Laman web jurnal: https://ejournal.unama.ac.id/index.php/processor

Jurnal Processor

P-ISSN: 1907-6738 | E-ISSN: 2528-0082

Analisis Kinerja Algoritma K-Nearest Neighbor dan Random Forest untuk Deteksi Serangan pada Jaringan Perangkat IoT

Muhammad Ilham Mansis[1*], Ferika Syavina Putri[2], Mulia Rohmayati Siregar[3], Kurniabudi[4]

[1,2,3]Fakultas Ilmu Komputer, Program Studi informatika, Universitas Dinamika Bangsa, Jambi, Indonesia.
*Penulis Korespondensi, Email: mansisilham88@gmail.com

Abstrak- Deteksi serangan pada jaringan perangkat Internet of Things (IoT) menjadi tantangan penting dalam menjaga keamanan sistem yang semakin kompleks dan rentan terhadap ancaman siber. Sebagai upaya dalam mengatasi permasalahan tersebut, penelitian ini bertujuan untuk mengevaluasi kinerja algoritma K-Nearest Neighbor (KNN) dan Random Forest dalam mendeteksi berbagai jenis serangan pada jaringan perangkat IoT. Dataset yang digunakan adalah Aposemat IoT-23, yang berisi 1.446.599 entri data lalu lintas jaringan dari berbagai jenis serangan seperti Benign, DDoS, Attack, dan lainnya. Tahapan metode meliputi data preprocessing, data cleaning, label encoding, setelah itu dilakukan pelatihan model dan evaluasi menggunakan metrik accuracy, precision, recall, f1-score, ROC-AUC, serta 5-Fold Cross-Validation. Hasil penelitian menunjukkan bahwa algoritma Random Forest memiliki kinerja lebih baik dibandingkan KNN, dengan F1-Macro Score sebesar 0,9396, ROC-AUC 0,9955, serta accuracy sebesar 92,20%. Sementara itu, KNN mencatatkan F1-Macro Score sebesar 0,9256, ROC-AUC 0,9867, dan accuracy sebesar 92,51%. Random Forest juga menunjukkan performa yang lebih stabil pada semua lipatan validasi silang. Berdasarkan temuan ini, Random Forest dinilai lebih efektif dalam mendeteksi serangan pada jaringan IoT.

Kata Kunci: Internet of Things; Deteksi Serangan; K-Nearest Neighbor (KNN); Random Forest; Aposemat IoT-23.

Abstract−Detecting attacks on Internet of Things (IoT) device networks is an important challenge in maintaining the security of increasingly complex systems that are vulnerable to cyber threats. This study aims to evaluate the performance of the KNearest Neighbor (KNN) and Random Forest algorithms in detecting various types of attacks on IoT networks in a multi-class setting. The dataset used is Aposemat IoT-23, which contains 1,446,599 network traffic data entries from various types of attacks such as Benign, DDoS, Attack, and others. The method stages include data preprocessing, data cleaning, label encoding, followed by model training and evaluation using accuracy, precision, recall, f1-score, ROC-AUC metrics, and 5-Fold Cross-Validation. The results of the study show that the Random Forest algorithm performs better than KNN, with an F1-Macro Score of 0.9396, ROC-AUC of 0.9955, and accuracy of 92.20%. Meanwhile, KNN recorded an F1-Macro Score of 0.9256, ROC-AUC of 0.9867, and accuracy of 92.51%. Random Forest also showed more stable performance across all cross-validation folds. Based on these findings, Random Forest is considered more effective in detecting attacks on IoT networks.

Keywords: Internet of Things; Attack Detection; K-Nearest Neighbor; Random Forest; Aposemat IoT-23.

1. PENDAHULUAN

Evolusi dan perjalanan teknologi digital menunjukkan perjalanan panjang dan transformasional dari teknologi sederhana menuju inovasi yang semakin kompleks dan terpadu[1]. Dalam era digital, peran *Internet of Things* (IoT) dapat dikaitkan dengan kemajuan teknologi, serta kemudahan dalam hal aspek dan koneksi. Konsep *Internet of Things* (IoT) berarti kemampuan objek yang cerdas untuk terhubung ke jaringan internet, yang memungkinkannya melakukan komunikasi dan pertukaran data dengan benda disekitarnya, kondisi lingkungan sekitar, serta perangkat berbasis kecerdasan buatan lainnya[2]. Setiap aspek kehidupan manusia berubah karena perkembangan perangkat *Internet of Things* (IoT). *Internet of Things* (IoT) berfungsi untuk menjembatani dunia digital dengan aktivitas manusia sehingga dapat mempermudah berbagai pekerjaan[3]. IoT telah diterapkan diberbagai sektor industri diantaranya pertanian, kesehatan, logistik dan manufaktur[4].

Dengan kemudahan yang diberikan oleh *Internet of Things* (IoT), pengguna pasti mendapat manfaat. Namun, semua keuntungan *Internet of Things* (IoT) tentunya tidak bisa menghindari ancaman keamanan yang mengintai[5]. Perangkat IoT kerap memiliki kelemahan dari sisi keamanan karena keterbatasan kapasitas pemrosesan, kurangnya pembaruan perangkat lunak secara berkala, dan penerapan fitur proteksi yang belum optimal. Akibatnya, ancaman keamanan pada jaringan *Internet of Things* (IoT) terus meningkat[6]. Ancaman ini dapat terjadi pada perangkat lunak, perangkat keras, jaringan, dan saat diintegrasikan dengan sistem lain.

Badan Siber dan Sandi Negara (BSSN) menerbitkan laporan tahunan “Lanskap Keamanan Siber Indonesia” menurut laporan tersebut, sepanjang tahun 2024 terjadi sebanyak 330.527.636 anomali lalu lintas di seluruh wilayah Indonesia, dengan jenis anomali yang tertinggi yaitu *Mirai Botnet* dengan total sebanyak 81.286.596

Submitted: 07 September 2025; Reviewed : 23 September 2025; Accepted: 23 Oktober 2025; Published: 31 Oktober 2025

aktivitas yang merupakan salah satu jenis serangan yang menargetkan perangkat *Internet of Things* (IoT). Anomali tertinggi tercatat pada bulan Desember dengan 112.085.045 anomali, dan jumlah anomali paling sedikit terjadi pada bulan Mei, yaitu sebanyak 12.273.078 kasus. Aktivitas yang tidak sesuai ini dapat mengurangi kinerja jaringan dan perangkat, pencurian data sensitif, kerusakan reputasi dan kepercayaan perusahaan.

Tingginya jumlah anomali ini menunjukkan bahwa perangkat *Internet of Things* (IoT) rentan terhadap eksploitasi dan bahwa ancaman siber terus meningkat. Kerentanan ini disebabkan perangkat IoT umumnya memiliki sumber daya terbatas, desain keamanan yang lemah, dan sering kali tidak mendukung pembaruan keamanan secara berkala.[7] Banyak perangkat menggunakan password default, komunikasi tanpa enkripsi, serta berasal dari berbagai vendor dengan standar keamanan yang tidak seragam. Kondisi ini memperluas permukaan serangan dan memudahkan peretas mengambil alih perangkat untuk digunakan dalam botnet atau serangan skala besar seperti DDoS[8].

