Jurnal Pseudocode Vol 13 (1) Februari 2026, hal 28-35

ISSN 2355-5920 | e-ISSN 2655-1845
https://ejournal.unib.ac.id/pseudocode/index

Penerapan Algoritma *Random Forest* dan *Support Vector Machine* Untuk Deteksi *Distributed Denial of Service*

Asep Ririh Riswaya*, Abdul Fadlil, Anton Yudhana

Universitas Ahmad Dahlan, Jl. Kapas No.9, Semaki, Kec. Umbulharjo, Kota Yogyakarta, Daerah Istimewa Yogyakarta 55166, Indonesia

Informasi Naskah:
Diterima: 06 Januari 2026/ Direview: 08 Januari 2026/ Direvisi: 06 Februari 2026/ Disetujui Terbit: 20 Februari 2026
DOI: 10.33369/pseudocode.13.1.28-35
*Korespondensi: 2537083013@webmail.uad.ac.id

Abstract

Distributed Denial of Service (DDoS) attacks significantly threaten the availability of modern network services by overwhelming resources with malicious traffic. The increasing complexity of these attacks, including multi-vector and low-rate variants, reduces the effectiveness of traditional detection methods based on signatures and static rules. This study explores the effectiveness of Random Forest (RF) and Support Vector Machine (SVM) algorithms in detecting DDoS attacks using the CICDDoS2019 dataset, focusing on the impact of various decision threshold values on performance.

The CICDDoS2019 dataset consists of 431,371 network traffic flows with 80 numerical features. Preprocessing involves eliminating null values, standardizing numerical attributes, and encoding labels into binary classifications of normal and DDoS traffic. The dataset is then divided into training and testing sets at a 70:30 ratio. Performance evaluation is done using a confusion matrix to calculate accuracy, precision, recall, and F1-score. Results show that both algorithms perform well, but Random Forest offers greater consistency, with a threshold of 0.5 achieving the best balance in metrics.

Keywords: DDoS, Machine Learning, Random Forest, Support Vector Machine, Threshold

1. Pendahuluan

Seiring dengan transformasi digital di berbagai sektor seperti pendidikan, pemerintahan, layanan publik, dan layanan komersial, kebutuhan masyarakat modern terhadap layanan daring dan infrastruktur jaringan terus meningkat. Namun, peningkatan ketergantungan terhadap layanan berbasis jaringan ini juga diiringi dengan munculnya berbagai kendala operasional, seperti keterbatasan kapasitas bandwidth, kompleksitas manajemen trafik, serta meningkatnya potensi gangguan terhadap stabilitas sistem jaringan. Dalam kondisi tersebut, serangan siber bertipe *Distributed Denial of Service* (DDoS) menjadi semakin *urgent* untuk ditangani, karena mampu mengeksploitasi kerentanan infrastruktur jaringan dan melumpuhkan layanan secara masif melalui pembanjiran trafik berbahaya. Serangan DDoS saat ini semakin kompleks, termasuk serangan *low-rate* dan *multi-vector*, yang dirancang untuk menyerupai trafik biasa sehingga sulit dideteksi oleh metode konvensional. Metode konvensional seperti *signature-based* atau *rule-based intrusion detection system* (IDS) sering gagal mendeteksi serangan baru atau serangan dengan pola trafik yang tidak pernah dikenali sebelumnya [1]. Oleh karena itu, pendekatan berbasis *Machine Learning* (ML) menjadi pilihan yang menarik karena mampu mengidentifikasi pola dan generalisasi pada data trafik. Ini terutama berlaku untuk dataset yang representatif terhadap serangan modern. CICDDoS2019 adalah dataset *benchmark* yang populer yang menyajikan data aliran trafik (*flow*) berlabel baik trafik biasa maupun berbagai jenis serangan DDoS modern. Dataset ini ideal untuk pelatihan dan evaluasi model ML [2].

Dengan dataset seperti CICDDoS2019, beberapa algoritma pembelajaran mesin populer, seperti *Random Forest* (RF) dan *Support Vector Machine* (SVM), telah banyak digunakan dalam studi deteksi DDoS. Penelitian sebelumnya menunjukkan bahwa RF dan SVM dapat melakukan klasifikasi trafik normal dan DDoS dengan baik [3], [4]. SVM sangat baik dalam membentuk *threshold* nonlinier dan beroperasi dengan baik pada data yang seimbang dan tidak seimbang. Di sisi lain, RF dianggap cukup kuat untuk menangani data berukuran besar dan cenderung tahan terhadap *overfitting* [5].

Namun, sebagian besar penelitian menggunakan *threshold* tetap, juga dikenal sebagai *threshold* statik. Misalnya, mereka menetapkan probabilitas lebih dari 0,5 sebagai klasifikasi "serangan". Namun, output model ML seperti RF atau SVM yang menggunakan probabilitas seharusnya memiliki fleksibilitas dalam menentukan *threshold* sesuai kebutuhan operasional. Nilai *threshold* dapat menghasilkan hasil yang berbeda antara *False Positive* (FP), yang merupakan trafik normal yang dianggap serangan, dan *False Negative* (FN), yang merupakan trafik yang lolos sebagai normal. Jika *threshold* terlalu sensitif, risiko FP meningkat, sementara jika *threshold* terlalu konservatif, risiko FN meningkat [6].

Hanya sejumlah kecil penelitian yang secara sistematis mengkaji dampak perubahan *threshold* terhadap metrik kinerja seperti *Accuracy*, *Precision*, *Recall*, F1-*Score* dan

Jurnal Pseudocode Vol 13 (1) Februari 2026, hal 28-35

trade-off FP/FN [7], [8]. Oleh karena itu, penelitian masih kurang dalam mempelajari bagaimana nilai *threshold* mempengaruhi kinerja model ML dalam deteksi DDoS. Ini terutama berlaku untuk dataset realistis seperti CICDDoS2019.

Untuk mendeteksi DDoS pada dataset CICDDoS2019, penelitian ini mengusulkan evaluasi komparatif algoritma *Random Forest* dan SVM dengan berbagai variasi *threshold*. Fokus utama adalah menilai dampak *threshold* terhadap metrik kinerja dan menemukan konfigurasi model *threshold* terbaik yang menghasilkan keseimbangan optimal antara deteksi serangan dan minimnya peringatan palsu [9].

Fokus utama penelitian ini adalah untuk meningkatkan keandalan deteksi serangan DDoS berbasis ML. Pertama, penelitian ini mengevaluasi secara menyeluruh kinerja dua algoritma yang sangat popular yakni *Random Forest* dan *Support Vector Machine* (SVM). Sumber data trafik berlabel adalah dataset CICDDoS2019. Sejauh mana kedua algoritma mampu membedakan trafik normal dari serangan.

Selain itu, penelitian ini mempelajari secara khusus bagaimana perubahan nilai *threshold* dalam pengambilan keputusan model memengaruhi berbagai metrik kinerja, termasuk *Accuracy*, *Precision*, *Recall* dan *F1-Score*, serta bagaimana *threshold* berdampak pada keseimbangan antara *false positive* dan *false negative*. Analisis ini penting karena dengan memilih *threshold* yang tepat, maka akan menentukan apakah sistem deteksi lebih sensitif terhadap serangan atau lebih toleran terhadap trafik normal agar tidak menghasilkan alarm palsu yang berlebihan.

