ML-trained model and method for blocking dangerous queries*

Tetiana Korobeinikova[1,*,†] and Nazar Kravchuk[1,†]

[1] Lviv Polytechnic National University, 12 Stepan Bandera str., 79000 Lviv, Ukraine

Abstract

This work aimed to develop a trained machine-learning model that automatically detects and blocks dangerous requests in cybersecurity systems. The main classification algorithms were analyzed during the study, such as logistic regression, support vector machine, decision trees, and deep neural networks. For each algorithm, models were trained using a set of network traffic data, which allowed us to automate the process of classifying and detecting dangerous requests in real-time. The models were trained in Matlab based on the Network Traffic Dataset. The study's main results were a comparison of the effectiveness of different algorithms using metrics such as accuracy, F-score, precision, and recall. In particular, the decision tree-based model demonstrated the highest level of accuracy, which ensured the effective detection of known and new types of attacks. In addition, methods of blocking dangerous queries are considered. The supervised learning method accurately detects known threats but requires significant computing resources. Signature-based methods have shown effectiveness in detecting known attacks, but they are unable to detect new, unknown threats. Behavioral analysis and anomaly detection are good at detecting new attacks but can be circumvented by encryption or polyforms. The blacklist and whitelist methods are easy to implement but require regular updates of the lists and are not always effective against new threats. In general, the results obtained confirm the high efficiency of the proposed methods and models in real conditions and the possibility of their integration into modern cybersecurity systems to automate the protection process.

Keywords

vulnerability, data protection, cyberattacks, data classification, selection automation, machine learning, LR, SVM, DT, DNN

1. Introduction

Modern cybersecurity threats require the use of effective methods to detect and block dangerous requests automatically. Machine learning (ML) plays a key role in this process, allowing the creation of automated classifiers to analyze traffic and identify potentially malicious activities. The main classification algorithms include logistic regression (LR), which estimates the probability of a query belonging to a certain class; support vector machine (SVM), which finds the optimal hyperplane for class separation; decision trees (DT), which builds a hierarchical decision-making structure based on rules; and deep neural networks (DNN), which can handle complex dependencies in large amounts of data. However, the effective application of these algorithms requires their software implementation, in particular, the creation of an automated classification model that will accurately identify dangerous queries in real time. In addition, it is important to analyze and improve existing methods of blocking dangerous requests to increase the effectiveness of cyber defense by combining ML approaches with traditional methods.

The goal of this work is to develop a trained model using ML algorithms, which was not implemented in the reviewed works. The tasks were to analyze classification algorithms and develop their architectures, train an automated classification model based on these algorithms, and analyze existing methods for blocking dangerous queries.

*CSDP'2025: Cyber Security and Data Protection, July 31, 2025, Lviv, Ukraine

* Corresponding author.

† These authors contributed equally.

tetiana.i.korobeinikova@lpnu.ua (T. Korobeinikova); nazar.v.kravchuk@lpnu.ua (N. Kravchuk)

0000-0003-2487-8742 (T. Korobeinikova); 0009-0002-1008-4765 (N. Kravchuk)

2. Literature overview

For a better understanding, similar works should be considered. For example, V. Gnatyuk et al. [1] developed a data model to improve the cybersecurity of content management systems, which allows identifying vulnerabilities and taking preventive measures to increase the level of protection. Researchers A. Nafiiev et al. [2] provided an optimized model for malware detection using ML techniques. The study compared several models, of which the model based on binary data representation demonstrated the highest performance in terms of F-score, precision, and recall. Moreover, A. Yanko et al. developed a model for detecting a low-rate denial-of-service attack (DDoS) using ML in software-defined networking [3]. The authors emphasized the importance of online classifiers. They demonstrated the high efficiency of the proposed model, which achieved an average detection rate of 99.7% for normal and DDoS traffic, which outperforms the results of previous models.

On the other hand, A. González Álvarez et al. conducted a study of the impact of optimization on the performance of pre-trained ML models for image classification [4]. It was found that dynamic quantization significantly reduces inference time and energy consumption, making it a good choice for scalable systems, while global model pruning causes high costs due to longer optimization time. As for the work of I.G.A. Mulyawarman et al. in [5], they considered policies for blocking dangerous content on the Internet. Policies for blocking dangerous requests should be sensitive to human rights and not violate the principles of democratic freedoms. In addition, H. Zhao et al. [6] proposed a continual forgetting technique for trained models that effectively removes unwanted information without significantly affecting the rest of the model's knowledge. This is an important solution for maintaining privacy and security, as it allows to keep the model free of dangerous or sensitive data while minimizing the negative impact on the rest of the functionality. At the same time, the authors C. Sun et al. [7] have developed a framework for blocking entity resolution tasks based on pre-trained language models. This approach effectively filters comparisons and speeds up the process of solving entities, demonstrating an advantage in working with textual and impure data.

In his turn, A. Vrincean [8] compared traditional blocking approaches to query processing with newer methods, particularly non-blocking I/O. The author investigated how these methods can be used to improve server efficiency in the context of scalability of query processing, where non-blocking technologies significantly reduce the overhead of creating and maintaining threads, especially with high I/O requirements. The study by J. R. García et al. [9] presents the THREAD architecture, which allows medical data collection while protecting users from dangerous actions. THREAD also allows tracking the origin of data and its use, which is an important aspect of the ethical use of personal information for training ML models in the healthcare industry. Additionally, D. Papathanasiou et al. proposed the MYRTO methodology [10], which allows for optimal placement of IoT data (Internet of Things) in Pervasive Edge Computing ecosystems, considering processing efficiency and latency reduction, which is important when using ML models to process data requests, in cybersecurity. This helps to improve the efficiency of data processing and classification of dangerous requests on different network nodes. Additionally, Brydinskyi et al. [11] conducted a comprehensive comparison of modern deep learning models for speaker verification, highlighting the effectiveness of advanced neural architectures in improving accuracy and robustness, which can be analogously applied to the classification of dangerous queries in cybersecurity contexts. Furthermore, recent works by Shevchuk et al. [12] emphasize the importance of designing secured services for authentication, authorization, and accounting, providing a foundational approach to protecting systems against unauthorized or harmful requests through robust security policies and mechanisms.

3. Initial information that establishes research

The study examined the main classification algorithms used to detect and block dangerous requests automatically in cybersecurity systems. LR, SVM, DT, and DNN are analyzed. For each algorithm, a general description and examples of use in the field of cybersecurity are presented [13], as well as their query classification schemes. As part of the LR analysis, a model was built based on such parameters as Internet Protocol (IP), Hypertext Transfer Protocol (HTTP), and principal component analysis (PCA) [14]. For SVM, the algorithm was tested on a sample of network traffic to determine its effectiveness in detecting attacks [15]. In the case of DT, the accuracy of query classification was tested using the Uniform Resource Locator (URL) as a key classification parameter [16]. For DNN, a multilayer neural network was built, and the model was trained and tested to evaluate its effectiveness in detecting threats [17].

