PENGEMBANGAN MODEL ENSEMBLE STACKING UNTUK KLASIFIKASI SERANGAN DISTRIBUTED DENIAL OF SERVICE



Disusun Oleh:

N a m a : Juan Parez Mangku Alamsyah

NIM : 21523231

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

2025

HALAMAN PENGESAHAN DOSEN PEMBIMBING

PENGEMBANGAN MODEL ENSEMBLE STACKING UNTUK KLASIFIKASI SERANGAN DISTRIBUTED DENIAL OF SERVICE

TUGAS AKHIR

Disusun Oleh:

N a m a : Juan Parez Mangku Alamsyah

NIM : 21523231

Yogyakarta, 23 Juli 2025

Pembimbing 1,	Pembimbing 2,
Fayruz Rahma, S.T., M.Eng.	Arrie Kurniawardhani, S.Si., M.Kom.

HALAMAN PENGESAHAN DOSEN PENGUJI

PENGEMBANGAN MODEL ENSEMBLE STACKING UNTUK KLASIFIKASI SERANGAN DISTRIBUTED DENIAL OF SERVICE

TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 6 Agustus 2025

Tim Penguji

Arrie Kurniawardhani, S.Si., M.Kom

Anggota 1

Sri Mulyati, S.Kom., M.Kom.

Anggota 2

Kholid Haryono, S.T., M.Kom.

Mengetahui,
Ketua Program Studi Informatika – Program Sarjana
Fakultas Teknologi Industri
Universitas Islam Indonesia

Ir. Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D. IPM., ASEAN Eng

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Juan Parez Mangku Alamsyah

NIM : 21523231

Tugas akhir dengan judul:

PENGEMBANGAN MODEL ENSEMBLE STACKING UNTUK KLASIFIKASI SERANGAN DISTRIBUTED DENIAL OF SERVICE

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 23 Juli 2025



Juan Parez Mangku Alamsyah

HALAMAN PERSEMBAHAN

Segala puji dan rasa syukur yang terdalam saya panjatkan kepada Allah SWT atas segala kekuatan, kesehatan, dan kesempatan yang telah Ia berikan hingga saya mampu menyelesaikan perjalanan ini. Shalawat serta salam senantiasa tercurah kepada junjungan Nabi Muhammad SAW yang menjadi cahaya dan inspirasi di sepanjang hidup saya.

Dengan segala kerendahan hati, karya kecil ini saya persembahkan untuk orang-orang yang selalu hadir dalam langkah perjalanan penulis:

Untuk Ayah dan Ibu tercinta, terima kasih telah atas cinta, doa, serta dukungan tanpa batas dalam setiap langkah hidup penulis.

Untuk kakakku tersayang, terima kasih selalu menjadi sumber semangat, inspirasi, serta teladan dalam menghadapi berbagai tantangan. Terima kasih atas segala pengorbanan, nasihat, dan kasih sayang yang tiada henti, yang senantiasa menguatkan penulis.

Untuk teman-teman yang senantiasa menemani langkah-langkah penulis di setiap suka maupun duka, terima kasih atas segala doa, cerita, dan kehangatan. Kehadiran kalian adalah cahaya dalam perjalanan panjang yang sering terasa gelap, dan penulis bersyukur pernah dipertemukan dengan kalian.

Dan untuk diri saya sendiri, terima kasih karena sudah mau bertahan meski sering goyah, terus berjalan meski langkah terasa berat, dan tetap berjuang di tengah rasa takut serta ragu yang tak pernah benar-benar hilang. Saya belajar, bahwa bahkan langkah kecil yang penuh air mata pun mampu membawa saya lebih jauh dari yang pernah saya bayangkan.

HALAMAN MOTO

"*If Allah helps you, none can defeat you. But if He denies you help, then who else can help you? So in Allah let the believers put their trust."*

(QS. Ali Imran: 160)

"Sebagai manusia, tugas kita hanyalah berusaha dan berdoa, sementara hasilnya sepenuhnya kita serahkan kepada Allah."

(CSD)

KATA PENGANTAR

Assalamu'alakum Warrahmatullahi Wabarakatuh,

Puji dan syukur penulis panjatkan ke hadirat Allah SWT atas limpahan rahmat dan hidayah-Nya, sehingga penulis dapat menyelesaikan penyusunan tugas akhir ini. Penulisan laporan ini turut menjadi wadah untuk mencurahkan berbagai perasaan, mulai dari suka cita hingga tantangan yang dialami sepanjang prosesnya. Dalam proses penyelesaian tugas akhir ini, penulis memperoleh banyak dukungan dan bantuan dari berbagai pihak. Oleh karena itu, dengan penuh rasa hormat dan terima kasih, penulis ingin menyampaikan apresiasi kepada:

1. Kedua orang tua serta kakak yang senantiasa berdoa dan memberikan dukungan dalam bentuk apa pun kepada penulis.
2. Bapak Prof. Fathul Wahid, S.T., M.Sc., Ph.D. selaku Rektor Universitas Islam Indonesia.
3. Bapak Prof., Dr., Ir., Hari Purnomo, M.T., IPU, ASEAN.Eng. Pimpinan Fakultas Teknologi Industri Universitas Islam Indonesia.
4. Bapak Dr. Raden Teduh Dirghayu, S.T., M.Sc. Ketua Jurusan Informatika Universitas Islam Indonesia.
5. Bapak Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D. Ketua Program Studi Informatika Universitas Islam Indonesia.
6. Ibu Fayruz Rahma, S.T., M.Eng. dan Ibu Arrie Kurniawardhani, S.Si., M.Kom. selaku dosen pembimbing tugas akhir yang senantiasa memberikan ilmu, arahan serta waktu untuk membimbing penulis mengerjakan tugas akhir ini.
7. Bapak Galang Prihadi Mahardhika, S.Kom., M.Kom., dan Bapak Dr. Feri Wijayanto, S.T., M.T. selaku dosen pembimbing akademik penulis.
8. Teman-teman saya, khususnya Arya, Bintang, Siraj, Nabila, Tanziila, Naela, Anra, Satriya, Zahwa, Rain, Asista, Kharisma, Nabil, Ihwan, Agung, dan yang tidak bisa penulis sebutkan satu persatu, yang telah menemani, mendukung, dan memberikan warna selama masa perkuliahan.

Yogyakarta, 23 Juli 2025

Juan Parez Mangku Alamsyah

SARI

Serangan *Distributed Denial of Service* (DDoS) merupakan salah satu bentuk ancaman siber yang umum terjadi dan berdampak serius karena dapat melumpuhkan layanan jaringan dengan membanjiri lalu lintas dari berbagai sumber secara bersamaan. Kompleksitas dan variasi serangan ini menjadikan proses deteksi semakin menantang, terutama dalam konteks klasifikasi *multiclass* yang menuntut identifikasi jenis serangan secara spesifik. Penelitian ini bertujuan untuk membangun model klasifikasi serangan DDoS berbasis *ensemble learning* dengan pendekatan *stacking*, untuk meningkatkan akurasi dan ketahanan model klasifikasi. Dataset CIC-DDoS2019 digunakan karena memuat beragam jenis serangan DDoS dan lalu lintas normal. Proses penelitian mencakup tahapan pengumpulan data, analisis data eksploratif, pembersihan data, transformasi label, pembagian data latih dan uji, penskalaan data, serta seleksi fitur. Delapan algoritma *machine learning* diuji, meliputi Naïve Bayes, K-Nearest Neighbors, Support Vector Machine, Logistic Regression, Decision Tree, Random Forest, AdaBoost, dan XGBoost. Model *stacking* dalam penelitian ini mengadopsi dan mengevaluasi empat kombinasi *base learner* dan *meta learner* yang diadaptasi dari studi sebelumnya, serta satu kombinasi model *stacking* yang diusulkan oleh peneliti berdasarkan analisis performa model individual. Model *stacking* yang diusulkan oleh peneliti menggabungkan Random Forest, AdaBoost, XGBoost, dan Decision Tree sebagai *base learner*, dengan Logistic Regression sebagai *meta learner*. Evaluasi model dilakukan berdasarkan metrik *accuracy*, *precision*, *recall*, *f1-score*, ROC-AUC, dan *inference time*. Hasil eksperimen menunjukkan bahwa model *stacking* yang diusulkan peneliti mampu mencapai *accuracy* hingga 0,7965 dan nilai AUC sebesar 0,9832, serta mengungguli sebagian besar model individual dalam hal *accuracy, precision*, *recall*, *f1-score*, dan nilai AUC. Namun demikian, model ini memiliki waktu *inference* yang relatif tinggi, sehingga meskipun efektif dalam meningkatkan *accuracy* klasifikasi serangan DDoS perlu mempertimbangkan efisiensi komputasi secara lebih lanjut.

Kata kunci: *Distributed Denial of Service*, *Ensemble Learning*, *Stacking*, *Machine Learning*.

GLOSARIUM

DDoS	Distributed Denial of Service. Serangan siber yang membanjiri layanan jaringan untuk membuatnya tidak tersedia.
Machine Learning	Cabang kecerdasan buatan yang memungkinkan komputer belajar dari data untuk membuat prediksi atau keputusan.
Ensemble Learning	Teknik yang menggabungkan beberapa model pembelajaran untuk meningkatkan performa prediksi.
Stacking	Pendekatan ensemble learning yang menggabungkan beberapa model dasar dengan meta-learner untuk menghasilkan prediksi akhir.
Base Learner	Model-model dasar dalam stacking yang digunakan untuk menghasilkan prediksi awal.
Meta Learner	Model yang digunakan dalam stacking untuk menggabungkan output dari base learner menjadi prediksi akhir.
CIC-DDoS2019	Dataset publik dari Canadian Institute for Cybersecurity yang berisi berbagai jenis serangan DDoS.
Hyperparameter	Parameter eksternal model machine learning yang tidak dipelajari dari data dan harus disetel secara manual.
Grid Search	Metode sistematis untuk mencari kombinasi hyperparameter terbaik berdasarkan kinerja model.
Inference Time	Waktu yang dibutuhkan model untuk membuat prediksi terhadap data baru setelah proses pelatihan selesai.
ROC-AUC	Receiver Operating Characteristic - Area Under the Curve, metrik evaluasi yang mengukur kemampuan klasifikasi model.
Label Encoding	Teknik konversi label kategorikal menjadi nilai numerik untuk digunakan dalam machine learning.

DAFTAR ISI

HALAMAN JUDUL i
HALAMAN PENGESAHAN DOSEN PEMBIMBING ii
HALAMAN PENGESAHAN DOSEN PENGUJI iii
HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR iv
HALAMAN PERSEMBAHAN v
HALAMAN MOTO vi
KATA PENGANTAR vii
SARI viii
GLOSARIUM ix
DAFTAR ISI x
DAFTAR TABEL xii
DAFTAR GAMBAR xiii
BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 3
1.3 Batasan Masalah 3
1.4 Tujuan Penelitian 4
1.5 Manfaat Penelitian 5
1.6 Sistematika Penulisan 5
BAB II KAJIAN PUSTAKA 6
2.1 Landasan Teori 6
2.1.1 Distributed Denial of Service Attack (DDoS) 6
2.1.2 Dataset 8
2.1.3 Machine Learning 9
2.1.4 Algoritma Machine Learning 9
2.1.5 Ensemble Learning 11
2.1.6 Seleksi Fitur 15
2.1.7 Pengukuran Kinerja Model 17
2.2 Tinjauan Pustaka 20
BAB III METODOLOGI PENELITIAN 27
3.1 Pengumpulan Data 28
3.2 Analisis Data Eksploratif 29
3.3 Prepemrosesan Data 29
3.3.1 Pembersihan Data 29
3.3.2 Transformasi Label 30
3.3.3 Pembagian Data 30
3.3.4 Penskalaan Data 31
3.3.5 Seleksi Fitur 32
3.4 Penyesuaian Hyperparameter 33
3.5 Algoritma Machine Learning 35
3.6 Ensemble Learning 35
3.7 Evaluasi Model 36
3.8 Lingkungan Eksperimen 37
3.8.1 Spesifikasi Perangkat Keras (Hardware) 37
3.8.2 Lingkungan Pemrograman 38
3.8.3 Library yang Digunakan 38
BAB IV HASIL DAN PEMBAHASAN 39

4.1 Pengumpulan Data 39
4.2 Analisis Data Eksploratif 41
4.3 Prepemrosesan Data 44
4.3.1 Data Cleaning 44
4.3.2 Transformasi Label 46
4.3.3 Pembagian Data 46
4.3.4 Penskalaan Data 47
4.3.5 Seleksi Fitur 48
4.4 Penyesuaian Hyperparameter 52
4.4.1 Penyesuaian Hyperparameter pada Skenario Mutual Information 53
4.4.2 Penyesuaian Hyperparameter pada Skenario Recursive Feature Elimination 54
4.5 Pemodelan Ensemble Stacking 54
4.6 Evaluasi Model 57
4.6.1 Confusion Matrix 57
4.6.2 Accuracy, Precision, Recall, F1-Score dan Inference Time 67
4.6.3 ROC dan AUC 71
BAB V KESIMPULAN DAN SARAN 74
5.1 Kesimpulan 74
5.2 Saran 75
DAFTAR PUSTAKA 76
LAMPIRAN 83

DAFTAR TABEL

Tabel 2.2.1 Penelitian Terdahulu mengenai Klasifikasi Serangan Siber 23
Tabel 3.4.1 *Hyperparameter* yang Diuji 34
Tabel 4.1.1 Distribusi Kelas Setelah *Random Sampling* 40
Tabel 4.3.1 Distribusi Kelas Setelah *Random Sampling* Ulang 45
Tabel 4.3.2 Transformasi Label Target Menggunakan *Label Encoding* 46
Tabel 4.3.3 Perbandingan Metode Penskalaan 47
Tabel 4.3.4 Daftar Nama Fitur Berdasarkan Hasil Seleksi Fitur dengan Cross Validation 50
Tabel 4.3.5 Perbandingan Akurasi Model pada Beberapa Metode Seleksi Fitur 51
Tabel 4.4.1 Nilai Hyperparameter Optimal dari Mutual Information 53
Tabel 4.4.2 Nilai *Hyperparameter* Optimal dari RFE 54
Tabel 4.5.1 Kombinasi Model *Stacking* Berdasarkan Studi Terdahulu 55
Tabel 4.6.1 Kinerja Model pada Seleksi Fitur MI dan RFE 67
Tabel 4.6.2 Kinerja Model *Ensemble Stacking* pada Seleksi Fitur MI 69
Tabel 4.6.3 Kinerja Model *Ensemble Stacking* pada Seleksi Fitur RFE 70

DAFTAR GAMBAR

Gambar 1.1.1 Tren Jumlah Serangan DDoS 2
Gambar 2.1.1 Taksonomi Serangan DDoS 7
Gambar 2.1.2 Arsitektur *Ensemble Learning Stacking* 15
Gambar 3.1 Metodologi Penelitian 28
Gambar 4.1.1 Rincian *File Training Day* CIC-DDoS2019 39
Gambar 4.1.2 Distribusi Jumlah Baris Data pada Setiap *File Training Day* Dataset CIC-DDoS2019 40
Gambar 4.2.1 Struktur dan Tipe Data 41
Gambar 4.2.2 Statistik Deskriptif pada Beberapa Fitur Dataset 42
Gambar 4.2.3 Pemeriksaan *Missing Value* pada Dataset 43
Gambar 4.2.4 Pemeriksaan Data Duplikat 43
Gambar 4.2.5 Jumlah Baris yang Mengandung Nilai *Inf* atau *-Inf* 43
Gambar 4.2.6 Korelasi antar Fitur 44
Gambar 4.3.1 Proses Penghapusan Fitur 46
Gambar 4.3.2 Pembagian Data Latih dan Data Uji 47
Gambar 4.3.3 MI dengan *Cross Validation* 49
Gambar 4.3.4 Pearson's dengan *Cross Validation* 49
Gambar 4.3.5 RFE dengan *Cross Validation* 49
Gambar 4.3.6 Nilai Regularisasi Optimal Lasso dengan *Cross Validation* 50
Gambar 4.6.1 *Confusion Matrix* MI Model DT 57
Gambar 4.6.2 *Confusion Matrix* MI Model SVM 57
Gambar 4.6.3 Confusion Matrix MI Model NB 58
Gambar 4.6.4 *Confusion Matrix* MI Model LR 58
Gambar 4.6.5 *Confusion Matrix* MI Model KNN 58
Gambar 4.6.6 *Confusion Matrix* MI Model RF 58
Gambar 4.6.7 *Confusion Matrix* MI Model AB 59
Gambar 4.6.8 *Confusion Matrix* MI Model XGB 59
Gambar 4.6.9 Confusion Matrix MI Model Stacking_1 (`passthrough=True`) 59
Gambar 4.6.10 *Confusion Matrix* MI Model Stacking_1 (`passthrough=False`) 59
Gambar 4.6.11 *Confusion Matrix* MI Model Stacking_2 (`passthrough=True`) 60
Gambar 4.6.12 *Confusion Matrix* MI Model Stacking_2 (`passthrough=False`) 60

Gambar 4.6.13 *Confusion Matrix* MI Model Stacking_3 (passthrough=True).....60
Gambar 4.6.14 *Confusion Matrix* MI Model Stacking_2 (passthrough=False).....60
Gambar 4.6.15 *Confusion Matrix* MI Model Stacking_4 (passthrough=True).....61
Gambar 4.6.16 *Confusion Matrix* MI Model Stacking_4 (passthrough=False).....61
Gambar 4.6.17 *Confusion Matrix* MI Model Peneliti (passthrough=True).....61
Gambar 4.6.18 *Confusion Matrix* MI Model Peneliti (passthrough=False) 61
Gambar 4.6.19 *Confusion Matrix* RFE Model DT 62
Gambar 4.6.20 *Confusion Matrix* RFE Model SVM.....62
Gambar 4.6.21 *Confusion Matrix* RFE Model NB.....63
Gambar 4.6.22 *Confusion Matrix* RFE Model LR 63
Gambar 4.6.23 *Confusion Matrix* RFE Model KNN.....63
Gambar 4.6.24 *Confusion Matrix* RFE Model RF.....63
Gambar 4.6.25 *Confusion Matrix* RFE Model AB 64
Gambar 4.6.26 *Confusion Matrix* RFE Model XGB 64
Gambar 4.6.27 *Confusion Matrix* RFE Model Stacking_1 (passthrough=True) 64
Gambar 4.6.28 *Confusion Matrix* RFE Model Stacking_1 (passthrough=False).....64
Gambar 4.6.29 *Confusion Matrix* RFE Model Stacking_2 (passthrough=True) 65
Gambar 4.6.30 *Confusion Matrix* RFE Model Stacking_2 (passthrough=False).....65
Gambar 4.6.31 *Confusion Matrix* RFE Model Stacking_3 (passthrough=True) 65
Gambar 4.6.32 *Confusion Matrix* RFE Model Stacking_3 (passthrough=False).....65
Gambar 4.6.33 *Confusion Matrix* RFE Model Stacking_4 (passthrough=True) 66
Gambar 4.6.34 *Confusion Matrix* RFE Model Stacking_4 (passthrough=False).....66
Gambar 4.6.35 *Confusion Matrix* RFE Model Peneliti (passthrough=True) 66
Gambar 4.6.36 *Confusion Matrix* RFE Model Peneliti (passthrough=False).....66
Gambar 4.6.37 Kurva ROC dengan Fitur Seleksi MI.....72
Gambar 4.6.38 Kurva ROC dengan Fitur Seleksi RFE 73

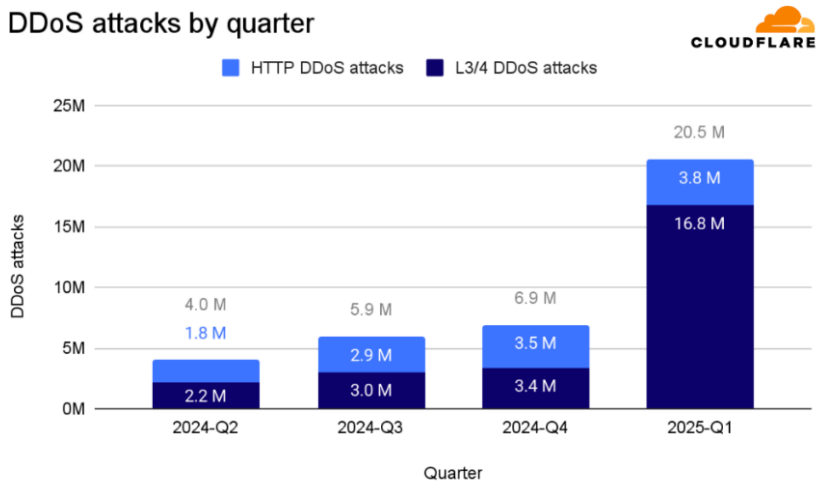
BAB I
PENDAHULUAN

1.1 Latar Belakang

Kemajuan teknologi informasi telah menjadikan jaringan komputer sebagai infrastruktur vital yang menopang hampir seluruh aktivitas digital modern. Ketergantungan yang tinggi terhadap jaringan ini membawa berbagai kemudahan, tetapi sekaligus membuka celah bagi munculnya ancaman-ancaman serius yang dapat mengganggu kestabilan dan keamanan sistem. Ancaman siber kini menjadi semakin kompleks dan canggih sehingga kebutuhan akan keamanan siber menjadi semakin penting dibandingkan sebelumnya (Najar & Manohar Naik, 2024). Serangan siber dapat berdampak pada ekonomi, rusaknya reputasi, dan bahkan hilangnya data sensitif (Najar & Manohar Naik, 2024). Salah satu ancaman yang paling umum dan berbahaya adalah serangan *Distributed Denial of Service* (DDoS), yang dapat mengakibatkan gangguan besar pada ketersediaan layanan dan kerugian finansial yang signifikan.

Serangan DDoS menggunakan banyak perangkat yang dikompromikan untuk membanjiri jaringan lalu lintas dengan berlebihan sehingga mengganggu ketersediaan layanan, menurunkan kinerja, dan bahkan dapat menghentikan fungsi sistem jaringan sepenuhnya (Hossain & Islam, 2024; Mall, Abhishek, Manimurugan, Shankar, & Kumar, 2023). Serangan DDoS bisa dilancarkan dari berbagai lokasi di seluruh dunia. Karena sifatnya yang tersebar luas, serangan ini sulit untuk dicegah atau dihentikan secara efektif (Hossain & Islam, 2024). Tantangan dalam mendeteksi serangan DDoS adalah sulitnya membedakan lalu lintas jaringan normal dengan serangan (Hossain & Islam, 2024).

Berdasarkan laporan Cloudflare (Yoachimik & Pacheco, 2025), pada kuartal pertama tahun 2025, Cloudflare berhasil memblokir lebih dari 20,5 juta serangan DDoS. Jumlah ini mencerminkan peningkatan signifikan sebesar 358% dibandingkan tahun sebelumnya pada kuartal yang sama, serta peningkatan 198% dibandingkan kuartal sebelumnya. Gambaran lebih rinci terkait tren pertumbuhan serangan DDoS per kuartal dapat dilihat pada Gambar 1.1.1. Grafik tersebut menampilkan akumulasi jumlah serangan mulai dari kuartal kedua tahun 2024 hingga kuartal pertama tahun 2025, dengan peningkatan tajam yang terjadi pada Q1 2025. Serangan DDoS yang terjadi pada lapisan jaringan (Layer 3/4) menunjukkan pertumbuhan paling signifikan dan menjadi kontributor utama terhadap peningkatan total jumlah serangan.



Gambar 1.1.1 Tren Jumlah Serangan DDoS

Sumber: (Yoachimik & Pacheco, 2025).

Sebagai upaya untuk mengatasi permasalahan tersebut, dikembangkan berbagai model klasifikasi berbasis *machine learning* untuk mendeteksi dan mengklasifikasikan serangan DDoS. Model ini dirancang untuk membedakan antara lalu lintas jaringan normal dengan berbagai jenis serangan DDoS. Beberapa penelitian sebelumnya telah menunjukkan efektivitas metode ini. Pada penelitian yang dilakukan oleh (Fathima, Devi, & Faizaanuddin, 2023), algoritma Logistic Regression berhasil mencapai akurasi sebesar 91,1%, Random Forest mencapai 97,6%, dan KNN mencapai 97% dalam membedakan lalu lintas jaringan normal dengan serangan DDoS. Penelitan lain yang dilakukan oleh (Gupta dkk., 2022), memanfaatkan algoritma Random Forest dan berhasil mencapai akurasi 99,9%, Decision Tree mencapai akurasi 99,9%, KNN mencapai akurasi 99,8%, Support Vector Machine mencapai akurasi 99,4%, Logistic Regession mencapai akurasi 98,3%, dan Naïve Bayes mencapai akurasi 82% dalam membedakan lalu lintas jaringan normal dengan serangan DDoS. Meskipun hasil akurasi dari penelitian tersebut tinggi, penerapan metode yang digunakan masih terbatas pada klasifikasi biner, yaitu hanya membedakan antara lalu lintas normal (*benign traffic*) dan serangan DDoS, tanpa mempertimbangkan klasifikasi *multiclass* untuk mengidentifikasi berbagai jenis serangan secara spesifik. Identifikasi berbagai jenis serangan secara spesifik dalam klasifikasi serangan DDoS penting dilakukan agar sistem pertahanan dapat merespons dengan strategi mitigasi yang tepat sesuai karakteristik masing-masing serangan. Dengan didasarkan pada pemahaman bahwa setiap model *machine learning* memiliki keterbatasan dan

potensi kesalahan, penggunaan algoritma individual sering kali tidak cukup untuk menangani permasalahan deteksi serangan yang lebih kompleks.

Solusi yang dapat diterapkan untuk mengatasi keterbatasan tersebut adalah penggunaan pendekatan *ensemble learning*. *Ensemble learning* adalah suatu pendekatan dalam *machine learning* yang mengombinasikan beberapa algoritma pembelajaran, dikenal sebagai *weak learners* atau *base models*, guna meningkatkan kemampuan prediksi dari masing-masing algoritma individual (Comito & Pizzuti, 2022). *Ensemble learning* didasarkan pada pemahaman bahwa setiap model *machine learning* memiliki keterbatasan dan potensi kesalahan sehingga pendekatan ini bertujuan meningkatkan kinerja model dengan memanfaatkan keunggulan dari berbagai model dasar secara bersamaan (Mienye & Sun, 2022).

Dalam penelitian ini, dikembangkan sebuah model berbasis *ensemble learning* untuk mengklasifikasikan serangan DDoS dalam skenario klasifikasi *multiclass*. Model ini dibangun dengan mengombinasikan beberapa algoritma *machine learning*, yaitu K-Nearest Neighbors, Naive Bayes, Logistic Regression, Decision Tree, Support Vector Machine, Adaptive Boosting, Extreme Gradient Boosting, dan Random Forest. Berbagai kombinasi algoritma serta konfigurasi *ensemble learning* yang dihasilkan akan diuji untuk memperoleh performa terbaik. Kinerja *model ensemble* selanjutnya dibandingkan dengan model individual untuk menilai sejauh mana pendekatan *ensemble learning* mampu meningkatkan kemampuan prediksi dan ketahanan model dalam mengklasifikasikan berbagai jenis serangan DDoS.

1.2 Rumusan Masalah

Berdasarkan pada latar belakang yang telah dipaparkan, terdapat beberapa rumusan masalah yaitu sebagai berikut:

1. Bagaimana mengembangkan model klasifikasi serangan DDoS berbasis *ensemble learning* untuk mengklasifikasikan berbagai jenis serangan DDoS dalam kasus klasifikasi *multiclass*?
2. Bagaimana kinerja model *ensemble learning* dibandingkan dengan model individual dalam mengklasifikasikan serangan DDoS pada skenario klasifikasi *multiclass*?

1.3 Batasan Masalah

Berikut ini batasan masalah yang digunakan dalam penelitian ini:

1. Penelitian ini hanya berfokus pada klasifikasi serangan DDoS.

2. Dataset yang digunakan dalam penelitian ini yaitu dataset CIC-DDoS2019 yang diunduh dari situs resmi University of New Brunswick, melalui Canadian Institute for Cybersecurity (CIC) ("DDoS 2019 | Datasets | Research | Canadian Institute for Cybersecurity | UNB").
3. Pengambilan data dilakukan dengan dua tahap *random sampling*: tahap pertama mengambil maksimal 50.000 *instance* per kelas dari dataset asli untuk keperluan eksplorasi awal dan pembersihan data; kemudian dilakukan *random sampling* ulang sebanyak maksimal 30.000 *instance* per kelas dari data yang telah dibersihkan. Data hasil *sampling* kedua inilah yang digunakan dalam seluruh proses lanjutan seperti penskalaan data, seleksi fitur, penyesuaian *hyperparameter*, pelatihan model, dan evaluasi model. Pembatasan jumlah data dilakukan karena keterbatasan sumber daya komputasi.
4. Penelitian tidak mencakup mitigasi atau respons terhadap serangan DDoS secara *real-time*, melainkan berfokus pada pengembangan model untuk klasifikasi serangan DDoS berbasis data historis.
5. Evaluasi kinerja model terbatas pada metriks *accuracy*, *recall*, *precision*, *f1-score*, *Receiver Operating Characteristic* (ROC), *Area Under Curve* (AUC), dan *inference time*.

1.4 Tujuan Penelitian

Penelitian ini memiliki beberapa tujuan utama yang ingin dicapai dalam pengembangan model klasifikasi serangan DDoS berbasis ensemble learning:

1. Mengembangkan model klasifikasi serangan DDoS menggunakan pendekatan *ensemble learning* berbasis *stacking* untuk klasifikasi serangan DDoS.
2. Menganalisis dan mengevaluasi kinerja model *ensemble stacking* dalam mengklasifikasikan serangan DDoS berdasarkan metriks *accuracy*, *precision*, *recall*, *f1-score*, *Receiver Operating Characteristic* (ROC), *Area Under Curve* (AUC), dan *inference time*.
3. Membandingkan performa model *ensemble stacking* dengan algoritma individual dalam skenario klasifikasi *multiclass*.

1.5 Manfaat Penelitian

Selain tujuan utama di atas, penelitian ini juga memberikan beberapa manfaat yang diharapkan dapat berkontribusi dalam bidang keamanan siber:

1. Penelitian ini dapat menambah wawasan dan pengetahuan di bidang keamanan jaringan, khususnya dalam penerapan teknik *machine learning* untuk klasifikasi serangan DDoS
2. Memberikan kontribusi referensi ilmiah dalam mengeksplorasi potensi metode *ensemble learning* untuk pengembangan model klasifikasi serangan DDoS.

