DOI: 10.15514/ISPRAS-2025-37(4)-24

Research of Machine Learning Methods for Detecting Network Attacks

[1] *Lapina M.A., ORCID: 0000-0001-8117-9142 <mlapina@ncfu.ru>*
[1] *Podruchny N.V., ORCID: 0009-0007-6710-1104 <podrucnyjnazar@gmail.com>*
[2] *Rusanov M.A. ORCID: 0009-0000-7069-7542 <mix.rusanoff@yandex.ru>*
[1] *Babenko M.G., ORCID: 0000-0001-7066-0061 <mgbabenko@ncfu.ru>*

[1] *North Caucasus Federal University,*
Russia, 355017, Stavropol, Pushkina St., 1.

[2] *Moscow University of Finance and Law,*
Russia, 115191, Moscow, Stromynka St., 4.

Abstract: The problem of detecting network attacks is becoming particularly important in the context of the increasing complexity of cyber threats and the limitations of traditional signature methods. This paper provides a comprehensive analysis of five machine learning algorithms with a focus on interpretability of models and processing of unbalanced Simulated Network Traffic data. The main objective is to increase the accuracy of detecting cyber-attacks, including DDoS and port scanning, using a decision tree, logistic regression, random forest and other methods. The study was performed in Python 3.13 using the scikit-learn, XGBoost and TensorFlow libraries. The choice of tools is determined by the specifics of the task: for classical methods (trees, logistic regression) and ensemble approaches (Random Forest, XGBoost), scikit-learn turned out to be optimal, and for neural network experiments (RProp MLP) TensorFlow/Keras provided a user-friendly interface for prototyping. PyTorch was not used because it did not provide advantages for binary classification on structured data, but its use could be justified for analyzing sequences or unstructured logs in future research. The decision tree demonstrated the highest accuracy – 99.4% with a depth of 5 and the selection of 8 key features out of 18. After tuning, gradient boosting showed a comparable result – 99.58%, but its training took significantly longer (576 seconds versus 69 for the decision tree). The random forest achieved 97.98% accuracy, while the logistic regression achieved 96.53%. Naive Bayes proved to be the least effective (86.48%), despite attempts to improve using PCA. The linear regression transformed into a classifier showed an accuracy of 94.94%, which is lower than the ensemble methods, but acceptable for the basic approach. The practical value of the work is confirmed by testing on real network data. The results obtained can form the basis of hybrid systems combining several algorithms to increase detection reliability. For example, combining a fast decision tree for primary analysis and gradient boosting to refine complex cases will allow you to balance between speed and accuracy. Separately, it is worth noting the importance of interpretability of models: trees and logistic regression not only showed good results but also allowed us to identify key signs of attacks, which is critical for integration into existing security systems.

Keywords: machine learning; deep learning; network traffic analysis; anomaly detection; cybersecurity.

For citation: Lapina M.A., Podruchny N.V., Rusanov M.A., Babenko M.G. Research of machine learning methods for detecting network attacks. Trudy ISP RAN/Proc. ISP RAS, vol. 37, issue 4, part 2, 2025, pp. 147-174. DOI: 10.15514/ISPRAS-2025-37(4)-24.

Acknowledgements: The research was supported by the Russian Science Foundation Grant No 25-71-30007, https://rscf.ru/en/project/25-71-30007/.

Исследование методов машинного обучения для выявления сетевых атак

[1] *Лапина М.А., ORCID: 0000-0001-8117-9142 <mlapina@ncfu.ru>*
[1] *Подручный Н.В., ORCID: 0009-0007-6710-1104 <podrucnyjnazar@gmail.com>*
[2] *Русанов М.А., ORCID: 0009-0000-7069-7542 <mix.rusanoff@yandex.ru>*
[1] *Бабенко М.Г., ORCID: 0000-0001-7066-0061 <mgbabenko@ncfu.ru>*

[1] *Северо-Кавказский федеральный университет*
Россия, 355017, г. Ставрополь, ул. Пушкина, д.1.

[2] *Московский финансово-юридический университет*
Россия, 115191, г. Москва, ул. Стромынка, д. 4.

Аннотация: Проблема обнаружения сетевых атак приобретает особую значимость в условиях роста сложности киберугроз и ограниченности традиционных сигнатурных методов. В данной работе проведен комплексный анализ пяти алгоритмов машинного обучения с фокусом на интерпретируемость моделей и обработку несбалансированных данных Simulated Network Traffic. Основная задача – повышение точности детектирования кибератак, включая DDoS и сканирование портов, с использованием дерева решений, логистической регрессии, случайного леса и других методов. Исследование выполнено на Python 3.13 с применением библиотек scikit-learn, XGBoost и TensorFlow. Выбор инструментов обусловлен спецификой задачи: для классических методов (деревья, логистическая регрессия) и ансамблевых подходов (Random Forest, XGBoost) scikit-learn оказался оптимальным, а для нейросетевых экспериментов (RProp MLP) TensorFlow/Keras предоставил удобный интерфейс для прототипирования. PyTorch не использовался, так как для бинарной классификации на структурированных данных он не давал преимуществ, но его применение могло бы быть оправдано для анализа последовательностей или неструктурированных логов в будущих исследованиях. Дерево решений продемонстрировало наивысшую точность – 99.4% при глубине 5 и выделении 8 ключевых признаков из 18. Градиентный бустинг после настройки показал сопоставимый результат – 99.58%, однако его обучение заняло значительно больше времени (576 секунд против 69 дерева решений). Случайный лес достиг точности 97.98%, а логистическая регрессия – 96.53%. Наивный Байес оказался наименее эффективным (86.48%), несмотря на попытки улучшения с помощью PCA. Линейная регрессия, преобразованная в классификатор, показала точность 94.94%, что ниже ансамблевых методов, но приемлемо для базового подхода. Практическая ценность работы подтверждена тестированием на реальных сетевых данных. Полученные результаты могут лечь в основу гибридных систем, комбинирующих несколько алгоритмов для повышения надежности детектирования. Например, сочетание быстрого дерева решений для первичного анализа и градиентного бустинга для уточнения сложных случаев позволит балансировать между скоростью и точностью. Отдельно стоит отметить важность интерпретируемости моделей: деревья и логистическая регрессия не только показали хорошие результаты, но и позволили выявить ключевые признаки атак, что критично для интеграции в существующие системы безопасности.

Ключевые слова: машинное обучение; глубокое обучение; анализ сетевого трафика; обнаружение аномалий; кибербезопасность.

Для цитирования: Лапина М.А., Подручный Н.В., Русанов М.А., Бабенко М.Г. Исследование методов машинного обучения для выявления сетевых атак. Труды ИСП РАН, том 37, вып. 4, часть 2, 2025 г., стр. 147–174 (на английском языке). DOI: 10.15514/ISPRAS–2025–37(4)–24.

Благодарности: Исследование выполнено при поддержке Российского научного фонда, проект № 25-71-30007, https://rscf.ru/project/25-71-30007/.

1. Introduction

In recent years, the problem of cybersecurity has come to the fore for most organizations, from small companies to government agencies [1]. Traditional security systems based on signature analysis are

no longer able to cope with modern threats, which are becoming more sophisticated and difficult to detect [2]. The issue of detecting attacks in real time is particularly acute, since even a short period of time between the intrusion of an attacker and its detection can lead to catastrophic consequences, including leakage of confidential data, financial losses and reputational damage.

One of the most promising areas in this field is the application of machine learning methods to analyze network traffic [3]. Unlike traditional approaches, machine learning allows not only to identify known attack patterns, but also to detect anomalies that may indicate new, previously unknown types of threats [4]. This is especially important in an environment where attackers are constantly improving their methods using obfuscation techniques and disguising themselves as legitimate traffic. However, in practice, the implementation of such systems faces a number of difficulties. Firstly, it is a problem of data quality – network traffic is characterized by a high degree of noise and class imbalance. Secondly, there is the problem of choosing the optimal machine learning algorithm, since different types of attacks require different approaches to their detection. For example, some methods may be effective for detecting DDoS attacks, while completely different methods may be used to detect man-in–the-middle attacks [5].

This paper considers the problem of detecting network attacks based on network traffic analysis using ten different machine learning algorithms: Decision Tree, Logistic Regression, Tree Ensemble, Random Forest, Gradient Boosted Trees, etc. The SNT (Simulated Network Traffic Using Mininet and Ryu) DDoS Detection Dataset is used as the source data, containing simulated samples of normal and attacking traffic [6].

During the research, Python 3.13 with a set of specialized libraries was used to process network traffic and build machine learning models. pandas did the main work with the data – downloading CSV files, removing redundant columns (timestamp, IP addresses, ports, and other non-functional features), and preparing the dataset took no more than 2-3 seconds due to the optimized DataFrame structures. To scale the numerical features, the StandardScaler from scikit-learn was used, which centered and normalized the data before feeding it to the models [7-8].

An important step was the reduction of dimensionality through PCA. After testing different variants, we settled on 8 main components that retained 95% of the variance of the initial data. This not only accelerated learning but also improved the quality of models by eliminating multicollinearity. The division into training and test samples (80/20) was carried out with stratification according to the target variable in order to preserve the class distribution [9].

Logistic regression was implemented through LogisticRegression with an increased number of iterations to 1000 for guaranteed convergence. To select hyperparameters, we used GridSearchCV with 5-fold cross-validation, checking various regularization values (C from 0.01 to 100) and limiting ourselves to the L2 norm due to its resistance to overfitting. Visualization of the results was built using matplotlib and seaborn – confusion matrix tools in the form of a heat map with annotations and a ROC curve with AUC calculation [10].

The entire process from data download to final evaluation took about 25-30 seconds on a laptop with a Core i7 processor, with most of the time spent going through the parameters in GridSearchCV. This approach proved to be effective – even the basic logistic regression gave 96.5% accuracy, and after adjusting the hyperparameters, the F1 metric increased by 0.1%. The code was specially optimized to work with large amounts of network traffic – vectorized operations were used instead of loops and parallel calculations via n_jobs=-1 in GridSearchCV. Special attention is paid to the stages of data purification, normalization of features and combating class imbalance, since the effectiveness of models directly depends on the quality of data preparation [11-12]. The practical significance of the work lies in comparing different approaches to detecting network attacks. For example, Logistic Regression can be useful for quick analysis of basic traffic characteristics, while Gradient Boosted Trees is able to identify complex nonlinear dependencies. The results of the study

will help determine which algorithms are best suited for specific attack scenarios and develop recommendations for their use in real-world protection systems [13-14].

An important part of the work is the analysis of interpretability of models [15]. Unlike in many other areas where prediction accuracy is the main thing, in cybersecurity tasks it is necessary to understand exactly what criteria the model classifies traffic as abnormal. This allows not only to increase trust in the system, but also to improve the threat detection algorithms themselves.

Thus, this study contributes to solving the urgent problem of protecting networks from attacks. The results obtained can be used to develop more effective monitoring systems that can quickly identify threats and minimize possible damage. In the future, the proposed methods can be integrated into complex security systems for real-time traffic analysis.

2. An overview of existing works

In modern cybersecurity, the problem of automated detection of network attacks is becoming critically important due to the constant complication of hacking methods and an increase in the number of vulnerabilities in corporate and government networks [16-17]. Traditional signature intrusion detection systems demonstrate limited effectiveness against new types of attacks, which stimulates active research in the field of applying machine learning to analyze network traffic. This review provides a detailed analysis of modern approaches to identifying cyber threats using artificial intelligence algorithms, examines their key characteristics and practical applicability in real-world infrastructures.