Evaluasi ini juga menyajikan rekomendasi konfigurasi terbaik antara algoritma dan *threshold* yang diuji, yang dapat digunakan sebagai referensi untuk penerapan sistem deteksi DDoS secara praktis. Hasil evaluasi ini diharapkan dapat membantu administrator jaringan dalam memilih pengaturan model yang tidak hanya kinerja tinggi secara statistik tetapi juga seimbang dalam menangani kesalahan deteksi yang mungkin terjadi dalam berbagai kondisi operasional.

2. Metodologi Penelitian

Metode pada artikel ini menggunakan metodologi eksperimen komputasional yang terstruktur untuk mengevaluasi model *machine learning* pada jaringan. Pada bagian *Experimental Methodology*, [10], [11] menjelaskan secara rinci pemilihan dataset, skenario pengujian, konfigurasi model, pembagian data latih uji, metrik evaluasi, serta pengaturan lingkungan komputasi, lalu melanjutkan dengan bagian evaluasi yang membahas hasil eksperimen.

Penggunaan Dataset CICDDoS2019 adalah standar baru dalam pengujian keamanan jaringan karena mencakup trafik *benign* serta berbagai jenis serangan DDoS kontemporer seperti DNS, LDAP dan SNMP. Dataset ini mewakili kondisi lalu lintas jaringan nyata dengan lebih akurat daripada dataset sebelumnya [12].

Preprocessing data, yang terdiri dari tiga langkah penting, adalah tahap berikutnya. Pertama, untuk menjaga integritas data selama proses pelatihan, nilai kosong atau nilai yang tidak ada dihapus. Kedua, agar perbedaan skala antar fitur tidak bias terhadap kinerja algoritma tertentu, seperti *Support Vector Machine*, fitur numerik dinormalisasi atau distandarisasi [13]. Ketiga, metode *encoding* mengubah label data serangan DDoS diberi label 1 dan trafik normal diberi label 0. Ini memudahkan klasifikasi biner [14].

Setelah data siap, dataset dibagi menjadi 70% untuk instrusi dan 30% untuk pengujian. Dalam studi deteksi intrusi, metode ini sering digunakan untuk memastikan bahwa model memiliki data latih yang cukup dan data uji yang representatif untuk mengukur kemampuan generalisasi [15].

Dua algoritma *machine learning*, *Random Forest* (RF) dan *Support Vector Machine* (SVM), diterapkan secara paralel pada tahap Pelatihan Model. *Random Forest* dipilih karena ketahanan terhadap *overfitting* dan kemampuan untuk menangani data yang sangat besar, sementara SVM dikenal dapat menemukan *hyperplane* yang ideal untuk pemisahan kelas yang kompleks [16].

Setelah itu, prediksi probabilitas dilakukan. Di sini, masing-masing model yang diuji tidak hanya memprediksi label kelas secara langsung, tetapi juga menghasilkan skor kemungkinan keanggotaan kelas DDoS. Skor ini menjadi dasar untuk Penerapan Variasi *Threshold*, di mana penelitian ini menguji tiga *threshold*, yaitu 0,3, 0,5 dan 0,7. Jika probabilitas prediksi lebih besar dari *threshold*, maka data diklasifikasikan sebagai DDoS, sedangkan jika tidak, maka data diklasifikasikan sebagai normal, metode ini memungkinkan analisis fleksibilitas model dalam menghadapi sensitivitas *trade-off* deteksi [17].

Selanjutnya, kinerja model dievaluasi pada tahap evaluasi kinerja, di mana *Confusion Matrix* digunakan untuk menghitung *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN). *Accuracy*, *Precision*, *Recall*, dan *F1-Score* dihitung berdasarkan komponen tersebut untuk memberikan gambaran performa yang menyeluruh [18]. Berikut hitungan *confusion matrix*:

Tabel 1 evaluasi *confusion matrix* yang digunakan untuk mengukur kinerja model klasifikasi terutama pada kasus biner (DDoS = 1, Normal = 0), terdiri dari empat komponen dapat dilihat dari tabel 1.

Tabel 1. Evaluasi *Confusion Matrix*

	Prediksi Positif	Prediksi Negatif
Aktual Positif	*True Positive* (TP)	*True Negative* (TN)
Aktual Negatif	*False Positive* (TN)	*False Negative* (FN)

Keterangan:

(TP) adalah *True Positive* (Jumlah data serangan DDoS yang benar-benar diprediksi sebagai DDoS oleh model)

(TN) adalah *True Negative* (Jumlah data trafik normal yang benar-benar diprediksi sebagai normal)

(FP) adalah *False Positive* / *Type-I Error* (Jumlah data trafik normal yang salah diprediksi sebagai DDoS) mengakibatkan *false* alarm.

(FN) adalah *False Negative* / *Type-II Error* (Jumlah data serangan DDoS yang salah diprediksi sebagai normal) berbahaya karena serangan lolos terdeteksi.

Akurasi (*accuracy*) mengukur proporsi prediksi model yang benar terhadap seluruh data.

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN} \quad (1)$$

Mengukur Presisi (*precision*) seberapa tepat model memprediksi kelas positif (DDoS).

$$Precision = \frac{TP}{TP+FP} \quad (2)$$

Jurnal Pseudocode Vol 13 (1) Februari 2026, hal 28-35

Recall Mengukur seberapa banyak DDoS yang berhasil terdeteksi dengan benar.

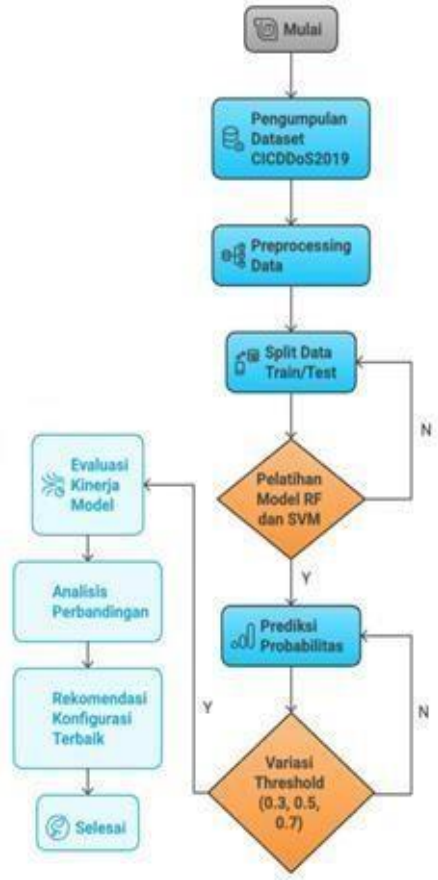
$$Recall = \frac{TP}{TP + FN} \quad (3)$$

F1-Score adalah *harmonic mean* antara *Precision* dan *Recall*.

$$F1 = \frac{2\ X\ Recall\ x\ Precision}{Recall + Precision} \quad (4)$$

Terakhir, analisis perbandingan dilakukan untuk melihat bagaimana perubahan *threshold* berdampak pada tingkat kesalahan FP dan FN. Tujuan analisis ini adalah untuk menemukan kombinasi model dan *threshold* terbaik yang memberikan keseimbangan optimal antara deteksi serangan yang akurat dan minimnya peringatan palsu, yang merupakan komponen penting dalam sistem keamanan jaringan operasional [19]. Penelitian komparatif terbaru menunjukkan bahwa analisis seperti ini penting untuk di teliti dan dikembangkan. Di sini, prototipe terbaik dipilih dan diusulkan sebagai model yang dapat disesuaikan oleh praktisi keamanan jaringan.

