Application of Machine Learning to Detect Mirai Infection Attacks on Computer Networks Using the Gotham 2025 Dataset.

[1]Priyanti
Program Studi Rekayasa Sistem Komputer, Fakultas Ilmu Teknik , Universitas Bina Insan
Priyantip298@gmail.com

[2]Bunga Intan
Program Studi Informatika, Fakultas Ilmu Teknik , Universitas Bina Insan
Bungaintan@univbinainsan.ac.id

[3]Sri Tita Faulina
Program Studi Informatika, Fakultas Ilmu Teknik , Universitas Bina Insan
stitabta@gmail.com

Received: 1 January 202X | Revised: 2 February 202X | Accepted: 3 March 202X ("JTI dates" style)

 DOI: https://doi.org/10.32767/jti.v18i2.xxxx

Abstract

Network developments increase the risk of cyberattacks, particularly the Mirai botnet, which performs activities such as scanning, brute force, infection, and volumetric flooding. This study aims to build a machine learning model to classify these attacks using the Gotham 2025 dataset, which contains both benign and attack traffic. Data imbalance is addressed using balancing techniques to prevent model bias using the SMOTE technique. Several algorithms, such as Random Forest and Support Vector Machine (SVM), are used, with evaluations using accuracy, precision, recall, F1-score, confusion matrix, MCC and Kappa. The results are expected to demonstrate an effective model for detecting Mirai attacks and improving network security.

Keywords-component; Computer Networks, , Machine Learning, Gotham2025, SMOTE, Attack Detection.

I. Introduction

The rapid evolution of computer network and internet technologies has led to an increased reliance on network systems across various sectors. However, this development has been accompanied by a rise in cyber threats capable of compromising data confidentiality, integrity, and accessibility. Modern cyber attacks frequently exploit vulnerabilities in network protocols and user identity verification mechanisms to gain unauthorized access to systems. Consequently, network security has become a critical priority that demands serious attention in the management of information technology infrastructure.[1]

One frequently encountered threat is a botnet malware attack, such as Mirai. Mirai infects devices by scanning for those with unsecured Telnet ports and then launching a brute-force attack using default username and password combinations. Once authentication succeeds, the malware infects the device and incorporates it into a botnet controlled via a Command and Control server. Infected devices are not only used to launch attacks—such as Distributed Denial of Service (DDoS)—but also continue scanning to automatically spread the infection to other devices.[2]

Traditional approaches to threat detection and prevention often lack speed and struggle to handle attacks that are constantly evolving and becoming increasingly complex. Machine learning offers a new, faster, and more adaptable approach. This research aims to enhance network security by utilizing machine learning algorithms such as Support Vector Machine (SVM) and Random Forest. This method enables real-time analysis of large datasets, the detection of anomalous patterns, and the identification of previously unseen attacks. A review of the literature indicates that machine learning algorithms can improve threat detection accuracy to up to 95% across the cases studied.[3]

Although numerous studies have applied machine learning to attack detection systems, most prior research still relies on outdated datasets. The majority of studies utilizing machine learning (ML) for network intrusion detection systems employ older datasets such as KDD-CUP99, NSL-KDD, UNSW-NB15, and CICIDS-2017.[4] Literature reviews indicate that this reliance on classic datasets constitutes a major issue in intrusion detection research, as these datasets are considered no longer representative of current network traffic characteristics. This limitation can hinder the ability of machine learning models to adapt to real-world scenarios. Therefore, there is a need for more realistic, complex, and up-to-date datasets that clearly depict modern attacks—such as the Gotham 2025 Dataset.[5]

II. Ease of use

This study employs a quantitative experimental method, testing several machine learning algorithms on the Gotham 2025 dataset. The experiments were conducted to evaluate and compare the capabilities of Support Vector Machine and Random Forest algorithms in detecting attacks, based on specific evaluation metrics. To facilitate this research, the author used several tools.

A. Python

Python is a dynamic programming language, meaning users do not need to specify data types before storing values within a program. Unlike languages such as C++ or Java—which typically require an understanding of memory management or class structures—Python allows beginners to focus on programming logic and fundamental concepts without being distracted by complex technical details; for this reason, I chose to use Python for data processing in this research.[6]

B. Google Colab

Google Colab is a cloud-based service also known as Colab. Based on Jupyter Notebook, Colab enables the application of machine learning and deep learning concepts. It provides free access to GPUs, which are crucial for developing deep learning concepts; these GPUs assist researchers in conducting their work by offering advanced infrastructure for implementing machine learning and deep learning techniques.[7]

C. SMOTE

In this study, the author employs the Synthetic Minority Over-sampling Technique (SMOTE) due to the imbalance in the dataset. SMOTE is a commonly used method for balancing data; compared to other methods, it generates more diverse new samples, thereby helping the model perform better across various situations.[8]

III. Regarding the content

A. Computer networks

Computer networks serve as the fundamental basis for global communication and information exchange in an increasingly digitally connected world. A computer network is an interconnected system that enables devices to share data and communicate with one another. These networks facilitate resource sharing, accelerate data transfer, enhance communication efficiency, and provide remote access to information. The demand for computer networks arises from the need to share resources, expedite data transfer processes, improve communication, and ensure reliable system operation through redundancy mechanisms.[9]