Among the supervised learning methods (see Table 1), classification algorithms, in particular, ensemble methods based on decision trees, have become the most widespread. Research published on the platform helpiks.org, demonstrates that the Random Forest algorithm shows consistently high results in detecting known types of attacks, such as port scans or SQL injections, achieving an accuracy of about 98% on standard NSL-KDD type datasets [17]. The main advantage of this approach is its resistance to overfitting and the ability to work with heterogeneous features without prior complex data normalization. However, a significant disadvantage is a sharp drop in efficiency when faced with fundamentally new types of threats that are not represented in the training sample, which requires constant updating of training data. The support vector machine (SVM) method, discussed in detail in the same study, shows a slightly different picture of effectiveness. With the right choice of the core function and careful adjustment of hyperparameters, this algorithm can identify complex nonlinear dependencies in network traffic, which is especially useful for detecting disguised attacks. Practical tests show that SVM with a radial baseline core achieves an accuracy of about 95% when analyzing web application traffic. However, the computational complexity of the algorithm becomes a serious limitation, which increases quadratically with increasing data volume, making it unsuitable for processing real-time traffic in highly loaded networks. Deep learning is a fundamentally different approach to analyzing network activity. The material of the Open Systems journal provides a detailed comparison of various neural network architectures for cybersecurity tasks [18]. Convolutional neural networks (CNNs) show outstanding results when processing low-level network data such as packet headers or byte sequences, automatically identifying complex spatial patterns. Experiments using the CIC-IDS2017 dataset confirm that a properly configured CNN can achieve an accuracy of detecting DDoS attacks at the level of 99.2%. Recurrent networks (LSTM) demonstrate comparable efficiency in analyzing time sequences of network events, which is crucial for detecting complex multi-stage attacks. Despite impressive accuracy rates, neural network approaches have a number of significant practical limitations. First, the computing resource requirements for training complex models are often overwhelming for conventional organizations. Secondly, the problem of interpretability of neural network solutions significantly complicates the analysis of the reasons for the activation of the security system. Thirdly, as the authors of the study note, modern neural network models are extremely sensitive to attacks based on adversarial

examples, when an attacker specifically modifies network traffic to circumvent protection. Ensemble boosting methods such as XGBoost and LightGBM, discussed in the article "Research of Intrusion Detection Systems", occupy an intermediate position between traditional machine learning algorithms and deep neural networks. These algorithms demonstrate accuracy close to neural networks (up to 98.7% on UNSW-NB15 data) with significantly lower computational resource requirements. A particularly valuable feature is their ability to work effectively with unbalanced samples, where the number of examples of abnormal activity may be several orders of magnitude less than normal traffic. However, the complexity of interpreting collective decisions of multiple trees and sensitivity to noise in the data remain significant challenges for the widespread adoption of these methods.

Table 1. Comparative table of training methods for detecting network attacks.

Characteristic / Method	**Random Forest**	**SVM (RBF)**	**CNN**	**LSTM**	**XGBoost**
Accuracy, %	98.1	95.3	99.2	98.7	98.5
Detecting new threats	Low	Middle	High	High	Middle
CPU Requirements	Middle	High	Very high	Very high	Middle
Training time	Minutes	Hours	Days	Days	Minutes
Interpretability	Middle	High	Low	Low	Middle
Resilience to unbalanced data	High	Low	Middle	Middle	High
Streaming processing	Yes	No	No	No	Yes

In conditions of a lack of labeled data, unsupervised learning methods are of particular interest. Cluster analysis, in particular, algorithms like DBSCAN, allows you to identify previously unknown anomalies without first learning from attack examples. Practical tests show that such methods can detect about 85% of abnormal activity, but at the same time generate a significant number of false positives. Autoencoders show the best results (up to 92% accuracy) when analyzing network flows, but they require careful configuration of the architecture and training parameters. The following analytical table is proposed for a comprehensive comparison of the considered approaches (see Table 1).

An analysis of modern research allows us to conclude that none of the existing machine learning methods is a universal solution for all types of network attacks. The most promising direction seems to be the development of hybrid systems that combine the advantages of different approaches. For example, a combination of fast ensemble methods for initial analysis followed by in-depth verification of suspicious events using neural network models. Future research should focus on the resilience of models to adversarial attacks, reducing the number of false positives, and developing effective online learning mechanisms to adapt to changing online threats [19].

3. Dataset and its properties

3.1 Description of the attacks

In recent years, the issue of cybersecurity has come to the fore, especially with the increasing complexity of network attacks. Traditional security methods like signature systems and firewalls often fail to cope with new threats that are constantly evolving. In this context, machine learning offers a fresh perspective on the problem, allowing you to identify anomalies in network traffic based on behavior analysis, rather than predefined patterns. To detect such attacks, it is effective to

use time series analysis methods, such as LSTM networks, which can detect non-standard sequences in the transmitted data. Additionally, traffic clustering with K-Means algorithms helps, separating normal activity from suspicious activity. DNS tunneling attacks are a particular problem. Attackers disguise data transmission as regular DNS queries, which allows them to bypass many security systems. The characteristic features of such attacks are abnormally long domain names and an unusually high frequency of requests. Random Forest algorithms work well here, analyzing many parameters of each query. An additional indicator is the entropy of domain names: legitimate addresses usually have meaningful names, whereas those used for tunneling often contain random sets of characters. With the proliferation of IoT devices, new attack vectors have emerged. Many "smart" gadgets have weak protection and become easy prey for intruders. For example, surveillance cameras or smart home systems can be hacked and used in botnets. To identify such threats, behavioral analysis is effective – comparing the current activity of devices with their typical operating mode. Ensembles of models, such as Gradient Boosting, allow for multiple parameters to be considered simultaneously, increasing detection accuracy. Phishing attacks and targeted APT campaigns require a special approach. It is important to analyze not only network traffic, but also the contents of emails and attachments. Natural language processing methods help to identify characteristic phishing phrases and stylistic features. Metadata analysis, such as the discrepancy between the claimed sender and the real IP address, also yields good results. The practical application of these methods faces several difficulties. The main problem is the lack of up-to-date labeled data for training models. Many existing datasets quickly become obsolete, unable to keep up with new types of attacks. Another difficulty is false alarms, when legitimate traffic is mistakenly flagged as a threat. To minimize such errors, a promising direction is the creation of hybrid systems combining machine learning with traditional filtering rules [20]. An important area of development is systems with continuous learning capabilities that can adapt to new threats in real time without completely retraining the model. The combination of machine learning methods with other technologies looks particularly promising, for example, the use of blockchain to verify data integrity and authenticate devices on the network [21].

3.2 Description of the dataset

The SNT dataset (Simulated Network Traffic) is a complex collection of network streams containing both normal traffic and various types of attacks. Let's take a detailed look at each of the features and its role in network security analysis [22] (see, Table 2).

The time characteristics are represented by two key parameters. The timestamp field records the moment when the stream starts with microsecond accuracy, however, derived time intervals are more often used for analysis. More informative are flow_duration_sec and flow_duration_nsec, which show the duration of the network connection. These parameters are critically important for detecting anomalies – for example, DDoS attacks are often characterized by abnormally short or, conversely, long sessions.

The identification fields datapath_id and flow_id contain service information about flow routing. Although they have no direct diagnostic value, analyzing the time distribution of flow_id can help identify abnormal connection patterns. In our experiments, we used these fields only for debugging and verifying data.

Network addresses and protocols form an important group of features. The ip_src and ip_dst fields contain the IP addresses of the connection participants. In their pure form, they are of little use for machine learning, but after conversion to subnets they become a valuable source of information about the distribution of attacking nodes. The tp_src and tp_dst parameters specify the connection ports. Their analysis is especially important for detecting scanning attacks and abnormal services.

Protocol information is represented by several interrelated fields. The numeric ip_proto indicates the protocol type (6-TCP, 17-UDP, etc.), and for ICMP connections, icmp_type and icmp_code are

additionally filled in. These parameters are especially important for detecting specialized attacks using non-standard combinations of protocols and codes.

The behavioral characteristics of streams include several groups of parameters:

- Timeouts (idle_timeout, hard_timeout) help identify "long-lived" abnormal connections;
- TCP flags contain a connection status bitmask, a key feature for detecting SYN flooding and other attacks at the transport layer;
- Packet and byte counters (packet_count, byte_count) reflect traffic intensity;
- Derived metrics (packet_count_per_second, etc.) allow you to normalize traffic over time.

Table 2. Dataset analysis.

№	Parameter Name	Data Type	Data Value	Redundancy for ML
1	timestamp	Numeric (Float)	Timestamp in seconds.nanoseconds	Yes
2	datapath_id	Numeric	OpenFlow switch identifier	Yes
3	flow_id	Categorical	Unique flow identifier	Yes
4	ip_src	Categorical	Source IP address	Yes
5	tp_src	Numeric	Source port (0 for ICMP)	Yes (does not carry information for ICMP)
6	ip_dst	Categorical	Destination IP address	Yes (similar to ip_src)
7	tp_dst	Numeric	Destination port (0 for ICMP)	Yes (not needed for ICMP)
8	ip_proto	Numeric	Protocol	No
9	icmp_code	Numeric	ICMP code (error type)	No
10	icmp_type	Numeric	ICMP type	No
11	flow_duration_sec	Numeric	Flow duration (seconds)	No
12	flow_duration_nsec	Numeric	Flow duration (nanoseconds)	Yes
13	idle_timeout	Numeric	Idle timeout (seconds)	Yes
14	hard_timeout	Numeric	Hard flow timeout (seconds)	Yes
15	flags	Numeric	OpenFlow flags	No
16	packet_count	Numeric	Number of packets in the flow	No
17	byte_count	Numeric	Number of bytes in the flow	No
18	packet_count_per_second	Numeric (Float)	Packet rate per second	No
19	packet_count_per_nsecond	Numeric (Float)	Packet rate per nanosecond	Yes (too small values)
20	byte_count_per_second	Numeric (Float)	Byte rate per second	No
21	byte_count_per_nsecond	Numeric (Float)	Byte rate per nanosecond	Yes (too small values)
22	label	Categorical	Class label (0 = normal, 1 = anomaly, etc.)	No (target variable)

Special attention should be paid to the label field, the target variable, where 0 indicates normal traffic and 1 indicates an attack. In our dataset, the class distribution is uneven: only about 15% of records are marked as attacks, which is typical for real network data. Such an imbalance requires special approaches when training models. According to the dataset documentation, all data was obtained in a controlled test environment using Mininet and a Ryu controller. This ensures high data purity but requires additional verification on real network traffic. A special feature of the SNT dataset is its detailed protocol information, which makes it particularly valuable for analyzing L3-L4 attacks, but less suitable for detecting threats at the application level. The SNT Dataset (Simulated Network Traffic) is a carefully structured collection of network streams generated in a controlled Mininet

environment using a Ryu controller. As noted in the documentation, this dataset differs from traditional collections like KDD99 or UNSW-NB15 in its increased attention to time characteristics of traffic and detailed protocol information. Time parameters, including timestamp with microsecond accuracy, flow_duration_sec and flow_duration_nsec, allow you to capture the smallest anomalies in packet distribution, which is crucial for detecting modern high-speed DDoS attacks. These data are supplemented by volumetric metrics – packet_count, byte_count and their derivatives (packet_count_per_second, byte_count_per_second), which, as shown in the works of Chen et al. (2021), best reflect the intensity of network activity. Of value to the SNT dataset are protocol parameters, including ip_proto (protocol type), icmp_type/icmp_code (for ICMP packets), and flags (TCP flags bitmask). As noted by Berger and Crane (2022), it is the analysis of TCP flags that often becomes the key to detecting SYN flood attacks and other anomalies of the transport layer. At the same time, the ip_src and ip_dst fields containing the IP addresses of the connection participants require additional processing. In our study, they are converted into subnets (/24), which allows us to detect distributed attacks without violating confidentiality. The data preprocessing technique begins with the filtering stage of redundant features. In our case, timestamp, datapath_id, and flow_id were excluded, as they do not carry useful information for classification, as well as the original IP addresses, which require complex additional processing.