We used Matlab and the Network Traffic Dataset obtained in comma-separated values (CSV) format to implement the automated query classification from the Kaggle platform [18]. The data were pre-loaded, cleaned, and transformed. The training (80%) and test (20%) samples were used to ensure the proper quality of training ML models (LR, SVM, DT, DNN). For DNN, the model architecture was defined, and training was performed with visualization of the training process. After outputting the DNN training result for data classification, the effectiveness of LR, SVM, and DT models was compared. Then, the prediction was performed, and the accuracy of the classification models was evaluated using the accuracy, F-score, precision, and recall metrics.

In addition, the study used methods of blocking dangerous queries [19], their advantages and disadvantages to assess the feasibility of combining approaches [20]. The supervised learning method is implemented using dynamic classifier selection (DCS), where the optimal classifier is selected for each query according to [21]

$$C' = \arg\max_{C_i \in C} confidence(C_i, x) \tag{1}$$

where C' is the optimal classifier selected to process the query; $C = \{C_1, C_2, ..., C_n\}$ is the set of available classifiers; C_i is a separate classifier from the set C; x is the input query to be classified; $confidence(C_i, x)$ is the confidence function in the classifier C_i for the query x, which evaluates how well this classifier is the best for the current query.

For signature blocking, we used query verification against a set of known attacks, which is formalized by [22]

$$S(x) = \begin{cases} 1, \text{if } x \in S \\ 0, \text{else} \end{cases} \tag{2}$$

where x is an input request; S is a set of known attack signatures.

In turn, the method of behavioral analysis and anomalous detection was based on an assessment of the deviation of a new request from the average value of the profile, which is determined by [23]

$$D = \frac{|x_{new} - \mu|}{\sigma} \tag{3}$$

where D is the distance between the new query and the average value of the profile; x_{new} is the value of the new query; μ is the average value of the profile; σ is the standard deviation of the profile. Moreover, the blacklist and whitelist methods were implemented using the following [24]

$$B(x) = \begin{cases} 1, \text{if } x \in W \\ 0, \text{if } x \in B \end{cases} \tag{4}$$

where W is a set of allowed queries (whitelist); B is a set of prohibited queries (blacklist).

4. Analysis of machine learning classification algorithms

As information technology advances, the number of cyberattacks aimed at unauthorized access to confidential data is growing. Information protection is becoming one of the key challenges for cybersecurity systems, requiring the use of innovative approaches to detecting and neutralizing threats. One of the most promising areas is the use of ML methods and algorithms that automate the process of classifying and blocking dangerous requests. In cybersecurity, this helps to improve the accuracy of detecting potential attacks, minimize the risk of false blocking, and adapt algorithms to new types of threats. In this context, developing an effective ML-trained model for analyzing queries and determining their danger is a pressing task that significantly impacts improving information systems' protection level. Choosing an optimal ML classifier to implement an automated mechanism for blocking dangerous queries is important. Modern classification algorithms, such as LR, SVM, DT, and DNN, have different characteristics that affect the performance of a cybersecurity system (Table 1). LR is one of the basic ML algorithms used for binary classification. Due to its mathematical simplicity and effectiveness in cases where the data has linearly separated classes, LR allows you to quickly identify potentially malicious queries and take measures to block them.

Table1
Classification algorithms: description and examples of use in cybersecurity

Algorithm	Description	Example of use in cybersecurity
LR	An algorithm for binary classification that predicts the probability of an object belonging to a certain category based on independent variables.	It is used for basic classification tasks (e.g., identifying suspicious requests as 'attacker' or 'normal').
SVM	A classification method that uses hyperplanes to divide data into classes, capable of working with linearly and nonlinearly separated data.	It is used to classify malicious queries based on large datasets with many features and non-linear relationships.
DT	An algorithm that builds a decision tree for classification, where each node is a certain feature, and the branches are possible values.	It is used to detect malicious requests, where easy allows requests to be blocked.
DNN	A network of neurons with many layers that can learn complex, non-linear patterns in data.	Suitable for detecting complex anomalies in large semantic data sets, such as real-time threat analysis.

The main limitation of LR is its poor ability to handle complex non-linear dependencies between features. However, its use is justified in cases where it is necessary to quickly assess the risk of threats based on key request parameters, such as the source IP address, HTTP request structure, frequency of requests, etc. For a better understanding, consider the process of classifying requests using LR in a cybersecurity system (Figure 1).

The provided scheme starts by processing input data, such as query parameters or user behavior. This data is passed to the model to calculate the probability that the request is malicious or normal. The result is a threat probability that is compared to a predefined threshold. If the calculated probability exceeds this threshold, the request is classified as malicious and is subject to blocking. This model allows you to quickly and accurately detect suspicious requests, reducing the risk of cyberattacks.

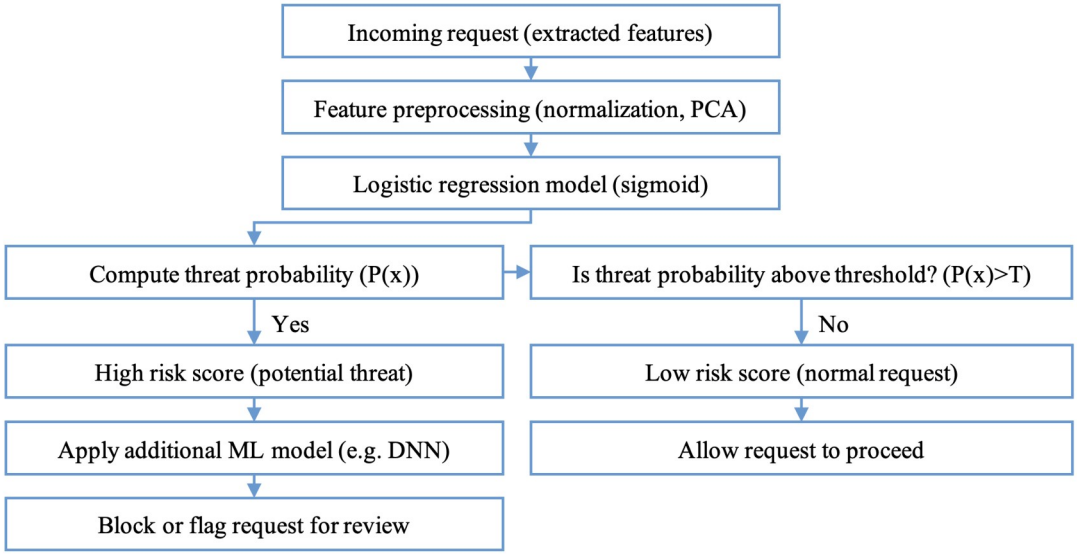

Figure 1: Scheme of query classification using the LR algorithm

On the other hand, SVM algorithm is a powerful classification method that uses hyperplanes to divide data into classes. The advantage of SVM is the ability to efficiently process both linearly and nonlinearly separated data. SVM also has the ability to adapt to changes in the types of attacks and queries, which makes it a reliable tool for protecting information systems from new threats. In the process of classification using SVM, an optimal hyperplane is first constructed to maximize the distance between classes. This allows you to separate malicious queries from normal ones effectively. After that, the query is compared with this hyperplane to determine its class. It is worth considering the classification of queries using the SVM algorithm (Figure 2).

