*T*imur Tengah

Menghadapi Tantangan Keamanan Era Digital: Pendekatan Pembelajaran Mesin untuk Analisis Lalu Lintas Jaringan

Anthony Edbert Feriyanto, Filbert Aurelian Tjiaranata, M. Arvin Wijayanto

{anthony.edbert, filbert.aurelian, m.arvin} @ ui.ac.id
2306165654, Fakultas Ilmu Komputer, Universitas Indonesia, Depok, 16424
2306152336, Fakultas Ilmu Komputer, Universitas Indonesia, Depok, 16424
2306259780, Fakultas Ilmu Komputer, Universitas Indonesia, Depok, 16424
Corresponding Author: Anthony Edbert Feriyanto

INTISARI

Dalam era digital yang semakin maju, keamanan siber menjadi sektor vital yang harus terus dijaga. Salah satu tantangan utama adalah menangani serangan lalu lintas jaringan yang beragam dan canggih. Penelitian ini memanfaatkan *train dataset* yang mengandung enam label terkait lalu lintas, dengan distribusi data yang bersifat *imbalace*, di mana kategori mayoritas pada data adalah *Benign* dan *Background*. Untuk mengatasi masalah ini, kami menggunakan model *Gradient Boosting* CatBoost yang telah melalui proses *fine-tuning* dan penerapan *auto-weighting* untuk menangani ketidakseimbangan data. Evaluasi dilakukan menggunakan metrik campuran dengan bobot 0.5 untuk *balance accuracy score* dan 0.5 untuk *accuracy score*, menghasilkan skor 0.87270 di *private leaderboard* dan 0.87257 di *public leaderboard*. Hasil penelitian ini menunjukkan efektivitas pendekatan pembelajaran mesin dalam meningkatkan keamanan jaringan dengan analisis lalu lintas yang lebih akurat dan adaptif, memberikan kontribusi signifikan terhadap perlindungan jaringan di era digital.

KATA KUNCI

network attack detection, *class imbalance*, *machine learning*, *cybersecurity*, *traffic analysis*.

I. PENDAHULUAN

Pertumbuhan penggunaan internet global telah mencapai tingkat yang belum pernah terjadi sebelumnya. Menurut laporan Cisco Annual Internet Report, jumlah pengguna internet diproyeksikan mencapai $5,3$ miliar pada tahun 2023, meningkat dari $3,9$ miliar pada tahun 2018. Seiring dengan pertumbuhan ini, volume lalu lintas data global diperkirakan akan mencapai 397 exabytes per bulan pada 2022, naik drastis dari 122 exabytes per bulan pada 2017. Namun, peningkatan konektivitas ini juga membawa risiko keamanan yang signifikan. Laporan dari Cybersecurity Ventures memproyeksikan bahwa kejahatan siber akan menyebabkan kerugian global sebesar $\$10,5$ triliun per tahun pada 2025, meningkat dari \$3 triliun pada 2015. Angka-angka ini menunjukkan urgensi dalam meningkatkan keamanan jaringan di era digital.

Tantangan utama dalam mengamankan jaringan modern terletak pada kompleksitas dan kecepatan evolusi ancaman siber. Menurut laporan IBM X-Force Threat Intelligence Index, lebih dari $8,5$ miliar catatan data terekspos pada tahun 2019 saja,

dengan 71% serangan yang berhasil dimotivasi secara finansial. Sementara itu, studi dari Ponemon Institute mengungkapkan bahwa rata-rata waktu yang dibutuhkan untuk mengidentifikasi dan mengatasi pelanggaran data adalah 280 hari, menunjukkan kesenjangan signifikan dalam kemampuan deteksi dan respons. Keragaman serangan, dari *phishing* canggih hingga *malware* yang terus berkembang, membuat pendekatan keamanan tradisional semakin tidak memadai. Terlebih lagi, munculnya teknologi seperti *Internet of Things* (IoT) dan 5G menambah lapisan kompleksitas baru, dengan Gartner memperkirakan akan ada 25 miliar perangkat IoT terhubung pada tahun 2021.

Menghadapi lanskap ancaman yang dinamis ini, pengembangan sistem deteksi dan klasifikasi lalu lintas jaringan yang lebih canggih menjadi kebutuhan mendesak. Tujuannya adalah menciptakan model prediktif yang dapat menganalisis volume data besar secara real-time, dengan kemampuan untuk mendeteksi anomali dan mengklasifikasikan berbagai jenis lalu lintas dengan akurasi tinggi. Menurut studi yang diterbitkan dalam Journal of Network and Computer Applications, penerapan teknik pembelajaran mesin lanjutan dalam deteksi intrusi jaringan telah menunjukkan peningkatan akurasi hingga 99, 9% dalam beberapa kasus, sambil mengurangi tingkat *false positives* secara signifikan. Dengan mengintegrasikan analitik prediktif dan kecerdasan buatan, Penyedia jaringan dapat meningkatkan kemampuan mereka untuk mengidentifikasi dan merespons ancaman yang muncul secara proaktif, potensial mengurangi waktu deteksi dari hari menjadi menit, dan pada akhirnya memperkuat postur keamanan siber mereka secara keseluruhan.

II. Analisis dan Pembahasan

Bagian Analisis dan Pembahasan ini merupakan bagian terpenting dalam penelitian ini. Pada bagian ini, dibahas keseluruhan mengenai pemahaman data dan uji coba prediksi. Adapun sub-bagian yang termasuk antara lain dimulai dari deskripsi dataset, *Exploratory Data Analysis*, *Feature Engineering*, *Modelling*, Hasil dan Analisis, serta Kesimpulan.

A. Deskripsi Dataset

Dataset yang digunakan dalam kompetisi ini merupakan *dataset* mengenai *traffic* internet yang tercatat di suatu situs web tertentu. Setiap baris pada data memliki fitur-fitur yang mengidentifikasikan sebuah *traffic* yang terjadi. Fitur-fitur ini saling memberikan pengaruh terhadap jenis aliran trafik tersebut. Adapun penjelasan lebih lanjut mengenai makna dari setiap fitur dibahas dalam tabel 1 di halaman berikutnya.

