STARNAS 2025
Seminar Teknologi dan Sains Nasional

Optimasi Model Random Forest pada Sistem Deteksi Serangan Jaringan IoT dengan Seleksi Fitur dan Hyperparameter Tuning

Fayruz Rahma
Jurusan Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia
fayruz.rahma@uii.ac.id

Abstract
The Internet of Things (IoT) has become an integral part of various aspects of life, from smart homes to critical infrastructure. However, the increasing adoption of IoT also brings significant security challenges, including the growing number of cyberattacks. Therefore, intrusion detection in IoT networks is crucial for identifying and preventing potential threats. This study develops an attack detection model using the Random Forest (RF) algorithm, trained with the ToN_IoT dataset. To enhance model efficiency and accuracy, feature selection is applied using Recursive Feature Elimination (RFE), while parameter optimization is performed through Grid Search with 3-fold cross-validation. The resulting model exhibits high performance, with all evaluation scores exceeding 0.99 based on the Confusion Matrix and AUC-ROC, demonstrating its effectiveness in detecting attacks.
Keywords: *network intrusion detection system; IoT; random forest; feature selection; hyperparameter tuning*

Abstrak
Internet of Things (IoT) telah menjadi bagian penting dalam berbagai aspek kehidupan, mulai dari *smart home* hingga infrastruktur kritis. Namun, peningkatan adopsi IoT juga membawa tantangan keamanan yang signifikan, termasuk serangan siber yang semakin meningkat. Oleh karena itu, deteksi intrusi dalam jaringan IoT menjadi hal yang krusial untuk mengidentifikasi dan mencegah ancaman potensial. Penelitian ini mengembangkan model deteksi serangan menggunakan algoritma Random Forest (RF) yang dilatih dengan dataset ToN_IoT. Untuk meningkatkan efisiensi dan akurasi model, diterapkan seleksi fitur menggunakan Recursive Feature Elimination (RFE) serta optimasi parameter melalui Grid Search dengan 3-fold cross-validation. Model yang dihasilkan menunjukkan performa tinggi dengan semua skor evaluasi di atas 0.99 berdasarkan Confusion Matrix dan AUC-ROC, membuktikan efektivitasnya dalam mendeteksi serangan.
Keywords: *sistem deteksi intrusi jaringan; IoT; random forest; seleksi fitur; hyperparamter tuning*

1. PENDAHULUAN

Internet of Things (IoT) telah menjadi bagian penting dalam kehidupan modern, mulai dari *smart home* hingga infrastruktur kritis seperti transportasi dan kesehatan. Peningkatan adopsi IoT berlangsung sangat pesat, tetapi tidak diimbangi dengan kesiapan keamanan yang memadai. Kompleksitas desain dan implementasi perangkat yang heterogen menjadikan keamanan sebagai salah satu tantangan terbesar dalam sistem IoT (Moustafa, Ahmed, & Ahmed, 2021). Kondisi ini tercermin dari tingginya intensitas serangan, misalnya perangkat *smart home* rata-rata mendapat sekitar 10 serangan setiap 24 jam

(NETGEAR Security Team, 2024) Dibandingkan dengan tahun-tahun sebelumnya, serangan keamanan IoT meningkat hingga 107% pada awal tahun 2024 (O'Shea, 2024). Situasi tersebut menunjukkan bahwa deteksi intrusi menjadi elemen vital dalam menjaga integritas, kerahasiaan, dan ketersediaan data pada jaringan IoT.

Di tengah tingginya risiko serangan, penerapan *machine learning* menjadi solusi yang banyak digunakan untuk mendeteksi aktivitas berbahaya secara otomatis. Dengan pendekatan *supervised learning*, model tidak perlu diprogram secara eksplisit, tetapi akan mempelajari pola-pola serangan dari data historis untuk mengenali ancaman yang telah diketahui maupun serangan baru yang polanya serupa (Rahma, Rachmadi, Pratomo, & Purnomo, 2023). Model *machine learning* yang diterapkan pada Network Intrusion Detection System (NIDS) memungkinkan pemantauan trafik pada level jaringan sehingga perangkat IoT tetap terlindungi meskipun memiliki keterbatasan sumber daya. Selain itu, model *mechine learning* mampu mendeteksi ancaman dalam jumlah data besar dan jenis serangan yang beragam.

Untuk menjawab tantangan tersebut, penelitian ini mengusulkan pengembangan model deteksi intrusi pada jaringan IoT menggunakan algoritma Random Forest (RF). Algoritma ini dipilih karena telah menunjukkan performa yang baik pada berbagai penelitian terdahulu pada dataset berbeda (Li, Ma, Li, & Li, 2023) (Ndichu, Ban, Takahashi, & Inoue, 2023). Penelitian ini menggunakan dataset ToN_IoT yang memiliki 43 fitur sehingga dilakukan seleksi fitur menggunakan Recursive Feature Elimination (RFE) untuk mengurangi kompleksitas pelatihan sekaligus mempertahankan fitur paling relevan dalam memprediksi serangan. Setelah proses seleksi menghasilkan 25 fitur terbaik, dilakukan *hyperparameter tuning* dengan Grid Search yang dievaluasi menggunakan *3-fold cross validation* agar hasil tidak bias ke potongan data tertentu. Kinerja model kemudian dievaluasi menggunakan *confusion matrix* (akurasi, *precision*, *recall*, *F1-score*) serta Area Under Curve (AUC-ROC) sehingga efektivitas solusi dapat diukur secara komprehensif dan objektif.

2. KERANGKA TEORI

2.1. Sistem Deteksi Serangan Jaringan

Serangan jaringan dapat menimbulkan kerusakan yang serius. Oleh karena itu, langkah dan kontrol keamanan jaringan menjadi sangat penting untuk mencegah atau memitigasi serangan jaringan. Salah satu cara untuk mengendalikan keamanan jaringan adalah melalui deteksi intrusi jaringan. Berdasarkan metode penerapannya, *intrusion detection system* (IDS) dikategorikan menjadi *host-based* IDS dan *network-based* IDS (Ahmad, Khan, Shiang, Abdullah, & Ahmad, 2020). *Host-based* IDS dipasang pada perangkat akhir (seperti komputer desktop atau server), sedangkan *network-based* IDS umumnya diimplementasikan pada sisi perimeter jaringan. Penelitian ini berfokus pada *network-based* IDS.

Pada tahap awal pengembangannya, IDS menggunakan pola atau tanda tangan serangan yang telah dikenal untuk mengidentifikasi serangan pada jaringan. Namun, pendekatan ini tidak mampu mendeteksi

jenis serangan baru. Seiring perkembangannya, IDS mulai memanfaatkan *machine learning* dan analisis perilaku untuk memantau lalu lintas jaringan.