B. Mirai Infection

Mirai is a type of malware that behaves like a worm, capable of spreading automatically by infecting devices and integrating them into a botnet. The Mirai infection process begins with a scanning phase to locate devices with open Telnet services (ports 23 and 2323). Subsequently, Mirai launches a brute-force attack using default username and password combinations to gain access to the device. Upon success, information about the target is sent to a server, and a loader downloads and executes the malware tailored to the device's architecture. Once infected, the malware conceals its activity and takes control of the system, connecting the device to a Command and Control (C&C) server. Infected devices are not only used to launch DDoS attacks but also continue scanning to spread the infection to other devices.[10]

C. Machine Learning on Mirai Infection

This study employs a supervised learning approach, taking into account the characteristics of the data used. According to official documentation from the data provider, the data initially lacked appropriate class labels. Therefore, prior to the supervised learning process, the dataset underwent a labeling phase to assign a category to each data point based on the type of attack being analyzed. The availability of these labels enables the supervised learning process to construct a classification model, in which an algorithm is trained to map input features to explicitly defined attack categories.[11]

D. Random Forest

Decision trees form the foundation of Random Forest (RF), an algorithm belonging to the ensemble learning category. During the training process, the algorithm operates by constructing multiple independent decision trees. For the decision-making process, each node within a tree selects a subset of features at random. Meanwhile, all trees in the forest are trained on data subsets generated via the bagging (bootstrap aggregating) method. This mechanism helps mitigate the overfitting issues often associated with standalone decision trees. Furthermore, it enhances the model's overall stability and accuracy. In the classification process, the final decision is determined based on the majority vote from the individual trees.[12]

E. Support Vector Machine

Support Vector Machines (SVM) originate from statistics, and the fundamental principle of SVM is to find an optimal linear hyperplane in the feature space that maximally separates two target classes. Geometrically, the SVM modeling algorithm identifies an optimal hyperplane with a maximum margin to separate the two classes.[10] Although SVM operates based on linear principles, it has evolved to address non-linear problems.[13]

F. Gotham 2025 Dataset

The Gotham 2025 dataset encompasses various types of attacks—such as DoS attacks, network scanning, Telnet brute force, infections, and CoAP amplification—as well as different stages of command-and-control communication. Data collection was performed in a distributed manner, with network traffic recorded separately for each IoT device in the segment between the IoT gateway and the device itself. The dataset contains approximately 2.6 times more malicious traffic than benign traffic. In total, the dataset comprises 3.13 million network traffic packets. However, the study does not clearly specify the exact proportion of malicious traffic relative to benign traffic. The dataset consists of 23 features, including labels, and is publicly accessible in PCAP and CSV formats.[14]

G. Confusion matrix

A confusion matrix is a table used to classify model results, providing comprehensive information regarding correct and incorrect predictions for each category. Values within the table—such as True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN)—are used to calculate various model evaluation metrics, including accuracy, precision, recall, and F1-score. Accuracy is calculated by dividing the number of correct predictions by the total number of predictions. Precision indicates how well the model identifies truly positive data out of all results the model classifies as positive. Recall indicates the model's effectiveness in detecting all actual positive cases. The F1-score is the harmonic mean of precision and recall, thereby providing a balanced assessment of both aspects.[15]

IV. Prepare your article before styling

Before proceeding to the evaluation stage, the dataset used undergoes several processing steps, as follows:

A. Collection of the Gotham 2025 dataset, comprising normal traffic and Mirai infection attacks.

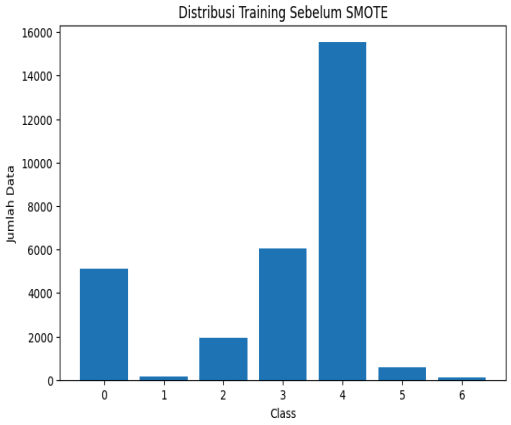
B. Data processing, including the removal of duplicates, invalid data, and missing values, as well as feature scale standardization.

C. Data splitting, allocating 80% for training and 20% for testing to avoid bias during evaluation.

Tabel 1. Gotham2025 Dataset

Kategori Serangan	Packet
Benign	6404
Ingress Tool Transfer	11318
File Download	3613
Mirai C&C Communication	1074
C&C Communication	258
Reporting	250
Merlin C&C Communication	29080

Figure 1 shows the number of datasets per sub-label after data processing.



