UDC 004.02
DOI: 10.15587/1729-4061.2024.317471

EVALUATION AND OPTIMIZATION OF THE NAIVE BAYES ALGORITHM FOR INTRUSION DETECTION SYSTEMS USING THE USB-IDS-1 DATASET

Nurbek Konyrbaev
PhD, Associate Professor, Head of Department*
Yevheniy Nikitenko
Associate Professor**
Vadym Shtanko
PhD Student**
Valerii Lakhno
Professor**
Zharasbek Baishemirov
Corresponding author
PhD, Professor
Department of Mathematics and Mathematical Modelling
Postdoctoral Researcher
Department of Science
Abai Kazakh National Pedagogical University
Dostyk ave., 13, Almaty, Republic of Kazakhstan, 050010
Professor
School of Applied Mathematics
Kazakh-British Technical University
Tole bi str., 59, Almaty, Republic of Kazakhstan, 050010
E-mail: zbai.kz@gmail.com
Sabit Ibadulla
PhD*
Asem Galymzhankyzy
Master, Teacher*
Erkebula Myrzabek*
*Department of Computer Science
Institute of Engineering and Technology
Korkyt Ata Kyzylorda University
Aiteke bi str., 29A, Kyzylorda, Republic of Kazakhstan, 120014
**Department of Computer Systems, Networks and Cybersecurity
National University of Life and Environmental Sciences of Ukraine
Heroiv Oborony str., 15, Kyiv, Ukraine, 03041

This study takes a look into the application of the Naive Bayes machine learning algorithm to enhance the accuracy of Intrusion Detection Systems (IDS). The primary focus is to assess the algorithm's performance in detecting various types of network attacks, particularly Denial of Service (DoS) attacks. This research proposes using Naive Bayes to improve intrusion detection systems that struggle to keep pace with evolving cyber threats. This study evaluated the efficiency scores of the Naive Bayes classifying model for two different dependency scenarios and identified strong and weak properties of this model. The Naive Bayes classifier demonstrated satisfactory results in detecting network intrusions, especially in binary classification scenarios where the goal is to distinguish normative and malicious traffic due to its simplicity and efficiency. However, its performance declined in multi-class classification tasks, where multiple types of attacks need to be differentiated. The study also highlighted the importance of data quality and quantity in training machine learning models because of the impact of those parameters on the model efficiency. The USB-IDS-1 dataset, while useful, has limitations in terms of the variety of attacks. Using datasets with a wider range of attack types could significantly improve the accuracy of IDS. The findings of this research can be applied to such domains as network security, cybersecurity, and data science. The Naive Bayes classifier can be integrated into IDS systems to enhance their ability to detect and respond to cyber threats. However, it is essential to consider the limitations of the algorithm and the specific conditions of its environment. To maximize the effectiveness of the Naive Bayes classifier, it could be promising to optimize and normalize the data to improve the accuracy of the model and combine Naive Bayes with the other machine learning algorithms to address its limitations

Keywords: intrusion detection systems (IDS), Naive Bayes method, python, machine learning, Denial of Service (DoS) attacks, USB-IDS-1 dataset

Received 01.10.2024
Received in revised form 25.11.2024
Accepted 06.12.2024
Published 25.12.2024

***How to Cite**: Konyrbaev, N., Nikitenko, Y., Shtanko, V., Lakhno, V., Baishemirov, Z., Ibadulla, S., Galymzhankyzy, A., Myrzabek, E. (2024). Evaluation and optimization of the naive bayes algorithm for intrusion detection systems using the USB-IDS-1 dataset. Eastern-European Journal of Enterprise Technologies, 6 (2 (132)), 74–82. https://doi.org/10.15587/1729-4061.2024.317471*

1. Introduction

In today's rapidly evolving digital landscape, cyber threats are becoming increasingly sophisticated, posing significant risks to individuals, organizations, and critical infrastructure. Traditional Intrusion Detection Systems (IDS) frequently encounter difficulties in effectively detecting and responding to these threats, particularly in the face of novel and zero-day attacks.

Traditional intrusion detection systems are limited by their reliance on predefined signatures and rules, which are easily circumvented by attackers. There is a growing need for advanced, adaptive IDS solutions that can automatically learn and adapt to new attack patterns.

Machine learning and deep learning present promising prospects for the development of intelligent IDS. However, these techniques often require significant computational resources and large amounts of training data. This may pose a challenge, particularly for environments with limited resources or real-time applications.

Therefore, there is a pressing need to explore machine learning algorithms that are both effective and efficient in detecting network intrusions. By identifying algorithms that can accurately classify network traffic with minimal computational overhead, it will be possible to develop more robust and scalable IDS solutions.

Accordingly, research focused on developing efficient, adaptive Intrusion Detection Systems using machine learning methods is especially relevant. As cyber threats continue to evolve, it becomes increasingly important to seek solutions that not only enhance detection accuracy but also reduce computational demands, making intelligent IDS more applicable across various environments, including resource-constrained and real-time settings.

2. Literature review and problem statement

Intrusion Detection Systems (IDS) play a pivotal role in modern cybersecurity, attracting considerable interest from both researchers and practitioners. A fundamental concern in this domain is the ability of IDS to effectively detect intrusions or suspicious activities within the networks they are deployed in. While significant advancements have been achieved, numerous unresolved challenges persist, making this a fertile area for further investigation.