2.2. Random Forest

Random Forest merupakan algoritma *machine learning* yang dirancang untuk melakukan klasifikasi pada dataset berukuran besar. Algoritma ini mampu menangani data dengan berbagai dimensi dan skala, sekaligus memberikan performa yang tinggi. Proses klasifikasi dilakukan dengan menggabungkan banyak pohon keputusan (*decision tree*) yang dibangun menggunakan data latih yang tersedia. Random Forest memanfaatkan pohon keputusan untuk menjalankan proses seleksi data. Pohon-pohon ini dibagi secara rekursif berdasarkan data yang memiliki kesamaan kelas. Semakin banyak pohon yang digunakan, akurasi hasil cenderung semakin meningkat. Penentuan hasil klasifikasi dilakukan melalui proses *voting* berdasarkan pohon-pohon yang telah terbentuk (Rohmah, 2024).

2.3. Seleksi Fitur

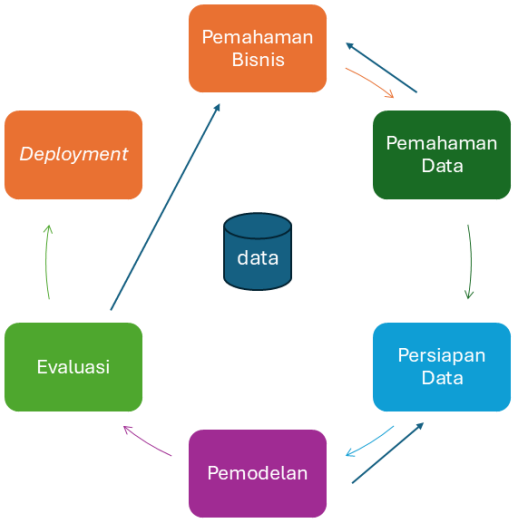
"Fitur" merujuk pada properti atau karakteristik terukur dari suatu data, yaitu atribut spesifik yang membantu menggambarkan fenomena yang sedang diamati (Belcic & Stryker, 2025). Seleksi fitur adalah proses memilih fitur-fitur paling relevan dari suatu dataset untuk digunakan dalam membangun dan melatih model *machine learning*. Dengan mengurangi ruang fitur menjadi subset yang dipilih, seleksi fitur dapat meningkatkan kinerja model kecerdasan buatan sekaligus menurunkan kebutuhan komputasi.

2.4. Hyperparameter Tuning

Dalam proses pelatihan model *machine learning*, setiap dataset dan model memerlukan kumpulan *hyperparameter* yang berbeda, yaitu sekumpulan variabel yang menentukan cara model bekerja. Penentuan *hyperparameter* hanya dapat dilakukan melalui serangkaian percobaan, dengan memilih suatu set *hyperparameter* dan menjalankannya pada model. Proses ini disebut *hyperparameter tuning* (AWS, 2025). Pada dasarnya, model dilatih secara berurutan dengan berbagai kombinasi *hyperparameter*. Proses ini dapat dilakukan secara manual atau menggunakan metode otomatis yang tersedia.

3. METODOLOGI

Penelitian ini mengadopsi metodologi Cross Industry Standard Process for Data Mining (CRISP-DM). Terdapat enam tahapan dalam CRISP-DM (Gambar 1), yaitu pemahaman bisnis (*business understanding*), pemahaman data (*data understanding*), persiapan data (*data preparation*), pemodelan (*modeling*), evaluasi (*evaluation*), dan *deployment* (Alwi, 2024). Karena penelitian ini tidak diimplementasikan langsung dan hanya untuk kebutuhan riset, tahapan *deployment* tidak dilakukan.



Gambar 1. Tahapan dalam metodologi CRISP-DM (Alwi, 2024)

3.1. Pemahaman Bisnis

Untuk memahami kasus dan rencana solusi yang akan dilakukan, disusun pernyataan masalah (*problem statements*), tujuan (*goals*), dan pernyataan solusi (*solution statements*).

3.2. Pemahaman Data

Data yang dipakai dalam proyek ini adalah Dataset TON_IoT (Moustafa, The TON_IoT Datasets, 2021). Dataset TON_IoT diciptakan untuk mengevaluasi *fidelity* dan efisiensi berbagai aplikasi keamanan siber berbasis Kecerdasan Buatan (AI) pada jaringan IoT. Potongan dataset khusus untuk *training* & *testing* disediakan pada tautan OneDrive, pada *folder* Train_Test_datasets > Train_Test_network_dataset > train_test_network.csv.

Pemahaman data dilakukan dengan membaca referensi dan mengeksplorasi data menggunakan fungsi info(), head(), dan describe() yang telah tersedia pada *library* Pandas.

3.3. Persiapan Data

Beberapa tahap persiapan data dilakukan, antara lain:

1. Drop Fitur yang Tidak Perlu

   Beberapa fitur tidak diperlukan pada pelatihan model, seperti: alamat IP ('src_ip' & 'dst_ip') dan nomor port ('src_port' & 'dst_port'). Model diharapkan mengenali pola trafik jaringan normal/serangan, terlepas dari asal/tujuan serangan. Pada dunia nyata, penyerang seringkali menggunakan alamat IP dan port yang berubah-ubah sehingga model tidak perlu mengenali alamat IP dan port dalam mendeteksi serangan. Penghapusan alamat IP dan nomor port ketika melatih model juga direkomendasikan oleh pembuat dataset dalam papernya (Moustafa, A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets, 2021).

   Selain itu, atribut jenis serangan ('type') dihapus karena model akan fokus belajar mendeteksi serangan, bukan mengklasifikasi jenis serangan. *Violation features* ('weird_name', 'weird_addl',

'weird_notice') juga tidak diperlukan dalam melatih model karena pada *real network data* fitur ini tidak ada.

2. Ubah Tipe Objek ke Tipe Angka (Encoding)

Teknik Label Encoding digunakan untuk mengubah nilai kategorikal menjadi nilai numerik. Teknik ini dipilih karena simpel. Teknik *one-hot encoding* tidak digunakan karena banyaknya fitur yang perlu diubah ke tipe angka. Jika *one-hot encoding* digunakan, fitur bisa bertambah sampai ratusan. Ini dapat membuat proses komputasi pelatihan model menjadi lebih berat dan kurang efisien. Maka, digunakanlah teknik *label encoding*.

3. Memisahkan Variabel X dan Y

Dilakukan pemisahan variabel X dan y. Target (variabel y) diambil dari kolom 'label'. Kolom ini berisi angka 0 (menunjukkan trafik normal) dan 1 (menunjukkan trafik serangan).

4. Seleksi Fitur

Seleksi fitur dilakukan karena terlalu banyaknya fitur pada dataset dapat membuat durasi pelatihan model menjadi lama. Seleksi fitur dilakukan menggunakan teknik Recursive Feature Elimination (RFE) karena mudah dikonfigurasi dan digunakan. Selain itu, RFE efektif dalam memilih fitur (kolom) dalam kumpulan data pelatihan yang lebih atau paling relevan dalam memprediksi variabel target (Brownlee, 2020). Sebanyak 25 fitur dipilih. Jumlah ini sama dengan jumlah seleksi fitur yang dilakukan pembuat dataset ketika dia mengembangkan model deteksi serangan untuk mengecek kualitas dataset yang dia buat (Moustafa, A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets, 2021).