Gambar 1. Data Distribution Before SMOTE

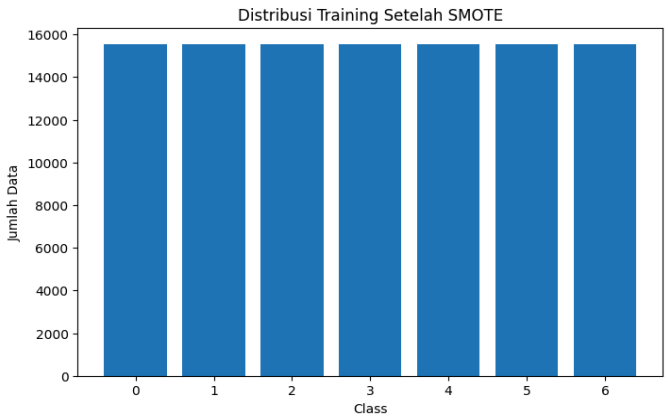
Figure 1 shows the size of the dataset after data processing and splitting. The data split resulted in seven classes. Class 4 contains the largest amount of data, with approximately 15,500 samples. Meanwhile, Class 3 has around 6,000 samples, and Class 0 has about 5,100 samples. Conversely, some classes have significantly fewer samples: Class 2 has around 2,000, Class 5 has about 600, Class 1 has approximately 200, and Class 6 has only about 100 samples.

V. Using the template

After completing the data processing phase, the subsequent steps leading to the evaluation stage are as follows:

A. SMOTE

This study employs the Synthetic Minority Over-sampling Technique (SMOTE) to address the class imbalance issue in the training data. SMOTE works by generating synthetic data for the minority class based on data characteristic proximity (nearest neighbors), thereby making the data distribution across classes more proportional without simply duplicating existing data.



Gambar 2. Data Distribution After SMOTE

Figure 2 shows the distribution of the training data following the application of the Synthetic Minority Over-sampling Technique (SMOTE). The graph reveals that the number of samples across each class has been balanced, with all classes containing a relatively equal amount of data—approximately 15,500 samples. This demonstrates that the SMOTE process successfully resolved the class imbalance issue previously present in the dataset.

B Training Process Using the SVM Algorithm

Following the processes of preprocessing, data standardization, dataset splitting, and data balancing using the SMOTE method, the next stage involves training the model using the Support Vector Machine algorithm. The model training process aims to build a classification model capable of learning data patterns and characteristics from each class, thereby enabling it to make predictions on previously unprocessed data.

TRAINING SVM

```
from sklearn.svm import SVC
import time

# Hitung waktu training
start_time = time.time()

svm_model = SVC(
    kernel='rbf',
    C=10,
    gamma='scale',
    probability=True,
    random_state=42
)

svm_model.fit(X_train_smote, y_train_smote)

svm_train_time = time.time() - start_time

print(f"Training selesai dalam {svm_train_time:.2f} detik")
```

```
Training selesai dalam 46.72 detik
```

Figure 3 Data training process using SVM

Following the completion of data preprocessing, feature standardization, and data balancing using the Synthetic Minority Over-sampling Technique (SMOTE), the next stage involves model training using the Support Vector Machine (SVM) algorithm. This study employs the Radial Basis Function (RBF) kernel, as it is capable of handling nonlinear relationships between features, making it suitable for the complex patterns characteristic of network traffic data. The parameters utilized for the SVM model include kernel = RBF, C = 10, gamma = scale, and probability = True. The C parameter controls the trade-off between maximizing the margin and minimizing classification errors; a value of C = 10 was selected to prioritize minimizing classification errors without significantly compromising the model's generalization capability. Meanwhile, the gamma = scale parameter automatically adjusts the gamma value based on the number of features and data variance, resulting in a more stable training process.

The training process utilized the SMOTE-generated data (X_train_smote and y_train_smote) to ensure a balanced number of samples across all classes. Using SMOTE-generated data aims to mitigate model bias toward the majority class and enhance the model's ability to recognize patterns within minority classes.

Based on the training results shown in Figure 4.8, the SVM model required 46.72 seconds to complete the training process. This duration was influenced by several factors, including the significant increase in data volume resulting from SMOTE compared to the original dataset, the number of features used in the classification process, and the computational complexity of the RBF kernel, which requires calculating the kernel function for every pair of training data points.

```
============================================================
HASIL SVM
============================================================
Accuracy  : 0.9995
Precision : 0.9995
Recall    : 0.9995
F1-Score  : 0.9995
MCC       : 0.9992
Kappa     : 0.9992

Classification Report
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1281
           1       0.97      0.97      0.97        35
           2       1.00      1.00      1.00       485
           3       1.00      1.00      1.00      1513
           4       1.00      1.00      1.00      3884
           5       1.00      1.00      1.00       143
           6       1.00      1.00      1.00        32

    accuracy                           1.00      7373
   macro avg       1.00      1.00      1.00      7373
weighted avg       1.00      1.00      1.00      7373
```

Figure 4 Classification Report SVM

After the Support Vector Machine (SVM) model was trained using the SMOTE-oversampled training data, the next step was to evaluate it against the testing data. Testing was conducted using several metrics—Accuracy, Precision, Recall, F1-Score, Matthews Correlation Coefficient (MCC), and Cohen's Kappa—along with a Classification Report to assess the model's performance for each class. The test results show that the SVM model

achieved an Accuracy of 99.95%, Precision of 99.95%, Recall of 99.95%, and an F1-Score of 99.95%. These values indicate that the model is capable of classifying the test data with a very high level of accuracy. Additionally, the model yielded a Matthews Correlation Coefficient (MCC) of 0.9992 and a Cohen's Kappa of 0.9992. Both metrics are very close to the maximum value (1), demonstrating a high degree of agreement between the predicted results and the actual labels, as well as robust model performance despite the dataset containing multiple classes. Figure 4.12 displays the Classification Report results for each class. Most classes achieved precision, recall, and F1-score values of 1.00, indicating that nearly all data points within those classes were correctly classified. This demonstrates the model's ability to effectively learn the characteristics of each class following the SMOTE data balancing process.

