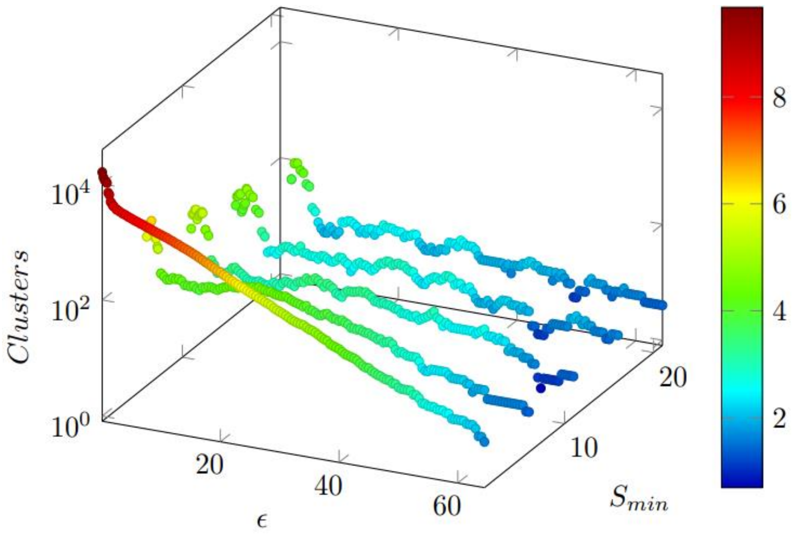

Fig. 5. Clusters of the DBSCAN algorithm.

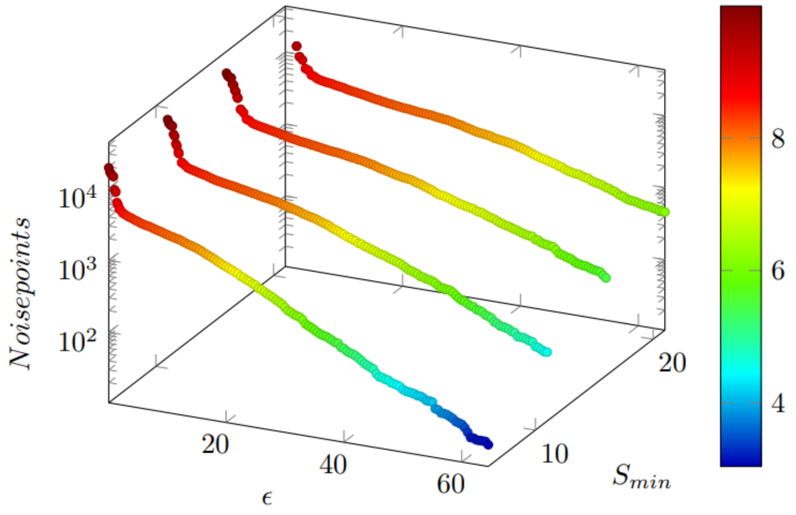

Fig. 6. Noise indicators for the DBSCAN algorithm.

5. Analysis of the results obtained

A comparative analysis of machine learning algorithms is carried out in relation to a sample of transaction data for online purchases. The study demonstrated a different degree of effectiveness of the models and the speed of their work in solving a specific task of searching for fraudulent monetary transactions. The following are the training time indicators for each model (Table 7).

Table 7. The working time of the algorithms, sec.

Method	Average, sec	T_{max}, sec	T_{min}, sec	σ
Random Forest	0,08	0,091674	0,074065	0,003564
GB Scikit-learn	0,046629	0,068366	0,043099	0,005155
GB CatBoost Without the use of cat_features	0,024148	0,052511	0,015452	0,009639
GB CatBoost With the use of cat_features	0,019016	0,035456	0,014613	0,002828
k-мерные деревья	0,798228	1,235431	0,628746	0,140186
DBSCAN	1,271589	4,436863	0,460014	0,445787

From the data presented in Table 7, we can draw the following conclusions:

- the average running time of the GB (cat_features) algorithm is 4.2 times faster than Random Forest, GB (Scikit-learn) is 2.4 times, GB (without cat_features) parameter is 1.2 times, KD-trees are 41.9 times, DBSCAN is 66.8 times.
- the maximum operating time of the GB (cat_features) algorithm is 2,5 times faster than Random Forest, 1,8 times faster than GB (Scikit-learn), 1,5 times faster than GB (without cat_features), 34,7 times faster than KD-trees, 125,1 times faster than DBSCAN.
- the minimum time algorithm GB (cat_features) is faster than Random Forest by 5,06 times, GB (Scikit-learn) by 2,9 times, GB (without cat_features) by 1,05 times, KD-trees by 43,02 times, DBSCAN by 31,5 times.
- the standard deviation algorithm GB (cat_features) works better than Random Forest by 1,3 times, GB (Scikit-learn) by 1,8 times, GB (without cat_features) by 3,4 times, KD-trees by 49,6 times, DBSCAN by 157,6 times.

For the convenience of comparing the operating time, a graph is presented below, which shows the average time of each algorithm (Fig. 7).

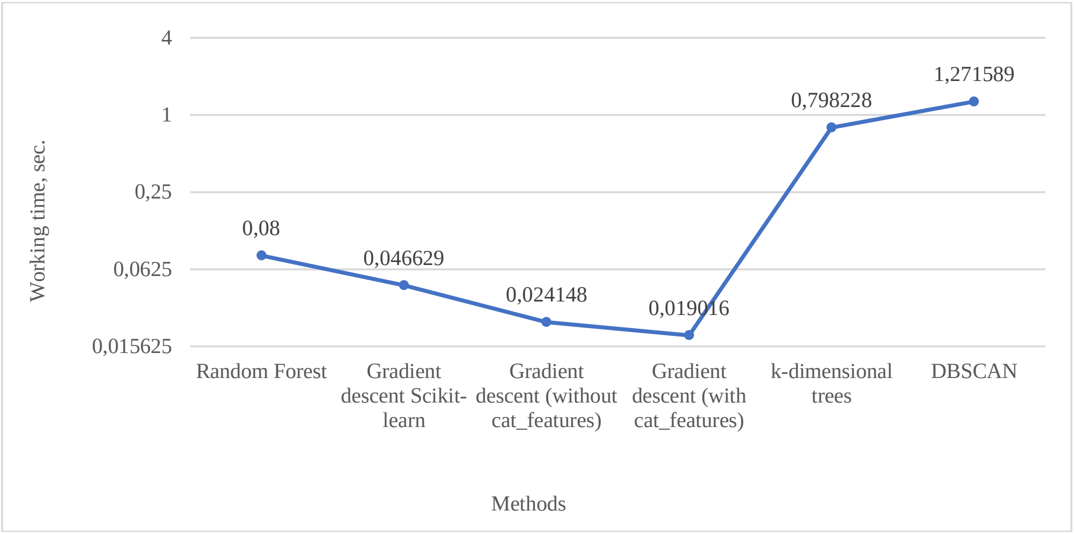

Fig. 7. Model training time.

Based on the indicators of the training time of the models, they can be divided into those that learn faster (random forest (0,08), decision trees with a GB (0,046629), GB (without cat_features)

(0,024148), GB (with cat_features) (0,019016)) and those that learn slower (KD-trees (0,798228), DBSCAN (1,271589)).

The indicators from the Confusion matrix for classification algorithms are also given (Table 8).

Table 8. Confusion matrix for classification algorithms.

Method	Accuracy	TN	FN	FP	TP
Random Forest	1	12753	0	0	190
GB Scikit-learn	1	12753	0	0	190
GB CatBoost Without the use of cat_features	1	12753	0	0	190
GB CatBoost With the use of cat_features	1	12753	0	0	190
k-мерные деревья	0,999	12753	1	0	189

As can be seen from the above data, almost all algorithms, not counting KD-trees, identified all classes without errors, that is, they identified 190 fraudulent transactions and 12753 cor legal rect ones. There are no objects identified as falsely negative or falsely positive. The KD-trees method correctly identified 189 out of 190 fraudulent transactions and all 12,753 legal transactions. One object is identified as falsely negative, there are no false positive objects.