5. Normalisasi Fitur Angka

Proses normalisasi hanya dilakukan pada fitur yang aslinya sudah berbentuk angka, bukan fitur kategorikal. Normalisasi diperlukan karena rentang nilai fitur yang sangat beragam. Normalisasi juga diterapkan agar model kelak tidak bias pada fitur tertentu yang bernilai besar. Normalisasi dilakukan menggunakan fungsi MinMaxScaler().

6. Splitting Data.

Dilakukan pemisahan data *training* dan data *testing* dengan rasio 8:2.

3.4. Pemodelan

Model deteksi serangan pada jaringan IoT ini dikembangkan menggunakan algoritma Random Forest (RF) karena algoritma ini menunjukkan performa yang baik pada sistem deteksi serangan lainnya dengan dataset yang berbeda (contoh: model RF dengan dataset CIC-IDS2017 (Li, Ma, Li, & Li, 2023), model RF dengan data serangan nyata pada suatu perusahaan (Ndichu, Ban, Takahashi, & Inoue, 2023)). Untuk mendapatkan parameter terbaik, dilakukan *hyperparameter tuning* menggunakan metode Grid Search. Metode ini mencoba semua kombinasi parameter berbeda yang dimasukkan pada parameter *grid* dan menghasilkan kombinasi parameter terbaik (Okamura, 2020). Evaluasi kombinasi parameter

menggunakan *3-fold cross validation* agar penilaian performa model tidak bias ke potongan data training tertentu.

1. Hyperparameter Tuning dengan Grid Search

   Algoritma Random Forest memiliki beberapa parameter yang bisa diatur untuk mendapatkan performa terbaik. Pada tahap ini, parameter yang akan dicari nilai terbaiknya adalah max_depth, n_estimators, max_features, dan min_samples_leaf.

a) max_depth = Pilihan kedalaman pohon RF yang dicoba adalah 3, 5, 7, dan 10.
b) n_estimators = Pilihan jumlah estimasi adalah 100, 200, dan 300.
c) max_features = Pilihan jumlah fitur maksimal pada pohon RF adalah 10, 20, dan 25 (karena total fitur pilihan pada data training adalah 25).
d) min_samples_leaf = Pilihan jumlah sampel pada daun pohon RF adalah 1, 2, dan 4.

Pemilihan kombinasi nilai parameter terbaik dilihat berdasarkan nilai akurasi model menggunakan evaluasi 3-*cross validation*.

2. Melatih Model Berdasarkan Parameter Terbaik

   Dilakukan pelatihan model, dengan nilai parameter model menggunakan kombinasi parameter RF terbaik yang telah ditemukan pada proses Grid Search sebelumnya. Parameter lain tidak diatur sehingga menggunakan nilai *default*.

3.5. Evaluasi

Evaluasi model yang telah dilatih dilakukan menggunakan *confusion matrix* (yang kemudian diolah lebih lanjut untuk mendapatkan nilai akurasi, presisi, *recall*, dan *F1-Score*) dan Area Under Curve (kurva AUC-ROC).

1. Confusion Matrix memetakan hasil prediksi dengan nilai sebenarnya. Tabel 1 merupakan kategorisasi nilai True Positive (TP), False Positive (FP), False Negative (FN), dan True Negative (TN). TP menunjukkan bahwa serangan benar diprediksi suatu serangan. FP menunjukkan bahwa suatu trafik normal dideteksi oleh model sebagai serangan. FN menunjukkan bahwa suatu serangan diprediksi sebagai trafik normal. Sementara itu, TN menunjukkan bahwa suatu trafik normal benar diprediksi sebagai trafik normal. Nilai akurasi dihitung menggunakan Persamaan (1), nilai presisi menggunakan Persamaan (2), nilai *recall* menggunakan Persamaan (3), dan nilai F1-*Score* menggunakan Persamaan (4).

Tabel 1. Pemetaan TP, FP, FN, dan TN pada *confusion matrix*

	(Predicted) Positif	(Predicted) Negatif
Aktual Positif	**TP**	**FN**
Aktual Negatif	**FP**	**TN**

$$Akurasi = \frac{(TP+TN)}{(TP+TN+FP+FN)} \qquad (1)$$

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

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

$$F1\ Score = \frac{(2\times Presisi\times Recall)}{(Presisi+Recall)} \quad (4)$$

2. Area Under Curve (AUC) - Receiver Operating Characteristics (ROC)
   ROC digunakan untuk mengukur kinerja model pada masalah klasifikasi. AUC adalah area di bawah kurva ROC. Nilai AUC-ROC berada pada rentang 0—1. Nilai AUC-ROC menunjukkan performa model yang lebih baik bila semakin mendekati nilai 1.

4. HASIL DAN PEMBAHASAN

Berikut ini dijelaskan hasil dari tiap tahap penelitian yang telah dilakukan:

4.1. Hasil Pemahaman Bisnis

Poin-poin masalah yang terdapat pada keamanan sistem IoT antara lain:

a) Kerentanan perangkat IoT. Sebagian besar perangkat IoT memiliki keterbatasan daya komputasi dan memori sehingga sulit untuk menerapkan langkah keamanan canggih secara langsung pada perangkat.
b) Volume data yang besar. IoT menghasilkan data dalam jumlah besar yang harus diproses secara *real-time* untuk mendeteksi ancaman. Hal ini menambah kompleksitas dalam pengelolaan jaringan.
c) Beragam jenis serangan: Jaringan IoT sering menjadi target serangan, seperti *distributed denial-of-service* (DDoS), serangan *malware*, *sniffing*, dan lainnya. Serangan ini dapat menyebabkan pencurian data, gangguan operasional, hingga ancaman keselamatan.

Tujuan dari penggunaan *machine learning* dalam deteksi serangan sistem IoT antara lain:

a) Mengamankan perangkat IoT pada level jaringan. Sebagian besar perangkat IoT memiliki keterbatasan sumber daya sehingga model *machine learning* akan diterapkan pada perangkat khusus jaringan yang akan memantau trafik.
b) Mendeteksi serangan pada volume data tinggi, dengan dukungan sumber daya jaringan yang mumpuni. Pengamanan menjadi lebih efektif.
c) Mendeteksi serangan dalam beragam bentuk. *Machine learning* mampu mengenali pola berbagai jenis serangan dan mendeteksinya dengan akurasi yang tinggi.

Solusi yang ditawarkan pada penelitian ini adalah:

a) Model deteksi serangan pada jaringan IoT ini dikembangkan menggunakan algoritma Random Forest (RF) karena algoritma ini menunjukkan performa yang baik pada sistem deteksi serangan lainnya dengan dataset yang berbeda (contoh: model RF dengan dataset CIC-

IDS2017 (Li, Ma, Li, & Li, 2023), model RF dengan data serangan nyata pada suatu perusahaan (Ndichu, Ban, Takahashi, & Inoue, 2023)).

b) Seleksi fitur dilakukan karena terlalu banyaknya fitur pada dataset (43 fitur) dapat membuat durasi pelatihan model menjadi lama. Seleksi fitur dilakukan menggunakan teknik Recursive Feature Elimination (RFE) karena mudah dikonfigurasi dan digunakan. Selain itu, RFE efektif dalam memilih fitur (kolom) dalam kumpulan data pelatihan yang lebih atau paling relevan dalam memprediksi variabel target (Brownlee, 2020). Sebanyak 25 fitur dipilih. Jumlah ini sama dengan jumlah seleksi fitur yang dilakukan pembuat dataset ketika dia mengembangkan model deteksi serangan untuk mengecek kualitas dataset yang dia buat (Moustafa, A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets, 2021).

c) Dilakukan hyperparameter tuning menggunakan metode Grid Search. Metode ini mencoba semua kombinasi parameter berbeda yang dimasukkan pada parameter grid dan menghasilkan kombinasi parameter terbaik (Okamura, 2020). Evaluasi kombinasi parameter menggunakan 3-fold cross validation agar penilaian performa model tidak bias ke potongan data training tertentu.

d) Penilaian performa akhir model menggunakan beberapa metode pengukuran: confusion matrix (akurasi, precision, recall, F-1 score) dan Area Under Curve (kurva AUC-ROC).