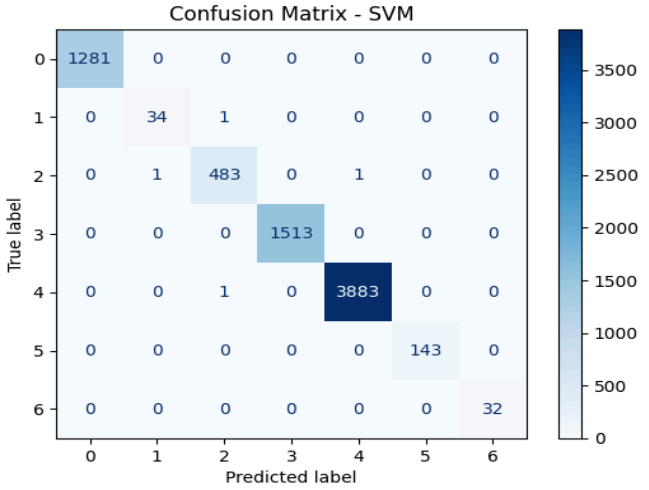


Figure 5 Confusion matrix of the SVM

Figure 5 displays the confusion matrix resulting from classification using the Support Vector Machine (SVM) algorithm. The confusion matrix is used to evaluate the model's ability to classify each class by comparing true labels against predicted labels. Values along the main diagonal indicate the number of data points correctly classified, while values off the diagonal represent instances of misclassification.

Figure 5 shows that the majority of samples lie along the main diagonal, indicating that the SVM model achieves a very high level of classification accuracy. For class 0, all 1,281 samples were correctly predicted with no misclassifications. Similar results were observed for class 3, where all 1,513 samples were correctly classified. Likewise, for classes 5 and 6, all 143 and 32 samples, respectively, were accurately identified by the model. Regarding class 4, out of a total of 3,884 test samples, 3,883 were correctly classified, while one sample was misclassified as class 2. This error is negligible compared to the total number of samples in that class and thus does not significantly impact the model's overall performance.

Misclassification also occurred in Class 1; out of 35 tested samples, 34 were correctly predicted, while one sample was predicted as Class 2. Furthermore, regarding Class 2, out of a total of 485 samples, 483 were correctly identified, whereas one sample was incorrectly predicted as Class 1 and another as Class 4. Figure 4.10 displays the confusion matrix resulting from the classification using the Support Vector Machine (SVM) algorithm. The confusion matrix is used to evaluate the model's ability to classify each class by comparing the true labels against the predicted labels. Values along the main diagonal indicate the number of data points correctly classified, while values outside the diagonal represent the number of misclassifications.

Figure 5 shows that the majority of samples lie along the main diagonal, indicating that the SVM model possesses a very high level of classification accuracy. For Class 0, all 1,281 samples were correctly predicted with no misclassifications. Similar results were observed for Class 3, where all 1,513 samples were correctly classified. Likewise, for Classes 5 and 6, all 143 and 32 samples, respectively, were accurately identified by the model. In Class 4, out of a total of 3,884 test samples, 3,883 were correctly classified, while one sample was incorrectly classified as Class 2. This error is negligible compared to the total number of samples in that class and thus does not significantly impact the model's overall performance. Misclassification also occurred in Class 1; out of 35 tested samples, 34 were correctly predicted, while one sample was predicted as Class 2. Furthermore, regarding Class 2, out of a total of 485 samples, 483 were correctly identified, whereas one sample was incorrectly predicted as Class 1 and another as Class 4.

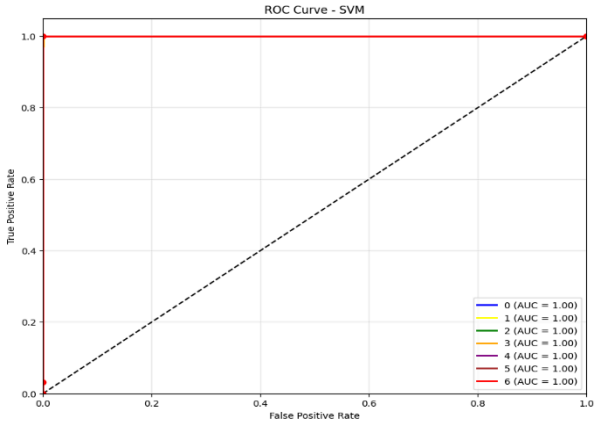


Figure 6 ROC curve of the SVM

Figure 6 shows that all SVM curves are positioned in the upper-left area, with an AUC (Area Under the Curve) value of 1.00 for each class. AUC values range from 0 to 1, where a value closer to 1 indicates superior classification performance. This study involves seven classes: one non-malicious class and six attack-type classes. Although only a single red line is

visible on the graph, there are actually seven ROC curves; they overlap because the model distinguishes between the classes so effectively that the curves share similar shapes. The red line represents the last curve plotted, thereby obscuring the others.