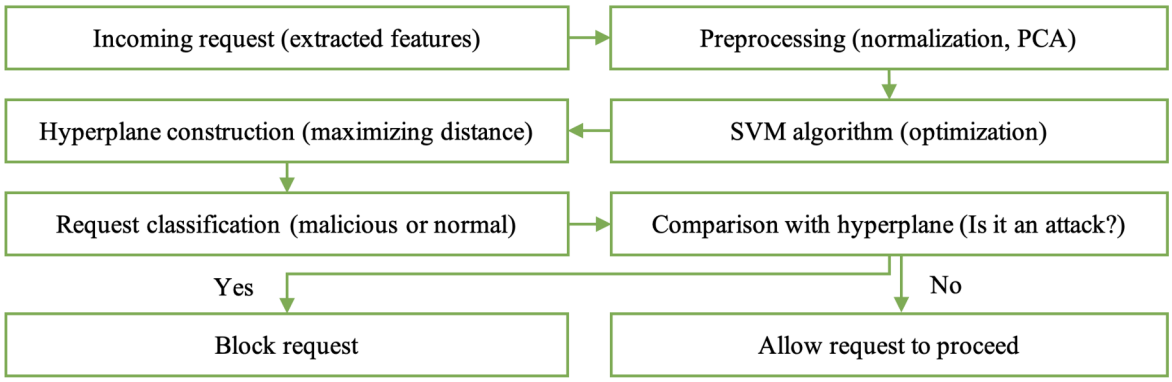

Figure 2: Scheme of query classification using the SVM algorithm

First, the input query with extracted features goes through a preprocessing stage, where the data is normalized and, if necessary, PCA is applied. Next, a hyperplane is constructed to maximize the distance between classes, allowing for a clear query distribution. After that, the query is compared to the hyperplane, and based on this comparison, it is determined whether the query is malicious (attack) or normal. If a malicious request is detected, it is blocked, while a normal request is allowed to be processed further. In turn, the DT algorithm works on the principle of sequentially dividing data into subgroups depending on the values of certain characteristics. This makes it possible to clearly distinguish between normal and malicious queries using a hierarchical decision-making structure. The main advantage of DT is its interpretability, which makes it easy to track the logic of classification and threat identification. The disadvantage may be the tendency to overlearn,

especially if the tree becomes too deep. For a more detailed analysis, consider the classification of requests using DT (Figure 3).

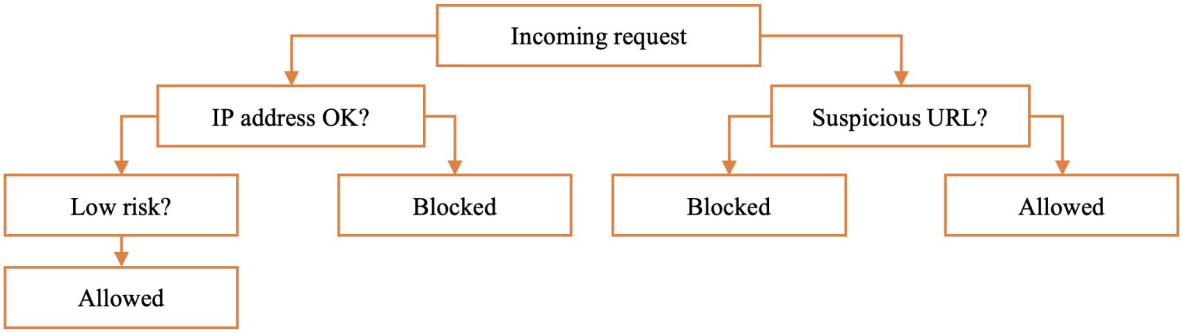

Figure 3: Scheme of query classification using the DT algorithm

The first step is to receive an HTTP request and extract key characteristics from it. If the address is known and safe, the risk level is checked. If the URL is suspicious, the request can be blocked. Finally, if all checks are successful, the request is allowed, otherwise it is blocked. In addition, DNNs can also efficiently analyze large data sets and detect complex, non-linear patterns.

Due to its multi-layered architecture, it can accurately classify cybersecurity requests, adapt to new threats, and minimize false blocking. One of the key advantages of DNNs is the ability to automatically extract features, which allows you to find hidden correlations in the input data. This algorithm can detect complex attacks that are difficult to identify using traditional methods. However, DNN requires significant computing resources and may have problems with the interpretability of solutions, which is an important aspect of cybersecurity systems. To understand how query classification works using DNNs, we should analyze this process (Figure 4).

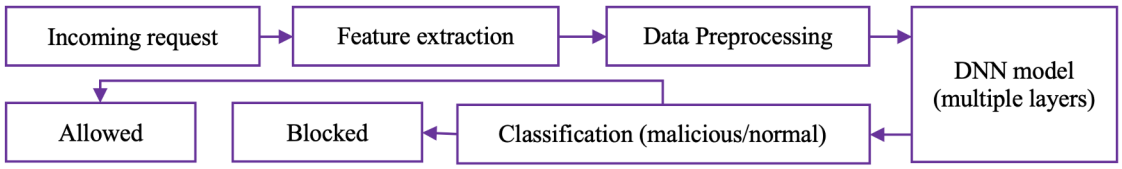

Figure 4: Scheme of query classification using the DNN algorithm

The system receives a request to be verified. After that, the key characteristics of the request are extracted, such as the IP address, URL structure, HTTP headers, request content, etc. Next, the data is normalized and converted into a format suitable for neural network processing. A multilayer model processes data by passing through several hidden layers that are trained on a large sample. DNN determines the probability of a request belonging to a certain class: “malicious” or “normal”. A request is blocked if it is identified as dangerous; if not, it is passed on for further processing.

5. Development and training of an automated classification model

For the practical implementation of automated query classification, it is necessary to implement an effective mechanism for training the ML model. For this purpose, we propose to use the Matlab tool and the *Network Traffic Dataset* [18]. First, the dataset is loaded from a CSV file using *readtable*, while keeping the variable names unchanged (Figure 5). for training and testing the models. Next, the *rmmissing* function removes rows with missing values. The code then checks all variables and, if they are text (cell type), converts them to categorical variables and indexes them to replace text values with numeric values.

```
>> % Load dataset
file = 'C:\Cybersecurity\Midterm_53_group.csv';
data = readtable(file, 'VariableNamingRule', 'preserve');
>> % Remove missing values
data = rmmissing(data);
>> % Convert categorical and text variables to numeric
varNames = data.Properties.VariableNames;
for i = 1:length(varNames)
    if iscell(data.(varNames{i})) % Check if variable is a cell array
        data.(varNames{i}) = grp2idx(categorical(data.(varNames{i}))); % Convert to categorical index
    end
end
>> % Assign random labels for classification (0 - Normal, 1 - Attack)
numSamples = height(data);
data.Class = categorical(randi([0, 1], numSamples, 1), [0, 1], {'Normal', 'Attack'});
>> % Split into training (80%) and testing (20%)
cv = cvpartition(height(data), 'HoldOut', 0.2);
trainData = data(training(cv), :);
testData = data(test(cv), :);
>> % Convert tables to arrays
X_train = table2array(trainData(:, 1:end-1));
Y_train = trainData.Class;
X_test = table2array(testData(:, 1:end-1));
Y_test = testData.Class;
```

Figure 5: Data preprocessing for classification: loading, cleaning, and transforming variables

The next step is to generate random class labels for classification (normal traffic or attack). After that, the data is divided into training (80%) and test (20%) samples using *cvpartition*. Finally, the tables are converted into arrays, where *X_train* and *X_test* contain the features, and *Y_train* and *Y_test* contain the corresponding class labels After data preprocessing, the next step is to train the ML model (Figure 6). Since the amount of data is significant (315309 rows), a subset of 10000 rows is selected for the convenience and speed of data training. To do this, we use the *randperm* function, which generates random indices that are used to select appropriate samples of features (*Xtrain_subset*) and class labels (*Y_train_subset*). Next, three classical models are trained: LR using *fitclinear*, SVM using *fitcsvm*, and DT using *fitctree*. After that, the class labels are converted to the categorical format so that they can be used for deep learning (DL). Finally, the DNN architecture is defined, which consists of an input layer (*featureInputLayer*), two hidden fully connected layers with 100 and 50 neurons, respectively, ReLU activation functions, an output layer with two neurons, a *softmaxLayer* to convert the output values into probabilities, and a *classificationLayer* for classification.