4.2. Hasil Pemahaman Data

Dataset ini memiliki 211.043 baris dan 44 kolom. Puluhan kolom ini dapat diklasifikasikan menjadi tujuh bagian: (1) *connection features* (Tabel 2), (2) *statistical features* (Tabel 3), (3) DNS *features* (Tabel 4), (4) SSL *features* (Tabel 5), (5) HTTP *features* (Tabel 6), (6) *violation features* (Tabel 7), dan (7) *labelling attributes* (Tabel 8) (Moustafa, A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets, 2021). Hasil eksplorasi untuk memahami dataset ditampilkan pada Gambar 2, Gambar 3, dan Gambar 4.

Tabel 2. Connection Features pada Dataset ToN_IoT

ID	Fitur	Tipe	Deskripsi
0	src_ip	String	Alamat IP sumber (*source IP address*)
1	src_port	Number	*Source port*
2	dst_ip	String	Alamat IP tujuan (*destination IP address*)
3	dst_port	Number	*Destination port*
4	proto	String	Protokol lapisan *transport* dari koneksi aliran (*flow connections*)
5	service	String	Protokol yang terdeteksi secara dinamis, seperti DNS, HTTP, dan SSL
6	duration	Number	Durasi waktu koneksi paket, yang dihitung dengan mengurangi 'waktu paket terakhir terdeteksi' dan 'waktu paket pertama terdeteksi'

7	src_bytes	Number	Jumlah *byte* sumber, yaitu *payload byte* yang berasal dari nomor urut (*sequence number*)
8	dst_bytes	Number	Jumlah *byte* tujuan, yaitu *payload byte* yang merespons dari nomor urut (*sequence number*) TCP.
9	conn_state	String	Beragam status koneksi, seperti S0 (*connection without replay*), S1 (*connection established*), and REJ (*connection attempt rejected*)
10	missed_bytes	Number	Jumlah *byte* yang hilang dalam *content gaps*

Tabel 3. Statistical Features pada Dataset ToN_IoT

ID	Fitur	Tipe	Deskripsi
11	src_pkts	Number	Jumlah paket original, yaitu jumlah paket yang diperkirakan berasal dari sistem sumber.
12	src_ip_bytes	Number	Jumlah *byte* IP original, yaitu total panjang bidang *header* IP dari sistem sumber.
13	dst_pkts	Number	Jumlah paket tujuan, yaitu jumlah paket yang diperkirakan berasal dari sistem tujuan
14	dst_ip_bytes	Number	Jumlah *byte* IP tujuan, yaitu total panjang bidang *header* IP dari sistem tujuan.

Tabel 4. Fitur-fitur DNS pada Dataset ToN_IoT

ID	Fitur	Tipe	Deskripsi
15	dns_query	String	Subjek nama domain dari kueri DNS.
16	dns_qclass	Number	Nilai yang menentukan kelas kueri DNS.
17	dns_qtype	Number	Nilai yang menentukan jenis kueri DNS.
18	dns_rcode	Number	Nilai kode respons dalam respons DNS.
19	dns_AA	String	Jawaban otoritatif DNS, di mana T menunjukkan bahwa server adalah otoritatif untuk kueri.
20	dns_RD	String	Permintaan rekursi DNS, di mana T menunjukkan permintaan pencarian kueri secara rekursif.
21	dns_RA	String	Rekursi tersedia pada DNS, di mana T menunjukkan server mendukung kueri rekursif.
22	dns_rejected	String	Penolakan DNS, di mana kueri DNS ditolak oleh server.

Tabel 5. Fitur-fitur SSL pada Dataset ToN_IoT

ID	Fitur	Tipe	Deskripsi
23	ssl_version	String	Versi SSL yang ditawarkan oleh server
24	ssl_cipher	String	SSL *cipher suite* yang dipilih oleh server
25	ssl_resumed	String	SSL *flag* menunjukkan sesi yang dapat digunakan untuk memulai koneksi baru, di mana T mengacu pada koneksi SSL yang dimulai
26	ssl_established	String	SSL *flag* menunjukkan pembentukan koneksi antara dua pihak, di mana T mengacu pada pembentukan koneksi
27	ssl_subject	String	Subjek sertifikat X.509 yang ditawarkan oleh server

28	ssl_issuer	String	Pemilik/pencipta SLL dan sertifikat digital yang tepercaya (certificate authority)

Tabel 6. Fitur-fitur HTTP pada Dataset ToN_IoT

ID	Fitur	Tipe	Deskripsi
29	http_trans_depth	String	*Pipelined depth* ke koneksi HTTP
30	http_method	String	Metode permintaan HTTP, seperti GET, POST, dan HEAD
31	http_uri	String	URI yang digunakan dalam permintaan HTTP
32	http_version	String	Versi HTTP yang digunakan, seperti V1.1
33	http_request_body_len	Number	Ukuran konten sebenarnya yang tidak terkompresi dari data yang ditransfer dari klien HTTP
34	http_response_body_len	Number	Ukuran konten sebenarnya yang tidak terkompresi dari data yang ditransfer dari server HTTP
35	http_status_code	Number	Kode status yang dikembalikan oleh server HTTP
36	http_user_agent	String	Nilai header User-Agent dalam protokol HTTP
37	http_orig_mime_types	String	Vektor tipe mime yang diurutkan dari sistem sumber dalam protokol HTTP
38	http_resp_mime_types	String	Vektor tipe mime yang diurutkan dari sistem tujuan dalam protokol HTTP

Tabel 7. Fitur-fitur yang Berkaitan dengan Pelanggaran

ID	Fitur	Tipe	Deskripsi
39	weird_name	String	Nama-nama anomali/pelanggaran terkait protokol yang terjadi
40	weird_addl	String	Informasi tambahan terkait dengan anomali/pelanggaran protokol
41	weird_notice	String	Ini menunjukkan jika pelanggaran/anomali tersebut menimbulkan pemberitahuan (notice)