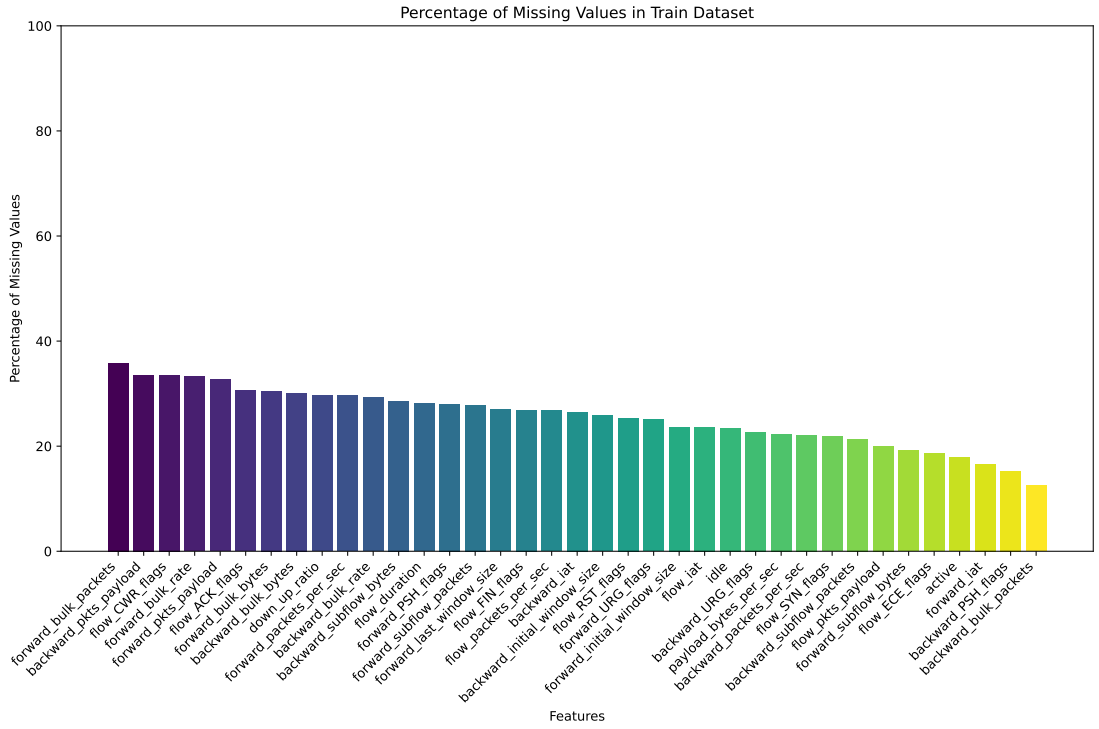
Dalam kompetisi ini, dataset dipecah menjadi 2 bagian, *train* dan *test dataset*. *Train dataset* terdiri dari 416.473 baris data, sedangkan untuk *test dataset* terdiri dari 138.805 baris data. Kedua dataset memiliki total 42 fitur, dengan *train dataset* memiliki 1 kolom tambahan, yakni '*traffic*' sebagai label. Label '*traffic*' adalah identifikasi kategorikal tipe *traffic* yang terjadi. Penjelasan dan distribusi setiap tipe '*traffic*' dijelaskan dalam tabel 2 berikut.

B. Exploratory Data Analysis

1) **Tipe Data**.
   Langkah paling awal dalam melakukan penelitian terkait dataset adalah memastikan tipe data dari setiap kolom sudah sesuai dan konsisten. Kami melakukan pengecekan tipe data dari seluruh 42 fitur dalam dataset. Ditemukan bahwa terdapat 3 fitur yang memiliki tipe data `categorical`, yakni `id`, `origin_host`, dan `response_host`. Kemudian didapati juga bahwa ada 2 fitur yang bertipe data `int64`, yaitu `origin_port` dan `response_port`. Selain kelima fitur tersebut, 37 fitur sisanya memiliki tipe data seragam `float64`. Jika diperhatikan lebih lanjut, fitur-fitur yang tidak memiliki tipe data

Fitur	Penjelasan
id	ID untuk setiap aliran jaringan
origin_host	Alamat IP client
origin_port	Nomor port yang digunakan oleh client
response_host	Alamat IP server
response_port	Nomor port yang digunakan oleh server
flow_duration	Durasi aliran jaringan
forward_packets_per_sec	Laju pengiriman paket dari client ke server per detik
backward_packets_per_sec	Laju pengiriman paket dari server kembali ke client per detik
flow_packets_per_sec	Laju keseluruhan paket yang dikirim per detik dalam seluruh aliran
down_up_ratio	Rasio antara jumlah paket atau byte yang diunduh dan diunggah
flow_FIN_flags	Jumlah bendera FIN yang menandakan akhir dari koneksi TCP
flow_SYN_flags	Jumlah bendera SYN yang digunakan untuk memulai koneksi TCP
flow_RST_flags	Jumlah bendera RST yang menandakan reset koneksi TCP
forward_PSH_flags	Jumlah bendera PSH yang menandakan data harus segera diproses oleh server
backward_PSH_flags	Jumlah bendera PSH yang menandakan data harus segera diproses oleh client
flow_ACK_flags	Jumlah bendera ACK yang digunakan untuk mengonfirmasi penerimaan paket
forward_URG_flags	Jumlah bendera URG yang menandakan data mendesak dari client
backward_URG_flags	Jumlah bendera URG yang menandakan data mendesak dari server
flow_CWR_flags	Jumlah bendera CWR yang menandakan pengurangan ukuran jendela kongesti
flow_ECE_flags	Jumlah bendera ECE yang menunjukkan adanya kemacetan jaringan
forward_pkts_payload	Ukuran rata-rata payload dalam paket dari client ke server
backward_pkts_payload	Ukuran rata-rata payload dalam paket dari server ke client
flow_pkts_payload	Ukuran rata-rata payload dalam seluruh aliran, mencakup paket dari client dan server
forward_iat	Rata-rata waktu antar kedatangan (Inter Arrival Time, IAT) antara paket dari client ke server
backward_iat	Rata-rata waktu antar kedatangan (IAT) antara paket dari server ke client
flow_iat	Rata-rata waktu antar kedatangan (IAT) dalam seluruh aliran, mencakup paket dari client dan server
payload_bytes_per_sec	Laju transfer byte payload per detik dalam aliran jaringan
forward_subflow_packets	Jumlah paket dalam subflow dari client ke server
backward_subflow_packets	Jumlah paket dalam subflow dari server ke client
forward_subflow_bytes	Jumlah byte dalam subflow dari client ke server
backward_subflow_bytes	Jumlah byte dalam subflow dari server ke client
forward_bulk_bytes	Jumlah byte bulk dalam aliran dari client ke server
backward_bulk_bytes	Jumlah byte bulk dalam aliran dari server ke client
forward_bulk_packets	Jumlah paket bulk dalam aliran dari client ke server
backward_bulk_packets	Jumlah paket bulk dalam aliran dari server ke client
forward_bulk_rate	Laju lalu lintas bulk dalam aliran dari client ke server
backward_bulk_rate	Laju lalu lintas bulk dalam aliran dari server ke client
active	Rata-rata waktu aktif dalam aliran, menggambarkan periode ketika data sedang ditransmisikan
idle	Rata-rata waktu idle dalam aliran, menggambarkan periode ketika tidak ada data yang ditransmisikan
forward_initial_window_size	Ukuran Window Flow Control yang tersedia dari client ke server pada awal koneksi
backward_initial_window_size	Ukuran Window Flow Control yang tersedia dari server ke client pada awal koneksi
forward_last_window_size	Ukuran Window Flow Control dari client ke server pada akhir koneksi

