DOI: 10.15514/ISPRAS-2023-1(2)-1

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: 0000-0000-0000-0000 < *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 time- and quality-efficient algorithm for the data sample under consideration has been defined and implemented. 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. Analysis of the use of machine learning algorithms to prevent attacks in the web environment. Trudy ISP RAN/Proc. ISP RAS

Relevance. The security issues of the web environment and user applications are very relevant. Due to the fact that web applications are accessible over the network and very often contain vulnerabilities, they are regularly the targets of cyber-attacks. 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.

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, which significantly increases the effectiveness of website protection.

The Bayesian network. A Bayesian network is a graphical model that encodes probabilistic relationships between variables of interest [11, 12]. These networks are a combination of two different mathematical fields: graph theory and probability theory [20].

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 a large number of neurons used for processing and analyzing information, its structure is similar to the nervous system of a living organism [13, 14]. Neural networks consist of basic blocks similar to neurons. These blocks interact with each other on the basis of connections, the strength of which can be changed as a result of the learning process or modification of the algorithm [19].

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.

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 [15-17]. 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 [18]. 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 [5]. The most common attacks include the following: attacks on clients (98%), data leakage (91%), 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%) [8] (Fig. 1.).

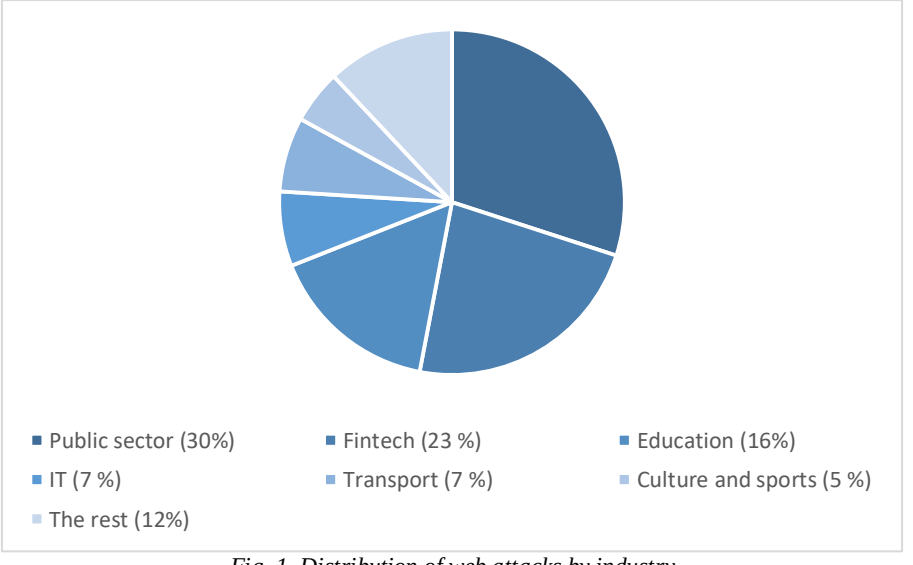

Fig. 1. Distribution of web attacks by industry

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 [6]. 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 are able to 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 [7].

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.

The most common attacks include the following:

1. Implementation of SQL commands. Embed SQL commands into a web form or query that will be processed by the database, which can take full control over it.

2. Implementation of shell commands. By embedding a shell command into a script or application, an attacker gains remote access to the system and executes any commands, including installing additional malware.

3. Cross-site scripting. This is an attack in which an attacker inserts a client script (for example, JavaScript) on a site page that can collect information about visitors and fully control their session.

4. Cross-Site Request Forgery (CSRF). The CSRF attack uses malicious websites or emails to automatically perform actions on behalf of the user without his knowledge.

1. Introduction

The urgent need to ensure information security on the Internet is easily explained by several factors, including the massive interconnection of heterogeneous and distributed systems, the availability of large amounts of confidential information in end systems maintained by corporations and government agencies, the easy spread of automated malicious software by attackers, the ease with which computer crimes can be committed anonymously, and the lack of forensic evidence in computer crimes, this makes it extremely difficult to detect and prosecute criminals.