For instance, the research presented in [1] provides an extensive exploration of machine learning (ML) and deep learning (DL) techniques as applied to IDS. This publication begins by defining foundational concepts and categorizes IDS based on deployment (network-based and host-based) and detection strategies (signature-based and anomaly-based). The authors propose a structured pipeline for employing ML in IDS, comprising three primary phases: data preprocessing, training, and testing. A diverse array of algorithms is explored, ranging from classical models like decision trees, K-nearest neighbors, and support vector machines to advanced DL architectures such as convolutional neural networks (CNN), recurrent neural networks (RNN), and long short-term memory networks (LSTM). In addition to this broad algorithmic coverage, the work examines performance evaluation metrics and synthesizes insights from multiple comparative analyses conducted by other experts in the field. However, there is insufficient discussion regarding how dataset configurations, such as size, diversity, or preprocessing techniques, impact learning models performance. The research lacks a deep dive into the specifics of each reviewed machine learning model. This limitation indicates that further investigation into the dependency between dataset properties and IDS learning efficient is needed. Moreover, the authors draw attention to several broader challenges that hinder the adoption of ML-based IDS. These include the scarcity of relevant and systematically organized datasets for model training, imbalances within existing datasets, the suboptimal performance of IDS in real-world scenarios, and the high computational demands required to deploy these systems effectively.

The issue of dataset relevance is examined in greater depth in [2], where older datasets such as KDDCUP99 and NSKDD are critically evaluated. These datasets are shown to be outdated and insufficient for addressing the complex requirements of contemporary IDS. To address these limitations, [2] delves into the creation and characteristics of the UNSW-NB15 dataset. The paper provides a meticulous account of the algorithms and tools used for data collection, as well as a detailed taxonomy of its features, including the assignment of attack and normal traffic labels. A comparative analysis between KDDCUP99 and UNSW-NB15 highlights the latter's advantages, particularly its improved relevancy, data balance, and feature richness, which collectively contribute to better ML performance. All listed datasets have extensive descriptions, but researchers don't provide any figures on how the latter and more extensive ones affect the performance of machine learning models. This research did not present efficiency scores because this was not its main goal.

Further comparative work is presented in [3], where the UNSW-NB15 dataset is evaluated alongside Bot-IoT and CSE-CIC-IDS2018. This study examines the network protocols utilized during data collection and the types of attacks included. Despite their advancements, these datasets remain constrained by the environments in which they were created, which limits their applicability in replicating real-world attack scenarios. This observation underscores the importance of designing datasets that more accurately reflect the dynamic nature of real-world networks. In general, this research provides extensive comparison data for the efficiency scores between diverse datasets and various machine learning algorithms. However, it lacks information regarding the impact of dataset parameters on the efficiency of the studied algorithms.

The literature also highlights the value of specialized datasets for IDS. For instance, [4] investigates the AWID3 dataset, which is tailored for IEEE 802.11 wireless networks. This dataset includes data on attacks specific to wireless environments, such as the recently discovered Krack and Kr00k vulnerabilities. These datasets emphasize the importance of contextual specificity in IDS training, as wireless networks present unique challenges compared to traditional wired environments. As this study reviews the variance of the attacks, the key features and scenarios of their appearance, do not contain any information about appliance of those data for the machine learning.

In the realm of Internet of Things (IoT) security, the work in [5] utilizes the CIC IDS 2017 dataset to assess the efficacy of deep learning methods, including CNNs, RNNs, and LSTMs. While these advanced models demonstrate strong detection capabilities, their high computational requirements pose a barrier to deployment in resource-constrained environments, such as IoT devices. This research compares the efficiency scores of multiple machine learning models using various training datasets, but it lacks information about how the structure of the aforementioned dataset affects machine learning model efficiency.

Synthetic datasets like USB-IDS-1, discussed in [6], represent another avenue for progress. This dataset integrates data from multiple OSI model layers and simulates diverse

DDoS attack scenarios under varying server configurations. By capturing both protected and unprotected network node conditions, USB-IDS-1 provides valuable insights into attack dynamics. This research establishes a new dataset with extensive network data that includes traffic for various network scenarios. It requires a closer look and detailed study to understand how machine learning models will behave on such detailed data. Follow-up work in [7] leverages this dataset to investigate feature selection techniques using genetic algorithms. The results demonstrate that reducing the feature space can significantly enhance computational efficiency without sacrificing classification accuracy, offering a pathway to more scalable IDS solutions. This study examines the effectiveness scores of various machine learning models based on the number of features used for classification. It contains efficiency scores for specific combinations of data types, which are very variable. These suggest further research related to the efficiency of the machine learning model based on the specifics of the USB-IDS-1 dataset.

The field has also seen significant focus on individual machine learning methodologies. For example, [8] provides an in-depth analysis of RNNs, highlighting their capability to model sequential data patterns effectively. This research includes a systematic study of the RNN machine learning model, provides data about feature optimization algorithm, and includes data about feature selection. Furthermore, in contrast to the previous studies, this study comprises data regarding the duration of model training. Similar to [7], it also emphasizes the dependence of the model on the quantity of features. There was no investigation into the efficiency dependence on the other data parameters, except for feature enhancement. Similarly, [9] explores the twin support vector machine framework, emphasizing its high precision and suitability for specific classification tasks. However, this model is quite complex and requires additional implementation efforts, which could be a crucial drawback when building an IDS machine learning framework.