Tabel 1: Tabel Fitur dan Penjelasan Aliran Jaringan

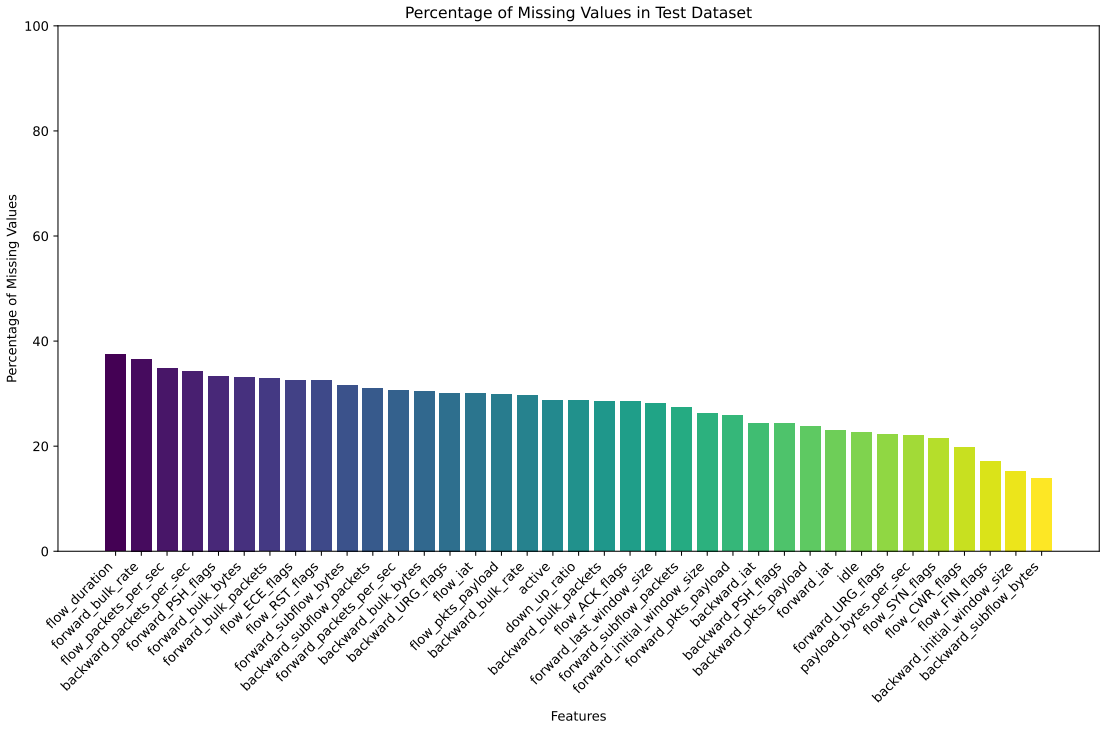


Gambar 1: Missing Values pada Train Dataset

`float64` adalah fitur-fitur yang berpengaruh penting terhadap identitas data. Fitur id adalah nilai unik yang berfungsi sebagai pembeda satu row dengan lainnya. Sedangkan fitur `origin_host`, `response_host`, `origin_port`, dan `response_port` merupakan identitas dari pengirim dan penerima aliran trafik.

2) **Missing Values**.
Identifikasi missing values merupakan tahapan selanjutnya dari penelitian kami. Hal ini penting untuk memastikan bahwa data yang akan digunakan dalam proses modelling adalah data yang representatif dan berkualitas tinggi. Oleh karena itu, kami mengawali dengan analisis setiap kolom dalam dataset train dan test untuk menentukan persentase missing values dari setiap kolom tersebut.
Ternyata, hampir keseluruhan fitur memiliki missing values yang cenderung besar. Bahkan beberapa fitur kehilangan hingga 33 persen atau sepertiga dari banyak datanya. Besarnya persentase *missing values* pada semua kolom di kedua dataset, membuat kami perlu melakukan analisis lebih lanjut terutama dalam merancang strategi imputasi yang sesuai dengan karakteristik masing-masing kolom.

3) **Ketidakseimbangan pada Kelas Label**. Distribusi kelas label pada dataset tidak luput dari pengamatan kami. Sebagai alat visualisasi, Histoplot digunakan untuk menggambarkan distribusi kelas label seperti terlihat pada gambar 3. Plot tersebut menunjukkan bahwa dataset ini bersifat *imbalance* dimana frekuensi kemunculan kelas label tidak merata dengan kelas label lainnya. Label `Benign` terlihat memiliki frekuensi kemunculan yang paling banyak, diikuti dengan label `Background`. Kedua kelas label tersebut termasuk dalam klasifikasi aliran trafaik normal. Adapun kelas label yang termasuk dalam aliran trafik berbahaya seperti `Probing`, `Bruteforce`, `Bruteforce-XML`, dan `XMRIGCC CryptoMiner` cenderung lebih sedikit kemunculannya. Melihat dari tipe aliran trafik, ketidakseimbangan label ini sebenarnya sesuai dengan keadaan



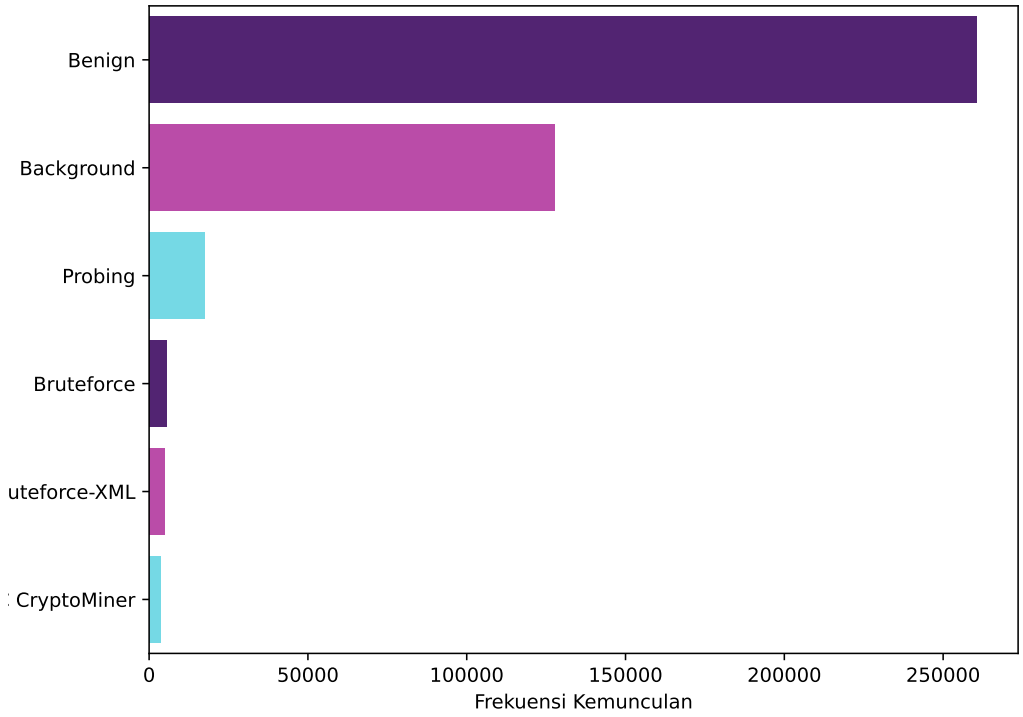
Gambar 2: Missing Values pada Test Dataset