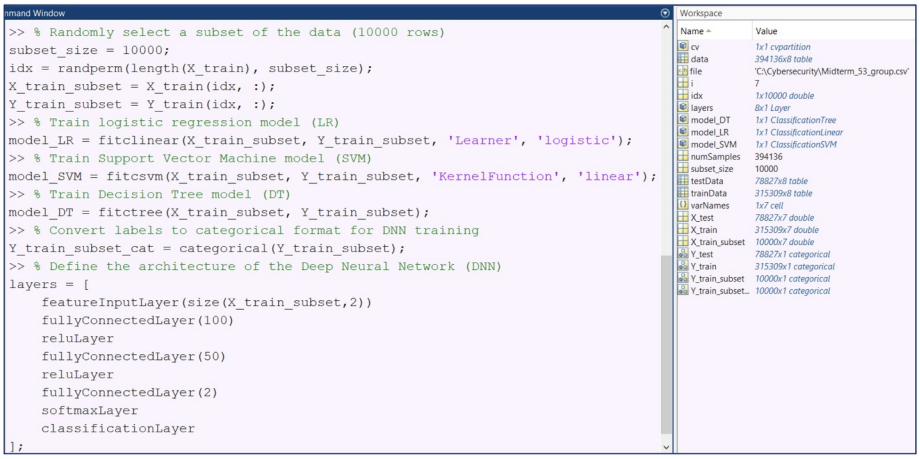

Figure 6: Training ML models and defining DNN architecture

Next, we describe the process of training a DNN for data classification using Matlab, using training options such as the number of epochs (50), mini-batch size (32), and the '*sgdm*' optimization algorithm (Figure 7).

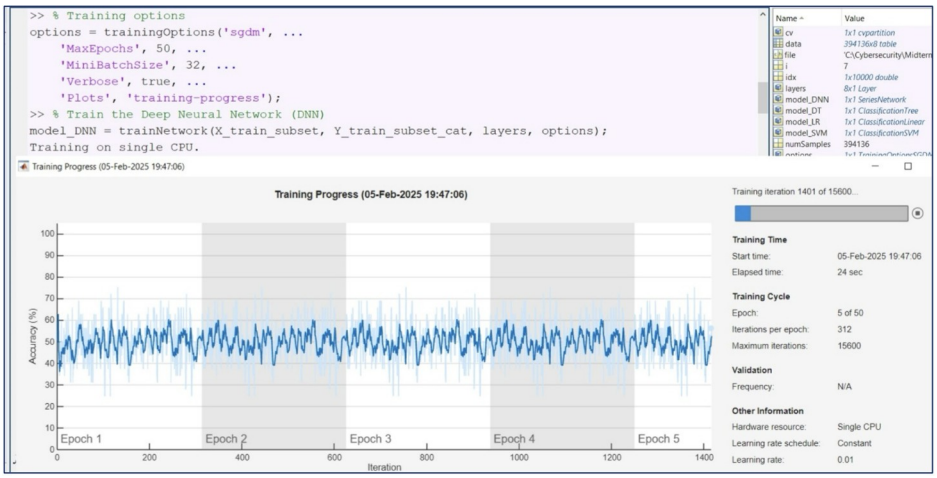

Figure 7: DNN training process for data classification

During training, information on each iteration's accuracy, loss, and time is displayed, allowing you to track the model's progress. As you can see, at iteration 1401 out of 15600, the total elapsed time was 24 seconds, with the model being on the 5th epoch out of 50, and the number of iterations per epoch was 312. This allows us to estimate the speed of learning and the potential need to optimize the model parameters.

As a result, the training process ends when the maximum number of epochs is reached, displaying model performance metrics at each stage of training (Figure 8). The analysis of these metrics allows us to assess the stability and convergence rate of the model, as well as to identify possible over- or under-training problems. Based on this data, optimization parameters such as learning rate, number of neurons in hidden layers, or mini-packet size can be adjusted to improve the model's overall performance.

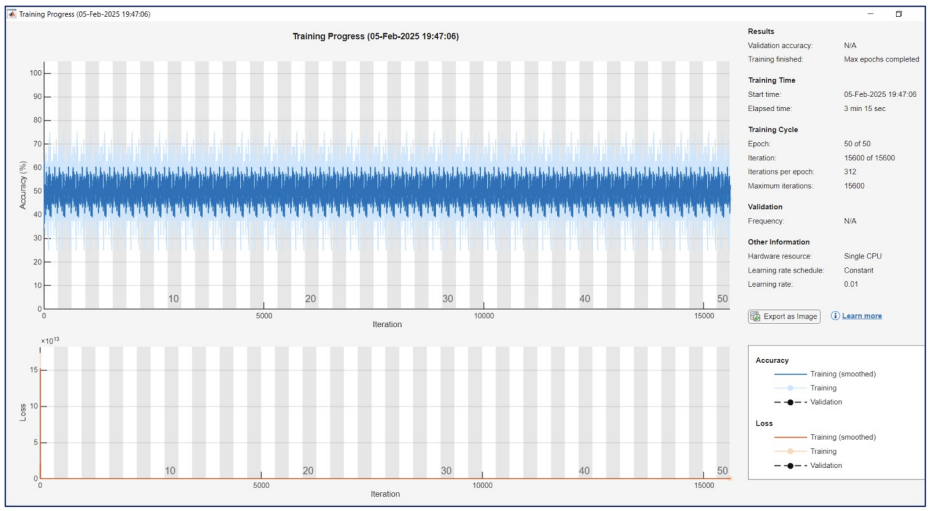

Figure 8: DNN training result for data classification

Next, the four classification models (LR, SVM, DT, DNN) are predicted on the test dataset, and the accuracy of each model is calculated (Figure 9). Accuracy is calculated as the ratio of correct predictions to the total number of test cases. In addition, a confusion matrix is built for the DNN model, which allows you to visually assess the number of correct and incorrect predictions for each class.

