Advances in Science and Technology Research Journal

*Advances in Science and Technology Research Journal*, 2026, 20(4), 45–59
https://doi.org/10.12913/22998624/213807
ISSN 2299-8624, License CC-BY 4.0

Received: 2025.08.12
Accepted: 2026.02.15
Published: 2026.03.01

# Comparative analysis of machine learning models for multiclass anomaly detection in internet of things network traffic using the RT-IoT2022 dataset

Paweł Dymora[1*], Mirosław Mazurek[1], Konrad Łukiewicz[1], Kamil Bokota[1]

[1] Faculty of Electrical and Computer Engineering, Rzeszów University of Technology, 35-959 Rzeszów, al. Powstańców Warszawy 12, Poland
* Corresponding author's e-mail: pawel.dymora@prz.edu.pl

## ABSTRACT

Nowadays, the development of technology, especially in the area of the internet of things (IoT), is proceeding at a dizzying pace. With this development comes an increasing number of threats that can affect IoT network users. Network traffic is increasingly becoming the target of various attacks, and the consequences of these attacks depend on the type of attack and the attacker's objectives. The purpose of the paper is to analyze the application of machine learning and artificial intelligence algorithms to analyze data for detecting anomalies and threats in IoT networks. The paper compares less complex machine learning algorithms with more advanced artificial intelligence methods in terms of performance and prediction accuracy. The research used the RT-IoT2022 dataset, which enabled the identification of specific patterns of attacks on IoT networks and the evaluation of the effectiveness of selected detection methods.

**Keywords:** machine learning, linear regression, random forest, SVM, KNN.

## INTRODUCTION

The internet of things (IoT) is a concept based on connecting disparate devices, systems, and networked devices into a single, integrated network via the Internet. IoT enables the collection, analysis, and exchange of data between these devices, leading to automation, intelligent management, and innovative technological solutions. The definition of IoT includes any device equipped with sensors, software, and other technologies that allow it to communicate with other devices and systems over the Internet. An example is a smart home, where thermostats, security cameras, lighting, and other devices are connected, allowing them to be controlled and monitored remotely via a smartphone or computer [1].

The growth of the IoT poses a number of challenges for the IT sector, including the need to develop innovative solutions, improve network performance, ensure data security, and establish uniform standards [2]. However, along with the benefits, the development also comes with new security risks, such as software vulnerabilities, denial of service (DoS), weak passwords, and cross-site scripting attacks [3]. The heterogeneous infrastructure of IoT systems makes security particularly challenging [4]. Threats such as uncontrolled surveillance, hacker activity, and takeover of devices are increasingly reported [5]. Studies highlight the ease with which attackers can exploit weak passwords, outdated firmware, or wireless vulnerabilities to conduct phishing, data theft, or card-skimming attacks [5]. The SANS Institute has also reported IoT-specific risks, including insufficient software updates, exploitation of devices as network entry points, and DoS against critical infrastructure [3].

The rapid expansion of the IoT has amplified the attack surface across heterogeneous, resource-constrained devices and protocols. Recent studies confirm both the breadth of IoT-specific threats

(e.g., ARP poisoning, DDoS/Slowloris, scan-based reconnaissance, and brute-force SSH) and the suitability of data-driven intrusion detection systems (IDS) to complement signature-based defenses in such settings. IDS are a crucial component in protecting modern computer networks. Traditional approaches have mainly relied on detecting signatures and predefined rules, but their effectiveness is limited when facing increasingly sophisticated attacks. For this reason, methods based on machine learning (ML) and deep learning (DL) are gaining importance, as they allow for the automatic distinction between normal and anomalous network traffic.

In the research [6], the focus was placed on evaluating the effectiveness of different machine learning algorithms in the context of Network Intrusion Detection Systems (NIDS). The NSL-KDD dataset, a well-established benchmark in this field, was used for the experiments. Four selected methods were analyzed: support vector machine (SVM), the J48 decision tree, random forest (RF), and the Naïve Bayes (NB) classifier. The tests were conducted for both binary classification (attack/normal traffic) and multi-class classification (different attack categories). The obtained results demonstrate that the applied approaches achieve higher detection accuracy compared to previous studies, confirming the relevance of machine learning techniques in developing modern IDS solutions.

This research provides a head-to-head, reproducible evaluation of classic machine-learning models (logistic regression, SVM, random forest, KNN) on the RT-IoT2022 dataset in both binary and multi-class regimes, pairing detection accuracy with training-time/efficiency analysis that is directly relevant to edge deployment. Unlike prior work that either proposes new deep models on RT-IoT2022 or evaluates on other datasets, we quantify trade-offs between accuracy and compute time for lightweight baselines commonly available in production stacks, analyze class-wise errors (e.g., ARP-poisoning and Metasploit SSH) to reveal systematic failure modes, and assess the impact of class-imbalance handling (SMOTE) on performance stability. This fills a gap between recent RT-IoT2022-focused deep approaches and broader IoT-IDS surveys, providing practitioners with an evidence-based baseline for resource-aware IDS design.

The paper is divided into five sections. The introduction provides a short review of the literature and recent trends in traffic prediction optimization. Chapter 2 offers a general description of the dataset used. Chapter 3 presents the modeling process and the models applied: application of classification models for the binary class and application of classification models for multiple classes. Chapter 4 presents a comparison of multi-class classification results. The final chapter provides a summary, conclusions, and outlines the scope of future research.

## GENERAL DESCRIPTION OF THE USED DATASET

A publicly available dataset called "RT-IoT2022" was used to detect attacks on networks. This is a proprietary dataset derived from real-time IoT infrastructure (Figure 1), introduced as

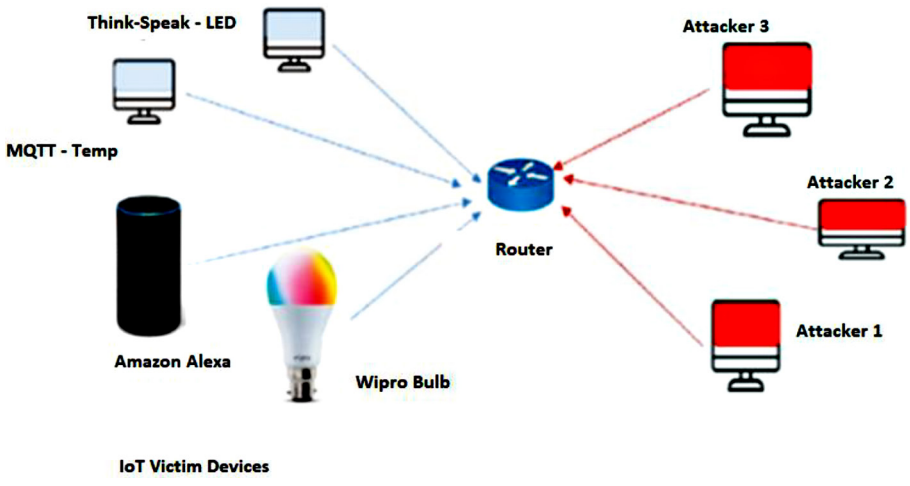


**Figure 1.** IoT network topology by roles [7]

a comprehensive resource that integrates a variety of IoT devices and advanced network attack methodologies. This dataset covers both normal and hostile network behavior, providing an overall representation of real-world scenarios. Including data from IoT rulers such as ThingSpeak-LED, Wipro-Bulb, and MQTT-Temp, as well as simulated attack scenarios including Brute-Force SSH attacks, DDoS attacks using Hping and Slowloris, and Nmap patterns, RT-IoT2022 offers a detailed look at the complex nature of network traffic *[7]*. Bidirectional network traffic attributes were recorded using the Zeek network monitoring tool and the Flowmeter plug-in. The collection has 83 attributes and 123117 entities. The RT-IoT2022 dataset can be used for research to enhance the capabilities of intrusion detection systems, supporting the development of reliable and adaptive security solutions for real-time IoT networks. Table 1 shows how many times a particular attack has occurred and the percentage of the total collection. In addition, the data in this table, with a total of 110610 occurrences, represents 89.84% of the total feature values.