Jenis Trafik	**Penjelasan**
Background	Lalu lintas rutin atau latar belakang yang biasanya tidak menimbulkan ancaman.
Benign	Traffic yang tidak menunjukkan tanda-tanda aktivitas berbahaya atau anomali.
Probing	Aktivitas pemindaian atau penjelajahan untuk menemukan kerentanan atau target.
Bruteforce	Upaya penyerangan dengan mencoba berbagai kombinasi kata sandi untuk mendapatkan akses.
XMRIGCC CryptoMiner	Aktivitas terkait dengan penambangan cryptocurrency secara tersembunyi.
Bruteforce-XML	Serangan brute force yang menargetkan aplikasi berbasis XML.

Tabel 2: Tabel Jenis Trafik dan Penjelasan

pada dunia nyata dimana jenis aliran trafik berbahaya muncul lebih sedikit dibandingkan aliran trafik normal. Akan tetapi, ketidakseimbangan kelas label ini dapat menyebabkan model memiliki bias lebih kepada kelas mayoritas, sehingga harus diterapkan metode untuk mengatasi masalah ketidakseimbangan kelas label tersebut.

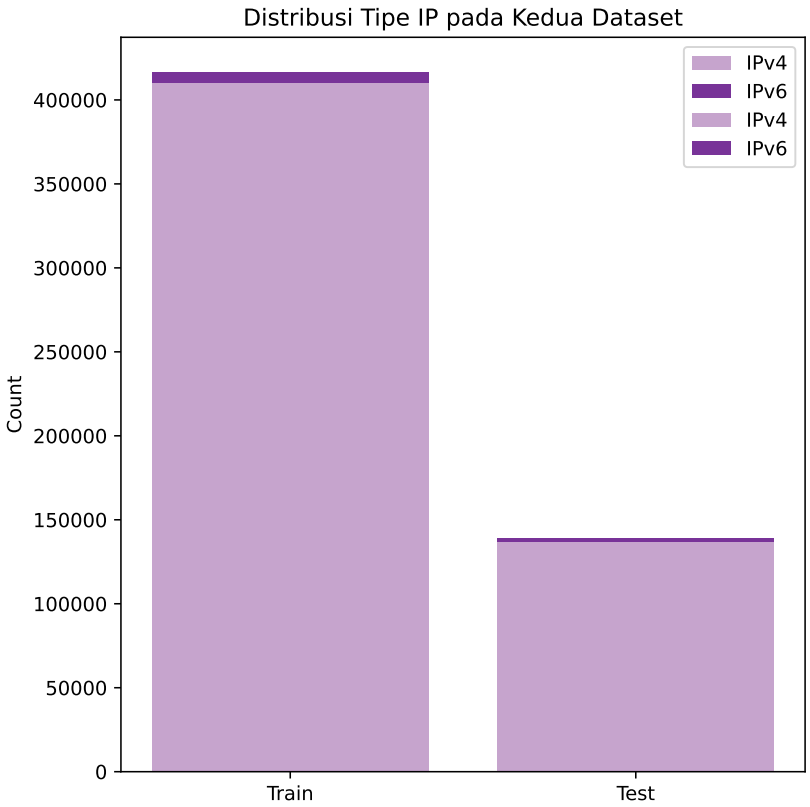
4) **Alamat IP**.
Pada penelitian ini, kami melakukan eksplorasi yang sangat banyak dan mendalam terhadap kolom alamat IP. Mulai dari analisis distribusi dan karakteristik dari alamat IP server maupun client, hingga analisis pola / pattern yang lebih rumit. Pertama, hasil eksplorasi data yang kami lakukan terhadap kolom-kolom alamat IP mengungkapkan adanya dua tipe IP yang berbeda pada dataset, yakni IPv4 dan IPv6.



Gambar 3: Distribusi Label pada Train Dataset

Memahami pentingnya keberadaan dua tipe alamat IP ini mampu memberikan konteks dan informasi yang lebih luas lagi terhadap data yang dimiliki, serta dapat menjadi awal yang krusial untuk proses feature engineering nantinya. IPv4 memiliki ruang alamat 32-bit dengan 4 blok utama dalam alamatnya dan memiliki '.' sebagai pemisah block. Sedangkan, IPv6 memiliki ruang alamat 128-bit serta memiliki ':' sebagai pemisah block. Hal ini membuat perlakuan serta proses pembuatan fitur terhadap alamat IP nantinya juga harus menyesuaikan dengan perbedaan-perbedaan yang dimiliki oleh kedua tipe alamat IP. Dari visualisasi distribusi tipe alamat IP di kedua dataset pelatihan dan pengujian, terlihat bahwa jumlah alamat IPv4 jauh lebih banyak dibandingkan dengan IPv6. Fenomena ini menandakan lagi-lagi adanya ketidakseimbangan signifikan antara dua tipe IP, yang dapat mempengaruhi model prediksi dalam beberapa cara. Hal ini memberikan insight bahwa ada kemungkinan nantinya model akanmenjadi bias ke arah alamat IPv4 karena keberlimpahan datanya, potensial mengurangi akurasi ketika berhadapan dengan alamat IPv6, atau sebaliknya yakni model akan menjadi bias ketika berhadapan dengan alamat IPv6 dikarenakan sample data yang sangat sedikit. Menghadapi situasi ini, ada langkah yang mungkin bisa dilakukan untuk membantu model prediksi dan menyesuaikan analisis lebih lanjut adalah teknik seperti resampling untuk menyeimbangkan jumlah sampel dari kedua tipe IP dalam pelatihan model, baik dengan oversampling pada data IPv6 atau melakukan undersampling IPv4.