Tahap penelitian dapat dilihat pada gambar 1 dan akan menghasilkan temuan tentang konfigurasi model *threshold* yang dianggap paling ideal untuk deteksi DDoS pada dataset CICDDoS2019. Rekomendasi ini dapat digunakan sebagai landasan untuk pengembangan lebih lanjut, seperti penggabungan dengan sistem deteksi intrusi nyata atau pengujian lingkungan nyata.



Gbr. 1. Tahapan Penelitian

3. Hasil dan Pembahasan

Eksperimen dilakukan untuk mengevaluasi kinerja algoritma *Random Forest* (RF) dan *Support Vector Machine* (SVM) dalam mendeteksi serangan DDoS menggunakan dataset CICDDoS2019. Evaluasi difokuskan pada pengaruh variasi nilai *threshold* keputusan terhadap metrik kinerja, yaitu *Accuracy*, *Precision*, *Recall* dan *F1-Score*, serta perubahan nilai *false positive* (FP) dan *false negative* (FN). Model dilatih menggunakan data *training* sebesar 70% dan diuji pada data *testing* sebesar 30%. Setiap model menghasilkan probabilitas kelas DDoS, yang kemudian dikonversi menjadi label kelas melalui penerapan *threshold* sebesar 0,3, 0,5, dan 0,7.

1. Tahap *Preprocessing*
   Dataset CICDDoS2019 yang digunakan dalam penelitian ini terdiri dari 431.371 aliran trafik jaringan dengan 80 fitur numerik yang merepresentasikan karakteristik statistik komunikasi jaringan. Tahap awal *preprocessing* dilakukan dengan memuat dataset dalam format CSV dan memverifikasi struktur data untuk memastikan konsistensi jumlah fitur dan observasi dilihat dari Gambar 2.

```
# ================================================
# 1. Import library
# ================================================
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix

# ================================================
# 2. Upload dataset CICDDoS2019 (CSV)
# ================================================
from google.colab import files
uploaded = files.upload()

# Asumsikan file bernama "data.csv"
df = pd.read_csv(list(uploaded.keys())[0])

print("Dataset dimensi:", df.shape)
df.head()
```

```
Choose Files  cicddos2019_dataset.csv
cicddos2019_dataset.csv(text/csv) - 154662153 bytes, last modified: 13/11/2025 - 100% done
Saving cicddos2019_dataset.csv to cicddos2019_dataset.csv
Dataset dimensi: (431371, 80)
```

Gbr. 2. Tahapan *Preprocessing*

a. Menghapus nilai kosong
   Selanjutnya, kolom yang tidak relevan seperti indeks otomatis hasil ekspor data dihapus karena tidak memiliki kontribusi terhadap proses klasifikasi dengan hasil pada Gambar 3.

```
# Menghapus nilai kosong
df = df.dropna()
print("Dataset setelah menghapus missing values:", df.shape)
```

```
Dataset setelah menghapus missing values: (431371, 80)
```

Gbr. 3. Tahapan *Preprocessing Cleaning* Data

b. Standarisasi fitur numerik
   Tahap pembersihan data dilakukan pada Gambar 4

Jurnal Pseudocode Vol 13 (1) Februari 2026, hal 28-35

dengan menghilangkan nilai kosong (*missing values*) untuk mencegah terjadinya bias dan kesalahan komputasi pada proses pelatihan model, mengingat algoritma *Support Vector Machine* sangat sensitif terhadap keberadaan nilai tidak terdefinisi [20].

```
# Pisahkan fitur dan label
# Catatan: ubah 'Label' sesuai nama kolom pada dataset kamu
X = df.drop(['Label', 'Class'], axis=1)
y = df['Label']

# Standarisasi fitur numerik
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

print("Preprocessing selesai: fitur telah distandarisasi")
```

```
Preprocessing selesai: fitur telah distandarisasi
```

Gbr. 4. Tahapan *Preprocessing* Distandarisasi Data

c. *Encoding* label (Normal = 0, DDoS = 1)

Label data yang semula berupa kategori tekstual kemudian dikodekan ke dalam bentuk numerik biner dilakukan pada Gambar 5, di mana trafik normal diberi label 0 dan DDoS diberi label 1. Pendekatan klasifikasi biner ini dipilih untuk menyederhanakan proses deteksi serangan dan meningkatkan stabilitas model dalam membedakan trafik normal dan berbahaya [21]. Seluruh fitur numerik selanjutnya distandarisasi menggunakan metode *standardization* sehingga setiap fitur memiliki nilai rata-rata nol dan simpangan baku satu. Proses ini bertujuan untuk mengatasi perbedaan skala antar fitur dan meningkatkan performa algoritma yang sensitif terhadap skala data, khususnya *Support Vector Machine* [22].

```
# Encoding label
y = y.replace({'BENIGN':0, 'DDoS':1, 'DDOS':1})
print("Jumlah label:\n", y.value_counts())
```

```
Jumlah label:
 Label
DrDoS_NTP         121368
TFTP               98917
Benign             97831
Syn                49373
UDP                18090
DrDoS_UDP          10420
UDP-lag             8872
MSSQL               8523
DrDoS_MSSQL         6212
DrDoS_DNS           3669
DrDoS_SNMP          2717
LDAP                1906
DrDoS_LDAP          1440
Portmap              685
NetBIOS              644
DrDoS_NetBIOS        598
UDPLag                55
WebDDoS               51
Name: count, dtype: int64
```

Gbr. 5. Tahapan *Preprocessing Encoding* Label Data

2. Data *Training* dan *Testing*

Tahap akhir *preprocessing* dilakukan dengan membagi dataset menjadi data latih dan data uji dengan rasio 70:30 dapat dilihat paga Gambar 6. Pembagian ini bertujuan untuk memastikan proses evaluasi model dilakukan secara objektif serta menghindari terjadinya *overfitting*. Seluruh tahapan *preprocessing* ini dirancang untuk menghasilkan data yang bersih, konsisten, dan representatif sehingga mampu mendukung analisis kinerja algoritma *Random Forest* dan *Support Vector Machine* dalam mendeteksi serangan DDoS secara efektif [23].

```
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.3, random_state=42, stratify=y
)
```