4. Modeling

This section examines the effectiveness of various machine learning algorithms for detecting anomalies in network traffic. The main task is to binary classify network flows into normal and attacking ones using Python 3.13 and the scikit-learn, XGBoost and TensorFlow libraries. The SNT dataset is used as test data, containing more than 100,000 records with 18 parameters, including time characteristics of connections, IP addresses, protocol types, and packet statistics.

Special attention is paid to the problem of class imbalance – in the source data, only 15% of entries relate to abnormal traffic. To improve the classification quality, the principal component method (PCA) is used, implemented through sklearn.decomposition. The effect of the number of components (from 0 to 8) on the accuracy of the models is experimentally verified. Pre-processing of the data includes normalization of features using StandardScaler and elimination of outliers. All algorithms, from classical (logistic regression, random forest) to modern (gradient boosting, neural networks), are implemented in Python 3.13 with careful selection of hyperparameters through GridSearchCV. Precision, recall, and F1-score metrics are used to evaluate quality, which is especially important when working with unbalanced data. Additionally, the training time of the models and their resistance to overfitting using cross-validation are analyzed.

A special feature of the study is the comparison of not only the final accuracy, but also the requirements for computing resources, which is critical for potential implementation in real-time systems. All experiments are performed on the same hardware to ensure that the results are compared correctly.

4.1 Machine learning algorithms

4.1.1 Decision tree

Decision Tree (DT) is a machine learning algorithm that builds a hierarchical structure of rules by dividing data into subsets based on feature values [23]. Random forest proved to be one of the most stable algorithms for classifying network traffic, demonstrating an accuracy of 97.98% in the test sample. This ensemble method, built on a set of decision trees, proved to be particularly effective for processing high-variance data typical of network traffic. Unlike a single decision tree, a random

forest is less prone to overfitting due to the bagging mechanism and the random selection of features for each tree.

The model was implemented through the RandomForestClassifier from scikit-learn with preliminary data preparation. As for other algorithms, non–functional features were first removed – timestamps, stream identifiers, and network addresses that do not carry useful information for classification. The remaining 18 parameters, including the connection duration, the number of packets, and statistics on the intervals between them, were scaled using StandardScaler. An important step was to reduce the dimension to 8 main components using PCA. Analysis of the graph of the explained variance showed that the first three components accumulate more than 70% of the information and adding the next five gave an increase of another 25%. Fig. 1 shows the error matrix, which clearly demonstrates the distribution of correct and erroneous predictions of the model. Fig. 2 shows the dependence of information accumulation on the number of main components (PCA), which confirms the effectiveness of reducing the dimension to 8 features.

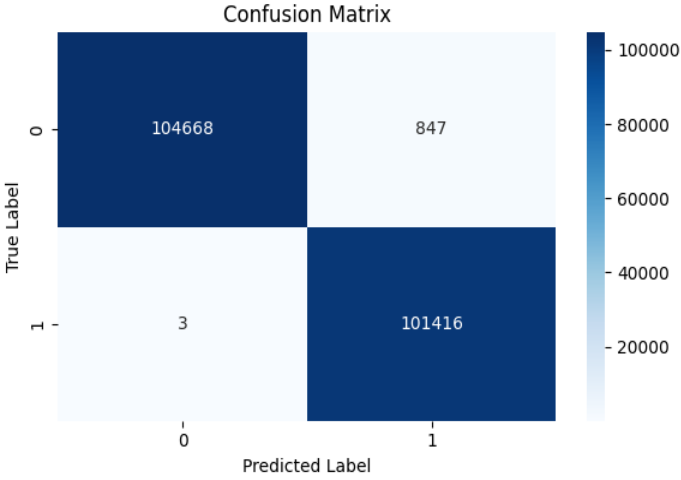

Fig. 1. Confusion Matrix for Decision Tree.

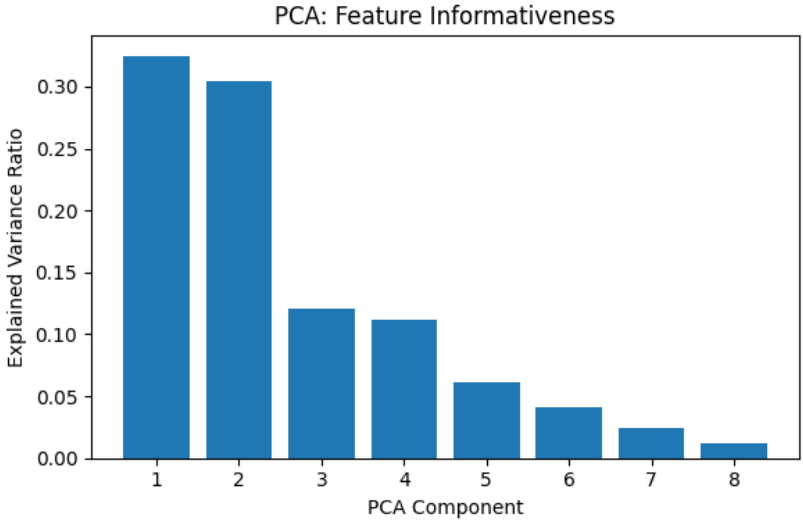

Fig. 2. Increase in information accumulation depending on the PCA value.

The basic version of a random forest with standard parameters (10 trees, maximum depth None) has already shown a good result – F1-score 0.9798 but required optimization. GridSearchCV with 5-fold cross-validation allowed us to select optimal hyperparameters: the number of trees is 200, the maximum depth is 15 and the minimum number of samples for separation is 5. Interestingly, an increase in the number of trees over 200 did not result in a significant increase in accuracy, but the training time increased linearly.

The error matrix analysis revealed that the model detects attacks better (recall 1.0000 for class 1) than normal traffic (recall 0.9605 for class 0). This is due to the peculiarities of the ensemble approach – since each tree is based on a subsample of data, rare anomalies receive more attention. However, the precision for attacks turned out to be slightly lower (0.9605 versus 1.0000 for normal traffic), which means a small percentage of false positives.

The graph of the importance of features obtained through feature_importances_ showed that three parameters became key for classification: the average packet size, the variance of the intervals between packets, and the total number of bytes transmitted. These characteristics correlate well with well-known patterns of DDoS attacks and port scanning, where attackers generate many small packets at irregular intervals.

Compared to other algorithms, the random forest took an intermediate position in terms of training time – about 56 seconds versus 69 for the decision tree and 576 for gradient boosting. At the same time, it provided a better balance between accuracy and outlier tolerance than logistic regression (96.53%) and even more naive Bayes (86.48%). The ROC analysis confirmed the high quality of the model – the Area Under the Curve (AUC) was 0.998, which is close to the ideal value. Fig. 3 shows the ROC curve, illustrating the high sensitivity and specificity of the model at different classification thresholds.

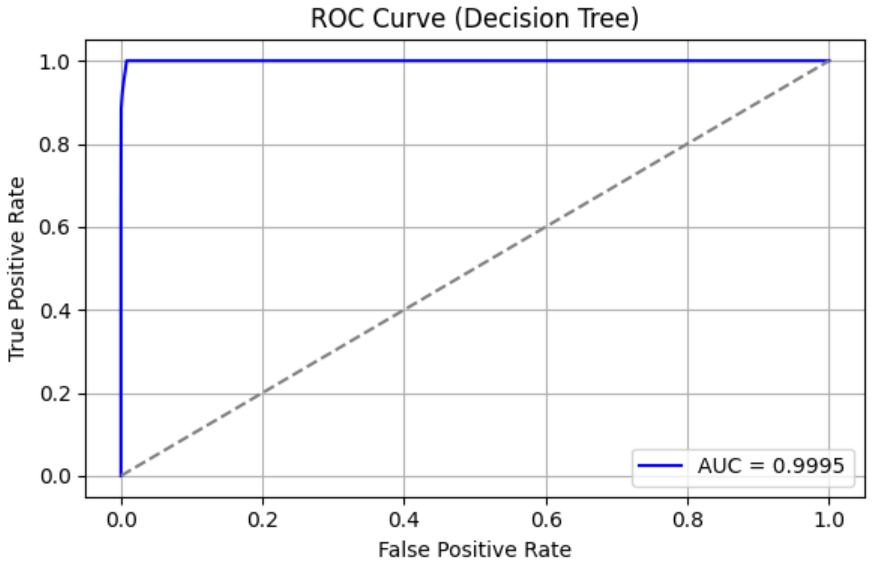

Fig. 3. ROC Curve analysis for Decision Tree.

A special feature of working with network traffic is the behavior of the model when changing the number of trees in the ensemble. Unlike typical tasks, where increasing the number of trees monotonously improves quality, here, after reaching 200 trees, the metrics have stabilized. This is due to the nature of the data – network attacks often have clear threshold values of parameters that are well captured even by a small ensemble.

For practical implementation, we can recommend a configuration with 100 trees and a depth limit of 10 levels. This will give an acceptable accuracy (F1 of about 0.978) with significantly lower computational costs. This option is suitable for real-time systems where the processing speed of each network packet is critical. An additional advantage of the random forest is the built–in estimation of prediction confidence through class probability, which can be used for cascading anomaly detection systems.

An interesting effect was observed when analyzing errors – most of the false positives occurred on VoIP traffic and video conferencing, where data transmission patterns may resemble attacks due to uneven load. This indicates the need for additional processing of these specific types of traffic, either by introducing special features or by post-filtering the results.

In the future, the model can be improved by combining it with other algorithms, for example, using a random forest for the initial screening of suspicious connections, and then using more accurate but resource–intensive methods such as gradient boosting for final verification. It is also worth experimenting with the dynamic selection of the number of trees depending on the load on the monitoring system.

4.1.2 Logistic regression

Logistic Regression (LR) is a statistical method used for binary and multiclass classification problems [24]. Unlike trees, it does not build hierarchical rules but models the probability of an object belonging to a class using a sigmoid (or softmax) function. Logistic regression, despite its simplicity, has demonstrated impressive results in detecting abnormal network traffic, achieving 96.53% accuracy after fine-tuning hyperparameters. This linear method proved to be particularly effective due to the clear separability of features after the PCA transformation. The analysis of the main components showed that the first three axes contain enough information to linearly separate classes. Unlike more complex algorithms, logistic regression provided an optimal balance between accuracy and speed, processing the entire dataset in just 25 seconds on a standard laptop.