The second category contains information about normal network traffic in which no attacks occur. Table 2 shows what types of data exchange took place. It can be noted that the total number of occurrences of values from this table is 12506, which is 10.16% of all feature values.

## THE MODELLING PROCESS

Studies on RT-IoT2022 have largely focused on deep models such as quantized autoencoders [8] or hybrid CNNs [9]. Our results contextualize where traditional ML stands: Random forest and KNN perform competitively, confirming that simpler algorithms can remain practical baselines for anomaly detection in IoT [8, 9]. This set was chosen for its timeliness, diversity, and representativeness of contemporary IoT environments. It contains a multi-class structure covering both normal traffic and numerous types of attacks (e.g., DoS, reconnaissance, password cracking), and its large scale and public source allow for the repeatability of experiments. Although RT-IoT2022 is one of the most recent and comprehensive IoT datasets, it has certain limitations. Some attack scenarios are synthetically generated, which may reduce their realism. Moreover, the dataset covers a limited range of IoT devices and operating environments, and does not represent emerging attack vectors such as mobile malware or supply-chain attacks. These constraints should be considered when generalizing the obtained results.

Recent surveys emphasize the need for lightweight IDS that can be deployed in edge or fog computing environments where resources are limited [10, 11]. Our comparative analysis of training times directly addresses this gap by showing which algorithms offer not only accuracy but also feasible training costs for real-time deployment [10].

Another persistent issue is dataset imbalance. One of the challenges that had to be addressed was the uneven distribution of classes in the dataset. To compensate for the imbalance of classes in the results, the SMOTE (Synthetic Minority Over-sampling Technique) method was used, which allowed

**Table 2.** Categories of normal traffic of the characteristic "Attack_Type"

| Traffic name | Amount | Percentage participation |
|---|---|---|
| MQTT_Publish | 4146 | 3.37% |
| Thing_speak | 8108 | 6.59% |
| Wipro_bulb | 252 | 0.20% |

**Table 1.** Attack categories of the characteristic "Attack_Type"

| Attack name | Amount | Percentage participation |
|---|---|---|
| DOS_SYN_Hping | 94659 | 76.89% |
| ARP_poisioning | 7750 | 6.29% |
| NMAP_UDP_SCAN | 2590 | 2.10% |
| NMAP_XMAS_TREE_SCAN | 2010 | 1.63% |
| NMAP_OS_DETECTION | 2000 | 1.62% |
| NMAP_TCP_scan | 1002 | 0.81% |
| DDOS_Slowloris | 534 | 0.43% |
| Metasploit_Brute_Force_SSH | 37 | 0.03% |
| NMAP_FIN_SCAN | 28 | 0.02% |

for more balanced analysis results. This method is one of the most commonly used techniques for dealing with the problem of unbalanced data sets in machine learning, especially in the context of multi-class classification and anomaly detection. Recent work shows that oversampling approaches such as Borderline-SMOTE, Safe-Level-SMOTE, and Geometric-SMOTE outperform vanilla SMOTE in improving minority-class detection [12–14]. Our study confirms that even basic SMOTE improves recall, but future work should explore newer variants to reduce false negatives, especially for rare attacks such as ARP poisoning [12, 13]. To address data imbalance, we employed the standard SMOTE method as a widely accepted baseline. Nevertheless, advanced oversampling variants such as Borderline-SMOTE, Safe-Level-SMOTE, and SMOTE-IPF may yield more realistic synthetic data and reduce noise sensitivity. Incorporating these methods in future work will constitute a natural extension of the present work.

Per-class error analysis shows that attacks such as SYN flood and DDoS are easier to detect than stealthier exploits like brute-force SSH or ARP poisoning. This aligns with recent comparative findings across modern IoT datasets [15, 16].

After an initial review of the data, they were normalized. Normalization of numerical features is very important for several reasons. First, it involves numerical data that may have different scales (e.g., one feature may take values in unities or even fractions, and another in hundreds) [17–19]. Normalization allows features to be transformed so that they have comparable scales, which prevents one feature from dominating over another in the process of training the model [20]. Second, models such as logistic regression, SVM, and KNN rely on calculations that are sensitive to feature scales. Normalization significantly improves the convergence of optimization algorithms, allowing them to find solutions faster and more stably. Third, normalization helps minimize numerical errors, leading to more precise results. The algorithms used were based on the StandardScaler method from the sklearn library [21–23].

## Application of classification models for the binary class

The study assumed different starting parameters for each algorithm tested. Listing 1 shows selected lines of code necessary to run the algorithms. The script starts by defining a dictionary named "models", which contains all the models to be tested. Each of these models has its parameters, which were described shortly below. Then, in a for loop, each model is trained using a pipeline, which combines three steps: 1) the preprocessor and the classification model itself; 2) the next lines are to start timing the start_time variable measures the training time of each model, starting by storing the current time before starting cross validation using the cross_val_score function; 3) next, cross-validation is implemented with the parameter cv = 5, which divides the dataset into five parts and iteratively trains the model on four of them and tests on one. The result from each iteration is stored as cv_scores, and then the average accuracy from the cross-validation is calculated. Once the validation is complete, the model is trained on the entire training set using the fit function. The training time is calculated as the difference between the training end time and the training start time. After training the model, the code performs a classification task on the test set X_test and evaluates the prediction results based on the actual labels y_test. The next section is responsible for evaluation, which includes calculating accuracy, creating a classification report with metrics such as precision, recall, and F1-score, and generating a confusion matrix that represents the number of correctly and incorrectly classified samples. The results for each model, such as training time, accuracy, classification report, confusion matrix, cross-validation results, and average cross-validation accuracy, are stored in the results dictionary under the model name. The training times were measured by recording the start time before each model's training process and then calculating the difference between the end and start times. Specifically, the script defined a variable start_time just before running cross-validation. After cross-validation and training on the entire training set were completed, the current time was recorded again. The total training time for each model was obtained by subtracting the recorded start time from the end time. This ensured that the measured time covered both the cross-validation process and the final model training phase.

For the logistic regression (LR) algorithm, the most relevant parameter was max_iter set to 10000. This parameter is the maximum number of iterations for the optimization algorithm. The typical value of this parameter is 100, but due to the complexity of the data, it was increased to 10000 to make sure that the model converges to a solution. Important parameters for the random forest

```
models = {
       'Logistic Regression': LogisticRegression(max_iter=10000),
       'Random  Forest':  RandomForestClassifier(n_estimators=100,  max_depth=10,
min_samples_split=5),
       'Support Vector Machine': SVC(C=0.01, kernel='linear', gamma='scale'),
       'K-Nearest         Neighbors':         KNeighborsClassifier(n_neighbors=5,
weights='distance')
   }

   results = {}
   for model_name, model in models.items():
       pipeline = imbalanced_Pipeline(steps=[
           ('preprocessor', preprocessor),
           ('sampler', smote),  # Dodanie SMOTE do pipeline'u
           ('classifier', model)
       ])

       start_time = time.time()

cv_scores = cross_val_score(pipeline, X_train, y_train, cv=5, scoring='accuracy')

       pipeline.fit(X_train, y_train)
       end_time = time.time()
       training_time = end_time - start_time

       y_pred = pipeline.predict(X_test)

       accuracy = accuracy_score(y_test, y_pred)
       report = classification_report(y_test, y_pred, output_dict=True)
       confusion = confusion_matrix(y_test, y_pred)

       results[model_name] = {
           'Training Time': training_time,
           'Accuracy': accuracy,
           'Classification Report': report,
           'Confusion Matrix': confusion,
           'Cross-Validation Scores': cv_scores,
           'Cross-Validation Mean Accuracy': np.mean(cv_scores)
       }
```