With the development of web technologies and a large increase in the volume of information, this problem is only getting worse, and web security is becoming more and more important. 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 and effective 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.

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

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 a number 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 [4]. 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 – theoretical description of models; section 3 – modeling; section 4 – analysis of the results; section 5 – conclusions.

2. 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.

2.1. Forests of decision trees

This method consists in combining several classifiers, which creates a more complex, in most cases more effective, classifier. Combining decision trees is a proven technique for creating high-quality classifiers. Such forests of decision trees are also often called ensembles.

The article discusses three forests that are most often used in practice: random forests, decision trees with acceleration or gradient boosting (gradientboosted decision trees), gradient descent.

2.1.1. 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, generalized forecasts of random forests are performed taking into account the statistical mode (typicality) of individual forecasts from classification trees

(i.e., each tree «votes» for its own version of the forecast) and the statistical average of the forecasts of individual trees for regression trees.

To build each tree of a random forest, the following happens:

1. A subset of training objects is randomly selected from the entire dataset. This subset may contain duplicate objects.

2. 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.

3. A decision tree is built on the selected subset of data and features. When building a tree, an information criterion is used (for example, the entropy criterion), which allows you to choose the best feature for splitting data at each level of the tree.

4. 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:

1. Generate a selection of X_n bootstraps.
2. Build a decision tree b_n from a sample of X_n:
   - according to a given criterion, the best feature is selected, a split occurs in the tree, and repeats until the selection is exhausted;
   - the tree is built until there are no more than nmin objects in each leaf or until it reaches a certain height;
   - for each partition, m random features are first selected from n initial ones, and the optimal separation of the sample is sought only among them.

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:

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

in classification tasks, it is recommended to take:

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

and in regression tasks:

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

where n is the number of features.

2.1.2. Gradient descent

Gradient descent is an optimization algorithm used to minimize errors in a machine learning model.

Table 1. Advantages and disadvantages of the algorithm

№	Advantages	Disadvantages
1	The gradient descent method is very flexible and can process both continuous and discrete data	Adding new features or data is a complex process because it requires rebuilding the entire model
2	With a relatively short setup time, the prediction accuracy is high	The decision trees that make up the gradient descent method can be difficult to interpret
3	It has a high resistance to noise in the training data	The gradient descent method can be quite difficult to set up and optimize

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 [2]. In other words, if the function f(x, y, z) is given, then the gradient is calculated using the formula:

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

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) in order 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» [10].

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 optimal learning rate is a key parameter, since too large a step can lead to skipping the minimum, and too small makes the optimization process slow.

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

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

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

There are three main implementations of gradient descent: LightGBM, XGBoost and CatBoost, below are the main characteristics of these implementations (Table 2.).

Table 2. Comparative analysis of gradient descent algorithms

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

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, also this implementation is stated as the fastest and optimized, therefore, the study compares it with Scikit-learn, the most frequently implemented.

2.2. k-dimensional trees

A 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 nearest neighbor (kNN) 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:

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

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 k-d 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.

There are homogeneous and heterogeneous k-d trees. For homogeneous k-d trees, each node stores a record. In the heterogeneous variant, the internal nodes contain only keys, the leaves contain links to records.

In an inhomogeneous k-d tree:

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

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 x1<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.

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

2.3. 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 useful for detecting 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:

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

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:

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

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.

3. 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 correct transaction (label column, Fig. 2.). The data set is taken from [4]. 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 (Fig.2.).

	accountAgeDays	numItems	localTime	paymentMethod	paymentMethodAgeDays	label
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

Fig. 2. Model training time

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 4), as well as characteristics from the Confusion matrix and Classification Report for classification algorithms.