C. Training Process Using the Random Forest Algorithm

The research proceeded with model training using the Random Forest algorithm as a comparative model. Random Forest was selected because it is an ensemble learning algorithm capable of improving classification accuracy by combining multiple decision trees constructed randomly (bootstrap aggregating, or bagging). This approach mitigates the risk of overfitting—a common issue with single decision trees—and yields a more stable model.

The Random Forest model was constructed using 300 decision trees (n_estimators = 300) with a random_state of 42 to ensure reproducibility across experiments. Additionally, the parameter n_jobs = -1 was employed to utilize all available CPU cores, thereby accelerating the computation time. Training was conducted using the SMOTE-processed data (X_train_smote and y_train_smote), which featured a balanced class distribution. By using this oversampled data, each class had a relatively equal number of samples, allowing the Random Forest to learn the characteristics of all classes proportionally without being dominated by the majority class.

Based on the training results shown in Figure 7, the Random Forest algorithm required 65.44 seconds to complete the training process. This duration encompasses the independent construction of 300 decision trees, followed by the aggregation of their individual predictions via a majority voting mechanism to produce the final model. Compared to the Support Vector Machine (SVM) algorithm, which required 46.72 seconds for training, the Random Forest model demanded a longer computation time; this is attributable to the process of building hundreds of decision trees during the training phase. Nevertheless, Random Forest offers advantages such as the ability to handle nonlinear relationships between features, greater robustness against data noise, and the capacity to provide information regarding the importance of each feature used in the classification process.

TRAINING RANDOM FOREST

```
from sklearn.ensemble import RandomForestClassifier

start_time = time.time()

rf_model = RandomForestClassifier(
    n_estimators=300,
    random_state=42,
    n_jobs=-1
)

rf_model.fit(X_train_smote, y_train_smote)

rf_train_time = time.time() - start_time

print(f"Training selesai dalam {rf_train_time:.2f} detik")
```

```
Training selesai dalam 65.44 detik
```

Gambar 7 Training Data Using Random Forest

```
============================================================
HASIL RANDOM FOREST
============================================================
Accuracy  : 1.0000
Precision : 1.0000
Recall    : 1.0000
F1-Score  : 1.0000
MCC       : 1.0000
Kappa     : 1.0000

Classification Report
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1281
           1       1.00      1.00      1.00        35
           2       1.00      1.00      1.00       485
           3       1.00      1.00      1.00      1513
           4       1.00      1.00      1.00      3884
           5       1.00      1.00      1.00       143
           6       1.00      1.00      1.00        32

    accuracy                           1.00      7373
   macro avg       1.00      1.00      1.00      7373
weighted avg       1.00      1.00      1.00      7373
```

Figure 8 Random Forest Classification Report

As shown in Figure 4.13, the Random Forest algorithm achieved an Accuracy of 100%, Precision of 100%, Recall of 100%, and F1-Score of 100%. Additionally, the model yielded a Matthews Correlation Coefficient (MCC) of 1.0000 and a Cohen's Kappa of 1.0000. These values indicate that all test data were correctly classified without any prediction errors.

The Classification Report reveals that all classes achieved precision, recall, and F1-score values of 1.00. This demonstrates that the Random Forest model was able to perfectly identify all samples within each class. The test dataset comprised 1,281 samples for class 0, 35 for class 1, 485 for class 2, 1,513 for class 3, 3,884 for class 4, 143 for class 5, and 32 for class 6; all these samples were accurately predicted according to their true labels.

The macro average and weighted average values, both reaching 1.00, demonstrate consistent model performance across all classes, whether majority or minority. This indicates that the data balancing process using SMOTE positively contributed to the Random Forest model's ability to learn the characteristics of each class equally well.

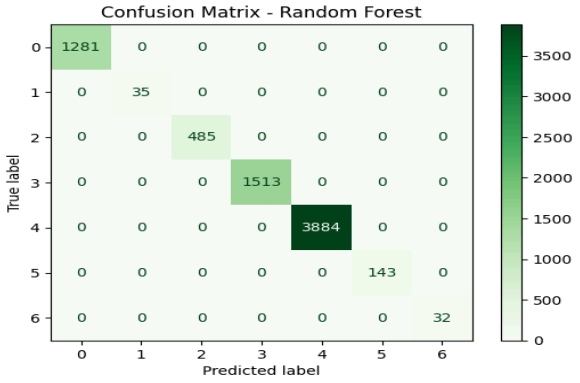


Figure 9 Confusion Matrix Random Forest

Figure 9 displays the confusion matrix resulting from the classification of test data using the Random Forest algorithm. The confusion matrix is used to evaluate the model's ability to classify each class by comparing the actual labels (true labels) with the predicted labels. Values along the main diagonal indicate the number of data points correctly classified, while values outside the diagonal represent the number of misclassifications.