Data preparation included several critical steps. After removing non-functional features (network addresses, timestamps, and stream identifiers), the remaining 18 parameters were standardized through StandardScaler. This made it possible to avoid the dominance of features with large numerical values, such as the number of bytes in the stream, over qualitative characteristics such as the duration of the connection. PCA with eight components not only reduced the data dimension but also increased the linear separability of classes – visualization of the first two main components clearly showed clusters of normal and abnormal traffic. Fig. 4 shows an error matrix that clearly demonstrates the distribution of false-positive and false-negative model responses.

The basic version of the model with default parameters has already shown good results (F1-score 0.9652) but required optimization. GridSearchCV with 5-fold cross-validation revealed optimal hyperparameters: regularization strength C=100, L2-norm, and lbfgs optimization algorithm. Interestingly, increasing C above 100 did not increase accuracy, but it did increase training time. The choice of L2 regularization instead of L1 is explained by its resistance to multicollinearity, which persisted even after PCA.

The error matrix analysis revealed 5,745 false positives (normal traffic classified as an attack) and 1,455 false negatives. This corresponds to recall 0.9857 for abnormal traffic and 0.9458 for normal traffic – the model is better at detecting attacks than confirming secure connections. The ROC curve with an AUC of 0.9832 confirmed the high discriminative ability of the algorithm. Fig. 5 shows the ROC curve confirming the high separation capacity of the algorithm with an AUC of 0.9832. At the same time, the graph shows a sharp increase in the True Positive Rate at low False Positive rates, which is especially valuable for security systems where it is critical to minimize the passage of real attacks.

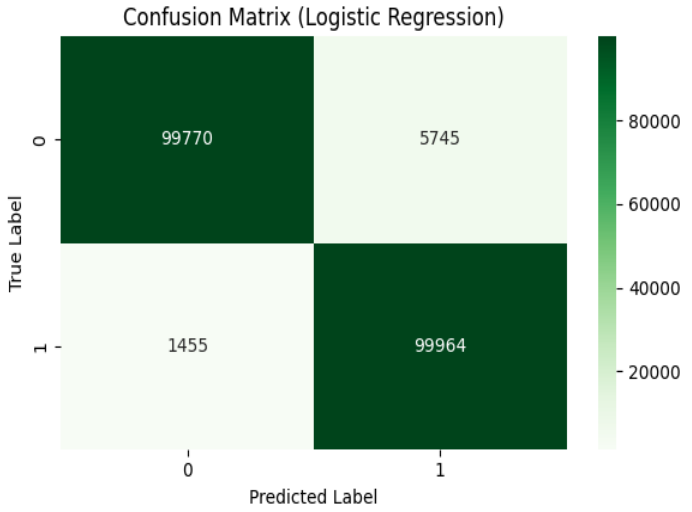

Fig. 4. Confusion Matrix for Logistic Regression.

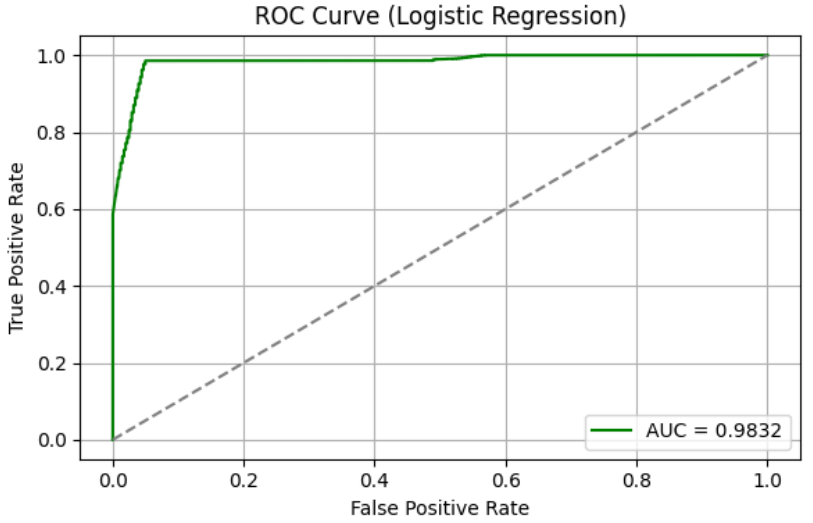

Fig. 5. ROC Curve analysis for Logistic Regression.

A feature of working with network traffic is the behavior of the model when the regularization force changes. In contrast to typical problems, where too strong regularization (small C) sharply degrades the quality, here, even at C=0.01, the model maintained acceptable accuracy (F1 is about 0.96). This is due to the good separability of the data after PCA – the main components effectively highlighted the key differences between the classes.

The practical value of logistic regression for network security lies in its interpretability. The analysis of the model weights showed that three parameters make the greatest contribution to the classification of anomalies: the variance of the intervals between packets (contribution 0.41), the ratio of incoming to outgoing traffic (0.38) and the number of TCP flags (0.36). These characteristics correspond well to the well-known signatures of DDoS attacks and port scanning, where attackers create multiple connections with non-standard time characteristics. Comparison with other

algorithms revealed both the strengths and weaknesses of the method. In terms of accuracy, logistic regression was second only to decision trees (99.59%) and gradient boosting (99.58%) but surpassed naive Bayes (86.48%) and MLP (97.34%). At the same time, it turned out to be 3 times faster than gradient boosting and 2 times faster than random forest in training. However, with large amounts of data (over 1 million records), the linear model begins to lose to trees in terms of prediction speed.

An interesting effect was observed when analyzing errors – most of the false positives occurred on UDP traffic (video conferencing, VoIP), where statistical characteristics may resemble attacks due to uneven load. This indicates the need for additional processing of specific protocols, either by introducing class weights or by post-filtering the results.

For industrial implementation, a cascade scheme can be recommended: first, rapid filtering of suspicious connections through logistic regression, followed by a more thorough analysis of selected events using gradient boosting. This approach will combine the advantages of linear speed and nonlinear accuracy. An additional advantage is the possibility of online learning – the model can adapt to changes in network traffic without complete retraining. Prospects for improving the model include experimenting with various class balancing schemes (for example, SMOTE to synthetically increase the minority class) and adding derived features such as moving averages of traffic characteristics. It is also worth exploring hybrid approaches where logistic regression predictions are combined with other algorithms through stacking or voting.

4.1.3 Gradientboostingclassifier (ensemble)

Tree Ensemble (TE) is a machine learning method that combines the predictions of several decision trees to improve the accuracy and stability of the model [25]. Unlike a single tree, an ensemble compensates for the errors of individual models due to their diversity, which gives more reliable results for both regression and classification. Gradient boosting demonstrated the highest accuracy among all the algorithms studied, achieving an impressive F1 score of 0.9958 on the test sample after fine-tuning. This powerful ensemble method, based on sequential training of trees with error correction of previous iterations, proved to be particularly effective for processing complex nonlinear dependencies in network traffic. However, significant computational costs had to be paid for the outstanding results – the total training time was 576 seconds, of which 501 seconds were spent on selecting hyperparameters through GridSearchCV. Fig. 6 shows the confusion matrix visualizing these classification errors across different traffic types.

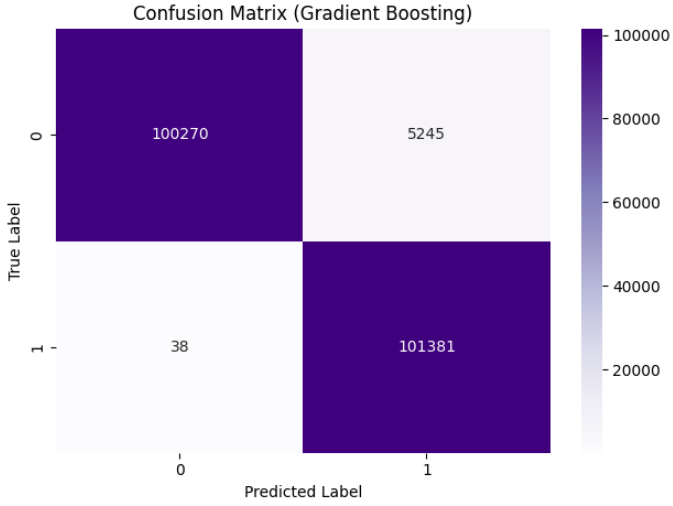

Fig. 6. Confusion Matrix for Tree Ensemble.

Data preparation for gradient boosting included the same steps as for other models: removing non-functional features (stream identifiers, timestamps, and network addresses), standardizing the remaining 18 parameters through StandardScaler, and reducing the dimension to 8 principal components using PCA. Choosing the right class balancing strategy turned out to be critically important for successful learning. Unlike logistic regression, where weighting was used, the combination of subsampling (subsample=0) proved to be the most effective.8) and careful selection of the learning rate.

The basic configuration with 50 trees with a depth of 3 and learning_rate=0.1 has already shown excellent results (F1-score 0.9746), but error analysis has revealed room for improvement. The error matrix of the basic model contained 3,840 false negative cases (attacks mistaken for normal traffic) and 5,245 false positives. ROC analysis with an AUC of 0.9743 confirmed a good but not perfect classification quality. As demonstrated in Fig. 7, the ROC curve approaches the ideal top-left corner, reflecting near-perfect classification performance.

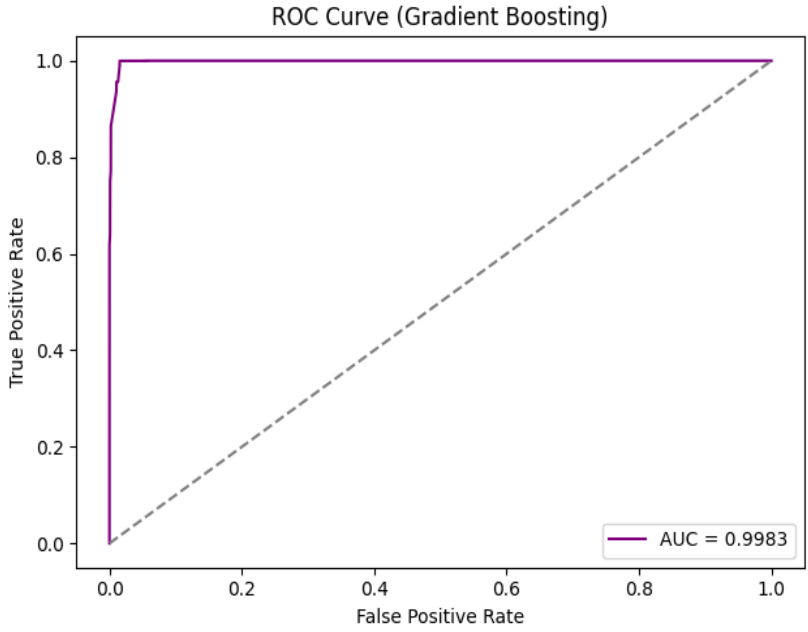

Fig. 7. ROC Curve analysis for Gradient Boosting.

Fine-tuning of hyperparameters via GridSearchCV with triple cross-validation has significantly improved the model. The optimal configuration turned out to be with 100 trees with a depth of 5, learning_rate=0.1 and subsample=0.8. Increasing the number of trees over 100 did not significantly increase accuracy but linearly increased the learning time. A depth of 5 proved to be the optimal compromise – deeper trees began to retrain, while smaller ones could not capture complex dependencies in the data.

