IBIMA Publishing
Communications of International Proceedings
https://ibimapublishing.com/p-articles/45AI/2025/4534625/
Vol. 2025 (20), Article ID 4534625

Comparison of the Effectiveness of Machine Learning Models in the Area of Security Breach Detection*

Oliwia NOWICKA and Adrian P. WOZNIAK

Military University of Technology, Warsaw, Poland

Correspondence should be addressed to: Oliwia NOWICKA, oliwia.roszkowska@student.wat.edu.pl

* Presented at the 45th IBIMA International Conference, 25-26 June 2025, Cordoba, Spain

Abstract

As the number of new and increasingly difficult to detect threats increases, the effectiveness of classification methods in intrusion detection systems (IDS) becomes particularly important. Previous research approaches mainly focus on detecting incidents in a binary approach, while issues related to multi-class classification, which includes recognizing multiple types of attacks simultaneously, remain insufficiently explored. This paper attempts to fill this gap by comparing the effectiveness of four popular machine learning algorithms - Support Vector Machine (SVM), Naive Bayes (Gaussian and Bernoulli variants), Random Forest, and XGBoost - evaluating their performance in both binary and multiclass classification tasks on the real CIC-IDS-2017 dataset. The data was subjected to comprehensive preprocessing, including cleaning, dimensionality reduction, and class balancing. Experiments assessed models based on accuracy, precision, recall, F1-score (macro and weighted averages), and computational efficiency. Results demonstrate that ensemble methods, particularly Random Forest and XGBoost, significantly outperform others in detection accuracy, with F1-scores exceeding 98% in binary and 85% (macro average) in multiclass classification after hyperparameter tuning. SVM showed solid performance but with higher computational costs, while naive Bayes models offered fast training but lower detection effectiveness. The findings confirm the suitability of tree-based ensemble models for intrusion detection systems (IDS), highlighting their robustness, scalability, and accuracy in identifying both known and novel threats.

Keywords: Security, Breach detection, Machine Learning

Introduction

Machine Learning (ML) plays a key role in security incident detection systems, offering a range of advanced capabilities. Models learn the normal behavior of systems and networks, which allows them to detect behavior that deviates from the learned pattern, suggesting a security breach. Using techniques such as clustering or statistical analysis, the system is able to identify unusual patterns that may indicate new ways of attack. What's more, ML-based systems can automatically respond to breaches by isolating infected elements or blocking malicious traffic. Contextual analysis conducted by advanced algorithms, enabling the reduction of false alarms by correlating multiple data points. AI allows for faster threat categorization by establishing relationships between seemingly unrelated events. In the context of comparing the performance of ML models, it is crucial to understand their strengths and weaknesses in specific detection tasks. For example, supervised methods (e.g. SVM) may be more effective in classifying known threats, while unsupervised methods (e.g. K-means) are better at detecting anomalies.

Cite this Article as: Oliwia NOWICKA and Adrian P. WOZNIAK, Vol. 2025 (20) "Comparison of the Effectiveness of Machine Learning Models in the Area of Security Breach Detection " Communications of International Proceedings, Vol. 2025 (20), Article ID 4534625, https://doi.org/10.5171/2025.4534625

Four supervised learning algorithms were selected to compare the effectiveness of detecting anomalies in network traffic: the Random Forest algorithm, the gradient boosting algorithm – XGBoost, the naive Bayesian classifier, and the Support Vector Machine – SVM. The choice of these methods resulted from the need to take into account different modeling approaches – from ensemble methods based on decision trees, through iterative boosting of prediction with regularization, to probabilistic techniques and algorithms based on maximizing the decision margin.