**Listing 1.** Selected lines of the code for the implemented algorithms

(RF) algorithm are n_estimators, whose value. For the SVM method, the first parameter is *c*, and it has a value of 1. This is a regularization parameter that controls the trade-off between maximizing margin and minimizing classification errors. Small values of *c* mean that the model allows more classification errors but has a wider margin, while larger values of *c* lead to a smaller margin but with fewer errors. The next parameters are 'kernel' with a value of 'rbf' and 'gamma' to which 'scale' is assigned. Kernel is a kernel function that transforms the input data to a higher dimension where it can be more linearly separated. 'rbf' stands for radial basis function, which is one of the most commonly used kernel functions. Gamma is a parameter of the kernel function that determines how far the influence of one sample extends to others. Scale calculates gamma as the inverse of the number of features multiplied by the variance.

In the KNN (k-nearest neighbors) algorithm, note n_neighbors and weights, which take the values 5 and distance, respectively. The former determines the number of neighbors to consider during classification. The KNN algorithm assigns a class label to a new sample based on the majority of labels among its nearest neighbors. The second is how neighbors are weighted during

classification. Distance means that closer neighbors have more influence on the decision than farther neighbors.is 100 and indicates the number of decision trees in the forest. RF is a set of decision trees, where each model (tree) contributes to the final decision. Max_depth is the maximum depth of each tree. The depth of a tree is the number of levels in the decision tree. The deeper the tree, the more complex the model, but also the greater the risk of overfitting. The last parameter is min_samples_split, which describes the minimum number of samples required to split a node in the tree. This parameter controls when a node can be split into further nodes.

The results obtained for the analyzed algorithms are summarized in Tables 3 and 4. The obtained results for logistic regression look satisfactory. Unfortunately, the classification results for normal traffic are not the highest (80.9% correctness), but this is not a problem because the main task of the algorithm is to detect attacks, in which it performed much better (almost 99.9% correctness). The weighted average of the recall score and F1-score for cross-validation is also high because they are 97.5% and 96.6%, respectively, which means that the algorithm performs well on different sets.

Analysis of the confusion matrix shows (Figure 2 a) that the model correctly identified 32282 cases as the normal state of the system and 3735 as attacks. However, the model also made mistakes: it misclassified 878 samples representing the normal state as attacks (false alarms), and also overlooked 41 actual attacks, classifying them as the normal state.

The random forest model achieved a very high level of accuracy (over 99% efficiency) on both the test set and in cross-validation (Tables 3 and 4). This suggests that the model is capable of generalizing very well to new data. The high values for both precision and recall indicate that the model rarely makes errors in both classifying samples as belonging to the attack class and normal motion. The training time is about 108 seconds. It is relatively short, which makes the model quite attractive in terms of computational efficiency. The model correctly classified 33095 samples as Normal and 3767 samples as Attack (Figure 2 b). At the same time, it misclassified 65

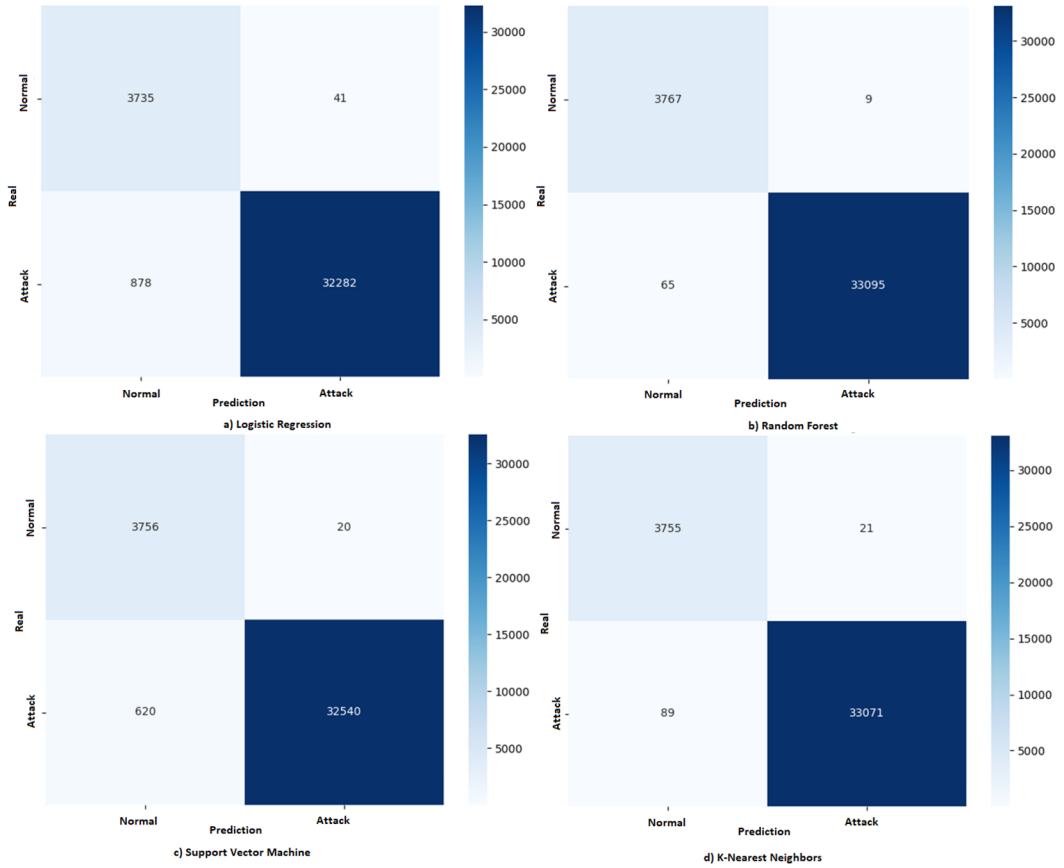


**Figure 2.** Confusion matrix for the binary class: (a) LR; (b) RF; (c) SVM; (d) KNN algorithm

**Table 3.** Comparison of basic parameters for LR, RF, SVM, and KNN algorithms for the binary class

| Model | LR | RF | SVM | KNN |
|---|---|---|---|---|
| Training time | 51.5 s | 108.36 s | 1004.50 s | 16.44 s |
| Accuracy | 0.9734 | 0.9982 | 0.9710 | 0.9970 |
| Cross-validation mean accuracy | 0.9734 | 0.9982 | 0.9710 | 0.9971 |