Gbr. 6. Tahapan *Traning* dan *Testing*

3. Pelatihan Model *Random Forest*

Hasil pelatihan algoritma *Random Forest* menunjukkan kinerja klasifikasi yang sangat tinggi pada dataset CICDDoS2019, dengan nilai *accuracy* keseluruhan mencapai 0,99. Nilai *weighted average* untuk *Accuracy*, *Precision*, *Recall* dan F1-*Score* yang masing-masing mendekati 0,99 mengindikasikan bahwa model mampu mengklasifikasikan mayoritas data secara akurat meskipun dataset memiliki distribusi kelas yang tidak seimbang. Kinerja yang sangat baik juga terlihat pada sebagian besar kelas serangan utama, seperti DrDoS-NTP, SYN, dan TFTP, yang masing-masing mencapai nilai presisi dan *recall* mendekati atau sama dengan 1,00. Hal ini menunjukkan bahwa *Random Forest* efektif dalam mempelajari pola trafik jaringan yang kompleks dan mampu membedakan trafik normal dan serangan DDoS dengan tingkat kesalahan yang sangat rendah.

Meskipun demikian, performa model pada beberapa kelas dengan jumlah data yang sangat terbatas, seperti UDPLag dan WebDDoS, menunjukkan penurunan nilai *recall* dan *F1-Score*. Kondisi ini mengindikasikan bahwa keterbatasan jumlah sampel pada kelas minoritas masih menjadi tantangan dalam proses klasifikasi, meskipun tidak berdampak signifikan terhadap performa global model. Nilai *macro average F1-Score* sebesar 0,89 mencerminkan adanya variasi kinerja antar kelas, namun secara keseluruhan *Random Forest* tetap menunjukkan stabilitas dan keandalan yang tinggi sebagai model deteksi DDoS. Temuan ini memperkuat peran *Random Forest* sebagai algoritma yang efektif untuk sistem deteksi serangan DDoS berbasis *machine learning* pada dataset berskala besar. Performa model *Random Forest* secara keseluruhan dapat dilihat pada gambar 7 dan gambar 8 dibawah ini.

```
print("=== Random Forest Performance ===")
print(confusion_matrix(y_test, y_pred_rf))
print(classification_report(y_test, y_pred_rf))
```

```
=== Random Forest Performance ===
[[29339     1     0     0     3     0     0     1     0     0     1     0
      4     0     0     0     0     0]
 [    1   999     0     0     3     0     0     0     7    54     0     8
      0     1    28     0     0     0]
 [    1     4   393     1     0     0    23     0     5     5     0     0
      0     0     0     0     0     0]
 [    0     0     6  1853     1     0     3     0     0     1     0     0
      0     0     0     0     0     0]
 [   15     0     1     1 36391     1     0     0     0     1     0     0
      0     1     0     0     0     0]
 [    3     0     1     3     0   128     7     6     0    23     5     3
      0     0     0     0     0     0]
 [    0     0    19     3     0    12   767     0     4     6     4     0
      0     0     0     0     0     0]
 [    0     0     0     0     0     0     4     4  3094     0    22     0     0
      0     1     1     0     0     0]
 [    0     4    10     0     0     0    11     0   539     8     0     0
      0     0     0     0     0     0]
 [    0    12     1     1     0     0     2     0    20  2507     1     0
      2     3     7     1     0     0]
 [    0     0     0     0     0     3     1     0     0     1   179     7
      1     0     1     0     0     0]
 [    5     0     0     0     0     0     0     0     0     2     3   181
     13     0     2     0     0     0]
 [   15     0     0     0     0     0     0     0     0     1     1     5
  14775     9     2     3     1     0]
 [   14     0     0     0     3     0     0     0     1     1     0     1
      9 29639     1     6     0     0]
 [    0    28     0     0     0     0     0     1     0    31     0     1
      3     0  5226   131     6     0]
 [    1     8     0     0     0     0     1     0     3     1     0     0
      2     3   468  2174     1     0]
 [    1     0     0     0     0     0     0     0     0     0     0     0
      2     0     8     0     5     0]
 [    4     0     0     0     4     0     0     0     0     0     0     0
      0     1     0     0     0     6]]
```

Jurnal Pseudocode Vol 13 (1) Februari 2026, hal 28-35

Gbr. 7. Tahapan Latih *Random Forest* - A

	precision	recall	f1-score	support
Benign	1.00	1.00	1.00	29349
DrDoS_DNS	0.95	0.91	0.93	1101
DrDoS_LDAP	0.91	0.91	0.91	432
DrDoS_MSSQL	1.00	0.99	0.99	1864
DrDoS_NTP	1.00	1.00	1.00	36411
DrDoS_NetBIOS	0.86	0.72	0.78	179
DrDoS_SNMP	0.94	0.94	0.94	815
DrDoS_UDP	1.00	0.99	0.99	3126
LDAP	0.93	0.94	0.94	572
MSSQL	0.94	0.98	0.96	2557
NetBIOS	0.92	0.93	0.93	193
Portmap	0.88	0.88	0.88	206
Syn	1.00	1.00	1.00	14812
TFTP	1.00	1.00	1.00	29675
UDP	0.91	0.96	0.94	5427
UDP-lag	0.94	0.82	0.87	2662
UDPLag	0.38	0.31	0.34	16
WebDDoS	1.00	0.40	0.57	15
accuracy			0.99	129412
macro avg	0.92	0.87	0.89	129412
weighted avg	0.99	0.99	0.99	129412

Gbr. 8. Tahapan Latih *Random Forest* - B

4. Pelatihan Model SVM

Hasil pelatihan algoritma *Support Vector Machine* menunjukkan bahwa model mampu mencapai kinerja klasifikasi yang tinggi secara keseluruhan, dengan nilai *accuracy* sebesar 0,98 pada dataset CICDDoS2019. Nilai *weighted average* untuk *Accuracy*, *Precision*, *Recall* dan *F1-Score* yang masing-masing mencapai 0,98 mengindikasikan bahwa SVM mampu mengklasifikasikan sebagian besar data dengan baik, khususnya pada kelas-kelas yang memiliki jumlah sampel besar seperti DrDoS_NTP, SYN, dan TFTP, yang masing-masing menunjukkan nilai *F1-Score* mendekati atau sama dengan 1,00. Hal ini menunjukkan bahwa SVM efektif dalam membangun batas pemisah antar kelas ketika pola data cukup jelas dan representatif.