Random Forest Algorithm

Random forest is one of the most popular supervised learning algorithms, belonging to the group of ensemble learning methods, the main idea of which is to combine the results of several decision trees in order to increase the accuracy and stability of the model. The random forest model uses the bagging (bootstrap-aggregating) technique for training, which consists in randomly selecting samples and features from the training set with replacement. Thanks to this, each tree is trained on a slightly different subset of data, which reduces the variance of the model and limits the risk of overfitting. The construction of each tree in the forest is carried out independently, with additional randomness in the selection of features - instead of analyzing all of them, their random subset is selected, most often with the size $m=\sqrt{p}$, where p is the total number of available features. This double layer of randomness - both data and feature sampling - allows for the creation of an ensemble of trees with high diversity, which translates into a greater ability to generalize the model. The division of nodes in the set is based on criteria such as the Gini index or entropy. Once all trees are built, prediction for new data is made via a democratic voting system (for classification) or by calculating the arithmetic mean of the results (for regression). In the case of voting, when a new data sample is fed to the model, each decision tree in the forest generates its own label. The final result is the result of voting and choosing the prediction that received the most votes.

Gradient Boosting Algorithm – XGBoost

XGBoost is a relatively new supervised learning method. Unlike classical methods, XGBoost combines gradient boosting techniques with a number of system innovations, which allows for higher accuracy while maintaining scalability. The basic idea of the algorithm is gradient boosting - an ensemble technique of machine learning, consisting in sequentially building models, each of which tries to correct the errors made by its predecessors. This process continues until a certain level of error is reached or a fixed number of models are generated. The key innovation of XGBoost is the introduction of regularization directly to the objective function, which allows for simultaneous optimization of model results and control of its complexity. In practice, this means that XGBoost not only strives to minimize the error between predictions and actual values, but also penalizes overly complex trees. This approach significantly reduces the risk of overfitting. By penalizing the complexity of the model structure, the XGBoost objective function contributes to reducing the number of false alarms and improving the precision of classification.

Naive Bayes Classifier

The naive Bayes classifier is one of the simplest classification algorithms used in supervised learning, based on probability theory. It works by using Bayes' theorem to estimate the probability of a given sample belonging to a specific class. The key assumption of the algorithm is the so-called "naive" independence of features - it is assumed that individual input features are conditionally independent of each other, which significantly simplifies calculations, although it does not always reflect the actual dependencies in the data. The basic mechanism of the algorithm is Bayes' theorem used to calculate the conditional probability of a given sample belonging to a specific class, taking into account the observed features describing this sample.

Support Vector Machine Classifier

Support Vector Machine (SVM) is a robust tool in the field of supervised learning from the group of margin-maximizing classifiers, i.e. those whose task is to find the optimal decision boundary (hyperplane) that separates data belonging to different classes in the feature space. The basic mechanism of the classifier, distinguishing it from other algorithms, is the attempt to maximize the margin, which is the distance between the decision boundary and the closest data samples from each class. These samples are called support vectors. In practice, the algorithm not only finds the line dividing the classes (hyperplane), but also tries to place this boundary as far as

possible from the closest points of each class, thus maximizing the separating space. Formally, the problem for linearly separable data is formulated as an optimization problem in which the objective function is minimized.