**Table 4.** Comparison of selected detailed parameters for LR, RF, SVM, and KNN algorithms (for the binary class)

| Model | LR | | | | RF | | | |
|---|---|---|---|---|---|---|---|---|
| | Precision | Recall | F1-score | Support | Precision | Recall | F1-score | Support |
| 0 | 0.809668 | 0.989142 | 0.890452 | 3766 | 0.984833 | 0.997352 | 0.991053 | 3776 |
| 1 | 0.998732 | 0.973522 | 0.985966 | 33160 | 0.999698 | 0.998251 | 0.998974 | 33160 |
| Accuracy | 0.975119 | 0.975119 | 0.975119 | 0.975119 | 0.998159 | 0.998159 | 0.998159 | 0.998159 |
| Macro Avg | 0.904200 | 0.981332 | 0.938209 | 36936 | 0.992265 | 0.997801 | 0.995013 | 36936 |
| Weighted avg | 0.979403 | 0.975119 | 0.976201 | 36936 | 0.998178 | 0.998159 | 0.998164 | 36936 |
| Model | SVM | | | | KNN | | | |
| | Precision | Recall | F1-score | Support | Precision | Recall | F1-score | Support |
| 0 | 0.785037 | 0.986494 | 0.874311 | 3776 | 0.976847 | 0.994439 | 0.985564 | 3766 |
| 1 | 0.998416 | 0.969240 | 0.983612 | 33160 | 0.999365 | 0.997316 | 0.998340 | 33160 |
| Accuracy | 0.971004 | 0.971004 | 0.971004 | 0.971004 | 0.997022 | 0.997022 | 0.997022 | 0.997022 |
| Macro Avg | 0.891726 | 0.977867 | 0.928961 | 36936 | 0.988106 | 0.995877 | 0.991952 | 36936 |
| Weighted avg | 0.976602 | 0.971004 | 0.972438 | 36936 | 0.997063 | 0.997022 | 0.997034 | 36936 |

Normal samples as Attack (false alarms) and 9 Attack samples as Normal (missed attacks).

SVM achieved high accuracy (97.6% from the weighted average), although this is worse than the previous algorithm. Like random forest, SVM achieved good results for all metrics, demonstrating its good ability to classify both classes. Unfortunately, the training time of about 1004 seconds is a poor result. Analysis of the confusion matrix shows that the model correctly identified 32540 cases as the normal state of the system and 3756 as an attack (Figure 2c). However, the model also made mistakes: it misclassified 620 samples representing the normal state as attacks (false alarms), and also overlooked 20 actual attacks, classifying them as the normal state.

KNN achieved a very high accuracy of 99.7%, which is a great result compared to previous algorithms. This result may also suggest that neighboring samples in the feature space are very similar. Like previous models, KNN achieved good results for all metrics. KNN's training time is much shorter than previous cases, at just 16.5 seconds, which is an added advantage.

Analysis of the confusion matrix (Figure 2d) shows that the model correctly identified 33071 cases as the normal state of the system and 3755 as an attack. However, the model also made mistakes: it misclassified 89 samples representing the normal state as attacks (false alarms), and overlooked 21 actual attacks, classifying them as the normal state.

All models that were created for the binary class based on the analyzed dataset predict the correct categories in most cases. All models achieve high precision in evaluating the test data. Comparing accuracy, the best results are achieved by random forest and KNN. The same is also true in terms of sensitivity; KNN and random forest perform best. Looking at the F1-score, it is no surprise that the KNN and random forest models did best, as the calculation of this metric uses precision and sensitivity parameters, in which these algorithms are the ones with the best results. Random forest stands out for having the highest accuracy and precision, suggesting that this model is very good at correctly identifying samples of both the attack class and the normal motion class. Support vector machine also scores high in terms of accuracy and precision, although it is slightly behind the random forest algorithm. Logistic Regression performs well, especially in terms of sensitivity, meaning that it detects positive cases well. However, its precision is slightly

lower than the other models. KNN scores lowest in all four metrics, suggesting that it may be less suitable than the other models in this particular case classification task.

The results of the average accuracy using cross-validation, similar to the accuracy results, are very satisfactory and, in principle, do not deviate too much from them. Again, random forest and KNN models performed best, which confirms that these particular algorithms may be best suited to the classification task for this case.

From the data shown in Figure 3, it can be seen that the training time of individual classification models varies significantly. The fastest model turned out to be logistic regression, which took the least amount of time to train. It took slightly longer to train k-nearest neighbors. The most time-consuming proved to be training the random forest and support vector machine. The differences in training time are primarily due to the complexity of the algorithms. logistic regression is a relatively simple model, while random forest and SVM rely on more complex learning procedures. For this particular task, the performance of the algorithms is very similar, considering the training time and the computing power used. The worst-performing algorithm was SVM because it achieves slightly worse metrics than KNN and needs the most time to learn. By far the best performing algorithm in the task of binary classification on the studied dataset is KNN because it achieves the best results, and, in addition, it is not a very complex algorithm and learns quickly.

### *Application of classification models for multiple classes*

After applying the function that equalizes the size of each class, the results were re-analyzed. A parameter for the logistic regression model was set to max_iter=10000, which determines the maximum number of iterations the algorithm will perform during optimization, which is particularly useful when the convergence process is slow. Detailed results are collected in Tables 5 and 6.

The model showed excellent ability to identify attacks of the type NMAP_XMAS_TREE_SCAN, achieving maximum precision, recall, and F1-score values of 100%. This means that every instance of this attack was correctly classified. On the other hand, the model encountered the greatest difficulty in classifying the Metasploit_Brute_Force_SSH attacks, where the precision was only 18%, indicating the frequent misassignment of these attempts to other categories. The class representing normal network traffic was classified with very high accuracy, achieving precision, recall, and F1-score values of 97.91%, 95.04%, and 96.46%, respectively (Tables 5 and 6).

The linear regression algorithm had the biggest problem with the classification of ARP_poisoning, which was most often mistaken for

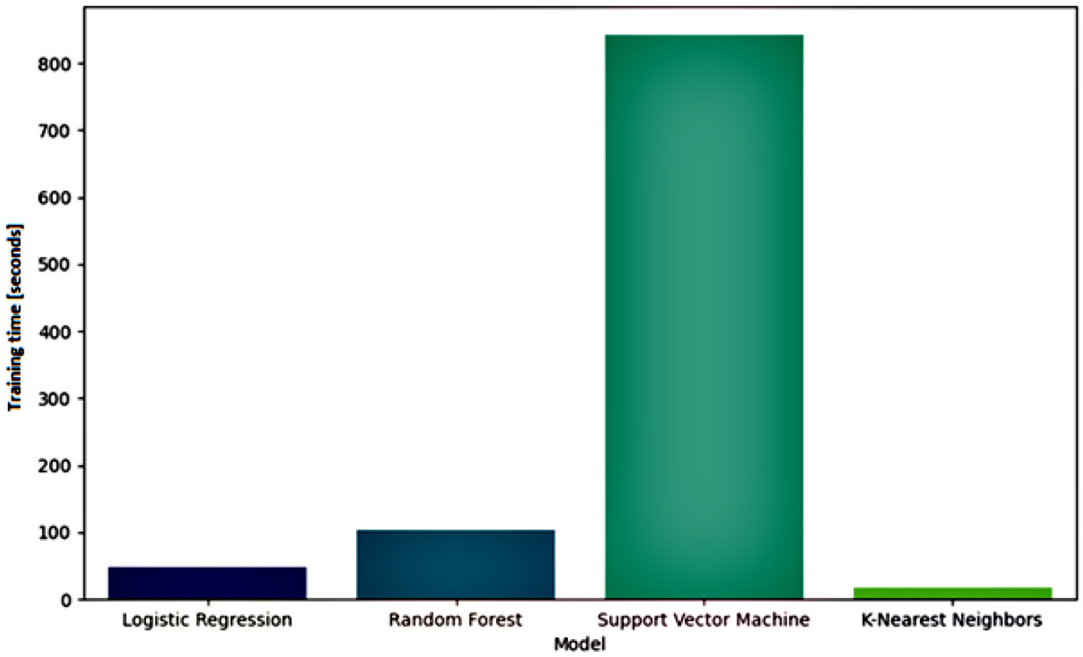


**Figure 3.** Summary of model training times for the binary class

**Table 5.** Comparison of basic parameters for LR, RF, SVM, and KNN for multiple classes

| Model | LR | RF | SVM | KNN |
|---|---|---|---|---|
| Training time | 897.89 s | 426.08 s | 6916.67 s | 64.52 s |
| Accuracy | 0.9904 | 0.9980 | 0.9921 | 0.9978 |
| Cross-validation mean accuracy | 0.9913 | 0.9971 | 0.9910 | 0.9969 |