Salah satu metode yang diusulkan para peneliti adalah *Intrusion Detection System (IDS)*. Berbagai algoritma *machine learning* telah digunakan sebagai IDS[9]. Pada penelitian [10] telah membuktikan bahwa algorithma *machine learning* mampu meningkatkan IDS. Begitu pula halnya dengan penelitian *IDS* pada lingkungan *IoT.* Hasil survey telah menunjukkan pemnggunaan *machine learning* dalam mengenali trafik serangan dan menunjukkan hasil yang memuaskan[11]. Namun demikian masih diperlukan penelitian-penelitian lebih lanjut untuk menguji algorithma, khususnya menggunakan dataset yang ideal. Oleh karena itu penelitian ini bertujuan menguji algorithma *k-Nearest Neighbor (kNN)* dan *Random Forest (RF)* untuk mendeteksi DDoS pada dataset Aposemat IoT23. *K-NN* merupakan metode klasifikasi non-parametrik yang tidak memerlukan asumsi distribusional terhadap data dan tidak melibatkan proses pelatihan eksplisit[12]. Pendekatan ini efektif dalam mengenali pola serangan non-linear pada sistem deteksi intrusi (IDS) melalui pengukuran kedekatan berbasis jarak terhadap tetangga terdekat dalam ruang fitur[13]. Selain itu, k-NN bersifat adaptif terhadap perubahan data, karena penambahan sampel baru secara langsung memperbarui model representasi tanpa memerlukan pelatihan ulang. Sementara, *Random Forest* menawarkan ketahanan terhadap *overfitting* melalui agregasi prediksi dari banyak pohon keputusan yang dibangun dari subset data dan fitur acak[14]. Metode ini mampu menangani data berdimensi tinggi dan mengidentifikasi fitur penting secara intrinsik, yang sangat berguna dalam mendeteksi pola serangan yang kompleks pada sistem deteksi intrusi (IDS)[15]. *Random Forest* juga memiliki kemampuan generalisasi yang baik serta robust terhadap noise dan outlier, karakteristik umum dalam lalu lintas jaringan.

Dalam penelitian IDS, diperlukan data yang presentatif, oleh karena itu penelitian ini menggunakan dataset Aposemat IoT23. Dataset Aposemat IoT23 memiliki keunggulan karena merupakan rekaman lalu lintas jaringan IoT nyata dengan skenario serangan *in-the-wild* dan aktivitas normal yang autentik, berbeda dari dataset sintetis generasi sebelumnya[16]. Dataset ini dilengkapi label serangan yang akurat, metadata kontekstual, serta variasi protokol khas perangkat IoT modern[17]. Karakteristik tersebut menjadikan IoT23 lebih representatif dan relevan untuk evaluasi sistem deteksi intrusi berbasis pembelajaran mesin di lingkungan IoT kontemporer.

Penulis menyakini bahwa penelitian ini memberikan kontribusi signifikan dalam pengembangan sistem deteksi intrusi (IDS) untuk lingkungan *Internet of Things (IoT)* melalui evaluasi komparatif yang komprehensif antara dua algoritma *K-NN* dan *RF* menggunakan dataset Aposemat IoT-23 yang realistis dan representatif. Penelitian ini tidak hanya menguji kinerja model berdasarkan metrik standar seperti *accuracy, precision, recall,* dan *F1-score,* tetapi juga memperkuat validitas temuan melalui *5-Fold Cross-Validation* dan analisis *ROC-AUC* dalam skenario klasifikasi multikelas.

2. METODOLOGI PENELITIAN

Pada bagian ini dipaparkan kerangka kerja penelitian sebagai rujukan dalam pelaksanaan penelitian. Bagian ini juga memaparkan rancangan eksperimen yang meliputi dataset, data *preprocessing, data split,* Algoritma *K-Nearest Neighbor* dan *Random Forest, training, testing,* Evaluasi model dan validasi model. Setiap tahapan pada ekperimen dijelaskan secara rinci untuk memberikan wawasan bagaimana eksperimen dilaksanakan.

2.1 Kerangka Kerja Penelitian

Guna memberikan pemahaman yang lebih mendalam terkait tahapan-tahapan yang akan dilakukan dalam penelitian ini, penyusunan kerangka kerja penelitian menjadi aspek yang krusial. Tujuan dari penyusunan kerangka ini adalah untuk menjelaskan secara sistematis langkah-langkah yang diperlukan dalam menyelesaikan permasalahan yang diangkat. Adapun bentuk kerangka kerja penelitian yang digunakan dapat dilihat pada Gambar 1.

Gambar 1. Kerangka Kerja Penelitian

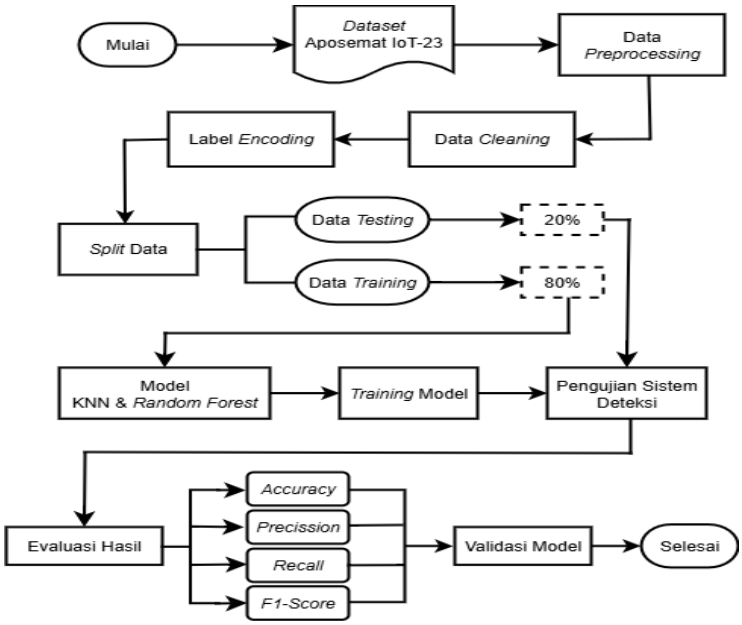
https://doi.org/10.33998/processor.2025.20.2.2594
Submitted: 07 September 2025; Reviewed: 23 September 2025; Accepted: 23 Oktober 2025; Published: 31 Oktober 2025

Berdasarkan pada kerangka kerja penelitian yang telah di gambarkan pada gambar 1 maka dapat di uraikan untuk setiap tahapannya sebagai berikut:

a. Identifikasi Masalah
   Pada tahap ini, penulis melakukan identifikasi masalah utama terkait keamanan pada jaringan perangkat IoT yang rentan terhadap serangan siber, untuk dilakukannya deteksi serangan pada jaringan perangkat IoT. Menggunakan pembelajaran mesin dengan model *K-Nearest Neighbor* dan *Random Forest*, guna mengetahui kinerja model mana yang paling baik dalam melakukan deteksi serangan pada jaringan perangkat IoT.
b. Studi Literatur
   Pada tahap ini, penulis akan mengumpulkan sumber-sumber referensi teori dan konsep dari literatur yang berkaitan dengan topik penelitian. Penulis juga akan mencari data dari berbagai sumber, termasuk buku online, artikel, dan jurnal yang berkaitan dengan topik penelitian. Semua sumber ini akan menghasilkan informasi yang akan digunakan dalam penyelesaian penelitian yang diangkat.
c. Persiapan Data
   Dataset yang digunakan adalah dataset_combined.csv, *dataset* ini merupakan hasil penggabungan dan pembersihan dari file pada himpunan data Aposemat IoT-23 dari Stratosphere IPS, yang berisi lalu lintas jaringan normal dan berbagai jenis serangan, seperti *DDoS*, *Command and Control*, *Okiru*, dan lainnya. Data dipersiapkan melalui tahap *preprocessing*, meliputi penanganan *missing values*, penghapusan nilai yang tidak relevan, serta pengubahan fitur kategorik menjadi numerik menggunakan *encoding* label. Setelah itu, data dibagi menjadi data latih dan data uji dengan rasio 80:20. Selanjutnya, model dilatih dan diuji menggunakan algoritma *K-Nearest Neighbor* dan *Random Forest*.
d. Analisis dan Evaluasi
   Pada tahap ini, algoritma *K-Nearest Neighbor* dan *Random Forest* digunakan untuk membangun model klasifikasi multikelas terhadap lalu lintas jaringan pada *dataset* IoT. Evaluasi dilakukan menggunakan *confusion matrix* untuk melihat distribusi prediksi, serta metrik *accuracy*, *precision*, *recall*, dan *F1-score* guna mengukur kinerja model dalam membedakan antara aktivitas normal dan berbagai jenis serangan.
e. Validasi Model
   Pada tahap ini, metode validasi yang digunakan adalah *5-Fold Cross Validation*. Metode ini membagi *dataset* menjadi lima bagian (*fold*) yang kurang lebih berukuran sama. Pelatihan dan pengujian dilakukan sebanyak lima iterasi, di mana pada setiap putaran satu bagian data digunakan untuk pengujian, dan empat bagian sisanya digunakan untuk pelatihan.

2.2 Rancangan Eksperimen

Adapun rancangan dari alur eksperimen yang akan dilakukan penulis pada penelitian ini yaitu dapat dilihat pada gambar 2.



Gambar 2. Rancangan Eksperimen

https://doi.org/10.33998/processor.2025.20.2.2594
Submitted: 07 September 2025; Reviewed: 23 September 2025; Accepted: 23 Oktober 2025; Published: 31 Oktober 2025

Berdasarkan gambar 2, maka eksperimen pada penelitian ini dapat dijelaskan sebagai berikut.

- Eksperimen menggunakan data yang diperoleh dari dataset Aposemat IoT-23. Dataset ini dilengkapi trafik serangan dan trafik normal (benign)
- Data preprocessing merupakan langkah penting dalam menyiapkan dataset sebelum digunakan dalam pelatihan dan pengujian model machine learning[20]. Tahapan ini bertujuan meningkatkan kualitas data dengan cara menghilangkan data duplikat, menangani missing value, menyederhanakan label kelas, serta memastikan seluruh fitur berada dalam format dan skala yang sesuai. Proses ini dilakukan agar model dapat belajar secara optimal dari data yang tersedia. Tahap ini mencakup beberapa proses penting, antara lain pembersihan data (data cleaning), transformasi label ke bentuk numerik (label encoding). Penjelasan lebih rinci mengenai masing-masing tahapan tersebut disampaikan pada subbab berikut.
- Data Cleaning. Tahap pembersihan data dilakukan untuk memastikan bahwa data yang digunakan dalam proses pelatihan dan pengujian model berada dalam kondisi yang layak dan terbebas dari gangguan kualitas. Proses ini mencakup beberapa langkah penting, seperti pengecekan adanya data duplikat, identifikasi nilai yang hilang (missing values), serta penanganan terhadap nilai-nilai yang tidak logis. Pemeriksaan terhadap data duplikat dilakukan untuk menghindari pengaruh ganda dari entri yang sama, yang dapat menyebabkan bias dalam proses pelatihan model. Selanjutnya, data juga diperiksa dari kemungkinan adanya nilai kosong atau hilang pada fitur-fitur penting, meskipun dalam dataset ini tidak ditemukan nilai kosong secara eksplisit.
- Label Encoding. Pada tahap ini, dilakukan proses label encoding untuk mengubah nilai pada kolom target (label) yang semula bertipe kategori menjadi bentuk numerik. Hal ini diperlukan karena algoritma klasifikasi seperti K-Nearest Neighbor dan Random Forest hanya dapat memproses label target dalam bentuk angka, bukan teks atau string. Oleh karena itu, proses pengkodean dilakukan agar masing-masing kelas dapat dikenali sebagai representasi numerik yang sah oleh model. Proses encoding ini dilakukan menggunakan LabelEncoder dari pustaka scikit-learn, yang memberikan nilai integer unik untuk setiap kategori label secara otomatis. Pemilihan metode ini didasarkan pada kesederhanaan, efisiensi, serta kemampuannya menjaga konsistensi pemetaan label tanpa mengubah makna dari tiap kelas.
- Split Data. Pada tahap ini dilakukan pemisahan data latih (*training set*) dan data uji (*testing set*) dengan perbandingan 80% porsi pelatihan dan 20% untuk pengujian. Proses pembagian ini menggunakan fungsi *train_test_split* dari pustaka *scikit-learn*, disertai dengan parameter *random_state=42* guna menjaga reprodusibilitas hasil. Selain itu, digunakan pula parameter *stratify* agar proporsi distribusi kelas pada data target tetap seimbang di kedua subset. Pemilihan rasio 80:20 didasarkan pada praktik umum dalam *machine learning* yang memberikan keseimbangan optimal antara jumlah data untuk pelatihan dan evaluasi. Data latih digunakan untuk membangun dan melatih model klasifikasi, sementara data uji digunakan untuk mengevaluasi performa model terhadap data yang belum pernah dikenali sebelumnya.
- *Training* model dilakukan untuk membangun kemampuan klasifikasi pada algoritma K-Nearest Neighbour dan Random Forest. Proses ini menggunakan data pelatihan yang merupakan 80% dari total dataset untuk mengekstrak pola dan karakteristik yang relevan dalam menentukan kelas target. Usai tahap pelatihan, evaluasi kinerja dari masing-masing model dilakukan dengan memanfaatkan data pengujian yang berjumlah 20% dari keseluruhan dataset. Data training ini bertujuan untuk mengukur kemampuan generalisasi model terhadap data yang belum pernah dilihat sebelumnya, sehingga dapat diketahui seberapa baik model dalam melakukan klasifikasi pada kondisi nyata.
- Pengujian sistem deteksi dilakukan untuk memvalidasi kinerja model *K-Nearest Neighbour* dan *Random Forest* yang telah dilatih sebelumnya. Tahap pengujian menggunakan data *testing* sebesar 20% dari total *dataset* untuk mengukur tingkat akurasi kedua algoritma dalam melakukan klasifikasi. Proses pengujian meliputi input data baru ke dalam sistem untuk memverifikasi konsistensi output prediksi. Model juga diuji dengan berbagai skenario data untuk mengevaluasi stabilitas dan reliabilitas dalam kondisi penggunaan yang bervariasi.
- Evaluasi model dilakukan dengan membandingkan nilai *accuracy*, *precision*, *recall*, dan *F1-score* antar model, baik secara agregat maupun per kelas. Evaluasi juga dilakukan menggunakan metrik ROC AUC (*Receiver Operating Characteristic – Area Under the Curve*) dengan pendekatan *One-vs-Rest* untuk menilai sejauh mana model mampu membedakan antar kelas. Nilai ROC AUC yang mendekati angka 1 mengindikasikan bahwa model memiliki kemampuan klasifikasi yang baik. Evaluasi ini bertujuan untuk memberikan pemahaman yang lebih mendalam mengenai keunggulan serta keterbatasan masing-masing algoritma dalam mendeteksi serangan pada jaringan perangkat IoT

https://doi.org/10.33998/processor.2025.20.2.2594
Submitted: 07 September 2025; Reviewed: 23 September 2025; Accepted: 23 Oktober 2025; Published: 31 Oktober 2025