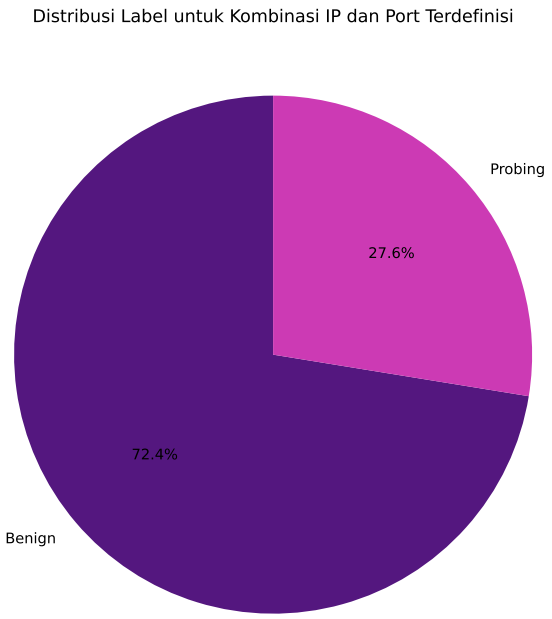
5) **Keunikan Dataset**.
Setelah menyelesaikan EDA dataset secara umum, kami berusaha untuk melakukan analisis lebih mendalam terkait dataset. Kami mencoba mencari keunikan ataupun pola yang bisa saja terbentuk dalam dataset. Setelah penelitian, akhirnya terdapat



Gambar 4: Distribusi Tipe Alamat IP pada Train Dataset

beberapa hal unik yang berhasil kami temukan. Hal unik ini dapat dimanfaatkan untuk memberikan pemaparan insight yang lebih menarik dan meningkatkan performa metrik skor.

a) **Sang 'Pelaku'**. Terdapat enam jenis aliran trafik yang dapat terjadi: `Background, Benign, Probing, Bruteforce, XMRIGCC CryptoMiner,` dan `Bruteforce-XML`. Untuk setiap jenis aliran tersebut, kami mencari tahu dalang dibalik terjadinya aliran dengan cara mengecek `origin_host`. Ternyata, ditemukan bahwa beberapa kelas label hanya memiliki satu jenis `origin_host` saja. Satu `origin_host` unik ini bertanggungjawab atas 'perilaku'-nya dalam melakukan kejahatan siber.
Dalam penelitian ini, kami mengidentifikasi dan menganalisis pola traffic jaringan yang spesifik, yang menghubungkan origin_host '103.255.15.150' dengan response_host '128.199.242.104' pada response_port 443. Analisis awal menunjukkan bahwa sekitar 73% dari traffic ini diklasifikasikan sebagai "benign", sementara sisanya, yang berpotensi sebagai "probing", mencapai 27%. Temuan ini didukung oleh visualisasi data menggunakan pie chart5, yang memperlihatkan distribusi label *traffic* secara jelas. Berdasarkan temuan ini, langkah selanjutnya dalam penelitian kami adalah melaksanakan serangkaian proses feature engineering yang lebih mendalam. Kami berencana untuk memanfaatkan spesifikasi host dan port yang telah teridentifikasi sebagai faktor kunci dalam model prediktif kami. Tujuan utamanya adalah untuk mengembangkan algoritma yang dapat secara efektif membedakan antara traffic 'benign' dan 'probing' dalam skenario jaringan yang serupa. Dalam tahapan feature engineering, kami akan mengeksplorasi berbagai metrik yang berhubungan dengan durasi sesi, frekuensi paket, dan pola lainnya yang mungkin memberikan indikasi awal adanya upaya probing



Gambar 5: Distribusi Label Traffic untuk Kombinasi Host dan Port Terdefinisi

atau serangan.

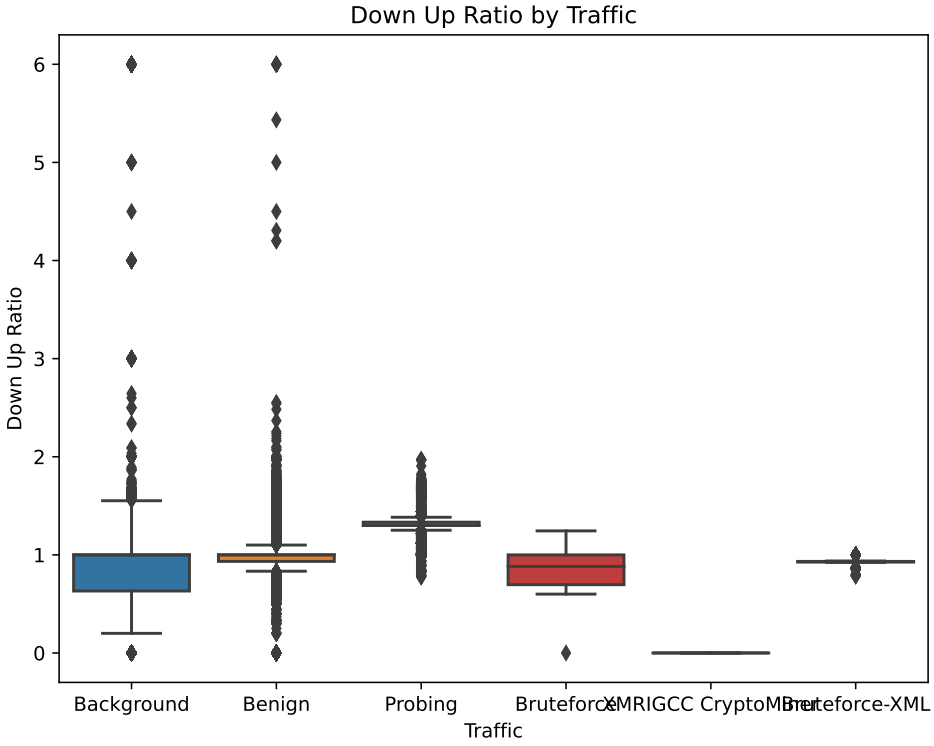
b) **Ciri Khas Fitur pada Setiap Kelas**.

i) **Down Up Ratio Feature**.

Pada fitur *down up ratio*, terdapat persebaran nilai yang cukup mencolok di antara kategori `Background` dan `Benign`. Kedua kategori ini menunjukkan distribusi nilai yang cukup luas, dengan banyak titik data yang bersifat sebagai outlier. Hal ini menunjukkan bahwa pada *down up ratio*, wajar jika terjadi lonjakan nilai yang signifikan. Lonjakan ini mungkin disebabkan oleh variasi aktivitas jaringan yang normal dan latar belakang, di mana fluktuasi dalam *download* dan *upload* data sering terjadi.

Sementara itu, pada label `Probing`, persebaran nilai outlier mulai lebih merapat dan berjarak sekitar 1 unit dari pusat data, menunjukkan konsistensi yang lebih tinggi dibandingkan dengan kategori sebelumnya. Hal ini dapat mengindikasikan pola perilaku yang lebih teratur dalam aktivitas *probing*, di mana fluktuasi rasio *down up* lebih terkendali.