6. Conclusions

The accuracy of detecting fraudulent transactions using Random Forest, GB (Scikit-learn), GB (CatBoost) algorithms is 100%, and the KD-trees algorithm is 99,9%. Thus, we used the transaction classification time as a comparison criterion.

Taking into account the minimum execution time, the GB method was chosen as the main model using the cat_features parameter. It is also worth noting that the implementation of the GB method without the cat_features parameter learns almost as quickly as with it, so it can also be considered within the processing used in the study of a data sample. The average running time of the algorithms is 0,019016 sec. and 0,024148 sec. respectively.

The GB algorithm using the cat_features parameter works 4,2 times faster than Random Forest, GB (Scikit-learn) 2,4 times, GB (without cat_features) 1,2 times, KD-trees 41,9 times, DBSCAN 66,8 times.

The main parameters used by the algorithm in model training and prediction are the index of the first used tree (ntree_start), the index of the first unused tree (ntree_end) and the list of categorical features (cat_features).

The operating time of these models turned out to be minimal, so it can be assumed that these are the most optimal models for processing the data sample under consideration.

References

[1]. T. Hastie, R. Tibshirani, J. Friedman, “The Elements of Statistical Learning”, p. 745, August 2009.
[2]. C. Chio, D. Freeman, “Machine Learning and Security”, p. 386, February 2017.
[3]. Power, R. “Tangled Web: Tales of Digital Crime from the Shadows of Cyberspace”, pp. 396-397, 2000.
[4]. Uyazvimosti i ugrozy veb-prilozhenij v 2020-2021 gg. Official website – URL: https://www.ptsecurity.com/ru-ru/research/analytics/web-vulnerabilities-2020-2021/#id5.
[5]. Andress, J., “The Basics of Information Security”, Second Edition, Chapter 3, Authorization and Access Control, p. 190, 2014.
[6]. R. Hamsa Veni, A. Hariprasad Reddy, C. Kesavulu, “Identifying Malicious Web Links and Their Attack Types in Social Networks”, International Journal of Scientific Research in Computer Science, Engineering and Information Technology, pp.1060-1066, March-April 2018.
[7]. R. F. Fouladi, C. E. Kayatas, E. Anarim, “Frequency based DDoS attack detection approach using naive Bayes classification”, International Conference on Telecommunications and Signal Processing (TSP), June 2016.
[8]. Todd A. Stephenson, “An Introduction to Bayesian Network Theory and Usage”, p. 29, 2000.