The Confusion matrix clearly shows how accurately the ML model performs the task of classifying objects. The following concepts are used to describe the combinations that can be obtained when comparing algorithm responses and true labels:

- True Positive (TP) – the model identified the object as class «1», and it really is class «1»;
- False Positive (FP) – the model identified the object as class «1», while it represents class «0» (the model was wrong);
- True Negative (TN) – the model identified the object as class «0», and it really is class «0»;
- Fasle Negative (FN) – the model identified the object as class «0», while it represents class «1» (the model was wrong).

Also, in the confusion matrix there is a quantitative accuracy indicator that reflects the number of correctly classified samples from all the samples present in the test set.

The characteristics from the Classification Report are also based on the indicators described above and allow you to get a more complete picture of the quality of model training within each class. The Classification Report includes the following metrics: precision, recall, F1 score and support. They are calculated for each class separately, which helps to get more detailed information about the operation of the model and the effectiveness of the selected learning algorithm.

Precision is an indicator of how many True Positive results were actually correct. In fact, it is the percentage ratio of the True Positive forecasts of the model to the total number of positive forecasts. This indicator demonstrates the ability of the trained model to distinguish one class from another (formula 9).

$$precision = \frac{TP}{TP - FP}. \quad (9)$$

Recall is a measure that shows what proportion of objects of a positive class out of all objects of a positive class the model has found. Demonstrates the ability of the model to detect the current class (formula 10).

$$recall = \frac{TP}{TP + FN}. \quad (10)$$

F1 score is the average harmonic value for recall and precision. The metric can take values from 0 (when either recall or precision tend to zero, the worst value) to 1 (the best value) (formula 11).

$$F1 = \frac{2 * TP}{2 * TP + FP + FN}. \quad (11)$$

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

Clustering algorithms have a large number of their own specific metrics for evaluating the quality of the model.

3.1. Forests of decision trees

Before you start training machine learning models, you need to prepare a dataset for use.

First of all, it is necessary to perform direct encoding (One-Hot Encoding, NONE), that is, to convert categorical features into numerical ones without arbitrary ordering. This means that we do not number the attribute values in ascending order, for example, numbers from 1 to 3, because this can lead to incorrect operation of machine learning algorithms, but create a new column for each variant of the attribute value, then assign the value «1»" if the category is suitable, «0» if not.

There is only one categorical feature in the selected dataset – PaymentMethod. Thus, if the payment method during the transaction is a credit card, then the value «1» will be in the paymentMethod_creditcard column, and «0» in the other columns with payment methods.

Further, after dividing the data into training and test data, the following sets were obtained:

- X_train – a data set for training a model without a binary label label;
- X_test – a data set for testing the performance of a model without a binary label;
- Y_train – a dataset for training a model containing only the label column (target values);
- Y_test – a dataset for testing the health of the model, containing only the label column (target values).

During training, the model is given access to the X_train and Y_train sets so that it can find the relationship between the transaction attributes (X_train) and the target values of the label column (Y_train).

3.1.1. 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.

To assess the accuracy and efficiency of the model, the above-described metrics are used, in particular, work time indicators, Confusion matrix and Classification Report.

The results of the model trained using the random forest algorithm are presented in Table 3.

Table 3. Classification Report for the random forest method

	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 identified all classes without errors, that is, it identified 190 fraudulent transactions and 12753 correct 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 8 (Section 4). The average operating time of the model is 0,08 seconds..

3.1.2. Gradient descent

One of the implementations of the gradient descent 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.

First, you need to import the Gradient Boosting Classifier 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). Gradient boosting 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.

To assess the accuracy and efficiency of the model, the above-described metrics are used, in particular, work time indicators, Confusion matrix and Classification Report.

The results of the model trained using the gradient descent algorithm in the implementation of Scikit-learn are presented in Table 4.

Table 4. Classification Report for the gradient descent method when implementing Scikit-learn

	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 identified all classes without errors, that is, it identified 190 fraudulent transactions and 12753 correct 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 8 (Section 4). The average operating time of the model is 0.046629seconds.