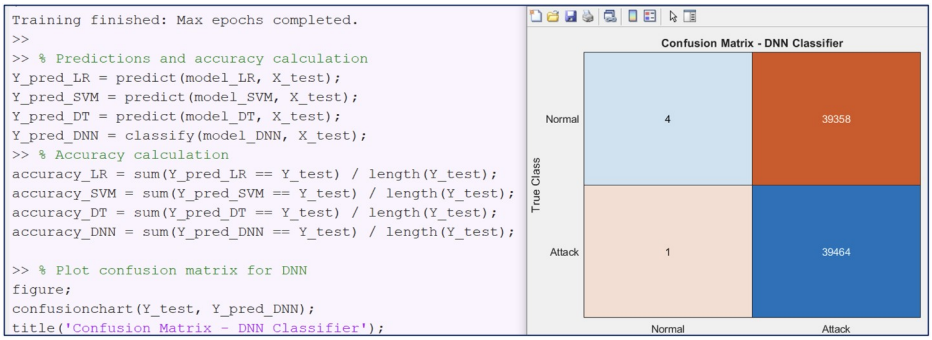

Figure 9: Prediction and accuracy assessment of classification models

The results obtained from the DNN confusion matrix show that the classification model has some problems with correctly recognizing queries of the 'Normal' class, as a significant proportion of such queries were misclassified as 'Attack' (39358 cases). At the same time, the model does a good job of classifying queries of the 'Attack' class, correctly identifying them as 'Attack' (39464 cases). Compared to other methods (LR, SVM, DT), DNN may have better results when configuring the right parameters and architecture, but it can also suffer from the problem of class imbalance. In addition, the accuracy of each model is displayed as a percentage. The results show that all four models (LR, SVM, DT, DNN) have similar accuracies, around 50% (Figure 10).

It is worth noting that the results for LR, SVM, and DT classification models show different efficiencies in classifying normal queries and attacks.

The LR model has a significant number of false positives and false negatives, which indicates difficulties with accurate classification. The SVM model improves accuracy by reducing the number of false positives, but there are still errors. At the same time, DT demonstrates the best accuracy among all models, reducing the number of misclassifications due to its flexibility in class distribution. The confusion matrices can evaluate these results, which distinguish between correctly and incorrectly classified queries as 'Normal' and 'Attack' (Figure 11).

```
>> % Plot confusion matrix for Logistic Regression (LR)
figure;
confusionchart(Y_test, Y_pred_LR);
title('Confusion Matrix - Logistic Regression (LR)');
>> % Plot confusion matrix for Support Vector Machine (SVM)
figure;
confusionchart(Y_test, Y_pred_SVM);
title('Confusion Matrix - Support Vector Machine (SVM)');
>> % Plot confusion matrix for Decision Tree (DT)
figure;
confusionchart(Y_test, Y_pred_DT);
title('Confusion Matrix - Decision Tree (DT)');
>> % Optionally, print accuracy for each model
fprintf('Accuracy of Logistic Regression (LR): %.2f%%\n', accuracy_LR * 100);
fprintf('Accuracy of Support Vector Machine (SVM): %.2f%%\n', accuracy_SVM * 100);
fprintf('Accuracy of Decision Tree (DT): %.2f%%\n', accuracy_DT * 100);
fprintf('Accuracy of Deep Neural Network (DNN): %.2f%%\n', accuracy_DNN * 100);
Accuracy of Logistic Regression (LR): 50.31%
Accuracy of Support Vector Machine (SVM): 49.91%
Accuracy of Decision Tree (DT): 50.02%
Accuracy of Deep Neural Network (DNN): 50.07%
```

Figure 10: Evaluation of the accuracy of classification models: LR, SVM, DT and DNN

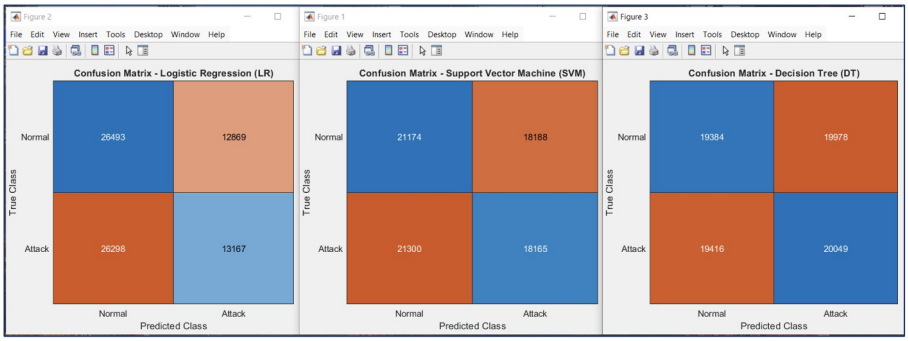

Figure 11: Comparison of LR, SVM, and DT model performance

Thus, LR showed lower efficiency than other models, with many false positive and false negative classifications. This indicates difficulties in accurately recognizing classes, particularly for attacks. LR has limited flexibility in class distribution, which may be the main reason for its errors. On the other hand, SVM has shown improvement over LR by reducing the number of false classifications. However, like LR, the SVM model still makes mistakes when classifying some queries, especially for the Normal class. It does a better job of distinguishing between classes due to the flexible use of the hyperplane. In turn, DT proved to be the most effective among the three models, reducing the number of false classifications due to its ability to adapt flexibly to the data. The DT model demonstrated better accuracy in recognizing both Normal and Attack queries. This may indicate that DT is better suited for solving problems with more complex relationships between classes. In addition, DNN showed similar results to the other methods but was the most susceptible to the problem of class imbalance. Although DNN can achieve better results when configured with the right parameters and architecture, its effectiveness is limited due to the need for DL and the difficulty in choosing the optimal settings.

6. Discussion

The results of this study are focused on the development of a trained ML model for automatically detecting and blocking dangerous requests in cybersecurity systems using LR, SVM, DT, and DNN algorithms. At the same time, P. Bova et al. presented a quantitative model for assessing the dangerous capabilities of artificial intelligence (AI), including mechanisms for early warning of potential AI risks [25]. Both studies use intelligent systems to ensure security, but the current results emphasize practical methods of real-time traffic classification for cyber defense, providing greater accuracy in threat detection, while the work under review is more focused on general strategies to prevent potential AI risks.

Like the work conducted, where ML algorithms such as SVM and LR are used, the study by S. Khan et al. [26] focuses on the classification of user interface errors using these algorithms, as well as text vectorization techniques such as Term Frequency-Inverse Document Frequency (TF-IDF) and Bag of Words (BoW). The best accuracy is achieved with SVM, TF-IDF, and data balancing techniques. In addition, the authors apply Natural Language Processing (NLP) and Random Forest (RF) methods.

The current study also uses SVM and LR, but additionally considers DT and DNN algorithms, which allows achieving higher accuracy and reliability in automating real-time attack detection. This makes it possible to detect both known and new types of threats more efficiently, which is an important aspect of cybersecurity compared to the interface error-oriented methods in the work cited above.

The results of this work are aimed at automating the detection and blocking of dangerous requests in cybersecurity systems with a trained ML model using LR, SVM, DT, and DNN algorithms. The study by U. Ahmed et al. [27] also applied ML and deep learning (DL) algorithms to classify network traffic to improve network security. They tested SVM, k-nearest neighbours

(KNN), RF, DT, long short-term memory (LSTM), and artificial neural networks (ANN), where SVM and RF were found to be effective for use in IDS, and DL models showed high accuracy in recognizing complex attacks. Both studies demonstrate the effectiveness of SVM and DT algorithms in the field of cybersecurity, and the conclusions of this work complement the study by emphasizing the practical application of these algorithms in real-time to block threats automatically.