1.6 Sistematika Penulisan

Penelitian ini disusun secara sistematis dalam lima bab, yaitu:

1. BAB I PENDAHULUAN

   Berisi latar belakang penelitian, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, serta sistematika penulisan yang digunakan.
2. BAB II KAJIAN PUSTAKA

   Menguraikan landasan teori yang digunakan dalam penelitian ini serta tinjauan pustaka yang berisi studi terdahulu sebagai acuan dan pembanding terhadap penelitian yang dilakukan.
3. BAB III METODOLOGI PENELITIAN

   Menjelaskan tahapan penelitian secara rinci, meliputi pengumpulan data, analisis data ekploratif, prepemrosesan data, pemilihan model *ensemble learning*, pelatihan model, hingga evaluasi kinerja model yang digunakan dalam penelitian.
4. BAB IV HASIL DAN PEMBAHASAN

   Memaparkan hasil penelitian yang diperoleh, dilanjutkan dengan analisis dan pembahasan secara mendalam terkait kelebihan, kekurangan, serta perbandingan kinerja model yang dikembangkan dalam penelitian ini.
5. BAB V KESIMPULAN DAN SARAN

   Berisi kesimpulan dari keseluruhan penelitian, serta saran yang dapat digunakan sebagai pertimbangan untuk pengembangan penelitian selanjutnya.

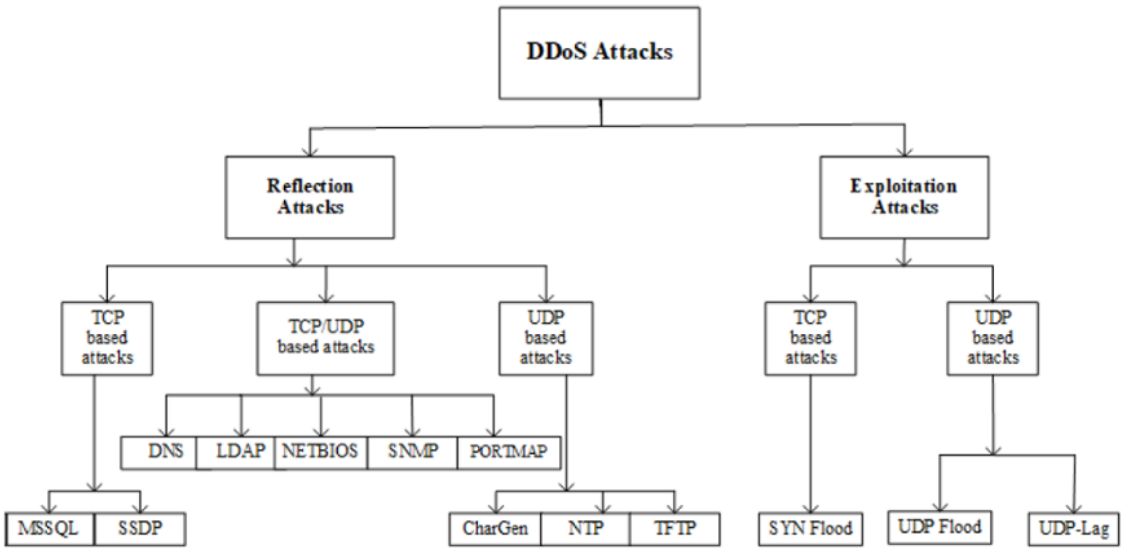
BAB II
KAJIAN PUSTAKA

2.1 Landasan Teori

Pada bagian ini akan membahas landasan teoritis yang menjadi pondasi utama dalam penelitian ini. Landasan teori memiliki peranan yang sangat penting untuk memperkaya pemahaman terhadap konsep-konsep utama yang terlibat dalam penelitian ini. Dengan pemahaman yang mendalam mengenai teori-teori yang relevan, diharapkan penelitian ini dapat memberikan kontribusi dalam memperluas wawasan di bidang *machine learning* dan keamanan siber.

2.1.1 Distributed Denial of Service Attack (DDoS)

Distributed Denial of Service (DDoS) adalah salah satu bentuk serangan siber yang dilakukan dengan melibatkan banyak komputer untuk membanjiri lalu lintas internet ke sistem target sehingga sistem mengalami penurunan kinerja secara signifikan atau bahkan menjadi tidak responsif sama sekali (Butt dkk., 2024). Dalam serangan DDoS, pelaku kejahatan mengendalikan banyak mesin yang telah terinfeksi, dikenal sebagai botnet, untuk mengirimkan sejumlah besar permintaan palsu sehingga membanjiri sumber daya korban dan membuatnya tidak dapat diakses oleh pengguna yang sah (Alhayani & Murphy, 2023). Setelah botnet terbentuk, penyerang dapat mengarahkan serangan dengan mengirimkan perintah jarak jauh ke setiap bot yang merupakan perangkat internet valid sehingga lalu lintas serangan sulit dibedakan dari lalu lintas yang sah (Adedeji, Abu-Mahfouz, & Kurien, 2023). Dalam penelitian yang dilakukan oleh (Sharafaldin, Lashkari, Hakak, & Ghorbani, 2019), serangan DDoS dikelompokkan menjadi dua kategori utama berdasarkan teknik penyerangannya, yaitu serangan berbasis refleksi (*reflection-based attacks*) dan serangan berbasis eksploitasi (*exploitation-based attacks*). Klasifikasi ini disusun untuk menjelaskan bagaimana pola serangan bekerja dan dari mana sumber serangannya berasal. Diagram taksonomi serangan ini disajikan secara visual dalam Gambar 2.1.1.



Gambar 2.1.1 Taksonomi Serangan DDoS

Sumber: (Sharafaldin dkk., 2019)

Serangan DDoS Berbasis Refleksi (Reflection-based Attacks)

Serangan berbasis refleksi merupakan jenis serangan DDoS yang dilakukan secara tidak langsung, di mana pelaku serangan menyembunyikan identitasnya dengan memanfaatkan layanan dari pihak ketiga yang sah sebagai perantara (reflektor). Teknik ini dilakukan dengan mengirimkan paket permintaan ke server reflektor menggunakan alamat IP sumber yang telah dipalsukan, yakni disamakan dengan alamat IP milik korban. Akibatnya, server reflektor akan merespons permintaan tersebut dan mengirimkan paket balasan ke korban, bukan ke pelaku serangan.

Serangan ini umumnya dilakukan melalui *application layer protocols* menggunakan *transport layer protocols*, seperti *Transmission Control Protocol* (TCP), *User Datagram Protocol* (UDP), atau kombinasi keduanya. Beberapa contoh serangan *reflection-based* berbasis TCP antara lain adalah MSSQL dan SSDP. Sementara itu, serangan *reflection-based* yang memanfaatkan UDP mencakup CharGen, NTP, dan TFTP. Terdapat serangan tertentu yang dapat digunakan melalui TCP maupun UDP, seperti DNS, LDAP, NetBIOS, dan SNMP.

Serangan DDoS Berbasis Eksploitasi (Exploitation-based Attacks)

Berbeda dari *reflection-based*, serangan *exploitation-based* dilakukan secara langsung ke sistem target tanpa menggunakan pihak ketiga. Tujuan dari serangan ini adalah

mengeksploitasi cara sistem menangani permintaan jaringan atau koneksi, misalnya dengan menyalahgunakan proses handshake TCP atau mengirimkan paket UDP secara berlebihan ke *port* acak. Serangan ini dapat dilakukan melalui *application layer protocols* menggunakan *transport layer protocols*, seperti TCP dan UDP. Contoh serangan eksploitasi berbasis TCP adalah SYN Flood, sementara serangan berbasis UDP meliputi UDP Flood dan UDP-Lag.

2.1.2 Dataset

Dataset merupakan kumpulan instance atau data yang digunakan untuk membangun maupun mengevaluasi kinerja model *machine learning* (Gong, Liu, Xue, Li, & Meng, 2023). Dataset mencakup berbagai tipe data, seperti data numerik, teks, gambar, atau suara, sesuai dengan masalah yang ingin diselesaikan. Umumnya, dataset dibagi menjadi dua bagian utama, yaitu *training set* yang digunakan untuk melatih model dan *test set* yang digunakan untuk mengevaluasi kinerja model.

CIC-DDoS2019

Dataset CIC-DDoS2019 merupakan kumpulan data yang dikembangkan oleh Canadian Institute for Cybersecurity (CIC) untuk merepresentasikan berbagai jenis serangan Distributed Denial of Service (DDoS) dalam lingkungan jaringan yang realistis. Data dikumpulkan dari aktivitas lalu lintas jaringan yang berlangsung selama beberapa hari, dan mencakup berbagai skenario serangan.

Proses pengumpulan data dilakukan secara menyeluruh, mencakup rekaman packet capture (PCAP) dan *event log*. Untuk mengekstrak informasi penting dari data mentah tersebut, digunakan alat CICFlowMeter-V3 yang mampu menghasilkan lebih dari 80 fitur lalu lintas jaringan. Setiap aliran data kemudian diberi label berdasarkan informasi seperti waktu kejadian, alamat IP, *port*, protokol, dan jenis serangan yang terjadi. Hasil ekstraksi ini disimpan dalam format file CSV secara terpisah untuk setiap mesin sehingga memudahkan analisis lebih lanjut. Struktur data yang rapi dan detail ini menjadikan CIC-DDoS2019 sebagai salah satu dataset yang sangat berguna untuk penelitian dan pengembangan model deteksi serangan DDoS berbasis *machine learning*.

CIC-DDoS2019 mencakup berbagai jenis serangan DDoS reflektif modern seperti PortMap, NetBIOS, LDAP, MSSQL, UDP, UDP-Lag, SYN, NTP, DNS, dan SNMP, yang direkam dalam skenario simulasi serangan nyata selama periode tertentu. Pada tahap pelatihan, terdapat 12 jenis serangan yang terekam, meliputi NTP, DNS, LDAP, MSSQL, NetBIOS,

SNMP, SSDP, UDP, UDP-Lag, WebDDoS, SYN, dan TFTP. Sementara itu, pada tahap pengujian, terdapat 7 jenis serangan yang direkam, yaitu PortScan, NetBIOS, LDAP, MSSQL, UDP, UDP-Lag, dan SYN.

2.1.3 Machine Learning

Machine learning (ML) merupakan bagian dari kecerdasan buatan (AI) yang berkaitan dengan proses melatih komputer agar mampu membuat prediksi menggunakan dataset dan algoritma yang tersedia (Thai, 2022). Sebagaimana dalam studi yang ditulis oleh (Roihan, Abas Sunarya, & Rafika, 2020), *Machine learning* dapat didefinisikan sebagai aplikasi komputer dan algoritma matematika yang diadopsi dengan cara pembelajaran yang berasal dari data dan menghasilkan prediksi di masa yang akan datang. *Machine learning* berkaitan erat dengan data karena data merupakan komponen utama yang digunakan untuk melatih algoritma agar dapat mengenali pola, membuat prediksi, dan mengambil keputusan. Salah satu pendekatan dalam *machine learning* adalah *supervised learning*, yaitu metode yang menggunakan data berlabel untuk menyelesaikan tugas seperti klasifikasi dan regresi.

Supervised Learning

Supervised learning adalah metode yang berfokus pada memperoleh fungsi atau model terbaik untuk menggambarkan pola hubungan antara data berlabel, dengan mempelajari keterkaitan antara variabel *input* (fitur atau prediktor) (Shaer & Shami, 2023; Yüksel, Börklü, Sezer, & Canyurt, 2023). Tujuan dari proses ini adalah untuk mengidentifikasi pola umum yang menghubungkan *input* dengan *output* sehingga dapat dilakukan prediksi terhadap *output* pada data baru yang belum pernah tersedia sebelumnya, di mana nilai *input* telah diketahui namun *output* yang terkait belum teramati (Badillo dkk., 2020). Menurut (Badillo dkk., 2020) terdapat dua kategori utama dalam *supervised learning*: (i) klasifikasi, di mana nilai *output* bersifat kategorikal, dan (ii) regresi, di mana nilai *output* bersifat numerik.

2.1.4 Algoritma Machine Learning

Algoritma *machine learning* merujuk pada teknik komputasi yang dapat menemukan cara untuk menghubungkan sekumpulan *input* dengan *output* yang diinginkan melalui proses pembelajaran dari data yang relevan (Santosh, Das, & Ghosh, 2022). Dalam konteks deteksi serangan DDoS, algoritma *machine learning* digunakan untuk mengklasifikasikan lalu lintas jaringan sebagai normal atau berbahaya berdasarkan pola yang dipelajari dari data historis.

Berikut ini digunakan algoritma *machine learning* dari kategori *supervised learning* yang umum digunakan dalam tugas klasifikasi serangan jaringan.

Decision Tree

Decision Tree (DT) merupakan salah satu algoritma *supervised learning* yang bekerja dengan membagi ruang fitur secara rekursif ke dalam beberapa region berdasarkan nilai dari fitur *input* sehingga membentuk struktur menyerupai pohon (Butt dkk., 2024). Pada Decision Tree, setiap daun merepresentasikan hasil klasifikasi dan setiap cabang merupakan kombinasi fitur yang mengarahkan pada kelas target tertentu (Ye, Dobson, & McKeever, 2012).

Support Vector Machine

Support Vector Machine adalah algoritma *supervised learning* yang memetakan data ke ruang fitur berdimensi lebih tinggi menggunakan fungsi kernel, lalu mencari *hyperplane* optimal untuk klasifikasi secara efektif (Liu dkk., 2024). *Hyperplane* ini merupakan batas keputusan (*decision boundary*) yang digunakan untuk menentukan keanggotaan kelas dari suatu data baru

Naïve Bayes

Naïve Bayes adalah model klasifikasi probabilistik yang didasarkan pada teorema Bayes, dengan asumsi bahwa setiap fitur memberikan kontribusi yang independen dan setara terhadap kelas target (Misra & Li, 2020). Algoritma Naïve Bayes memiliki sejumlah kelebihan, antara lain sangat direkomendasikan untuk digunakan pada data numerik maupun tekstual, serta dikenal mudah diimplementasikan dan digunakan. Namun demikian, algoritma ini juga memiliki beberapa kelemahan, seperti asumsi independensi bersyarat yang sering kali tidak terpenuhi pada data dunia nyata, serta kinerja yang menurun ketika terdapat korelasi tinggi antar fitur.

Logistic Regression

Logistic Regression adalah dalah metode prediksi yang digunakan untuk menentukan atau menjelaskan nilai dari suatu variabel target yang bersifat kategorik, berdasarkan satu atau lebih variabel penjelas yang bisa berupa angka maupun kategori (Messaoud dkk., 2020). Logistic Regression dapat dikembangkan menjadi Multinomial Logistic Regression, yang dapat digunakan untuk menyelesaikan permasalahan klasifikasi *multiclass* dengan

memanfaatkan fungsi *softmax* (Kamble & Dale, 2022). Fungsi *softmax* digunakan untuk mengubah sekumpulan angka menjadi probabilitas, yaitu nilai antara 0 sampai 1, dan jika dijumlahkan semuanya akan menghasilkan 1.

K-Nearest Neighbors

k-Nearest Neighbors (k-NN) adalah algoritma pembelajaran berbasis *instance* yang didasarkan pada prinsip bahwa suatu data cenderung berada dekat dengan data lain yang memiliki sifat atau karakteristik serupa (Cover & Hart, 1967). Algoritma k-Nearest Neighbors (kNN) bekerja layaknya sebuah sistem *voting*, di mana label kelas mayoritas dari *k* tetangga terdekat (dengan *k* merupakan bilangan bulat) menentukan label kelas untuk data baru (Sakr dkk., 2024).

2.1.5 Ensemble Learning

Ensemble learning adalah metode dalam *machine learning* yang mengombinasikan beberapa model algoritma, dikenal sebagai *weak learner* atau *base model*, untuk meningkatkan kinerja prediksi dari algoritma pembelajaran yang digunakan (Comito & Pizzuti, 2022). Dibandingkan mengandalkan satu model tunggal, prediksi yang dihasilkan oleh masing-masing model individual digabungkan menggunakan suatu aturan kombinasi untuk memperoleh prediksi tunggal yang lebih akurat.

Random Forest

Random Forest (RF) adalah metode *ensemble* berbasis teknik *bagging* (*bootstrap aggregating*) yang terdiri dari sejumlah pohon keputusan yang dilatih secara paralel pada *subset* data yang diambil secara acak. Selama proses pelatihan, setiap pohon dibangun dengan *subset* acak dari data *input* (Kisner, Ding, & Thomas, 2022). Setiap pohon memberikan prediksi terhadap suatu kelas, lalu hasil akhir ditentukan berdasarkan suara terbanyak dari semua pohon (Kisner dkk., 2022).

Berdasarkan studi yang dilakukan oleh (Rani dkk., 2023), model Random Forest secara umum bekerja melalui mekanisme berikut ini.

1) Setiap pohon keputusan dilatih pada data yang berbeda-beda, diperoleh melalui proses pemilihan data secara acak (*random sampling*) dari dataset asli. Proses ini menggunakan teknik sampling with replacement, yang memungkinkan satu data

terpilih lebih dari satu kali. Hal ini menghasilkan setiap pohon dilatih pada versi dataset yang dimodifikasi.

2) Setiap pohon hanya menggunakan *subset* fitur secara acak, bukan seluruh fitur yang tersedia. Pendekatan ini berbeda dengan pohon keputusan konvensional yang mempertimbangkan semua fitur dalam menentukan pemisahan terbaik.
3) Pemilihan fitur secara acak meningkatkan keragaman antar pohon sehingga model *ensemble* menjadi lebih *robust* dan tidak terlalu sensitif terhadap pola tertentu dalam data pelatihan.
4) Untuk melakukan klasifikasi, setiap pohon memberikan prediksi secara independen. Hasil akhir ditentukan berdasarkan *majority voting* dari seluruh pohon dalam *ensemble*.

(Rani dkk., 2023) mengemukakan bahwa model Random Forest memiliki beberapa keunggulan, antara lain:

1) Mengurangi risiko *overfitting* sehingga mampu meningkatkan akurasi klasifikasi dibandingkan dengan penggunaan pohon keputusan tunggal.
2) Fleksibel untuk berbagai jenis permasalahan, baik masalah klasifikasi maupun regresi.
3) Mampu menangani *missing value* secara otomatis.

Adaptive Boosting (AdaBoost)

Adaboost merupakan algoritma *machine learning* yang didasarkan pada konsep *boosting*. *Boosting* merupakan salah satu *ensemble learning* yang membentuk himpunan klasifikasi melalui *resampling* data sehingga setiap *classifier* berikutnya memperoleh data latih yang paling informatif (Zhao, 2019). AdaBoost adalah algoritma berbasis iterasi yang bertujuan melatih sejumlah model *weak learner* pada dataset yang sama, kemudian mengombinasikan model-model tersebut untuk membentuk sebuah model akhir yang lebih kuat dan akurat (Zhao, 2019).

Berdasarkan studi (Rani dkk., 2023), secara keseluruhan model AdaBoost bekerja melalui tahapan sebagai berikut:

1) Beberapa *decision stump* dibuat pada tahap awal. *Stump* merupakan pohon keputusan sederhana yang hanya terdiri dari satu *node* dan dua daun. Model ini dikategorikan sebagai *weak learner* yang sesuai untuk pendekatan *boosting*.
2) Data yang salah diklasifikasikan oleh *stump* awal akan diberikan prioritas lebih tinggi. Setelah setiap iterasi, bobot data akan diperbarui, di mana data yang diklasifikasikan salah akan diberi bobot lebih besar.

3) Tingkat kesalahan pada *stump* pertama akan memengaruhi proses pembentukan *stump-stump* berikutnya.
4) Pada iterasi pertama, jumlah *stump* yang dibuat sama dengan jumlah fitur yang ada pada dataset.
5) Dari sekumpulan *stump* tersebut, dipilih *stump* dengan kinerja terbaik sebagai *base learner*.
6) Bobot data kembali diperbarui, dengan penekanan pada *instance* yang salah klasifikasi agar memiliki prioritas lebih tinggi.
7) Dataset baru dibentuk berdasarkan bobot tersebut, di mana *instance* yang salah klasifikasi memiliki frekuensi lebih tinggi dibandingkan *instance* yang diklasifikasikan dengan benar.
8) Proses ini diulang secara iteratif hingga semua data berhasil diklasifikasikan dengan benar, atau hingga jumlah *estimator* yang ditentukan tercapai.
9) Dalam proses klasifikasi, algoritma mempertimbangkan performa masing-masing *stump* saat melakukan *voting*.
10) Dengan demikian, kontribusi *voting* dari tiap stump tidak bersifat setara; stump yang memiliki akurasi lebih tinggi akan memiliki bobot suara yang lebih besar.

Menurut studi yang dilakukan oleh (Rani dkk., 2023), model AdaBoost memiliki sejumlah keunggulan, antara lain:

1) Cenderung memiliki tingkat kesalahan generalisasi yang lebih rendah.
2) Sederhana dan mudah untuk diimplementasikan.
3) Kompatibel dengan berbagai algoritma klasifikasi.

XGBoosting

Extreme Gradient Boosting (XGBoost) adalah algoritma *machine learning* yang bekerja berdasarkan pendekatan *gradient boosted decision trees* (Chen & Guestrin, 2016). Istilah *gradient boosting* berasal dari konsep untuk meningkatkan kinerja sebuah model lemah dengan menggabungkannya bersama beberapa model lemah lainnya secara iteratif hingga membentuk model yang lebih kuat. XGBoost merupakan salah satu algoritma pembelajaran yang dirancang untuk meningkatkan akurasi model secara keseluruhan dengan meminimalkan risiko *overfitting* selama proses pelatihan (Ahmetoglu & Das, 2022).

Berdasarkan studi (Rani dkk., 2023), secara keseluruhan model AdaBoost bekerja melalui tahapan sebagai berikut.

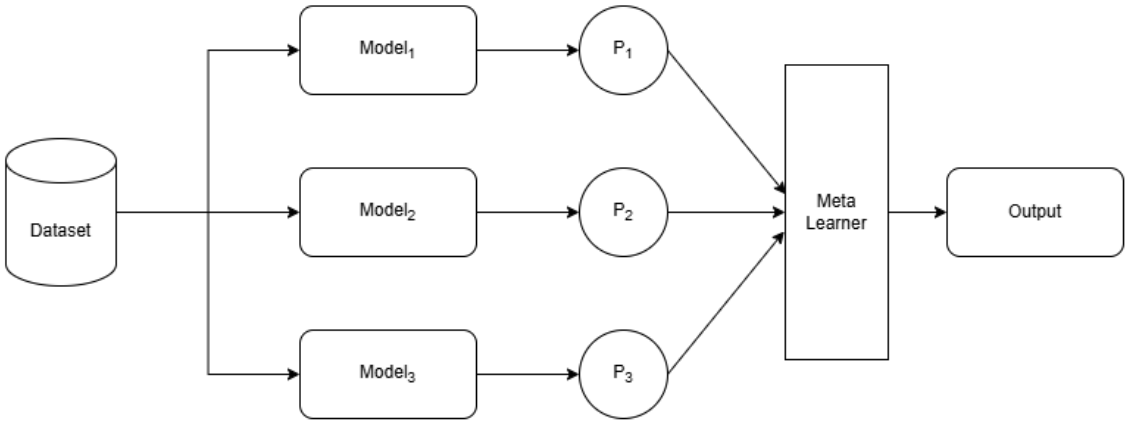
1) Seperti algoritma *gradient boosting* lainnya, XGBoost menggunakan pohon Keputusan baru untuk memperbaiki kesalahan yang dibuat oleh model sebelumnya.
2) Ide utama dari penambahan pohon keputusan ini adalah untuk meminimalkan *loss function* dengan mengoreksi kesalahan yang terjadi pada iterasi sebelumnya.
3) XGBoost tidak membangun struktur pohon secara penuh, melainkan secara *greedy*. Algoritma ini bersifat *level-wise*, yaitu tumbuh berdasarkan tingkat (*level*) alih-alih berdasarkan daun (*leaf-wise*).
4) Karena sifat algoritma *gradient boosting* yang cenderung sensitif terhadap *overfitting*, maka penting untuk melakukan penyetelan *hyperparameter* guna mengurangi risiko tersebut.
5) Komponen regularisasi dalam XGBoost memberikan penalti terhadap model yang terlalu kompleks sehingga membantu mencegah *overfitting*

Menurut (Rani dkk., 2023), model XGBoost memiliki sejumlah keunggulan, antara lain:

1) Memiliki fleksibilitas yang tinggi sehingga dapat disesuaikan dengan berbagai jenis permasalahan.
2) Mendukung pemrosesan paralel (*multiprocessing*), yang memungkinkan proses pelatihan dilakukan lebih cepat dan efisien.
3) Mendukung penerapan regularisasi sehingga membantu mengurangi risiko *overfitting*.
4) Memberikan kontrol yang lebih baik terhadap data yang hilang, memungkinkan penanganan data tidak lengkap secara lebih efektif.

Stacking

Stacking menggabungkan beberapa model dasar (*base models*) untuk meningkatkan akurasi dan ketangguhan (*robustness*) dari hasil prediksi (Arya, Kumar, & Ahmad, 2023).Salah satu keunggulan *stacking* adalah kemampuannya untuk menggunakan model-model yang heterogen sebagai *base learner*, di mana *output* dari beberapa model dasar (*level-0 learners*) digunakan sebagai *input* bagi *meta-learner* (*level-1 learner*) untuk menghasilkan prediksi akhir yang lebih akurat (Lazzarini, Tianfield, & Charissis, 2023). *Stacking* berguna ketika beberapa model *machine learning* memiliki keahlian khusus dalam menyelesaikan tugas tertentu (Mienye & Sun, 2022). Gambaran umum dari *ensemble learning stacking* dapat dilihat pada Gambar 2.1.2.



Gambar 2.1.2 Arsitektur *Ensemble Learning Stacking*

2.1.6 Seleksi Fitur

Menurut (Bellotti, Nouretdinov, Yang, & Gammerman, 2014a), seleksi fitur atau *feature selection* adalah proses memilih *subset* fitur dari sekumpulan fitur yang tersedia dengan tujuan untuk memenuhi satu atau lebih dari tujuan berikut:

1) Memilih *subset* fitur yang memaksimalkan kinerja algoritma pembelajaran.
2) Mengurangi kebutuhan penyimpanan komputer dan/atau waktu komputasi untuk mengklasifikasikan data tanpa secara signifikan menurunkan kinerja algoritma pembelajaran pada suatu permasalahan.
3) Mendeteksi *subset* fitur yang berkaitan dengan permasalahan alami yang sedang diteliti.

Seleksi fitur bertujuan untuk memilih fitur-fitur yang paling representatif dari sekumpulan fitur sehingga meningkatkan diskriminasi antara kelas mayoritas dan minoritas serta meningkatkan akurasi klasifikasi (Yuan dkk., 2023). Proses seleksi fitur memilih *subset* fitur yang lebih kecil namun lebih representatif dari fitur-fitur awal (Nssibi, Manita, & Korbaa, 2023). Terdapat tiga metode umum dalam seleksi fitur, yaitu *filter*, *wrapper*, dan *embedded feature selection* (Bellotti dkk., 2014a).

Filter-based Methods

Filter-based method memilih fitur terbaik dengan menggunakan fungsi pemeringkatan yang memberikan skor relevansi berdasarkan sampel data (Bellotti dkk., 2014a). Semakin tinggi relevansi suatu fitur, maka semakin tinggi pula peringkatnya. Pemilihan fitur dilakukan dengan dua pendekatan: memilih sejumlah fitur dengan peringkat tertinggi secara tetap, atau

memilih fitur-fitur yang memiliki skor di atas ambang batas tertentu (Das, Venugopal, Shiva, & Sheldon, 2020).

a) Mutual Information

Mutual Information adalah ukuran statistik yang digunakan untuk mengevaluasi sejauh mana suatu variabel memberikan informasi terhadap variabel lainnya (Cheng, Sun, Yao, Xu, & Cao, 2022). Nilai Mutual Information yang lebih tinggi mengindikasikan adanya ketergantungan yang lebih kuat antara kedua variabel, artinya satu variabel dapat memberikan informasi yang signifikan terhadap variabel lainnya (Rafie, Moradi, & Ghaderzadeh, 2023). Dalam konteks seleksi fitur, variabel dengan nilai Mutual Information tinggi terhadap variabel target dianggap lebih relevan dan layak untuk dipertahankan dalam proses pemodelan (Zhou, Wang, & Zhu, 2022).

b) Pearson's Correlation

Metode Pearson's Correlation memberikan nilai antara -1 hingga 1, di mana nilai 0 menunjukkan tidak adanya korelasi, nilai 1 menunjukkan korelasi positif sempurna, dan nilai -1 menunjukkan korelasi negatif sempurna (Nettleton, 2014).

Wrapper-based Methods

Wrapper-based method memilih *subset* fitur yang relevan berdasarkan pengukuran kinerja dari suatu metode pembelajaran (Rodriguez-Galiano, Luque-Espinar, Chica-Olmo, & Mendes, 2018). *Subset* fitur yang memberikan kinerja terbaik akan dipilih untuk digunakan pada tahap akhir.

a) Recursive Feature Elimination

Recursive Feature Elimination (RFE) adalah metode seleksi fitur yang bekerja dengan cara melatih sebuah model dan secara bertahap menghapus fitur terlemah (atau beberapa fitur terlemah) hingga jumlah fitur yang ditentukan tercapai (N. V. Sharma & Yadav, 2021). RFE membutuhkan jumlah fitur yang akan dipertahankan sebagai parameter yang harus ditentukan, namun sering kali jumlah fitur yang optimal tersebut tidak diketahui sejak awal (N. V. Sharma & Yadav, 2021).

Embedded Methods

Beberapa algoritma *machine learning* digunakan dalam metode *embedded*, di mana algoritma tersebut memiliki mekanisme bawaan untuk memilih fitur tertentu yang dijalankan selama proses pelatihan model (Bellotti dkk., 2014a). Dalam metode *embedded method*, tugas

seleksi fitur dan klasifikasi terintegrasi dalam satu algoritma secara menyeluruh sehingga kedua proses tersebut tidak dapat dipisahkan (Maldonado, Riff, & Neveu, 2022).

a) LASSO Regression

LASSO melakukan regularisasi L1, yaitu dengan memberikan batasan pada jumlah nilai absolut dari parameter model dan memberikan penalti terhadap koefisien variabel dengan mengecilkan beberapa variabel menuju nol (Das dkk., 2020). Setelah proses regularisasi selesai, fitur-fitur yang memiliki nilai koefisien regresi nol akan dieliminasi (Das dkk., 2020).

2.1.7 Pengukuran Kinerja Model

Pengukuran kinerja model merupakan langkah penting untuk menilai sebaik apa model tersebut mampu melakukan prediksi secara akurat dan konsisten terhadap data. Secara umum, pengukuran kinerja ini dilakukan dengan cara menguji bagaimana performa model *ensemble learning* dibandingkan dengan model individual dalam hal *accuracy*, *precision*, *recall*, *f1-score*, *confusion matrix*, *Receiver Operating Characteristic* (ROC), *Area Under the Curve* (AUC), dan *inference time*.