Tabel 8. Label Atribut pada Dataset ToN_IoT

ID	Fitur	Tipe	Deskripsi
42	label	Number	Tag rekaman normal dan serangan, di mana 0 menunjukkan normal dan 1 menunjukkan serangan
43	type	String	Tag kategori serangan, seperti serangan normal, DoS, DDoS, dan *backdoor*, serta catatan normal

```
df = pd.read_csv('train_test_network.csv')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 211043 entries, 0 to 211042
Data columns (total 44 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   src_ip                  211043 non-null  object
 1   src_port                211043 non-null  int64
 2   dst_ip                  211043 non-null  object
 3   dst_port                211043 non-null  int64
 4   proto                   211043 non-null  object
 5   service                 211043 non-null  object
 6   duration                211043 non-null  float64
 7   src_bytes               211043 non-null  int64
 8   dst_bytes               211043 non-null  int64
 9   conn_state              211043 non-null  object
 10  missed_bytes            211043 non-null  int64
 11  src_pkts                211043 non-null  int64
 12  src_ip_bytes            211043 non-null  int64
 13  dst_pkts                211043 non-null  int64
 14  dst_ip_bytes            211043 non-null  int64
 15  dns_query               211043 non-null  object
 16  dns_qclass              211043 non-null  int64
 17  dns_qtype               211043 non-null  int64
 18  dns_rcode               211043 non-null  int64
 19  dns_AA                  211043 non-null  object
 20  dns_RD                  211043 non-null  object
 21  dns_RA                  211043 non-null  object
 22  dns_rejected            211043 non-null  object
 23  ssl_version             211043 non-null  object
 24  ssl_cipher              211043 non-null  object
 25  ssl_resumed             211043 non-null  object
 26  ssl_established         211043 non-null  object
 27  ssl_subject             211043 non-null  object
 28  ssl_issuer              211043 non-null  object
 29  http_trans_depth        211043 non-null  object
 30  http_method             211043 non-null  object
 31  http_uri                211043 non-null  object
 32  http_version            211043 non-null  object
 33  http_request_body_len   211043 non-null  int64
 34  http_response_body_len  211043 non-null  int64
 35  http_status_code        211043 non-null  int64
 36  http_user_agent         211043 non-null  object
 37  http_orig_mime_types    211043 non-null  object
 38  http_resp_mime_types    211043 non-null  object
 39  weird_name              211043 non-null  object
 40  weird_addl              211043 non-null  object
 41  weird_notice            211043 non-null  object
 42  label                   211043 non-null  int64
 43  type                    211043 non-null  object
dtypes: float64(1), int64(16), object(27)
memory usage: 70.8+ MB
```

Gambar 2. Info dataset ToN_IoT

```
df.head()
```

	src_ip	src_port	dst_ip	dst_port	proto	service	duration	src_bytes	dst_bytes	conn_state	...	http_response_body_len	http_status_code	http_user_agent	http_orig_mime_types	http_resp_mime_types	weird_name	weird_addl	weird_notice	label	type
0	192.168.1.37	4444	192.168.1.193	49178	tcp	-	290.371539	101568	2592	OTH	...	0	0	-	-	-	-	-	-	1	backdoor
1	192.168.1.193	49180	192.168.1.37	8080	tcp	-	0.000102	0	0	REJ	...	0	0	-	-	-	-	-	-	1	backdoor
2	192.168.1.193	49180	192.168.1.37	8080	tcp	-	0.000148	0	0	REJ	...	0	0	-	-	-	-	-	-	1	backdoor
3	192.168.1.193	49180	192.168.1.37	8080	tcp	-	0.000113	0	0	REJ	...	0	0	-	-	-	-	-	-	1	backdoor
4	192.168.1.193	49180	192.168.1.37	8080	tcp	-	0.000130	0	0	REJ	...	0	0	-	-	-	-	-	-	1	backdoor

5 rows × 44 columns

Gambar 3. Bagian awal dataset ToN_IoT yang ditampilkan dengan fungsi head()

```
df.describe()
```

	src_port	dst_port	duration	src_bytes	dst_bytes	missed_bytes	src_pkts	src_ip_bytes	dst_pkts	dst_ip_bytes	dns_qclass	dns_qtype	dns_rcode	http_request_body_len	http_response_body_len	http_status_code	label
count	211043.000000	211043.000000	211043.000000	2.110430e+05	2.110430e+05	2.110430e+05	211043.000000	2.110430e+05	211043.000000	2.110430e+05	211043.000000	211043.000000	211043.000000	211043.000000	2.110430e+05	211043.000000	211043.000000
mean	38646.519543	3495.153770	7.700887	2.581136e+05	2.588046e+05	3.443234e+04	9.595220	7.760822e+02	3.846861	1.584687e+03	227.630805	3.610909	0.123989	0.065418	1.449295e+02	0.303905	0.763081
std	19307.271048	10191.624778	564.141946	1.709490e+07	1.802563e+07	5.261621e+06	91.778821	2.229703e+04	330.705796	1.901795e+05	2720.713562	23.797747	0.598804	9.243405	3.047244e+04	8.270377	0.425193
min	1.000000	0.000000	0.000000	0.000000e+00	0.000000e+00	0.000000e+00	0.000000	0.000000e+00	0.000000	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000e+00	0.000000	0.000000
25%	34608.000000	65.000000	0.000000	0.000000e+00	0.000000e+00	0.000000e+00	1.000000	4.800000e+01	0.000000	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000e+00	0.000000	1.000000
50%	44754.000000	80.000000	0.000170	0.000000e+00	0.000000e+00	0.000000e+00	1.000000	8.200000e+01	1.000000	4.000000e+01	0.000000	0.000000	0.000000	0.000000	0.000000e+00	0.000000	1.000000
75%	51133.000000	1253.000000	0.054196	1.300000e+02	8.900000e+01	0.000000e+00	4.000000	4.150000e+02	2.000000	1.340000e+02	0.000000	0.000000	0.000000	0.000000	0.000000e+00	0.000000	1.000000
max	65528.000000	65467.000000	93516.929170	3.890855e+09	3.913853e+09	1.854527e+09	24623.000000	6.522626e+06	121942.000000	8.639552e+07	32769.000000	255.000000	5.000000	2338.000000	1.342438e+07	404.000000	1.000000

Gambar 4. Ringkasan statistikal data yang ditampilkan dengan fungsi describe()

4.3. Hasil Persiapan Data

Berikut ini adalah hasil dari tahap persiapan data, sesuai langkah-langkah yang telah dijelaskan pada subbab 3.3.