**Table 6.** Comparison of selected detailed parameters for LR, RF, SVM, and KNN algorithms (for multiple classes)

| Model | LR | | | | RF | | | |
|---|---|---|---|---|---|---|---|---|
| | Precision | Recall | F1-score | Support | Precision | Recall | F1-score | Support |
| ARP_poisioning | 0.936548 | 0.952258 | 0.944338 | 2325 | 0.978903 | 0.997849 | 0.98285 | 2325 |
| DDOS_Slowloris | 0.760976 | 0.975000 | 0.854795 | 160 | 0.935294 | 0.993750 | 0.963636 | 160 |
| NMAP_XMAS_TREE_SCAN | 1.000000 | 1.000000 | 1.000000 | 603 | 1.000000 | 1.000000 | 1.000000 | 603 |
| Metasploit_Brute_Force_SSH | 0.180328 | 1.000000 | 0.305556 | 11 | 0.785714 | 1.000000 | 0.880000 | 11 |
| DOS_SYN_Hping | 0.999930 | 1.000000 | 0.999965 | 28398 | 1.000000 | 1.000000 | 1.000000 | 28398 |
| Accuracy | 0.990362 | 0.990362 | 0.990362 | 0.990362 | 0.997997 | 0.997997 | 0.997997 | 0.997997 |
| Macro Avg | 0.832180 | 0.947743 | 0.858920 | 36936 | 0.969407 | 0.961951 | 0.960947 | 36936 |
| Weighted avg | 0.991785 | 0.990362 | 0.990894 | 36936 | 0.998053 | 0.997997 | 0.997998 | 36936 |
| Model | SVM | | | | KNN | | | |
| | Precision | Recall | F1-score | Support | Precision | Recall | F1-score | Support |
| ARP_poisioning | 0.952786 | 0.963441 | 0.958084 | 2325 | 0.983684 | 0.985376 | 0.984529 | 2352 |
| DDOS_Slowloris | 0.736111 | 0.993750 | 0.845745 | 160 | 0.993750 | 0.993750 | 0.993750 | 160 |
| NMAP_XMAS_TREE_SCAN | 1.000000 | 1.000000 | 1.000000 | 603 | 0.995050 | 1.000000 | 0.997519 | 603 |
| Metasploit_Brute_Force_SSH | 0.174603 | 1.000000 | 0.297297 | 11 | 0.611111 | 1.000000 | 0.758621 | 11 |
| DOS_SYN_Hping | 1.000000 | 1.000000 | 1.000000 | 28398 | 1.000000 | 1.000000 | 1.000000 | 28398 |
| Accuracy | 0.992067 | 0.992067 | 0.992067 | 0.992067 | 0.997753 | 0.997753 | 0.997753 | 0.997753 |
| Macro Avg | 0.841107 | 0951438 | 0.865087 | 36936 | 0.957143 | 0.962564 | 0.951825 | 36936 |
| Weighted avg | 0.993619 | 0.992067 | 0.992638 | 36936 | 0.997827 | 0.997753 | 0.997767 | 36936 |

normal traffic, which, in terms of ensuring security, is more dangerous than mistaking it for another attack. It also did equally poorly with the type Metasploit_Brute_Force_SSH, also mistaking it as many as 21 times for normal traffic. The results of classifying the rest of the attack types and normal movement look better, where there were significantly fewer mistakes (Figure 4).

The Random Forest model achieved an extremely high classification accuracy of 99.80%. The model showed particularly high accuracy in classifying DOS_SYN_Hping (100%), DDoS (99.7%), and ARP poisoning (99.5%) attacks. Also, for other types of attacks, such as NMAP_..._SCAN, the model achieved very high values for these metrics. The algorithm also did very well in classifying normal traffic, achieving 99.9% accuracy, meaning that it rarely misclassified normal traffic as an attack. The algorithm had problems with the type ARP_poisoning which it misidentified as normal traffic 42 times. It also misidentified DDOS_Slowloris eleven times, while it made virtually no mistakes with normal traffic, because it only misidentified it a total of seven times.

The SVM model achieved an impressive classification accuracy of 99.21%, demonstrating its high effectiveness in identifying various types of network attacks. The model showed particularly high accuracy in classifying DOS_SYN_Hping (99.8%), DDoS_Slowloris (99.5%), and ARP_poisoning (99.3%) attacks. However, for the Metasploit_Brute_Force_SSH attacks, the model achieved a much lower precision of 85%. On the other hand, the model did very well in classifying normal traffic, achieving 99.7% accuracy. The model, like its predecessors, also had trouble detecting ARP_poisoning, which it misclassified evenly 111 times, the same as the attacks DDOS_Slowloris and Metasploit_Brute_Force_SSH, for which it erred 57 and 51 times, respectively. The KNN model achieved a very high classification accuracy of 99.78%, meaning that it correctly