Dalam proses evaluasi model klasifikasi, diperlukan indikator dasar untuk mengukur tingkat keberhasilan maupun kesalahan prediksi model terhadap data yang diuji. Dasar perhitungan berbagai metrik evaluasi tersebut mengacu pada empat komponen utama, yaitu:

i. *True Positive* (TP): Jumlah data yang memang berasal dari suatu kelas tertentu dan berhasil diprediksi dengan tepat oleh model sebagai bagian dari kelas tersebut.
ii. *True Negative* (TN): Jumlah data yang berasal dari kelas lain dan berhasil dikenali oleh model sebagai bukan bagian dari kelas yang tersebut.
iii. *False Positive* (FP): Jumlah data yang seharusnya berasal dari kelas lain, tetapi salah diprediksi oleh model sebagai bagian dari kelas tertentu.
iv. *False Negative* (FN): Jumlah data yang sebenarnya termasuk dalam suatu kelas tertentu, namun salah diklasifikasikan oleh model sebagai kelas lain.

Confusion Matrix

Confusion matrix merupakan representasi visual yang menunjukkan jumlah prediksi benar dan salah dari model dalam mengklasifikasikan setiap kelas (Hemalatha dkk., 2022). *Confusion matrix* merupakan matriks dua dimensi, di mana satu dimensi merepresentasikan

kelas sebenarnya dari suatu objek, dan dimensi lainnya menunjukkan kelas yang diprediksi oleh model klasifikasi.

Accuracy, Precion, Recall, F1-score

Accuracy merupakan indikator yang menunjukkan tingkat kesesuaian antara hasil prediksi model dengan kondisi atau fakta sebenarnya yang dimodelkan (Sammut & Webb, 2017). *Accuracy* menggambarkan seberapa banyak prediksi model yang benar, baik untuk kelas positif maupun negatif, dibandingkan dengan seluruh data yang diuji. Nilai *accuracy* dapat dihitung menggunakan Persamaan 2.1.

$$Acc = \frac{TP + TN}{TP + FP + TN + FN} \quad (2.1)$$

Precision merupakan perbandingan antara jumlah *true positive* (TP) dengan total keseluruhan prediksi positif yang dihasilkan oleh model (Sammut & Webb, 2017). *Precision* menunjukkan seberapa banyak prediksi positif yang dibuat oleh model benar-benar tepat sasaran. Rumus untuk menghitung *precision* ditunjukkan pada Persamaan 2.2.

$$P = \frac{TP}{TP + FP} \quad (2.2)$$

Recall merupakan perbandingan antara jumlah *true positive* dengan total seluruh *instance* yang benar-benar termasuk dalam kelas positif (Sammut & Webb, 2017). *Recall* mengukur seberapa baik model dalam menemukan seluruh kasus positif yang sebenarnya ada di dalam data. Nilai dari metrics *recall* didapatkan dari Persamaan 2.3.

$$Recall = \frac{TP}{TP + FN} \quad (2.3)$$

F1-Score adalah rata-rata harmonis dari *precision* dan *recall*, yang digunakan untuk menilai keseimbangan antara keduanya. Secara sederhana, *F1-score* memberikan gambaran seberapa baik model mampu menghasilkan prediksi positif yang tepat, sekaligus memastikan tidak banyak kasus positif yang terlewat. Nilai *f1-score* dapat dihitung dengan Persamaan 2.4

$$F1 - Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (2.4)$$

Receiver Operating Characteristic (ROC) dan Area Under the Curve (AUC)

Receiver Operating Characteristic (ROC) adalah representasi visual yang menggambarkan tingkat keberhasilan dan kesalahan yang terjadi pada sebuah model klasifikasi (Roy, Kar, & Das (2015). Kurva ini dibuat dengan menggambarkan *true positive rate* (TPR) pada sumbu vertikal (Y) dan *false positive rate* (FPR) pada sumbu horizontal (X) sehingga bentuk kurva tersebut membantu dalam memahami seberapa akurat prediksi yang dihasilkan oleh model (Roy dkk., 2015).

Inference Time

Inference time adalah waktu yang dibutuhkan oleh model *machine learning* untuk menghasilkan prediksi terhadap satu atau lebih sampel data setelah model tersebut selesai dilatih. *Inference time* merupakan salah satu metrik evaluasi penting untuk menggambarkan efisiensi operasional suatu model, terutama ketika model diimplementasikan dalam sistem nyata yang memerlukan respons cepat dan *real-time*, seperti dalam sistem deteksi intrusi atau deteksi serangan DDoS.

Dalam studi deteksi serangan DDoS, *inference time* menjadi parameter yang krusial karena sistem deteksi sering diharapkan bekerja secara real-time atau *near real-time*. Keterlambatan dalam menghasilkan prediksi dapat menyebabkan kegagalan dalam memitigasi serangan dengan tepat waktu

2.2 Tinjauan Pustaka

Penelitian dalam bidang klasifikasi serangan Distributed Denial of Service (DDoS) terus berkembang, terutama dengan pemanfaatan pendekatan *machine learning* dan *ensemble learning*. Beberapa studi terdahulu telah menunjukkan keberhasilan dalam mengklasifikasikan serangan DDoS, baik secara *binary* maupun *multiclass*. Namun, masih terdapat sejumlah keterbatasan yang membuka peluang untuk penelitian lebih lanjut, khususnya dalam konteks klasifikasi *multiclass* yang kompleks dan penggunaan model *ensemble* heterogen seperti *stacking*.

Studi (Thalapala, Reddy, & Guravaiah, 2025) mengembangkan model deteksi DDoS dengan pendekatan klasifikasi *multiclass* (4 kelas) menggunakan dataset CSE-CIC-IDS2018. Pendekatan yang digunakan mencakup model individual seperti Decision Tree dan Logistic Regression, serta model *bagging* dengan Random Forest. Evaluasi dilakukan menggunakan metrik *accuracy*, *recall*, *f1-score*, dan *confusion matrix*. Namun, studi ini belum mengeksplorasi potensi pendekatan *ensemble* dengan algoritma heterogen, serta belum mencakup metrik seperti *precision*, ROC-AUC, dan *inference time*, yang penting untuk pengukuran performa secara menyeluruh.

Sementara itu, studi (Kaur, Kumar, & Aggarwal, 2025) memanfaatkan dataset SDN-DAD untuk klasifikasi *multiclass* (5 kelas) dengan model individual dan *boosting* (Adaptive Boosting). Algoritma individual yang digunakan meliputi Gaussian Naive Bayes dan Logistic Regression. Evaluasi model yang dilakukan mencakup metrik *accuracy*, *precision*, *recall*, *f1-score*, dan ROC-AUC. Namun demikian, penelitian ini belum mengombinasikan model heterogen dalam *ensemble* dan belum melibatkan metrik *confusion matrix* dan *inference time* dalam evaluasi performa.

Penelitian oleh (Butt dkk., 2024) mengusulkan berbagai model *machine learning* dan *ensemble* untuk klasifikasi biner pada dataset DDoS-SDN. Model individual yang digunakan meliputi Naive Bayes, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Logistic Regression, dan Decision Tree, sedangkan *ensemble* meliputi Random Forest dan Extreme Gradient Boosting (XGBoost). Metrik evaluasi tergolong komprehensif, mencakup *accuracy*, *precision*, *recall*, *f1-score*, *specificity*, *confusion matrix*, dan ROC-AUC. Namun, pendekatan *ensemble* masih homogen dan terbatas pada klasifikasi biner.

(Sakr dkk., 2024) mengembangkan model deteksi serangan DDoS menggunakan dataset CICDDoS2019. Model individual seperti Decision Tree, SVM, dan KNN, serta *ensemble* Random Forest dan Gradient Boosting, digunakan dalam penelitian ini. Evaluasi dilakukan dengan *accuracy*, *precision*, *recall*, dan *f1-score*. Namun, studi ini belum melibatkan kombinasi algoritma heterogen, serta tidak mencakup metrik evaluasi seperti *confusion matrix*, ROC-AUC, dan *inference time*. Masalah klasifikasi juga masih terbatasa pada klasifikasi biner.

Pada studi (Kumar & Ishigaki, 2024) memperkenalkan model *ensemble stacking* dengan pendekatan *meta-learner* berbasis Logistic Regression. Model ini menggabungkan LSTM, Random Forest, dan KNN sebagai *base-learners* untuk klasifikasi biner menggunakan dataset CICDDoS2019. Evaluasi dilakukan menggunakan *accuracy*, *f1-score*, dan *confusion matrix*. Namun, metrik evaluasi masih belum lengkap karena tidak mencakup *precision*, *recall*, ROC-AUC, dan *inference time*, serta terbatas pada klasifikasi biner.

Penelitian serupa dilakukan oleh (Alasali & Dakkak, 2024) dengan pendekatan *multi-layer stacking* dalam lingkungan SDN. *Base-learners* yang digunakan meliputi Random Forest, KNN, Decision Tree, Logistic Regression, dan Multilayer Perceptron, dengan *meta-learner* berupa Random Forest. Evaluasi dilakukan dengan metrik *accuracy*, *sensitivity*, *specificity*, *precision*, *recall*, *f1-score*, dan ROC-AUC. Namun, fokus penelitian masih terbatas pada klasifikasi biner serta belum menggunakan metriks *confusion matrix* dan *inference time*.

Studi (Rani dkk., 2023) menggunakan pendekatan *machine learning* pada komunikasi Device-to-Device (D2D) dengan dataset CICDDoS2019. Model *ensemble* yang digunakan termasuk Random Forest, Extreme Gradient Boosting, Adaptive Boosting, dan Light Gradient Boosting Machine. Meskipun metrik evaluasi seperti *accuracy*, *precision*, *recall*, *f1-score*, ROC-AUC, dan *error rate* digunakan, penelitian ini tidak menyertakan *confusion matrix* dan *inference time*, serta belum menerapkan pendekatan *ensemble* dengan kombinasi algoritma yang heterogen.

Studi oleh (Arya dkk., 2023) menerapkan pendekatan *ensemble stacking* untuk deteksi serangan DDoS dengan *base-learners* seperti SVM, Naive Bayes, Decision Tree, dan Random Forest, serta *meta-learner* Logistic Regression. Metrik evaluasi meliputi *accuracy*, *precision*, *recall*, dan *f1-score*. Meski telah menggunakan *stacking*, studi oleh (Arya dkk., 2023) masih menerapkan klasifikasi biner serta evaluasi performa model masih terbatas karena tidak mencakup *confusion matrix*, ROC-AUC, dan *inference time*.

Berdasarkan kajian literatur di atas, dapat disimpulkan bahwa sebagian besar penelitian terdahulu masih fokus pada klasifikasi biner dan penggunaan *ensemble* yang bersifat homogen.

Belum banyak studi yang mengintegrasikan model *stacking* dengan algoritma heterogen dalam konteks klasifikasi *multiclass*. Oleh karena itu, penelitian ini bertujuan untuk mengisi celah tersebut dengan mengembangkan model *ensemble stacking* dengan kombinasi algoritma heterogen untuk klasifikasi *multiclass* serangan DDoS, serta mengevaluasi performanya secara komprehensif menggunakan berbagai metrik evaluasi termasuk *accuracy*, *precision*, *recall*, *f1-score*, ROC-AUC, *confusion matrix*, dan *inference time*.

Tabel 2.2.1 Penelitian Terdahulu mengenai Klasifikasi Serangan Siber

Judul	Klasifikasi	Dataset	Pendekatan	Algoritma Machine Learning	Matriks Evaluasi	Batasan
MLADAD: Machine Learning Algorithms Analysis on DDoS Attack Detection in SDN-IoT Networks (Thalapala dkk., 2025)	*Multiclass (4 class)*	CSE-CIC-IDS2018	Model Individual	Decision Tree	*accuracy, recall, f1-score, confusion matrix*	Belum mengeksplorasi pendekatan *ensemble* dengan algoritma heterogen; metrik evaluasi belum mencakup *precision*, ROC-AUC, dan *inference time.*
				Logistic Regression		
			Model Bagging	Random Forest		
Enhancing DDoS Defense in SDN Using Hierarchical Machine Learning Models (Kaur dkk., 2025)	*Multiclass (5 class)*	SDN-DAD	Model Individual	Gaussian Naive Bayes	*accuracy, precision, recall, f1-score, ROC-AUC*	Model terbatas pada *ensemble boosting* dan individual, belum menggabungkan algoritma yang heterogen; evaluasi model belum mencakup *confusion matrix* dan *inference time*.
				Logistic Regression		
			Model Boosting	Adaptive Boosting		
Enhanced DDoS Detection Using Advanced Machine Learning and Ensemble Techniques in Software Defined Networking (Butt dkk., 2024)	*Binary*	DDoS SDN	Model Individual	Naive Bayes	*accuracy, precision, recall, f1-score, specificity, confusion matrix, ROC-AUC, run time*	Fokus masih pada klasifikasi biner; pendekatan *ensemble* masih homogen; meskipun metrik cukup komprehensif, *inference time* belum ditunjukkan secara eksplisit.
				K-Nearest Neighbors		
				Support Vector Machine		
				Logistic Regression		
				Decision Tree		
			Model Bagging	Random Forest		
			Model Boosting	Extreme Gradient Boosting		

Machine Learning-Based Detection of DDoS Attacks on IoT Devices in Multi-Energy Systems (Sakr dkk., 2024)	*Binary*	CICDDoS2019	Model Individual	Decision Tree	*accuracy, precision, recall, f1-score*	Evaluasi model terbatas pada klasifikasi biner; Tidak mencakup pendekatan *ensemble* heterogen; metrik evaluasi belum mencakup *confusion matrix*, ROC-AUC dan *inference time*
				Support Vector Machine		
				K-Nearest Neighbors		
			Model Bagging	Random Forest		
			Model Boosting	Gradient Boosting		
Advanced DDoS Attack Classification Using Ensemble Model with Meta-Learner (Kumar & Ishigaki, 2024)	*Binary*	CICDDoS2019	Model Individual	Support Vector Machine	*accuracy, f1-score, confusion matrix*	Masih terbatas pada klasifikasi biner; evaluasi metrik tidak mencakup *precision*, *recall*, ROC-AUC, dan *inference time*
				K-Nearest Neighbors		
			Model Bagging	Random Forest		
			Model Stacking	*Base-learners:* *· LSTM* *· Random Forest* *· K-Nearest Neighbors* *Meta-learner:* *· Logistic Regression*		

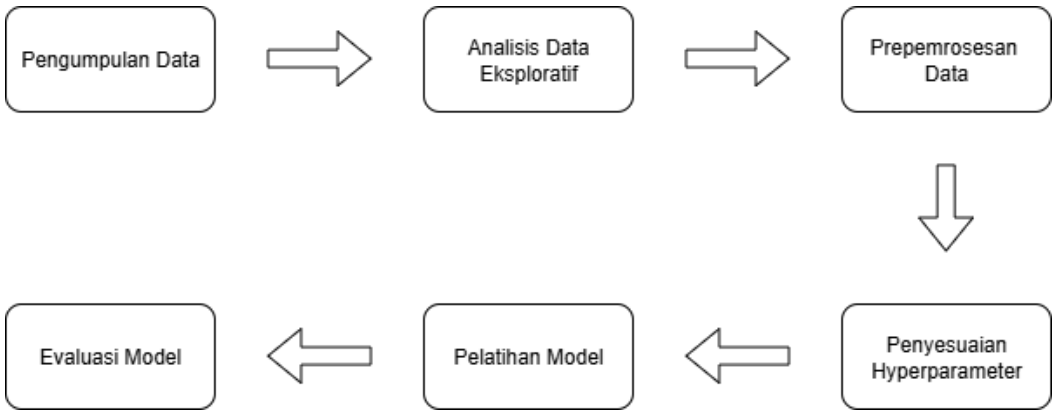
A Novel DDoS Detection Method Using Multi-Layer Stacking in SDN Environment (Alasali & Dakkak, 2024)	*Binary*	-	Model Individual	K-Nearest Neighbors	*accuracy, sensitivity, specificity, precision, recall, f1-score, ROC-AUC*	Evaluasi model terbatas pada klasifikasi biner; metrik evaluasi belum mencakup *confusion matrix* dan *inference time*
				Decision Tree		
				Logistic Regression		
				Multilayer Perceptron		
			Model Bagging	Random Forest		
			Model Stacking	*Base-learners:* *· Random Forest* *· K-Nearest Neighbors* *· Decision Tree* *· Logistic Regression* *· Multilayer Perceptron* *Meta-learner:* *· Random Forest*		
Detection of DDoS Attacks in D2D Communications Using Machine Learning Approach (Rani dkk., 2023)	*Binary*	CICDDoS2019	Model Bagging	Random Forest	*accuracy, precision, recall, f1-score, ROC-AUC, error rate*	Evaluasi masih terbatas pada klasifikasi biner; belum mencakup ensemble dengan algoritma heterogen; metrik evaluasi belum mencakup *confusion matrix* dan *inference time*
			Model Boosting	Extreme Gradient Boosting		
				Adaptive Boosting		
				Light Gradient Boosting Machine		

DDoS Attack Detection Using Ensemble Machine Learning Approach (Arya dkk., 2023)	*Binary*	CICDDoS2019	Model Individual	Support Vector Machine	*accuracy, precision, recall, f1-score*	Evaluasi masih terbatas pada klasifikasi biner; metrik evaluasi tidak mencakup *confusion matrix*, ROC-AUC, dan *inference time*
				Naive Bayes		
				Decision Tree		
			Model Bagging	Random Forest		
			Model Stacking	*Base-learners:* · *Support Vector Machine* · *Naive Bayes* · *Decision Tree* · *Random Forest* *Meta-learner:* · *Logistic Regression*		
Pengembangan Model Ensemble Stacking untuk Klasifikasi Serangan Distributed Denial of Service	*Multiclass (12 class)*	CICDDoS2019	Model Individual	Support Vector Machine	*accuracy, precision, recall, f1-score, confusion matrix,* ROC-AUC, *inference time*	Evaluasi sudah menerapkam klasifikasi *multiclass* (12 kelas); pendekatan *ensemble* telah menerapkan algoritma heterogen; metrik evaluasi mencakup *accuracy, precision, recall, f1-score, confusion matrix*, ROC-AUC, dan *inference time.*
				Decision Tree		
				Naive Bayes		
				Logistic Regression		
				KNN		
			Model Bagging	Random Forest		
			Model Boosting	AdaBoost		
				XGBoost		
			Model Stacking	*Base-learners:* · Random Forest, · XGBoost, · AdaBoost, · Decision Tree *Meta-learners:* · Logistic Regression		

Berdasarkan Tabel 2.2.1, sebagian besar studi klasifikasi pengembangan model untuk klasifikasi serangan DDoS masih berfokus pada klasifikasi biner serta pemanfaatan pendekatan *ensemble* dengan algoritma *machine learning* yang bersifat homogen. Studi yang telah menerapkan pendekatan *ensemble stacking* pun pada umumnya tetap berada pada skenario biner dengan cakupan metrik evaluasi yang belum komprehensif, khususnya karena belum memasukkan ROC-AUC, *confusion matrix*, serta *inference time*. Kesenjangan utama yang teridentifikasi ialah belum adanya penerapan *ensemble stacking* dengan algoritma *machine learning* yang bersifat heterogen untuk klasifikasi *multiclass* dengan jumlah kelas yang kaya pada dataset CICDDoS2019 disertai evaluasi yang benar-benar menyeluruh. Penelitian ini, Pengembangan Model *Ensemble Stacking* untuk Klasifikasi Serangan Distributed Denial of Service, telah berhasil mengisi gap yang ditemukan pada penelitian-penelitian sebelumnya melalui pengembangan model berbasis *ensemble stacking* dengan algoritma *machine learning* yang bersifat heterogen untuk klasifikasi *multiclass* (12 kelas) menggunakan dataset CICDDoS2019. Model yang dikembangkan mengombinasikan beberapa *base learners*, yaitu Random Forest, XGBoost, AdaBoost, dan Decision Tree, dengan Logistic Regression sebagai *meta-learner*. Evaluasi yang dilakukan juga telah mencakup metrik yang komprehensif, meliputi *accuracy, precision, recall, f1-score*, ROC-AUC, *confusion matrix*, serta *inference time* sehingga mampu menilai kinerja sekaligus efisiensi komputasi model secara menyeluruh.

BAB III
METODOLOGI PENELITIAN

Rangkaian tahapan yang digunakan dalam pengembangan model *ensemble learning* untuk mengidentifikasi serangan DDoS pada kasus klasifikasi *multiclass* dapat dilihat pada Gambar 2.2.1 Tahapan ini mencakup pengumpulan data, analisis data eksploratif, prepemrosesan data, penyesuaian *hyperparameter*, pelatihan model, dan evaluasi model.



Gambar 2.2.1 Metodologi Penelitian

3.1 Pengumpulan Data

Data yang digunakan dalam penelitian ini adalah dataset CIC-DDoS2019, yang diperoleh dari situs resmi University of New Brunswick, melalui Canadian Institute for Cybersecurity (CIC) ("DDoS 2019 | Datasets | Research | Canadian Institute for Cybersecurity | UNB"). Dataset CIC-DDoS2019 dipilih karena memuat berbagai jenis serangan DDoS. Hal ini sesuai dengan tujuan penelitian, yaitu mengidentifikasi dan mengklasifikasikan serangan DDoS dalam skenario klasifikasi *multiclass*.

Data yang digunakan untuk pelatihan model berasal dari *training day*, yang mencakup satu jenis lalu lintas jaringan normal (*benign*) serta 12 jenis serangan DDoS. Adapun jenis serangan DDoS yang terdapat dalam bagian training day, antara lain DrDoS_LDAP, DrDoS_NTP, DrDoS_NetBIOS, DrDoS_MSSQL, DrDoS_DNS, DrDoS_SNMP, DrDoS_SSDP, DrDoS_UDP, UDP-lag, TFTP, SYN, dan WebDDoS. Namun, serangan WebDDoS tidak disertakan dalam proses pelatihan model karena volume lalu lintasnya yang sangat rendah dibandingkan dengan jenis serangan lainnya sehingga dianggap kurang representatif untuk model.

Sebagai langkah awal, dilakukan *random sampling* sebanyak 50.000 baris data secara acak dari bagian *training day*. *Sampling* ini dilakukan untuk mengurangi beban komputasi pada tahap eksplorasi awal, tanpa menghilangkan keragaman dari jenis serangan yang ada dalam data mentah.

3.2 Analisis Data Eksploratif

Analisis eksploratif terhadap data dilakukan untuk memperoleh pemahaman awal mengenai karakteristik dataset CIC-DDoS2019, termasuk pola distribusi kelas, serta potensi permasalahan seperti *missing value*, data tidak terdefinisi, dan data yang terduplikasi. Dataset ini terdiri dari satu kelas lalu lintas normal (*benign*) serta 11 jenis serangan DDoS, yaitu DrDoS_LDAP, DrDoS_NTP, DrDoS_NetBIOS, DrDoS_MSSQL, DrDoS_DNS, DrDoS_SNMP, DrDoS_SSDP, DrDoS_UDP, UDP-lag, TFTP, dan SYN.

Selain itu, dilakukan analisis statistik deskriptif dan penilaian kualitas data, yang menunjukkan adanya sejumlah *missing value*, nilai tak terdefinisi seperti *infinite* atau -*infinite*, dan baris data yang terduplikasi. Temuan ini menjadi dasar dalam perencanaan tahapan prepemrosesan, yang bertujuan untuk memastikan data yang digunakan dalam pelatihan model berada dalam kondisi yang bersih, konsisten, dan layak diproses lebih lanjut.

3.3 Prepemrosesan Data

Pre-pemrosesan data merupakan tahap krusial dalam *machine learning* yang bertujuan untuk memastikan kualitas data (Munagala dkk., 2024). Tahapan ini bertujuan untuk membersihkan, merapikan, dan mempersiapkan data agar sesuai dengan kebutuhan model *machine learning*. Tahapan prepemrosesan data mencakup pembersihan data, transformasi label, pembagian data, penskalaan data, dan seleksi fitur.

3.3.1 Pembersihan Data

Tahapan pembersihan data atau *data cleaning* mencakup penghapusan *missing value*, data yang terduplikasi, nilai tak terdefinisi seperti *infinite* atau -*infinite*, transformasi label, serta penghapusan fitur yang tidak relevan. Tahapan ini bertujuan untuk memastikan bahwa data yang digunakan dalam pelatihan model bersih, valid, dan tidak mengandung nilai yang dapat menyebabkan bias atau kesalahan dalam pemrosesan.

Dilakukan proses *random sampling* ulang sebanyak 30.000 *instance* per kelas guna menyeimbangkan distribusi data antar kelas. Proses ini menghasilkan total 360.000 data, yang terdiri 11 jenis serangan DDoS dan 1 kelas *benign* (normal). Penentuan jumlah ini mempertimbangkan keterbatasan sumber daya komputasi, sehingga diperlukan kompromi antara ukuran data dan efisiensi proses pelatihan model.

Beberapa fitur seperti Flow ID, Source IP, Destination IP, Source Port, Destination Port, Timestamp dihapus karena model diharapkan mengenali pola lalu lintas jaringan normal atau serangan, terlepas dari asal dan tujuan serangan. Pada penelitian yang dilakukan oleh (Moustafa, 2021) penghapusan fitur alamat IP dan *port* disarankan untuk mencegah model bergantung pada informasi yang bersifat spesifik, serta agar model dapat mengenali pola serangan secara lebih umum dan menyeluruh. Selain itu, fitur 'Fwd Header Length.1' juga dihapus karena setelah dilakukan pemeriksaan, fitur tersebut memiliki nilai yang identik dengan fitur 'Fwd Header Length' sehingga dianggap redundan.

3.3.2 Transformasi Label

Label target yang semula berupa data kategorikal dalam bentuk teks diubah menjadi representasi numerik menggunakan metode *Label Encoding*. Metode ini dipilih karena lebih efisien dibandingkan *One-Hot Encoding*, terutama pada dataset dengan banyak kelas. *Label Encoding* menjaga dimensi dataset tetap kompak tanpa menambah kompleksitas spasial. Proses ini dilakukan menggunakan fungsi LabelEncoder dari pustaka `sklearn.preprocessing`.

3.3.3 Pembagian Data

Setelah melalui tahapan pembersihan data dan transformasi label, dataset dibagi menjadi dua *subset*, yaitu data latih dan data uji. Pembagian dilakukan menggunakan fungsi `train_test_split` dari pustaka Scikit-learn dengan rasio 7:3, yaitu 70% untuk data latih dan 30% untuk data uji.

Untuk memastikan bahwa distribusi label target tetap proporsional pada kedua *subset*, parameter `stratify=y` diterapkan. Penggunaan parameter `stratify=y` penting untuk menjaga representasi kelas yang seimbang, terutama pada dataset dengan distribusi kelas yang tidak merata, sehingga dapat meminimalkan bias selama proses pelatihan dan evaluasi model.

3.3.4 Penskalaan Data

Penskalaan atau *scaling* digunakan untuk menyamakan satuan data agar lebih umum sehingga jarak antar data menjadi lebih kecil atau sebanding (V. Sharma, 2022). Penskalaan data sering dilakukan agar hasil pemodelan prediktif tetap akurat, terutama jika variabel *input* memiliki ukuran atau skala yang berbeda-beda (Fan, Chen, Wang, Wang, & Huang, 2021). Proses penskalaan data hanya dilakukan pada fitur yang aslinya sudah berbentuk angka, bukan fitur yang berupa kategori. Normalisasi *MinMax* dan Standardisasi *StandardScaler* merupakan metode dalam penskalaan data yang digunakan dalam penelitian ini.

a. StandardScaler

   Metode *StandardScaler* mengubah setiap fitur agar memiliki nilai rata-rata (*mean*) 0 dan standar deviasi 1. Rumus dari *StandardScaler* dapat dilihat pada Persamaan 3.1.

$$x' = \frac{x - \mu}{\sigma} \quad (3.1)$$

   Di mana, μ adalah nilai rata-rata (*mean*) dan σ adalah standar deviasi.

b. MinMax Scaler

   Metode *MinMax Scaler* mengubah nilai dari setiap fitur ke dalam rentang 0 hingga 1. Rumus dari *MinMax Scaler* dapat dilihat pada Persamaan 3.2

   Di mana, x_{min} dan x_{max} merupakan nilai minimum dan maksimum dari variabel x.

$$x' = \frac{x - x_{min}}{x_{max} - x_{\min}} \quad (3.2)$$

Setelah proses penskalaan diterapkan, data yang telah ditransformasi dari masing-masing metode kemudian digunakan untuk melatih delapan algoritma *machine learning*, yaitu Naive Bayes, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Logistic Regression, Decision Tree, Random Forest, AdaBoost, dan XGBoost. Proses pelatihan ini dilanjutkan dengan evaluasi kinerja model menggunakan metrik *accuracy*, dengan tujuan untuk mengamati seberapa besar pengaruh masing-masing metode penskalaan terhadap hasil klasifikasi *multiclass*.

Dari hasil evaluasi tersebut, metode penskalaan yang menghasilkan rata-rata *accuracy* tertinggi di antara seluruh algoritma akan dipilih sebagai metode penskalaan utama. Metode inilah yang kemudian akan digunakan dalam tahapan selanjutnya, yakni seleksi fitur.

3.3.5 Seleksi Fitur

Seleksi fitur adalah proses memilih sebagian fitur dari semua fitur yang ada, dengan tujuan untuk membantu algoritma bekerja lebih baik, menghemat waktu dan memori komputer, serta menemukan fitur yang benar-benar berkaitan dengan masalah yang sedang diteliti (Bellotti, Nouretdinov, Yang, & Gammerman, 2014). Metode seleksi fitur yang digunakan dalam penelitian ini yaitu Pearson Correlation, Mutual Information, Recursive Feature Elimination (RFE), dan Least Absolute Shrinkage and Selection Operator with Cross Validation (LassoCV). Pemilihan keempat metode ini didasarkan pada hasil adaptasi dari sebuah studi (Das, Venugopal, Shiva, & Sheldon, 2020) yang membandingkan berbagai metode seleksi fitur dalam skenario klasifikasi berbasis *machine learning*. Berdasarkan studi (Das dkk., 2020), keempat metode ini menunjukkan performa yang optimal dalam meningkatkan akurasi model. Meskipun masih berada dalam domain deteksi serangan DDoS, studi tersebut menggunakan dataset yang berbeda dan hanya menerapkan klasifikasi biner. Sementara itu, pada penelitian ini mengadopsi pendekatan klasifikasi *multiclass* sehingga diperlukan evaluasi ulang terhadap efektivitas metode seleksi fitur dalam konteks dataset CIC-DDoS2019.