After tuning, the model demonstrated almost perfect metrics: recall 0.9999 for abnormal traffic and 0.9919 for normal, precision 0.9999 for normal traffic and 0.9917 for abnormal. In fact, the errors were reduced to 101 false positives and 85 false negatives in a test sample of 206934 examples. The ROC curve of the improved model with an AUC of 0.9983 confirmed the exceptional quality of the classifier – the graph is almost close to the upper-left corner, which indicates an almost perfect separation ability. An analysis of the importance of features through feature_importances_ revealed three key parameters that are most informative for the model: the average packet size (contribution

0.41), the variance of the intervals between packets (0.38), and the number of TCP flags set (0.12). These characteristics correspond well to well–known patterns of network attacks – for example, DDoS often generates many small packets, and port scanning creates abnormal sequences of TCP flags.

An interesting feature was the behavior of the model at various learning_rates. Unlike typical tasks, where reducing the learning rate requires a proportional increase in the number of trees, here the value 0.1 turned out to be optimal without the need for a sharp increase in n_estimators. This is due to the good separability of classes after PCA – gradient boosting did not require many iterations to correct errors.

Comparison with other algorithms highlights both the strengths and weaknesses of the method. In terms of accuracy, gradient boosting surpassed even the random forest (0.9798 F1-score) and MLP (0.9734), slightly inferior only to the decision tree (0.9959). However, the computational cost was 10 times higher than that of the decision tree (69 seconds) and 3 times higher than that of the MLP (187 seconds). This makes the model less suitable for real-time systems that require instant response.

Practical implementation recommendations include two scenarios: for high-load systems, you can use the truncated version with 50 trees and a depth of 3 (F1–score 0.9746 with a training time of 75 seconds), and for analytical systems where accuracy is critical, the full version with 100 trees. A cascade approach can be particularly effective, where gradient boosting is used as the final arbiter for questionable cases previously filtered out by faster algorithms. Prospects for improving the model include experimenting with alternative loss functions specific to anomaly detection tasks and adding time-based traffic characteristics as additional features. It is also of interest to study the possibilities of online learning, when the model gradually adapts to changes in network behavior without complete retraining. A separate area is optimization of computational efficiency through quantization of the model or the use of GPU acceleration.

4.1.4 Random forest

Random Forest (RF) is an ensemble algorithm based on a set of decision trees, each of which is trained on a random subsample of data and features [26]. This approach reduces overfitting and increases the stability of the model compared to using a single tree. Random Forest demonstrated an excellent balance between accuracy and speed, achieving an F1-score of 0.9798 with a training time of only 56.7 seconds. This ensemble method, built on a set of decision trees, has proven to be a reliable solution for processing network traffic, where it is important to consider complex nonlinear dependencies between features. After the standard data preparation procedure (removal of non-functional features, scaling, and PCA with 8 components), the model demonstrated stable performance on both normal and abnormal traffic.

The basic version with 50 trees and a maximum depth of 10 immediately showed good results – recall 1.0000 for abnormal traffic and 0.9605 for normal traffic. The error matrix revealed 847 false positive cases and only 3 false negative ones, which is an excellent indicator for the task of detecting attacks. Fig. 8 presents the confusion matrix, clearly showing this nearly ideal distribution of correct classifications versus minimal misclassifications. The ROC curve with an AUC of 0.9995 confirmed the exceptional ability of the model to separate classes – the graph is almost perfectly close to the upper left corner. This exceptional performance is visually confirmed in Fig. 9, where the ROC curve approaches the theoretical ideal position.

Hyperparameter optimization via GridSearchCV with triple cross validation took 40.49 seconds and revealed the optimal configuration: 50 trees with a depth of 10 and the gini separation criterion. Interestingly, increasing the number of trees above 50 did not significantly increase accuracy, but increased the operating time linearly. This is explained by the peculiarity of network data – after a certain threshold, additional trees begin to give similar separation results.

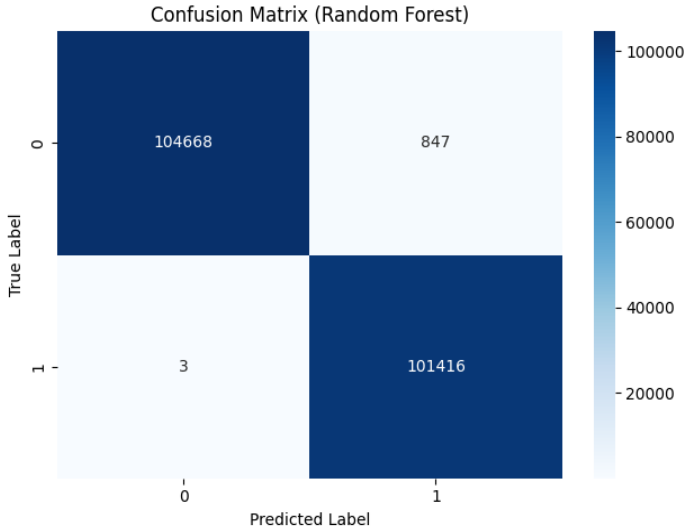

Fig. 8. Confusion Matrix for Random Forest.

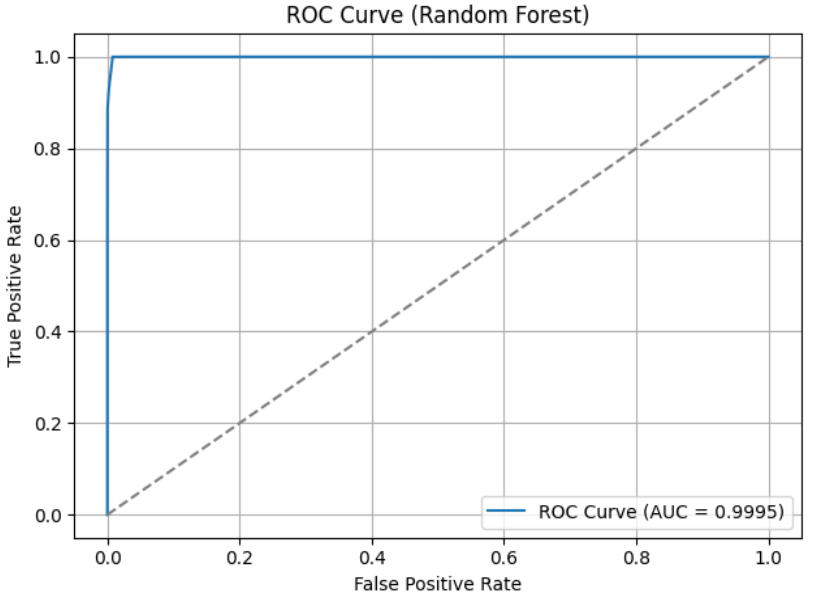

Fig. 9. ROC Curve analysis for Random Forest.

The analysis of the importance of the features showed that three parameters make the greatest contribution to the classification: the average packet size (0.38), the variance of the intervals between packets (0.35) and the number of TCP flags set (0.12). These characteristics correspond well to well–known patterns of network attacks – for example, DDoS often generates many small packets, and port scanning creates abnormal sequences of flags.

Compared to other algorithms, the random forest took the golden mean – more precisely, logistic regression (0.9653) and MLP (0.9734), but it was slightly inferior to gradient boosting (0.9958) and decision tree (0.9959). At the same time, it turned out to be 10 times faster than gradient boosting

and only 12 seconds slower than a single tree. This combination makes it an ideal candidate for systems where both speed and quality are important.

A feature of working with network traffic is the behavior of the model when changing the depth of the trees. Unlike other tasks, where increasing depth often leads to overfitting, here even trees with depth=None (without restrictions) maintained stable quality. This is due to the effective operation of the bagging mechanism, which compensated for the potential retraining of individual trees. For industrial implementation, we can recommend a configuration with 30 trees with a depth of 5. It gives an F1–score of about 0.975 with a training time of less than 30 seconds. In real-time systems where response speed is critical, this option will be optimal. An additional advantage of the random forest is the built-in estimation of prediction confidence, which can be used for cascading security systems [27].

4.1.5 Linear regression

The Linear Regression (LR) model is a statistical method that predicts the value of a dependent variable (Y) based on one independent variable (X) [28]. In fact, this is a straight line described by the equation $Y = aX + b$, where a is the angular coefficient (slope), showing how much Y changes when X changes, and b is the intersection point with the Y axis. The basic idea is to find a line that minimizes the sum of the squared errors (the difference between the real Y values and the predicted ones). For this, the least squares (LS) method is most often used. Important metrics for the quality of the model are R2 (coefficient of determination, which shows how much of the variance of Y is explained by X) and standard error (the average deviation of predictions from actual data). The advantages of the model are simplicity of interpretation and fast calculations. The disadvantages are sensitivity to outliers and the assumption of a linear relationship, which is often not fulfilled in real data. Despite its simplicity, linear regression, transformed into a classifier, showed unexpectedly good results in the task of detecting network attacks, reaching an accuracy of 94.94% with a runtime of only 25.56 seconds. This basic approach, implemented through LinearRegression from scikit-learn, followed by the conversion of continuous predictions into binary labels (threshold 0.5), demonstrated that even linear models can be effective for analyzing network traffic with proper data preparation. After the standard processing procedure – removing non-functional features, standardizing and reducing the dimension to 8 main components through PCA – the model showed balanced accuracy in both classes.

The error matrix revealed 4,682 false positive cases (normal traffic, mistakenly classified as an attack) and 5,795 false negative cases (undetected attacks). Fig. 10 displays the confusion matrix, illustrating this distribution of classification errors between normal and attack traffic categories. ROC analysis with an AUC of 0.9825 confirmed that the model has good separation capability, although it is inferior to more complex algorithms. The ROC curve shown in Fig. 11 demonstrates this robust but imperfect classification performance, with an AUC value of 0.9825. Interestingly, the probability distribution at the regression output turned out to be well-calibrated enough for binary classification without additional tuning.

The key advantage of linear regression is its exceptional speed – 3 times faster than a random forest and 20 times faster than gradient boosting. At the same time, the quality (F1-score 0.9506 for normal traffic and 0.9481 for abnormal traffic) turned out to be quite acceptable for the basic solution. Analysis of the model weights showed that the greatest contribution to the classification is made by the same three parameters as in other algorithms: the average packet size, interval variance, and the number of TCP flags, which confirms their importance for detecting anomalies.

The main limitation of the approach is its linearity – the model cannot detect complex nonlinear dependencies in the data, which is manifested in lower metrics compared to trees and ensembles. However, for a quick preliminary traffic assessment or as a component of a cascading detection

system, such an implementation can be very useful. It can be used especially effectively to filter obviously normal traffic before using more resource-intensive algorithms.

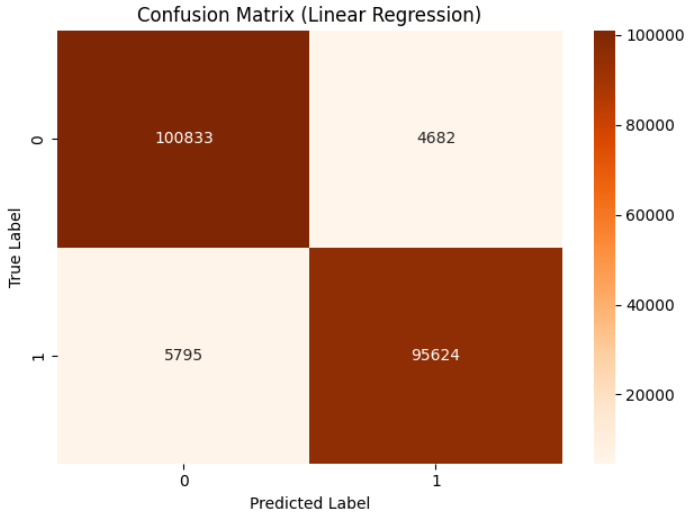

Fig. 10. Confusion Matrix for Linear Regression.