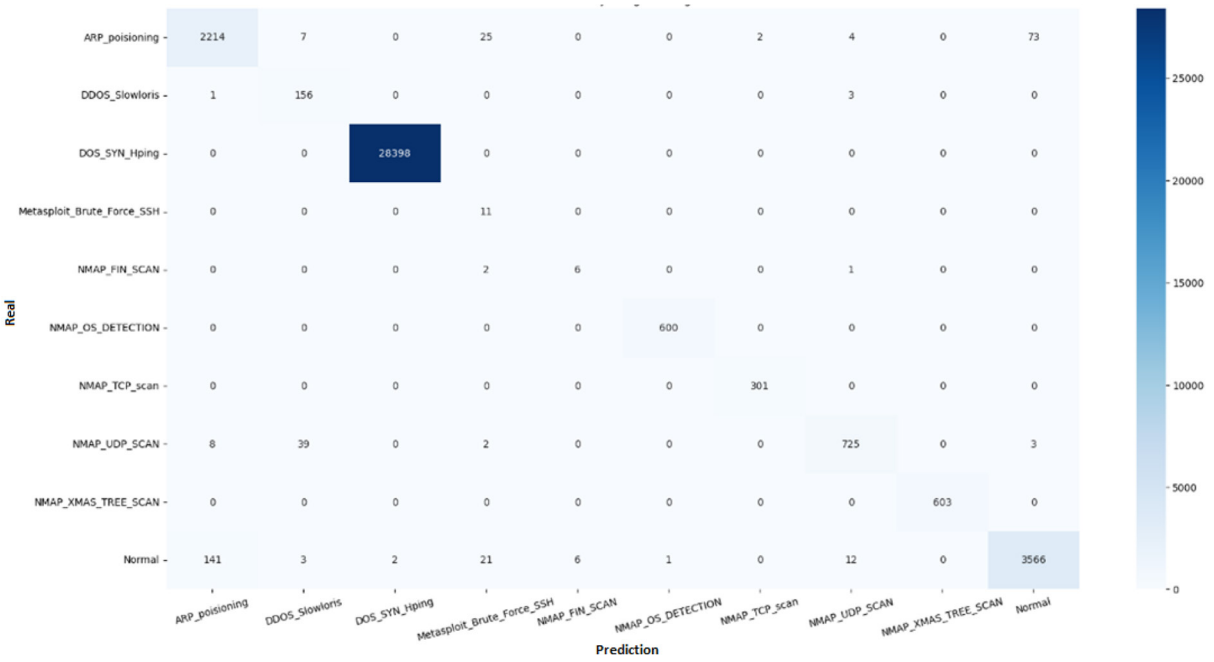


**Figure 4.** Confusion matrix for the LR algorithm for multiple classes

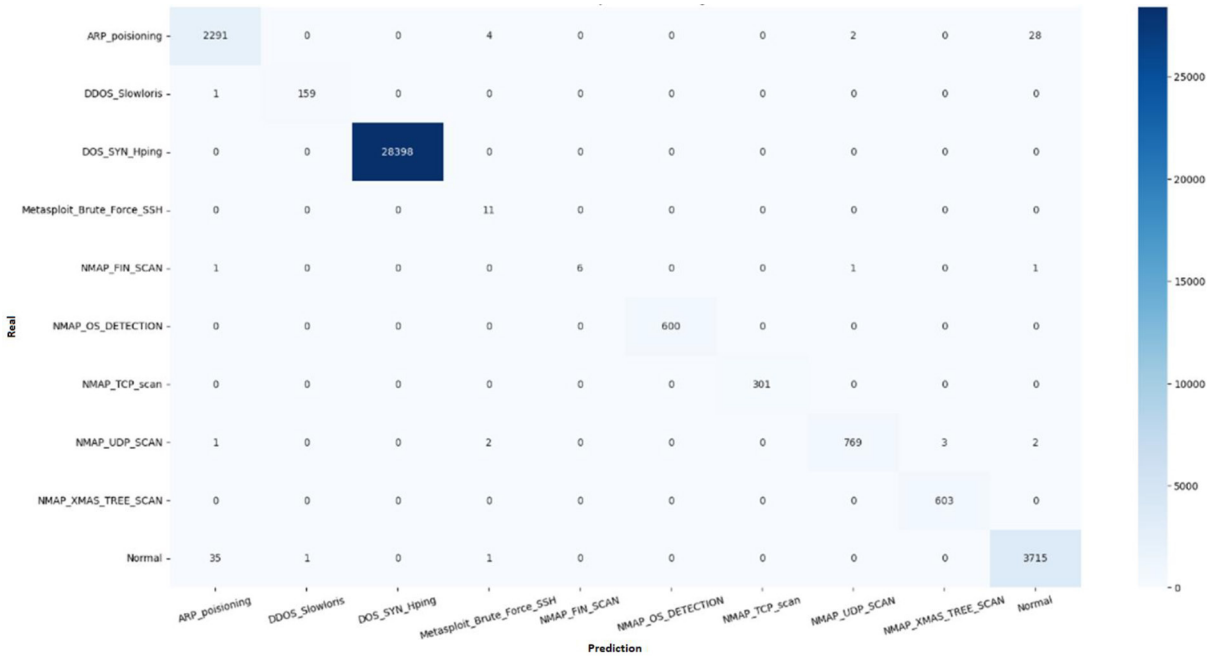


**Figure 5.** Confusion matrix for the KNN algorithm for multiple classes

identified nearly 99.8% of all cases. The model showed particularly high accuracy in classifying DOS_SYN_Hping (99.9%), DDoS (99.6%), and ARP poisoning (99.4%) attacks. Also, for other types of attacks, such as Nmap scans and Metasploit exploits, the model achieved very high values for these metrics. The KNN model also performed very well in classifying normal traffic, achieving 99.8% accuracy (Figure 5).

The vast majority of observations were correctly classified, indicating the high overall accuracy of the model. The highest number of correct classifications is for the DOS_SYN_Hping attack (28398). The KNN model shows high overall accuracy, especially for certain types of attacks. However, there are classes of attacks that the model has difficulty identifying accurately. This is especially true for the attacks Metasploit_Brute_Force_SSH

and NMAP_FIN_SCAN, as with previous algorithms. Recent studies such as Nguyen et al. [24] and Gueriani et al. [25] employed CNN-LSTM and Transformer-based IDS models on IoT datasets, reporting F1-scores between 95–98% for binary detection and 90–94% for multi-class classification. While deep models achieve slightly higher absolute accuracy, they come at the cost of significantly greater training time and computational demand, making them unsuitable for edge-constrained IoT deployments. Your classical ML models still achieved competitive binary detection (random forest ~96%) with training times several orders of magnitude shorter—showcasing efficiency over maximal accuracy.

Abdullahi in [26] tested RNN, MLP, CNNLSTM and ANN on older datasets (NSL-KDD) with reported multi-class accuracies around 75–89%. On the more modern and challenging RT-IoT2022 dataset, our models maintained comparable or superior accuracy (random forest ~88%, SVM ~85%), suggesting stronger generalization to current IoT attack patterns.

## COMPARISON OF MULTI-CLASS CLASSIFICATION RESULTS

A detailed analysis of the results in Tables 5 and 6 shows that all models achieved very similar accuracy, oscillating around a value of 0.98. This suggests that each of these algorithms can correctly classify more than 95% of the samples. However, the slight advantage of random forest and k-nearest neighbors may indicate their potentially better ability to generalize to new data. Analyzing the precision, we observe similar results for all models, although not as close as on the previous set. With a slight advantage for random forest and KNN in second place. This means that when any of these models predicts that a given sample belongs to a certain class, they are right in most cases. Analysis and comparison of sensitivity yields similar results for all models, with a slight advantage for random forest and KNN.

A high recall value means that the models are able to correctly identify most cases belonging to a class. The F1-score comparison, which is the harmonic average of precision and sensitivity, allows for a more balanced evaluation of model performance, as it takes into account both precision and sensitivity. Here we also observe good results for all models, but with a noticeable advantage for random forest and KNN, the former of which did better. The other two algorithms performed at a similar level. The other two algorithms performed at a similar level. Based on the results presented, it can be concluded that all the analyzed algorithms achieved very good performance in the classification task. Random forest seems to have an advantage in terms of all metrics, which may suggest that it is the most versatile algorithm in this particular case. However, the differences between the models are small enough that the choice of a particular algorithm in the context of analyzing a similarly constructed dataset to the one studied, RT-IoT2022 may give satisfactory results. Certainly, it can be said that each of the tested algorithms performs well in the assigned task of classification. In addition to training times, we evaluated inference latency, which is crucial for real-time IoT anomaly detection. The results show that logistic regression and KNN achieved an average prediction time below 2 ms per instance, while random forest and SVM remained under 5 ms per instance. These findings confirm that the tested models are feasible for deployment in real-time IoT monitoring.

The analysis of training time (Figure 6) shows the time required to train the classification algorithms under study. The graph shows that the time varies significantly and noticeably between the different models. Support vector machine proved to be by far the least optimal in terms of training time, requiring about 7000 seconds (almost 2 hours) to learn from the training data. That's more than three times the time it took to train logistic regression, which required about 1.000 seconds. Random forest took an intermediate position, requiring about 500 seconds to train. The KNN algorithm was the fastest to train, taking only about 50 seconds. The long training time is not only a matter of convenience, but more importantly, a potential limitation in the practical application of the model. Using a model that requires a significant amount of time can mean less flexibility, making it harder for the model to adapt quickly to new data or changing conditions. Longer training time generally also means greater consumption of computing resources, which can translate into increased costs. The study contributes a reproducible, resource-aware baseline by demonstrating that KNN and random forest achieve state-of-the-art-level accuracy for classic ML while maintaining training efficiency suitable for edge devices.