Unlike this study, which focuses on the ML model and its algorithms, the work of B. Ayyorgun [28] is aimed at using ML to work on edge devices in noisy environments. The main difference is that the current study considers network traffic classification for cyber defense purposes, while the presented work deals with IoT-based physical event analytics. At the same time, both works have a common aspect—the use of ML for real-time and the importance of model optimization. Thus, the results of this work complement the current study, particularly in optimizing ML models for detecting cybersecurity anomalies, which opens up prospects for improving the effectiveness of real-time protection.

The study by G.A. López-Ramírez et al [29] uses ML methods to predict path loss in millimetre wave (mmWave) networks, including LR, ANN, and Extreme Gradient Boosting (XGBoost), improving the prediction accuracy compared to traditional methods. Similarly, the current work also applies the LR algorithm to classify network traffic to detect dangerous requests. The results of both studies confirm the effectiveness of using LR to improve prediction and classification accuracy, although the current work focuses on automating real-time threat detection rather than path loss prediction.

While M. Kim et al. developed ML models to predict the risk of outcomes in medical processes [30] and databases [31] and cloud systems [32], the current work focuses on using ML to train the model to detect and block dangerous queries. The work used the CatBoost model to predict treatment effectiveness, while the current study uses LR, SVM, DT, and DNN algorithms to automate the classification and detection of threats in real-time. The results of both studies confirm the effectiveness of ML in high-precision prediction and classification, but the scope and types of algorithms are different.

The study compared the effectiveness of ML algorithms and methods of blocking dangerous requests, and the DT-based model demonstrated the best results. In turn, the study by A. Aggarwal et al. [33] used ML regression to predict the physical parameters of systems and achieved high accuracy in forecasting. While both studies confirm the effectiveness of ML for providing accuracy in prediction and classification, the current study is distinguished by its focus on real-time detection and blocking of cyber threats. This makes it focused on real-world cybersecurity needs, as opposed to general methods of predicting physical parameters.

In addition to ML algorithms, the findings include an overview of methods for blocking dangerous requests, such as supervised learning, signature methods, behavioral analysis, and anomaly detection, as well as black-and-white lists. At the same time, the study by E. Peixoto et al [34] proposed an environmentally sustainable approach to automating ML processes in dynamic data environments, which involves the reuse of models based on data similarity metrics. This approach reduces the frequency of model retraining without losing performance, which can be useful for cybersecurity, where frequent updates of threat detection models are critical. Thus, this study's results confirm the feasibility of blocking dangerous queries considered in the current work since optimizing the processes of updating models contributes to the efficiency of real-time threat detection.

Similarly to this study, the study by M. D'Orazio et al. [35] uses classification algorithms, in particular LR and SVM. At the same time, the current study additionally uses DT and DNN, while the work uses neural networks (NN) and naïve bayes (NB). Both works are aimed at automating query classification, so they complement each other, demonstrating the versatility of ML methods in different domains. This also highlights the potential of using NLP to analyze text queries and identify potential threats.

Similarly to this study, which focuses on ML and DNN for classifying dangerous queries, J.-J. Hou et al. [36] focuses on DL with an emphasis on convolutional neural networks (CNNs) and recurrent neural networks (RNNs) for recognising dangerous behaviour. The use of DNNs in the current work and CNNs and RNNs in the above study shows that different NNs can be adapted to the tasks of automated recognition of dangerous actions, so these works extend each other, confirming the wide capabilities of NNs.

This study aims to develop methods for blocking dangerous queries, while Z. Su et al. [37] investigate data compression methods for pre-trained models. The results of both works can be complementary, since memory optimisation, as in the above work, can improve the performance of current methods in real-world conditions, especially for resource-intensive algorithms. In addition, the use of compression techniques can be useful for reducing the load on cybersecurity systems.

It is worth noting that the study focuses on the development of an ML model in Matlab using LR, SVM, DT, and DNN classification algorithms. Currently, in the work of Z. Al Shara et al. [38] applied KMeans and Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) clustering algorithms to recover the links between pool requests and issues on GitHub, achieving 91.5% accuracy with BIRCH. Although both works use ML algorithms, the current approach focuses on classification to identify and block dangerous requests, while the other work focuses on clustering. Therefore, these studies complement each other, as classification and clustering are interrelated processes in data analysis. Classification allows you to assign categories to objects based on their characteristics while clustering groups similar objects. It can be useful for further improving classification models or identifying new, previously unknown patterns.

This work focuses on the application of ML algorithms using trained models with indicators of accuracy and adaptability, and K.E. Brown et al. [39] compare large language models (LLM), such as GPT-3.5 (Generative Pre-trained Transformer) and GPT-4, with traditional ML methods. The current work focuses on the effectiveness and stability of using classical ML methods and algorithms to accurately and reliably detect dangerous queries, which is an important aspect for ensuring security in cyberspace. In contrast to the work, this paper demonstrates the advantage in stability and adaptability of such methods, ensuring accuracy in detecting dangerous queries, which is critical for cybersecurity.

Like this work, the study by T. Matyja et al. [40] uses the Matlab environment. This paper discusses business process optimization using simulation methods, where the authors use SimEvents in Matlab/Simulink to model and generate artificial query sequences. They apply ML techniques to transform real data into random data. At the same time, the current work uses Matlab to apply classical ML algorithms, using such functions as fitclinear, fitcsvm, fitctree, and Training Progress. Thus, the results of the presented work confirm the conclusions of the current study, indicating the effectiveness of using ML in query analysis tasks and the convenience of using Matlab for this purpose [41–43].

This study has shown that ML effectively automates the classification of insecure requests, and the methods of supervised learning, signature analysis, behavioral analysis, and black- and whitelists have different efficiencies. In turn, M. Rahimifar et al. presented a method for predicting resource utilization and inference latency of NNs before their implementation on field-programmable gate arrays (FPGAs) [44]. Although this approach is focused on optimizing hardware implementation, both papers emphasize the importance of automated evaluation of ML models to improve their efficiency [45].

Finally, the study demonstrated the effectiveness of DNNs in classifying insecure queries and provided a software implementation in Matlab. H. Bakır [46] proposed TuneDroid, a method for optimizing CNN configurations to improve Android malware detection through code visualization. While TuneDroid focuses on improving the accuracy of CNNs through Bayesian optimization, the current work proposes an approach to automated threat classification in general network traffic, not just for Android.

Thus, the proposed ML model for blocking dangerous requests outperforms the considered approaches due to the combination of accuracy, the ability to detect both known and new threats,

and the optimal balance between performance and computational costs. The combination of different ML methods and algorithms allowed us to achieve high classification accuracy, and the use of a neural network ensured efficiency in detecting anomalies in real-time. Thus, the proposed approach has the potential to be integrated into modern cybersecurity systems, improving their ability to detect and neutralize threats and risks [47, 48] autonomously.

7. Conclusions

This work has shown that ML algorithms such as LR, SVM, DT, and DNN are effective tools for automated classification and blocking of dangerous requests in cybersecurity systems. The analysis showed that DT provides the highest classification accuracy due to its ability to recognize complex patterns in input data. SVM works well with large amounts of data and complex hyperplanes of separation, while LR is fast and easy to implement but inferior to other models in complex scenarios. At the same time, DNN has demonstrated high efficiency in detecting complex attack patterns, but its use requires significant computing resources.