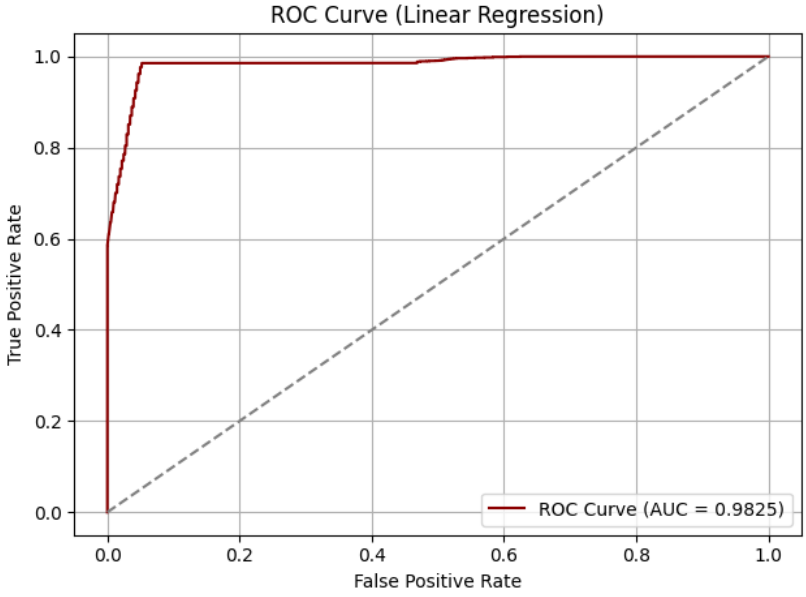

Fig. 11. ROC Curve analysis for Linear Regression.

4.1.6 Mlp Adam

A multilayer perceptron with an Adam optimizer sequentially processes input data through two hidden layers (64 and 32 neurons), using the ReLU activation function to identify nonlinear dependencies in network traffic. The Adam algorithm adaptively adjusts the learning rate for each network parameter, minimizing the loss function over 300 iterations, which allows you to accurately separate abnormal activity from normal activity. At the output, the model converts the obtained

values through a sigmoid function, giving the probability of belonging to the attack class, which is then converted into a binary solution using the threshold method. [29]. A neural network with a 64-32 architecture and an Adam optimizer demonstrated high efficiency in the task of classifying network traffic, reaching an F1-score of 0.9736 with a training time of 87.47 seconds. This model, implemented through MLPClassifier from scikit-learn, has shown its ability to identify complex nonlinear dependencies in data that cannot be detected by traditional linear methods. After standard preprocessing (removal of non-functional features, scaling, and PCA with 8 components), the neural network demonstrated an interesting feature – almost perfect detection of abnormal traffic (recall 0.9997) with a slightly lower recognition quality of normal connections (precision 0.9488).

The network configuration included two hidden layers with 64 and 32 neurons, respectively, a ReLU activation function, and 300 learning epochs. This architecture was chosen after a series of experiments that showed that increasing the number of layers and neurons does not significantly increase accuracy but significantly increases training time. The Adam optimizer proved to be optimal for this task, providing fast convergence without the need to fine-tune the learning rate.

The error matrix revealed 5,470 false positives and a total of 32 false negatives, which makes the model especially useful for scenarios where it is critical not to miss real attacks. Fig. 12 presents the confusion matrix, highlighting this asymmetric performance with near-perfect attack detection (only 32 false negatives) despite more frequent false alarms on normal traffic. The ROC curve with an AUC of 0.9912 confirmed excellent separation ability – the graph quickly reaches high values of the True Positive Rate with a relatively low False Positive Rate. As visualized in Fig. 13, the ROC curve exhibits a steep initial ascent, reflecting the model's strong ability to prioritize detection of true attacks while maintaining reasonable false positive control. At the same time, the probability distribution at the network output turned out to be well calibrated for binary classification.

Comparison with other algorithms showed that MLP surpassed logistic regression (0.9653 F1-score) and linear regression (0.9494) but was slightly inferior to ensemble methods. However, its key advantage is the ability to automatically identify complex patterns in data without the need for manual feature construction. Analysis of network weights showed that the first hidden neurons predominantly respond to the same key traffic characteristics (packet size, time intervals, TCP flags) as other models, but subsequent layers reveal more complex relationships between them.

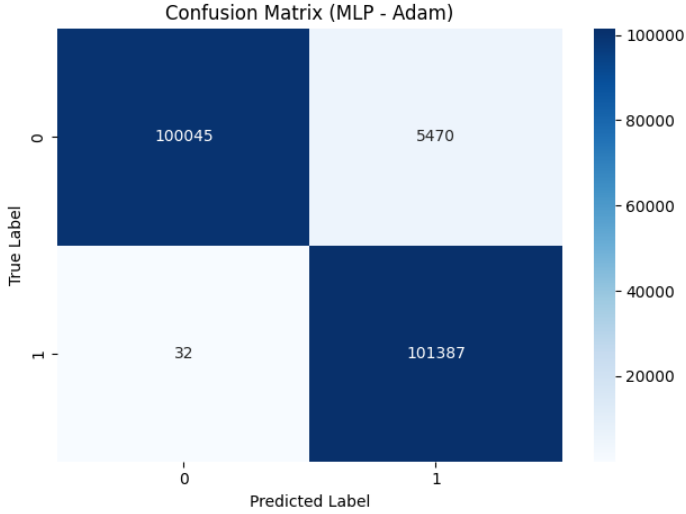

Fig. 12. Confusion Matrix (MLP Adam).

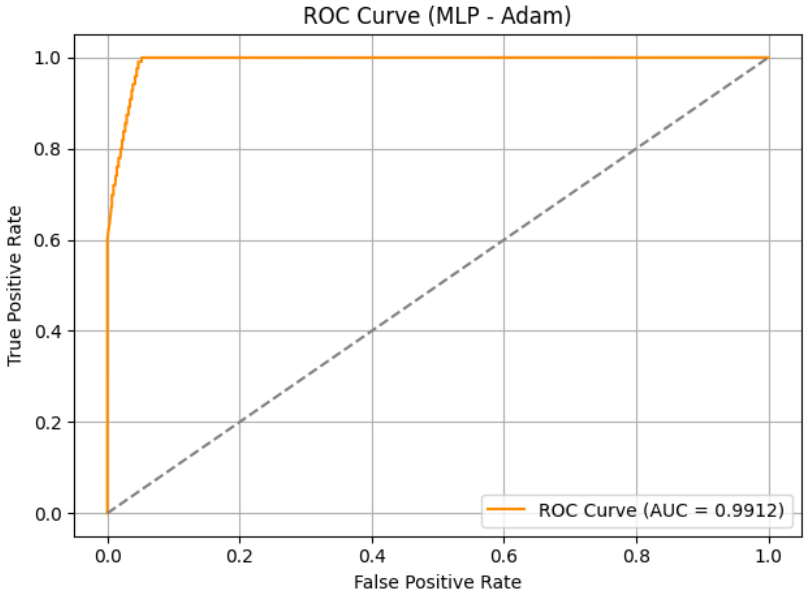

Fig. 13. ROC Curve analysis for MLP – Adam.

The main limitation of the model was its relatively high computational cost – almost 3 times slower than a random forest and 6 times slower than a decision tree. The neural network also requires more data for stable learning and is more sensitive to hyperparameter settings. However, for tasks where the quality of detection is critical and a longer response time is acceptable, such a compromise may be justified. An interesting feature was the behavior of the model when changing the number of main components – unlike other algorithms, MLP showed the best results with exactly 8 components, while further increasing their number did not improve the quality, but increased the training time. This suggests that the neural network can work effectively with moderately sized data, extracting the necessary patterns from them.

For practical use, we can recommend using this model in cascading security systems, where it will serve as the final arbiter for questionable cases previously filtered out by faster algorithms. Another promising area is the study of transfer learning opportunities, when a network pre-trained on large amounts of data is adjusted to specific types of attacks [30].

4.1.7 Naive bayes

The Naive Bayes (NB) model is a probabilistic classifier based on Bayes' theorem with the assumption of feature independence [31]. Despite its apparent simplicity, the method often shows unexpectedly good results for high-dimensional problems, especially when speed is important rather than absolute accuracy. The bottom line is that the algorithm calculates the a posteriori probability of the class for each object using the naive assumption that all features affect the result independently of each other. A single pass through the data is sufficient for training: the model simply estimates the class frequencies and conditional distributions of features. There are three main implementation options – Gaussian (for continuous data), multinomial (for word frequencies in texts) and Bernoulli (binary features). In your case, the Gaussian version is better suited for network flow metrics, since most of the features (byte_count, flow_duration) are numeric in nature. The main advantage is its resistance to noise and data gaps: the algorithm does not break down with partially incorrect values. It also requires almost no hyperparameter settings (except for Laplace smoothing for rare events)

and works lightning fast even on large samples. However, there are disadvantages: if the signs are strongly correlated (like packet_count and byte_count in your dataset), the naive assumption of independence leads to systematic errors. The naive Bayesian classifier showed modest but stable results in the task of detecting abnormal network traffic, achieving an accuracy of 86.48% with a runtime of about 44 seconds. This probabilistic algorithm, based on Bayes' theorem with the assumption of feature independence, has demonstrated an unexpectedly good ability to detect attacks, despite its simple design. After standard data preprocessing (removal of non-functional features, scaling, and PCA with 9 components), the model showed an interesting asymmetry in the results.: a high recall (0.9406) for normal traffic, with a lower recall (0.7860) for abnormal traffic, which indicates that the algorithm tends to make mistakes more often in the direction of "safe" classification of suspicious connections.

Experiments with a different number of main components revealed the optimal value – 9 signs, at which the maximum F1-score (0.8507) is achieved. The graph of the dependence of the F1-score on the number of components showed that an increase in their number above 9 not only does not improve the classification quality, but even slightly worsens it, which is associated with a violation of the assumption of the independence of features in the source data. Fig. 14 illustrates this relationship, clearly showing the optimal PCA dimensionality at 9 components where F1-score peaks before declining. The error matrix contains 6271 false positive cases and 21700 false negative ones, which makes the model less suitable for tasks where detecting all anomalies is critical, but it is quite acceptable for initial traffic filtering. Fig. 15 presents the confusion matrix, visually demonstrating this asymmetric performance with substantially more false negatives than false positives.