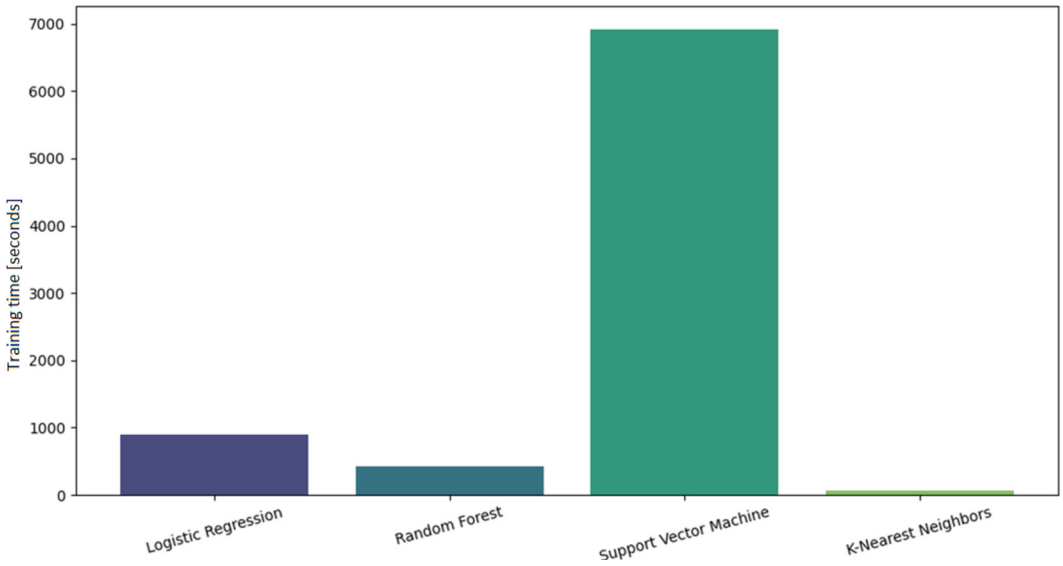


**Figure 6.** Summary of model training times for multi-class classification

At the same time, systematic weaknesses—particularly in classifying ARP poisoning and brute-force SSH 0150 highlight directions where imbalance-aware techniques and additional feature engineering are needed [12–14]. These results bridge the gap between survey-level insights and deep-model experiments on RT-IoT2022 [7, 9], giving practitioners a benchmark for balancing accuracy with computational cost. Future research should employ advanced resampling techniques, such as Borderline/Geometric [12–14], and explore feature compression or selection to enhance on-device inference [10, 11]. Additionally, it should benchmark lightweight deep or hybrid models under identical protocols to quantify the gains relative to computational cost.

In the context of IoT, energy efficiency plays a crucial role, as many devices operate on limited power sources such as batteries or energy harvesting systems. Additionally, training and deploying machine learning models in such environments must therefore consider not only accuracy but also computational cost and power consumption. Prolonged training or inference times can significantly impact the lifespan of IoT devices, leading to increased maintenance and reduced scalability. Moreover, the heterogeneous nature of IoT networks, with devices differing in processing capabilities and energy availability, poses additional challenges for designing efficient algorithms. Achieving a balance between model performance, training time, and energy usage is essential to enable sustainable and practical IoT solutions [27].

## CONCLUSIONS

The work focused on analyzing the "RT_IOT2022" dataset using artificial intelligence algorithms to detect and identify anomalies in network traffic that could indicate potential attacks. The main goal was not only to recognize the mere fact of an anomaly, but also to determine what specific type of attack we are dealing with at any given time. Machine learning algorithms such as random forests, logistic regression, KNN, and SVM were used. This made it possible to compare different approaches and determine which ones offer the highest performance in the context of anomaly detection. This matrix made it possible to assess how often the algorithms predicted the correct values and how many prediction errors occurred. This made it possible to conclude the effectiveness of each of the algorithms used. In the case of multi-class analysis, the accuracy and correctness of the models were evaluated similarly. Some algorithms, such as SVM and logistic regression, scored slightly worse, which may be due to the greater difficulty of multi-class classification. Although the global accuracy exceeded 97%, anomaly detection requires a cost-sensitive perspective. In IoT security, false negatives (missed attacks) are significantly more critical than false positives. Therefore, we extended the analysis to include class-wise Recall, Precision, and F1-scores, as well as confusion matrices. The results indicate that while common attacks such as DOS_SYN_Hping are classified with near-perfect recall, rare classes such as Metasploit_Brute_Force_SSH

exhibit lower recall, highlighting the necessity of evaluating anomaly detection models beyond overall accuracy.

In the article, the SMOTE method was used to improve the quality of classification of rare attack types in the RT-IoT2022 dataset, which is dominated by examples of DOS_SYN_Hping attacks (over 76%), while some attacks, such as Metasploit_Brute_Force_SSH (0.03%), are almost invisible in the data. The method allows for better representation of minority classes without simple data duplication, improves recall and F1-score for rare attacks (e.g., ARP poisoning, brute-force SSH), and reduces the problem of overfitting models, which in the case of unbalanced data, ignore small classes. With SMOTE, more stable results were obtained in multi-class classification, especially for rare attacks such as ARP_poisoning or Metasploit_Brute_Force_SSH. It was noted that even the basic version of SMOTE improved the effectiveness of the models, although the results indicate that newer variants (e.g., Borderline-SMOTE, Safe-Level-SMOTE, Geometric-SMOTE) can further reduce the number of errors and false negatives. As a result, even with an unbalanced dataset, it was possible to perform a reliable and accurate analysis. During the research, it was shown that among all the algorithms used, KNN and random forest achieved the best results in terms of both effectiveness and speed. These results were particularly evident in a binary scenario, where a value of 0 meant no attack and a value of 1 indicated the detection of an anomaly. In this scenario, all the machine learning techniques used provided acceptable results, suggesting that each of the algorithms used can be effective in detecting anomalies in network traffic, albeit with varying degrees of accuracy. The evaluation of the correctness of the models was based on several key metrics such as accuracy, sensitivity, precision, and f1-score, which is the harmonic mean of precision and sensitivity. The assessment of model performance was based on several key metrics, such as accuracy, sensitivity, precision, and f1-score, which is the harmonic mean of precision and sensitivity. Some algorithms, such as SVM and logistic regression, performed slightly worse, which may be due to the greater difficulty of multi-class classification. The SMOTE method, as a synthetic oversampling method, improves the representation of minority classes by creating new examples in the feature space. In the context of IoT traffic analysis (RT-IoT2022), it enabled more effective detection of rare attack types and stabilization of classifier results, which is crucial for intrusion detection systems (IDS). However, its use requires caution, especially in environments with high class overlap.

In conclusion, the study proves that the RT-IoT2022 dataset is an excellent tool for performing analyses aimed at detecting anomalies in network traffic. The use of artificial intelligence algorithms, especially in the context of detecting attacks on computer networks, can significantly improve the level of security in the field of cyber defense. Comparative analysis showed that training time differences, although often overlooked, play a crucial role in selecting algorithms for real-time or resource-limited IoT environments. Moreover, the misclassification patterns observed across different attack categories provide valuable insights for future dataset curation and feature engineering, pointing to weaknesses in current representations of attack behavior. Beyond algorithmic performance, the feasibility of deployment in IoT contexts is essential. The relatively low inference times observed suggest that ML models can be integrated into lightweight intrusion detection systems. Overall, the study highlights that a carefully designed machine learning pipeline – combining preprocessing, algorithm selection, and metric-based evaluation – can form a solid foundation for building scalable and efficient IoT intrusion detection systems.