Pada label `Bruteforce` dan `Bruteforce XML`, mayoritas nilai dari *down up ratio* bernilai 1, menunjukkan kestabilan dalam rasio tersebut untuk kategori ini. Hal ini mungkin disebabkan oleh pola serangan yang khas dari *bruteforce* yang biasanya melibatkan sejumlah permintaan yang hampir seimbang antara *download* dan *upload* data. Terakhir, untuk label `XMRIGCC Cryptominer`, fitur *down up ratio* memiliki nilai 0 di semua titik datanya, menunjukkan tidak adanya variasi atau aktivitas yang terdeteksi dalam rasio ini untuk kategori tersebut. Hal ini bisa



Gambar 6: Box Plot Down Up Ratio

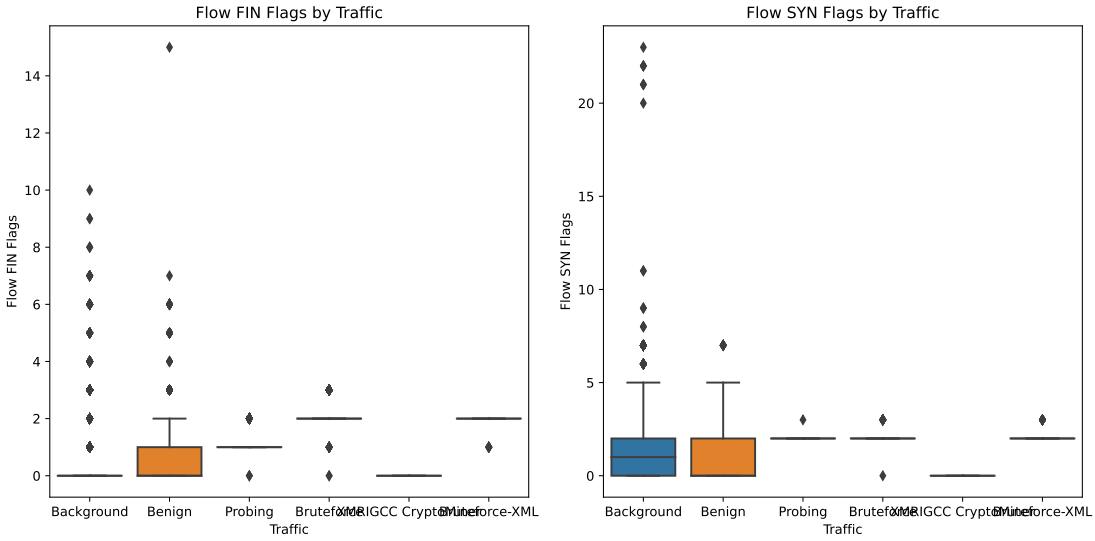
terjadi karena aktivitas *cryptomining* mungkin memiliki pola lalu lintas jaringan yang tidak terdeteksi oleh *down up ratio* atau mungkin karena *cryptominer* tidak menghasilkan aktivitas jaringan yang signifikan dalam konteks rasio *down up*.

ii) **Flow FIN dan SYN Flags Feature**.

Pada *FIN* dan *SYN flags* memiliki distribusi nilai yang cenderung mirip. Label *Benign* dan *Background* memiliki nilai yang tersebar dan dapat terlihat dari banyaknya titik-titik *outlier* pada *box plot*. Walaupun median dari kedua label ini adalah 0, hal ini menunjukkan bahwa meskipun sebagian besar aliran memiliki nilai FIN dan SYN yang rendah atau tidak ada sama sekali, terdapat beberapa aliran yang mengalami lonjakan nilai, mungkin akibat aktivitas jaringan yang normal seperti penutupan dan pembukaan koneksi TCP yang tidak teratur. Hal ini dapat mencerminkan lalu lintas jaringan yang sah tetapi memiliki pola variabilitas yang tinggi, seperti pembaruan otomatis atau komunikasi dengan server yang memiliki waktu respons yang berbeda.

Selain itu, *Bruteforce* dan *Bruteforce XML* memiliki karakteristik yang serupa dengan distribusi nilai yang lebih terkonsentrasi. Mayoritas nilai dari *flow_FIN_flags* dan *flow_SYN_flags* pada label ini cenderung lebih tinggi dibandingkan dengan label lain, yang dapat mengindikasikan adanya percobaan koneksi yang agresif ke berbagai port. Hal ini sejalan dengan sifat serangan *bruteforce*, di mana terjadi banyak upaya pembukaan koneksi (ditandai dengan SYN flags) dan kemungkinan penutupan atau kegagalan koneksi (ditandai dengan FIN flags).

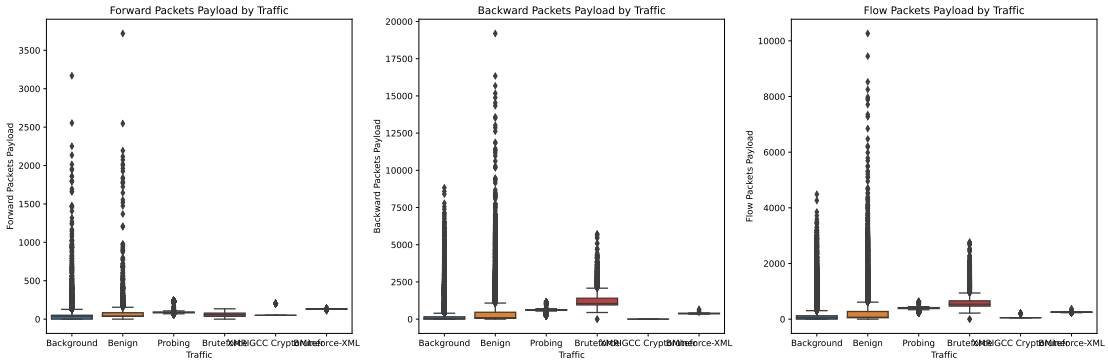
Sedangkan untuk label *XMRIGCC Cryptominer*, keseluruhan nilai adalah 0 untuk *flow_FIN_flags* dan *flow_SYN_flags*.



Gambar 7: Box Plot Flow FIN SYN Flags

Hal ini menunjukkan bahwa aktivitas jaringan dari label ini tidak melibatkan pembukaan dan penutupan koneksi TCP yang eksplisit atau mengindikasikan bahwa koneksi yang digunakan bersifat persisten dan stabil, tanpa adanya fluktuasi yang biasanya terlihat pada aliran jaringan yang lebih dinamis. Dalam konteks *cryptomining*, koneksi yang stabil dan persisten diperlukan untuk berkomunikasi dengan *pool mining* atau server pusat, sehingga sangat masuk akal jika tidak terdapat variasi nilai pada *FIN* dan *SYN flags*.