Setiap metode seleksi fitur diuji pada data yang telah diskalakan dengan metode terbaik dari tahap sebelumnya. Setelah fitur terseleksi, dilakukan pelatihan ulang terhadap delapan algoritma *machine learning*, yaitu Naive Bayes, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Decision Tree, Random Forest, Logistic Regression, AdaBoost, dan XGBoost. Evaluasi dilakukan kembali menggunakan metrik akurasi untuk membandingkan kinerja metode seleksi fitur dalam klasifikasi *multiclass*.

Metode seleksi fitur yang memberikan rata-rata *accuracy* terbaik di antara delapan algoritma tersebut akan dipilih sebagai metode utama yang digunakan pada tahap selanjutnya, yaitu penyesuaian *hyperparameter* dan pembangunan model akhir, yaitu model *ensemble learning*.

3.4 Penyesuaian Hyperparameter

Setelah diperoleh metode seleksi fitur terbaik melalui evaluasi pada tahap sebelumnya, langkah selanjutnya adalah melakukan penyesuaian *hyperparameter* (*hyperparameter tuning*) untuk mengoptimalkan performa model. Proses penyesuaian *hyperparameter* dilakukan menggunakan dataset yang telah diproses menggunakan metode penskalaan terbaik serta fitur-fitur yang dipilih oleh metode seleksi fitur terbaik.

Dalam penelitian ini, metode GridSearchCV digunakan untuk mengeksplorasi kombinasi *hyperparameter* secara sistematis. GridSearchCV bekerja dengan mengevaluasi semua kombinasi yang mungkin dari nilai *hyperparameter* yang telah ditentukan sehingga memberikan peluang untuk memperoleh hasil yang optimal (Ogunsanya, Isichei, & Desai, 2023). Metode GridSearchCV dikombinasikan dengan teknik *K-Fold cross-validation* untuk memastikan bahwa hasil evaluasi model bersifat stabil dan tidak bergantung pada *subset* data tertentu. *K-Fold cross-validation* adalah metode yang membagi dataset secara acak menjadi K bagian, di mana satu bagian digunakan untuk validasi dan sisanya digunakan untuk melatih model (Pachouly, Ahirrao, Kotecha, Selvachandran, & Abraham, 2022). Metode *K-Fold Cross-Validation* dapat menghasilkan prediksi yang lebih baik dengan mengoptimalkan pemilihan sampel dari dataset yang tersedia untuk pelatihan dan validasi (Pachouly dkk., 2022).

Pemilihan ruang pencarian (*search space*) untuk *hyperparameter* setiap algoritma dilakukan secara sistematis dengan mempertimbangkan praktik umum dalam studi literatur dan dokumentasi pustaka scikit-learn. Nilai-nilai yang digunakan bertujuan mencakup konfigurasi yang paling sering digunakan dalam klasifikasi berbasis *machine learning*, khususnya untuk data seperti CIC-DDoS2019. Proses *penyesuaian* dilakukan dengan metode *Grid Search* yang secara otomatis mengevaluasi seluruh kombinasi parameter pada setiap algoritma, dikombinasikan dengan metode *3-Fold cross-validation* untuk meningkatkan keandalan hasil. Tabel 3.4.1 merangkum ruang pencarian *hyperparameter* yang diuji untuk masing-masing algoritma *machine learning*.

Tabel 3.4.1 *Hyperparameter* yang Diuji

Model	Parameter yang Diuji
Decision Tree	criterion: ['gini', 'entropy', 'log_loss'], max_depth: [None, 5, 10, 20, 30], min_samples_split: [2, 5, 10], min_samples_leaf: [1, 2, 4], max_features: [None, 'sqrt', 'log2']
Support Vector Machine	kernel: ['linear', 'rbf', ‘poly’], C: [0.1, 0.5, 1], gamma: ['scale', 'auto']
Naive Bayes	var_smoothing: np.logspace(0, -9, num=100)

Logistic Regression	{
	solver: ['saga'],
	penalty: ['l1', 'l2', 'elasticnet'],
	C: [0.01, 0.1, 1, 10],
	l1_ratio: [0, 0.5, 1]
	},
	{
	solver: ['lbfgs'],
	penalty: ['l2'],
	C: [0.01, 0.1, 1, 10],
	},
	{
	solver: ['newton-cg'],
	penalty: ['l2'],
	C: [0.01, 0.1, 1, 10],
	}

KNN	n_neighbors: range (5,21,1), weights: ['uniform', 'distance'], metric: ['minkowski'], p: [1, 2]
Random Forest	max_depth: [None, 5, 10, 20, 30], n_estimators: [100, 200, 300], max_features: [None, 'sqrt', 'log2'], min_samples_leaf': [1, 2, 4]
AdaBoost	n_estimators: [100, 200, 300], learning_rate': [0.01, 0.1, 0.5, 1]
XGBoost	n_estimators: [100, 200,300], max_depth: [5, 10, 20], learning_rate: [0.01, 0.1], gamma: [0.1, 0.3], reg_alpha: [0.01, 0.1], reg_lambda: [1, 1.5, 2]

3.5 Algoritma Machine Learning

Dalam penelitian ini digunakan lima algoritma *machine learning* tunggal yang mewakili berbagai pendekatan pembelajaran, baik berbasis probabilistik, pohon keputusan, dan metode berbasis jarak. Algoritma *Machine Learning* yang digunakan dalam penelitian ini yaitu Decision Tree, Support Vector Machine (SVM), Naïve Bayes, Logistic Regression, dan K-Nearest Neighbors (KNN). Pemilihan algoritma didasarkan pada studi pustaka yang menunjukkan bahwa kelima algoritma tersebut sering digunakan dalam deteksi serangan siber, termasuk serangan *Distributed Denial of Service* (DDoS). Proses implementasi algoritma dilakukan menggunakan library Scikit-learn (`sklearn`). Scikit-learn (`sklearn`) merupakan salah satu library *machine learning* paling populer dan komprehensif dalam bahasa pemrograman Python. Library ini menyediakan berbagai algoritma *machine learning* untuk keperluan klasifikasi, regresi, klastering, reduksi dimensi, pemilihan fitur, serta evaluasi model.

3.6 Ensemble Learning

Dalam penelitian ini, pendekatan *ensemble learning* digunakan untuk meningkatkan performa klasifikasi *multiclass* dengan menggabungkan kekuatan dari beberapa algoritma *machine learning*. Dalam penelitian ini, diterapkan empat metode *ensemble*, yaitu Random Forest, AdaBoost, XGBoost, dan *Stacking*.

a. Random Forest (RF)

   Random Forest dipilih karena memiliki sejumlah keunggulan, antara lain: (i) *Overfitting* dapat dikurangi sehingga akurasi pada pohon keputusan (*decision tree*) meningkat, (ii) Dapat digunakan untuk permasalahan klasifikasi maupun regresi, dan (iii) Mampu menangani nilai-nilai yang hilang dalam data secara otomatis. (Rani dkk., 2023).

b. Adaptive Boosting (AdaBoost)

   Adaboost dipilih karena memiliki sejumlah keunggulan, antara lain: (i) Memiliki kecenderungan menghasilkan kesalahan generalisasi yang lebih rendah, (ii) Sederhana dan mudah diimplementasikan, dan (iii) Kompatibel dengan berbagai jenis *classifier* (Rani dkk., 2023).

c. XGBoost

   XGBoost merupakan pendekatan *gradient boosting* yang membangun pohon keputusan (*decision tree*) secara berurutan, di mana setiap pohon dibuat untuk memperbaiki

kesalahan dari pohon sebelumnya (Butt dkk., 2024). XGBoost dipilih karena memiliki sejumlah keunggulan, antara lain: (i) Memiliki fleksibilitas yang tinggi, (ii) Mendukung pemrosesan paralel (*multiprocessing*), (iii) Memungkinkan penerapan regularisasi, dan (iv) Memberikan kontrol yang lebih baik t erhadap data yang hilang. (Rani dkk., 2023).

d. Stacking

Dalam penelitian ini, metode *stacking* digunakan sebagai pendekatan *ensemble learning* yang menggabungkan beberapa model dasar (*base* learner) untuk meningkatkan akurasi dan ketahanan prediksi. *Output* dari masing-masing model dasar (*base learner*) dijadikan *input* bagi *meta-model*, yang dilatih untuk mengombinasikan prediksi tersebut menjadi hasil akhir yang lebih optimal (Arya, Kumar, & Ahmad, 2023). Dalam penelitian ini, proses *stacking* dilakukan dengan mengevaluasi beberapa kombinasi model, termasuk kombinasi dari model *machine learning* individual dan juga model *ensemble* seperti XGBoost, AdaBoost, dan Random Forest. Pemilihan kombinasi model *stacking* sebagian diadaptasi dari referensi studi terdahulu, meskipun studi-studi tersebut menggunakan dataset berbeda dan menerapkan klasifikasi biner. Oleh karena itu, dalam penelitian ini dilakukan penyesuaian dan eksplorasi ulang untuk menguji efektivitas kombinasi *stacking* dalam konteks klasifikasi *multiclass* terhadap 12 jenis kelas menggunakan dataset CIC-DDoS2019.

3.7 Evaluasi Model

Evaluasi model dilakukan untuk mengukur kinerja dari setiap model *machine learning* yang digunakan, baik model individual maupun model *ensemble*, dalam melakukan klasifikasi *multiclass* serangan *Distributed Denial of Service* (DDoS). Evaluasi dilakukan pada data uji (*test set*) yang diperoleh dari proses pembagian data pada tahap prepemrosesan data, dengan proporsi 70% data latih dan 30% data uji.

Sebelum dilakukan evaluasi, setiap model *machine learning* harus terlebih dahulu dilatih (*training*) menggunakan data latih. Proses pelatihan dilakukan menggunakan metode `.fit()` dari library Scikit-learn. Setelah model selesai dilatih, model tersebut digunakan untuk melakukan prediksi terhadap data uji dengan menggunakan fungsi `.predict()`.

Hasil prediksi kemudian dibandingkan dengan label sebenarnya pada data uji untuk menghitung metrik evaluasi performa model. Evaluasi dilakukan menggunakan yaitu metriks

accuracy, precision, recall, dan F1-score. Perhitungan metrik tersebut dilakukan dengan fungsi-fungsi dari modul `sklearn.metrics`, menggunakan pendekatan rata-rata *macro* (`average='macro'`) agar setiap kelas memiliki bobot yang seimbang, tanpa terpengaruh oleh jumlah data pada masing-masing kelas.

Evaluasi juga mencakup dua aspek tambahan untuk memberikan gambaran yang lebih menyeluruh terhadap performa model, yaitu *inference time* dan ROC-AUC. *Inference time* digunakan untuk mengukur efisiensi model dalam melakukan prediksi terhadap data baru. Pengukuran dilakukan dengan menghitung rata-rata waktu yang dibutuhkan oleh model untuk memprediksi sebanyak 100 sampel data dari data uji. Proses ini dilakukan menggunakan fungsi `time()` dari modul `time`, dan hasilnya disajikan dalam satuan milidetik (*ms*).

Sementara itu, ROC AUC (*Receiver Operating Characteristic - Area Under Curve*) digunakan untuk menilai kemampuan model dalam membedakan antar kelas secara menyeluruh. Penghitungan ROC AUC dilakukan menggunakan pendekatan One-vs-Rest (OvR), dengan memanfaatkan fungsi `roc_auc_score()` dari `sklearn.metrics`, serta parameter `multi_class='ovr'` dan `average='macro'`. Evaluasi ROC AUC hanya dilakukan pada delapan algoritma utama serta satu model *stacking ensemble* terbaik yang diperoleh dari tahap sebelumnya.

3.8 Lingkungan Eksperimen

Seluruh proses eksperimen dalam penelitian ini—mulai dari pengumpulan data, prepemrosesan data, penyesuaian *hyperparameter*, pelatihan model, hingga evaluasi performa model—dilakukan menggunakan perangkat keras dan perangkat lunak tertentu untuk menjamin *reproducibility* dan konsistensi hasil.

Eksperimen dalam penelitian ini dilakukan sepenuhnya secara lokal menggunakan bahasa pemrograman Python. Pengembangan dilakukan di dalam Visual Studio Code (VS Code) yang dilengkapi dengan ekstensi Jupyter, sehingga memungkinkan eksekusi notebook secara interaktif di dalam editor.

3.8.1 Spesifikasi Perangkat Keras (Hardware)

Eksperimen dijalankan pada perangkat pribadi dengan spesifikasi sebagai berikut:

- Prosesor (CPU): Intel® Core™ i7-13700H @ 2.40GHz (14 core / 20 thread)

- RAM: 16 GB DDR5
- Penyimpanan: SSD 1 TB
- GPU: Tidak digunakan (semua proses berjalan pada CPU)
- Sistem Operasi: Windows 11 Home Single Language 64-bit

3.8.2 Lingkungan Pemrograman

Pengembangan dan eksekusi kode dilakukan menggunakan:

- Bahasa Pemrograman: Python 3.12.4
- IDE: Visual Studio Code (VS Code)
- Extension: Jupyter
- Mode Eksekusi: Notebook .ipynb dijalankan langsung melalui Jupyter kernel di VS Code
- Manajer Paket: pip (Python Package Installer)

3.8.3 Library yang Digunakan

Berikut adalah daftar pustaka utama yang digunakan dalam eksperimen beserta versinya:

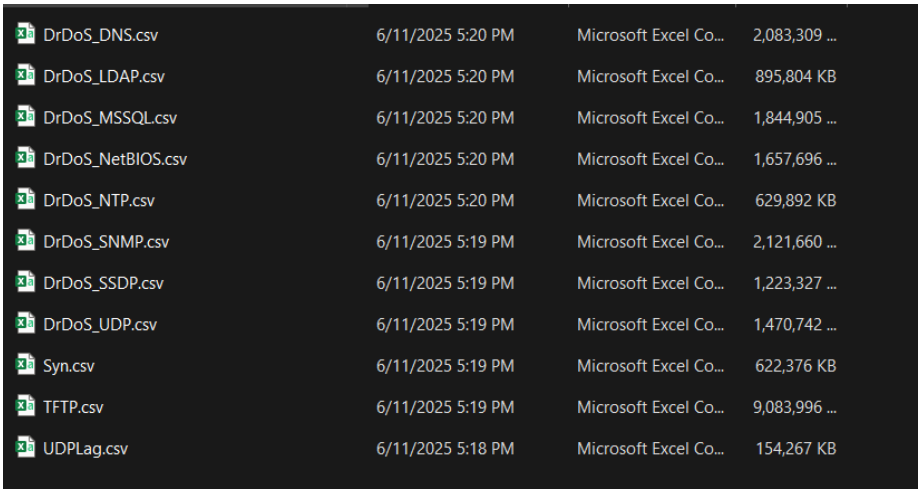
Library / Package	Versi
numpy	2.0.1
scikit-learn	1.6.1
matplotlib	3.10.1
pandas	2.2.2
seaborn	0.13.2
joblib	1.4.2
xgboost	3.0.2

BAB IV
HASIL DAN PEMBAHASAN

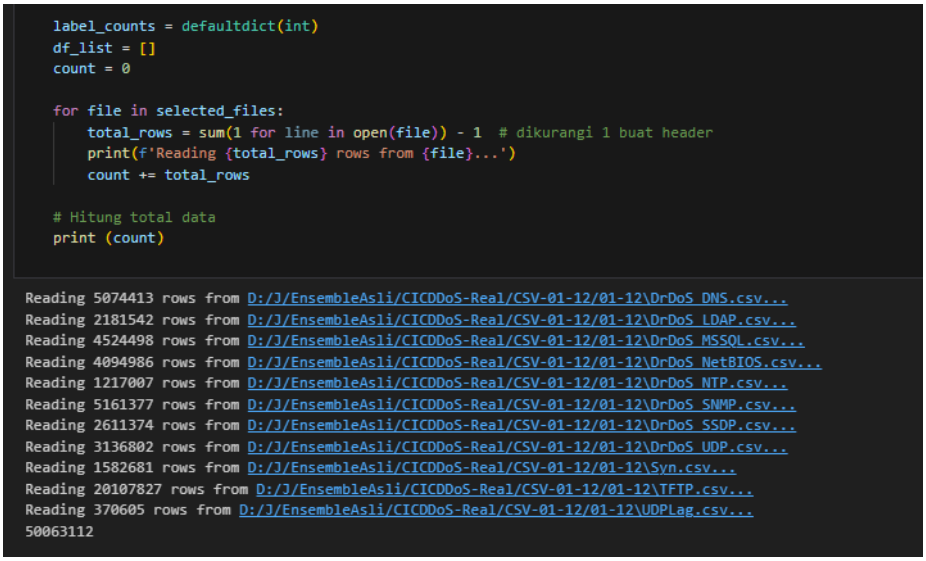
4.1 Pengumpulan Data

Dataset yang digunakan dalam penelitian ini adalah CIC-DDoS2019, yang diperoleh dari situs resmi Canadian Institute for Cybersecurity (CIC), University of New Brunswick. Dataset CIC-DDoS2019 dipilih karena menyediakan lalu lintas jaringan yang mencerminkan skenario dunia nyata serta mencakup berbagai jenis serangan DDoS. Ketersediaan beragam tipe serangan membuat dataset ini sangat relevan untuk skenario klasifikasi *multiclass*, yang menjadi fokus utama dalam penelitian ini.

Penelitian ini menggunakan data dari sesi *training day* yang terdiri atas 11 *file* CSV. Masing-masing *file* merepresentasikan lalu lintas jaringan dari satu atau dua jenis serangan dan lalu lintas normal (*benign*). Total data mentah dari seluruh *file* mencapai 50.063.112 baris. Rincian *file* dan jumlah baris data setiap *file*-nya ditampilkan pada Gambar 4.1.1 dan Gambar 4.1.2.



Gambar 4.1.1 Rincian *File Training Day* CIC-DDoS2019



```
label_counts = defaultdict(int)
df_list = []
count = 0

for file in selected_files:
    total_rows = sum(1 for line in open(file)) - 1  # dikurangi 1 buat header
    print(f'Reading {total_rows} rows from {file}...')
    count += total_rows

# Hitung total data
print (count)
```

```
Reading 5074413 rows from D:/J/EnsembleAsli/CICDDoS-Real/CSV-01-12/01-12\DrDoS_DNS.csv...
Reading 2181542 rows from D:/J/EnsembleAsli/CICDDoS-Real/CSV-01-12/01-12\DrDoS_LDAP.csv...
Reading 4524498 rows from D:/J/EnsembleAsli/CICDDoS-Real/CSV-01-12/01-12\DrDoS_MSSQL.csv...
Reading 4094986 rows from D:/J/EnsembleAsli/CICDDoS-Real/CSV-01-12/01-12\DrDoS_NetBIOS.csv...
Reading 1217007 rows from D:/J/EnsembleAsli/CICDDoS-Real/CSV-01-12/01-12\DrDoS_NTP.csv...
Reading 5161377 rows from D:/J/EnsembleAsli/CICDDoS-Real/CSV-01-12/01-12\DrDoS_SNMP.csv...
Reading 2611374 rows from D:/J/EnsembleAsli/CICDDoS-Real/CSV-01-12/01-12\DrDoS_SSDP.csv...
Reading 3136802 rows from D:/J/EnsembleAsli/CICDDoS-Real/CSV-01-12/01-12\DrDoS_UDP.csv...
Reading 1582681 rows from D:/J/EnsembleAsli/CICDDoS-Real/CSV-01-12/01-12\Syn.csv...
Reading 20107827 rows from D:/J/EnsembleAsli/CICDDoS-Real/CSV-01-12/01-12\TFTP.csv...
Reading 370605 rows from D:/J/EnsembleAsli/CICDDoS-Real/CSV-01-12/01-12\UDPLag.csv...
50063112
```

Gambar 4.1.2 Distribusi Jumlah Baris Data pada Setiap *File Training Day* Dataset CIC-DDoS2019

Untuk menghindari beban komputasi yang terlalu besar pada tahap eksplorasi awal dan pelatihan model, dilakukan *random sampling* sebanyak 50.000 data dari setiap kelas, termasuk kelas *benign*. Hasil distribusi kelas setelah *random sampling* disajikan pada Tabel 4.1.1.

Tabel 4.1.1 Distribusi Kelas Setelah *Random Sampling*

Label	Jumlah Data
BENIGN	50.000
DrDoS_DNS	50.000
DrDoS_LDAP	50.000
DrDoS_MSSQL	50.000
DrDoS_NTP	50.000
DrDoS_NetBIOS	50.000
DrDoS_SNMP	50.000
DrDoS_SSDP	50.000
DrDoS_UDP	50.000
SYN	50.000
TFTP	50.000
UDP-lag	50.000
WebDDoS	439

Berdasarkan hasil *sampling* yang dilakukan, diketahui bahwa jumlah data untuk serangan WebDDoS hanya sebanyak 439 data. Jumlah ini tergolong sangat kecil dibandingkan dengan jenis serangan lainnya yang memiliki volume data jauh lebih besar. Oleh karena itu, serangan

WebDDoS tidak disertakan dalam proses pelatihan model, karena volume yang terlalu rendah dikhawatirkan tidak cukup representatif dan dapat menyebabkan ketidakseimbangan kelas yang signifikan dalam proses klasifikasi *multiclass*. Total data yang digunakan dalam penelitian ini adalah 600.000 data, terdiri dari 12 kelas (11 jenis serangan dan 1 kelas *benign*) dengan distribusi seimbang.

4.2 Analisis Data Eksploratif

Analisis Data Eksploratif bertujuan untuk memahami struktur, kualitas, dan pola dalam data sebelum dilakukan prepemrosesan data dan pelatihan model. Struktur dataset diperiksa menggunakan fungsi `df.info()` untuk mengetahui jumlah entri, jumlah fitur, tipe data, serta jumlah nilai *non-null* di setiap kolom. Seperti ditunjukkan pada Gambar 4.2.1, dataset terdiri dari 600.000 baris dan 88 kolom (fitur), dengan mayoritas data bertipe numerik (`float64` sebanyak 45 fitur, `int64` sebanyak 37 fitur), dan sisanya `object` sebanyak 6 fitur.

```
#info dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 600000 entries, 0 to 599999
Data columns (total 88 columns):
 #   Column                       Non-Null Count   Dtype
---  ------                       --------------   -----
 0   Unnamed: 0                   600000 non-null  int64
 1   Flow ID                      600000 non-null  object
 2    Source IP                   600000 non-null  object
 3    Source Port                 600000 non-null  int64
 4    Destination IP              600000 non-null  object
 5    Destination Port            600000 non-null  int64
 6    Protocol                    600000 non-null  int64
 7    Timestamp                   600000 non-null  object
 8    Flow Duration               600000 non-null  int64
 9    Total Fwd Packets           600000 non-null  int64
 10   Total Backward Packets      600000 non-null  int64
 11  Total Length of Fwd Packets  600000 non-null  float64
 12   Total Length of Bwd Packets 600000 non-null  float64
 13   Fwd Packet Length Max       600000 non-null  float64
 14   Fwd Packet Length Min       600000 non-null  float64
 15   Fwd Packet Length Mean      600000 non-null  float64
 16   Fwd Packet Length Std       600000 non-null  float64
 17  Bwd Packet Length Max        600000 non-null  float64
 18   Bwd Packet Length Min       600000 non-null  float64
 19   Bwd Packet Length Mean      600000 non-null  float64
...
 86   Inbound                     600000 non-null  int64
 87   Label                       600000 non-null  object
dtypes: float64(45), int64(37), object(6)
memory usage: 407.4+ MB
```

Gambar 4.2.1 Struktur dan Tipe Data

Statistik deskriptif digunakan untuk memahami karakteristik nilai pada masing-masing fitur numerik, seperti nilai minimum, maksimum, rata-rata (*mean*), simpangan baku (*standard*

deviation), dan distribusi kuartil (25%, 50%, dan 75%). Analisis ini dilakukan menggunakan fungsi `df.describe()`, sebagaimana ditunjukkan pada Gambar 4.2.2

```
df.describe()
```

```
c:\Users\Lenovo\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\core\nanops.py:1016: RuntimeWarning: invalid value encountered in subtract
  sqr = _ensure_numeric((avg - values) ** 2)
c:\Users\Lenovo\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\core\nanops.py:1016: RuntimeWarning: invalid value encountered in subtract
  sqr = _ensure_numeric((avg - values) ** 2)
```

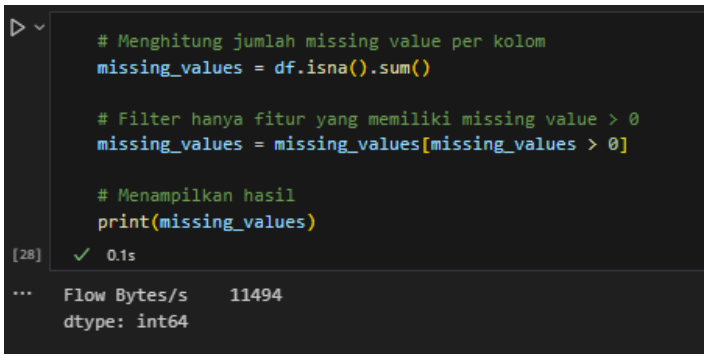
	Unnamed: 0	Source Port	Destination Port	Protocol	Flow Duration	Total Fwd Packets	Total Backward Packets	Total Length of Fwd Packets	Total Length of Bwd Packets	Fwd Packet Length Max	...	min_seg_size_forward
count	600000.000000	600000.000000	600000.000000	600000.000000	6.000000e+05	600000.000000	600000.000000	6.000000e+05	6.000000e+05	600000.000000	...	6.000000e+05
mean	91492.988912	26080.425003	30931.906478	14.686845	2.036414e+06	9.065573	0.584228	3.676246e+03	4.440170e+02	581.953550	...	-3.636198e+07
std	135760.746167	25366.801737	20118.645089	4.511784	1.227610e+07	288.394114	12.843996	2.640125e+04	3.639919e+04	543.046476	...	1.931956e+08
min	0.000000	0.000000	0.000000	0.000000	0.000000e+00	1.000000	0.000000	0.000000e+00	0.000000e+00	0.000000	...	-1.408238e+09
25%	12017.000000	772.000000	13046.000000	17.000000	1.000000e+00	2.000000	0.000000	4.580000e+02	0.000000e+00	229.000000	...	0.000000e+00
50%	38147.500000	21359.000000	30940.000000	17.000000	2.000000e+00	2.000000	0.000000	1.048000e+03	0.000000e+00	414.000000	...	2.000000e+01
75%	106389.000000	52110.000000	48690.000000	17.000000	1.292000e+03	4.000000	0.000000	2.944000e+03	0.000000e+00	1025.000000	...	2.000000e+01
max	805189.000000	65534.000000	65535.000000	17.000000	1.200000e+08	99588.000000	4602.000000	1.526642e+07	1.099376e+07	32120.000000	...	1.480000e+03

8 rows × 82 columns

Gambar 4.2.2 Statistik Deskriptif pada Beberapa Fitur Dataset

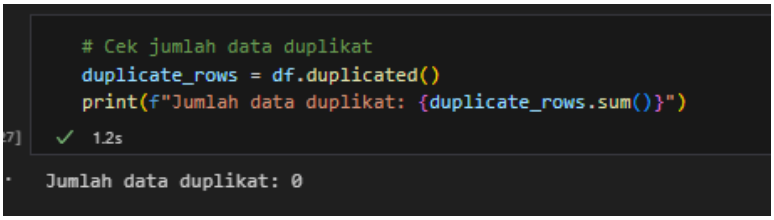
Beberapa fitur memiliki nilai maksimum yang sangat besar, seperti *Flow Duration* yang mencapai nilai 1.2e+08, mengindikasikan keberadaan *outlier* yang ekstrem. Selain itu, terdapat fitur seperti *min_seg_size_forward* dengan nilai minimum negatif -1.4e+09. Adanya *RuntimeWarning* selama proses `df.describe()` memperkuat indikasi bahwa masih terdapat nilai tidak valid yang perlu dibersihkan lebih lanjut sebelum proses pemodelan.

Pemeriksaan nilai hilang dilakukan dengan menggunakan fungsi `df.isna().sum()` untuk menghitung jumlah *missing value* pada setiap kolom. Hasil analisis ditunjukkan pada Gambar 4.2.3, di mana hanya satu fitur yang terdeteksi memiliki nilai hilang, yaitu Flow Bytes/s dengan jumlah 11.494 data yang terdapat *missing value*. Fitur ini akan diproses lebih lanjut pada tahap prepemrosesan untuk menentukan penanganan yang sesuai seperti penghapusan baris nilai yang mengandung *missing value*.



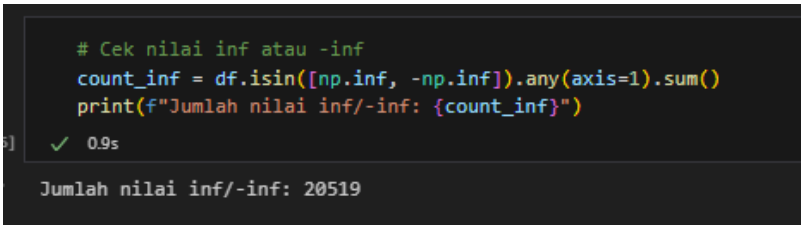
Gambar 4.2.3 Pemeriksaan *Missing Value* pada Dataset

Langkah selanjutnya adalah memeriksa adanya data yang terduplikasi menggunakan `df.duplicated()`. Hasilnya dapat dilihat pada Gambar 4.2.4, tidak terdapat baris yang terduplikasi dalam dataset.