Configuration of experiments

Data Collection

The CIC-IDS-2017 dataset, developed by the Canadian Institute for Cybersecurity, is one of the key tools used in research on the detection of computer network security breaches (Sharafaldin et al., 2018) The data was collected in a controlled laboratory environment, over a period of five business days, from July 3 to 7, 2017. It includes both normal network traffic and a wide range of security incidents. The attacks included in the dataset include brute force attacks implemented on FTP and SSH protocols, DoS and DDoS attacks, web attacks such as SQL injection or XSS, botnet activity, port scanning, and exploitation of vulnerabilities - Heartbleed. The structure of the dataset was carefully thought out and organized in the form of eight CSV files, each of which corresponds to a specific period (day or part of a day) and type of network activity. Each record in the set is characterized by an extremely rich description - it contains 79 attributes that describe both network flow statistics, IP addresses, port numbers, as well as time features, information about network protocols (such as TCP or UDP), protocol-specific flags and statistics calculated in different time windows. Such an extensive data structure allows for detailed analyses and precise selection of the most important parameters, which is the basis for applying supervised learning methods in classification tasks. An additional advantage of selecting the security data set is the careful and systematic labeling of data. Each record is assigned to one of two main groups: "BENIGN", representing normal network traffic, or a specific type of attack. The attack classes present in the dataset are: BENIGN, DDoS, DoS, PortScan, Brute Force, Web Attack, Bot, Infiltration, and Heartbleed. Systematic and precise labeling allows for conducting research on both binary and multi-class classification, which is the basis for applying supervised learning methods. However, it should be emphasized that one of the main challenges related to the analysis of CIC-IDS-2017 is the unbalanced nature of the data. The records representing normal traffic are much more numerous than those describing the individual types of attacks. Such a distribution of data requires the use of specialized techniques to counteract the problem of unbalanced sets, such as oversampling, random removal of samples, or the use of algorithms that are resistant to the unbalanced distribution of classes. To better illustrate what the data looks like in this dataset, Table 1 shows basic statistics for three sample features commonly used in network traffic analysis. These values show the large variability and outlier values that are typical of real network traffic. Negative values in some features (e.g., Flow Duration, Flow Bytes/s) are most likely due to measurement artifacts and were included during data preprocessing.

Table 1: statistics for three sample features

Feature	Mean	Std	Min	Max
Flow Duration	16 581 323.77	35 224 259.61	-13.00	119 999 998.00
Total Fwd Packets	10.28	794.17	1.00	219 759.00
Flow Bytes/s	1 409 834.54	26 562 624.93	-261 000 000.00	2 071 000 000.00

Data Preparation

The data preparation process allowed us to obtain a consistent and optimized set necessary for conducting analyses and correctly building classification models. It included a number of technical steps, such as importing data from CSV files, merging them, deep cleaning, optimization (including reducing memory consumption) and dimension reduction. This approach allowed us to eliminate erroneous and irrelevant information, which translated into increased precision of experimental results and enabled an accurate comparison of the effectiveness of selected supervised learning algorithms. The source data was imported from eight CSV files, each of which reflected network activity observed on a different day, and merged into a single data frame, which allowed us to maintain consistent record numbering and a full representation of network traffic. In the next part of the data preparation process, the key stage was to improve the quality of the data through deep cleaning: duplicates were removed, unnecessary spaces in column names were removed, and all empty and infinite values (changed to empty) were replaced with the median. Mapping and combining attack classes was the next stage of data preparation, which allowed for standardizing the type designations occurring in the set of security incidents. To simplify further analysis and the modeling process, related attack types were combined into one generalized

category. This procedure allowed for reducing the complexity of the classifier and made it easier to interpret the results, especially in the context of balancing data between individual categories, which was important for multi-class classification. In order to optimize memory usage when working with over 28 million records, numeric types were converted from the default 64-bit to 32-bit where the range of values allowed, which resulted in a significant reduction in resource consumption. However, before the data were used in ML algorithms, dimensionality reduction was performed using the Principal Component Analysis (PCA) method, and specifically its variant enabling batch processing - IncrementalPCA from the scikit-learn library.