CatBoost is another implementation of the gradient descent 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).

To assess the accuracy and efficiency of the model, the above-described metrics are used, in particular, work time indicators, Confusion matrix and Classification Report.

The results of the model trained using the gradient descent algorithm for the implementation of CatBoost are presented in Table 5.

Table 5. Classification Report for the gradient descent method when implementing CatBoost without using the cat_features 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 identified all classes without errors, that is, it identified 190 fraudulent transactions and 12753 correct 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 8 (Section 4). The average operating time of the model is 0,024148 seconds.

By default, categorical features are not taken into account in the process of training a model using Cut Boost, 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 (Fig.3.) and with it, i.e. using the cat_features parameter (Fig. 4.).

	Feature Id	Importances
0	accountAgeDays	78.826096
1	localTime	6.487206
2	numItems	4.092288
3	paymentMethodAgeDays	3.228341
4	paymentMethod_storecredit	3.123910
5	paymentMethod_creditcard	2.912722
6	paymentMethod_paypal	1.329436

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

	Feature Id	Importances
0	accountAgeDays	84.247069
1	paymentMethodAgeDays	8.237712
2	localTime	3.882424
3	numItems	3.632795
4	paymentMethod	0.000000

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

As you can see, the importance of features in the learning process for models with the cat_features parameter and without this parameter is quite different.

To evaluate the accuracy and efficiency of the model without preprocessing categorical features, the indicators of operation time, Confusion matrix and Classification Report are used.

The results of the model trained using the gradient descent algorithm when implementing Cut Boost using the cat_features parameter is presented in Table 6.

Table 6. Classification Report for the gradient descent method when implementing CatBoost using the cat_features 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 identified all classes without errors, that is, it identified 190 fraudulent transactions and 12753 correct 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 8 (Section 4). 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.

3.2. k-dimensional trees

The k-dimensional tree algorithm is most often used as part of the (preliminary stage) k nearest neighbor algorithm. As part of the study, a classifier based on the k-nearest neighbor algorithm using k-dimensional trees was implemented.

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 nearest neighbors 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 k-dimensional trees that will be the basis of the algorithm.

To assess the accuracy and efficiency of the model, the above-described metrics are used, in particular, work time indicators, Confusion matrix and Classification Report.

The results of the model trained using the k nearest neighbor algorithm based on k-dimensional trees are presented in Table 7.

Table 7. Classification Report for the k nearest neighbor method based on k-dimensional trees

	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 correct 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 8 (Section 4). The average operating time of the model is 0,798228 seconds.

3.3. DBSCAN

DBSCAN is one of the most well–known and frequently used clustering algorithms. That is why it was decided to test the effectiveness of his work within the framework of the ongoing research.

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 data set: epsilon (eps) – 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 eps indicate a higher density required to form a cluster. In the course of 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 core sample in the cluster, but are not themselves 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 (Noisepoints) are used.

The operating time of the DBSCAN algorithm is shown below (Fig. 5.). 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.