Taken together, these findings reveal a critical need to advance research in two key directions. First, there is a pressing requirement to develop and utilize datasets that are not only relevant and comprehensive but also reflective of real-world network dynamics. Characteristics such as dataset balance, richness of features, and diversity of attack scenarios play a pivotal role in determining IDS performance. Second, the exploration of scalable and computationally efficient ML techniques is vital, particularly for applications in environments with limited resources. Addressing these challenges will enable the development of IDS that are robust, adaptable, and capable of effectively countering the ever-evolving landscape of cyber threats.

3. The aim and objectives of the study

The aim of this study is to determine the effectiveness of the Naive Bayes machine learning algorithm in enhancing Intrusion Detection Systems (IDS) by improving their ability to detect various network attacks, including Denial of Service (DoS) attacks. The establishment of relationships between the algorithm's performance and the quantitative and qualitative features of the training dataset will allow understanding limitations and possible appliance scenarios of the Naive Bates model for the intrusion detection system's machine learning.

To achieve this aim, the following objectives are set:

– evaluation of the performance of the Naive Bayes machine learning model in dependence on data quantitative properties such as amount of data entries using the USB-IDS-1 dataset;

– evaluation of the performance of the Naive Bayes machine learning model in dependence on data qualitative properties such as data variance using the USB-IDS-1 dataset.

4. Materials and methods

4. 1. Object and hypothesis of the study

The object of this study is a Naive Bayes machine learning model. This model is based on the Bayes theorem, which is a fundamental tool in probability theory. The word "naive" in the name of this model refers to the use of the normal distribution law to calculate posterior Bayesian probability.

The main hypothesis of this research is that the Naive Bayes model has low efficiency for large, homogenous datasets that contain multiple data classes. However, due to the mathematical principles upon which this model is based (calculation of the probability of data belonging to a specific data class), its efficiency may increase with the decrease in the data classes. It could be assumed that with less data classes, the model will be able to make more accurate classifying decisions, as it will have less number of probability values to compare.

This research assumes that the data features of the training dataset are independent of one another. This simplification allows for the use of the Naive Bayes learning model, though it is not reflecting real data scenarios, as the features of the network data could have dependency in a real world. For example, the transfer duration of a packet may depend on its size. Furthermore, this model uses normal data distribution, which has limitations in the representation of real network traffic. In reality, traffic patterns can vary across various networks, such as wireless networks, local networks with particular security settings, Internet of Things networks, etc.

Let's consider the formulation of Bayes' theorem: Let A and B be two arbitrary events, with $P(B)\neq 0$. Then, the conditional probability of event A, given that event B has occurred, is expressed by the formula:

$$P(A|B)=\frac{P(A)\times P(B|A)}{P(B)}, \quad (1)$$

where $P(A|B)$ is the conditional probability of event A given that event B has occurred (posterior probability);

$P(B|A)$ is the conditional probability of event B given that event A has occurred (likelihood probability);

$P(A)$ is the prior probability of event A (initial estimate of probability);

$P(B)$ is the prior probability of event B (marginal probability).

Let's also briefly review the formula for the normal Gaussian distribution, which is another important component of this model:

$$f(x)=\frac{1}{\sigma\sqrt{2\pi}}e^{\frac{(x-a)^2}{2\sigma^2}}, \quad (2)$$

where x is the value of the random variable;
a is the expected value (mean);
σ is the standard deviation.

This method is quite simple because all the necessary parameters of the machine learning model are calculated using simple formulas, making this model easy to implement.

Another advantage is its computational efficiency, as the complexity of training such a model is linear in relation to both the number of training examples and the number of data features. In turn, the complexity of classification is linear with respect to the number of features and does not depend on the number of training examples. Another advantage of this method is the easy scalability of training: Naive Bayes works with low-order probability estimates derived from training data, and these estimates are quite simple to update as new training data becomes available. This method always uses all attributes for all predictions, making it relatively insensitive to noise in the classified examples [10].

The Naive Bayes model assumes that each feature in the data is independent of the others and submits a normal distribution within each class.

The principle of work of the Naive Bayes classifier is to select the class or category of data for which the posterior probability is the highest for the object being classified.

Since this classifier uses normal distribution to calculate posterior probability, training such a machine learning model involves calculating the parameters of the normal distribution, i.e. the means and standard deviations for the classes for which the classification is performed.

The mean value is calculated using the formula:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i, \tag{3}$$

where n is the number of entries in the dataset.

For the standard deviation calculations, the following formula was used:

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}, \tag{4}$$

where n is the number of the entries in the dataset;
$\bar{x}$ is the mean value calculated by the formula mentioned before.

During classification, based on the calculated means and standard deviations, the probability of the object belonging to a particular class is determined. These results are easy to interpret, as the outcome is the probability that the object belongs to one data class or another.

To evaluate the model's performance, the accuracy and precision metrics was used.

Accuracy was calculated by the formula:

$$Accuracy = \frac{TP + TN}{TP+TP+FP+FN}, \tag{5}$$

And precision was calculated using the following formula:

$$Precision = \frac{TP}{TP+FP}, \tag{6}$$

where TP (true positive) is the number of the attack data entries classified as attacks;

TN (true negative) is the number of normal traffic entries classified as normal traffic;

FP (false positive) is the number of the normal entries classified as attacks;

FN (false negative) is the number of attack data entries classified as normal traffic.