Namun demikian, performa SVM menunjukkan variasi yang cukup signifikan antar kelas, sebagaimana tercermin dari nilai *macro average F1-Score* sebesar 0,78. Beberapa kelas dengan jumlah sampel yang sangat terbatas, seperti UDPLag dan WebDDoS, tidak berhasil terklasifikasi dengan baik, yang ditunjukkan oleh nilai *Precision*, *Recall*, F1-*Score* sebesar 0,00. Selain itu, pada beberapa kelas serangan seperti DrDoS_LDAP, DrDoS_NetBIOS, dan Portmap, nilai *recall* relatif lebih rendah, yang mengindikasikan meningkatnya jumlah *false negative*. Temuan ini menunjukkan bahwa meskipun SVM memiliki performa global yang tinggi, model ini lebih sensitif terhadap ketidakseimbangan data dibandingkan *Random Forest*, sehingga memerlukan penyesuaian lebih lanjut seperti optimasi parameter atau penanganan kelas minoritas untuk meningkatkan keandalannya dalam deteksi serangan DDoS. Performa model *Random Forest* secara keseluruhan dapat dilihat pada gambar 9.

```
print("=== SVM Performance ===")
print(confusion_matrix(y_test, y_pred_svm))
print(classification_report(y_test, y_pred_svm))
```

```
=== SVM Performance ===
[[29294     1     0     0     3     5     0     0     0     0     9    13
    21     0     3     0     0]
 [   15  1022     0     0     1     0     0     0     0     0     0    16
     0    23    24     0     0     0]
 [    1     1   247    18     0     0   165     0     0     0     0     0
     0     0     0     0     0]
 [    1     0    17  1759     0     0    86     0     0     0     0     0
     0     1     0     0     0]
 [  123     0     1    31 36246     9     0     0     0     0     0     0
     1     0     0     0     0]
 [    4     0     0    50     0   111     7     2     0     0     5     0
     0     0     0     0     0]
 [    0     0    74     2     0    29   710     0     0     0     0     0
     0     0     0     0     0]
 [    0     0     0    85     0     0     6  3013     0    22     0     0
     0     0     0     0     0]
 [    1     0    10     0     0     0     8     0   504    49     0     0
     0     0     0     0     0]
 [    2    38     0     0     0     0    13     1    71  2417     0     0
     1     7     7     0     0]
 [    1     0     0     0     0     0     0     0     0     1   185     0
     1     0     5     0     0]
 [   14    42     0     0     0     0     0     0     0     0     0   140
     4     0     6     0     0]
 [   71     5     0     0     0     0     0     0     0     0     1     4
 14692     2    15    22     0     0]
 [   30     9     0     0     2     0     0     0     2    48     0     0
    62 29517     3     2     0     0]
 [    1   120     0     0     0     0     0     8     0    74     0     0
     0     9  5215     0     0     0]
 [    9    35     0     0     0     0     0     0     0     0     0     0
    63     0   650  1905     0     0]
 [    1     0     0     0     0     0     0     0     0     2     0     0
     2     1    10     0     0     0]
 [   14     0     0     1     0     0     0     0     0     0     0     0
     0     0     0     0     0     0]]
```

	precision	recall	f1-score	support
Benign	0.99	1.00	0.99	29349
DrDoS_DNS	0.80	0.93	0.86	1101
DrDoS_LDAP	0.71	0.57	0.63	432
DrDoS_MSSQL	0.90	0.94	0.92	1864
DrDoS_NTP	1.00	1.00	1.00	36411
DrDoS_NetBIOS	0.72	0.62	0.67	179
DrDoS_SNMP	0.71	0.87	0.78	815
DrDoS_UDP	1.00	0.96	0.98	3126
LDAP	0.87	0.88	0.88	572
MSSQL	0.92	0.95	0.94	2557
NetBIOS	0.93	0.96	0.94	193
Portmap	0.81	0.68	0.74	206
Syn	0.99	0.99	0.99	14812
TFTP	1.00	0.99	1.00	29675
UDP	0.88	0.96	0.92	5427
UDP-lag	0.99	0.72	0.83	2662
UDPLag	0.00	0.00	0.00	16
WebDDoS	0.00	0.00	0.00	15
accuracy			0.98	129412
macro avg	0.79	0.78	0.78	129412
weighted avg	0.98	0.98	0.98	129412

Gbr. 9. Tahapan Latih *Support Vector Machine*

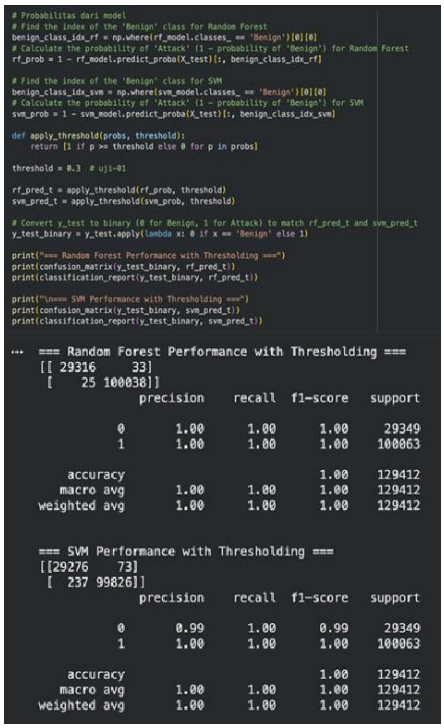
5. Variasi *Threshold* untuk RF dan SVM

Setiap algoritma klasifikasi berbasis *machine learning* yang digunakan dalam penelitian ini, yaitu *Random Forest* dan *Support Vector Machine*, menghasilkan keluaran berupa nilai probabilitas yang merepresentasikan tingkat keyakinan model terhadap suatu sampel apakah termasuk kelas serangan DDoS atau trafik normal. Nilai probabilitas ini berada pada rentang 0 hingga 1 dan memberikan informasi yang lebih banyak dibandingkan prediksi kelas secara langsung. Pendekatan berbasis probabilitas memungkinkan sistem deteksi untuk tidak hanya memberikan keputusan biner, tetapi juga mempertimbangkan tingkat kepercayaan model terhadap setiap prediksi, yang sangat penting dalam konteks keamanan jaringan dengan karakteristik trafik yang dinamis dan tidak selalu terpisah.

Untuk mengonversi nilai probabilitas tersebut menjadi label kelas akhir, penelitian ini menerapkan nilai *threshold* keputusan sebesar 0,3, 0,5, dan 0,7. *Threshold* berfungsi sebagai batas keputusan, di