1. Drop Fitur yang Tidak Perlu.

   Dengan penghapusan delapan fitur yang telah disebutkan pada subbab 3.3, kini dataset untuk tahap berikutnya memiliki 36 kolom, seperti yang ditampilkan pada Gambar 5.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 211043 entries, 0 to 211042
Data columns (total 36 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   proto                  211043 non-null  object
 1   service                211043 non-null  object
 2   duration               211043 non-null  float64
 3   src_bytes              211043 non-null  int64
 4   dst_bytes              211043 non-null  int64
 5   conn_state             211043 non-null  object
 6   missed_bytes           211043 non-null  int64
 7   src_pkts               211043 non-null  int64
 8   src_ip_bytes           211043 non-null  int64
 9   dst_pkts               211043 non-null  int64
 10  dst_ip_bytes           211043 non-null  int64
 11  dns_query              211043 non-null  object
 12  dns_qclass             211043 non-null  int64
 13  dns_qtype              211043 non-null  int64
 14  dns_rcode              211043 non-null  int64
 15  dns_AA                 211043 non-null  object
 16  dns_RD                 211043 non-null  object
 17  dns_RA                 211043 non-null  object
 18  dns_rejected           211043 non-null  object
 19  ssl_version            211043 non-null  object
 20  ssl_cipher             211043 non-null  object
 21  ssl_resumed            211043 non-null  object
 22  ssl_established        211043 non-null  object
 23  ssl_subject            211043 non-null  object
 24  ssl_issuer             211043 non-null  object
 25  http_trans_depth       211043 non-null  object
 26  http_method            211043 non-null  object
 27  http_uri               211043 non-null  object
 28  http_version           211043 non-null  object
 29  http_request_body_len  211043 non-null  int64
 30  http_response_body_len 211043 non-null  int64
 31  http_status_code       211043 non-null  int64
 32  http_user_agent        211043 non-null  object
 33  http_orig_mime_types   211043 non-null  object
 34  http_resp_mime_types   211043 non-null  object
 35  label                  211043 non-null  int64
dtypes: float64(1), int64(14), object(21)
memory usage: 58.0+ MB
```

Gambar 5. Fitur-fitur dataset setelah drop fitur yang tidak diperlukan

2. Ubah Tipe Objek ke Tipe Angka (Encoding).

   Sebelum dilakukan *encoding*, contoh nilai fitur tertampil pada Gambar 6. Setelah dilakukan *label encoding*, contoh nilai fitur tertampil pada Gambar 7.

```
# Menampilkan sampel data sebelum encoding (masih nilai kategorikal)
df.loc[:,['proto','service','conn_state','dns_query', 'ssl_version', 'http_version']].head()
```

	proto	service	conn_state	dns_query	ssl_version	http_version
0	tcp	-	OTH	-	-	-
1	tcp	-	REJ	-	-	-
2	tcp	-	REJ	-	-	-
3	tcp	-	REJ	-	-	-
4	tcp	-	REJ	-	-	-

Gambar 6. Contoh nilai fitur sebelum *encoding*

```
# Menampilkan sampel data setelah encoding (sudah bernilai numerikal)
new_df.loc[:,['proto','service','conn_state','dns_query', 'ssl_version', 'http_version']].head()
```

	proto	service	conn_state	dns_query	ssl_version	http_version
0	1	0	0	1	0	0
1	1	0	1	1	0	0
2	1	0	1	1	0	0
3	1	0	1	1	0	0
4	1	0	1	1	0	0

Gambar 7. Contoh nilai fitur setelah *encoding*

3. Memisahkan Variabel X dan y

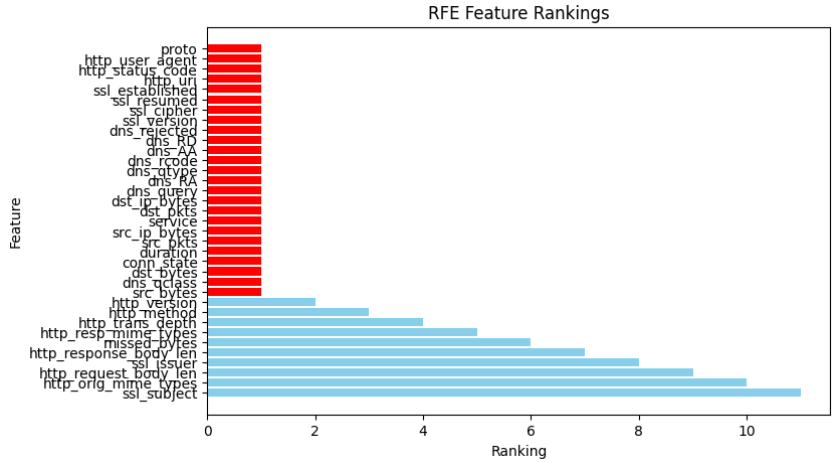
Proses pemisahan variabel X dan y tertampil pada Gambar 8. Variabel y merupakan target pelatihan model.

```
# Memisahkan X dan y. Target (y) adalah label dengan tipe binary (0 = trafik normal dan 1 = trafik serangan)
X = new_df.drop(['label'], axis=1)
y = new_df['label']
```

Gambar 8. Proses pemisahan variabel X dan y

4. Seleksi Fitur

Gambar 9 merupakan visualisasi ranking dari fitur terpilih. Dua puluh lima fitur terpilih mendapatkan ranking 1, sedangkan fitur lainnya mendapat ranking 2 dan seterusnya. Dengan demikian, data untuk tahap berikutnya memiliki 25 kolom saja, seperti yang ditampilkan pada Gambar 10.



Gambar 9. Fitur terpilih setelah proses eliminasi fitur rekursif

```
X_selected = X.iloc[:, selected_features]
X_selected
```

	proto	service	duration	src_bytes	dst_bytes	conn_state	src_pkts	src_ip_bytes	dst_pkts	dst_ip_bytes	...	dns_RD	dns_RA	dns_rejected	ssl_version	ssl_cipher	ssl_resumed	ssl_established	http_trans_depth	http_uri	http_status_code
0	1	0	290.371539	101568	2592	0	108	108064	31	3832	...	0	0	0	0	0	0	0	0	0	0
1	1	0	0.000102	0	0	1	1	52	1	40	...	0	0	0	0	0	0	0	0	0	0
2	1	0	0.000148	0	0	1	1	52	1	40	...	0	0	0	0	0	0	0	0	0	0
3	1	0	0.000113	0	0	1	1	48	1	40	...	0	0	0	0	0	0	0	0	0	0
4	1	0	0.000130	0	0	1	1	52	1	40	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
211038	1	5	65.376610	2665	322	9	5	2925	5	590	...	0	0	0	0	0	0	0	0	0	0
211039	1	5	65.710346	1987	322	9	6	2307	5	590	...	0	0	0	0	0	0	0	0	0	0
211040	1	5	65.766512	3922	322	9	7	4294	6	642	...	0	0	0	0	0	0	0	0	0	0
211041	1	5	65.753940	2401	322	9	6	2721	5	590	...	0	0	0	0	0	0	0	0	0	0
211042	1	5	65.771855	3181	322	9	7	3553	6	642	...	0	0	0	0	0	0	0	0	0	0

211043 rows × 25 columns

Gambar 10. Dataset setelah proses seleksi fitur

5. Normalisasi Fitur Angka

Gambar 11 menunjukkan sampel fitur sebelum dilakukan normalisasi. Gambar 12 menunjukkan sampel fitur setelah normalisasi. Terlihat bahwa rentang data setelah normalisasi adalah 0—1.