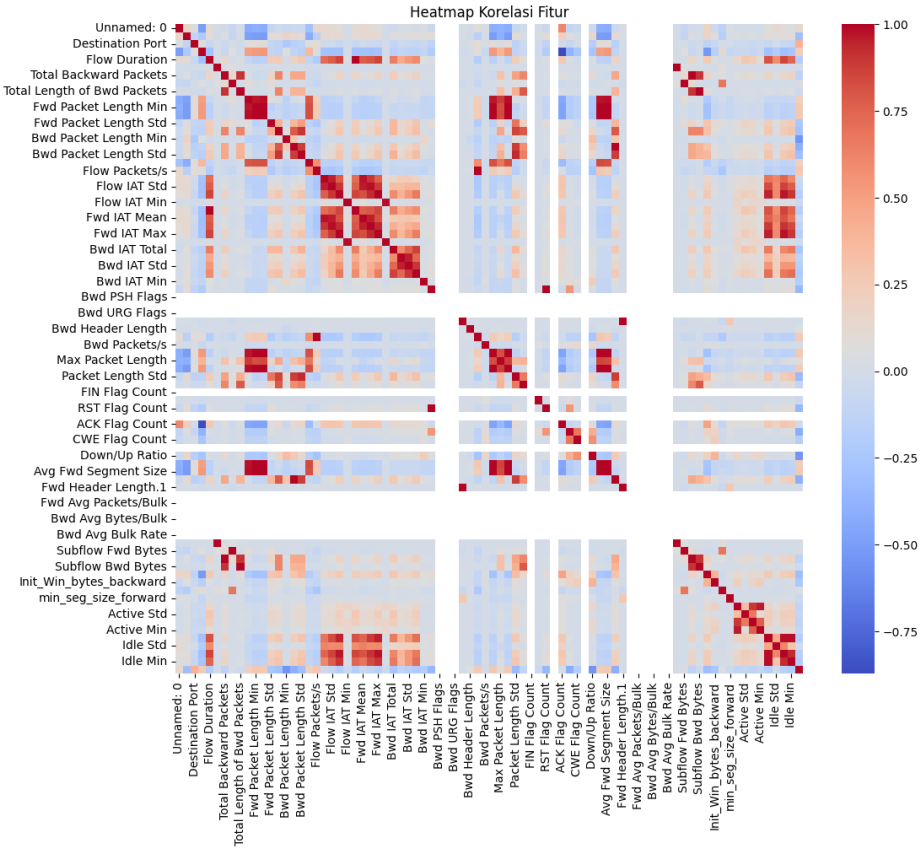
Gambar 4.2.4 Pemeriksaan Data Duplikat

Nilai tak terhingga (*infinity*) dapat menyebabkan kesalahan selama pelatihan model. Pemeriksaan dilakukan dengan `df.isin([np.inf, -np.inf])`. Hasilnya disajikan pada Gambar 4.2.5, yang menunjukkan bahwa terdapat 20.519 baris yang mengandung nilai *inf* atau -*inf*. Penanganan terhadap nilai ini akan dilakukan pada tahap prepemrosesan dengan cara menghapus baris data yang memiliki nilai tersebut.



Gambar 4.2.5 Jumlah Baris yang Mengandung Nilai *Inf* atau -*Inf*

Visualisasi korelasi antar fitur numerik dilakukan menggunakan heatmap korelasi untuk mengidentifikasi pasangan fitur dengan hubungan linier yang tinggi. Hasil visualisasi dapat dilihat pada Gambar 4.2.6, yang menunjukkan bahwa beberapa fitur memiliki korelasi sangat tinggi (lebih dari 0,75). Namun, pada tahap analisis data ekploratif ini belum dilakukan penghapusan fitur yang berkorelasi tinggi secara manual, karena tahap seleksi fitur akan dilakukan secara sistematis menggunakan berbagai metode pada tahap prepemrosesan data. *Heatmap* ini tetap memberikan gambaran awal mengenai redundansi antar fitur.



Gambar 4.2.6 Korelasi antar Fitur

4.3 Prepemrosesan Data

4.3.1 Data Cleaning

Proses pembersihan data dilakukan secara menyeluruh terhadap data hasil *sampling* awal sebanyak 600.000 *instance*, yang terdiri dari 50.000 *instance* untuk setiap kelas, meliputi 11 jenis serangan DDoS dan 1 kelas normal (*benign*). Tahapan pembersihan data mencakup penghapusan nilai hilang (*missing values*) dan nilai tidak terdefinisi, seperti *infinite* dan negatif *infinite*, yang

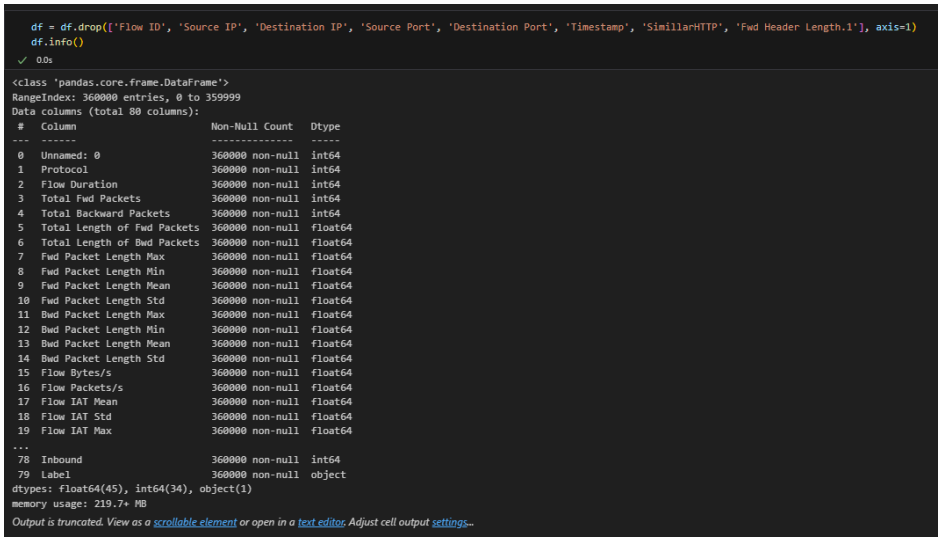
keduanya telah teridentifikasi dan dianalisis sebelumnya pada tahap analisis data eksploratif. Seluruh baris yang mengandung *missing value* maupun nilai tak terdefinisi dihapus guna menjaga integritas data dan mencegah gangguan pada proses penskalaan serta perhitungan jarak antar *instance* dalam tahapan pemodelan selanjutnya.

Setelah proses penghapusan *missing value* dan nilai tak terdefinisi, dilakukan proses *random sampling* ulang sebanyak 30.000 data untuk setiap kelasnya agar distribusi data tetap seimbang. Hasil *random sampling* untuk 30.000 baris data setiap kelasnya disajikan dalam Tabel 4.3.1.

Tabel 4.3.1 Distribusi Kelas Setelah *Random Sampling* Ulang

Label	Jumlah Data
BENIGN	30.000
DrDoS_DNS	30.000
DrDoS_LDAP	30.000
DrDoS_MSSQL	30.000
DrDoS_NTP	30.000
DrDoS_NetBIOS	30.000
DrDoS_SNMP	30.000
DrDoS_SSDP	30.000
DrDoS_UDP	30.000
SYN	30.000
TFTP	30.000
UDP-lag	30.000

Dilakukan penghapus fitur yang dianggap tidak relevan, seperti Flow ID, Source IP, Destination IP, Source Port, Destination Port, Timestamp, dan Fwd Header Length.1. Fitur-fitur tersebut dihapus agar model dapat mengenali pola serangan secara lebih umum dan menyeluruh. Proses penghapusan fitur dapat dilihat pada Gambar 4.3.1.



Gambar 4.3.1 Proses Penghapusan Fitur

4.3.2 Transformasi Label

Dilakukan transformasi label target dari format teks menjadi numerik dengan menggunakan *Label Encoding*. Metode *Label Encoding* dipilih karena efisien dibandingkan metode *One-Hot Encoding* yang akan meningkatkan dimensi dataset secara signifikan. Hasil transformasi label target dapat dilihat pada Tabel 4.3.2.

Tabel 4.3.2 Transformasi Label Target Menggunakan *Label Encoding*

Label	Hasil Enkode
BENIGN	0
DrDoS_DNS	1
DrDoS_LDAP	2
DrDoS_MSSQL	3
DrDoS_NTP	4
DrDoS_NetBIOS	5
DrDoS_SNMP	6
DrDoS_SSDP	7
DrDoS_UDP	8
SYN	9
TFTP	10
UDP-lag	11

4.3.3 Pembagian Data

Dataset kemudian dibagi menjadi data latih dan data uji menggunakan fungsi `train_test_split` dari pustaka Scikit-learn dengan rasio 7:3. Parameter `stratify=y`

digunakan agar distribusi kelas tetap proporsional di kedua *subset*. Proses ini penting untuk menghindari bias distribusi kelas dalam evaluasi model. Proses pembagian data menjadi data latih dan data uji dapat dilihat pada Gambar 4.3.2.

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

Gambar 4.3.2 Pembagian Data Latih dan Data Uji

4.3.4 Penskalaan Data

Terdapat dua metode penskalaan data yang digunakan dalam penelitian ini, yaitu StandardScaler dan MinMaxScaler. Penskalaan diterapkan hanya pada fitur numerik, sedangkan fitur kategorikal, yaitu Protocol dan Inbound, tidak termasuk dalam proses ini. Data hasil penskalaan dari kedua metode kemudian digunakan untuk melatih delapan algoritma *machine learning,* yaitu Gaussian Naïve Bayes (NB), K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Logistic Regression (LR), Decision Tree (DT), Random Forest (RF), Adaptive Boosting (AB), Extreme Gradient Boosting (XGB).

Evaluasi dilakukan dengan membandingkan *accuracy* dari delapan algoritma *machine learning* yang diuji pada tiga kondisi, yaitu tanpa penskalaan, dengan *MinMaxScaler*, dan dengan *StandardScaler*. Perbandingan metrik *accuracy* dari setiap algoritma dengan metode penskalaan yang berbeda dapat dilihat pada Tabel 4.3.3.

Tabel 4.3.3 Perbandingan Metode Penskalaan

Algoritma Machine Learning	***Accuracy***		
	Tanpa Penskalaan	**MinMaxScaler**	**StandardScaler**
Decision Tree	0.76	0.75	0.76
SVM	0.35	0.61	0.69
Naïve Bayes	0.19	0.46	0.46
Logistic Regression	0.42	0.66	0.67
KNN	0.76	0.76	0.76
Random Forest	0.77	0.76	0.76
AdaBoost	0.58	0.58	0.58
XGBoost	0.8	0.79	0.79

Berdasarkan hasil tersebut, dapat dilihat bahwa *StandardScaler* menunjukkan performa yang paling konsisten dan unggul pada sebagian besar algoritma. khususnya pada algoritma Support Vector Machine (SVM), Naïve Bayes (NB), dan Logistic Regression (LR), peningkatan

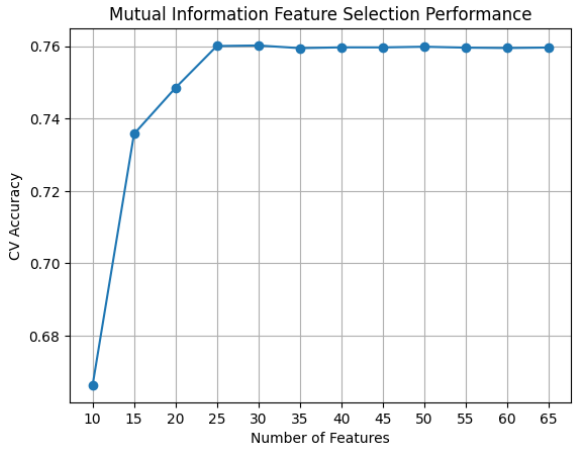
akurasi sangat signifikan saat menggunakan *StandardScaler* dibandingkan sebelum menggunakan metode penskalaan.

Sementara itu, *MinMaxScaler* juga memberikan peningkatan performa pada beberapa algoritma, tetapi tidak melebihi atau menyaingi konsistensi hasil dari metode *StandardScaler*. Oleh karena itu, *StandardScaler* dipilih sebagai metode penskalaan utama yang akan digunakan dalam tahap selanjutnya, yaitu proses seleksi fitur.

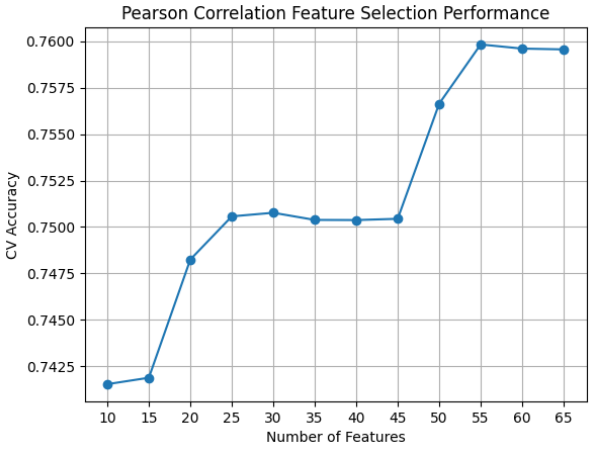
4.3.5 Seleksi Fitur

Empat metode seleksi fitur digunakan dalam penelitian ini, yaitu Mutual Information (MI), Pearson's Correlation, Recursive Feature Elimination (RFE), dan Least Absolute Shrinkage and Selection Operator with Cross Validation (LassoCV). Setiap metode seleksi fitur diterapkan pada data yang telah dinormalisasi dengan metode terbaik dari tahap sebelumnya, yaitu metode StandardScaler.

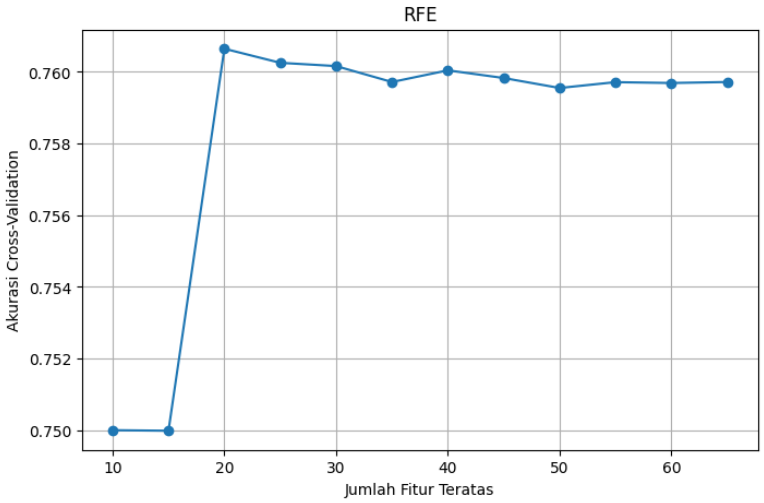
Untuk metode seleksi fitur MI, Pearson, dan RFE, proses seleksi fitur dilakukan melalui pendekatan eksperimen bertahap untuk menentukan jumlah fitur optimal. Setiap metode diuji menggunakan *3-fold cross-validation*, dengan rentang jumlah fitur yang dievaluasi mulai dari 10 hingga 65 fitur, dengan kenaikan sebesar 5 fitur per iterasi. Evaluasi dilakukan berdasarkan metrik akurasi, menggunakan model Random Forest sebagai evaluator. Nilai jumlah fitur optimal ditentukan berdasarkan jumlah fitur yang menghasilkan akurasi tertinggi pada proses *cross validation* tersebut. Gambar 4.3.3 merupakan proses menentukan jumlah fitur optimal dengan Mutual Information dengan *3-fold cross-validation.* Gambar 4.3.4 merupakan proses menentukan jumlah fitur optimal dengan Pearson's dengan *3-fold cross-validation.* Gambar 4.3.5 merupakan proses menentukan jumlah fitur RFE optimal dengan *3-fold cross-validation*.



Gambar 4.3.3 MI dengan *Cross Validation*

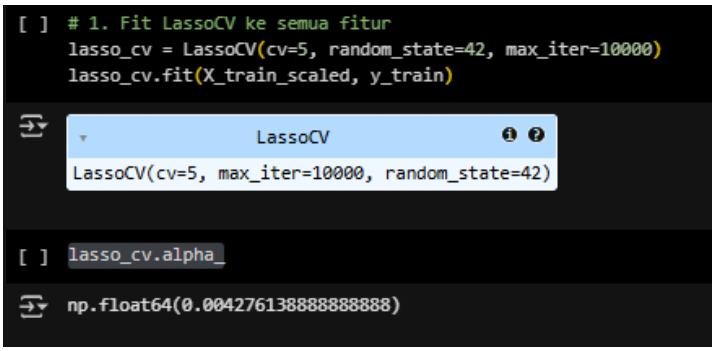


Gambar 4.3.4 Pearson's dengan *Cross Validation*



Gambar 4.3.5 RFE dengan *Cross Validation*

Sementara itu, untuk metode LassoCV, digunakan fungsi bawaan dari pustaka Scikit-learn yang secara otomatis melakukan penentuan parameter regularisasi terbaik melalui *cross-validation*. Setelah diperoleh nilai regularisasi optimal, model Lasso digunakan untuk menilai kontribusi setiap fitur melalui nilai koefisiennya. Fitur-fitur yang memiliki nilai koefisien nol dianggap tidak memberikan kontribusi terhadap target klasifikasi sehingga dieliminasi secara otomatis dari himpunan fitur yang akan digunakan dalam tahap pelatihan model. Nilai regularisasi optimal dari Lasso dapat dilihat pada Gambar 4.3.6.



Gambar 4.3.6 Nilai Regularisasi Optimal Lasso dengan *Cross Validation*

Setelah dilakukan proses seleksi fitur pada masing-masing metode, diperoleh jumlah fitur optimal berbeda-beda, tergantung pada karakteristik pemilihan fitur dari masing-masing pendekatan. Tabel 4.3.4 menyajikan jumlah fitur optimal yang diperoleh serta daftar nama fitur terpilih berdasarkan hasil terbaik dari setiap metode seleksi fitur.

Tabel 4.3.4 Daftar Nama Fitur Berdasarkan Hasil Seleksi Fitur dengan *Cross Validation*

Metode Seleksi Fitur	Jumlah Fitur	Nama Fitur
Mutual Information	25	['Init_Win_bytes_forward', 'Fwd IAT Std', 'Fwd Header Length', 'Fwd IAT Max', 'Fwd IAT Mean', 'Flow IAT Std', 'Fwd IAT Total', 'Unnamed: 0', 'Flow IAT Max', 'act_data_pkt_fwd', 'Flow Duration', 'Flow IAT Mean', 'Fwd Packets/s', 'Flow Packets/s', 'Min Packet Length', 'Flow Bytes/s', 'Fwd Packet Length Min', 'Subflow Fwd Bytes', 'Total Length of Fwd Packets', 'Avg Fwd Segment Size', 'Fwd Packet Length Max', 'Fwd Packet Length Mean', 'Average Packet Size', 'Max Packet Length', 'Packet Length Mean']
Pearson's Correlation	55	['ACK Flag Count', 'Inbound', 'Unnamed: 0', 'Max Packet Length', 'Fwd Packet Length Max', 'Packet Length Mean', 'Average Packet Size', 'Flow Bytes/s', 'Avg Fwd Segment Size', 'Fwd Packet Length Mean', 'Fwd Packet Length Min', 'Min Packet Length', 'URG Flag Count', 'Protocol', 'Down/Up Ratio', 'Bwd Packet Length Min', 'Flow IAT Mean', 'CWE Flag Count', 'Fwd IAT Mean', 'Avg Bwd Segment Size', 'Bwd Packet Length Mean', 'Flow IAT Std', 'Fwd PSH Flags', 'RST Flag Count', 'Fwd IAT Std', 'Packet Length Std', 'Init_Win_bytes_backward', 'Bwd Packet Length Std', 'Bwd Packet Length Max', 'Fwd Packet Length Std', 'Bwd IAT Total', 'Idle Std', 'act_data_pkt_fwd', 'Fwd IAT Max', 'Bwd IAT Max', 'Flow IAT Max', 'Total Backward Packets', 'Subflow Bwd Packets', 'Active Mean', 'Active Max', 'Packet Length Variance', 'Bwd IAT Min', 'Active Min', 'Fwd IAT Total', 'Idle Max', 'Bwd IAT Std', 'Flow Duration', 'Total

		Length of Fwd Packets', 'Subflow Fwd Bytes', 'min_seg_size_forward', 'Idle Mean', 'Bwd Header Length', 'Active Std', 'Bwd IAT Mean', 'SYN Flag Count']
Recursive Feature Elimination	20	['Unnamed: 0', 'Flow Duration', 'Total Length of Fwd Packets', 'Fwd Packet Length Max', 'Fwd Packet Length Min', 'Fwd Packet Length Mean', 'Flow Bytes/s', 'Flow IAT Max', 'Fwd IAT Total', 'Fwd IAT Mean', 'Fwd IAT Max', 'Fwd Header Length', 'Min Packet Length', 'Max Packet Length', 'Packet Length Mean', 'ACK Flag Count', 'Average Packet Size', 'Avg Fwd Segment Size', 'Subflow Fwd Bytes', 'Inbound']
LassoCV	44	['Unnamed: 0', 'Protocol', 'Flow Duration', 'Total Fwd Packets', 'Total Length of Fwd Packets', 'Fwd Packet Length Min', 'Fwd Packet Length Std', 'Bwd Packet Length Min', 'Bwd Packet Length Mean', 'Bwd Packet Length Std', 'Flow Bytes/s', 'Flow IAT Std', 'Fwd IAT Mean', 'Fwd IAT Std', 'Fwd IAT Max', 'Fwd IAT Min', 'Bwd IAT Total', 'Bwd IAT Mean', 'Bwd IAT Max', 'Bwd IAT Min', 'Fwd PSH Flags', 'Bwd Header Length', 'Bwd Packets/s', 'Packet Length Variance', 'SYN Flag Count', 'RST Flag Count', 'ACK Flag Count', 'URG Flag Count', 'CWE Flag Count', 'Down/Up Ratio', 'Average Packet Size', 'Avg Bwd Segment Size', 'Subflow Fwd Packets', 'Subflow Fwd Bytes', 'Init_Win_bytes_forward', 'Init_Win_bytes_backward', 'act_data_pkt_fwd', 'min_seg_size_forward', 'Active Std', 'Active Min', 'Idle Mean', 'Idle Std', 'Idle Max', 'Inbound']

Setelah diperoleh *subset* fitur optimal dari masing-masing metode seleksi, dilakukan evaluasi ulang terhadap kinerja delapan algoritma *machine learning* dengan menggunakan *subset* fitur tersebut. Perbandingan performa dilakukan berdasarkan metrik akurasi, untuk menentukan metode seleksi fitur yang paling efektif dalam meningkatkan kemampuan klasifikasi pada skenario *multiclass*. Nilai akurasi dari masing-masing metode seleksi fitur dan algoritma *machine learning* dapat dilihat pada Tabel 4.3.5.

Tabel 4.3.5 Perbandingan Akurasi Model pada Beberapa Metode Seleksi Fitur

Algoritma Machine Learning	***Accuracy***			
	MI	**Corr**	**RFE**	**LassoCV**
Decision Tree	0.75	0.75	0.75	0.75
SVM	0.71	0.68	0.73	0.67
Naïve Bayes	0.5	0.46	0.5	0.46
Logistic Regression	0.65	0.66	0.66	0.66
KNN	0.76	0.76	0.76	0.76
Random Forest	0.76	0.75	0.76	0.76
AdaBoost	0.63	0.61	0.51	0.58
XGBoost	0.79	0.78	0.79	0.79

Berdasarkan hasil pada Tabel 4.7 terlihat bahwa metode Mutual Information (MI) dan Recursive Feature Elimination (RFE) menghasilkan performa yang relatif serupa dan konsisten pada sebagian besar algoritma *machine learning*. Misalnya, pada algoritma Decision Tree (DT), K-Nearest Neighbors (KNN), Random Forest (RF), dan Extreme Gradient Boosting (XGB), akurasi yang diperoleh dari MI dan RFE bersifat identik atau hampir identik.

Pada algoritma Support Vector Machine (SVM), metode RFE mencatat akurasi tertinggi dibandingkan metode seleksi fitur lainnya, yaitu sebesar 0,73. Sementara itu, pada algoritma Adaptive Boosting (AB), metode MI memberikan akurasi tertinggi sebesar 0,63. Adapun pada algoritma Naïve Bayes (NB), baik MI maupun RFE sama-sama menghasilkan akurasi tertinggi sebesar 0,50, unggul dibandingkan dua metode seleksi fitur lainnya.

Temuan ini mengindikasikan bahwa baik MI maupun RFE memiliki kapabilitas yang kuat dalam menyeleksi fitur-fitur yang relevan terhadap target klasifikasi, serta mampu mempertahankan performa model secara konsisten di berbagai algoritma. Berdasarkan pertimbangan tersebut, kedua metode tersebut dipilih untuk digunakan pada tahap penyesuaian *hyperparameter* berikutnya. Namun, penggunaannya tidak digabungkan dalam satu proses, melainkan diterapkan secara terpisah dalam dua scenario, yaitu:

- Skenario 1: Penyesuaian *hyperparameter* dilakukan pada model yang dibangun menggunakan *subset* fitur hasil seleksi metode Mutual Information (MI).
- Skenario 2: Penyesuaian *hyperparameter* dilakukan pada model dengan *subset* fitur hasil seleksi metode Recursive Feature Elimination (RFE).

Strategi ini memungkinkan evaluasi yang lebih objektif terhadap pengaruh masing-masing metode seleksi fitur terhadap performa model setelah proses optimasi *hyperparameter* dilakukan.

4.4 Penyesuaian Hyperparameter

Untuk meningkatkan kinerja masing-masing algoritma *machine learning*, dilakukan proses penyesuaian (*tuning*) *hyperparameter* terhadap delapan algoritma yang digunakan. Proses ini diterapkan secara terpisah berdasarkan dua skenario seleksi fitur yang telah ditentukan sebelumnya, yaitu Mutual Information (MI) dan Recursive Feature Elimination (RFE). Penyesuaian *hyperparameter* dilakukan menggunakan teknik *Grid Search* dengan *3-fold cross-validation* pada data latih dan ruang pencarian parameter yang telah dijelaskan pada BAB 3 Metodologi Penelitian.

4.4.1 Penyesuaian Hyperparameter pada Skenario Mutual Information

Pada skenario ini, optimasi *hyperparameter* dilakukan terhadap model yang dibangun menggunakan *subset* fitur hasil seleksi metode Mutual Information. Hasil optimasi *hyperparameter* ditampilkan pada Tabel 4.4.1

Tabel 4.4.1 Nilai *Hyperparameter* Optimal dari Mutual Information

Algoritma	Parameter
Decision Tree	criterion = 'entropy' max_depth = 10 max_features = None, min_samples_leaf = 2 min_samples_split = 2
SVM	C = 10 gamma = 'scale' kernel = 'rbf'
Naïve Bayes	var_smoothing = np.float64(1e-06)
Logistic Regression	C = 10 penalty = 'l2' solver = 'newton-cg'
KNN	n_neighbors = 9 weights = 'uniform' metric = 'minkowski' p = 1
Random Forest	max_depth = 20 max_features = 'log2' min_samples_leaf = 4 n_estimators = 300
AdaBoost	learning_rate = 1 n_estimators = 300
XGBoost	gamma = 0.3 learning_rate = 0.1 max_depth = 5 n_estimators = 300 reg_alpha = 0.01 reg_lambda = 2

4.4.2 Penyesuaian Hyperparameter pada Skenario Recursive Feature Elimination

Untuk skenario kedua, optimasi *hyperparameter* dilakukan pada model yang menggunakan *subset* fitur hasil dari Recursive Feature Elimination (RFE). Hasil akhir optimasi *hyperparameter* disajikan pada Tabel 4.4.2.

Tabel 4.4.2 Nilai *Hyperparameter* Optimal dari RFE

Algoritma	Parameter
Decision Tree	criterion = 'entropy' max_depth = 10 max_features = None, min_samples_leaf = 4 min_samples_split = 2
SVM	C = 1 gamma = 'scale' kernel = 'rbf'
Naïve Bayes	var_smoothing = np.float64(0.01)
Logistic Regression	C = None solver = 'newton-cg'
KNN	n_neighbors = 9 weights = 'uniform' metric = 'minkowski' p = 1
Random Forest	max_depth = 20 max_features = ‘sqrt’ min_samples_leaf = 4 n_estimators = 300
AdaBoost	learning_rate = 0.5 n_estimators = 300
XGBoost	gamma = 0.3 learning_rate = 0.1 max_depth = 5 n_estimators = 300 reg_alpha = 0.01 reg_lambda = 1

4.5 Pemodelan Ensemble Stacking

Pemilihan kombinasi model *stacking* dalam penelitian ini didasarkan pada adaptasi dari berbagai studi terdahulu yang telah menerapkan metode *stacking*, baik pada domain yang serupa, yaitu deteksi serangan siber, maupun domain yang berbeda. Meskipun sebagian besar referensi tersebut menggunakan pendekatan klasifikasi biner dan memanfaatkan dataset yang berbeda, struktur kombinasi model yang digunakan tetap dianggap relevan sebagai dasar eksplorasi dalam konteks klasifikasi *multiclass*.

Namun, penting untuk ditegaskan bahwa penelitian ini secara khusus membatasi penggunaan algoritma hanya pada delapan algoritma utama yang telah digunakan sejak awal eksperimen, yaitu:

- Decision Tree (DT)
- Support Vector Machine (SVM)
- Naïve Bayes (NB)
- Logistic Regression (LR)
- K-Nearest Neighbors (KNN)
- Random Forest (RF)
- Adaptive Boosting (AB)
- Extreme Gradient Boosting (XGBoost)

Dengan pembatasan ini, algoritma yang tidak termasuk dalam daftar tersebut tidak dilibatkan dalam proses *stacking*, meskipun ada di dalam referensi studi lain. Sebagai contoh, apabila sebuah studi menggunakan kombinasi *base learner* seperti Random Forest, XGBoost, dan Extra-Trees Classifier, maka dalam penelitian ini algoritma Extra-Trees Classifier tidak digunakan karena berada di luar ruang lingkup algoritma yang telah ditentukan.

Beberapa kombinasi model *stacking* yang diadopsi atau dijadikan acuan dari studi-studi terdahulu disajikan pada Tabel 4.5.1.

Tabel 4.5.1 Kombinasi Model *Stacking* Berdasarkan Studi Terdahulu

Kode Referensi	Studi/ Referensi	Domain	Base Learner	Meta Learner	Keterangan
Stacking_1	(Ghasemieh, Lloyed, Bahrami, Vajar, & Kashef, 2023)	Medis	KNN, Random Forest, Logistic Regression, SVM, Decision Tree, XGBoost	XGBoost	Klasifikasi biner, Dataset berbeda, Domain berbeda
Stacking_2			KNN, Random Forest, Logistic Regression, SVM, Decision Tree, XGBoost	Logistic Regression	
Stacking_3	(Urmi dkk., 2024)	Siber	Random Forest, XGBoost, Extra-Trees Classifier	Logistic Regression	Klasifikasi Biner, Dataset berbeda
Stacking_4	(Arya, Kumar, & Ahmad, 2023)	Siber	SVM, Naive Bayes, Decision Tree, Random Forest	Logistic Regression	Klasifikasi biner, Dataset sama

Berdasarkan hasil analisis dari berbagai studi terdahulu pada Tabel 4.5.1 serta hasil evaluasi pada model individual, peneliti mengusulkan satu model s*tacking ensemble*. Pemilihan ini tidak hanya mengacu pada frekuensi penggunaan di literatur, tetapi mempertimbangkan kesesuaian kekuatan masing-masing algoritma terhadap karakteristik data lalu lintas serangan DDoS yang bersifat berdimensi tinggi, mengandung *noise*, serta memiliki pola distribusi kelas yang kompleks. Model yang diusulkan terdiri atas:

- *Base-learners*: Random Forest (RF), AdaBoost, XGBoost, dan Decision Tree (DT)
- *Meta-learner*: Logistic Regression (LR)

Keempat algoritma *base learner* dipilih dengan pertimbangan sebagai berikut:

- Random Forest (RF): Mampu menangani data dengan jumlah fitur yang besar, mengurangi risiko *overfitting* melalui proses *bagging*, dan efektif dalam mempelajari interaksi *non-linear* antar fitur.
- AdaBoost: Memperkuat model lemah dengan fokus pada data yang sulit diklasifikasikan, sehingga membantu meningkatkan akurasi pada kelas yang sering salah prediksi.
- XGBoost: Menawarkan optimisasi *gradient boosting* yang efisien dengan kemampuan *regularization* untuk mengatasi *overfitting* dan menangani pola *non-linear* yang kompleks.
- Decision Tree (DT): Memberikan struktur pembelajaran yang sederhana, interpretatif, dan cepat.