iii) **Forward Packets Payload, Backward Packets Payload, Flow Packets Payload Feature**.
Dapat dilihat berdasarkan gambar *Box Plot Packets Payload*, label *benign* memiliki persebaran nilai yang cukup tinggi. Hal ini terjadi karena *benign* merupakan label dengan jumlah *outlier* yang paling banyak dibandingkan dengan label lainnya. Persebaran yang luas menunjukkan bahwa aktivitas jaringan yang tergolong *benign* memiliki keragaman dalam ukuran paket yang dikirim, kemungkinan disebabkan oleh variasi dalam jenis dan tujuan lalu lintas yang sah. Selain itu, label *Background* juga memiliki *outlier* tetapi jumlahnya lebih sedikit dibandingkan dengan label *benign*, menunjukkan variasi yang lebih rendah dalam lalu lintas jaringan yang tidak diketahui tujuannya.
Label lainnya, seperti *probing*, *XMRIGCC Cryptominer*, dan *Bruteforce XML*, mayoritas memiliki nilai 0 di ketiga fitur tersebut: `forward_packets_payload`, `backward_packets_payload`, dan `flow_packets_payload`. Hal ini mengindikasikan bahwa untuk aktivitas yang termasuk dalam kategori ini, umumnya tidak terdapat data muatan (*payload*) yang signifikan dalam paket yang dikirimkan, atau aktivitas tersebut tidak melibatkan pertukaran data dalam jumlah besar. Misalnya, pada *probing*, fokusnya adalah untuk memetakan jaringan dan mengidentifikasi titik lemah, sehingga biasanya tidak melibatkan pertukaran data dalam jumlah besar. Begitu juga dengan *cryptomining*, di mana koneksi yang digunakan cenderung persisten dan stabil tanpa fluktuasi yang berarti dalam *payload*.
Namun, analisis lebih lanjut pada fitur `backward_packets_payload` dan `flow_packets_payload` dapat membantu membedakan antara *bruteforce* dan *Bruteforce XML*. Pada label *Bruteforce*, nilai `forward_packets_payload` cenderung tidak 0 dan masih memiliki *outlier* yang jaraknya tidak terlalu jauh dari data yang representatif dari aktivitas *bruteforce*. Hal ini menunjukkan bahwa serangan *bruteforce* melibatkan pengiriman data yang lebih variatif ke arah server, mungkin berupa berbagai kombinasi nama pengguna dan kata sandi yang dicoba. Di sisi lain, pada



Gambar 8: Box Plot Packets Payload

Bruteforce XML, nilai `forward_packets_payload` dan `backward_packets_payload` terpusat pada 0, menunjukkan bahwa aktivitas ini lebih bersifat statis dan tidak melibatkan pertukaran data *payload* yang besar. Ini mungkin karena *bruteforce* yang menggunakan *XML* lebih terstruktur dan kemungkinan besar menggunakan permintaan standar dengan ukuran data yang kecil atau bahkan tanpa data *payload* yang signifikan. Sedangkan `flow_packets_payload` menunjukkan pola yang sama, di mana *Bruteforce XML* cenderung memiliki nilai 0, mengindikasikan aliran data yang lebih konsisten dan terstruktur.

6) ***Correlation Analysis***.

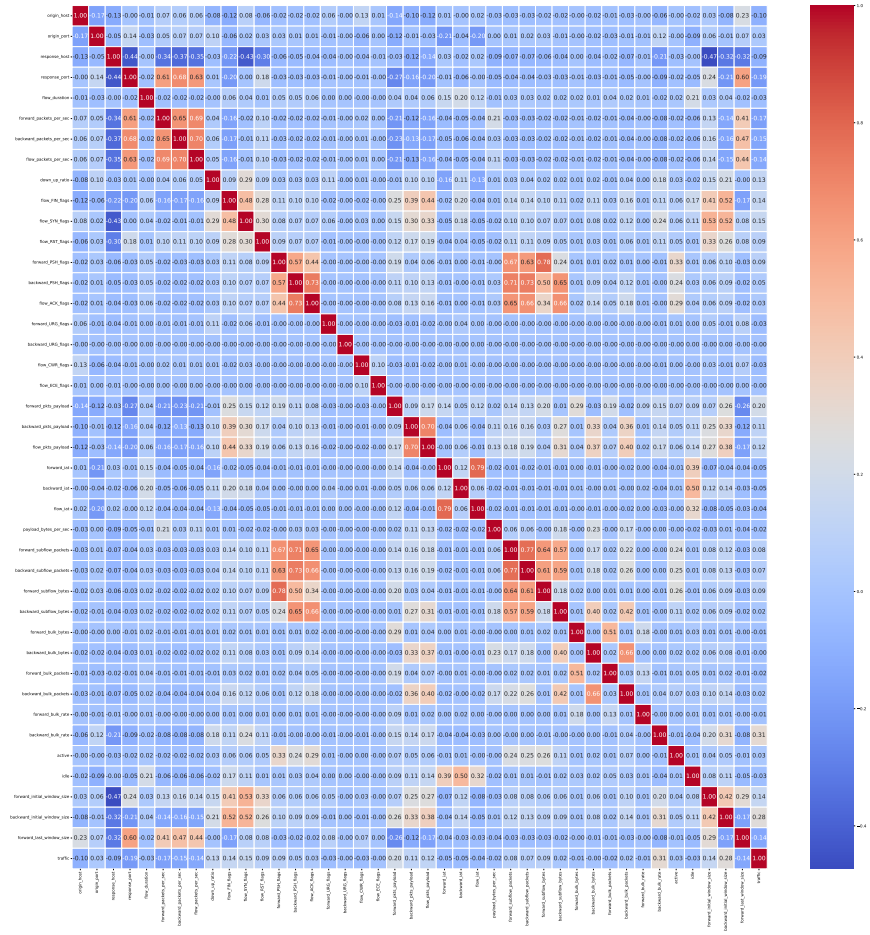
Size of Correlation	Interpretation
0.90 to 1.00 / -0.90 to -1.00	Very high positive / negative correlation
0.70 to 0.90 / -0.70 to -0.90	High positive / negative correlation
0.50 to 0.70 / -0.50 to -0.70	Moderate positive / negative correlation
0.30 to 0.50 / -0.30 to -0.50	Low positive / negative correlation
0.00 to 0.30 / -0.00 to -0.30	Very low positive / negative correlation

Tabel 3: Interpretation of Correlation Coefficient

Berdasarkan gambar 9 dan tabel 3 yang disajikan, terlihat bahwa tidak ada korelasi antar fitur yang berada dalam kisaran 0.9 hingga 1.0, yang menunjukkan bahwa tidak ada fitur yang memiliki hubungan sangat kuat satu sama lain. Namun, terdapat beberapa pasangan fitur yang menunjukkan korelasi kuat, yaitu dalam kisaran 0.7 hingga 0.9. Pasangan-pasangan fitur ini meliputi *flow_packets_per_sec* dan *backward_packets_per_sec*, *forward_subflow_bytes* dan *forward_PSH_flags*, *flow_ACK_flags* dan *backward_PSH_flags*, *forward_subflow_packets* dan *backward_PSH_flags*, serta *backward_subflow_bytes* dan *backward_PSH_flags*.