[9]. D. Atienza, A. Herrero, E. Corchado, “Neural analysis of http traffic for web attack detection”, Computational Intelligence in Security for Information Systems Conference, pp.201-212, January 2015.
[10]. B. Goyal, M. Bansal, “Competent Approach for Type of Phishing Attack Detection Using Multi-Layer Neural Network”, International Journal of Advanced Engineering Research and Science, pp. 210-215, January 2017.
[11]. Hervé Abdi, “A neural network primer”, Journal of Biological Systems, pp. 247-281, 1994.
[12]. Sivak M. A., Timofeev V. S., “Configuring robust neural networks to solve the classification problem”, Reports of Tomsk State University of Control Systems and Radioelectronics, pp.26-32, 2021.
[13]. N. Florian Epp, R. Funk, C. R. Cappo, “Anomaly-based web application firewall using HTTP-specific features and one-class SVM”, September 2017.
[14]. Z. Tian, “Distributed Deep Learning System for Web Attack Detection on Edge Devices”, IEEE Transactions on Industrial Informatics, November 2019.
[15]. Ye Jin, “A DdoS attack detection method based on SVM in software defined network”, Security and Communication Networks, April 2018.
[16]. S. Suthaharan, “Support Vector Machine”, Machine Learning Models and Algorithms for Big Data Classification, pp. 207-235, January 2016.
[17]. Otchet ob atakah na onlajn-resursy rossijskih kompanij. official website [https://www.ptsecurity.com/ru-ru/] – URL: https://rt-solar.ru/upload/iblock/34a/5w4h9o57axovdbv3ng7givrz271ykir3/Ataki-na-onlayn_resursy-rossiyskikh-kompaniy-v-2022-godu.pdf?ysclid=lubdnvft2p622633541.
[18]. Aktual'nye kiberugrozy: itogi 2021 goda. official website [https://www.ptsecurity.com/ru-ru/] – URL: https://www.ptsecurity.com/ru-ru/research/analytics/cybersecurity-threatscape-2021/.
[19]. Garfinkel, S. and Spafford, E.H., “Web Security and Commerce”, O'Reilly and Associates, pp. 450-470, February 1997.
[20]. Martynov A., Kandybla V., “Metod gradientnogo spuska v mashinnom obuchenii”, Zhurnal «Shag v nauku», pp. 4-8, 2022.
[21]. CatBoost is a high-performance open-source library for gradient boosting on decision trees. official website [https://catboost.ai/?ysclid=lwdjifxlqs185272725] – URL: https://catboost.ai/en/docs/concepts/speed-up-training?ysclid=lwdk46v95o460729700.
[22]. Gradient Boosting. official website [https://scikit-learn.org/stable/] – URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html.

Информация об авторах / Information about authors

Мария Анатольевна ЛАПИНА – кандидат физико-математических наук, доцент кафедры информационной безопасности автоматизированных систем Северо-Кавказского федерального университета. Сфера научных интересов: цифровые технологии, управление информационной безопасностью, процессный подход, образовательный процесс, криптография.

Maria Anatolyevna LAPINA – Cand. Sci. (Phys.-Math.), Associate Professor of the Department of Information Security of Automated Systems of the North Caucasus Federal University. Research interests: digital technologies, information security management, process approach, educational process, cryptography.

Виталия Валентиновна МОВЗАЛЕВСКАЯ – студентка кафедры информационной безопасности автоматизированных систем Северо-Кавказского федерального университета. Сфера научных интересов: программирование, машинное обучение.

Vitaliya Valentinovna MOVZALEVSKAYA – student of the Department of Information Security of Automated Systems of the North Caucasus Federal University. Research interests: programming, machine learning.

Марина Евгеньевна ТОКМАКОВА – студентка кафедры информационной безопасности автоматизированных систем Северо-Кавказского федерального университета. Сфера научных интересов: криптография, машинное обучение.

Marina Evgenievna TOKMAKOVA – student of the Department of Information Security of Automated Systems of the North Caucasus Federal University. Research interests: cryptography, machine learning.

Михаил Григорьевич БАБЕНКО – доктор физико-математических наук, заведующий кафедрой вычислительной математики и кибернетики Северо-Кавказского федерального университета. Сфера научных интересов: алгебраические структуры в полях Галуа, модулярная арифметика, нейрокомпьютерные технологии, цифровая обработка сигналов, криптографические методы защиты информации.

Mikhail Grigorievich BABENKO – Dr. Sci. (Phys.-Math.), Head of the Department of Computational Mathematics and Cybernetics of the North Caucasus Federal University. Research interests: algebraic structures in Galois fields, modular arithmetic, neurocomputer technologies, digital signal processing, cryptographic methods of information protection.

Виктор Павлович КОЧИН – кандидат технических наук, проректор по учебной работе и интернационализации образования Белорусского государственного университета. Сфера научных интересов: комплексные системы защиты информации, Информационно-коммуникационные технологии.

Viktor Pavlovich KOCHIN – Cand. Sci. (Tech.), Vice-Rector for Academic Affairs and Internationalization of Education of the Belarusian State University. Research interests: integrated information security systems, Information and communication technologies.