Figure 9 shows that all values lie on the main diagonal, while all off-diagonal elements are zero. This indicates that the Random Forest model successfully classified all test data correctly, without producing prediction errors for any class. Specifically, Class 0 correctly classified all 1,281 samples. Class 1 also successfully identified all 35 samples, while Class 2 correctly classified all 485 samples without error. The same applies to Classes 3, 4, 5, and 6, where all samples—numbering 1,513, 3,884, 143, and 32, respectively—were successfully predicted in accordance with their actual labels.

The absence of values outside the diagonal indicates that no False Positives or False Negatives occurred across any of the classes. In other words, the Random Forest model accurately classified the entire test set of 7,373 samples. These results are consistent with the Accuracy, Precision, Recall, and F1-Score values of 100%, as well as the Matthews Correlation Coefficient (MCC) and Cohen's Kappa values of 1.0000 obtained during the previous evaluation stage.

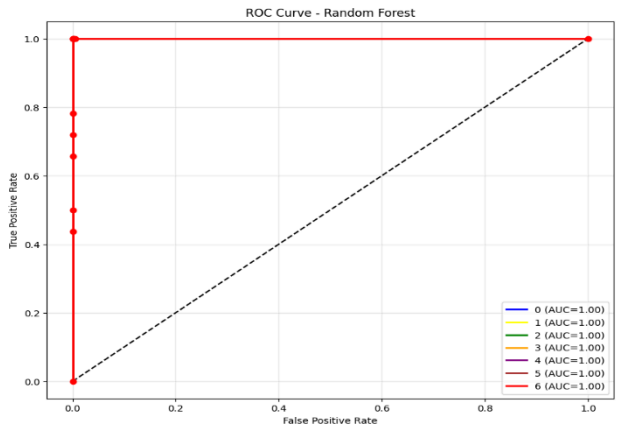


Figure 10 ROC curve of the Random Forest

Figure 10 shows the ROC curve for the Random Forest model, used to assess the model's ability to distinguish between the various classes. This study involves seven classes: one representing non-malicious activity and six representing different types of attacks. An AUC value of 1.00 was achieved for all classes, indicating that the Random Forest model performed exceptionally well in classifying the data. Although only a single red line is visible on the graph, there are actually seven overlapping curves—all sharing the same AUC value—meaning the line drawn last obscures the others.

VI. Citing References in Text

A. The references cited in this study are used to support theoretical concepts, descriptions of malware and datasets, machine learning methods, data preprocessing techniques, and evaluation methods. The citation style follows a numerical format, in which references are presented using numbers enclosed in square brackets, such as [1], [2], and [3].

B. In the Introduction section, reference [1] is used to support the discussion of network security and the increasing risks associated with cyber threats. Reference [2] is used to support the explanation of Mirai malware, including its infection mechanism, brute-force attacks, botnet formation, and Command and Control communication. Reference [3] supports the discussion of machine learning for network threat detection, while references [4] and [5] support the discussion of existing intrusion detection datasets and the need for more recent datasets such as the Gotham 2025 Dataset.

C. References [6], [7], and [8] are used to support the tools and techniques applied in this research. Reference [6] supports the description of Python as the programming language used for data processing, reference [7] supports the description of Google Colab as the computational environment, and reference [8] supports the use of the Synthetic Minority Over-sampling Technique (SMOTE) for handling imbalanced data.

D. In the theoretical background, reference [9] supports the definition and general concept of computer networks. Reference [10] supports the explanation of the Mirai infection process, while reference [11] supports the supervised machine learning approach used for attack classification. Reference [12] is used to support the theoretical explanation of the Random Forest algorithm, including decision trees, bootstrap aggregation, random feature selection, and majority voting. References [10] and [13] are used in the discussion of Support Vector Machine (SVM) and its ability to classify data using an optimal hyperplane and handle nonlinear problems.

E. Reference [14] is used to support the description of the Gotham 2025 Dataset, including its attack categories, data collection process, number of packets, features, and available data formats. Reference [15] supports the explanation of the confusion matrix and classification evaluation metrics, including accuracy, precision, recall, and F1-score.

F. References are provided when information, concepts, methods, or descriptions are obtained from external sources. However, the experimental results generated directly from this study, such as the number of samples after preprocessing, SMOTE results, training time, classification accuracy, precision, recall, F1-score, MCC,

Cohen's Kappa, confusion matrix results, and ROC-AUC values, are considered original results and therefore do not require external citations.