– Validasi model untuk mengukur konsistensi performa, guna menilai sejauh mana model dapat mengeneralisasi pola terhadap data yang tidak dikenali sebelumnya. Teknik validasi yang digunakan adalah *5-Fold Cross-Validation*, yaitu metode yang membagi data latih menjadi lima bagian (*fold*) dengan ukuran yang seimbang. Setiap *fold* secara bergiliran digunakan sebagai data validasi, sementara empat *fold* lainnya digunakan sebagai data pelatihan. Proses ini dilakukan sebanyak lima kali, sehingga setiap bagian data mendapat kesempatan yang sama sebagai data validasi. Dengan menggunakan teknik *5-Fold Cross-Validation*, evaluasi model menjadi lebih objektif karena tidak hanya bergantung pada satu pembagian data, serta membantu menilai kestabilan model terhadap variasi data pelatihan dan mencegah *overfitting*. Setiap fold menghasilkan skor evaluasi yang kemudian dianalisis berdasarkan nilai rata-rata, minimum, maksimum, dan standar deviasi untuk mengukur konsistensi performa model. Proses ini dievaluasi menggunakan metrik *F1-Macro Score*, yang dinilai sesuai untuk kasus klasifikasi multi-kelas karena mempertimbangkan keseimbangan antara *precision* dan *recall* pada seluruh kelas.

2.3 Dataset Aposemat IoT-23

Dataset yang digunakan dalam penelitian ini berasal dari Aposemat IoT-23. Kami mempertimbangkan Aposemat IoT-23, merupakan kumpulan data berlabel yang dibuat di laboratorium Avast. Pada dasarnya, tujuan dari kumpulan data besar ini adalah untuk menyediakan serangan IoT yang berlabel dan nyata[18]. Data dikumpulkan dari tahun 2018 hingga 2019. Dataset ini berisi 20 tangkapan dan label malware, dan sisanya tidak berbahaya. Selain itu, dataset ini memiliki 21 atribut fitur[19]. Dataset ini terdiri dari 23 file PCAP-CSV, dengan 20 file merepresentasikan berbagai skenario serangan dan 3 file lainnya menggambarkan lalu lintas jaringan normal. Namun, karena proses penggabungan dan pembersihan seluruh file asli memerlukan waktu yang cukup lama serta sumber daya komputasi yang besar, maka dalam penelitian ini penulis menggunakan beberapa file data saja yang di gabung menjadi satu, dimana dataset tersebut bernama dataset_Combined.csv, yang merupakan hasil penggabungan dan pembersihan dari beberapa file Aposemat IoT-23. Adapun untuk fitur dari *dataset* yang terdapat pada penelitian ini dapat dilihat pada tabel 1.

Tabel 1. Fitur *Dataset* Aposemat IoT-23

No	Fitur	Deskripsi
1	*Ts*	*Timestamp* saat koneksi tercatat
2	*Duration*	Durasi koneksi dalam detik
3	*Orig_bytes*	Jumlah *byte* yang dikirim dari pengirim
4	*Resp_bytes*	Jumlah *byte* yang diterima oleh penerima
5	*Missed_bytes*	Byte yang tidak berhasil ditangkap dalam koneksi
6	*Orig_pkts*	Jumlah paket yang dikirim oleh pengirim
7	*Orig_ip_bytes*	Jumlah *byte* IP yang dikirim oleh pengirim
8	*Resp_pkts*	Jumlah paket yang dikirim oleh penerima
9	*Resp_ip_bytes*	Jumlah *byte* IP yang dikirim oleh penerima
10	*Proto_icmp*	Indikator protokol ICMP (1 jika ICMP, 0 jika tidak)
11	*Proto_tcp*	Indikator protokol TCP (1 jika TCP, 0 jika tidak)
12	*Proto_udp*	Indikator protokol UDP (1 jika UDP, 0 jika tidak)
13	*Conn_state_OTH*	Indikator koneksi dengan status OTH (*Other*)
14	*Conn_state_REJ*	Indikator koneksi ditolak (*Rejected*)
15	*Conn_state_RSTO*	Indikator koneksi di-*reset* oleh *originator*
16	*Conn_state_RSTOS0*	Indikator *reset* oleh *originator*, tanpa respons dari penerima
17	*Conn_state_RSTR*	Indikator *reset* oleh *responder*
18	*Conn_state_RSTRH*	Indikator *reset* oleh *responder* dan *handshake*
19	*Conn_state_S0*	Indikator koneksi inisiasi tanpa respons
20	*Conn_state_S1*	Indikator koneksi berhasil terbentuk (*handshake* selesai)
21	*Conn_state_S2*	Indikator koneksi hanya satu arah (hanya *originator* mengirim data)
22	*Conn_state_S3*	Indikator koneksi tiga arah terbentuk (*handshake* tidak lengkap)
23	*Conn_state_SF*	Indikator koneksi selesai normal dengan 4-*way* FIN
24	*Conn_state_SH*	Indikator koneksi dengan SYN-ACK tanpa ACK (*half-open*)
25	*Conn_state_SHR*	Indikator *half-open connection* di-*reset* oleh *responder*

https://doi.org/10.33998/processor.2025.20.2.2594
Submitted: 07 September 2025; Reviewed: 23 September 2025; Accepted: 23 Oktober 2025; Published: 31 Oktober 2025

2.4 Algoritma *K-Nearest Neighbor*

Algoritma KNN bekerja dengan mengklasifikasikan data mengacu pada jarak kemiripan antara data yang diuji dan data pelatihan terdekatnya. Untuk menghitung tingkat kemiripan tersebut, digunakan perhitungan jarak *Euclidean* dengan persamaan 1[20].

$$d_i \sqrt{\sum_{i=i}^{k} (a_i - b_i)^2} \quad (1)$$

Di mana d_i adalah jarak antara data latih dan data uji, k jumlah tetanggan terdekat, a_i nilai fitur data latih, dan b_i nilai fitur data uji. Algoritma ini tidak melakukan proses pembelajaran secara eksplisit saat pelatihan, melainkan menentukan kelas dari data baru berdasarkan mayoritas tetangga terdekatnya dengan menggunakan parameter k sebagai jumlah tetangga yang dipertimbangkan[21]. Kelebihan *K-Nearest Neighbour* adalah kesederhanaannya dan keefektifannya untuk pola data yang relatif sederhana, namun sensitif terhadap dimensi data yang tinggi dan memerlukan standardisasi fitur.