MATLAB modeling based on the Network Traffic Dataset allowed us to evaluate the performance of the selected algorithms. Using the accuracy, F-score, precision, and recall metrics, we confirmed that DTs provide the best balance between computational speed and classification quality, while DNNs have the potential to detect complex and new attacks. The comparison also showed that combining several methods increases the overall effectiveness of a cybersecurity system. The considered methods of blocking dangerous requests allowed us to establish that the use of supervised learning based on DCS improves classification accuracy, as it adapts the choice of algorithm according to the nature of the input data. Signature analysis proved to be effective in recognizing known attacks, but its disadvantage is the inability to respond to new threats. Behavioral analysis and anomalous detection methods provide adaptability but have the risk of false positives. The use of black- and whitelists proved to be the least flexible but is useful as an additional defense mechanism.

Among the study's limitations are the use of only one dataset, which may affect the generalizability of the results, and the high computational complexity of DNN training, making it difficult to apply them in real time. In addition, signature-based attack detection methods demonstrate limited effectiveness against new threats, and behavioral analysis has a risk of false positives.

Further research could focus on expanding and diversifying datasets, optimizing NNs to improve performance, and developing hybrid approaches that combine different classification and blocking methods to achieve greater efficiency in cybersecurity systems.

Declaration on Generative AI

While preparing this work, the authors used the AI programs Grammarly Pro to correct text grammar and Strike Plagiarism to search for possible plagiarism. After using this tool, the authors reviewed and edited the content as needed and took full responsibility for the publication's content.

References

[1] V. Gnatyuk, S. Zozulja, Model of Data for Improving Cybersecurity Content Management System, Infocommun. Comput. Technol. 2(02) (2022). doi:10.36994/2788-5518-2021-02-02-15

[2] A. Nafiiev, D. Lande, Malware Detection Model based on Machine Learning, Bulletin of Cherkasy State Technological University 28(3) (2023). https://bulletin-chstu.com.ua/en/journals/t-23-3-2023/model-viyavlennya-shkidlivogo-programnogo-zabezpechennya-na-osnovi-mashinnogo-navchannya

[3] A. Yanko, A. Prokudin, I. Fil, O. Kruk, Detection of LDDoS Attacks using SDN Networks with Machine Learning Elements, Measuring and Computing Devices in Technological Processes (4) (2024) 287–296. doi:10.31891/2219-9365-2024-80-36

[4] Á. González Álvarez, J. Castaño, X. Franch, S. Martínez-Fernández, Impact of ML Optimization Tactics on Greener Pre-Trained ML Models, arXiv, 2024. doi:10.48550/arXiv.2409.12878

[5] I. G. A. Mulyawarman, P.G.A.S. Yasa, L. Cait, Blocking Dangerous Content in Electronic Communications Networks: Evidence from Netherlands, United States and Singapore, J. Human Rights Culture and Legal System 4(1) (2024) 237–262. URL: https://jhcls.org/index.php/JHCLS/article/view/216

[6] H. Zhao, F. Zhu, B. Ni, F. Zhu, G. Meng, Z. Zhang, Practical Continual Forgetting for Pre-trained Vision Models, in: CVPR 2025, CVF Open Access (2025). URL: https://openaccess.thecvf.com/content/CVPR2024/papers/Zhao_Continual_Forgetting_for_Pretrained_Vision_Models_CVPR_2024_paper.pdf

[7] C. Sun, Y. Jin, Y. Xu, D. Shen, T. Nie, X. Wang, Exploring the Design Space of Unsupervised Blocking with Pre-trained Language Models in Entity Resolution, in: Advanced Data Mining and Applications, Computer Science, 2023, 228–244. doi:10.1007/978-3-031-46661-8_16

[8] A. Vrincean, Optimizing Request Handling using Blocking & Non-Blocking I/O Middleware, Babes-Bolyai Univ., Cluj-Napoca, 2021. https://www.researchgate.net/publication/353103884_Optimizing_request_handling_using_blocking_non-blocking_IO_middleware

[9] J. R. García, J. Favela, C. E. Sánchez-Torres, Traceable Health Data for Consciously Trained ML Models, in: Proc. 15th Int. Conf. Ubiquitous Computing & Ambient Intelligence, 2023, 279–284. doi:10.1007/978-3-031-48642-5_28

[10] D. Papathanasiou, A. Tziouvaras, K. Kolomvatsos, MYRTO: An Efficient Pervasive Method for Hybrid ML-based Data Filtered Allocations, J. Intelligent Information Systems (2024). doi:10.1007/s10844-024-00909-1

[11] V. Brydinskyi, Y. Khoma, D. Sabodashko, M. Podpora, V. Khoma, A. Konovalov, M. Kostiak, Comparison of Modern Deep Learning Models for Speaker Verification, Appl. Sci. (Switzerland) 14(4) (2024) 1329-1–1329-12.

[12] D. Shevchuk, O. Harasymchuk, A. Partyka, N. Korshun, Designing Secured Services for Authentication, Authorization, and Accounting of Users, in: Cybersecurity Providing in Information and Telecommunication Systems II (CPITS-II), 3550, 2023, pp. 217–225.

[13] S. Tavasoli, 10 Types of Machine Learning Algorithms and Models, Simplilearn Solutions, 2025. https://www.simplilearn.com/10-algorithms-machine-learning-engineers-need-to-know-article

[14] S. Chalichalamala, N. Govindan, R. Kasarapu, Logistic Regression Ensemble Classifier for Intrusion Detection System in Internet of Things, Sensors 23(23) (2023) 9583. doi:10.3390/s23239583

[15] Katiyar, G. (2017). Off-Line Handwritten Character Recognition System Using Support Vector Machine. Amer. J. Neural Netw. Appl., 3(2), 22. URL: https://www.sciencepublishinggroup.com/article/10.11648/j.ajnna.20170302.12

[16] D. R. Patil, J. B. Patil, Malicious URLs Detection using Decision Tree Classifiers and Majority Voting Technique, Cybernetics and Information Technologies 18(1) (2018).