```
# Menampilkan sampel data sebelum normalisasi
new_df.loc[:,['duration', 'src_bytes', 'dst_bytes',
              'src_pkts', 'src_ip_bytes', 'dst_pkts', 'dst_ip_bytes']].describe()
```

	duration	src_bytes	dst_bytes	src_pkts	src_ip_bytes	dst_pkts	dst_ip_bytes
count	211043.000000	2.110430e+05	2.110430e+05	211043.000000	2.110430e+05	211043.000000	2.110430e+05
mean	7.700887	2.581136e+05	2.588046e+05	9.595220	7.760822e+02	3.846861	1.584687e+03
std	564.141946	1.709490e+07	1.802563e+07	91.778821	2.229703e+04	330.705796	1.901795e+05
min	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000e+00	0.000000	0.000000e+00
25%	0.000000	0.000000e+00	0.000000e+00	1.000000	4.800000e+01	0.000000	0.000000e+00
50%	0.000170	0.000000e+00	0.000000e+00	1.000000	8.200000e+01	1.000000	4.000000e+01
75%	0.054196	1.300000e+02	8.900000e+01	4.000000	4.150000e+02	2.000000	1.340000e+02
max	93516.929170	3.890855e+09	3.913853e+09	24623.000000	6.522626e+06	121942.000000	8.639552e+07

Gambar 11. Deskripsi beberapa fitur sebelum normalisasi

```
# Menampilkan sampel data setelah normalisasi
X_selected.loc[:,['duration', 'src_bytes', 'dst_bytes',
                  'src_pkts', 'src_ip_bytes', 'dst_pkts', 'dst_ip_bytes']].describe()
```

	duration	src_bytes	dst_bytes	src_pkts	src_ip_bytes	dst_pkts	dst_ip_bytes
count	2.110430e+05	2.110430e+05	2.110430e+05	211043.000000	211043.000000	211043.000000	2.110430e+05
mean	8.234752e-05	6.633852e-05	6.612526e-05	0.000390	0.000119	0.000032	1.834223e-05
std	6.032511e-03	4.393610e-03	4.605596e-03	0.003727	0.003418	0.002712	2.201266e-03
min	0.000000e+00	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000	0.000000e+00
25%	0.000000e+00	0.000000e+00	0.000000e+00	0.000041	0.000007	0.000000	0.000000e+00
50%	1.817853e-09	0.000000e+00	0.000000e+00	0.000041	0.000013	0.000008	4.629870e-07
75%	5.795261e-07	3.341168e-08	2.273974e-08	0.000162	0.000064	0.000016	1.551006e-06
max	1.000000e+00	1.000000e+00	1.000000e+00	1.000000	1.000000	1.000000	1.000000e+00

Gambar 12. Deskripsi beberapa fitur setelah normalisasi

6. Splitting Data

Proses pemisahan data menjadi data *training* dan data *testing* tertampil pada Gambar 13.

```
# Memisahkan dat (variable) y_train: Any
X_train, X_val, y_train, y_val = ms.train_test_split(X_selected, y, test_size=0.2, random_state = 0)
```

Gambar 13. Proses pemisahan data menjadi data training dan data testing

4.4. Hasil Pemodelan

1. Hasil *Hyperparameter Tuning* dengan *Grid Search*

Gambar 14 adalah kode Python yang digunakan untuk melakukan Grid Search. Hasil dari Grid Search adalah kombinasi parameter RF terbaik, yaitu: max_depth = 10, max_features = 10, min_samples_leaf = 1, n_estimators = 300, dengan nilai rerata akurasi terbaik sebesar 0.9112642385382433 (Gambar 15).

```
rf_grid = RandomForestClassifier()
gr_space = {
    'max_depth': [3,5,7,10],
    'n_estimators': [100, 200, 300],
    'max_features': [10, 20, 25],
    'min_samples_leaf': [1, 2, 4]
}

grid = GridSearchCV(rf_grid, gr_space, cv = 3, scoring='accuracy', verbose = 3)
model_grid = grid.fit(X_selected, y)

print('Best hyperparameters are '+str(model_grid.best_params_))
print('Best score is: ' + str(model_grid.best_score_))
```

Gambar 14. Kode yang digunakan untuk melakukan Grid Search

```
Best hyperparameters are {'max_depth': 10, 'max_features': 10, 'min_samples_leaf': 1, 'n_estimators': 300}
Best score is: 0.9112642385382433
```

Gambar 15. Hasil dari proses Grid Search

2. Melatih Model Berdasarkan Parameter Terbaik

Model telah dilatih (Gambar 16) menggunakan parameter terbaik yang telah didapatkan pada proses *grid search*.