2.5 Algoritma Random Forest

Random Forest membangun banyak pohon keputusan secara acak dari data pelatihan, kemudian menggabungkan hasilnya melalui mekanisme voting untuk memutuskan kelas akhir[22]. Metode ini unggul dalam menangani data dengan fitur yang kompleks dan dapat mengurangi kesalahan akibat *overfitting* melalui teknik *ensemble learning*[23]. *Random Forest* juga memiliki kemampuan untuk mengevaluasi pentingnya fitur dalam proses klasifikasi. Dalam algoritma *Random Forest*, pembangunan pohon keputusan dilakukan dengan memanfaatkan konsep *entropy* dan *information gain* untuk menentukan atribut yang paling optimal di setiap simpulnya. Nilai *entropy* dihitung menggunakan rumus:

$$Entropy\ (Y) = -\sum_i \rho(c\ |Y)\ log_2\ \rho(c|Y) \quad (2)$$

Di mana Y adalah himpunan data, c kelas dalam Y, dan $\rho(c|Y)$ proporsi kemunculan kelas c. Selanjutnya, untuk memilih atribut yang paling tepat digunakan sebagai pemisah data, digunakan perhitungan information gain dengan rumus:

$$Information\ Gain\ (Y,\ a) = Entropy\ (Y) - \sum v \in Values(a) \frac{Y_v}{Y_a} Entropy\ (Y_v) \quad (3)$$

Dengan a atribut, $Values(a)$ nilai-nilai atribut, Y_v subset data dengan nilai atribut v, dan Y_a jumlah total data dengan atribut a. Kedua model dilatih dengan data pelatihan, lalu diuji dengan data pengujian untuk mengevaluasi seberapa baik kinerjanya dalam mengklasifikasikan data baru[24].

2.6 Pengukuran Kinerja

Model yang digunakan pada penelitian ni dievaluasi menggunakan beberapa nilai metrik, yang memberikan wawasan aspek kinerja model yaitu[25]:

- *True Positive (TP)*: Jumlah data serangan yang diklasifikasikan dengan benar sebagai serangan.
- *False Positive (FP)*: Data normal yang salah diklasifikasikan sebagai serangan (alarm palsu).
- *False Negative (FN)*: Data serangan yang tidak terdeteksi (serangan terlewat).
- *True Negative (TN)*: Data normal yang diklasifikasikan dengan benar.
- *Accuracy* mengukur proporsi prediksi yang benar dari total data. Diukur dengan persamaan 4:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad (4)$$

- *True Positive Rate (TPR)*, atau *Recall*, mengukur kemampuan sistem untuk mendeteksi serangan dengan benar, yaitu proporsi data serangan yang berhasil dikenali[26].

$$TPR = \frac{TP}{TP+FN} \quad (5)$$

- *Precision* menunjukkan seberapa akurat prediksi serangan, yaitu proporsi data yang diprediksi sebagai serangan yang sebenarnya merupakan serangan.

$$\text{Precision} = \frac{TP}{TP+FP} \quad (6)$$

- *F-Measure* adalah rata-rata harmonis antara *Precision* dan *Recall (TPR),* memberikan gambaran tentang keseimbangan antara keduanya[27].

$$F\text{-Measure} = 2 \times \frac{\text{Precision} \times TPR}{\text{Precision} + TPR} \quad (7)$$

- *ROC (Receiver Operating Characteristic)* adalah kurva yang menggambarkan hubungan antara tingkat true positive rate (TPR) dan false positive rate (FPR) dari sebuah model klasifikasi saat ambang keputusan

https://doi.org/10.33998/processor.2025.20.2.2594
Submitted: 07 September 2025; Reviewed: 23 September 2025; Accepted: 23 Oktober 2025; Published: 31 Oktober 2025

berubah[28]. *AUC (Area Under the Curve)* adalah luas area di bawah kurva ROC yang menunjukkan kemampuan model dalam membedakan antara kelas positif dan negatif, dengan nilai AUC yang lebih dekat ke 1 menunjukkan performa yang lebih baik. Secara intuitif, AUC mewakili probabilitas bahwa model akan memberi peringkat lebih tinggi pada contoh positif daripada contoh negatif yang dipilih secara acak[29].

3. ANALISIS DAN HASIL

Pada bagian ini dipaparkan analisa dari hasil ekperimen. Pemaparan diawali dengan menyajikan data ekperimen, hasil tahapan *preprocesing,* hasil pengujian data *training* dan *testing*, evaluasi hasil dan pada bagian akhir dipaparkan hasil validasi model.

3.1 Data Eksperimen

Penelitian ini menggunakan *dataset* Aposemat IoT-23, merupakan kumpulan data yang dikembangkan oleh Stratosphere Laboratory sebagai bagian dari proyek Aposemat. *Dataset* ini dirancang untuk mendukung keamanan siber, khususnya deteksi serangan terhadap perangkat IoT. Dataset ini berisi data lalu lintas jaringan perangkat IoT yang terdiri berbagai jenis label kelas. Tabel 2. menyajikan daftar lengkap fitur yang digunakan dalam *dataset*, termasuk tipe datanya dan contoh nilai yang merepresentasikan isi dari setiap fitur.

Tabel 2. Informasi Fitur Pada *Dataset*

No	Fitur	Tipe Data	Contoh Data
1	*Ts*	Float64	1540469831.302625
2	*Duration*	Float64	0.003497
3	*Orig_bytes*	*Int64*	0
4	*Resp_bytes*	*Int64*	0
5	*Missed_bytes*	Float64	0.0
6	*Orig_pkts*	Float64	5.0
7	*Orig_ip_bytes*	Float64	212.0
8	*Resp_pkts*	Float64	3.0
9	*Resp_ip_bytes*	Float64	144.0
10	*Proto_icmp*	*Int64*	0
11	*Proto_tcp*	*Int64*	1
12	*Proto_udp*	*Int64*	0
13	*Conn_state_OTH*	*Int64*	0
14	*Conn_state_REJ*	*Int64*	0
15	*Conn_state_RSTO*	*Int64*	0
16	*Conn_state_RSTOS0*	*Int64*	0
17	*Conn_state_RSTR*	*Int64*	0
18	*Conn_state_RSTRH*	*Int64*	0
19	*Conn_state_S0*	*Int64*	0
20	*Conn_state_S1*	*Int64*	0
21	*Conn_state_S2*	*Int64*	0
22	*Conn_state_S3*	*Int64*	0
23	*Conn_state_SF*	*Int64*	1
24	*Conn_state_SH*	*Int64*	0
25	*Conn_state_SHR*	*Int64*	0

Selain fitur-fitur yang telah dijelaskan sebelumnya, *dataset* ini juga memiliki kolom label yang merepresentasikan kelas dari setiap sampel data. Kolom ini digunakan sebagai target dalam proses pelatihan dan pengujian model klasifikasi. Setiap label menunjukkan jenis aktivitas atau serangan tertentu dalam jaringan. Informasi lebih lanjut mengenai label yang terdapat dalam *dataset* disajikan pada Tabel 3.

https://doi.org/10.33998/processor.2025.20.2.2594
Submitted: 07 September 2025; Reviewed: 23 September 2025; Accepted: 23 Oktober 2025; Published: 31 Oktober 2025