G. All references cited in the text should correspond to entries in the reference list. Likewise, references included in the reference list should be cited within the article. This ensures consistency between the citations and the reference list

VII. ADDITIONAL CITATION RULES

Peneliti (Tahun)	Algoritma	Dataset	Hasil Penelitian
Belarbi ebal. (2025) – *Gotham Dataset 2025*	Analisis dataset & *preprocessing* trafik jaringan	Gotham Dataset 2025	Dataset Gotham 2025 menyediakan data trafik jaringan IoT yang lengkap dan beragam, termasuk serangan *brute force* berbasis *Telnet,* sehingga layak digunakan sebagai dataset penelitian deteksi serangan jaringan.
Hamza et al. (2024) – *Detecting Brute Force Attacks Using Machine Learning*	*SVM,DecisionTree, Random Forest*	Dataset trafik jaringan dan honeypot	Random Forest menunjukkan akurasi dan stabilitas tertinggi dalam mendeteksi serangan *brute force*, sedangkan SVM memberikan performa baik pada data yang telah melalui proses seleksi fitur.
Wahono et al. (2024) – *Brute Force Detection System Based on Machine Learning*	Decision Tree, Random Forest, Naïve Bayes	Dataset log jaringan	Hasil penelitian menunjukkan bahwa *Decision Tree dan Random Forest* bisa klasifikasikan serangan *brute force* dengan tingkat akurasi di atas 80% pada data log jaringan.
Manurung et al. (2025) – *Brute Force Attack Detection Using Machine Learning*	*Random Forest dan SVM*	Dataset IOT	*Random Forest* menghasilkan nilai *F1-score* lebih tinggi dibandingkan SVM pada dataset IoT dengan trafik tidak seimbang.

Previous research indicates that Random Forest performs well in detecting brute-force attacks and IoT network traffic. Studies by Hamzah et al. (2024) demonstrate that Random Forest exhibits stable performance, while research by Manurung et al. (2025) shows that it yields higher F1 scores than SVM. This study employs Random Forest and SVM on the Gotham Dataset 2025, which features traffic characteristics and attack categories distinct from those in earlier studies. Consequently, the results are compared with previous findings, taking into account differences in datasets, the number of classes, preprocessing steps, and the methods utilized.

VIII. USING AND CITING DATASETS

This study uses the Gotham 2025 dataset, which is a publicly available set of data meant for analyzing network traffic and doing research in cybersecurity. It includes both harmless and harmful network traffic that was gathered in a controlled network setting. The dataset includes different

kinds of network attacks, like those related to Mirai and other types of harmful traffic.

The Gotham 2025 dataset was acquired from its official website at https://zenodo.org/records/14502760 [14]. It gives data that's already been processed in CSV format, which is used here for preparing the data, choosing important features, classifying attacks, and evaluating machine learning models. This study specifically looks at network traffic linked to Mirai attacks and uses the existing labels to sort the traffic into different attack types and normal traffic.

This study properly credits the original source and related publications as required by the dataset's terms of use [14], [16]. These citations are used to give credit to the people who made the dataset and follow the rules for using this free-to-use dataset.

IX. COMPARATIVE ANALYSIS OF RESULTS

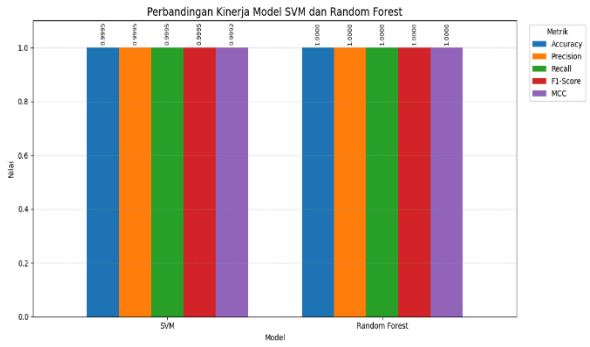


Figure 11 Comparison of SVM and RF algorithm performance

Based on Figure 11, both algorithms demonstrate very high classification performance. However, Random Forest shows slightly better results than the Support Vector Machine (SVM) across all evaluation metrics used. The SVM model achieved an Accuracy of 99.95%, Precision of 99.95%, Recall of 99.95%, F1-Score of 99.95%, and MCC of 0.9992. These values indicate that the SVM possesses excellent capability in distinguishing between all attack categories in the research dataset. The high MCC value also indicates a very strong relationship between the predicted results and the actual labels, unaffected by the class distribution imbalance following the SMOTE process. Meanwhile, the Random Forest algorithm achieved Accuracy, Precision, Recall, F1-Score, and MCC values of 1.0000 (or 100%) for each metric. These results demonstrate that all test data were correctly classified without any prediction errors across any of the classes. An MCC value of 1.0000 also indicates a perfect correlation between the classification results and the actual labels. When compared directly, the performance gap between the two algorithms is actually very small. The difference in accuracy is only about 0.05%: 99.95% for SVM and 100% for Random Forest. Similarly, for the Precision, Recall, and F1-Score metrics, the difference lies in the range of 0.0005. Although Random Forest numerically yielded higher performance, both algorithms demonstrated excellent classification capabilities on the dataset used. This difference can be explained by the characteristics of each algorithm. The Support Vector Machine operates by finding an optimal hyperplane capable of separating classes with a maximum margin; this approach is highly effective for high-dimensional data with complex boundaries. In contrast, Random Forest employs an ensemble learning approach, constructing multiple decision trees that are subsequently combined using a majority voting mechanism. This approach enables Random Forest to better handle data variations, reduce the risk of overfitting, and enhance the stability of prediction results. Furthermore, the high performance achieved by both algorithms is also attributable to the preprocessing stages carried out prior to classification. Processes such as data cleaning, categorical feature encoding, data standardization, and the application of the Synthetic Minority Over-sampling Technique (SMOTE) successfully produced a more balanced data distribution, ensuring that each class had an equal opportunity to be learned by the model. Consequently, both SVM and Random Forest were able to optimally recognize the characteristics of each class.