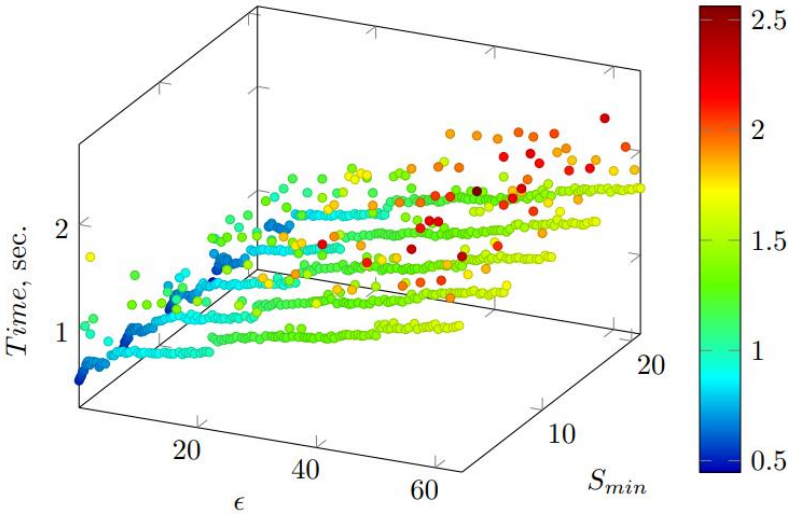

Fig. 5. The operating time of the DBSCAN algorithm

The values of the FMI parameter are shown below (Fig. 6.). We can see that when the eps parameter is increased, the FMI values increase. The min_samples parameter has no significant effect.

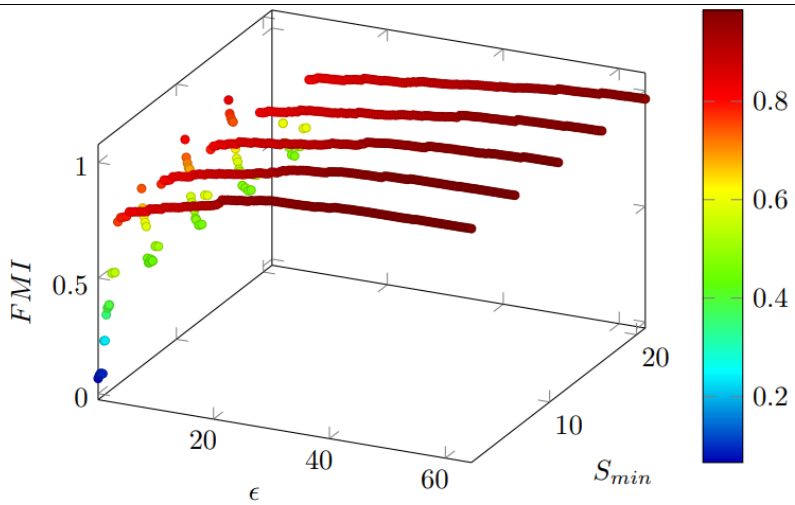

Fig. 6. FMI for the DBSCAN algorithm

Below are the indicators of the value of the homogeneity parameter (Fig. 7.). 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.

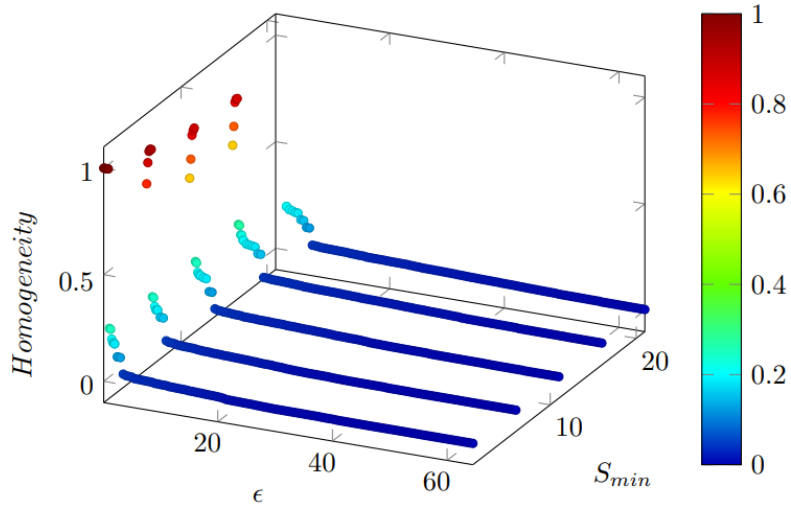

Fig. 7. Homogeneity for the DBSCAN algorithm

The clusters of the DBSCAN algorithm are shown below (Fig. 8.). 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.