Tabel 3. Informasi Label Kelas Pada *Dataset*

No	Label Kelas	Deskripsi	Jumlah Data
1	*PartOfAHorizontalPortScan*	Pemindaian *port* secara horizontal untuk mencari kerentanan	825.939
2	*Okiru*	*Malware* varian *Mirai* yang menyerang perangkat IoT	262.690
3	*Benign*	Aktivitas jaringan normal tanpa indikasi serangan	199.756
4	*DDoS*	Serangan yang membanjiri jaringan untuk membuat layanan tidak tersedia	138.777
5	*C&C*	Komunikasi bot dengan server pengendali (*Command and Control*)	15.100
6	*Attack*	Aktivitas serangan umum yang tidak terklasifikasi spesifik	3.915
7	*C&C-HeartBeat*	Sinyal periodik dari bot ke server C&C sebagai tanda aktif	349
8	*C&C-FileDownload*	Pengunduhan file dari server C&C ke perangkat korban	43
9	*C&C-Torii*	*Malware Torii* yang menargetkan perangkat IoT secara tersembunyi	30
	Total		**1.446.599**

3.2 Hasil Tahapan Data Preprocessing

Tahapan data *preprocessing* dilakukan sebelum proses pelatihan dan pengujian model, dengan tujuan untuk meningkatkan kualitas dan konsistensi data. *Preprocessing* merupakan tahap kritis yang dapat mempengaruhi kinerja model secara signifikan. Berikut ini adalah langkah-langkah *preprocessing* yang diterapkan:

a. Data *Cleaning*

Langkah pembersihan data bertujuan untuk mengatasi masalah kualitas data yang dapat mempengaruhi performa model. Proses ini meliputi penanganan *missing values* dan nilai-nilai yang tidak valid. Pada *dataset* yang digunakan, ditemukan nilai -1 pada beberapa atribut numerik seperti *duration, orig_bytes*, dan *resp_bytes*. Nilai tersebut secara logis tidak mungkin terjadi, karena ketiga atribut tersebut merepresentasikan durasi koneksi serta jumlah byte yang dikirim atau diterima, yang secara alami tidak dapat bernilai negatif. Berdasarkan dokumentasi dataset dan pemahaman terhadap konteks jaringan, nilai -1 tersebut diasumsikan sebagai indikator bahwa tidak ada data yang terekam pada sesi koneksi tersebut, atau data tersebut tidak berhasil dikumpulkan secara lengkap.

Untuk menjaga konsistensi dan kestabilan model dalam proses pelatihan, nilai -1 ini tidak dihapus, tetapi digantikan dengan 0 sebagai bentuk penyesuaian. Keputusan ini tidak dilakukan secara manual, melainkan diterapkan melalui skrip otomatis yang memeriksa keberadaan nilai -1 pada fitur-fitur terkait dan menggantinya menggunakan pendekatan programatik.

b. Label *Encoding*

Proses *encoding* dilakukan untuk mengubah label target yang berbentuk kategori menjadi format numerik. *Dataset* memiliki 9 kelas serangan yang berbeda dalam format *string*, sehingga perlu dikonversi menjadi nilai *integer* menggunakan *LabelEncoder* dari *scikit-learn*. Proses ini menghasilkan mapping dari setiap kelas ke nilai numerik 0-8. Tabel 4 menunjukkan hasil pemetaan label *encoding* untuk setiap kelas dalam *dataset*.

Tabel 4. Hasil Pemetaan Label *Encoding*

NO	Label Asli	Nilai Encoding
1	*Attack*	0
2	*Benign*	1
3	*C&C*	2
4	*C&C-FileDownload*	3
5	*C&C-HeartBeat*	4
6	*C&C-Torii*	5
7	*DDoS*	6
8	*Okiru*	7

https://doi.org/10.33998/processor.2025.20.2.2594
Submitted: 07 September 2025; Reviewed: 23 September 2025; Accepted: 23 Oktober 2025; Published: 31 Oktober 2025

NO	Label Asli	Nilai Encoding
9	*PartOfAHorizontalPortScan*	8

3.3 Hasil Pengujian Data *Training* dan *Testing*

Pada tahap ini, *dataset* dibagi menjadi dua bagian utama, yaitu data *training* sebesar 80 dan data *testing* sebesar 20%. Pembagian dilakukan menggunakan metode *stratified split*, yaitu teknik yang memastikan distribusi label pada data tetap proporsional di kedua subset. Artinya, masing-masing jenis serangan tetap terwakili secara seimbang di data *training* dan *testing*. Hal ini bertujuan untuk menjaga kemampuan model dalam mengenali semua kelas secara adil. Setelah dilakukan pemisahan, diperoleh 1.157.279 data pada subset *training* dan 289.320 data pada subset *testing*. Distribusi masing-masing label pada kedua subset disajikan pada Tabel 5.

Tabel 5. Distribusi Label pada Data *Training* dan *Testing*

No	Label	*Train (80%)*	*Test (20%)*
1	*Attack*	3.132	783
2	*Benign*	159.805	39.951
3	*C&C*	12.080	3.020
4	*C&C-FileDownload*	34	9
5	*C&C-HeartBeat*	279	70
6	*C&C-Torii*	24	6
7	*DDoS*	111.022	27.755
8	*Okiru*	210.152	52.538
9	*PartOfAHorizontalPortScan*	660.751	165.188
	Total	**1.157.279**	**289.320**

3.4 Evaluasi Hasil

Setelah model melalui proses pelatihan dan pengujian terhadap model *K-Nearest Neighbor* (KNN) dan *Random Forest*, dilakukan evaluasi perbandingan hasil pada data *training* dan *testing* untuk mengetahui model mana yang memiliki performa yang lebih baik serta lebih stabil dalam klasifikasi serangan IoT. Berikut tabel 6 menyajikan perbandingan antara model *K-Nearest Neighbor* dan *Random Forest* untuk setiap data *training* dan *testing*.

Tabel 6. Performa Model *K-Nearest Neighbor* dan *Random Forest*

Model	Data	*Accuracy*	*Precision Avg*	*Recall Avg*	*F1-Score Avg*	*ROC-AUC*
KNN	*Training*	92,82%	0,9638	0,9584	0,9610	0,9949
KNN	*Testing*	92,51%	0,9578	0,9194	0,9373	0,9867
RF	*Training*	92,72%	0,9781	0,9478	0,9614	0,9971
RF	*Testing*	92,20%	0,9506	0,9432	0,9444	0,9955

Tabel 6 menunjukkan hasil evaluasi performa dari dua algoritma klasifikasi, yaitu *K-Nearest Neighbor* (KNN) dan *Random Forest*, pada data pelatihan dan data pengujian. Evaluasi dilakukan menggunakan lima metrik utama, yaitu *accuracy*, *precision*, *recall*, *f1-score*, dan ROC-AUC.

Berdasarkan data pengujian, model *K-Nearest Neighbor* mencatat nilai akurasi sebesar 0,9251, sedikit lebih tinggi dibandingkan dengan *Random Forest* yang memperoleh akurasi 0,9220. Dari segi precision, *K-Nearest Neighbor* juga unggul dengan nilai 0,9578, menunjukkan bahwa model ini mampu menghasilkan prediksi positif yang lebih tepat dibandingkan *Random Forest* yang memiliki *precision* 0,9506. Namun, dari aspek *recall*, *Random Forest* lebih unggul dengan nilai 0,9432, sementara *K-Nearest Neighbor* memperoleh 0,9194. Hal ini mengindikasikan bahwa *Random Forest* lebih baik dalam mendeteksi seluruh *instance* positif dari masing-masing kelas. Kelebihan *Random Forest* juga terlihat pada nilai *f1-score*, yaitu 0,9444, lebih tinggi dibandingkan *K-Nearest Neighbor* yang memperoleh 0,9373, menandakan keseimbangan yang lebih baik antara *precision* dan *recall*.