X. CONCLUSION

A. The Support Vector Machine (SVM) algorithm shows very good classification performance with an Accuracy of 99.95%, Precision of 99.95%, Recall of 99.95%, F1-Score of 99.95%, Matthews Correlation Coefficient (MCC) of 0.9992, and Cohen's Kappa of 0.9992. Based on the confusion matrix, the model made only 4 classification errors out of 7,373 test data points, allowing it to recognize the characteristics of each class with very high accuracy.

B. The Random Forest algorithm performed the best in this study, achieving scores of 100% for Accuracy, Precision, Recall, F1-Score, MCC, and Cohen's Kappa. The confusion matrix results show that all 7,373 test data points were correctly classified with no prediction errors, making Random Forest the model with the best performance on the dataset used.

C. Based on the comparison of the two algorithms, Random Forest performed slightly better than Support Vector Machine, although the difference in their performance was not very big. Random Forest completely removed all the classification errors that were still happening with SVM, making it more effective at recognizing all traffic categories in the research dataset.

XI. REFERENCE LIST RULES

M. Zhao and G. E. Suh, “FPGA-Based Remote Power Side-Channel Attacks,” in *Proceedings - IEEE Symposium on*

Security and Privacy, Institute of Electrical and Electronics Engineers Inc., Jul. 2022, pp. 229–244. doi: 10.1109/SP.2018.00049.

[2] G. Bastos *et al.*, "Identifying and Characterizing Bashlite and Mirai CC Servers," *Proc. IEEE Symp. Comput. Commun.*, vol. 2019-June, 2019, doi: 10.1109/ISCC47284.2019.8969728.

[3] A. Purnomo, A. Kurniasih, A. Nuarminah, and S. Hartati, "Peran Artificial Intelligence dalam Deteksi Dini Ancaman Keamanan Jaringan," *Jurnal Minfo Polgan*, vol. 13, no. 2, pp. 2044–2048, Dec. 2024, doi: 10.33395/jmp.v13i2.14356.

[4] G. de C. Bertoli, L. A. P. Junior, F. A. N. Verri, A. L. dos Santos, and O. Saotome, "Bridging the gap to real-world for network intrusion detection systems with data-centric approach," no. NeurIPS, 2021, [Online]. Available: http://arxiv.org/abs/2110.13655

[5] H. M. R. U. Rehman *et al.*, "A systematic literature study of machine learning techniques based intrusion detection: datasets, models, challenges, and future directions," *J. Big Data*, vol. 12, no. 1, p. 264, 2025, [Online]. Available: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-025-01323-2

[6] "9+Volume+2,+Nomor+2,+Desember+2024+73-78".

[7] P. Gujjar J and N. Kumar V, "Google Colaboratory : Tool for Deep Learning and Machine Learning Applications," 2021.

[8] Y. Li, Y. Yang, P. Song, L. Duan, and R. Ren, "An improved SMOTE algorithm for enhanced imbalanced data classification by expanding sample generation space," *Sci. Rep.*, vol. 15, no. 1, pp. 1–21, 2025, doi: 10.1038/s41598-025-09506-w.

[9] K. et al 2023, "No Title 済無 No Title No Title No Title," vol. 32, no. 3, pp. 167–186, 2021.

[10] A. Amara Korba, A. Diaf, M. A. Bouchiha, and Y. Ghamri-Doudane, "Mitigating IoT botnet attacks: An early-stage explainable network-based anomaly detection approach," *Comput. Commun.*, vol. 241, no. January, p. 108270, 2025, doi: 10.1016/j.comcom.2025.108270.

[11] R. S. Nurhalizah, R. Ardianto, and P. Purwono, "Analisis Supervised dan Unsupervised Learning pada Machine Learning: Systematic Literature Review," *Jurnal Ilmu Komputer dan Informatika*, vol. 4, no. 1, pp. 61–72, Aug. 2024, doi: 10.54082/jiki.168.

[12] Z. Fatah and I. Addewiyah, "Penerapan Algoritma Random Forest dan Teknik SMOTE untuk Prediksi Kematian Akibat Gagal Jantung Menggunakan RapidMiner."

[13] N. Huda Ovirianti, M. Zarlis, and H. Mawengkang, "Support Vector Machine Using A Classification Algorithm," *Jurnal dan Penelitian Teknik Informatika*, vol. 6, no. 3, 2022, doi: 10.33395/sinkron.v7i3.

[14] A. Čenys, S. K. Hora, and N. Goranin, "Emulation-Based Dataset EmuIoT-VT for NIDS in IoT Systems," *Sensors*, vol. 25, no. 16, 2025, doi: 10.3390/s25165077.

[15] M. Ma'rufudin and A. Yudhistira, "Analisis Sentimen Petani Milenial Pada Media Sosial X Menggunakan Algortitma Support Vector Machine (SVM)," *Jurnal Pendidikan dan Teknologi Indonesia*, vol. 5, no. 3, pp. 845–857, 2025, doi: 10.52436/1.jpti.717.