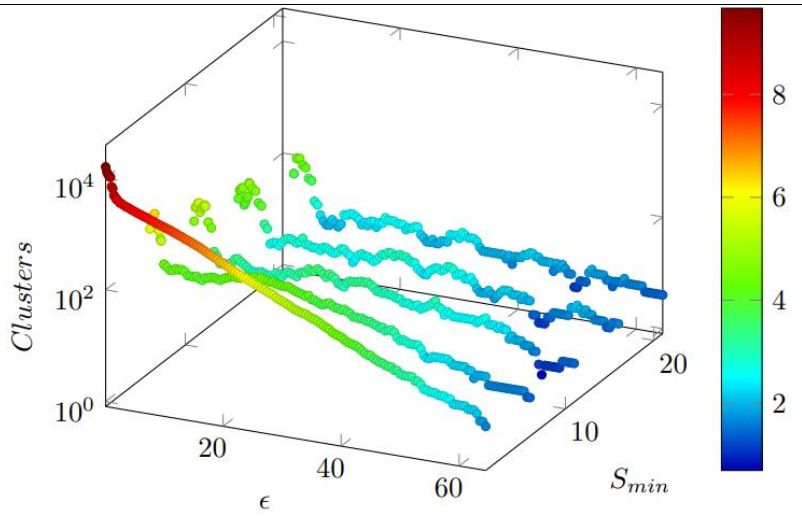

Fig. 8. Clusters of the DBSCAN algorithm

The noise indicators are shown below (Fig. 9.). 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.

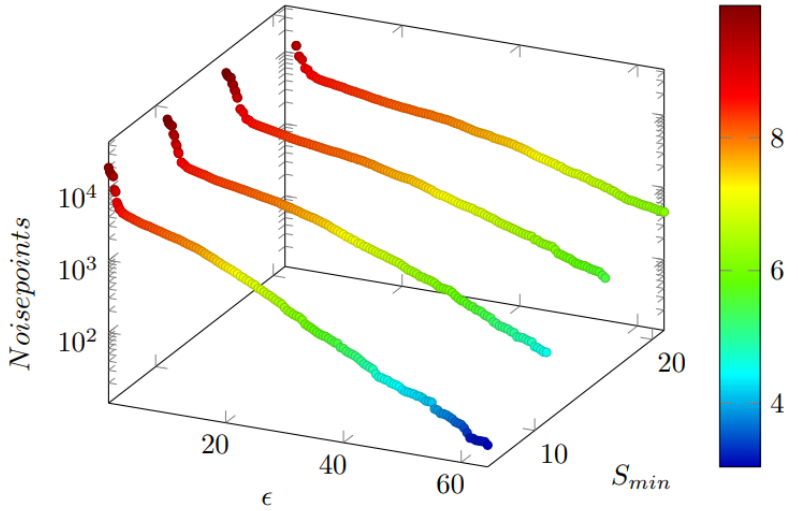

Fig. 9. Noise indicators for the DBSCAN algorithm

An estimate of the operating time of the model obtained during experiments on the same data is presented in Table 8 (Section 4). The average running time of the model is 1,2715 seconds, which is an order of magnitude slower than classification algorithms, for comparison, the running time of the slowest classification algorithm (k-dimensional trees) is 0,798228 seconds.

4. 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 3., Fig. 2.):

Table 8. The working time of the algorithms

Method	Average, sec	T_{max}, sec	T_{min}, sec	σ
Random Forest	0,08	0,091674	0,074065	0,003564
Gradient descent Scikit-learn	0,046629	0,068366	0,043099	0,005155
CatBoost gradient descent without cat_features	0,024148	0,052511	0,015452	0,009639
CatBoost gradient descent with cat_features	0,019016	0,035456	0,017613	0,002828
k-dimensional trees	0,798228	1,235431	0,628746	0,140186
DBSCAN	1,271589	4,436863	0,460014	0,445787