Jurnal Pseudocode Vol 13 (1) Februari 2026, hal 28-35

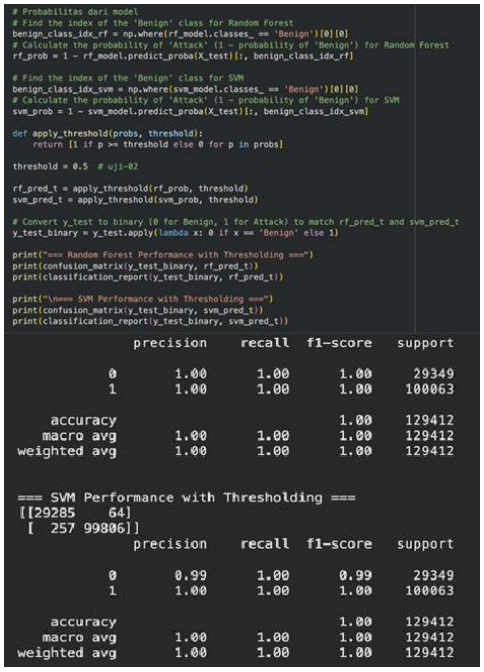
mana sampel dengan probabilitas lebih besar atau sama dengan *threshold* diklasifikasikan sebagai serangan DDoS, sedangkan sisanya dianggap sebagai trafik normal. Variasi *threshold* ini diterapkan untuk menganalisis *trade-off* antara *false positive* dan *false negative*, karena *threshold* yang lebih rendah cenderung meningkatkan sensitivitas deteksi, sementara *threshold* yang lebih tinggi bersifat lebih konservatif namun berpotensi melewatkan serangan. Pendekatan ini sejalan dengan penelitian terdahulu yang menekankan bahwa pemilihan *threshold* merupakan faktor krusial dalam sistem deteksi DDoS berbasis *machine learning* dan tidak dapat ditetapkan secara statis tanpa evaluasi empiris [24]. Hasil probabilitas dengan nilai *threshold* 0.3, 0.5, dan 0.7 dapat di lihat pada Gambar 10 sampai Gambar 12.



```
# Probabilitas dari model
# Find the index of the 'Benign' class for Random Forest
benign_class_idx_rf = np.where(rf_model.classes_ == 'Benign')[0][0]
# Calculate the probability of 'Attack' (1 - probability of 'Benign') for Random Forest
rf_prob = 1 - rf_model.predict_proba(X_test)[:, benign_class_idx_rf]

# Find the index of the 'Benign' class for SVM
benign_class_idx_svm = np.where(svm_model.classes_ == 'Benign')[0][0]
# Calculate the probability of 'Attack' (1 - probability of 'Benign') for SVM
svm_prob = 1 - svm_model.predict_proba(X_test)[:, benign_class_idx_svm]

def apply_threshold(probs, threshold):
    return [1 if p >= threshold else 0 for p in probs]

threshold = 0.3  # uji-01

rf_pred_t = apply_threshold(rf_prob, threshold)
svm_pred_t = apply_threshold(svm_prob, threshold)

# Convert y_test to binary (0 for Benign, 1 for Attack) to match rf_pred_t and svm_pred_t
y_test_binary = y_test.apply(lambda x: 0 if x == 'Benign' else 1)

print("=== Random Forest Performance with Thresholding ===")
print(confusion_matrix(y_test_binary, rf_pred_t))
print(classification_report(y_test_binary, rf_pred_t))

print("\n=== SVM Performance with Thresholding ===")
print(confusion_matrix(y_test_binary, svm_pred_t))
print(classification_report(y_test_binary, svm_pred_t))
```

```
... === Random Forest Performance with Thresholding ===
[[ 29316     33]
 [    25 100038]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     29349
           1       1.00      1.00      1.00    100063

    accuracy                           1.00    129412
   macro avg       1.00      1.00      1.00    129412
weighted avg       1.00      1.00      1.00    129412


=== SVM Performance with Thresholding ===
[[29276    73]
 [  237 99826]]
              precision    recall  f1-score   support

           0       0.99      1.00      0.99     29349
           1       1.00      1.00      1.00    100063

    accuracy                           1.00    129412
   macro avg       1.00      1.00      1.00    129412
weighted avg       1.00      1.00      1.00    129412
```

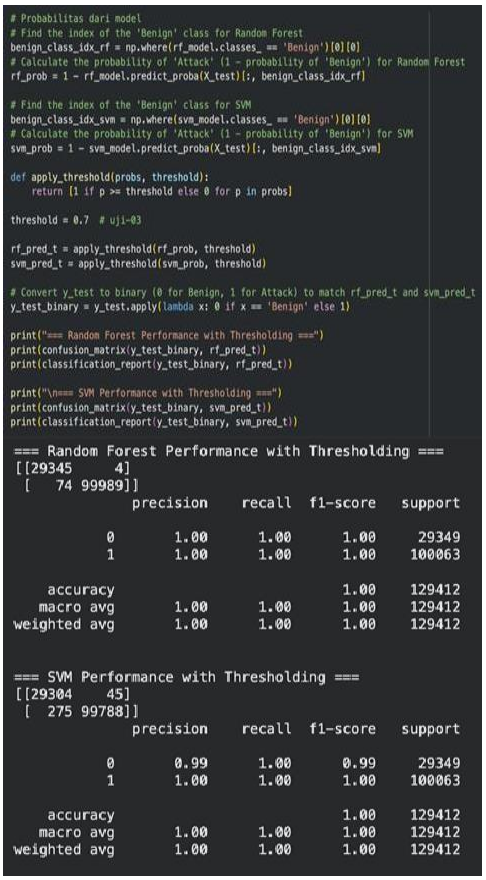
Gbr. 10. *Performent* RF dan SVM dengan *Threshold* 0.3



```
# Probabilitas dari model
# Find the index of the 'Benign' class for Random Forest
benign_class_idx_rf = np.where(rf_model.classes_ == 'Benign')[0][0]
# Calculate the probability of 'Attack' (1 - probability of 'Benign') for Random Forest
rf_prob = 1 - rf_model.predict_proba(X_test)[:, benign_class_idx_rf]

# Find the index of the 'Benign' class for SVM
benign_class_idx_svm = np.where(svm_model.classes_ == 'Benign')[0][0]
# Calculate the probability of 'Attack' (1 - probability of 'Benign') for SVM
svm_prob = 1 - svm_model.predict_proba(X_test)[:, benign_class_idx_svm]

def apply_threshold(probs, threshold):
    return [1 if p >= threshold else 0 for p in probs]

threshold = 0.5  # uji-02

rf_pred_t = apply_threshold(rf_prob, threshold)
svm_pred_t = apply_threshold(svm_prob, threshold)

# Convert y_test to binary (0 for Benign, 1 for Attack) to match rf_pred_t and svm_pred_t
y_test_binary = y_test.apply(lambda x: 0 if x == 'Benign' else 1)

print("=== Random Forest Performance with Thresholding ===")
print(confusion_matrix(y_test_binary, rf_pred_t))
print(classification_report(y_test_binary, rf_pred_t))

print("\n=== SVM Performance with Thresholding ===")
print(confusion_matrix(y_test_binary, svm_pred_t))
print(classification_report(y_test_binary, svm_pred_t))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     29349
           1       1.00      1.00      1.00    100063

    accuracy                           1.00    129412
   macro avg       1.00      1.00      1.00    129412
weighted avg       1.00      1.00      1.00    129412


=== SVM Performance with Thresholding ===
[[29285    64]
 [  257 99806]]
              precision    recall  f1-score   support

           0       0.99      1.00      0.99     29349
           1       1.00      1.00      1.00    100063

    accuracy                           1.00    129412
   macro avg       1.00      1.00      1.00    129412
weighted avg       1.00      1.00      1.00    129412
```