Pemilihan Logistic Regression sebagai *meta learner* didasarkan pada kemampuannya menggabungkan *output* probabilistik dari *base learner* yang kompleks menjadi prediksi akhir yang stabil, serta terbukti efektif di sejumlah studi terdahulu (Tabel 4.5.1) dalam mengurangi bias dan variansi pada tahap *meta level*.

Penelitian ini mengevaluasi dua konfigurasi *stacking* berdasarkan parameter `passthrough`. Pada konfigurasi pertama (`passthrough=False`), *meta learner* hanya menerima *output* prediksi dari *base learners*. Sementara pada konfigurasi kedua (`passthrough=True`), *meta learner* juga menerima fitur asli sebagai *input* tambahan.

Tujuan dari pengujian ini adalah untuk mengamati pengaruh keberadaan fitur asli terhadap performa akhir model. Evaluasi dilakukan pada dua metode seleksi fitur, yaitu Mutual Information (MI) dan Recursive Feature Elimination (RFE), menggunakan data hasil prepemrosesan serta *hyperparameter* yang telah dioptimasi untuk memastikan hasil yang representatif.

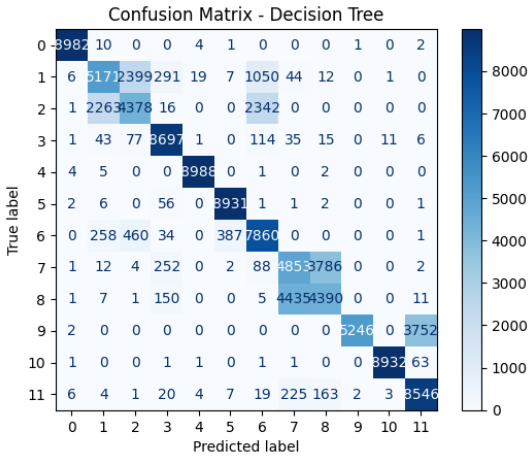
4.6 Evaluasi Model

4.6.1 Confusion Matrix

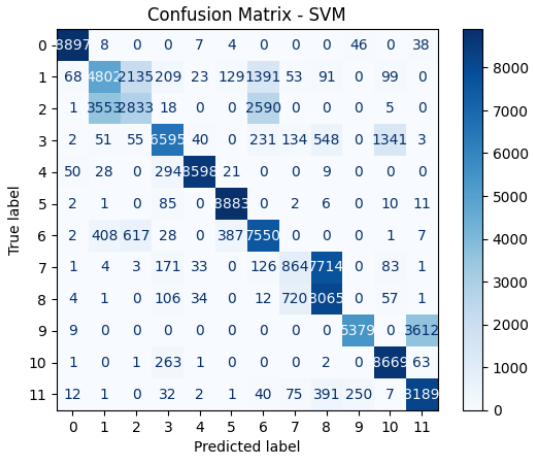
Hasil prediksi dari masing-masing model divisualisasikan dalam bentuk *confusion matrix* untuk memberikan gambaran yang lebih mendalam terkait performa klasifikasi pada setiap kelas. Visualisasi ini berguna dalam mengidentifikasi pola-pola keberhasilan maupun kesalahan klasifikasi yang terjadi, sehingga dapat digunakan sebagai dasar untuk analisis lebih lanjut.

Confusion Matrix dengan Mutual Information

Berdasarkan Gambar 4.6.1, model Decision Tree menunjukkan kinerja prediksi yang cukup baik pada beberapa kelas, yaitu 0, 3, 4, 5, 6, 10, dan 11. Namun, kesalahan prediksi terlihat pada kelas 1, 2, 7, 8, 9. Sedangkan pada Gambar 4.6.2, model Support Vector Machine (SVM) menunjukkan kinerja prediksi yang cukup baik pada kelas 0, 4, 5, 10, dan 11. Meskipun demikian, SVM gagal dalam memprediksi kelas 1, 2, 3, 6, 7, 8, dan 9.

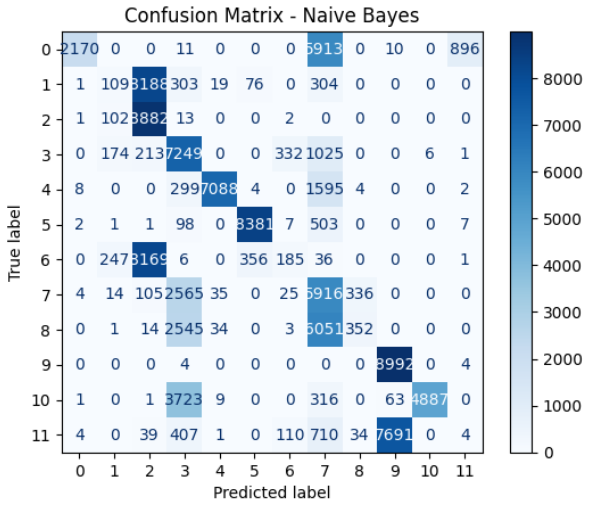


Gambar 4.6.1 *Confusion Matrix* MI Model DT

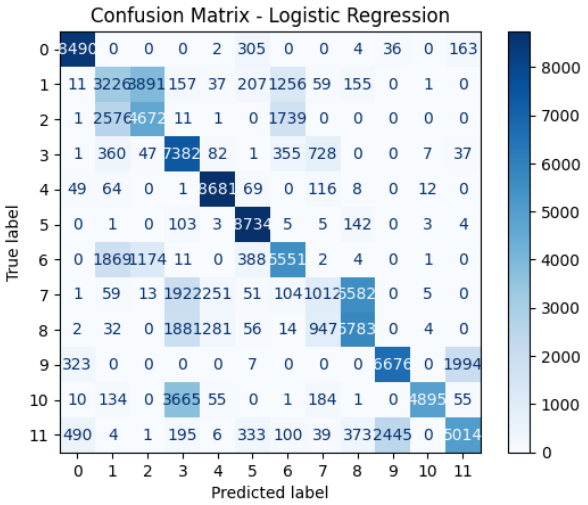


Gambar 4.6.2 *Confusion Matrix* MI Model SVM

Berdasarkan Gambar 4.6.3, model Naïve Bayes mampu melakukan prediksi dengan baik pada sejumlah kelas seperti 2, 3, 4, 5, 9. Akan tetapi, terdapat kesalahan dalam memprediksi kelas 0, 1, 6, 7, 8, 10, 11. Sementara itu, Gambar 4.6.4 menunjukkan bahwa model Logistic Regression bekerja cukup efektif pada kelas 0, 3, 4, 5, 6, 8, 9, 10, 11, meskipun tidak berhasil mengklasifikasikan kelas 1, 2, 3, 7, 8, dan 9 secara akurat.

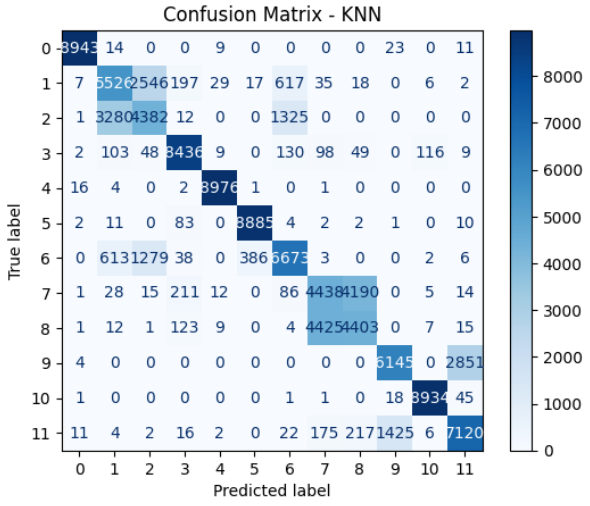


Gambar 4.6.3 Confusion Matrix MI Model NB

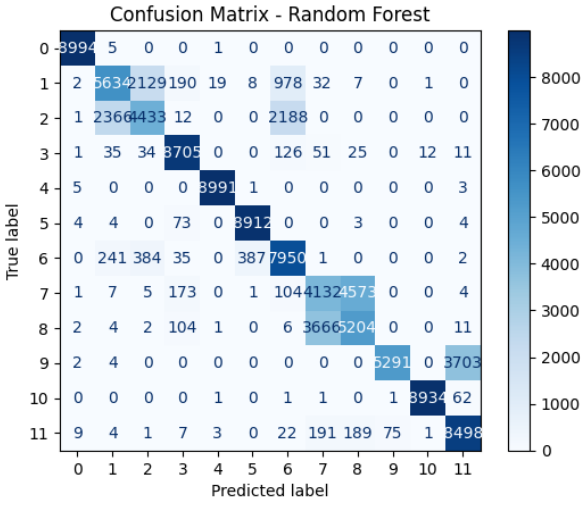


Gambar 4.6.4 *Confusion Matrix* MI Model LR

Dalam Gambar 4.6.5 terlihat bahwa model KNN berhasil memprediksi kelas 0, 1, 3, 4, 5, 6, 9, 10, dan 11 dengan baik. Namun, masih terjadi kesalahan klasifikasi pada kelas 2, 7, 8. Di sisi lain, Gambar 4.6.6 memperlihatkan bahwa model Random Forest cukup akurat dalam mengenali kelas 0, 1, 3, 4, 5, 6, 8, 9, 10, 11, tetapi kurang mampu membedakan kelas 2 dan 7.

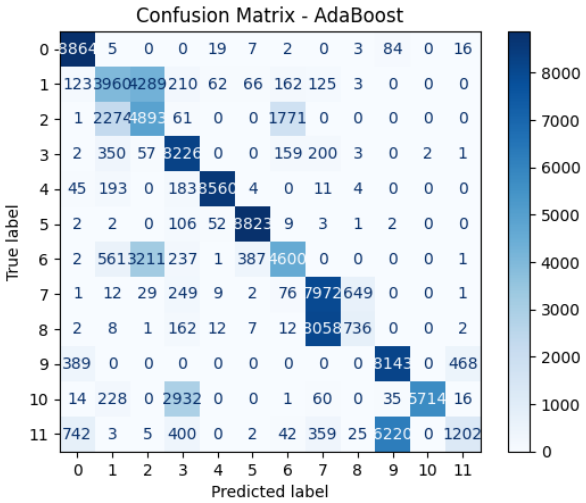


Gambar 4.6.5 *Confusion Matrix* MI Model KNN

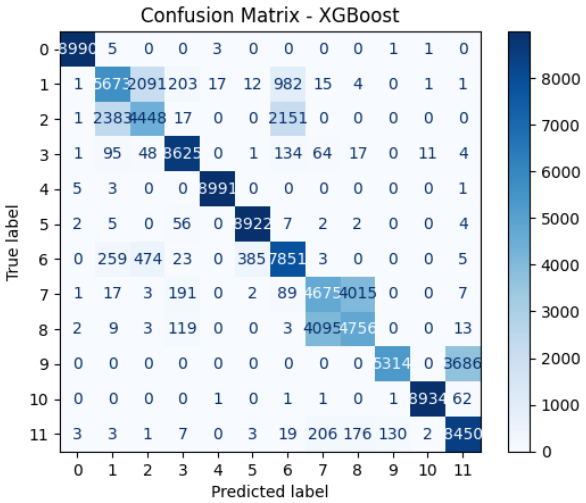


Gambar 4.6.6 *Confusion Matrix* MI Model RF

Seperti ditunjukkan pada Gambar 4.6.7, AdaBoost menunjukkan hasil yang baik dalam memprediksi kelas 0, 3, 4, 5, 7, 9, dan 10. Meskipun demikian, model ini kurang baik dalam menangani kelas 1, 2, 6, 8, dan 11. Sementara itu, pada Gambar 4.6.8, model XGBoost memiliki performa yang cukup baik terhadap kelas 0, 1, 3, 4, 5, 6, 9, 10, dan 11, tetapi menunjukkan kelemahan pada prediksi kelas 2, 7, dan 8.

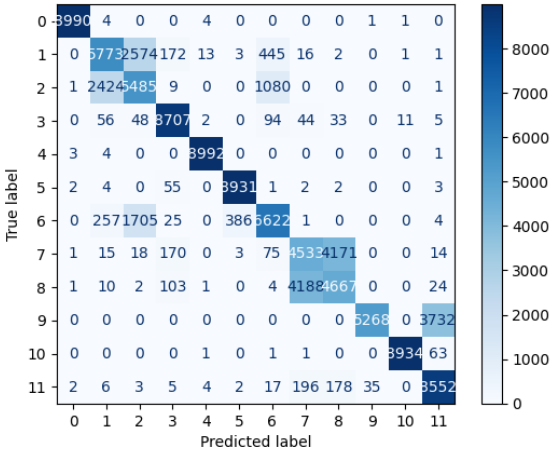


Gambar 4.6.7 *Confusion Matrix* MI Model AB

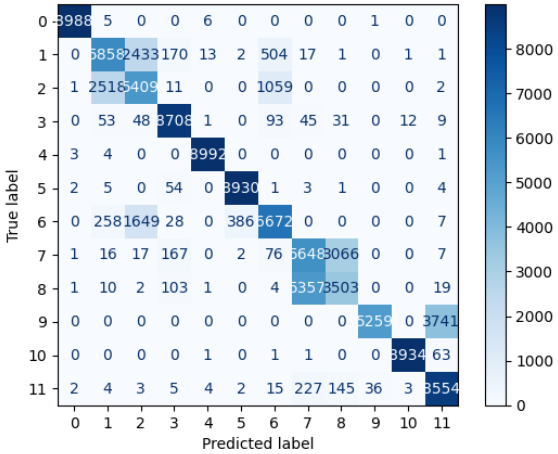


Gambar 4.6.8 *Confusion Matrix* MI Model XGB

Gambar 4.6.9 mengindikasikan bahwa model Stacking_1(`passthrough=True`) baik dalam mengklasifikasikan kelas 0, 1, 2, 3, 4, 5, 6, 9, 10, 11. Sebaliknya, performa model ini kurang optimal dalam mengklasifikasikan kelas 7 dan 8. Di Gambar 4.6.10, model Stacking_1(`passthrough=False`) baik dalam mengenali kelas 0, 1, 2, 3, 4, 5, 6, 7, 9, 10, dan 11, tetapi kurang dalam mengidentifikasi kelas 8.



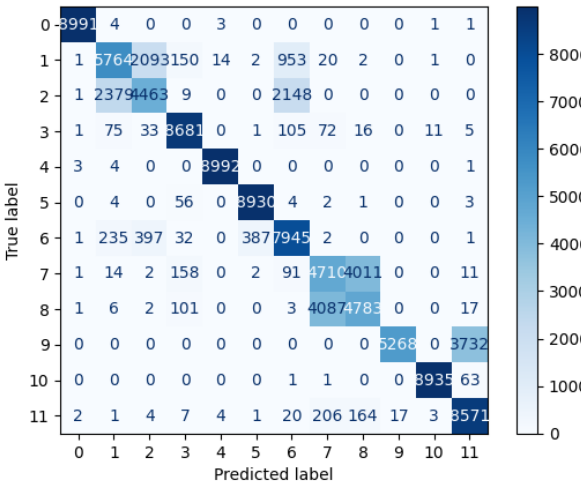
Gambar 4.6.9 Confusion Matrix MI Model Stacking_1 (`passthrough=True`)



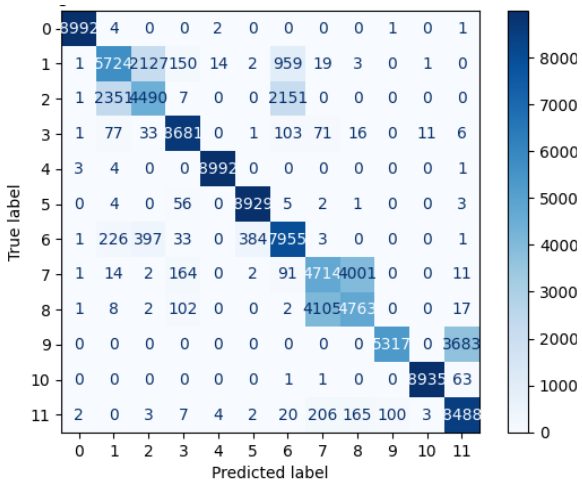
Gambar 4.6.10 *Confusion Matrix* MI Model Stacking_1 (`passthrough=False`)

Berdasarkan informasi dari Gambar 4.6.11, model Stacking_2(`passthrough=True`) mampu mengenali kelas 0, 1, 3, 4, 5, 6, 9, 10, dan 11 dengan baik, meskipun kurang baik dalam memprediksi kelas 2, 7, dan 8. Adapun Gambar 4.6.12 memperlihatkan bahwa model

Stacking_2 (`passthrough=False`) cukup efektif dalam memprediksi kelas 0, 1, 3, 4, 5, 6, 9, 10, dan 11, namun kesulitan dalam memetakan kelas 2, 7, dan 8.

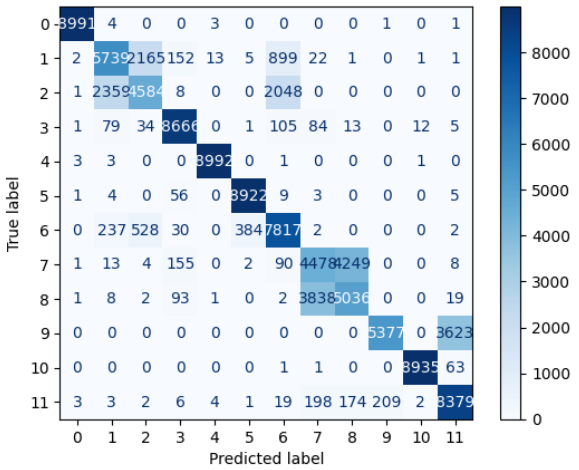


Gambar 4.6.11 *Confusion Matrix* MI Model Stacking_2 (`passthrough=True`)

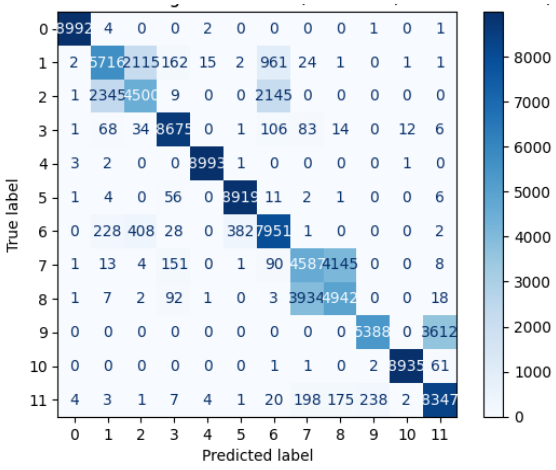


Gambar 4.6.12 *Confusion Matrix* MI Model Stacking_2 (`passthrough=False`)

Dari Gambar 4.6.13 dan Gambar 4.6.14, dapat diketahui bahwa model Stacking_3(`passthrough=True`) dan model Stacking_3(`passthrough=False`) memiliki performa prediksi yang baik pada kelas 0, 1, 3, 4, 5, 6, 9, 10, dan 11, tetapi masih mengalami banyak kesalahan pada kelas 2, 7, dan 8.

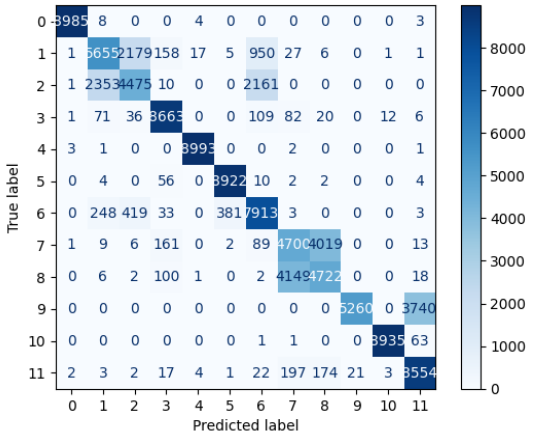


Gambar 4.6.13 *Confusion Matrix* MI Model Stacking_3 (`passthrough=True`)

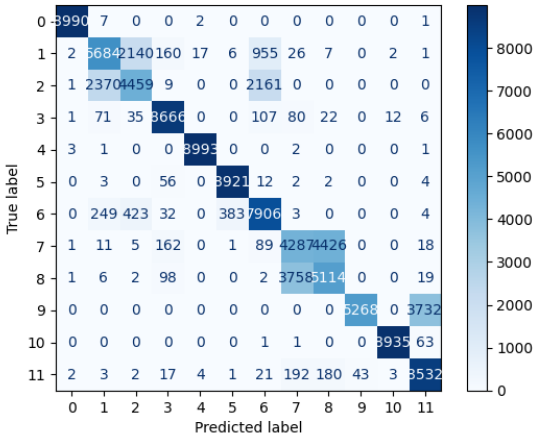


Gambar 4.6.14 *Confusion Matrix* MI Model Stacking_2 (`passthrough=False`)

Gambar 4.6.15 dan Gambar 4.6.16 menunjukkan bahwa model Stacking_4 baik dengan `passthrough=True` maupun `passthrough=False` mampu memprediksi kelas 0, 1, 3, 4, 5, 6, 9, 10, dan 11 dengan cukup baik, namun masih sering keliru pada kelas 2, 7, dan 8.

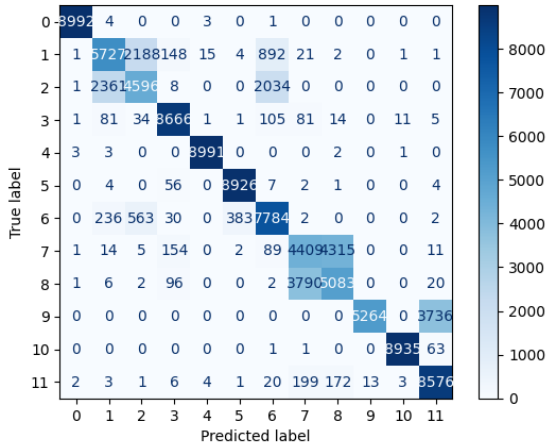


Gambar 4.6.15 *Confusion Matrix* MI Model Stacking_4 (`passthrough=True`)

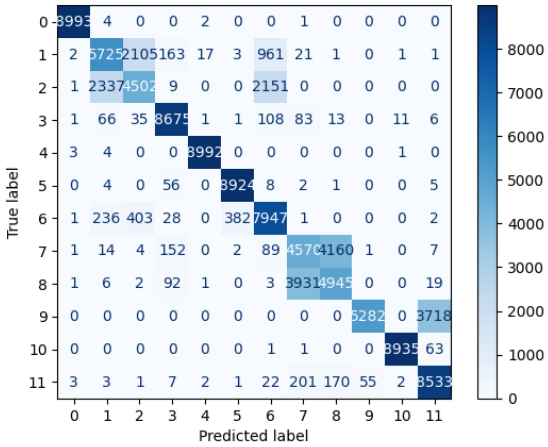


Gambar 4.6.16 *Confusion Matrix* MI Model Stacking_4 (`passthrough=False`)

Berdasarkan Gambar 4.6.17 dan Gambar 4.6.18, terlihat bahwa model Stacking_3 (baik dengan `passthrough=True` maupun `False`) berhasil melakukan prediksi yang cukup baik terhadap kelas 0, 1, 3, 4, 5, 6, 9, 10, dan 11, namun kesulitan dalam memprediksi kelas 2, 7, dan 8.



Gambar 4.6.17 *Confusion Matrix* MI Model Peneliti (`passthrough=True`)



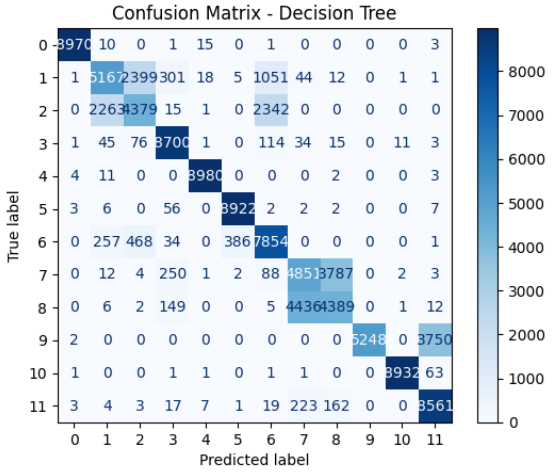
Gambar 4.6.18 *Confusion Matrix* MI Model Peneliti (`passthrough=False`)

Hasil evaluasi menggunakan *confusion matrix* pada skenario seleksi fitur Mutual Information menunjukkan bahwa sebagian besar model mengalami kesalahan dalam membedakan antara label kelas 1 (DrDoS_DNS) dan label kelas 2 (DrDoS_LDAP) serta antara label 7 (DrDoS_SSDP) dan label 8 (DrDoS_UDP). Selain itu, terdapat pola kesalahan yang konsisten di mana label asli kelas 2 (DrDoS_LDAP) sering diklasifikasikan sebagai label kelas

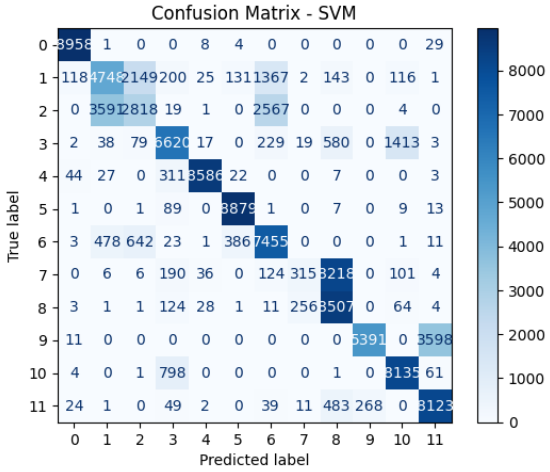
6 (DrDoS_SNMP), dan label 9 (SYN) juga sering kali diprediksi sebagai kelas 11 (UDP-lag). Hal ini mengisyaratkan adanya kemiripan karakteristik antar kelas yang menyebabkan model mengalami kesalahan klasifikasi pada kategori tersebut.

Confusion Matrix dengan RFE

Berdasarkan Gambar 4.6.19, model Decision Tree menunjukkan kinerja prediksi yang cukup baik pada beberapa kelas, yaitu 0, 3, 4, 5, 6, 9, 10, dan 11. Namun, kesalahan prediksi terlihat banyak pada kelas 1, 2, 7, dan 8. Sedangkan pada Gambar 4.6.20, model Support Vector Machine (SVM) menunjukkan kinerja prediksi yang cukup baik pada kelas 0, 3, 4, 5, 6, 8, 9, 10, dan 11. Meskipun demikian, SVM cukup banyak gagal dalam memprediksi kelas 1, 2, 7.

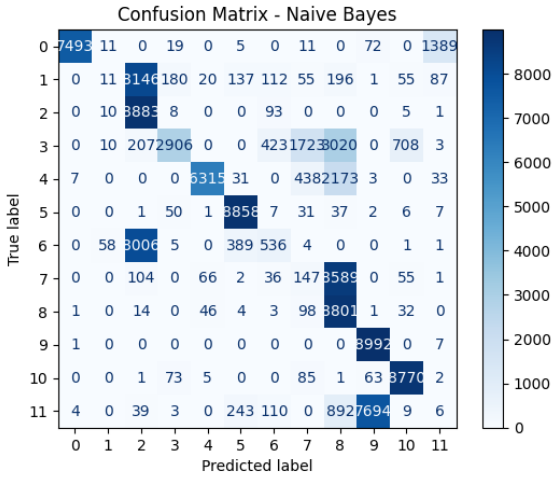


Gambar 4.6.19 *Confusion Matrix* RFE Model DT

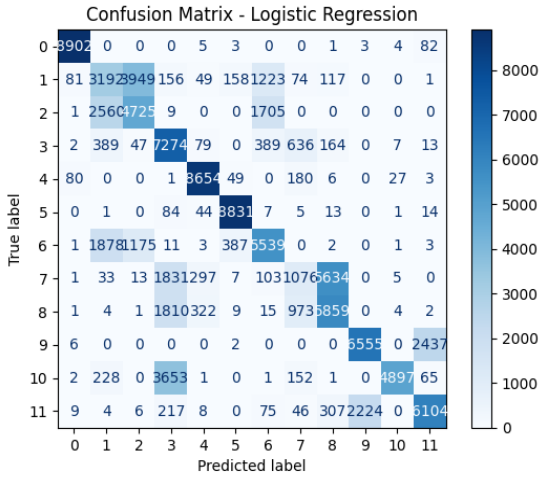


Gambar 4.6.20 *Confusion Matrix* RFE Model SVM

Berdasarkan Gambar 4.6.21, model Naïve Bayes mampu melakukan prediksi dengan baik pada sejumlah kelas seperti 0, 2, 4, 5, 8, 9, dan 10. Akan tetapi, masih banyak kesalahan dalam memprediksi kelas 1, 3, 6, 7, dan 11. Sementara itu, Gambar 4.6.22 menunjukkan bahwa model Logistic Regression bekerja cukup efektif pada kelas 0, 3, 4, 5, 6, 8, 9, dan 11, meskipun tidak berhasil mengklasifikasikan kelas 1, 2, 7, dan 10 secara baik.