[17] A. Subasi, Machine Learning Techniques, in: Practical Machine Learning for Data Analysis Using Python, 2020, 91–202. https://www.sciencedirect.com/science/article/abs/pii/B9780128213797000035

[18] R. Gattu, Network Traffic Dataset, Kaggle, 2024. https://www.kaggle.com/datasets/ravikumargattu/network-traffic-dataset

[19] I. Zhuravel, S. Semenyuk, Stochastic Models for Computer Malware Propagation, in: Proc. 2024 IEEE 17th Int. Conf. Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET), Lviv, Ukraine, 2024, 424–427. doi:10.1109/TCSET64720.2024.10755827

[20] T. Korobeinikova, I. Zhuravel, L. Mychuda, A. Sikora, The Practice of Block Symmetric Encryption for a Secure Internet Connection, in: Computational Intelligence Application Workshop (CIAW), 3861 (2024) 114–122.
[21] J. Brownlee, Dynamic Classifier Selection Ensembles in Python, Machine Learning Mastery, 2021. https://machinelearningmastery.com/dynamic-classifier-selection-in-python/
[22] What is Signature-based Detection?, Corelight Inc., 2025. https://corelight.com/resources/glossary/signature-based-detection
[23] A. Korchenko, Methods for Identifying Anomalous Conditions in Intrusion Detection Systems, Komprint, Kyiv, 2019. https://nubip.edu.ua/sites/default/files/u34/monografiya_korchenko_anna.pdf
[24] C. Kime, Whitelisting vs Blacklisting: How Are They Different?, TechnologyAdvice, eSecurity Planet, 2023. https://www.esecurityplanet.com/applications/whitelisting-vs-blacklisting-which-is-better/
[25] P. Bova, A. Di Stefano, T. A. Han, Quantifying Detection Rates for Dangerous Capabilities: A Theoretical Model of Dangerous Capability Evaluations, arXiv, 2024. doi:10.48550/arXiv.2412.15433
[26] S. Khan, S. Pal, User Interface Bug Classification Model Using ML and NLP Techniques: A Comparative Performance Analysis of ML Models, Int. J. Experimental Research and Review 45 (2024) 56–69. https://qtanalytics.in/journals/index.php/IJERR/article/view/3829
[27] U. Ahmed, et al., Signature-based Intrusion Detection using Machine Learning and Deep Learning Approaches Empowered with Fuzzy Clustering, Sci. Rep. 15 (2025). doi:10.1038/s41598-025-85866-7
[28] B. Ayyorgun, Developing Ultra-lite ML Models for Crash Detection in Noisy Environments, OSF Preprints (2025). doi:10.31219/osf.io/b6zpc
[29] G.A. López-Ramírez, A. Aragón-Zavala, Enhancing Indoor mmWave Communication With ML-Based Propagation Models, IEEE Access 99 (2025). doi:10.1109/ACCESS.2025.10835075
[30] M. Kim, G. Choi, J. Cheon, C. Yoo, J. Koo, Chemotherapy Selection for Advanced or Metastatic Pancreatic Cancer using Machine Learning Models Trained with Multi-Center Datasets, J. Clin. Oncol. 43 (2025) 738–738. doi:10.1200/JCO.2025.43.4_suppl.738
[31] T. Korobeinikova, et al., Web-Applications Fault Tolerance and Autoscaling Provided by the Combined Method of Databases Scaling, in: 12th Int. Conf. Advanced Computer Information Technologies (ACIT), 2022, 27–32. doi:10.1109/ACIT54803.2022.9913098
[32] O. Vakhula, I. Opirskyy, O. Mykhaylova, Research on Security Challenges in Cloud Environments and Solutions based on the “Security-as-Code” Approach, in: CEUR Workshop Proc. 3550, 2023, 55–69.
[33] A. Aggarwal, S. Kumar, S.K. Subbiah, P. Bajpai, Using ML-Supervised Learnings Based-Algorithms to Create a Relative Permeability Model, SPE Caspian Technical Conf. and Exhibition (2024). http://researchgate.net/publication/386139521
[34] E. Peixoto, D. Torres, D. Carneiro, B. M. L. Silva, R. Marques, Reusing ML Models in Dynamic Data Environments: A Data Similarity-based Approach for Efficient MLOps, Preprints (2025). doi:10.20944/preprints202501.1385.v1
[35] M. D’Orazio, G. Bernardini, E. Di Giuseppe, Influence of Pre-Processing Methods on the Automatic Priority Prediction of Native-Language End-Users’ Maintenance Requests through Machine Learning Methods, J. IT in Construction 29 (2024). https://itcon.org/paper/2024/6
[36] J.-J. Hou, B. Zhang, Y. Zhong, W. He, Research Progress of Dangerous Driving Behavior Recognition Methods Based on Deep Learning, World Electric Vehicle J. 16(2) (2025) 62. doi:10.3390/wevj16020062
[37] Z. Su, A. Ahmed, Z. Wang, A. Anwar, Y. Cheng, Everything You Always Wanted to Know About Storage Compressibility of Pre-Trained ML Models but Were Afraid to Ask, Proc. VLDB Endowment 17(8) (2024) 2036–2049. doi:10.14778/3659437.3659456

[38] Z. Al Shara, H. Eyal-Salman, A. Shatnawi, A.-D. Seriai, ML-Augmented Automation for Recovering Links Between Pull-Requests and Issues on GitHub, IEEE Access 99 (2023). doi:10.1109/ACCESS.2023.10015726

[39] K. E. Brown, C. Yan, Z. Li, X. Zhang, B. X. Collins, Y. Chen, E.W. Clayton, M. Kantarcioglu, Y. Vorobeychik, B. A. Malin, Not the Models You Are Looking For: Traditional ML Outperforms LLMs in Clinical Prediction Tasks, medRxiv (2024). doi:10.1101/2024.12.03.24318400

[40] T. Matyja, Z. Stanik, K. Włodkowski, A Method of Generating Customer Requests in a Car Rental Simulation Model, Sci. J. Silesian Univ. of Technology, Series Transport 123 (2024) 191–208.

[41] V. Buhas, et al., Using Machine Learning Techniques to Increase the Effectiveness of Cybersecurity, in: Cybersecurity Providing in Information and Telecommunication Systems, vol. 3188, no. 2 (2021) 273–281.

[42] V. Zhebka, et al., Methodology for Predicting Failures in a Smart Home based on Machine Learning Methods, in: Workshop on Cybersecurity Providing in Information and Telecommunication Systems, CPITS, vol. 3654 (2024) 322–332.

[43] M. Adamantis, V. Sokolov, P. Skladannyi, Evaluation of State-Of-The-Art Machine Learning Smart Contract Vulnerability Detection Method, Advances in Computer Science for Engineering and Education VII, vol. 242 (2025) 53–65. doi:10.1007/978-3-031-84228-3_5

[44] M. Rahimifar, H. Ezzaoui Rahali, A. Corbeil Therrien, Rule4ML: An Open-Source Tool for Resource Utilization and Latency Estimation for ML Models on FPGA, Machine Learning: Science and Technology 6(1) (2025). doi:10.1088/2632-2153/ada71c

[45] V. Zhebka, et al., Methodology for Choosing a Consensus Algorithm for Blockchain Technology, in: Workshop on Digital Economy Concepts and Technologies Workshop, DECaT, vol. 3665 (2024) 106–113.

[46] H. Bakır, A New Method for Tuning the CNN Pre-Trained Models as a Feature Extractor for Malware Detection, Pattern Analysis and Applications 28(1) (2025). doi:10.1007/s10044-024-01381-x

[47] T. Korobeinikova, I. Tachenko, R. Chekhmestruk, P. Mykhaylov, O. Romanyuk, S. Romanyuk, A General Method of Risk Estimation, in: 13th Int. Conf. Advanced Computer Information Technologies (ACIT), Wrocław, Poland, 2023, 410–413. doi:10.1109/ACIT58437.2023.10275626

[48] V. Susukailo, I. Opirsky, O. Yaremko, Methodology of ISMS Establishment Against Modern Cybersecurity Threats, in: M. Klymash, M. Beshley, A. Luntovskyy (eds), Future Intent-Based Networking, Lecture Notes in Electrical Engineering, vol. 831, Springer, Cham, 2022. doi:10.1007/978-3-030-92435-5_15