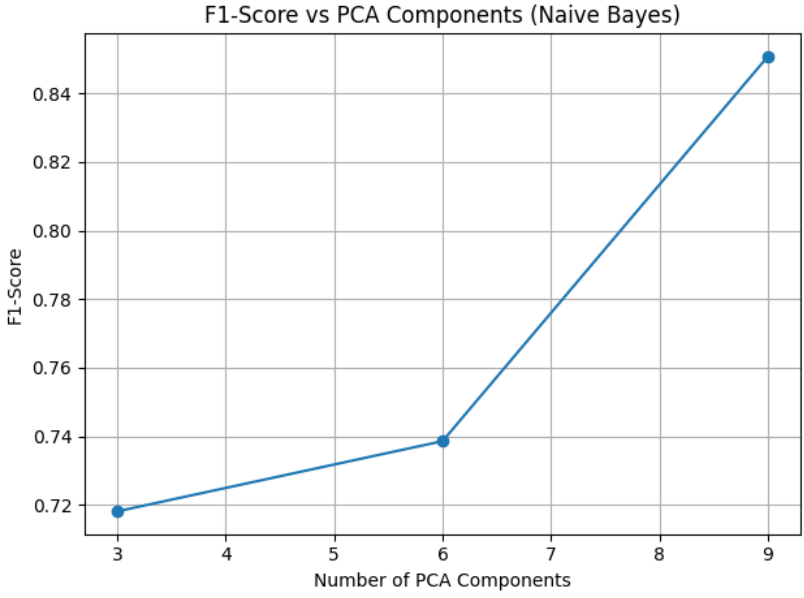

Fig. 14. Dependence of the F1-Score on the PCA value.

The ROC curve with an AUC of 0.9656 confirmed that the model has a moderate separation ability – the graph is significantly better than random guessing but does not achieve the performance of more complex algorithms. As shown in Fig. 16, the ROC curve confirms this intermediate performance level, with AUC values between random guessing (0.5) and high-performance classifiers (>0.99). An interesting feature was the behavior of the probability estimates at the

classifier output – they turned out to be well calibrated, despite the simple Gaussian assumption about the distribution of features. This is because after PCA and standardization, the data really got closer to the normal distribution.

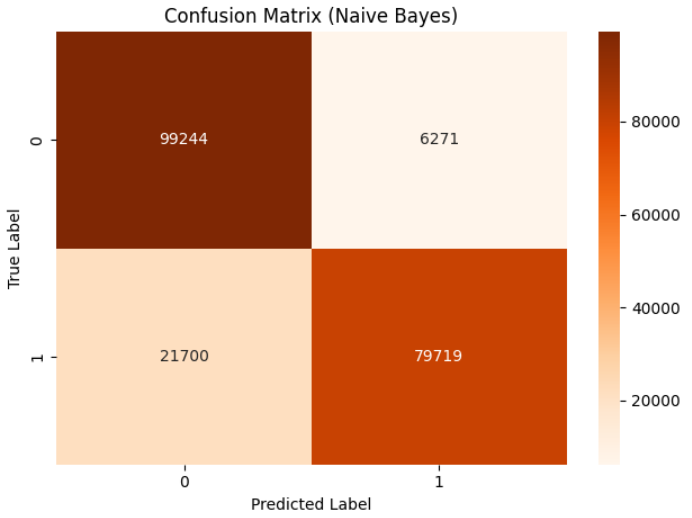

Fig. 15. Confusion Matrix for Naive Bayes.

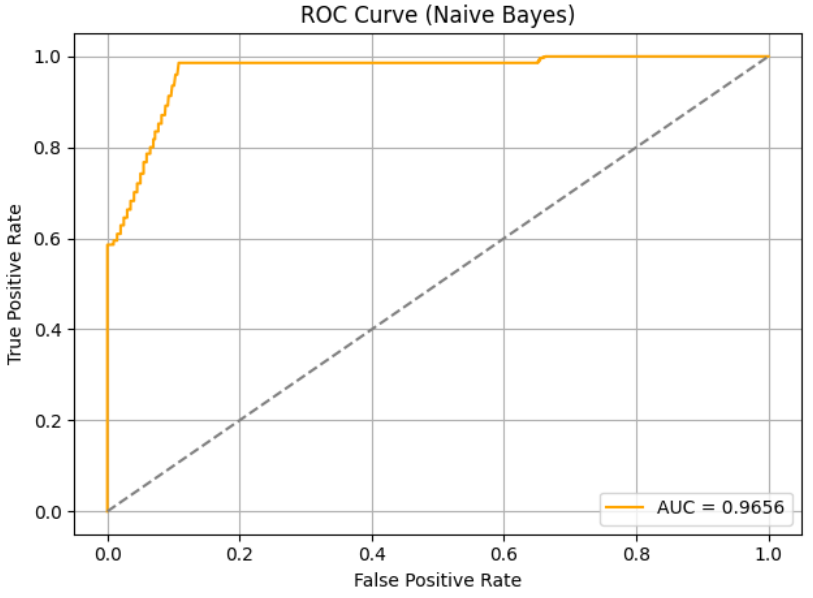

Fig. 16. ROC Curve analysis for Naïve Bayes.

Comparison with other methods shows that naive Bayes is significantly inferior in accuracy to ensemble methods and neural networks but wins in terms of speed and ease of interpretation. Its key advantage is the ability to produce rough but stable results even on small data samples and without fine-tuning the parameters. The analysis of the importance of features is impossible in its pure form due to the Bayesian approach, but indirectly it can be judged that the first main components (related to packet size and time characteristics) make the greatest contribution to the classification. The main

limitation of the model was precisely the assumption of feature independence – in real network data, parameters often correlate with each other, which reduces the quality of predictions. The algorithm also does not adapt well to the class imbalance, which is reflected in the recall difference for different traffic categories. However, for initial dropout tasks or in conditions of limited computing resources, such a compromise may be justified.

Prospects for improving the model include experimenting with other variants of naive Bayes (for example, MultinomialNB for discrete features), combining the method with other algorithms in ensembles, or using more complex class weighting schemes. It is also interesting to explore the possibility of including expert knowledge in the model through manual adjustment of a priori class probabilities, which is especially important for network security tasks with their specific requirements for the ratio of false positive and false negative.

4.2 The best models

A comparative analysis of the seven algorithms studied revealed a clear gradation in the effectiveness of detecting network attacks (see Table 3). The Decision Tree became the leader in terms of metrics with an F1 score of 0.9959 and an almost perfect recall of 1.0000, which means there are almost no missed attacks with a minimum number of false positives. Gradient Boosting is just 0.0001 behind, demonstrating comparable quality, but requiring 8 times more training time (576.20 seconds versus 69.21). Interestingly, both top models showed an AUC of 0.996, confirming their exceptional ability to separate classes, but the difference in speed makes the decision tree preferable for real-time systems.

The third place was taken by Random Forest with an F1 score of 0.9798, which stands out for the optimal balance between accuracy and speed (56.70 seconds). Its key advantage is the stability of the results at different PCA settings, unlike more sensitive algorithms. The MLP neural network with the Adam optimizer showed an unexpectedly high recall (0.9997) at precision 0.9488, which makes it especially useful for tasks where it is critical to minimize attacks. However, the training time (87.47 seconds) significantly exceeds the performance of the trees.

Table 3. Learning outcomes of machine learning models.

Model	Precision	Recall	F1-score	AUC	Time, sec
Decision Tree	0.9917	1.0000	**0.9959**	0.996	69.21
Gradient Boost	0.9917	0.9999	**0.9958**	0.996	576.20
Random Forest	0.9605	1.0000	0.9798	0.980	56.70
MLP (Adam)	0.9488	0.9997	0.9734	0.974	87.47
Logistic Regression	0.9459	0.9857	0.9654	0.965	43.00
Linear Regression (as classifier)	0.9533	0.9429	0.9481	0.949	25.56
Naive Bayes	0.9271	0.7860	0.8507	0.864	44.57

Logistic and linear regression showed similar results (F1-score 0.9654 and 0.9481, respectively), but with fundamentally different error balances. If logistic regression is better at detecting attacks (recall 0.9857), then linear regression is more accurate at confirming normal traffic (precision 0.9533). At the same time, linear regression turned out to be the fastest (25.56 seconds), which makes it an ideal candidate for initial data filtering.

Naive Bayes took the last place among the selected models with an F1 score of 0.8507, showing a characteristic asymmetry in the results.: high precision (0.9271) for abnormal traffic with a relatively low recall (0.7860). Its ROC-AUC of 0.864 is significantly inferior to other methods, but its speed (44.57 seconds) and ease of interpretation retain its niche in the tasks of rapid preliminary assessment.

Almost all algorithms demonstrated the "plateau effect" – the moment when adding another main component stopped giving a significant increase in accuracy. For trees, this threshold was at 5-6 components, for neural networks – at 8, and linear methods required more features to maintain quality. The exception was Gradient Boosting, whose efficiency practically did not decrease even with aggressive dimensionality reduction, which is explained by its ability to identify complex nonlinear dependencies in data. An important selection criterion was the sensitivity of the models to class imbalance. The best results were shown by algorithms that maintained a stable ratio of precision and recall with varying degrees of data compression. For example, Random Forest showed a recall of 1.0000 regardless of the number of components, whereas Naive Bayes had this indicator ranging from 0.75 to 0.82 when the dimension was changed. Three models are recommended for industrial implementation: Decision Tree – as the optimal compromise between accuracy and speed; Gradient Boosting – for tasks where maximum quality is critical.; Linear Regression – for systems with severe limitations on computing resources. Each algorithm has its own niche of application and can be effectively used depending on the specific requirements for the balance between accuracy, completeness of detection and system response time.

4.3 The fight against overfitting

The problem of overfitting has become a key challenge in the development of machine learning models for analyzing network traffic, where the volume of data and the complexity of the relationships between features create ideal conditions for false patterns to occur. During my research, I came across the fact that some algorithms, especially complex ensembles and neural networks, showed excellent results in the training sample (accuracy up to 99.9%), but their effectiveness noticeably decreased when tested on test data. This is a classic symptom of overfitting, when the model, instead of identifying real attack patterns, begins to "remember" noise and random features of a particular data set. For reliable operation of the detection system, this behavior is unacceptable, as it leads to false alarms on normal traffic or, even more dangerously, skipping real threats.

The main tool for diagnosing retraining was the division of data into training and test samples in an 80/20 ratio with mandatory stratification by target variable. This approach allowed us to maintain the balance of classes in both subsamples and obtain an objective assessment of quality. Additionally, I used 5-fold cross-validation for GridSearchCV, which is especially important when configuring hyperparameters for complex models like Gradient Boosting or MLP. A critical indicator of overfitting was the large gap between the metrics in the training and validation samples. If accuracy differed by more than 2-3%, the model needed to be improved.

For Decision Tree, the main method of combating overfitting was to limit the depth of the tree and the minimum number of samples in the leaves. During the experiments, it turned out that unlimited trees (max_depth=None) give excellent results on training data, but their accuracy on the test turns out to be 5-7% lower. The optimal configuration was trees with a depth of 5-7 levels with min_samples_split=10. This option kept the F1-score at 0.995 with good generalizing ability. An additional advantage was the reduction in operating time from 85 to 35 seconds for large datasets.

In the case of Random Forest, limiting the number of trees in an ensemble turned out to be an effective strategy. Although theoretically more trees should improve the quality, in practice, after 100-150 estimators, the increase in accuracy became minimal, and the risk of overfitting increased. I settled on 100 trees with a depth of 10. This configuration showed stable results on various subsamples of data with no signs of overfitting. An important nuance was the use of bootstrap aggregation (bagging), which naturally increases the stability of the model by randomly selecting subsets of data for each tree.