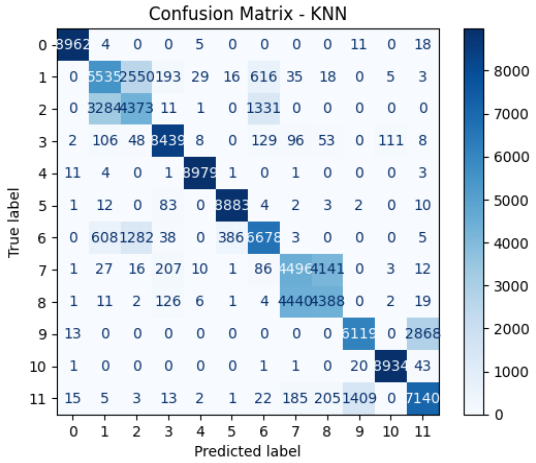


Gambar 4.6.21 *Confusion Matrix* RFE Model NB

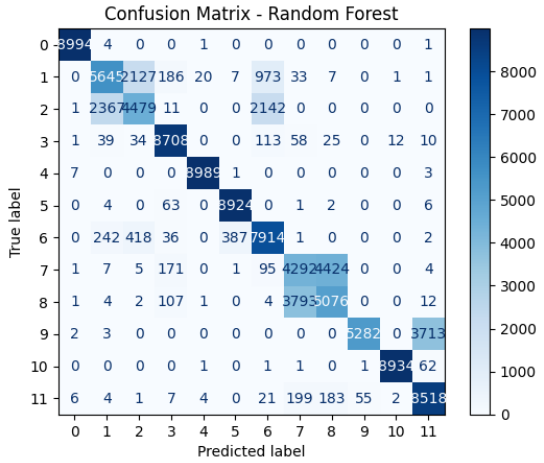


Gambar 4.6.22 *Confusion Matrix* RFE Model LR

Dalam Gambar 4.6.23 terlihat bahwa model KNN berhasil memprediksi kelas 0, 1, 3, 4, 5, 6, 9, 10, dan 11 dengan baik. Namun, masih banyak terjadi kesalahan klasifikasi pada kelas 2, 7, dan 8. Di sisi lain, Gambar 4.6.24 memperlihatkan bahwa model Random Forest cukup akurat dalam mengenali kelas 0, 1, 3, 4, 5, 6, 8, 9, 10, 11, tetapi kurang mampu membedakan kelas 2, 7, dan 8.

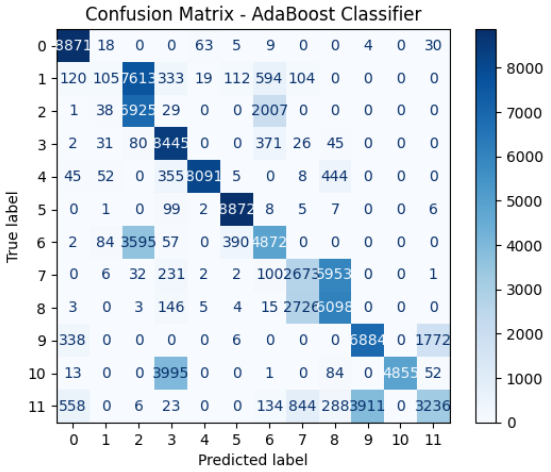


Gambar 4.6.23 *Confusion Matrix* RFE Model KNN

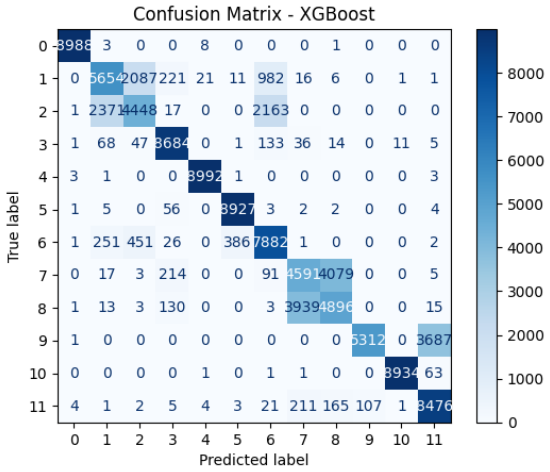


Gambar 4.6.24 *Confusion Matrix* RFE Model RF

Seperti ditunjukkan pada Gambar 4.6.25, AdaBoost menunjukkan hasil yang baik dalam memprediksi kelas 0, 2, 3, 4, 5, 8, dan 9. Meskipun demikian, model ini kurang baik dalam menangani kelas 1, 6, 7, 10, dan 11. Sementara itu, pada Gambar 4.6.26, model XGBoost memiliki performa yang cukup baik terhadap kelas 0, 1, 3, 4, 5, 6, 9, 10, dan 11, tetapi menunjukkan banyak kelemahan pada prediksi kelas 2, 7, dan 8.

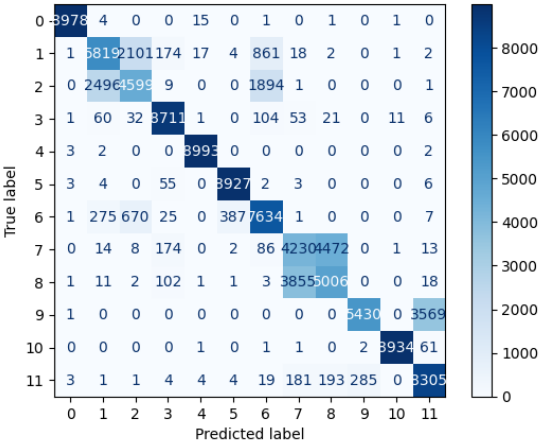


Gambar 4.6.25 *Confusion Matrix* RFE Model AB

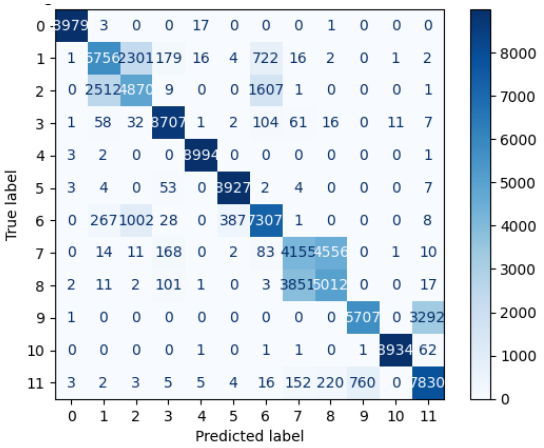


Gambar 4.6.26 *Confusion Matrix* RFE Model XGB

Gambar 4.6.27 dan Gambar 4.6.28 memperlihatkan bahwa model Stacking_1, baik dengan passthrough=True maupun passthrough=False, memberikan hasil prediksi yang cukup baik pada kelas 0, 1, 3, 4, 5, 6, 9, 10, dan 11. Namun, kedua model tersebut masih menunjukkan banyak kelemahan dalam mengklasifikasikan kelas 2, 7, dan 8.

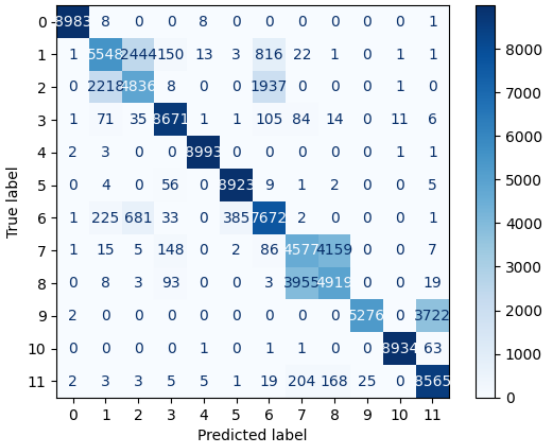


Gambar 4.6.27 *Confusion Matrix* RFE Model Stacking_1 (`passthrough=True`)

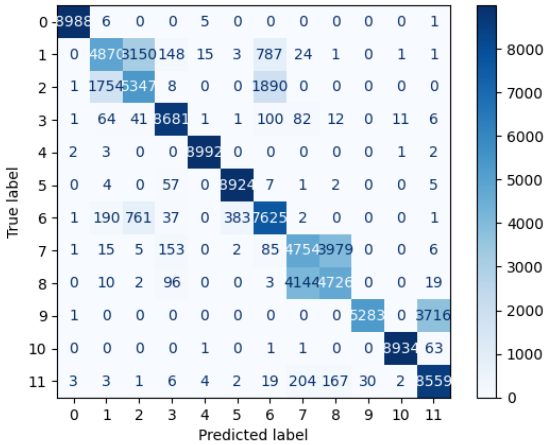


Gambar 4.6.28 *Confusion Matrix* RFE Model Stacking_1 (`passthrough=False`)

Berdasarkan informasi dari Gambar 4.6.29, model Stacking_2(`passthrough=True`) mampu mengenali kelas 0, 1, 3, 4, 5, 6, 9, 10, dan 11 dengan baik, meskipun kurang baik dalam memprediksi kelas 2, 7, dan 8. Adapun Gambar 4.6.30 memperlihatkan bahwa model Stacking_2 (`passthrough=False`) cukup efektif dalam memprediksi kelas 0, 2, 3, 4, 5, 6, 9, 10, dan 11, namun kesulitan dalam memetakan kelas 1, 7, dan 8.

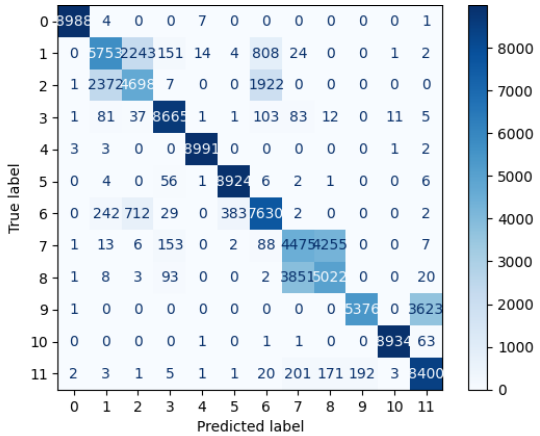


Gambar 4.6.29 *Confusion Matrix* RFE Model Stacking_2 (`passthrough=True`)

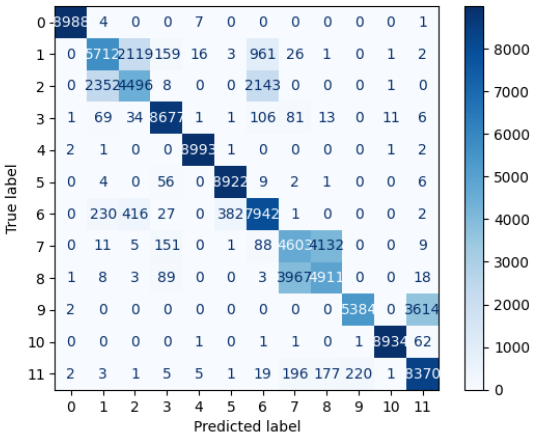


Gambar 4.6.30 *Confusion Matrix* RFE Model Stacking_2 (`passthrough=False`)

Berdasarkan Gambar 4.6.31 dan Gambar 4.6.32, dapat dilihat bahwa model Stacking_3 dengan konfigurasi `passthrough` aktif maupun tidak, mampu melakukan prediksi yang cukup baik terhadap kelas 0, 1, 3, 4, 5, 6, 9, 10, dan 11. Meskipun demikian, model masih sering melakukan kesalahan pada kelas 2, 7, dan 8.

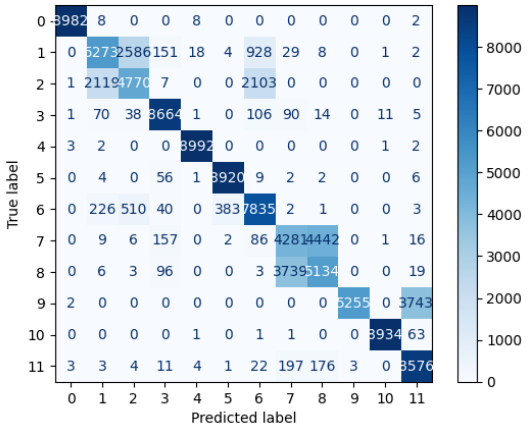


Gambar 4.6.31 *Confusion Matrix* RFE Model Stacking_3 (`passthrough=True`)

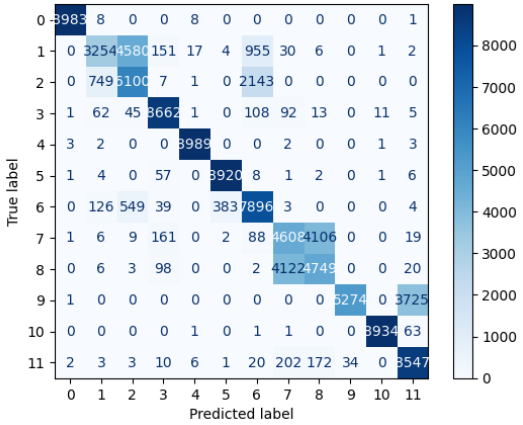


Gambar 4.6.32 *Confusion Matrix* RFE Model Stacking_3 (`passthrough=False`)

Gambar 4.6.33 mengindikasikan bahwa model Stacking_4(`passthrough=True`) cukup baik dalam mengklasifikasikan kelas 0, 1, 3, 4, 5, 6, 8, 9, 10, 11. Sebaliknya, performa model ini kurang optimal dalam mengklasifikasikan kelas 2 dan 7. Di Gambar 4.6.34, model Stacking_4(`passthrough=False`) baik dalam mengenali kelas 0, 2, 3, 4, 5, 6, 9, 10, dan 11, tetapi kurang baik dalam mengidentifikasi kelas 1, 7, dan 8.

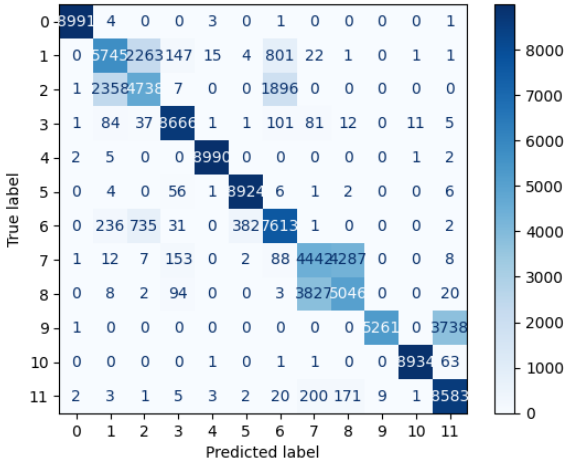


Gambar 4.6.33 *Confusion Matrix* RFE Model Stacking_4 (`passthrough=True`)

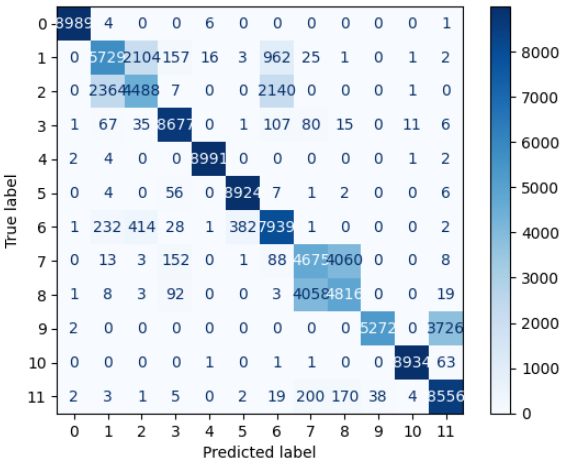


Gambar 4.6.34 *Confusion Matrix* RFE Model Stacking_4 (`passthrough=False`)

Hasil visualisasi pada Gambar 4.6.35 dan Gambar 4.6.36 menunjukkan bahwa model Stacking_3 dalam kedua konfigurasi passthrough dapat mengklasifikasikan sebagian besar kelas seperti 0, 1, 3, 4, 5, 6, 9, 10, dan 11 dengan baik, tetapi tetap mengalami kesalahan yang cukup sering pada kelas 2, 7, dan 8.



Gambar 4.6.35 *Confusion Matrix* RFE Model Peneliti (`passthrough=True`)



Gambar 4.6.36 *Confusion Matrix* RFE Model Peneliti (`passthrough=False`)

Evaluasi model menggunakan *confusion matrix* pada skenario seleksi fitur dengan Recursive Feature Elimination (RFE) menunjukkan bahwa banyak model mengalami kesulitan dalam membedakan label kelas 1 (DrDoS_DNS) dengan kelas 2 (DrDoS_LDAP), serta kelas 7 (DrDoS_SSDP) dengan kelas 8 (DrDoS_UDP). Selain itu, kesalahan yang sering terjadi

adalah saat label asli kelas 2 (DrDoS_LDAP) diklasifikasikan sebagai kelas 6 (DrDoS_SNMP), dan kelas 9 (SYN) sering terdeteksi keliru sebagai kelas 11 (UDP-lag).

Kesalahan prediksi ini sejalan dengan studi yang dilakukan (Chartuni & Márquez, 2021) yang menyebutkan bahwa model sulit membedakan kelas seperti DrDoS_DNS dan DrDoS_LDAP karena serangannya memiliki sifat dan data yang sangat mirip sehingga nilai atribut pada dataset juga mirip dan menurunkan kinerja model. Selain itu, proses pemilihan fitur dengan metode RFE dan MI yang dianggap paling relevan tetap memiliki risiko menghilangkan beberapa fitur yang sebenarnya penting untuk membedakan kelas yang mirip. Perbedaan kecil di antara dua kelas bisa terletak pada fitur-fitur tertentu yang jarang muncul atau tidak menonjol. Jika fitur-fitur tersebut ikut terhapus, model kehilangan "petunjuk" penting yang dibutuhkannya, sehingga prediksi menjadi lebih sulit dan akurasi menurun pada pasangan kelas yang sulit dibedakan.

4.6.2 Accuracy, Precision, Recall, F1-Score dan Inference Time

Evaluasi performa dilakukan terhadap masing-masing model *machine learning* berdasarkan empat metrik utama, yaitu *accuracy*, *precision*, *recall*, *f1-score,* dan *inference time.* Evaluasi ini mencakup delapan algoritma utama yang digunakan dalam penelitian, dengan dua skenario seleksi fitur, yaitu Mutual Information (MI) dan Recursive Feature Elimination (RFE). Tujuan dari evaluasi ini adalah untuk mengetahui sejauh mana efektivitas setiap model dalam mengklasifikasikan serangan DDoS ke dalam beberapa kelas. Hasil kinerja masing-masing model pada kedua skenario seleksi fitur disajikan pada Tabel 4.6.1.

Tabel 4.6.1 Kinerja Model pada Seleksi Fitur MI dan RFE

Model	MI					RFE				
	Acc	Prec	Rec	F1	Inference Time (ms)	Acc	Prec	Rec	F1	Inference Time (ms)
Decision Tree	0.7868	0.7941	0.7868	0.7823	0.0778	0.7866	0.7940	0.7866	0.7821	0.0738
SVM	0.7344	0.7380	0.7344	0.7124	10.36	0.7271	0.7344	0.7271	0.6988	11.4587
Naive Bayes	0.5019	0.5359	0.5019	0.4319	0.2920	0.5714	0.5442	0.5714	0.4977	0.5041
Logistic Regression	0.6493	0.6555	0.6493	0.6369	0.1217	0.663	0.6707	0.663	0.6522	0.1273
KNN	0.7672	0.7670	0.7672	0.7663	8.1767	0.7678	0.7677	0.7678	0.7669	7.5568
Random Forest	0.7933	0.8003	0.7933	0.7891	8.3855	0.794	0.8015	0.794	0.7900	8.5108
Adaboost	0.6638	0.6908	0.6638	0.6315	23.0547	0.6474	0.6580	0.6474	0.6204	23.1649
XGBoost	0.7903	0.7969	0.7903	0.7838	0.3420	0.7942	0.8009	0.7942	0.7903	0.7889

Berdasarkan Tabel 4.6.1, RF dan XGB menunjukkan performa klasifikasi terbaik di antara model individual. Pada skenario Mutual Information, RF mencapai *accuracy* sebesar 0,7933 dan *f1-score* 0,7891, sementara XGB menyusul dengan *accuracy* 0,7903 dan *f1-score* 0,7838. Hasil yang hampir identik juga terlihat pada skenario RFE, di mana RF dan XGB tetap menempati peringkat teratas dalam hal *accuracy* dan stabilitas metrik lainnya.Model RF dan XGB juga memiliki waktu *inference* yang relatif efisien, yaitu sekitar 8,3–8,5 milidetik untuk RF dan 0,34–0,78 milidetik untuk XGB, menjadikannya pilihan unggul secara performa maupun efisiensi waktu.

Di sisi lain, Naive Bayes (NB) mencatatkan hasil paling rendah dalam *accuracy* dan *F1-score*, meskipun memiliki waktu *inference* tercepat (0,29 milidetik pada MI dan 0,50 milidetik pada RFE). Hal ini menunjukkan bahwa kecepatan *inference* tidak selalu sejalan dengan kualitas prediksi, khususnya pada data kompleks dengan banyak kelas seperti pada kasus serangan DDoS. Support Vector Machine (SVM) dan K-Nearest Neighbors (KNN) memiliki waktu *inference* yang cukup tinggi, yaitu lebih dari 8 milidetik untuk KNN dan 11 milidetik untuk SVM.

Model Decision Tree (DT) dan Logistic Regression (LR) menunjukkan performa menengah pada seluruh metrik dengan waktu *inference* yang cepat (kurang dari 0,15 milidetik), menjadikan keduanya cukup efisien meskipun kalah dari RF dan XGB dalam hal *accuracy*. Sementara itu, AdaBoost (AB) mencatat waktu *inference* paling lama, yaitu sekitar 23 milidetik pada kedua skenario, dengan performa klasifikasi yang tergolong sedang.

Dari kedua skenario seleksi fitur, dapat disimpulkan bahwa RFE memberikan sedikit peningkatan pada model-model tertentu seperti Naive Bayes dan Logistic Regression, namun tidak secara signifikan mengubah peringkat performa antar model. Kombinasi antara *accuracy* tinggi dan efisiensi waktu *inference* menjadikan Random Forest dan XGBoost sebagai model yang paling seimbang.

Evaluasi Ensemble Stacking Skenario Mutual Information

Evaluasi model *ensemble stacking* dilakukan dengan menggunakan *subset* fitur hasil seleksi Mutual Information. Lima konfigurasi *stacking* dibandingkan, termasuk Model Peneliti sebagai model akhir yang dipilih. Evaluasi dilakukan terhadap dua konfigurasi parameter *passthrough* (*True* dan *False*), yang memengaruhi apakah *input* asli disertakan dalam model *meta-learner*. Tabel 4.6.2 Menampilkan perbandingan kinerja model *ensemble stacking* terhadap dua konfigurasi parameter *passthrough* (*True* dan *False*) pada skenario MI.

Tabel 4.6.2 Kinerja Model *Ensemble Stacking* pada Seleksi Fitur MI

Model	passthrough=True					passthrough=False				
	Acc	Prec	Rec	F1	Inference Time (Ms)	Acc	Prec	Rec	F1	Inference Time (ms)
Stacking_1	0.7912	0.8020	0.7912	0.7902	49.0763	0.7912	0.8020	0.7912	0.7889	47.5131
Stacking_2	0.7966	0.8047	0.7966	0.7928	48.0091	0.7961	0.8033	0.7961	0.7924	47.6859
Stacking_3	0.7955	0.8016	0.7955	0.7922	13.8666	0.7957	0.8017	0.7957	0.7922	14.8923
Stacking_4	0.7942	0.8024	0.7942	0.7905	19.2552	0.7940	0.8019	0.7940	0.7901	20.6050
Model Peneliti	0.7958	0.8039	0.7958	0.7924	48.3376	0.7965	0.8043	0.7965	0.7928	50.6018

Berdasarkan Tabel 4.6.2, Stacking_2 dengan `passthrough=True` mencatat performa terbaik, dengan *accuracy* sebesar 0,7966, *precision* 0,8047, *recall* 0,7966, dan *f1-score* 0,7928. Hasil ini sedikit mengungguli Model Peneliti dan konfigurasi *stacking* lainnya. Namun, Stacking_2 memiliki waktu *inference* yang cukup tinggi, yaitu sekitar 48 milidetik. Hal serupa juga berlaku untuk Model Peneliti dengan parameter `passthrough=False`, yang memiliki waktu *inference* tertinggi sebesar 50,60 milidetik, meskipun performanya sangat mendekati Stacking_2 dengan *accuracy* 0,7965 dan *f1-score* 0,7928.

Sebaliknya, Stacking_3 menunjukkan efisiensi waktu inferensi paling baik, dengan waktu hanya 13,87 milidetik (`passthrough=True`) dan 14,89 milidetik (`passthrough=False`). Meskipun demikian, performa klasifikasinya berada di bawah Stacking_2 dan Model Peneliti, dengan *accuracy* 0,7955–0,7957 dan *f1-score* 0,7922.

Stacking_1 dan Stacking_4 memiliki performa yang stabil dan cukup tinggi, tetapi masih berada di bawah tiga model teratas. Waktu *inference* keduanya berkisar antara 19–49 milidetik, tergantung konfigurasi `passthrough` yang digunakan.

Secara umum, penggunaan parameter `passthrough` dalam *ensemble stacking* menunjukkan pengaruh yang bervariasi tergantung pada konfigurasi model. Dalam skenario seleksi fitur menggunakan Mutual Information, efek dari `passthrough` tidak selalu bersifat konsisten. Pada beberapa model seperti Stacking_2, penggunaan `passthrough=True` justru memberikan peningkatan performa dibandingkan dengan `passthrough=False`. Hal ini mengindikasikan bahwa pada konfigurasi tertentu, penambahan fitur asli ke tahap *meta-learner* dapat memberikan informasi tambahan yang berguna dalam pengambilan keputusan akhir.

Evaluasi Ensemble Stacking Skenario Recursive Feature Elimination

Evaluasi model *ensemble stacking* dilakukan dengan menggunakan *subset* fitur hasil seleksi Recursive Feature Elimination (RFE). Lima konfigurasi *ensemble stacking* dibandingkan, termasuk Model Peneliti sebagai model akhir yang dipilih. Evaluasi dilakukan terhadap dua konfigurasi parameter *passthrough* (*True* dan *False*). Tabel 4.6.3 Menampilkan perbandingan kinerja model *ensemble stacking* terhadap dua konfigurasi parameter *passthrough* (*True* dan *False*) pada skenario RFE.

Tabel 4.6.3 Kinerja Model *Ensemble Stacking* pada Seleksi Fitur RFE

Model	passthrough=True					passthrough=False				
	Acc	Prec	Rec	F1	Inference Time (ms)	Acc	Prec	Rec	F1	Inference Time (ms)
Stacking_1	0.7922	0.7972	0.7922	0.7891	48.4806	0.7886	0.7905	0.7886	0.7866	47.9881
Stacking_2	0.7953	0.8035	0.7953	0.7924	49.7194	0.7933	0.8028	0.7933	0.7906	48.2597
Stacking_3	0.7949	0.8012	0.7949	0.7921	14.9549	0.7956	0.8018	0.7956	0.7921	16.2814
Stacking_4	0.7927	0.8012	0.7927	0.7892	20.9764	0.7862	0.8022	0.7862	0.7804	20.9697
Model Peneliti	0.7956	0.8039	0.7956	0.7927	53.0514	0.7962	0.8042	0.7962	0.7925	54.3377

Berdasarkan Tabel 4.6.3, Model Peneliti dengan `passthrough=False` mencatatkan performa terbaik di antara seluruh konfigurasi. Model ini memperoleh *accuracy* sebesar 0,7962, *precision* 0,8042, *recall* 0,7962, dan *F1-score* 0,7925. Performa ini mengungguli konfigurasi model *stacking* lainnya, meskipun memiliki *inference time* tertinggi, yaitu sebesar 53,05–54,33 milidetik.

Sementara itu, Stacking_3 dengan `passthrough=False` menunjukkan kinerja yang hampir setara dengan Model Peneliti, yaitu dengan *accuracy* 0,7956 dan *f1-score* 0,7921, tetapi memiliki *inference time* yang jauh lebih efisien, yaitu hanya 16,28 milidetik.

Model Stacking_2 dengan `passthrough=True` juga mencatatkan hasil yang kompetitif dengan *accuracy* 0,7953 dan *f1-score* 0,7924. Namun, model Stacking_2 memiliki *inference time* yang cukup tinggi, yaitu sebesar 49,72 milidetik. Sementara itu, Stacking_1 dan Stacking_4 secara umum menunjukkan performa yang lebih rendah dibandingkan tiga konfigurasi teratas, terutama pada saat `passthrough=False`.

Hasil ini menunjukkan bahwa penggunaan parameter `passthrough` tidak selalu memberikan dampak yang signifikan terhadap peningkatan performa klasifikasi. Pada beberapa model, seperti Stacking_3, dan Model Peneliti, penggunaan `passthrough=False` justru menghasilkan performa yang lebih baik tetapi berdampak pada peningkatan *inference time*. Di sisi lain, model seperti Stacking_1, Stacking_2, dan Stacking_4 menunjukkan bahwa aktivasi *passthrough* dapat meningkatkan metrik evaluasi, tetapi berdampak pada peningkatan *inference time*.

Dengan demikian, pemilihan model *stacking* terbaik perlu mempertimbangkan keseimbangan antara performa prediksi dan efisiensi komputasi, sesuai dengan kebutuhan dan konteks implementasi sistem. Dalam penelitian ini, Model Peneliti dipilih sebagai model utama untuk evaluasi lanjutan karena konsisten menghasilkan performa klasifikasi terbaik dalam berbagai metrik, baik pada skenario Mutual Information maupun RFE.

4.6.3 ROC dan AUC

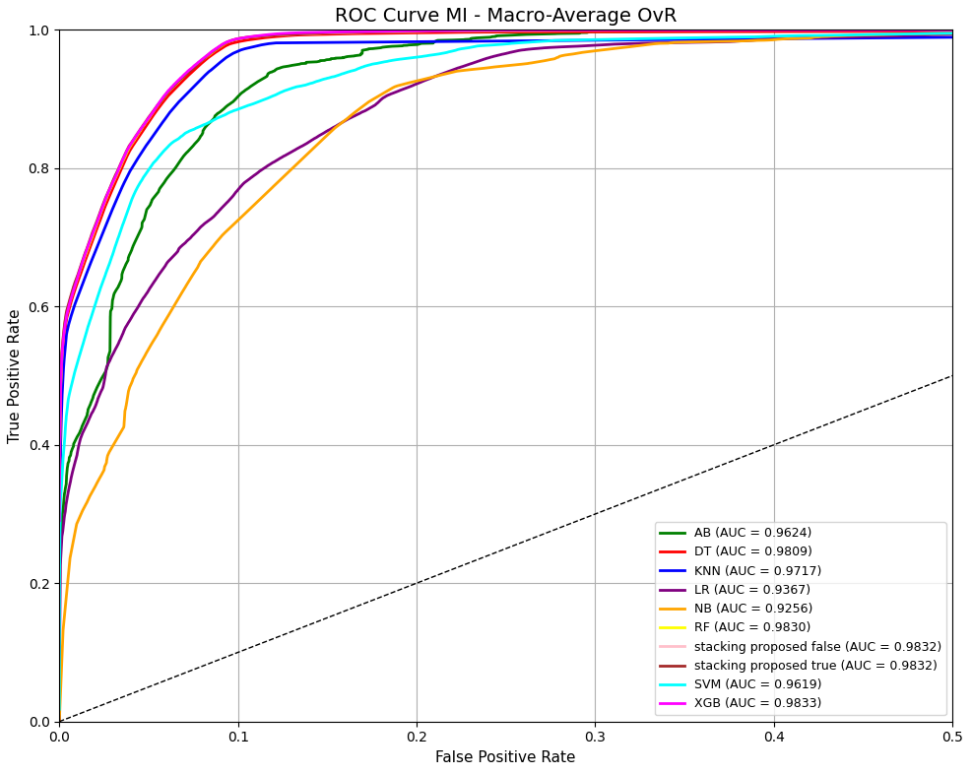
Receiver Operating Characteristic (ROC) dan *Area Under Curve* (AUC) merupakan metrik evaluasi penting untuk mengukur performa model. Dalam klasifikasi *multiclass* seperti pada penelitian ini, pendekatan yang digunakan adalah *macro-average One-vs-Rest* (OvR), yang menghitung rata-rata AUC dari setiap kelas. Evaluasi metriks ROC dan AUC dilakukan pada dua skenario seleksi fitur, yaitu Mutual Information dan RFE. Evaluasi menggunakan ROC-AUC tidak mencakup seluruh model *stacking* yang dikembangkan, melainkan hanya difokuskan pada model *stacking* terbaik yang diusulkan oleh peneliti.