In the experiment on the detection of security breaches, four supervised machine learning models were used: Support Vector Machine (SVM), Naive Bayes classifier (in two versions - GaussianNB and BernoulliNB), Random Forest and gradient boosting algorithm (XGBoost). Each of the models was prepared in two versions: basic, operating on default parameter settings, and optimized, in which hyperparameter tuning was performed using the Grid Search method. In order to improve the performance of all classification models, hyperparameter tuning was executed using GridSearchCV with the F1-score as the selection criterion; for the support vector machine, values of C equal to 0.1, 1, and 10 were investigated while maintaining the default 'rbf' kernel and gamma set to 'scale', yielding an optimal configuration of C = 1, kernel 'rbf', and gamma 'scale'. Similarly, the Random Forest model was tuned by testing n_estimators of 50, 100, and 200, max_depth values of 5, 10, and None, as well as min_samples_split values of 2 and 5, from which n_estimators = 200, max_depth = 10, and min_samples_split = 2 were determined to be most effective. In the case of XGBoost, experiments were conducted with n_estimators of 100 and 200, max_depth of 3 and 5, learning_rate of 0.1 and 0.2, and both subsample and colsample_bytree set to 0.8 and 1.0, resulting in the best performance for n_estimators = 200, max_depth = 5, learning_rate = 0.1, subsample = 1.0, and colsample_bytree = 1.0. Furthermore, for the Gaussian Naive Bayes classifier the var_smoothing parameter was varied over the range 1e-11, 1e-10, 1e-9, 1e-8, 1e-7, and 1e-6, with the optimal value being 1e-9, while for the Bernoulli Naive Bayes classifier the parameters alpha and binarize were examined at 0.5, 1.0, 1.5, and 2.0 for alpha and 0.0, 0.5, 1.0, and 1.5 for binarize, leading to the best results for alpha = 1.0 and binarize = 0.5. All of the hyperparameter tuning procedures were performed using cross-validation—five-fold for binary classification tasks and three-fold for multiclass scenarios—ensuring that the models are both robust and capable of effectively detecting complex network threats. During the experiment, two classification approaches were performed: binary and multiclass classification. The aim of the first one was to distinguish normal network traffic (marked with value 0) from an attack (marked with value 1) without delving into the category to which a given attack belonged. The multi-class approach, on the other hand, aimed not only to detect a security breach, but also to assign it a specific category – enabling the identification of whether a given network traffic corresponds to labels such as BENIGN, DDoS, DoS, PortScan, Brute Force, Web Attack, Infiltration or Heartbleed.

Measures for assessing the effectiveness of classifiers

Accuracy

The basic measure of the model's performance is accuracy, defined as the percentage of correctly classified samples in the entire test set. This is the ratio of the number of correct predictions to the number of all cases:

$$accuracy = \frac{number\ of\ correct\ predictions}{total\ number\ of\ cases}$$

Although high accuracy suggests that the model rarely makes mistakes, its interpretation can be problematic when the data is imbalanced. With such a distribution of data, the model can achieve high accuracy even though it ignores critical but infrequent attack classes. In such a case, the model may ignore such classes and the majority classes will dominate the result, so accuracy alone is not enough when evaluating models on a task with imbalanced classes.

Precision

Precision measures the percentage of correct positive predictions among all positive predictions. In the context of security incident detection tasks, it therefore measures what percentage of events classified by the model as an attack actually were one. Precision can be formally presented as:

$$precision = \frac{TP}{TP + FP}$$

where TP (True Positives) is the number of correctly detected attacks and FP (False Positives) is the number of cases where normal traffic was misinterpreted as an attack. High precision means that there are few false positives, which is crucial for anomaly detection systems.

Recall

Sensitivity measures what percentage of actual attacks were detected by the model. It is calculated as:

$$recall = \frac{TP}{TP + FN}$$

where, TP, as in the case of precision, is the number of correctly detected attacks, while FN (False Negatives) denotes the number of attacks that the model missed. A high sensitivity value means that the model effectively captures threats, although achieving a very high level of recall is often achieved at the cost of reducing precision, as the model may start to generate an excessive number of false alarms.

F1-score

To take into account both precision and sensitivity, the F1-score is used, which is the harmonic mean of these two metrics. The formula for the F1-score is as follows:

$$F1 = 2 * \frac{precision * recall}{precission + recall}$$

The F1 measure takes on a high value only when both precision and sensitivity are high. Therefore, in the case of unbalanced datasets, where the attack class is in the minority, the F1 measure is especially useful, as it simultaneously penalizes missing attacks (low sensitivity) and generating false alarms (low precision).