Gbr. 11. *Performent* RF dan SVM dengan *Threshold* 0.5



```
# Probabilitas dari model
# Find the index of the 'Benign' class for Random Forest
benign_class_idx_rf = np.where(rf_model.classes_ == 'Benign')[0][0]
# Calculate the probability of 'Attack' (1 - probability of 'Benign') for Random Forest
rf_prob = 1 - rf_model.predict_proba(X_test)[:, benign_class_idx_rf]

# Find the index of the 'Benign' class for SVM
benign_class_idx_svm = np.where(svm_model.classes_ == 'Benign')[0][0]
# Calculate the probability of 'Attack' (1 - probability of 'Benign') for SVM
svm_prob = 1 - svm_model.predict_proba(X_test)[:, benign_class_idx_svm]

def apply_threshold(probs, threshold):
    return [1 if p >= threshold else 0 for p in probs]

threshold = 0.7  # uji-03

rf_pred_t = apply_threshold(rf_prob, threshold)
svm_pred_t = apply_threshold(svm_prob, threshold)

# Convert y_test to binary (0 for Benign, 1 for Attack) to match rf_pred_t and svm_pred_t
y_test_binary = y_test.apply(lambda x: 0 if x == 'Benign' else 1)

print("=== Random Forest Performance with Thresholding ===")
print(confusion_matrix(y_test_binary, rf_pred_t))
print(classification_report(y_test_binary, rf_pred_t))

print("\n=== SVM Performance with Thresholding ===")
print(confusion_matrix(y_test_binary, svm_pred_t))
print(classification_report(y_test_binary, svm_pred_t))
```

```
=== Random Forest Performance with Thresholding ===
[[29345     4]
 [   74 99989]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     29349
           1       1.00      1.00      1.00    100063

    accuracy                           1.00    129412
   macro avg       1.00      1.00      1.00    129412
weighted avg       1.00      1.00      1.00    129412


=== SVM Performance with Thresholding ===
[[29304    45]
 [  275 99788]]
              precision    recall  f1-score   support

           0       0.99      1.00      0.99     29349
           1       1.00      1.00      1.00    100063

    accuracy                           1.00    129412
   macro avg       1.00      1.00      1.00    129412
weighted avg       1.00      1.00      1.00    129412
```

Gbr. 12. *Performent* RF dan SVM dengan *Threshold* 0.7

Tabel 2. Ringkasan Evaluasi RF dan SVM dengan Nilai *Threshold*

Threshold	Model	Accuracy	Precision	Recall	F1-Score
0.3	*Random Forest*	1.00	1.00	1.00	1.00
	SVM	1.00	0.99	1.00	0.99
0.5	*Random Forest*	1.00	1.00	1.00	1.00
	SVM	1.00	0.99	1.00	0.99
0.7	*Random Forest*	1.00	1.00	1.00	1.00
	SVM	1.00	0.99	1.00	0.99

Hasil evaluasi yang ditunjukan dengan tabel 2 bahwa algoritma *Random Forest* memiliki tingkat stabilitas kinerja yang sangat tinggi, di mana nilai *Accuracy*, *Precision*, *Recall* dan *F1-Score* tetap konsisten sempurna pada seluruh variasi *threshold* yang diuji. Kinerja ini mengindikasikan bahwa margin pemisahan antar kelas pada dataset CICDDoS2019 sudah sangat jelas, sehingga keputusan klasifikasi yang dihasilkan oleh model relatif pasti dan tidak bergantung pada nilai *threshold* probabilitas. Kondisi ini berbeda dengan algoritma *Support Vector Machine*, yang meskipun tetap menghasilkan akurasi yang sangat tinggi, menunjukkan nilai *precision* yang sedikit lebih rendah pada kelas normal (sekitar 0,99). Hal ini berarti masih terdapat sejumlah kecil trafik

normal yang teridentifikasi sebagai serangan (*false positive*), meskipun *recall* tetap sangat tinggi sehingga tidak ada serangan yang terlewatkan.

4. Kesimpulan

Berdasarkan hasil evaluasi, algoritma *Random Forest* menunjukkan kinerja yang lebih unggul dan stabil dibandingkan *Support Vector Machine* dalam mendeteksi serangan DDoS pada dataset CICDDoS2019. *Random Forest* mencapai nilai *accuracy* sebesar 0,99, lebih tinggi dibandingkan SVM yang memperoleh *accuracy* 0,98. Perbedaan kinerja ini semakin terlihat pada nilai *macro average F1-score*, di mana *Random Forest* mencapai nilai 0,89, sedangkan SVM hanya mencapai 0,78. Nilai *macro average* yang lebih tinggi pada *Random Forest* mengindikasikan bahwa model ini mampu mempertahankan performa yang lebih konsisten di seluruh kelas, termasuk kelas serangan dengan jumlah sampel yang relatif kecil. Selain itu, *Random Forest* menunjukkan nilai *Precision* dan *Recall* yang mendekati sempurna pada sebagian besar kelas utama seperti DrDoS_NTP, SYN, dan TFTP, yang mencerminkan kemampuan model dalam meminimalkan kesalahan klasifikasi secara menyeluruh.

Sebaliknya, *Support Vector Machine* menunjukkan sensitivitas yang lebih tinggi terhadap ketidakseimbangan data, yang tercermin dari kegagalan model dalam mengklasifikasikan beberapa kelas minoritas seperti UDPLag dan WebDDoS, dengan nilai *precision*, *recall* dan *F1-score* sebesar 0,00 (tidak terklasifikasi). Pada beberapa kelas serangan lainnya, SVM juga menunjukkan penurunan nilai *recall*, yang mengindikasikan meningkatnya jumlah *false negative*. Kondisi ini berpotensi berbahaya dalam konteks keamanan jaringan karena serangan yang tidak terdeteksi dapat berdampak langsung pada ketersediaan layanan. Dengan demikian, hasil perbandingan ini menunjukkan bahwa *Random Forest* lebih andal untuk digunakan sebagai sistem deteksi DDoS pada lingkungan jaringan berskala besar dan *heterogen*, sementara SVM memerlukan optimasi lanjutan, seperti penanganan ketidakseimbangan data atau penyesuaian parameter, agar dapat mencapai tingkat keandalan yang sebanding.

Analisis variasi *threshold* menunjukkan bahwa *threshold* 0,3 meningkatkan nilai *recall* namun berpotensi meningkatkan *false positive*, sedangkan *threshold* 0,7 meningkatkan *precision* namun berisiko melewatkan serangan yang sebenarnya terjadi. *Threshold* 0,5 memberikan keseimbangan terbaik antara *precision* dan *recall*, yang tercermin dari nilai *F1-score* tertinggi dan paling stabil pada kedua algoritma. Dengan demikian, *threshold* 0,5 direkomendasikan sebagai konfigurasi optimal untuk sistem deteksi DDoS berbasis *Random Forest* dan SVM. Temuan penelitian ini menegaskan bahwa evaluasi berbasis probabilitas dan variasi *threshold* sangat penting untuk meningkatkan keandalan sistem deteksi DDoS dalam menghadapi kondisi trafik jaringan yang dinamis.

Referensi

[1] D. Akgun, S. Hizal, and U. Cavusoglu, "A new DDoS attacks intrusion detection model based on deep learning for cybersecurity," *Comput. Secur.*, vol. 118, p. 102748, Jul. 2022, doi: 10.1016/j.cose.2022.102748.