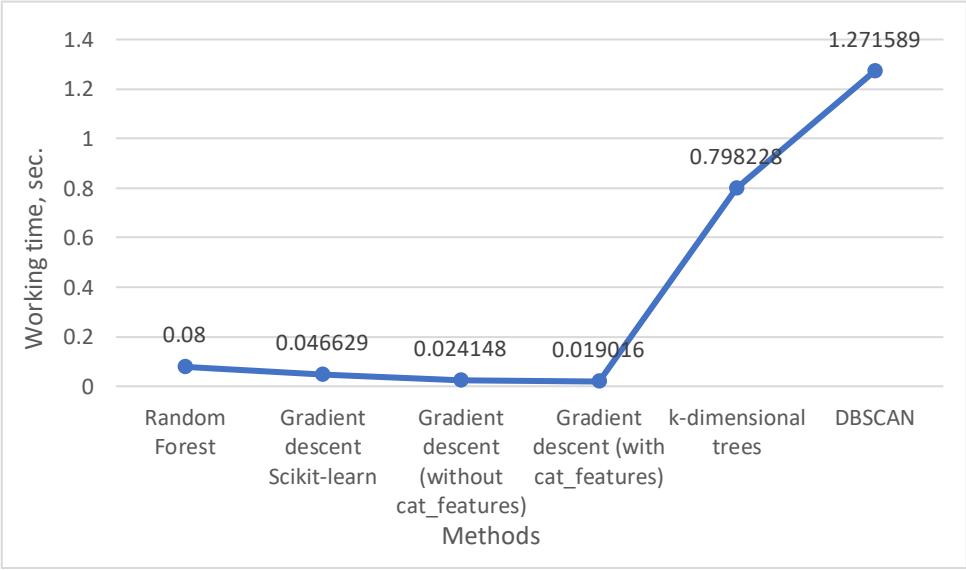

Fig. 10. Model training time

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 boosting gradient (0,046629), gradient descent (without cat_features) (0,024148), gradient descent (with cat_features) (0,019016)) and those that learn slower (k-dimensional trees (0,798228), DBSCAN (1,271589)).

The indicators from the Confusion matrix for classification algorithms are also given (Table. 9.):

Table 9. Confusion matrix for classification algorithms

Method	Accuracy	TN	FN	FP	TP
Random Forest	1	12753	0	0	190
Gradient descent Scikit-learn	1	12753	0	0	190
CatBoost gradient descent without cat_features	1	12753	0	0	190
CatBoost gradient descent with cat_features	1	12753	0	0	190
k-dimensional trees	1	12753	1	0	189

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

5. Conclusions

Taking into account the minimum execution time and metrics, the gradient descent method using the cat_features parameter was chosen as the main model. It is also worth noting that the implementation of the gradient descent method without this parameter is trained almost as quickly as with it, so it can also be considered within the framework of processing used in the study of data sampling.

Therefore, these methods should be used to process this type of data. At the same time, on the data under consideration, quality metrics take the following values: gradient descent without the cat_features parameter (TN = 12753, FN = 0, FP = 0, TP = 190), gradient descent from the cat_features parameter (TN = 12753, FN = 0, FP = 0, TP = 190). 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. 2017 – URL: https://link.springer.com/book/10.1007/978-0-387-84858-7.

[2]. Garfinkel, S. and Spafford, E.H.: Web Security and Commerce. O'Reilly and Associates. 1997 – URL: https://lira.epac.to/DOCS-TECH/Security/Web/Web%20Security%20&%20Commerce.pdf.

[3]. Power, R. Tangled Web: Tales of Digital Crime from the Shadows of Cyberspace. 2000 – URL: https://lira.epac.to/DOCS-TECH/Forensics/Tangled%20Web%20-%20Tales%20of%20Digital%20Crime%20from%20the%20Shadows%20of%20Cyberspace.pdf.