Macro and weighted average

In multiclass classification tasks, where it is important to distinguish between different types of attacks, two extensions of the F1 measure are used. The first one, the so-called F1-macro, is calculated as the arithmetic mean of the F1-scores determined for each class separately, which means that each class has an equal impact on the final result. This approach allows to take into account the effectiveness of detecting also rarely occurring types of attacks, which allows to illustrate the overall effectiveness of the model. The second extension is F1-weighted, i.e. a weighted average, in which weights are assigned depending on the number of examples for a given class. Thanks to this, dominant classes have a greater impact on the final result, which may, however, hide shortcomings in the detection of rare cases.

Computational performance

In addition to qualitative measures, the computational efficiency of the models was also assessed, taking into account training time and prediction time. Training time describes the time required to train the model on training data (using hyperparameters tuning), while prediction time is the speed of classifying new samples. In the case of models trained with hyperparameter tuning using GridSearchCV, the reported training time refers to the total cumulative time required to evaluate all tested hyperparameter combinations.

Results

Binary classification

Binary classification in IDS systems distinguishes normal traffic (marked as "BENIGN") from traffic containing potential attacks ("attack"). This type of analysis is crucial for improving network security, enabling early detection and response to threats, which helps protect data from unauthorized access or other malicious activities. The CIC-IDS-2017 dataset was used for the analysis. The data preparation process began with merging eight sets, which resulted in 2,830,743 rows and 79 columns. Then, a series of preprocessing operations were performed, such as removing duplicates, filling in missing values, eliminating columns with zero variance, and optimizing data types, which contributed to reducing memory consumption by more than 47 percent. To prepare the set for binary classification, oversampling was used to equalize the number of examples of both classes. Before this operation, the distribution of classes in the training set was as follows: 31,191 examples for the class marked as "attack" and 6,309 examples of normal traffic. After oversampling was applied, the sample sizes of both classes were unified, ensuring a balance between attack examples and normal traffic.

Table 2: Results for binary classification

Lp.	Model	Accuracy [%]	Precision (attack) [%]	Recall (attack) [%]	F1-score (attack) [%]	Train time [s]	Prediction time[s]
1.	SVM (base)	93.9	74.4	96.8	84.1	200.09	2.76
2.	SVM (GridSearchCV)	95.3	79.0	98.2	87.6	532,68	3.92
3.	GaussianNB (base)	46.7	23.7	97.9	38.2	0.03	0.01
4.	GaussianNB (GridSearchCV)	46.7	23.7	97.9	38.2	2.50	0.0
5.	BernoulliNB (base)	78.7	43.5	88.2	58.2	0.03	0.01
6.	BernoulliNB (GridSearchCV)	78.7	43.5	88.2	58.2	2.93	0.01
7.	Random Forest (base)	99.6	98.6	98.8	98.7	24.11	0.06
8.	Random Forest (GridSearchCV)	99.6	98.6	98.8	98.7	228.72	0.06
9.	XGBoost (base)	97.9	90.4	97.7	93.9	0.55	0.55
10.	XGBoost (GridSearchCV)	99.6	98.2	99.2	98.7	31.41	0.01

Table 3: Confusion Matrices for Binary Classification

Lp.	Model	Confusion Matrix
1.	SVM (base)	[9697 700 67 2036]
2.	SVM (GridSearch)	[9849 548 37 2066]
3.	GaussianNB (base)	[3774 6623 45 2058]
4.	GaussianNB (GridSearchCV)	[3778 6619 45 2058]
5.	BernoulliNB (base)	[7984 2413 248 1855]
6.	BernoulliNB (GridSearchCV)	[7984 2413 248 1855]
7.	Random Forest (base)	[10367 30 26 2077]
8.	Random Forest (GridSearchCV)	[10367 30 26 2077]
9.	XGBoost (base)	[10179 218 49 2054]
10.	XGBoost (GridSearchCV)	[10359 38 17 2086]