Accuracy reflects the ratio of correctly classified records to the total number of records. Precision is the ratio of correctly classified records assigned to a particular class to all records assigned to that class [11, 12].

4. 2. Rationale for selecting the dataset

As mentioned above, the network dataset used for training and evaluating intrusion detection systems is extremely important. Datasets like KDD Cup 99, NSL-KDD, DARPA 1998, and UNSW-NB15 are quite common for training and evaluating IDS, but they have certain drawbacks, such as being imbalanced or outdated. In this article, let's examine the USB-IDS-1 dataset. This dataset is relatively new (data collected in 2021) [6] and consists of 16 separate CSV files containing data on 4 different types of Denial of Service attacks, such as HULK, Slowloris, TCP-Flood, and Slowhttptest [13].

For each type of attack, data is collected under different network configurations, for example, the file HULK-NoDefence.csv contains data for a network without any security measures, whereas HULK-Reqtimeout.csv contains network data for this type of attack on a server protected by reqtimeout settings. Such data represents various network conditions, so they are quite relevant for IDS evaluation.

USB-IDS-1 also includes a separate file with network data collected under normal traffic conditions, i.e. the network is not subjected to any attacks. In total, this dataset contains more than 4.5 million records, each characterized by 82 features, such as the sender's IP address, the receiver's IP address, connection duration, the number of packets transmitted during the session, and timestamps that contain information about the session start time, etc. In fact, all the features are metadata of network packets rather than the packets themselves, as the payload in modern networks is usually encrypted, making it unsuitable for processing. The vast majority of the mentioned features are numerical, making this dataset well-suited for processing using the Naive Bayes method.

An undeniable advantage of this dataset is that all records include labels for the corresponding type of attack or indicate that the data is normal, therefore, USB-IDS-1 can be used for training and evaluating IDS without the need for additional data processing related to data labeling. This significantly speeds up the process of preparing data for use in machine learning models.

4. 3. Software tools

One of the most widely used programming languages for machine learning is Python. Among its advantages are the simple structure of the code and the speed of development. On the other hand, as an interpreted language, it has slower application execution speed in comparison with compiled languages like Java or C++. Furthermore, as a language without static typization, Python code could contain errors that are detectable only in application runtime. From a machine learning perspective, its advantage is the large number of specialized Python libraries, such as NumPy, Pandas, Scikit-learn, TensorFlow, and PyTorch. These libraries pro-

vide ready-made tools for data processing, model building, and their evaluation, which greatly simplify the work of researchers and developers. In overall the reason for the choosing of the Python language for the research mainly are: development speed and extensive support of various machine learning and data processing libraries, which allow the writing of simple and efficient code. Let's take a closer look at some of the aforementioned libraries, as to use them later for writing code.

Scikit-learn is one of the most popular Python libraries for machine learning. It offers a wide range of tools for data preprocessing, classification, regression, clustering, dimensionality reduction, and other machine learning tasks. Its features include standardization, normalization, encoding categorical variables, and other methods of preparing data for model training. The library also implements linear regression, logistic regression, support vector machines, decision trees, random forests, neural networks, and many more. Additionally, it allows the calculation of various accuracy metrics and the evaluation of machine learning model quality.

The NumPy library is one of the core Python libraries used for scientific computing. It provides a powerful N-dimensional array object, which forms the foundation for many other libraries, including Scikit-learn. It includes the capability to create, manipulate, and compute large multidimensional arrays.

The Pandas library, in turn, is designed for high-level data analysis and is built based on NumPy. It offers tools for working with tabular data, such as tables and series. Its features include working with DataFrame class objects, which are data structures that represent tabular data with rows and columns. It also simplifies working with various data formats, such as CSV, Excel, SQL, and others. Pandas allows performing various data manipulations, such as sorting, filtering, merging, and aggregating data.

Matplotlib is a library for creating static, animated, and interactive data visualizations in Python. It offers a wide range of chart types, from simple line graphs to complex three-dimensional plots. It allows displaying function plots, scatter plots, visualizing data distribution, and more.

4. 4. Description of the data processing algorithm

For the Naive Bayes method, as with other machine learning methods, there is a specific set of standard actions performed during model training. The first step is reading data from a storage medium, such as a file. In this case, let's use reading data from .csv files, as the USB-IDS-1 dataset consists of such data (Fig. 1).

Since the data in the network dataset have different nature, for example, it contains timestamps in the form of formatted dates, IP address ranges, individual IP addresses, NaN (not a number) values, and Infinity, and they require transformation or removal from the dataset, as they are unsuitable for numerical calculations. Thus, the first step in data preparation is removing features with non-numeric values (NaN and Infinity). Timestamps can easily be converted into millisecond values using Python, which are numeric and therefore meet the model's requirements. Additionally, for the model to work correctly, it is necessary to remove records with empty values from the dataset, as well as features that do not vary within a class. This is because it is impossible to perform calculations on empty values, and it is impossible to compute model parameters for non-varying features, such as the expected value and standard deviation for each feature within a class. The same applies to non-numeric values, as mentioned above.