[4]. C. Chio, D. Freeman: Machine Learning and Security. 2017 – URL: https://www.shabakeh-mag.com/sites/default/files/files/attachment/1396/06/1505397112.pdf.

[5]. Uyazvimosti i ugrozy veb-prilozhenij v 2020-2021 gg. official website [https://www.ptsecurity.com/ru-ru/] – URL: https://www.ptsecurity.com/ru-ru/research/analytics/web-vulnerabilities-2020-2021/#id5.

[6]. 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.

[7]. 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/.

[8]. Andress, J. The Basics of Information Security (Second Edition). Chapter 3 – Authorization and Access Control. 2014 – URL: https://www.sciencedirect.com/science/article/abs/pii/B9780128007440000038.

[9]. A. Trask: Grokking Deep Learning. 2019 – URL: https://codelibrary.info/files/1405_Grokaem-glubokoe-obuchenie.pdf.

[10]. Martynov A., Kandybla V.: Metod gradientnogo spuska v mashinnom obuchenii. Zhurnal «Shag v nauku». 2022 – URL: https://cyberleninka.ru/article/n/metod-gradientnogo-spuska-v-mashinnom-obuchenii.

[11]. 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. 2018 – URL: https://res.ijsrcseit.com/page.php?param=CSEIT1833525.

[12]. 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). 2016 – URL: https://www.researchgate.net/publication/304371694_Frequency_Based_DDoS_Attack_Detection_Approach_Using_Naive_Bayes_Classification.

[13]. D. Atienza, A. Herrero, E. Corchado: Neural analysis of http traffic for web attack detection. Computational Intelligence in Security for Information Systems Conference. 2015 – URL: https://link.springer.com/chapter/10.1007/978-3-319-19713-5_18.

[14]. 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. 2017 – URL: https://www.researchgate.net/publication/313320107_A_Competent_Approach_for_Type_of_Phishing_Attack_Detection_Using_Multi-Layer_Neural_Network.

[15]. N. Florian Epp, R. Funk, C. R. Cappo: Anomaly-based web application firewall using HTTP-specific features and one-class SVM. 2017 – URL: https://www.researchgate.net/publication/319490376_Anomaly-based_Web_Application_Firewall_using_HTTP-specific_features_and_One-Class_SVM.

[16]. Z. Tian: Distributed Deep Learning System for Web Attack Detection on Edge Devices. IEEE Transactions on Industrial Informatics. 2019 – URL: https://ieeexplore.ieee.org/document/8821336.

[17]. Ye Jin: A DdoS attack detection method based on SVM in software defined network. Security and Communication Networks. 2018 – URL: https://www.researchgate.net/publication/324752946_A_DDoS_Attack_Detection_Method_Based_on_SVM_in_Software_Defined_Network.

[18]. S. Suthaharan: Support Vector Machine. Machine Learning Models and Algorithms for Big Data Classification. 2016 – URL: https://link.springer.com/chapter/10.1007/978-1-4899-7641-3_9.

[19]. A neural network primer. Journal of Biological Systems. 1994 – URL: https://doi.org/10.1142/S0218339094000179.

[20]. Todd A. Stephenson: An Introduction to Bayesian Network Theory and Usage. 2000 – URL: https://infoscience.epfl.ch/record/82584.

Информация об авторах / Information about authors

Lapina Maria Anatolyevna – Associate Professor of the Department of Information Security of Automated Systems of the North Caucasus Federal University.

Movzalevskaya Vitaliya Valentinovna – a student of the Department of Information Security of Automated Systems of the North Caucasus Federal University.

Tokmakova Marina Evgenievna – a student of the Department of the Department of Information Security of Automated Systems of the North Caucasus Federal University.

Babenko Mikhail Grigorievich – Head of the Department of Computational Mathematics and Cybernetics of the North Caucasus Federal University.

Kochin Viktor Pavlovich – Vice-Rector for Academic Affairs and Internationalization of Education of the Belarusian State University.