Based on the obtained results, it can be clearly stated that algorithms based on decision trees, such as Random Forest and XGBoost, achieved the highest effectiveness in attack detection. After optimizing hyperparameters,

both models achieved an accuracy of 99.6%, and the F1-score for the attack class was 98.7%, which indicates high precision and the ability to detect threats with a minimum number of misclassifications. These models offer the best compromise between accuracy and training time, and their resistance to overtraining and the possibility of parallel data processing mean that they can be recommended as the main algorithms in IDS systems, where the most accurate attack detection at an acceptable computational cost is important. The SVM model, despite the longer training time and greater resource requirements, also presented satisfactory results – after optimization, the accuracy increased from 93.9% to 95.3%, and the F1-score from 84.1% to 87.6%, which makes it a valuable addition to hybrid intrusion detection systems, where combining different classifiers can increase resistance to various attacks. On the other hand, the Bayesian models – GaussianNB and BernoulliNB – despite a very short training time and low computational requirements, achieved much worse results, as the best F1-score for the attack class was only 58.2% for BernoulliNB. However, the speed of training and low resource requirements mean that naive Bayesian classifiers can be used as a solution for initial filtration of network traffic, the results of which require later verification by more advanced algorithms.

Multi-class classification

In the multiclass approach, the model's task was extended: the model not only had to detect that an attack had occurred, but also classify it into the correct category. The experiment used attack labels from the CIC-IDS-2017 dataset, distinguishing 8 types of attacks (including DDoS, DoS, PortScan, Brute Force, Web Attack, Bot, Infiltration, Heartbleed) and normal traffic (BENIGN). In total, the classifiers operated on 9 classes. After preprocessing (removing duplicates, filling in gaps, reducing zero-variance columns and optimizing data types - which allowed reducing memory consumption by over 47%) and applying oversampling, the sample size for the training set was unified for all attack categories - each class had 31,137 examples. The experiment compared four types of algorithms, as in the case of binary classification: SVM model, Naive Bayes classifier in two variants (GaussianNB and BernoulliNB), Random Forest and XGBoost. The results are presented both in the baseline version and after hyperparameter optimization using GridSearchCV.

Table 4: Results for multi-class classification

Lp.	Model	Accuracy [%]	Macro avg F1-score [%]	Weighted avg F1-score [%]	Train time [s]	Prediction time [s]
1.	SVM (base)	80.6	54.1	86.2	2365.13	56.53
2.	SVM (GridSearchCV)	86.0	57.4	89.8	4196.02	26.82
3.	GaussianNB (base)	36.7	35.3	43.8	0.13	0.03
4.	GaussianNB (GridSearchCV)	36.7	35.3	43.8	4.75	0.03
5.	BernoulliNB (base)	69.5	43.0	77.0	0.14	0.01
6.	BernoulliNB (GridSearchCV)	64.8	43.1	73.9	3.46	0.01
7.	Random Forest (base)	99.4	85.5	99.5	124.51	0.07
8.	Random Forest (GridSearchCV)	99.5	86.0	99.5	1668.51	0.04
9.	XGBoost (base)	96.4	68.1	96.9	5.58	0.01
10.	XGBoost (GridSearchCV)	99.6	85.2	99.6	958.39	0.03

Based on the obtained results, the following conclusions can be drawn. Ensemble methods such as Random Forest and XGBoost achieved the highest effectiveness in attack detection. After optimizing hyperparameters, an accuracy of about 99.6% was obtained, as well as an F1-score for the attack class of 98.7% in binary classification, and in multi-class tasks, Random Forest and XGBoost achieved an accuracy of 99.5% (macro-avg F1 = 86.0% and weighted-avg F1 = 99.5%) and 99.6% (macro-avg F1 = 85.2% and weighted-avg F1 = 99.6%), respectively. Both models are resistant to overfitting and can process data in parallel, which ensures stable operation on very large sets, and the minimal prediction time (< 0.1 s) makes them practical despite expensive training (approximately 1668 s and 958 s, respectively). The SVM model, despite higher computational requirements and a significant increase in training time (even over 4190 s for multiclass classification), showed improved effectiveness after optimization – in binary classification, accuracy increased from 93.9% to 95.3%,