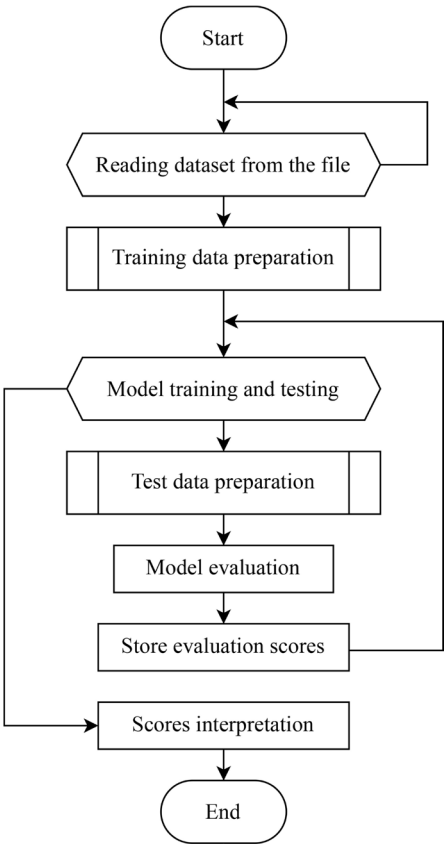

Fig. 1. General algorithm for data processing using the Naive Bayes method

Thus, after the previous data preparation, it is possible to obtain a dataset with 52 features, in comparison to the total of 82 features present in the initial dataset.

For additional randomization, the prepared dataset was shuffled to avoid sequential processing of records with the same labels. The complete input data preparation algorithm is shown in Fig. 2.

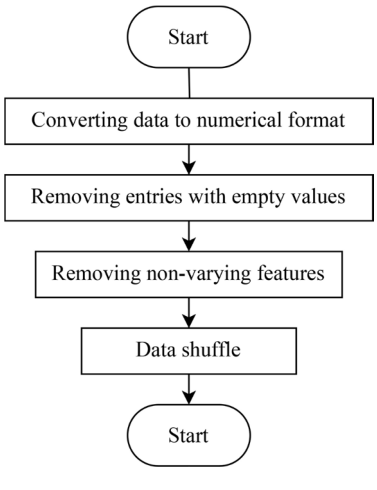

Fig. 2. Training data preparation algorithm

After the data preparation, the model is ready for training. Since the effectiveness (accuracy and precision) of the model was examined based on the amount of input data and the number of data classes, two groups of calculations with

different initial conditions were conducted. The first group of calculations processed data from all files in the training set for each iteration, i. e., considering all 16 blocks of attack data and 1 block of normal traffic data. The variable parameter for these calculations was the number of records, which was randomly selected using the Python random module within the range from 0 to 4500000 (the total number of records in the dataset). To establish the statistical dependence of the model's effectiveness on the number of records, the training and testing of the model was performed over 300 iterations (Fig. 3). Generally, the first group of calculations represents multi-class classification with the number of classes of 17 (16 attack related and one is for normative traffic). This experiment ignored data classes as the main its goal was to detect dependency on data number.

The second group of calculations had the following initial conditions: each iteration processed at least two blocks of training data, one of which had to be the block with normal traffic. A maximum of 17 blocks of data were used in a single iteration, which included 16 blocks of attack data and the block of normal traffic. For each iteration, the attack classes were randomly selected using the random module. Additionally, for each iteration, training and testing of the model were repeated 100 times. This group of calculations represented multi-class classification as well, but the number of classes was variable for each calculation. It varied from 2 up to 17. Data amount was ignored in this scenario. Moreover, the case with only two data classes was a partial case also known as binary classification. Binary classification for Naïve Bayes is no different from multi-class classification, as the probability of data entry is calculated for each class by the same formula. The only variable in those calculations was the number of classes for each iteration.

Let's take a closer look at how the model was trained and tested, as well as how it was evaluated for each individual calculation.

After the input data was prepared for each individual calculation, the labels were converted from text format to numerical format and recorded into a separate data array. Then, the data was split into training and test sets. When splitting the data, the labels were removed from the test data, so they no longer contained information about their class. Twenty percent of the original dataset was allocated to the test data, while the remaining 80 % was used for training. For the portion of the data allocated for training, the model parameters described in section 3 of this article – mean and standard deviation – were calculated.

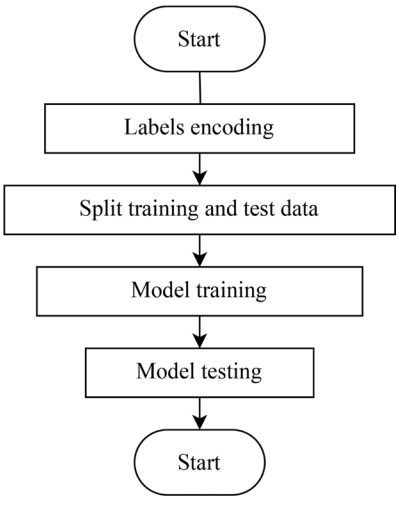

Fig. 3. Model training and testing algorithm

After training the model, it was tested. For each record in the test data, the posterior probability of belonging to each of the classes used in the calculation was computed. The calculated probability indicated the class to which the record belonged, after which the test data was labeled with the corresponding tags, completing the classification. After classification, using the tools from the scikit-learn library, the model was evaluated for each individual calculation by calculating the model's accuracy and precision. For both groups of calculations, the accuracy and precision values were recorded in separate .csv files, along with the initial conditions: the number of records and the number of attack classes, respectively.

The Naive Bayes algorithm was implemented to assess its performance in various classification scenarios for intrusion detection. Evaluation focused on metrics such as accuracy and precision.