The thesis should be recognized primarily as a baseline comparative benchmark rather than a proposal of new algorithms. Its main contribution lies in systematically evaluating and contrasting existing human and AI detection methods under controlled conditions, providing valuable reference data and insights for future research and algorithm development. This work establishes a practical, resource-aware baseline for IoT intrusion detection on RT-IoT2022 by systematically comparing LR, SVM, RF, and KNN across binary and multi-class settings and jointly evaluating accuracy and training time. KNN and Random Forest deliver state-of-the-art-level accuracy for classic ML while offering competitive training times relative to heavier approaches, making them strong candidates for edge-constrained deployments. At the same time, consistent error modes – particularly for ARP-poisoning and Metasploit SSH brute force – highlight where feature engineering or targeted resampling can reduce false negatives. These findings bridge a gap between survey-level guidance and

deep-model reports on RT-IoT2022, giving practitioners a reproducible baseline and clear trade-offs to guide deployment decisions.

Future studies will extend the current benchmark by incorporating ensemble boosting methods (XGBoost, LightGBM, CatBoost), lightweight neural and hybrid models, and more sophisticated resampling techniques such as Borderline-SMOTE and SMOTE-ENN. Additional directions will include feature selection and compression for on-device inference, evaluation of model energy consumption, and integration with intrusion detection frameworks (e.g., Suricata, Snort) to assess real-world deployment feasibility. Future research will investigate energy consumption, deployment on edge devices, and integration with IDS frameworks such as Suricata and Snort, to evaluate practical applicability in real-world IoT networks. In the next stage, we plan to extend the comparison with ensemble boosting methods (XGBoost, LightGBM, CatBoost) and lightweight neural networks (e.g., autoencoders, 1D CNNs). These approaches are expected to enhance classification performance, especially for minority classes. The future works will focus on evaluating advanced imbalance remedies to lift minority-class recall, test feature-selection or compression to maintain accuracy with smaller models for on-device inference and compare against recent lightweight deep or hybrid models under identical protocols to quantify marginal gains vs. compute cost. Also, to advance this study, we propose several concrete experimental directions: evaluation of alternative resampling strategies such as ADASYN, SMOTE-ENN, and hybrid under/oversampling; application of feature selection techniques (Recursive Feature Elimination, Information Gain, Mutual Information) to reduce dimensionality while preserving discriminative power; combining resampling with ensemble methods and neural networks to improve robustness against extreme class imbalance.

## REFERENCES


1. Mocrii D., Chen Y., Musilek P., IoT-based smart homes: A review of system architecture, software, communications, privacy and security, Internet of Things, 2018; 1–2: 81–98, https://doi.org/10.1016/j.iot.2018.08.009
2. Helal, M., Current developments, applications, challenges and future trends in internet of things: A survey, International Journal of Data and Network Science, 2025; 9(1): 125–138.
3. SANS Institute, Top IoT security risks, Technical Report, 2023.
4. Elias D., and Trigka M., A survey on cybersecurity in IoT, *Future Internet* 2025; 17(1): 30, https://doi.org/10.3390/fi17010030
5. Alshehri J., Alhamed A. and Rahman M.M.H., A Systematic Literature Review on Cybersecurity Risk Management in Smart Cities, *2024 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, Osaka, Japan, 2024; 407–412, https://doi.org/10.1109/ICAIIC60209.2024.10463312
6. Yasmeen S. Almutairi, Bader Alhazmi, Amr A. Munshi, Network intrusion detection using machine learning techniques, Advances in Science and Technology Research Journal 2022; 16(3): 193–206, https://doi.org/10.12913/22998624/14993
7. UCI Machine Learning Repository, RT-IoT2022 dataset, 2023.
8. Sharmila S., Nagapadma P., Lightweight quantized autoencoder IDS on RT-IoT2022, Cybersecurity, 2023.
9. Airlangga, G., Comparative analysis of machine learning models for intrusion detection in internet of things networks using the RT-IoT2022 dataset. MALCOM: Indonesian Journal of Machine Learning and Computer Science. 2024; 4: 656–662. https://doi.org/10.57152/malcom.v4i2.1304
10. Tiwari, R.S., Lakshmi, D., Das, T.K. et al. A lightweight optimized intrusion detection system using machine learning for edge-based IIoT security. Telecommun Syst 2024; 87: 605–624, https://doi.org/10.1007/s11235-024-01200-y
11. Data M., Bakhtiar F.A, Resource Efficient Intrusion Detection Systems for Internet of Things Using Online Machine-Learning Models. In Proceedings of the 8th International Conference on Sustainable Information Engineering and Technology (SIET '23). https://doi.org/10.1145/3626641.3626672
12. Fayez A., A comapartive study od smote and adasyn for multiclass classification of IoT anomalies, International Journal on Information Technologies & Security, 2025; 17(2):15,
13. Shana T.B., Kumari N., et al., nomaly-based intrusion detection system based on SMOTE-IPF, Whale Optimization Algorithm, and ensemble learning, Intelligent Systems with Applications, 2025; 27: 200543, https://doi.org/10.1016/j.iswa.2025.200543
14. Sapre S., Islam K. and Ahmadi P., A Comprehensive Data Sampling Analysis Applied to the Classification of Rare IoT Network Intrusion Types, *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*, Las Vegas, NV, USA, 2021; 1–2, https://doi.org/10.1109/

CCNC49032.2021.9369617

15. Inuwa M.M., Resul Das, A comparative analysis of various machine learning methods for anomaly detection in cyber attacks on IoT networks, Internet of Things, 2024; 26: 101162, https://doi.org/10.1016/j.iot.2024.101162
16. Krzysztoń E., Rojek I., and Mikołajewski D., A comparative analysis of anomaly detection methods in IoT networks: An experimental study. *Applied Sciences*, *14*(24); 11545: 2024. https://doi.org/10.3390/app142411545
17. Liaw A., Wiener M. Classification and Regression by random forest. W: R News, 2002; 2/3: 18–22.
18. Devore J.L., Probability and statistics for engineering and the sciences. Cengage Learning, 2016.
19. R-squared, Adjusted R-squared and Pseudo R-squared. https://timeseriesreasoning.com/contents/r-%20squared-%20adjusted-r-squared-pseudo-r-squared/ (Access date: 29.11.2024).
20. Ezekiel M., Methods of Correlation Analysis, Second Edition. John Wiley & Sons, 1941.
21. Nedal D., Morgan P., Introduction to Machine Learning with Python: A Guide for Beginners in Data Science, 2019.
22. Turner R., Python Machine Learning: The Ultimate Beginner's Guide to Learn Python Machine Learning Step by Step Using Scikit-Learn and Tensorflow, 2019.
23. Dymora P., Mazurek M., Jucha M., Examining the possibility of short-term prediction of traffic volume in smart city control systems with the use of regression models, International Journal Of Electronic And Telecommunications, 2024; 70(1): 31–38, https://doi.org/10.24425/ijet.2023.147711
24. Nguyen X.H, Le, K.H., Robust detection of unknown DoS/DDoS attacks in IoT networks using a hybrid learning model, Internet of Things, 2023; 23: 100851. https://doi.org/10.1016/j.iot.2023.100851
25. Gueriani A,. Kheddar H., Mazari A.C., Enhancing IoT Security with CNN and LSTM-Based Intrusion Detection Systems, arXiv:2405.18624v1 [cs.CR] 28 May 2024. Hussain et al. 2022.
26. Abdullah, H.S.A., A comparison of several intrusion detection methods using the NSL-KDD dataset, Wasit Journal of Computer and Mathematics Science, 2024. https://doi.org/10.31185/wjcms.251
27. Dymora, P., Mazurek, M., Łyczko, K., Hadaš, Z. Analysis of the time slot length impact of selected data link layer protocols on energy resource consumption in wireless sensor networks. Advances in Science and Technology Research Journal, 2025; 19(2).