Performa paling mencolok ditunjukkan pada metrik ROC-AUC, di mana *Random Forest* memperoleh nilai tertinggi yaitu 0,9955, sedangkan *K-Nearest Neighbor* hanya 0,9867. Nilai ROC-AUC ini mengindikasikan bahwa model *Random Forest* memiliki kemampuan diskriminatif yang lebih kuat dalam membedakan antara kelas-kelas target. Secara keseluruhan, meskipun *K-Nearest Neighbor* menunjukkan hasil yang kompetitif pada beberapa metrik seperti *accuracy* dan *precision*, namun model *Random Forest* memberikan performa yang lebih seimbang dan unggul terutama pada *recall*, *f1-score*, dan ROC-AUC. Oleh karena itu, *Random Forest* dapat dianggap sebagai model yang lebih optimal dan andal dalam tugas klasifikasi pada *dataset* ini.

https://doi.org/10.33998/processor.2025.20.2.2594
Submitted: 07 September 2025; Reviewed: 23 September 2025; Accepted: 23 Oktober 2025; Published: 31 Oktober 2025

3.5 Validasi Model

Pada bagian ini dipaparkan hasil validasi model dilakukan terhadap algoritma *K-NN* dan *Random-Forest.* Validasi model dilakukan untuk menunjukkan kehandalan *K-NN* dan *Random Forest* dalam mendeteksi serangan pada dataset Aposemat IoT-23. Validasi dilakukan menggunakan *5-Fold Cross-Validation.* Tabel 7 menunjukkan hasil evaluasi model *K-Nearest Neighbor* menggunakan *5-Fold Cross-Validation*, yang disajikan dalam bentuk *F1-Macro Score* untuk setiap *fold*.

Tabel 7. *5-Fold Cross-Validation* KNN (*F1 Macro Score*)

Fold	***F1-Macro Score***
Fold 1	0,9338
Fold 2	0,8988
Fold 3	0,9412
Fold 4	0,9355
Fold 5	0,9187
Rata-rata	0,9256
Minimum	0,8988
Maksimum	0,9412
Standar Deviasi	0,0153

Berdasarkan hasil validasi model KNN menggunakan teknik *5-Fold Cross-Validation*, diperoleh nilai *F1-Macro Score* rata-rata sebesar 0,9256. Nilai tertinggi terdapat pada *Fold* ke-3 sebesar 0,9412, sedangkan nilai terendah terjadi pada *Fold* ke-2 sebesar 0,8988. Dengan standar deviasi sebesar 0,0153, dapat disimpulkan bahwa performa model cukup stabil di tiap *fold* dengan variasi yang tidak terlalu besar.

3.6 Validasi Model *Random Forest*

Setelah melakukan validasi pada model *K-Nearest Neighbor*, proses yang sama juga diterapkan pada model *Random Forest* untuk mengevaluasi kemampuan generalisasi model terhadap data yang belum pernah dilihat sebelumnya. Teknik 5-*Fold Cross-Validation* kembali digunakan dengan membagi data menjadi lima bagian, di mana empat bagian digunakan untuk pelatihan dan satu bagian untuk validasi secara bergiliran. Hasil evaluasi model *Random Forest* menggunakan *5-Fold Cross-Validation* ditampilkan pada Tabel 8, dengan metrik yang digunakan berupa *F1-Macro Score* untuk setiap *fold*.

Tabel 8. *5-Fold Cross-Validation Random Forest* (*F1 Macro Score*)

Fold	***F1-Macro Score***
Fold 1	0,9499
Fold 2	0,9212
Fold 3	0,9587
Fold 4	0,9446
Fold 5	0,9234
Rata-rata	0,9396
Minimum	0,9212
Maksimum	0,9587
Standar Deviasi	0,0148

Nilai rata-rata yang mencapai 0,9396 serta standar deviasi yang rendah 0,0148 menunjukkan bahwa model *Random Forest* memiliki performa yang stabil dan tidak mengalami *overfitting* pada data latih. Dengan kata lain, model ini memiliki potensi yang baik untuk digunakan dalam prediksi terhadap data baru.

4. KESIMPULAN

Berdasarkan hasil evaluasi yang telah dilakukan terhadap model *K-Nearest Neighbor* dan *Random Forest*, baik melalui data pelatihan, data pengujian, maupun validasi menggunakan *5-Fold Cross-Validation*, dapat disimpulkan bahwa *Random Forest* menunjukkan performa yang lebih optimal secara keseluruhan. Pada tahap validasi silang, model KNN memperoleh *F1-Macro Score* rata-rata sebesar 0,9256 dengan standar deviasi 0,0153, sementara model *Random Forest* mencatatkan rata-rata *F1-Macro Score* sebesar 0,9396 dan standar deviasi yang lebih

https://doi.org/10.33998/processor.2025.20.2.2594
Submitted: 07 September 2025; Reviewed: 23 September 2025; Accepted: 23 Oktober 2025; Published: 31 Oktober 2025

rendah, yaitu 0,0148. Hal ini menunjukkan bahwa *Random Forest* memiliki kestabilan performa yang lebih baik antar *fold* dibandingkan KNN.

Selain itu, hasil evaluasi pada data pengujian menunjukkan bahwa *Random Forest* mencapai *F1-Score* rata-rata 0,9444 dan ROC-AUC sebesar 0,9955, lebih tinggi dibandingkan KNN yang mencatatkan *F1-Score* 0,9373 dan ROC-AUC 0,9867. Dari sisi *accuracy*, *precision*, dan *recall*, kedua model menunjukkan nilai yang kompetitif, namun *Random Forest* cenderung lebih konsisten dan unggul dalam membedakan antar kelas. Dengan mempertimbangkan seluruh hasil evaluasi tersebut, *Random Forest* dapat disimpulkan sebagai model klasifikasi yang paling tepat dan andal untuk digunakan dalam penelitian ini, khususnya dalam mendeteksi serangan pada jaringan perangkat IoT secara akurat dan stabil.

REFERENCES

[1] V. Maria, S. D. Rizky, and A. M. Akram, "Mengamati perkembangan teknologi dan bisnis digital dalam transisi menuju era industri 5.0," *Wawasan J. Ilmu Manajemen, Ekon. Dan Kewirausahaan*, vol. 2, no. 3, pp. 175–187, 2024.

[2] F. Nahdi and H. Dhika, "Analisis Dampak Internet of Things (IoT) Pada Perkembangan Teknologi di Masa Yang Akan Datang," *INTEGER J. Inf. Technol.*, vol. 6, no. 1, 2021.

[3] R. Fadillah and I. N. Fitriyani, "Edukasi Tentang Pemanfaatan Internet dan Teknologi Internet Of Things (IoT) di Kelurahan Padang Matinggi, Kecamatan Rantau Utara," *Sevaka Has. Kegiat. Layanan Masy.*, vol. 3, no. 1, pp. 130–136, 2025.

[4] E. Erwin *et al.*, *Pengantar Penerapan Internet Of Things: Konsep Dasar Penerapan IoT di berbagai Sektor*. PT. Sonpedia Publishing Indonesia, 2023.