5. Research results of the effectiveness of the Naive Bayes machine learning algorithm in enhancing Intrusion Detection Systems (IDS)

5. 1. Evaluation of the model performance in dependency of data amount

To investigate how data volume affects model performance, the dataset size was systematically varied during training and testing iterations.

Results:

– Accuracy dependency on the data amount: the model achieved an accuracy range between minimal value of 32 % and max value of 42 % with the regression approximately to 33–34 % as displayed in Fig. 4.

– Precision dependency on the data amount: model demonstrated precision values scatter between minimal value of 32 % and max value of 52 % with the regression approximately to 42–45 % as displayed in Fig. 5.

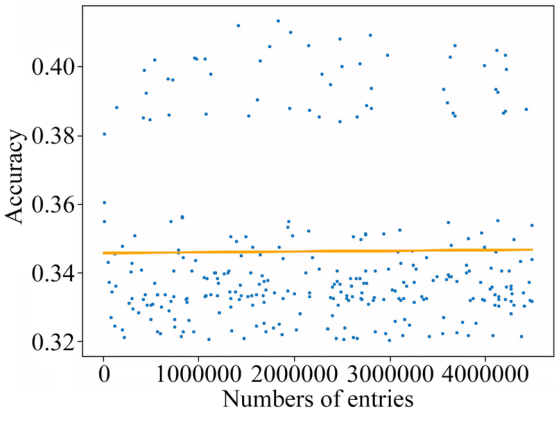

Fig. 4. Graph of the model accuracy dependence on the amount of data

The results presented in Fig. 4, 5 demonstrate that the model's performance, in terms of accuracy and precision, generally not depends on the amount of data used for training. Linear regression coefficient of $2 \cdot 10^{-10}$ display minimal dependency on the data volume for the accuracy. Similarly, the precision's linear regression coefficient was $2.08 \cdot 10^{-9}$ which also indicates insignificant value of dependency of the precision on the data amount.

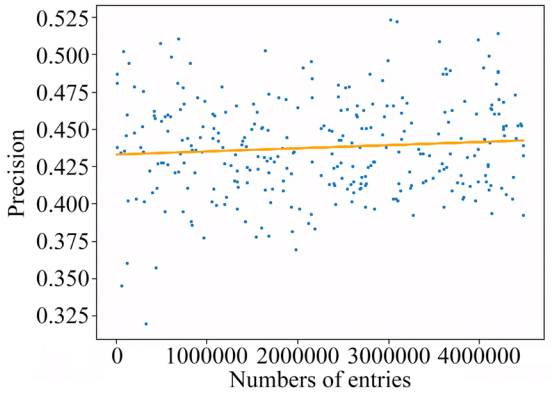

Fig. 5. Graph of the model precision dependence on the amount of data

Furthermore, it was noticed that in the Fig. 4 data points form two big groups. One of the groups contains points under accuracy value of approximately 35 %. Another group contains points over the accuracy value of approximately 38 %. This could be explained by data inconsistency. Despite that data was selected from a homogenous dataset that included all 16 types of attack data and data for normative traffic, data selection was randomized. This suggests that certain parts of the datasets impacted on the algorithm in terms of increasing accuracy values. These observations underline the need of feature engineering, data balancing and structuring or advanced algorithms to improve detection capabilities in diverse traffic scenarios.

5. 2. Evaluation of the model performance in dependency of data variance

The impact of data diversity, measured by the number of attack classes in the dataset, was analyzed to evaluate the algorithm's robustness in multi-class scenarios.

Results:

– Accuracy: a negative linear regression coefficient of $-3{,}9{\cdot}10^{-2}$ was observed. Maximum accuracy (more than 95 %) was achieved for datasets with normative traffic and a single attack class, while accuracy dropped to 33 % for datasets containing 16 attack classes (Fig. 6).

– Precision: precision exhibited similar trends, with a regression coefficient of $-2{,}9{\cdot}10^{-2}$. Precision values peaked at more than 95 % for datasets with fewer attack classes and declined to 40 % for datasets with 16 attack classes (Fig. 7). These results underscore the model's limitations in handling diverse datasets with multiple attack scenarios.

The trends depicted in Fig. 6, 7 clearly illustrate the limitations of the Naive Bayes algorithm in handling datasets with increasing diversity. As the number of attack classes rises, both accuracy and precision show a marked decline, evidenced by negative regression coefficients. These results highlight that the algorithm performs well in scenarios with low data diversity, such as datasets with normative traffic and a single attack class, achieving accuracy and precision levels exceeding 95 %. However, the dramatic performance drop when working with 16 attack classes underscores the algorithm's inability to capture complex relationships in high-dimensional, heterogeneous data. This finding suggests the importance of either adopting more advanced machine learning models or improving data preprocessing methods to enhance performance in such challenging scenarios. In addition, distinctive data points grouping was observed in the Fig. 6, 7 both for accuracy and precision scores. Instead of data points distribution around some core value similar to normal distribution, for scenarios with 2–4 attacks type it was detected few scattering cores. This could be explained by random selection of datasets with different attack types. Thus, some attack types are classified with higher efficiency than other ones, so this confirms once again that the nature of the data affects Naïve Bayes algorithm efficiency.