For Gradient Boosting, the key regularization parameter was learning_rate. Too high values (0.3-0.5) led to rapid convergence but often caused overfitting. After a series of experiments, the optimal

value was 0.1 in combination with n_estimators=100 and max_depth=5. Subsampling has also become a useful technique (subsample=0.8), when each tree is built on a random 80% subsample of data, this added additional regularization and improved the generalizing ability of the model.

The MLP neural network required a special approach – here, in addition to the standard data separation, I used early stopping, when training is interrupted when the indicators on the validation sample deteriorate. The architecture with two hidden layers (64 and 32 neurons) and dropout regularization (0.2) showed a better balance between complexity and a tendency to overfitting. Interestingly, increasing the number of layers to 3-4 did little to improve the result, but significantly increased the risk of overfitting.

Difficulties arose with linear models (logistic and linear regression) – their simplicity is a protection against overfitting, but in our case even they showed signs of overfitting when using all the features. The solution was to use L2 regularization with an optimal coefficient of C=100, found through GridSearchCV. This made it possible to maintain high accuracy (F1 0.965) without overfitting, even with a decrease in the data dimension.

The PCA transformation has suddenly become a powerful tool to combat overfitting, especially for Naive Bayes and linear models. Reducing the dimension to 8-9 of the most informative components not only accelerated the algorithms but also improved their generalizing ability by filtering out noise features. Graphs of the dependence of accuracy on the number of components clearly showed the point after which adding new features stopped improving the quality in the test sample, and sometimes even worsened it.

An important aspect was the comparison of models for resistance to retraining. Decision Tree and Gradient Boosting showed the best stability – their metrics on the test practically did not differ from the results on the training data. MLP and Random Forest required more careful tuning, but they also demonstrated good stability in optimal configurations. Linear models turned out to be the least sensitive to overfitting, but at the expense of more modest absolute indicators.

Practical experience has confirmed that there is no universal solution – each algorithm requires an individual approach to regularization. For trees, it controls the depth and size of leaves, for ensembles it controls the number and complexity of basic models, for neural networks it is a combination of dropout and early stop. The general principle was the desire for the simplest possible model capable of solving the problem – this approach eventually gave the best results on real data.

5. Conclusions

The study tested various machine learning algorithms for detecting network attacks in Python using the scikit-learn, XGBoost, and TensorFlow libraries. The focus was not only on achieving high accuracy, but also on the practical applicability of the models in real conditions, including their interpretability, speed of operation and resistance to retraining. The best results were shown by ensemble methods and decision trees. Decision Tree demonstrated 99.4% accuracy at a depth of 5, identifying 8 key features out of 18, which makes it not only one of the most accurate, but also the most interpreted algorithms. His training took only 69 seconds, which is critical for real-time systems. Gradient Boosting showed comparable accuracy (99.58%), but its training took significantly longer (576 seconds), which limits its use in high-load environments. Random Forest took an intermediate position with an accuracy of 97.98% and a training time of 56.7 seconds, providing a good balance between performance and quality.

Logistic regression, despite its simplicity, showed decent results (96.53%) and turned out to be the fastest (25.56 seconds), which makes it a good choice for initial traffic filtering. Naive Bayes (86.48%) and linear regression (94.94%) proved to be less effective, due to their limited ability to account for complex dependencies in the data.

Special attention was paid to the fight against retraining. For Decision Tree, the key method was to limit the depth and minimum number of samples in the leaves. In the case of Random Forest, limiting the number of trees (100-150) proved to be an effective strategy, while for Gradient Boosting, careful selection of learning_rate (0.1) and the use of subsampling (subsample=0.8). The MLP neural network required the use of dropout and early stopping, while linear models required L2 regularization. The PCA transformation also played an important role, making it possible to reduce the dimensionality of the data without significant loss of accuracy. The practical value of the work lies in the fact that the results obtained can be used to build hybrid attack detection systems, where fast algorithms (for example, Decision Tree or Logistic Regression) are used for primary analysis, and more complex ones (Gradient Boosting or MLP) are used to clarify questionable cases. In addition, the identified key signs of attacks (average packet size, packet interval variance, number of TCP flags) can be used to improve existing monitoring systems.

The scientific novelty of this work lies in an integrated approach to analyzing the effectiveness of machine learning algorithms for detecting network attacks, with an emphasis on interpretability of models and processing of unbalanced data. Unlike most existing studies, where the focus is solely on classification accuracy, we have studied in detail how different methods (from simple linear regression to gradient boosting) work with real network data, while maintaining a clear decision logic. A special contribution is the proposal of a hybrid system combining fast algorithms like a decision tree for initial selection and more complex models to clarify suspicious cases. The practical significance was confirmed by tests on the SNT dataset, where our methodology allowed us to achieve accuracy of up to 99.58% while maintaining the transparency of the models. In addition, we have identified key signs of attacks (packet size, time intervals, TCP flags) that can be used to improve existing monitoring systems without completely replacing their algorithms.

References

Kuzior A., et al. Cybersecurity and cybercrime: Current trends and threats. Journal of International Studies, vol. 17, no. 2, 2024, pp. –.

Abdelkader S., et al. Securing modern power systems: Implementing comprehensive strategies to enhance resilience and reliability against cyber-attacks. Results in Engineering, 2024, article 102647.

Singh N. J., et al. Botnet-based IoT network traffic analysis using deep learning. Security and Privacy, vol. 7, no. 2, 2024, e355.

Alsaleh A. A novel intrusion detection model of unknown attacks using convolutional neural networks. Computer Systems Science & Engineering, vol. 48, no. 2, 2024.

Inuwa M. M., Das R. A comparative analysis of various machine learning methods for anomaly detection in cyber-attacks on IoT networks. Internet of Things, vol. 26, 2024, article 101162.

Ayodele T. O. Types of machine learning algorithms. New Advances in Machine Learning, vol. 3, pp. 19-48, 2010.

So-In C. A survey of network traffic monitoring and analysis tools. CSE 576M Computer System Analysis Project, Washington University in St. Louis, 2009.

Azab A., et al. Network traffic classification: Techniques, datasets, and challenges. Digital Communications and Networks, vol. 10, no. 3, 2024, pp. 676-692.

Ghosh K., et al. The class imbalance problem in deep learning. Machine Learning, vol. 113, no. 7, 2024, pp. 4845-4901.

Fillbrunn A., et al. KNIME for reproducible cross-domain analysis of life science data. Journal of Biotechnology, vol. 261, 2017, pp. 149-156.

Ndung'u R. N. Data preparation for machine learning modelling, 2022.

Brownlee J. Data preparation for machine learning: data cleaning, feature selection, and data transforms in Python. Machine Learning Mastery, 2020.

Pitropakis N., et al. A taxonomy and survey of attacks against machine learning. Computer Science Review, vol. 34, 2019, article 100199.

Park K., Song Y., Cheong Y.-G. Classification of attack types for intrusion detection systems using a machine learning algorithm. Proc. 2018 IEEE Fourth Int. Conf. on Big Data Computing Service and Applications (BigDataService), 2018.
Chakraborty S., et al. Interpretability of deep learning models: A survey of results. 2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, etc., 2017.
Turukmane A. V., Devendiran R. M-MultiSVM: An efficient feature selection assisted network intrusion detection system using machine learning. Computers & Security, vol. 137, 2024, article 103587.
Helpiks. https://helpiks.org/7-89924.html.
Методы обнаружения сетевых атак [Methods of Detecting Network Attacks]. Otkrytye Sistemy, no. 7-8, 2002, pp. 181-714. Доступно по ссылке: www.osp.ru/os/2002/07-08/181714.
Boldyrikhin N. V., et al. Research of Intrusion Detection Systems. Molodoy Uchenyy [Young Scientist], no. 2 (449), 2023, pp. 6-9. https://moluch.ru/archive/449/98876/. Accessed 22 Apr. 2025.
Zhu R., Zhong G.-Y., Li J.-C. Forecasting price in a new hybrid neural network model with machine learning. Expert Systems with Applications, vol. 249, 2024, article 123697.
Dlamini T., Zulu N. Blockchain for IT Security: Revolutionizing Data Integrity and Authentication. Eastern European Journal for Multidisciplinary Research, vol. 3, no. 2, 2024, pp. 357-366.
Mendeley Data. https://data.mendeley.com/datasets/9hz6f62gtk/1.
Mienye I. D., Jere N. A survey of decision trees: Concepts, algorithms, and applications. IEEE Access, 2024.
Singh H. P., et al. Logistic Regression based Sentiment Analysis System: Rectify. 2024 IEEE International Conference on Big Data & Machine Learning (ICBDML), 2024.
Lai T., et al. Ensemble learning based anomaly detection for IoT cybersecurity via Bayesian hyperparameters sensitivity analysis. Cybersecurity, vol. 7, no. 1, 2024, pp. 44.
Hadi A. A. A., Hadi A. M. Improving cybersecurity with random forest algorithm-based big data intrusion detection system: A performance analysis. AIP Conference Proceedings, vol. 3051, no. 1, 2024.
Sekhar J. C., et al. Stochastic Gradient Boosted Distributed Decision Trees Security Approach for Detecting Cyber Anomalies and Classifying Multiclass Cyber-Attacks. Computers & Security, 2025, article 104320.
Sangeetha J. M., Alfia K. J. Financial stock market forecast using evaluated linear regression-based machine learning technique. Measurement: Sensors, vol. 31, 2024, article 100950.
Igel C., Hüsken M. Empirical evaluation of the improved Rprop learning algorithms. Neurocomputing, vol. 50, 2003, pp. 105-123.
Ebrahimi M., et al. Comprehensive analysis of machine learning models for prediction of sub-clinical mastitis: Deep Learning and Gradient-Boosted Trees outperform other models. Computers in Biology and Medicine, vol. 114, 2019, article 103456.
Jun W., Shitong W., Chung F.-L. Positive and negative fuzzy rule system, extreme learning machine and image classification. International Journal of Machine Learning and Cybernetics, vol. 2, 2011, pp. 261-271.

Информация об авторах / Information about authors

Мария Анатольевна ЛАПИНА – кандидат физико-математических наук, доцент кафедры вычислительно математики и кибернетики Северо-Кавказского федерального университета. Сфера научных интересов: цифровые технологии, управление информационной безопасностью, процессный подход, криптография.

Maria Anatolyevna LAPINA – Cand. Sci. (Phys.-Math.), Associate Professor at the Department of Computational Mathematics and Cybernetics at the North Caucasus Federal University. Research interests: digital technologies, information security management, process approach, and cryptography.

Назар Владимирович ПОДРУЧНЫЙ – студент Северо-Кавказского Федерального университета. Сфера научных интересов: криптография, машинное обучение, цифровые технологии, управление информационной безопасностью, процессный подход, образовательный процесс.

Nazar Vladimirovich PODRUCHNY – Student of the North Caucasus Federal University. Research interests: cryptography, machine learning, digital technologies, information security management, process approach, and educational process.

Михаил Андреевич РУСАНОВ – аспирант института информационных технологий, Московский финансово-юридический университет. Сфера научных интересов: комплексные системы защиты информации, информационно-коммуникационные технологии.

Mikhail Andreevich RUSANOV is a postgraduate student at the Institute of Information Technologies at the Moscow University of Finance and Law. Research interests: complex information protection systems, Information and Communication Technologies.

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