[2] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy," in *2019 International Carnahan Conference on Security Technology (ICCST)*, IEEE, Oct. 2019, pp. 1–8. doi: 10.1109/CCST.2019.8888419.

[3] M. Rani and N. Kumar, "A neural network based efficient leader–follower formation control approach for multiple autonomous underwater vehicles," *Eng. Appl. Artif. Intell.*, vol. 122, p. 106102, Jun. 2023, doi: 10.1016/j.engappai.2023.106102.

[4] A. Seifousadati, S. Ghasemshirazi, and M. Fathian, "A Machine Learning Approach for DDoS Detection on IoT Devices," *arXiv preprint*, Oct. 2021.

[5] M. N. Faiz, R. H. Maharrani, L. Sari, A. W. Muhammad, and A. R. Supriyono, "Classification of DDoS Attacks based on Network Traffic Patterns Using the k-Nearest Neighbor (k-NN) Algorithm," *Journal of Informatics Information System Software Engineering and Applications (INISTA)*, vol. 7, no. 2, pp. 173–182, May 2025, doi: 10.20895/inista.v7i2.1834.

[6] S.-R. Chen, S.-J. Chen, and W.-B. Hsieh, "Enhancing Machine Learning-Based DDoS Detection Through Hyperparameter Optimization," *Electronics (Basel).*, vol. 14, no. 16, p. 3319, Aug. 2025, doi: 10.3390/electronics14163319.

[7] R. P. Janivasya and I. D. A. Rachmawati, "DDoS Detection using Machine Learning Approach," *Procedia Comput. Sci.*, vol. 245, pp. 1157–1164, 2024, doi: 10.1016/j.procs.2024.10.345.

[8] S. Sadhwani, B. Manibalan, R. Muthalagu, and P. Pawar, "A Lightweight Model for DDoS Attack Detection Using Machine Learning Techniques," *Applied Sciences*, vol. 13, no. 17, p. 9937, Sep. 2023, doi: 10.3390/app13179937.

[9] F. L. Becerra-Suarez, I. Fernández-Roman, and M. G. Forero, "Improvement of Distributed Denial of Service Attack Detection through Machine Learning and Data Processing," *Mathematics*, vol. 12, no. 9, p. 1294, Apr. 2024, doi: 10.3390/math12091294.

[10] S. A. Ajagbe, J. B. Awotunde, and H. Florez, "Intrusion Detection: A Comparison Study of Machine Learning Models Using Unbalanced Dataset," *SN Comput. Sci.*, vol. 5, no. 8, p. 1028, Nov. 2024, doi: 10.1007/s42979-024-03369-0.

[11] D. Spiekermann, T. Eggendorfer, and J. Keller, "Deep Learning for Network Intrusion Detection in Virtual Networks," *Electronics (Basel).*, vol. 13, no. 18, p. 3617, Sep. 2024, doi: 10.3390/electronics13183617.

[12] E. S. Alghoson and O. Abbass, "Detecting Distributed Denial of Service Attacks using Machine Learning Models," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 12, 2021, doi: 10.14569/IJACSA.2021.0121277.

[13] X. Larriva-Novo, V. A. Villagrá, M. Vega-Barbas, D. Rivera, and M. Sanz Rodrigo, "An IoT-Focused Intrusion Detection System Approach Based on Preprocessing Characterization for Cybersecurity Datasets," *Sensors*, vol. 21, no. 2, p. 656, Jan. 2021, doi: 10.3390/s21020656.

[14] H. N. Thanh, "The data preprocessing in improving the classification quality of network intrusion detection systems," *EAI Endorsed Transactions on Context-aware Systems and Applications*, vol. 9, no. 1, Sep. 2023, doi: 10.4108/eetcasa.v9i1.3778.

[15] M. O. , Nurfajri and G. B. Herwanto, "Pendekatan Ensemble Learning untuk klasifikasi serangan DDoS," *Pseudocode*, vol. 12, no. 2, pp. 39–46, 2025.

[16] H. Haeruddin, E. Erick, and H. W. Aripradono, "Perbandingan Support Vector Machine, *Random Forest* Classifier, dan K-Nearest

Neighbour dalam Pendeteksian Anomali pada Jaringan DDos," *JTIM : Jurnal Teknologi Informasi dan Multimedia*, vol. 7, no. 1, pp. 23–33, Jan. 2025, doi: 10.35746/jtim.v7i1.628.

[17] M. A. , Faizin, D. T. , Kurniasari, N. , Elqolby, M. A. R. , Putra, and T. Ahmad, "Optimizing feature selection method in intrusion detection system using thresholding with XGBoost classification," *International Journal of Intelligent Engineering and Systems*, vol. 17, no. 3, pp. 214–226, Jun. 2024, doi: 10.22266/ijies2024.0630.18.

[18] S. Sathyanarayanan and R. Tantri, "Confusion Matrix-Based Performance Evaluation Metrics," *African Journal of Biomedical Research*, pp. 4023–4031, Nov. 2024, doi: 10.53555/AJBR.v27i4S.4345.

[19] D. , Kiswanto, F. , Ramadhani, N. M. , Surbakti, and N. A. Nasution, "Pengembangan dan Implementasi Sistem Deteksi Serangan DDoS Berbasis Algoritma *Random Forest*," *Bulletin of Information Technology (BIT)*, vol. 6, no. 3, pp. 247–256, 2025.

[20] O. Saidani *et al.*, "White blood cells classification using multi-fold pre-processing and optimized CNN model," *Sci. Rep.*, vol. 14, no. 1, p. 3570, Feb. 2024, doi: 10.1038/s41598-024-52880-0.

[21] F. Ferdiansyah, D. Antoni, M. Valdo, M. Mikko, C. Mukmin, and U. Ependi, "Machine Learning Models for DDoS Detection in Software-Defined Networking: A Comparative Analysis," *Journal of Information Systems and Informatics*, vol. 6, no. 3, pp. 1790–1803, Sep. 2024, doi: 10.51519/journalisi.v6i3.864.

[22] M. Cui, J. Chen, X. Qiu, W. Lv, H. Qin, and X. Zhang, "Multi-class intrusion detection system in SDN based on hybrid BiLSTM model," *Cluster Comput.*, vol. 27, no. 7, pp. 9937–9956, Oct. 2024, doi: 10.1007/s10586-024-04477-5.

[23] Z. Chen, M. Simsek, B. Kantarci, M. Bagheri, and P. Djukic, "Machine learning-enabled hybrid intrusion detection system with host data transformation and an advanced two-stage classifier," *Computer Networks*, vol. 250, p. 110576, Aug. 2024, doi: 10.1016/j.comnet.2024.110576.

[24] F. Ahmed, I. A. Sumra, and U. Amil, "A Comprehensive Review on DDoS Attack in Software-Defined Network (SDN): Problems and Possible Solutions," *Journal of Computing & Biomedical Informatics*, vol. 7, no. 1, pp. 353–363, 2024, doi: https://doi.org/10.56979/701/2024.