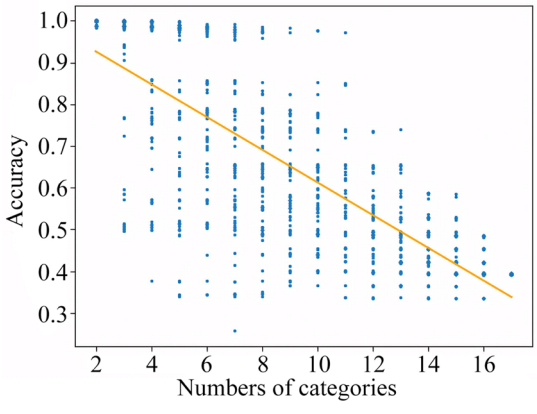

Fig. 6. Graph of the model accuracy dependence on the number of data categories

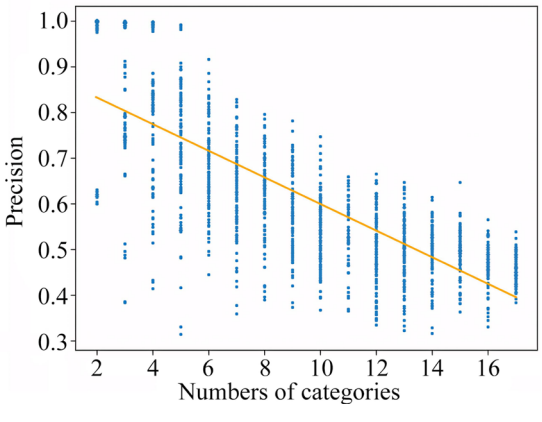

Fig. 7. Graph of the model precision dependence on the number of data categories

6. Discussion of research results ability the Naive Bayes to detect various network attacks

The results obtained in this study demonstrate that the accuracy and precision of the Naive Bayes model are largely independent of the size of the training dataset when the data are homogeneous (Fig. 4, 5). For instance, the linear regression coefficient for accuracy is $2{\cdot}10^{-10}$, and for precision, it is $2{,}8{\cdot}10^{-9}$, indicating negligible improvements with an increase in the dataset size. This suggests that for homogeneous data, increasing the dataset size does not significantly impact the model's performance. However, the number of data classes significantly affects these scores.

The findings indicate that the Naive Bayes algorithm performs well in binary classification, achieving accuracy and precision exceeding 95 % for normative traffic and single-attack classifications (Fig. 6). However, as the number of attack classes increases, both accuracy and precision decline. For 16 attack types and normative traffic, the model's accuracy drops to nearly a third of its binary classification performance, with a negative linear regression coefficient of $-3{,}9{\cdot}10^{-2}$ for accuracy and $-2{,}9{\cdot}10^{-2}$ for precision. This decline is likely due to the limitations of the Naive Bayes algorithm in handling diverse, multi-class data effectively. Also, for binary classification, two distinct groups of precision scores emerge (Fig. 7): one clustering around 95 % and another around 60 %. The latter may result from random data selection or imbalance in the dataset, which adversely affects performance.

Unlike previous studies such as [1, 3], which focused on evaluating machine learning models on homogeneous data, this research explores the impact of data diversity on the Naive Bayes model. The study identifies a notable pattern: the model's efficiency decreases with an increase in data diversity. This insight underscores the trade-off between the simplicity and speed of the Naive Bayes model and its limitations, such as its reliance on all features and the assumption of normal distribution.

All of the above points lead to the point that the Naive Bayes model for IDS machine learning has some strong points, such as:

– simplicity and promptness of implementation;

– low computational requirements make it suitable for systems with limited resources;

– scalability for basic classification tasks.

However, it does possess certain limitations, such as:

– reduced effectiveness in multi-class scenarios due to oversimplified assumptions;

– inability to fully leverage more complex datasets;

– sensitivity to data diversity can result in performance degradation with heterogeneous datasets.

These observations indicate that, although Naive Bayes is effective for lightweight applications, its utility in complex IDS environments is limited.

These findings contribute to the growing body of knowledge on the applicability of the Naive Bayes algorithm for intrusion detection systems (IDS). In general, it is possible to say that Naive Bayes model testing resulted in a wide range of efficiency score values, such as accuracy and precision. This suggests inconsistent work of the model. Therefore, it is imperative to conduct additional research in the areas of data optimization and balancing, as well as the utilization of diverse datasets for the evaluation of Naive Bayes model efficiency.

The findings of this research can be applied to various fields, including network security, cybersecurity, and data science. In the fields of network security and cybersecurity, these insights can be used to develop more robust and resilient IDS. Additionally, within data science, the findings can contribute to the development of novel algorithms and techniques for feature engineering and datasets optimization.

This study has several limitations. First, it evaluates the model using only accuracy and precision metrics, omitting critical measures such as recall, false alarm rate, true negative rate, and F-score. This omission limits the comprehensiveness of the performance evaluation. The research is based solely on the USB-IDS-1 dataset, restricting the generalizability of the findings to datasets of similar characteristics. The reproducibility of results may also vary with changes in data distribution and class imbalance.

One notable disadvantage of this study is the limited scope of metrics used, which could obscure other aspects of the model's performance. This can be addressed by incorporating additional metrics to provide a more holistic evaluation. Another drawback is the focus on a single dataset, which may not fully capture the Naive Bayes model's behavior across different data environments. Future research could extend this study by testing the model on diverse datasets and investigating its adaptability to varying data types.