[5] W. Wulan *et al.*, "Tinjauan Ancaman dan Risiko pada Sistem Keamanan Internet of Things, Berbasis Cloud Computing dalam Penggunaan E-Commerce dan Rencana Strategis," *J. Kewirausahaan dan Multi Talent.*, vol. 2, no. 2, pp. 126–137, 2024.

[6] D. R. Adhy, N. Anwar, S. Maesaroh, and R. Hermawan, *Machine Learning dan Internet of Things (Iot): Implementasi Machine Learning dalam Internet of Things*. PT. Star Digital Publishing, Yogyakarta-Indonesia, 2025.

[7] X. Yao, F. Farha, R. Li, I. Psychoula, L. Chen, and H. Ning, "Security and privacy issues of physical objects in the IoT: Challenges and opportunities," *Digit. Commun. Networks*, vol. 7, no. 3, pp. 373–384, 2021.

[8] M. Gelgi, Y. Guan, S. Arunachala, M. Samba Siva Rao, and N. Dragoni, "Systematic literature review of IoT botnet DDOS attacks and evaluation of detection techniques," *Sensors*, vol. 24, no. 11, p. 3571, 2024.

[9] E. E. Abdallah, W. Eleisah, and A. F. Otoom, "Intrusion Detection Systems using Supervised Machine Learning Techniques: A survey," *Procedia Comput. Sci.*, vol. 201, no. C, pp. 205–212, 2022, doi: 10.1016/j.procs.2022.03.029.

[10] A. H. Azizan *et al.*, "A machine learning approach for improving the performance of network intrusion detection systems," *Ann. Emerg. Technol. Comput.*, vol. 5, no. Special issue 5, pp. 201–208, 2021, doi: 10.33166/AETiC.2021.05.025.

[11] S. V. N. Santhosh Kumar, M. Selvi, and A. Kannan, "A Comprehensive Survey on Machine Learning-Based Intrusion Detection Systems for Secure Communication in Internet of Things," *Comput. Intell. Neurosci.*, vol. 2023, no. 1, 2023, doi: 10.1155/2023/8981988.

[12] D. Ortiz-Villaseñor, G. Trujillo-Hernández, O. Real-Moreno, M. J. Castro-Toscano, L. D. Medina-Madrazo, and D. Barrera-Román, "K-nearest neighbors regression and applications," in *Exploring Psychology, Social Innovation and Advanced Applications of Machine Learning*, IGI Global Scientific Publishing, 2025, pp. 295–316.

[13] S. M. Hussein and A. M. Ashir, "Machine Learning-Driven Intrusion Detection Systems: Reducing False Alarms and Enhancing Accuracy," *EURASIAN J. Sci. Eng.*, vol. 10, no. 3, pp. 85–96, 2024.

[14] E. Halabaku and E. Bytyçi, "Overfitting in Machine Learning: A Comparative Analysis of Decision Trees and Random Forests.," *Intell. Autom. \& Soft Comput.*, vol. 39, no. 6, 2024.

[15] M. A. Talukder, M. Khalid, and N. Sultana, "A hybrid machine learning model for intrusion detection in wireless sensor networks leveraging data balancing and dimensionality reduction," *Sci. Rep.*, vol. 15, no. 1, p. 4617, 2025.

[16] A. Khan and I. Sharma, "Guardians of the IoT: A Symphony of Ensemble Learning for DDoS Attack Resilience," in *2023 4th International Conference on Computation, Automation and Knowledge Management (ICCAKM)*, 2023, pp. 1–6.

[17] A. Oachesu, "Enhancing Smart Home Security with Generative AI: Threat Detection and Countermeasure Design." 2025.

[18] M. J. Gul and M. K.-R. Raazi Syed, "Network Attack Detection in IoT using Artificial Intelligence," in *2023 International Multi-disciplinary Conference in Emerging Research Trends (IMCERT)*, 2023, pp. 1–6. doi: 10.1109/IMCERT57083.2023.10075102.

[19] R. Rajalingam and K. Kavitha, "Original Research Article An efficient network optimized machine learning architecture framework for detection of malwares in IOT (NB-IOT) systems," *J. Auton. Intell.*, vol. 7, no. 5, 2024.

[20] M. Mohy-Eddine, A. Guezzaz, S. Benkirane, and M. Azrour, "An efficient network intrusion detection model for IoT security using K-NN classifier and feature selection," *Multimed. Tools Appl.*, vol. 82, no. 15, pp. 23615–23633, 2023.

[21] A. I. Saleh, F. M. Talaat, and L. M. Labib, "A hybrid intrusion detection system (HIDS) based on prioritized k-nearest neighbors and optimized SVM classifiers," *Artif. Intell. Rev.*, vol. 51, no. 3, pp. 403–443, 2019, doi: 10.1007/s10462-017-9567-1.

[22] M. Alduailij, Q. W. Khan, M. Tahir, M. Sardaraz, M. Alduailij, and F. Malik, "Machine-Learning-Based DDoS Attack Detection Using Mutual Information and Random Forest Feature Importance Method," *Symmetry (Basel).*, vol. 14,

https://doi.org/10.33998/processor.2025.20.2.2594
Submitted: 07 September 2025; Reviewed: 23 September 2025; Accepted: 23 Oktober 2025; Published: 31 Oktober 2025

no. 6, pp. 1–15, 2022, doi: 10.3390/sym14061095.
[23] A. Harris and A. Rahim, "Seleksi Fitur dengan Information Gain untuk Meningkatkan Deteksi Serangan DDoS Menggunakan Random Forest An Information Gain Feature Selection to Improve DDoS Detection using Random Forest," *Februari*, vol. 19, no. 1, pp. 56–66, 2020.
[24] Y. Chen, J. Hou, Q. Li, and H. Long, "DDoS attack detection based on random forest," in *Proceedings of 2020 IEEE International Conference on Progress in Informatics and Computing, PIC 2020*, 2020, pp. 328–334. doi: 10.1109/PIC50277.2020.9350788.
[25] O. Almomani, "A feature selection model for network intrusion detection system based on pso, gwo, ffa and ga algorithms," *Symmetry (Basel).*, vol. 12, no. 6, pp. 1–20, Jun. 2020, doi: 10.3390/sym12061046.
[26] A. Abd and A. Hadi, "Performance Analysis of Big Data Intrusion Detection System over Random Forest Algorithm," *Int. J. Appl. Eng. Res.*, vol. 13, no. 2, pp. 1520–1527, 2018, [Online]. Available: http://www.ripublication.com
[27] H. A. Ahmed, A. Hameed, and N. Z. Bawany, "Network intrusion detection using oversampling technique and machine learning algorithms," *PeerJ Comput. Sci.*, vol. 8, p. e820, Jan. 2022, doi: 10.7717/peerj-cs.820.
[28] S. Huang and K. Lei, "IGAN-IDS: An imbalanced generative adversarial network towards intrusion detection system in ad-hoc networks," *Ad Hoc Networks*, vol. 105, 2020, doi: 10.1016/j.adhoc.2020.102177.
[29] M. N. Chowdhury and K. Ferens, "A Support Vector Machine Cost Function in Simulated Annealing for Network Intrusion Detection," *Adv. Sci. Technol. Eng. Syst. J.*, vol. 4, no. 3, 2019, doi: 10.25046/aj040334.

https://doi.org/10.33998/processor.2025.20.2.2594
Submitted: 07 September 2025; Reviewed: 23 September 2025; Accepted: 23 Oktober 2025; Published: 31 Oktober 2025