and F1-score from 84.1% to 87.6%, while in multiclass, accuracy improved from 80.6% to 86.0% (macro-avg F1 increased from 54.1% to 57.4%, and weighted-avg F1 reached 89.8%). Due to the long training time, SVM is mainly used as a supplement in hybrid detection systems, enhancing resistance to various types of attacks. Naive Bayesian classifiers, such as GaussianNB and BernoulliNB, are characterized by very short training time (< 5 s) and minimal hardware requirements, but they achieve much worse results - the best F1-score was 58.2% for BernoulliNB (with macro-avg F1 of 35.3% for GaussianNB and about 43% for BernoulliNB, with weighted-avg F1 of 43.8% and about 74%, respectively). Due to the insufficient effectiveness in stand-alone detection, their use should be limited to the role of fast pre-filters, rejecting part of the normal traffic and passing selected data for further verification by more advanced algorithms. These results are comparable to the best published results for the CIC-IDS-2017 dataset. For example, recent studies using deep learning approaches, such as hybrid LSTM-Autoencoder models, have reported F1-scores up to 99.9% and accuracy exceeding 99.9% (Hnamte & Hussain, 2023; Sharafaldin et al., 2018). The advantage of the presented approach is the use of less computationally intensive models, which provide high detection performance while maintaining practical applicability for large-scale or real-time systems.

Conclusions

Research conducted on the CIC-IDS-2017 dataset enabled the evaluation of modern algorithms such as support vector machine (SVM), naive Bayes classifier, Random Forest and XGBoost, indicating their ability to automatically detect unusual patterns in network data and improve the effectiveness of detection systems compared to traditional rule-based methods. The data used for experiments were processed using a number of methods, including duplicate elimination, pre-cleaning, data type optimization, dimensionality reduction using PCA and the use of set equalization techniques (SMOTE oversampling). The comprehensive nature of the applied processing methods allowed for obtaining a consistent and representative data set, which is crucial for the proper evaluation of the performance of individual models. The most important achievement of the work was the confirmation that models based on decision trees, in particular XGBoost, can effectively use gradient boosting and regularization mechanisms, which allows for better coping with the problem of overfitting. Models such as Random Forest showed high stability and resistance to changing data conditions, which is a significant advantage when analyzing large sets of information. In turn, SVM and naive Bayes classifiers, despite their limitations, can still be an attractive solution due to their low computational cost and simplicity of implementation.

Acknowledgment

This work was supported by the Military University of Technology under Project UGB 531-000023-W500-22.

References

- Canadian Institute for Cybersecurity: Intrusion Detection Evaluation Dataset (CIC-IDS2017). https://www.unb.ca/cic/datasets/ids-2017.html
- Sharafaldin I., Lashkari A.H., Ghorbani A.A.: Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP), Funchal, Madeira, Portugal, 22-24.01.2018.
- Mitchell T.M.: Machine Learning. McGraw-Hill Science/Engineering/Math, New York, 1997
- Samuel A.L.: Some Studies in Machine Learning Using the Game of Checkers. IBM Journal of Research and Development, 3 (3), 210-229, 1959.
- Ahmad I., tłumaczenie Piwko Ł., „50 algorytmów, które powinien znać każdy programista: Klasyczne i nowoczesne algorytmy z dziedziny uczenia maszynowego, projektowania oprogramowania, systemów danych i kryptografii. Wydanie II, Helion, Gliwice, 2024
- Breiman L.: Random Forests. Machine Learning, 45 (1), 5-32, 2001. https://doi.org/10.1023/A:1010933404324
- Chen T., Guestrin C.: XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, California, USA, 13-17.08.2016. https://arxiv.org/pdf/1603.02754
- Cortes C., Vapnik V.: Support-Vector Networks. Machine Learning, 20(3), 273-297, 1995. https://doi.org/10.1007/BF00994018
- Hnamte, S., & Hussain, S. (2023). Hybrid LSTM-Autoencoder for Intrusion Detection in Computer Networks. Computers & Security, 127, 103026.