To address the limitations and disadvantages, future studies could explore combining the Naive Bayes algorithm with other machine learning methods to compensate for its weaknesses. For example, hybrid approaches could balance the algorithm's tendency to misclassify in multi-class scenarios. Additionally, investigating the model's stability under varying data conditions and expanding the scope of evaluation metrics could provide a deeper understanding of its potential for IDS development. Challenges in this development may include optimizing the computational complexity of hybrid models and ensuring the reproducibility of results across diverse datasets.

7. Conclusions

1. Through extensive experimentation, the effects of data volume on the effectiveness of the model were evaluated. Both accuracy and precision scores demonstrate consistent behavior for any size of the training dataset, with accuracy fluctuating between 33 % and 35 % and precision ranging between 40 % and 45 %. For some scenarios, the accuracy score improves significantly up to 38–40 %, suggesting that the model may be more efficient for optimized datasets.

2. The influence of training data variance on the model performance was examined. The best results were observed when classifying a single attack class alongside normative traffic, achieving an accuracy above 95 % with high precision. When multiple attack classes were introduced, performance decreased notably, with accuracy dropping to 30–35 % and precision falling to 40–45 %. This demonstrates the strengths of Naive Bayes in binary classification tasks. However, the model demonstrated significant limitations in multi-class scenarios, requiring significant refinement to improve its scalability and effectiveness for complex datasets.

Conflict of interest

The authors declare that they have no conflict of interest in relation to this research, whether financial, personal, authorship or otherwise, that could affect the research, and its results presented in this paper.

Financing

This research was funded by the Science Committee of the Ministry of Science and Higher Education of the Republic of Kazakhstan (Grant No. AP14869851).

Data availability

Data cannot be made available for reasons disclosed in the data availability statement.

Use of artificial intelligence

The authors confirm that they did not use artificial intelligence technologies when creating the current work.

Acknowledgements

This research has been funded by the Science Committee of the Ministry of Science and Higher Education of the Republic of Kazakhstan (Grant No. AP14869851).

References

1. Ahmad, Z., Shahid Khan, A., Wai Shiang, C., Abdullah, J., Ahmad, F. (2020). Network intrusion detection system: A systematic study of machine learning and deep learning approaches. Transactions on Emerging Telecommunications Technologies, 32 (1). https://doi.org/10.1002/ett.4150
2. Moustafa, N., Slay, J. (2015). UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). 2015 Military Communications and Information Systems Conference (MilCIS), 1–6. https://doi.org/10.1109/milcis.2015.7348942
3. Dwibedi, S., Pujari, M., Sun, W. (2020). A Comparative Study on Contemporary Intrusion Detection Datasets for Machine Learning Research. 2020 IEEE International Conference on Intelligence and Security Informatics (ISI). https://doi.org/10.1109/isi49825.2020.9280519
4. Chatzoglou, E., Kambourakis, G., Kolias, C. (2021). Empirical Evaluation of Attacks Against IEEE 802.11 Enterprise Networks: The AWID3 Dataset. IEEE Access, 9, 34188–34205. https://doi.org/10.1109/access.2021.3061609
5. Jose, J., Jose, D. V. (2023). Deep learning algorithms for intrusion detection systems in internet of things using CIC-IDS 2017 dataset. International Journal of Electrical and Computer Engineering (IJECE), 13 (1), 1134. https://doi.org/10.11591/ijece.v13i1.pp1134-1141
6. Catillo, M., Del Vecchio, A., Ocone, L., Pecchia, A., Villano, U. (2021). USB-IDS-1: a Public Multilayer Dataset of Labeled Network Flows for IDS Evaluation. 2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W), 1–6. https://doi.org/10.1109/dsn-w52860.2021.00012
7. Özsarı, M. V., Özsarı, Ş., Aydın, A., Güzel, M. S. (2024). USB-IDS-1 dataset feature reduction with genetic algorithm. Communications Faculty of Sciences University of Ankara Series A2-A3 Physical Sciences and Engineering, 66 (1), 26–44. https://doi.org/10.33769/aupse.1320795
8. Kasongo, S. M. (2023). A deep learning technique for intrusion detection system using a Recurrent Neural Networks based framework. Computer Communications, 199, 113–125. https://doi.org/10.1016/j.comcom.2022.12.010
9. Zou, L., Luo, X., Zhang, Y., Yang, X., Wang, X. (2023). HC-DTTSVM: A Network Intrusion Detection Method Based on Decision Tree Twin Support Vector Machine and Hierarchical Clustering. IEEE Access, 11, 21404–21416. https://doi.org/10.1109/access.2023.3251354
10. Sammut, C., Webb, G. I. (2010). Encyclopedia of Machine Learning. Springer New York, 1031. https://doi.org/10.1007/978-0-387-30164-8
11. Gushin, I., Sych, D. (2018). Analysis of the Impact of Text Preproccessing on the Results of Text Classification. Young Scientist, 10 (62), 264–266. Available at: https://molodyivchenyi.ua/index.php/journal/article/view/3755
12. Shkarupylo, V., Lakhno, V., Konyrbaev, N., Baishemirov, Z., Adranova, A., Derbessal, A. (2024). Hierarchical model for building composite web services. Journal of Mathematics, Mechanics and Computer Science, 122 (2), 124–137. https://doi.org/10.26577/jmmcs2024-122-02-b10
13. USB-IDS Datasets. Universita Degli Studi del Sannio. Available at: https://idsdata.ding.unisannio.it/datasets.html