Model *stacking* tersebut dipilih untuk dianalisis lebih lanjut menggunakan kurva ROC karena pada kedua skenario seleksi fitur, model ini secara konsisten mencatatkan hasil terbaik dalam metrik klasifikasi utama, melampaui model *stacking* lainnya dan sebagian besar model individual. Meskipun demikian, model ini memiliki waktu *inference* terburuk dibandingkan model lainnya. Oleh karena itu, evaluasi melalui kurva ROC AUC diperlukan untuk melihat apakah kompleksitas waktu *inference* tersebut sepadan dengan peningkatan performa klasifikasi secara menyeluruh.

ROC dan AUC Skenario Mutual Information

Berdasarkan Gambar 4.6.37, model *stacking* yang diusulkan peneliti dengan `passthrough=True` dan `passthrough=False` mencatat nilai AUC sebesar 0,9832, yang sangat kompetitif dengan XGBoost (0,9833) dan Random Forest (0,9830). Kurva ROC

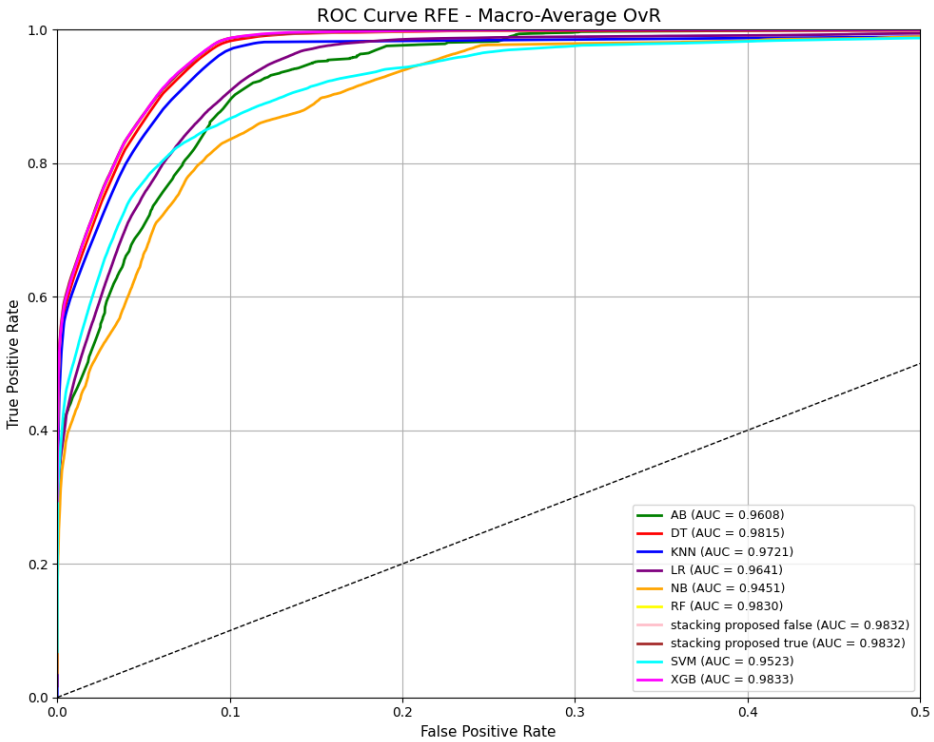
dari keempat model ini berada sangat dekat di area kiri atas grafik, menandakan performa yang sangat tinggi dalam membedakan antara kelas. Di sisi lain, model seperti AdaBoost (0,9624), KNN (0,9717), dan SVM (0,9619) menunjukkan performa yang masih baik namun berada di bawah model *stacking*. Model Naive Bayes dan Logistic Regression memiliki AUC yang paling rendah dibandingkan dengan model lain, masing-masing sebesar 0,9256 dan 0,9367.



Gambar 4.6.37 Kurva ROC dengan Fitur Seleksi MI

ROC dan AUC Skenario RFE

Pada Gambar Gambar 4.6.38, model *stacking* yang diusulkan peneliti menunjukkan stabilitas performa, dengan nilai AUC yang tetap berada di angka 0,9832. Nilai ini hampir sebanding dengan XGBoost (0,9833) dan Random Forest (0,9830), memperlihatkan bahwa model *stacking* tidak hanya unggul dalam *accuracy*, tetapi juga andal dalam membedakan antar kelas meskipun fitur yang digunakan berasal dari proses seleksi berbeda. Sementara itu, sebagian besar model individual lainnya, seperti AdaBoost, KNN, dan SVM, tetap menunjukkan performa AUC yang baik namun tidak setinggi model *ensemble*.



Gambar 4.6.38 Kurva ROC dengan Fitur Seleksi RFE

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Penelitian ini telah mengembangkan model klasifikasi serangan *Distributed Denial of Service* (DDoS) menggunakan pendekatan *ensemble learning* berbasis *stacking* dalam skenario klasifikasi *multiclass*. Model klasifikasi dikembangkan dengan memanfaatkan dataset CIC-DDoS2019 yang terdiri dari 12 kelas, yakni 11 jenis serangan DDoS dan satu kelas *benign* (normal). Proses pengembangan model dilakukan melalui tahapan prepemrosesan data, termasuk pembersihan data, penskalaan, dan seleksi fitur, penyesuaian *hyperparameter*, pengembangan model *ensemble stacking.* serta evaluasi kinerja model.

Dalam tahap seleksi fitur, empat metode digunakan yaitu Mutual Information (MI), Pearson's Correlation, Recursive Feature Elimination (RFE), dan LassoCV. Berdasarkan hasil evaluasi akurasi terhadap delapan algoritma *machine learning*, metode MI dan RFE menunjukkan performa yang paling optimal sehingga dipilih sebagai dasar dalam proses optimasi *hyperparameter* dan evaluasi model. Beberapa model *stacking* yang dikembangkan dalam penelitian ini, baik yang diadaptasi dari studi terdahulu maupun model *stacking* yang diusulkan oleh peneliti sendiri, menunjukkan performa klasifikasi yang kompetitif, bahkan melampaui model individual dan model Random Forest, AdaBoost, serta XGBoost. Model *stacking* usulan peneliti—yang menggabungkan Random Forest, AdaBoost, XGBoost, dan Decision Tree sebagai *base learner*, serta Logistic Regression sebagai *meta learner*—terbukti memberikan hasil terbaik dengan *accuracy* sebesar 0,7965 dan *f1-score* 0,7928 pada skenario fitur Mutual Information (MI), serta *accuracy* 0,7962 dan *f1-score* 0,7925 pada skenario fitur Recursive Feature Elimination (RFE). Namun demikian, peningkatan performa ini disertai dengan kompromi pada aspek efisiensi, khususnya pada *inference time*.

Konfigurasi parameter `passthrough` pada *meta learner* memiliki pengaruh yang bervariasi terhadap kinerja model *stacking*. Pada beberapa kombinasi dan skenario, aktivasi `passthrough=True` dapat meningkatkan performa klasifikasi, sedangkan pada kombinasi lainnya justru menghasilkan penurunan performa. Hal ini menunjukkan bahwa dampak dari parameter `passthrough` sangat bergantung pada kombinasi algoritma *stacking.* Secara keseluruhan, dibandingkan dengan model individual, pendekatan *ensemble stacking* terbukti mampu meningkatkan performa klasifikasi, khususnya dalam menangani klasifikasi *multiclass* yang kompleks, meskipun berdampak pada peningkatan *inference time*.

Pendekatan *ensemble stacking* yang dikembangkan dalam penelitian ini terbukti mampu meningkatkan performa klasifikasi dibandingkan model individual. Namun, hasil evaluasi *confusion matrix* menunjukkan bahwa model *stacking* masih mengalami kesalahan klasifikasi pada beberapa kelas dengan karakteristik data yang mirip. Penggunaan metode seleksi fitur seperti RFE dan MI tetap berisiko menghapus fitur atau atribut yang sebenarnya penting untuk membedakan kelas-kelas tersebut. Kondisi ini membuat model *stacking* kehilangan informasi pembeda yang dibutuhkannya, sehingga akurasi menurun pada pasangan kelas yang sulit dibedakan.

5.2 Saran

Berdasarkan hasil dan temuan penelitian ini, beberapa saran dapat diberikan untuk pengembangan dan penelitian selanjutnya:

1. Penggunaan metode *deep learning*, seperti Recurrent Neural Network (RNN) atau Convolutional Neural Network (CNN), dapat dipertimbangkan untuk meningkatkan kemampuan model dalam mengenali pola lalu lintas jaringan dan mengklasifikasikan serangan dengan lebih efektif, terutama dalam skenario klasifikasi *multiclass*.
2. Dalam penelitian ini, jumlah data yang digunakan dibatasi hingga 30.000 data per kelas melalui proses *random sampling*, mengingat keterbatasan sumber daya komputasi. Oleh karena itu, penelitian mendatang disarankan untuk memanfaatkan seluruh data yang tersedia, atau setidaknya menggunakan jumlah data yang lebih banyak, agar model yang dihasilkan dapat memberikan representasi yang lebih akurat dan kinerja yang lebih optimal.

DAFTAR PUSTAKA

Adedeji, K. B., Abu-Mahfouz, A. M., & Kurien, A. M. (2023). DDoS Attack and Detection Methods in Internet-Enabled Networks: Concept, Research Perspectives, and Challenges. *Journal of Sensor and Actuator Networks 2023, Vol. 12, Page 51*, *12*(4), 51. https://doi.org/10.3390/JSAN12040051

Ahmetoglu, H., & Das, R. (2022). A comprehensive review on detection of cyber-attacks: Data sets, methods, challenges, and future research directions. *Internet of Things*, *20*, 100615. https://doi.org/10.1016/J.IOT.2022.100615

Alasali, T., & Dakkak, O. (2024). A novel DDoS detection method using multi-layer stacking in SDN environment. *Computers and Electrical Engineering*, *120*, 109769. https://doi.org/10.1016/J.COMPELECENG.2024.109769

Alhayani, S., & Murphy, D. R. (2023). A Machine Learning-Based Distributed Denial of Service Detection Approach for Early Warning in Internet Exchange Points. *Computers, Materials and Continua*, *76*(2), 2235–2259. https://doi.org/10.32604/CMC.2023.038003

Arya, A., Kumar, A., & Ahmad, S. S. (2023). DDoS Attack Detection Using Ensemble Machine Learning Approach. *2023 14th International Conference on Computing Communication and Networking Technologies, ICCCNT 2023*. https://doi.org/10.1109/ICCCNT56998.2023.10306750

Badillo, S., Banfai, B., Birzele, F., Davydov, I. I., Hutchinson, L., Kam-Thong, T., … Zhang, J. D. (2020). An Introduction to Machine Learning. *Clinical Pharmacology and Therapeutics*, *107*(4), 871–885. https://doi.org/10.1002/CPT.1796

Bellotti, T., Nouretdinov, I., Yang, M., & Gammerman, A. (2014a). Feature Selection. Dalam *Conformal Prediction for Reliable Machine Learning: Theory, Adaptations and Applications* (hlm. 115–130). Morgan Kaufmann. https://doi.org/10.1016/B978-0-12-398537-8.00006-7

Butt, H. A., Harthy, K. S. Al, Shah, M. A., Hussain, M., Amin, R., & Rehman, M. U. (2024). Enhanced DDoS Detection Using Advanced Machine Learning and Ensemble Techniques in Software Defined Networking. *Computers, Materials and Continua*, *81*(2), 3003–3031. https://doi.org/10.32604/CMC.2024.057185

Chartuni, A., & Márquez, J. (2021). Multi-Classifier of DDoS Attacks in Computer Networks Built on Neural Networks. *Applied Sciences 2021, Vol. 11, Page 10609*, *11*(22), 10609. https://doi.org/10.3390/APP112210609

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *13-17-August-2016*, 785–794. https://doi.org/10.1145/2939672.2939785/SUPPL_FILE/KDD2016_CHEN_BOOSTING_SYSTEM_01-ACM.MP4

Cheng, J., Sun, J., Yao, K., Xu, M., & Cao, Y. (2022). A variable selection method based on mutual information and variance inflation factor. *Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy*, *268*, 120652. https://doi.org/10.1016/J.SAA.2021.120652

Comito, C., & Pizzuti, C. (2022). Artificial intelligence for forecasting and diagnosing COVID-19 pandemic: A focused review. *Artificial Intelligence in Medicine*, *128*, 102286. https://doi.org/10.1016/J.ARTMED.2022.102286

Cover, T. M., & Hart, P. E. (1967). Nearest Neighbor Pattern Classification. *IEEE Transactions on Information Theory*, *13*(1), 21–27. https://doi.org/10.1109/TIT.1967.1053964

Das, S., Venugopal, D., Shiva, S., & Sheldon, F. T. (2020). Empirical Evaluation of the Ensemble Framework for Feature Selection in DDoS Attack. *Proceedings - 2020 7th IEEE International Conference on Cyber Security and Cloud Computing and 2020 6th IEEE International Conference on Edge Computing and Scalable Cloud, CSCloud-EdgeCom 2020*, 56–61. https://doi.org/10.1109/CSCLOUD-EDGECOM49738.2020.00019

DDoS 2019 | Datasets | Research | Canadian Institute for Cybersecurity | UNB. (-a). Diambil 29-06-2025 dari https://www.unb.ca/cic/datasets/ddos-2019.html

DDoS 2019 | Datasets | Research | Canadian Institute for Cybersecurity | UNB. (-b). Diambil 29-06-2025 dari https://www.unb.ca/cic/datasets/ddos-2019.html

Fan, C., Chen, M., Wang, X., Wang, J., & Huang, B. (2021). A Review on Data Preprocessing Techniques Toward Efficient and Reliable Knowledge Discovery From Building Operational Data. *Frontiers in Energy Research*, *9*, 652801. https://doi.org/10.3389/FENRG.2021.652801/BIBTEX

Fathima, A., Devi, G. S., & Faizaanuddin, M. (2023). Improving distributed denial of service attack detection using supervised machine learning. *Measurement: Sensors*, *30*, 100911. https://doi.org/10.1016/J.MEASEN.2023.100911

Ghasemieh, A., Lloyed, A., Bahrami, P., Vajar, P., & Kashef, R. (2023). A novel machine learning model with Stacking Ensemble Learner for predicting emergency readmission of

heart-disease patients. *Decision Analytics Journal*, *7*, 100242. https://doi.org/10.1016/J.DAJOUR.2023.100242

Gong, Y., Liu, G., Xue, Y., Li, R., & Meng, L. (2023). A survey on dataset quality in machine learning. *Information and Software Technology*, *162*, 107268. https://doi.org/10.1016/J.INFSOF.2023.107268

Gupta, S., Grover, D., Alzubi, A. A., Sachdeva, N., Baig, M. W., & Singla6, J. (2022). Machine Learning with Dimensionality Reduction for DDoS Attack Detection. *Computers, Materials and Continua*, *72*(2), 2665–2682. https://doi.org/10.32604/CMC.2022.025048

Hemalatha, N. K., Brunda, R. N., Prakruthi, G. S., Prabhu, B. V. B., Shukla, A., & Narasipura, O. S. J. (2022). Sugarcane leaf disease detection through deep learning. Dalam *Deep Learning for Sustainable Agriculture* (hlm. 297–323). Academic Press. https://doi.org/10.1016/B978-0-323-85214-2.00003-3

Hossain, M. A., & Islam, M. S. (2024). Enhancing DDoS attack detection with hybrid feature selection and ensemble-based classifier: A promising solution for robust cybersecurity. *Measurement: Sensors*, *32*, 101037. https://doi.org/10.1016/J.MEASEN.2024.101037

Kamble, V. H., & Dale, M. P. (2022). Machine learning approach for longitudinal face recognition of children. *Machine Learning for Biometrics: Concepts, Algorithms and Applications*, 1–27. https://doi.org/10.1016/B978-0-323-85209-8.00011-0

Kaur, S., Kumar, K., & Aggarwal, N. (2025). Enhancing DDoS defense in SDN using hierarchical machine learning models. *Journal of Network and Computer Applications*, *239*, 104168. https://doi.org/10.1016/J.JNCA.2025.104168

Kisner, H., Ding, Y., & Thomas, U. (2022). Capacitive material detection with machine learning for robotic grasping applications. *Tactile Sensing, Skill Learning, and Robotic Dexterous Manipulation*, 59–79. https://doi.org/10.1016/B978-0-32-390445-2.00011-8

Kumar, A. I., & Ishigaki, G. (2024). Advanced DDoS Attack Classification using Ensemble Model with Meta-Learner. *Proceedings - International Conference on Computer Communications and Networks, ICCCN*. https://doi.org/10.1109/ICCCN61486.2024.10637575

Lazzarini, R., Tianfield, H., & Charissis, V. (2023). A stacking ensemble of deep learning models for IoT intrusion detection. *Knowledge-Based Systems*, *279*, 110941. https://doi.org/10.1016/J.KNOSYS.2023.110941

Liu, G., Li, X., Guo, Y., Zhang, L., Liu, H., & Ai, H. (2024). Ensemble multiclassification model for predicting developmental toxicity in zebrafish. *Aquatic Toxicology*, *271*, 106936. https://doi.org/10.1016/J.AQUATOX.2024.106936

Maldonado, J., Riff, M. C., & Neveu, B. (2022). A review of recent approaches on wrapper feature selection for intrusion detection. *Expert Systems with Applications*, *198*, 116822. https://doi.org/10.1016/J.ESWA.2022.116822

Mall, R., Abhishek, K., Manimurugan, S., Shankar, A., & Kumar, A. (2023). Stacking ensemble approach for DDoS attack detection in software-defined cyber–physical systems. *Computers and Electrical Engineering*, *107*, 108635. https://doi.org/10.1016/J.COMPELECENG.2023.108635

Messaoud, S., Bradai, A., Bukhari, S. H. R., Quang, P. T. A., Ahmed, O. Ben, & Atri, M. (2020). A survey on machine learning in Internet of Things: Algorithms, strategies, and applications. *Internet of Things*, *12*, 100314. https://doi.org/10.1016/J.IOT.2020.100314

Mienye, I. D., & Sun, Y. (2022). A Survey of Ensemble Learning: Concepts, Algorithms, Applications, and Prospects. *IEEE Access*, *10*, 99129–99149. https://doi.org/10.1109/ACCESS.2022.3207287

Misra, S., & Li, H. (2020). Noninvasive fracture characterization based on the classification of sonic wave travel times. Dalam *Machine Learning for Subsurface Characterization* (hlm. 243–287). Gulf Professional Publishing. https://doi.org/10.1016/B978-0-12-817736-5.00009-0

Moustafa, N. (2021). A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets. *Sustainable Cities and Society*, *72*, 102994. https://doi.org/10.1016/J.SCS.2021.102994

Munagala, V., Thudumu, S., Logothetis, I., Bhandari, S., Vasa, R., & Mouzakis, K. (2024). A comprehensive survey on machine learning applications for drilling and blasting in surface mining. *Machine Learning with Applications*, *15*, 100517. https://doi.org/10.1016/J.MLWA.2023.100517

Najar, A. A., & Manohar Naik, S. (2024). Cyber-Secure SDN: A CNN-Based Approach for Efficient Detection and Mitigation of DDoS attacks. *Computers & Security*, *139*, 103716. https://doi.org/10.1016/J.COSE.2024.103716

Nettleton, D. (2014). Selection of Variables and Factor Derivation. Dalam *Commercial Data Mining* (hlm. 79–104). Morgan Kaufmann. https://doi.org/10.1016/B978-0-12-416602-8.00006-6

Nssibi, M., Manita, G., & Korbaa, O. (2023). Advances in nature-inspired metaheuristic optimization for feature selection problem: A comprehensive survey. *Computer Science Review*, *49*, 100559. https://doi.org/10.1016/J.COSREV.2023.100559

Ogunsanya, M., Isichei, J., & Desai, S. (2023). Grid search hyperparameter tuning in additive manufacturing processes. *Manufacturing Letters*, *35*, 1031–1042. https://doi.org/10.1016/J.MFGLET.2023.08.056

Pachouly, J., Ahirrao, S., Kotecha, K., Selvachandran, G., & Abraham, A. (2022). A systematic literature review on software defect prediction using artificial intelligence: Datasets, Data Validation Methods, Approaches, and Tools. *Engineering Applications of Artificial Intelligence*, *111*, 104773. https://doi.org/10.1016/J.ENGAPPAI.2022.104773

Rafie, A., Moradi, P., & Ghaderzadeh, A. (2023). A Multi-Objective online streaming Multi-Label feature selection using mutual information. *Expert Systems with Applications*, *216*, 119428. https://doi.org/10.1016/J.ESWA.2022.119428

Rani, S. V. J., Ioannou, I., Nagaradjane, P., Christophorou, C., Vassiliou, V., Charan, S., … Pitsillides, A. (2023). Detection of DDoS attacks in D2D communications using machine learning approach. *Computer Communications*, *198*, 32–51. https://doi.org/10.1016/J.COMCOM.2022.11.013

Rodriguez-Galiano, V. F., Luque-Espinar, J. A., Chica-Olmo, M., & Mendes, M. P. (2018). Feature selection approaches for predictive modelling of groundwater nitrate pollution: An evaluation of filters, embedded and wrapper methods. *Science of The Total Environment*, *624*, 661–672. https://doi.org/10.1016/J.SCITOTENV.2017.12.152

Roihan, A., Abas Sunarya, P., & Rafika, A. S. (2020). Pemanfaatan Machine Learning dalam Berbagai Bidang: Review paper. *IJCIT (Indonesian Journal on Computer and Information Technology)*, *5*(1), 75–82. https://doi.org/10.31294/IJCIT.V5I1.7951

Roy, K., Kar, S., & Das, R. N. (2015). Validation of QSAR Models. *Understanding the Basics of QSAR for Applications in Pharmaceutical Sciences and Risk Assessment*, 231–289. https://doi.org/10.1016/B978-0-12-801505-6.00007-7

Sakr, H. A., Fouda, M. M., Ashour, A. F., Abdelhafeez, A., El-Afifi, M. I., & Refaat Abdellah, M. (2024). Machine learning-based detection of DDoS attacks on IoT devices in multi-energy systems. *Egyptian Informatics Journal*, *28*, 100540. https://doi.org/10.1016/J.EIJ.2024.100540

Sammut, C., & Webb, G. I. (Ed.). (2017). *Encyclopedia of Machine Learning and Data Mining*. *Encyclopedia of Machine Learning and Data Mining* (2 ed.). Springer New York, NY. https://doi.org/10.1007/978-1-4899-7687-1

Santosh, K., Das, N., & Ghosh, S. (2022). Introduction. Dalam *Deep Learning Models for Medical Imaging* (hlm. 1–27). Academic Press. https://doi.org/10.1016/B978-0-12-823504-1.00011-8

Shaer, I., & Shami, A. (2023). Data-driven methods for the reduction of energy consumption in warehouses: Use-case driven analysis. *Internet of Things*, *23*, 100882. https://doi.org/10.1016/J.IOT.2023.100882

Sharafaldin, I., Lashkari, A. H., Hakak, S., & Ghorbani, A. A. (2019). Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy. *Proceedings - International Carnahan Conference on Security Technology*, *2019-October*. https://doi.org/10.1109/CCST.2019.8888419

Sharma, V. (2022). A Study on Data Scaling Methods for Machine Learning. *International Journal for Global Academic & Scientific Research*, *1*(1), 31–42. https://doi.org/10.55938/ijgasr.v1i1.4

Sharma, N. V., & Yadav, N. S. (2021). An optimal intrusion detection system using recursive feature elimination and ensemble of classifiers. *Microprocessors and Microsystems*, *85*, 104293. https://doi.org/10.1016/J.MICPRO.2021.104293

Thai, H. T. (2022). Machine learning for structural engineering: A state-of-the-art review. *Structures*, *38*, 448–491. https://doi.org/10.1016/J.ISTRUC.2022.02.003

Thalapala, V., Reddy, A. K., & Guravaiah, K. (2025). MLADAD: Machine Learning Algorithms Analysis on DDoS Attack Detection in SDN-IoT Networks. *Procedia Computer Science*, *260*, 1121–1128. https://doi.org/10.1016/J.PROCS.2025.03.297

Urmi, W. F., Uddin, M. N., Uddin, M. A., Talukder, M. A., Hasan, M. R., Paul, S., … Imran, F. (2024). A stacked ensemble approach to detect cyber attacks based on feature selection techniques. *International Journal of Cognitive Computing in Engineering*, *5*, 316–331. https://doi.org/10.1016/J.IJCCE.2024.07.005

Ye, J., Dobson, S., & McKeever, S. (2012). Situation identification techniques in pervasive computing: A review. *Pervasive and Mobile Computing*, *8*(1), 36–66. https://doi.org/10.1016/J.PMCJ.2011.01.004

Yoachimik, O., & Pacheco, J. (2025, April 27). *Targeted by 20.5 million DDoS attacks, up 358% year-over-year: Cloudflare's 2025 Q1 DDoS Threat Report*. Diambil 29-06-2025 dari https://blog.cloudflare.com/ddos-threat-report-for-2025-q1/

Yuan, Y., Wei, J., Huang, H., Jiao, W., Wang, J., & Chen, H. (2023). Review of resampling techniques for the treatment of imbalanced industrial data classification in equipment condition monitoring. *Engineering Applications of Artificial Intelligence*, *126*, 106911. https://doi.org/10.1016/J.ENGAPPAI.2023.106911

Yüksel, N., Börklü, H. R., Sezer, H. K., & Canyurt, O. E. (2023). Review of artificial intelligence applications in engineering design perspective. *Engineering Applications of Artificial Intelligence*, *118*, 105697. https://doi.org/10.1016/J.ENGAPPAI.2022.105697

Zhao, D. (2019). Comparative analysis of different characteristics of automatic sleep stages. *Computer Methods and Programs in Biomedicine*, *175*, 53–72. https://doi.org/10.1016/J.CMPB.2019.04.004

Zhou, H., Wang, X., & Zhu, R. (2022). Feature selection based on mutual information with correlation coefficient. *Applied Intelligence*, *52*(5), 5457–5474. https://doi.org/10.1007/S10489-021-02524-X/METRICS

LAMPIRAN

1. Kode Program *Random Sampling* Pertama

```
csv_files = glob.glob('D:/J/EnsembleAsli/CICDDoS-Real/CSV-01-12/01-12/*.csv')
selected_files = csv_files[:]
print(selected_files)

label_limit = 50000  # jumlah maksimum per label
label_counts = defaultdict(int)
df_list = []

for file in selected_files:
    print(f"Processing file: {file}")

    # Baca seluruh isi file tanpa chunk
    df = pd.read_csv(file, low_memory=False)

    # Acak data agar setiap baris punya peluang sama
    df = df.sample(frac=1, random_state=42).reset_index(drop=True)

    # Filter baris yang belum mencapai batas jumlah per label
    filtered_df = df[df[' Label'].apply(lambda label: label_counts[label] <
label_limit)]

    # Update counter dan simpan data
    for label in filtered_df[' Label'].unique():
        count_in_df = (filtered_df[' Label'] == label).sum()
        remaining = label_limit - label_counts[label]
        take = min(count_in_df, remaining)
        if take > 0:
            selected_rows = filtered_df[filtered_df[' Label'] ==
label].head(take)
            df_list.append(selected_rows)
            label_counts[label] += take

# Gabungkan semua data
combined_df = pd.concat(df_list, ignore_index=True)
print(combined_df.shape)

# Simpan ke CSV
combined_df.to_csv('RandomSampling_cicddos2019.csv', index=False)
print("File saved as RandomSampling_cicddos2019.csv")
```

2. Kode Program *Random Sampling* Kedua

```
# Jumlah sampel per kelas
n_samples = 30000

# Mengacak seluruh baris DataFrame
df = df.sample(frac=1, random_state=42).reset_index(drop=True)

# Filter label yang jumlah datanya cukup
df_filtered = df[df['Label'].map(df['Label'].value_counts()) >= n_samples]

df_sampled = (
    df_filtered.groupby('Label')
    .apply(lambda x: x.sample(n=n_samples, random_state=42))
    .reset_index(drop=True)
)
```

3. Hasil Perbandingan Fitur 'Fwd Header Length’ dengan 'Fwd Header Length.1’

```
are_equal = (df['Fwd Header Length'] == df['Fwd Header Length.1']).all()
print("Apakah kolom 'x' dan 'x.1' identik?:", are_equal)
```

✓ 0.0s

```
Apakah kolom 'x' dan 'x.1' identik?: True
```

4. Kode Program Fungsi untuk Evaluasi Model berdasarkan *accuracy, precision, recall, f1-score,* dan *confunsion matrix*.

```
def evaluate_model(model, X_test, y_test, name="Model"):

    y_pred = model.predict(X_test)

    # Hitung metrik evaluasi
    acc = accuracy_score(y_test, y_pred.round())
    prec = precision_score(y_test, y_pred.round(), average='macro',
zero_division=0)
    rec = recall_score(y_test, y_pred.round(), average='macro', zero_division=0)
    f1 = f1_score(y_test, y_pred.round(), average='macro', zero_division=0)

    print(f"accuracy: {acc}")
    print(f"precision: {prec}")
    print(f"recall: {rec}")
    print(f"f1-score: {f1}")

    # Confusion matrix
    cm = confusion_matrix(y_test, y_pred)
    disp = ConfusionMatrixDisplay(confusion_matrix=cm)
    disp.plot(cmap=plt.cm.Blues)
    plt.title(f'Confusion Matrix - {name}')
    plt.show()
```