```
# buat model prediksi
modelRF = RandomForestClassifier(max_depth=10, max_features=10, min_samples_leaf=1, n_estimators=300) # angka berdasarkan hasil grid search
modelRF.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(max_depth=10, max_features=10, n_estimators=300)
```

Gambar 16. Proses pelatihan model menggunakan parameter terbaik

4.5. Hasil Evaluasi

Evaluasi model yang telah dilatih dilakukan menggunakan confusion matrix (yang kemudian diolah lebih lanjut untuk mendapatkan nilai akurasi, presisi, recall, dan F1-Score) dan Area Under Curve (kurva AUC-ROC).

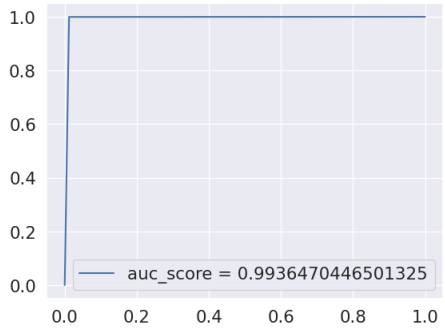
1. Confusion Matrix dari hasil pelatihan model ini tertampil pada Tabel 9. Akurasi dihitung menggunakan Persamaan (1) dan hasilnya adalah 0.9966357885758962. Presisi dihitung menggunakan Persamaan (2) dan hasilnya adalah 0.9963252323750116. *Recall* dihitung menggunakan Persamaan (3) dan hasilnya adalah 0.9992876389878279. F1-*Score* dihitung menggunakan Persamaan (4) dan hasilnya adalah 0.9978042368950054. Semua hasil yang didapatkan bernilai di atas 0.99. Artinya, performa model tersebut sudah baik.

Tabel 9. Nilai *confusion matrix* dari hasil pelatihan model

	(Predicted) Positif	(Predicted) Negatif
Aktual Positif	9803	119
Aktual Negatif	23	32264

2. AUC-ROC

   Gambar 17 merupakan visualisasi AUC-ROC dari performa model eksperimen. Nilai 0.9936470446501325 membuktikan bahwa performa model sudah baik.



Gambar 17. Kurva AUC-ROC untuk mengevaluasi model

3. Diskusi Hasil Evaluasi

   Berdasarkan performa hasil evaluasi tersebut, dapat dikatakan bahwa model telah mampu mendeteksi serangan pada trafik jaringan dengan baik. Pengembangan model ini merupakan suatu usaha untuk menyelesaikan masalah yang telah dipaparkan sebelumnya pada subbab Pemahaman Bisnis (*business understanding*). Pengembangan model ini juga berusaha mencapai tujuan yang diharapkan:

   a) Mengamankan perangkat IoT

      Model *machine learning* ini dapat diterapkan pada perangkat jaringan khusus, seperti Intrusion Detection System (IDS). Dengan pendekatan ini, IDS dengan *machine learning* dapat memantau dan menganalisis trafik. Hasil evaluasi menunjukkan bahwa model telah mendeteksi serangan dengan baik. Perangkat IoT tetap terlindungi tanpa mengorbankan performa atau fungsionalitas perangkat IoT karena pengamananan diterapkan pada level jaringan.

   b) Mendeteksi serangan pada volume data tinggi

      Dengan pelatihan dan pengujian model yang melibatkan cukup banyak sampel (211.043 sampel), model ini menunjukkan performa yang baik. Dengan dukungan sumber daya komputasi yang lebih baik, model ini dapat dimanfaatkan secara optimal untuk mendeteksi ancaman pada volume data tinggi. Dengan dukungan *machine learning*, sistem akan mampu mengidentifikasi ancaman secara cepat dan akurat sehingga pengamanan jaringan dapat dilakukan dengan lebih efektif.

   c) Mendeteksi Beragam Serangan

      Model telah berhasil mendeteksi berbagai trafik serangan (*backdoor*, DDoS, DoS, *ransomware*, dsb.) sebagai 'serangan'. Pendekatan ini memungkinkan deteksi yang akurat terhadap beragam ancaman sehingga sistem dapat memberikan perlindungan yang komprehensif untuk mencegah gangguan operasional IoT dan risiko keamanan lainnya.

Selain itu, tiap pernyataan solusi yang direncanakan (lihat subbab 4.1.) telah berdampak baik. Model deteksi serangan pada jaringan IoT yang dikembangkan menggunakan algoritma Random Forest (RF) menunjukkan performa yang optimal dalam mendeteksi serangan. Dengan penerapan Recursive Feature Elimination (RFE), jumlah fitur dataset berhasil direduksi menjadi 25 fitur yang paling relevan sehingga proses pelatihan menjadi lebih cepat. Teknik ini terbukti efektif dalam meningkatkan efisiensi sekaligus mempertahankan kualitas deteksi serangan. *Hyperparameter tuning* melalui Grid Search dengan evaluasi 3-*fold cross-validation* menghasilkan kombinasi parameter terbaik yang mampu meminimalkan bias dan meningkatkan generalisasi model pada data baru. Penilaian akhir performa model menggunakan Confusion Matrix (akurasi, presisi, *recall*, F1-*score*) dan AUC-ROC mengonfirmasi bahwa model mampu mendeteksi berbagai jenis serangan dengan akurasi tinggi, keseimbangan prediksi positif-negatif, dan kemampuan diskriminasi yang baik.

5. KESIMPULAN

Penelitian ini telah mengembangkan model deteksi serangan pada jaringan IoT. Model dilatih dan diuji dengan dataset ToN_IoT. Dengan algoritma Random Forest (RF), model mampu mendeteksi beragam jenis serangan secara akurat. Penerapan Recursive Feature Elimination (RFE) untuk seleksi fitur meningkatkan efisiensi pelatihan model, sementara Grid Search dengan 3-*fold cross-validation* memastikan konfigurasi parameter terbaik (max_depth = 10, max_features = 10, min_samples_leaf = 1, n_estimators = 300). Evaluasi performa akhir menggunakan Confusion Matrix dan AUC-ROC menunjukkan hasil yang sangat baik (semua skor di atas 0.99), membuktikan efektivitas model dalam mendeteksi ancaman.

UCAPAN TERIMA KASIH

Terima kasih kepada Dicoding dan DBS Foundation atas kesempatan Beasiswa DBS Foundation Coding Camp – kelas Machine Learning Terapan. Terima kasih kepada Jurusan Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia, atas dukungannya untuk publikasi makalah ini.

DAFTAR PUSTAKA

Ahmad, Z., Khan, A. S., Shiang, C. W., Abdullah, J., & Ahmad, F. (2020). Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Transactions on Emerging Telecommunications Technologies*. doi:https://doi.org/10.1002/ett.4150

Alwi, M. N. (2024, Juni 19). *CRISP-DM: Tahapan, Studi Kasus, Kelebihan, dan Kekurangan*. Retrieved from Dicoding: https://www.dicoding.com/blog/crisp-dm-tahapan-studi-kasus-kelebihan-dan-kekurangan/

AWS. (2025). *What is Hyperparameter Tuning?* Retrieved from AWS: https://aws.amazon.com/what-is/hyperparameter-tuning/

Belcic, I., & Stryker, C. (2025). *What is feature selection?* Retrieved from IBM: https://www.ibm.com/think/topics/feature-selection

Brownlee, J. (2020, Agustus 28). *Recursive Feature Elimination (RFE) for Feature Selection in Python*. Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/rfe-feature-selection-in-python/

Li, F., Ma, W., Li, H., & Li, J. (2023). Improving Intrusion Detection System Using Ensemble Methods and Over-Sampling Technique. *2022 4th International Academic Exchange Conference on Science and Technology Innovation (IAECST)*. Guangzhou, China: IEEE Xplore. doi:https://doi.org/10.1109/IAECST57965.2022.10062178

Moustafa, N. (2021). A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets. *Sustainable Cities and Society*. doi:https://doi.org/10.1016/j.scs.2021.102994

Moustafa, N. (2021, Juni 2). *The TON_IoT Datasets*. Retrieved from UNSW: https://research.unsw.edu.au/projects/toniot-datasets

Moustafa, N., Ahmed, M., & Ahmed, S. (2021). Data Analytics-Enabled Intrusion Detection: Evaluations of ToN_IoT Linux Datasets. *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. Guangzhou, China: IEEE Xplore. doi:https://doi.org/10.1109/TrustCom50675.2020.00100

Ndichu, S., Ban, T., Takahashi, T., & Inoue, D. (2023). Security-Alert Screening with Oversampling Based on Conditional Generative Adversarial Networks. *2022 17th Asia Joint Conference on Information Security (AsiaJCIS)*. Baoding, China: IEEE Xplore. doi:https://doi.org/10.1109/AsiaJCIS57030.2022.00011

NETGEAR Security Team. (2024). *Every 24 Hours, Home Networks See an Average of 10 Attacks. Is Your Network Secure?* Retrieved from NETGEAR: https://www.netgear.com/hub/network/2024-iot-threat-report/

Okamura, S. (2020, Desember 28). *GridSearchCV for Beginners*. Retrieved from Towards Data Science: https://towardsdatascience.com/gridsearchcv-for-beginners-db48a90114ee/

O'Shea, D. (2024, July 29). *IoT security attacks are escalating - SonicWall report*. Retrieved from Fierce Sensors: https://www.fiercesensors.com/iot-wireless/iot-security-attacks-are-escalating-sonicwall-report

Rahma, F., Rachmadi, R. F., Pratomo, B. A., & Purnomo, M. H. (2023). Assessing the Effectiveness of Oversampling and Undersampling Techniques for Intrusion Detection on an Imbalanced Dataset. *2023 IEEE Industrial Electronics and Applications Conference (IEACon)*. Penang, Malaysia: IEEE Xplore. doi:https://doi.org/10.1109/IEACon57683.2023.10370430

Rohmah, M. (2024, Juni 21). *Apa itu Random Forest? Pengertian, Cara Kerja & Contohnya*. Retrieved from Dibimbing: https://dibimbing.id/blog/detail/apa-itu-random-forest-pengertian-cara-kerja-contohnya