Korelasi antara *flow_packets_per_sec* dan *backward_packets_per_sec* dapat dijelaskan dengan melihat bahwa kedua fitur ini menggambarkan jumlah paket yang dikirim dalam arah yang berbeda ke depan dan ke belakang. Ketika aliran data mengandung banyak paket yang dikirim dalam satu arah, kemungkinan besar juga akan ada banyak paket yang dikirim dalam arah sebaliknya. Ini menjelaskan mengapa kedua fitur ini dapat memiliki korelasi yang kuat[9].

Korelasi antara *forward_subflow_bytes* dan *forward_PSH_flags* dapat dijelaskan dengan *forward_subflow_bytes* menunjukkan jumlah byte yang dikirim dalam aliran yang diarahkan ke depan, sedangkan *forward_PSH_flags* menunjukkan frekuensi di mana flag PUSH diatur pada paket. Jika lebih banyak data dikirim dalam aliran ke depan, maka kemungkinan ada lebih banyak paket yang memerlukan PUSH flag untuk mempercepat pengiriman data. Hal ini menjelaskan mengapa kedua fitur



Gambar 9: Korelasi antar Fitur pada Train Dataset

ini mungkin memiliki korelasi tinggi[7].

Selain itu, korelasi antara *flow_ACK_flags* dan *backward_PSH_flags* dapat dijelaskan dengan *flow_ACK_flags* menunjukkan jumlah pengakuan (ACK) dalam aliran, sementara *backward_PSH_flags* menunjukkan frekuensi flag PUSH diatur pada paket yang dikirim ke belakang. Penggunaan flag PUSH pada paket bisa mempengaruhi frekuensi pengakuan (ACK) yang diperlukan[2].

Selanjutnya, korelasi antara *forward_subflow_packets* dan *backward_PSH_flags* dapat dijelaskan dengan *forward_subflow_packets* mengukur jumlah paket dalam aliran yang diarahkan ke depan, sedangkan *backward_PSH_flags* menunjukkan flag PUSH pada paket yang dikirim ke belakang. Jika aliran data ke depan mengandung banyak paket, maka ada kemungkinan lebih banyak interaksi dengan aliran balik, termasuk pengaturan flag PUSH[9].

Serta, korelasi antara *backward_subflow_bytes* dan *backward_PSH_flags* dapat dijelaskan dengan *backward_subflow_bytes*

id	forward_packets_per_sec	backward_packets_per_sec	flow_packets_per_sec
CkwI1TlUCRApPfcJl	11125.474801	NaN	22250.949602
CBlrcc3dvtaHzyV4zj	30174.848921	30174.848921	60349.697842
CdpSX33u29yjDvnVzi	0.322699	0.242025	0.564724
CT23VJ1KsoKeCdWpx2	NaN	82.478178	164.956355
C6OJU51P50bwNKvnY6	NaN	NaN	72.516256
CWhujWex7PeGY4Q78	36.078793	46.902431	NaN
CA4mfd5WpmkjY07Qk	76.190116	76.190116	NaN
CE0eax4wzKi6EeAWc5	34.350759	38.167510	72.518269
C4EbK82xAdSARKweec	NaN	56.906226	NaN
CJk4Yq3AHNRTsiz1rf	NaN	39.660199	79.320398

Tabel 4: Sampel 10 data pertama dari *train dataset*.

menunjukkan jumlah byte yang dikirim dalam aliran balik, sementara *backward_PSH_flags* mengukur frekuensi flag PUSH diatur dalam aliran balik. Semakin banyak byte yang dikirim dalam aliran balik, semakin besar kemungkinan adanya paket yang membutuhkan flag PUSH untuk kecepatan pengiriman[7].

C. Preprocessing

1) **Special Feature Imputation**.
Berdasarkan tabel 4, yang merupakan sampel 10 data pertama dari *train dataset*, dapat dilihat bahwa data dengan `id` **CE0eax4wzKi6EeAWc5** memiliki nilai *flow_packets_per_sec* yang merupakan hasil penjumlahan antara *forward_packets_per_sec* dan *backward_packets_per_sec*. Oleh karena itu, selama salah satu dari ketiga kolom tersebut hanya terdapat satu kolom yang mengandung `NaN`, nilai `NaN` pada kolom tersebut dapat diimputasi dengan menggunakan rumus berikut:

$$\texttt{flow\_packets\_per\_sec} = \texttt{forward\_packets\_per\_sec} + \texttt{backward\_packets\_per\_sec}$$

2) **General Feature Imputation**.

a) **Modus dan Median sebagai Metode Imputasi Terbaik**.
Berdasarkan hasil *Exploratory Data Analysis*, teridentifikasi adanya sejumlah besar nilai null atau `NaN` pada berbagai *feature* di *Train* dan juga *Test Dataset*. Untuk menangani masalah ini, kami menerapkan dua pendekatan imputasi yang berbeda. Pertama, untuk kolom-kolom dengan distribusi nilai yang tidak seimbang, seperti berbagai jenis flag dalam aliran jaringan, seperti *flow_FIN_flags*, *flow_SYN_flags*, *flow_RST_flags*, *forward_PSH_flags*, *backward_PSH_flags*, *flow_ACK_flags*, *forward_URG_flags*, *backward_URG_flags*, *flow_CWR_flags*, *flow_ECE_flags*, dan beberapa *feature* selain *flags*, seperti *forward_initial_window_size*, *backward_initial_window_size*, dan *forward_last_window_size*, kami menggunakan metode imputasi *most frequent* atau modus. Metode ini dipilih karena karakteristik data yang cenderung memiliki nilai dominan tertentu pada *features* tersebut.
Selain distribusi *value* yang tidak seimbang pada kolom tersebut, *skewness* dari kolom-kolom ini juga menjadi pertimbangan dalam memilih metode imputasi. *Skewness* mengukur tingkat asimetri dalam distribusi data. Pada kolom dengan *skewness* yang tinggi, distribusi data cenderung condong ke satu sisi, menunjukkan adanya nilai dominan yang lebih sering muncul. Oleh karena itu, penggunaan metode *most frequent* pada kolom-kolom dengan *skewness* tinggi memastikan bahwa nilai `NaN` diisi dengan nilai yang paling sering muncul, sehingga menjaga representasi distribusi asli data.

Kedua, untuk fitur-fitur numerik lainnya, kami menggunakan metode imputasi dengan nilai median untuk meminimalkan pengaruh *outlier*. Penggunaan median dipilih karena kemampuannya dalam memberikan estimasi yang lebih *robust* terhadap nilai-nilai ekstrem dibandingkan dengan *mean*. Hal ini sangat penting terutama pada fitur dengan *skewness* rendah atau sedang, di mana *outlier* dapat mendistorsi nilai *mean*. Dengan menggunakan median, kami dapat menjaga distribusi data tetap seimbang, memberikan gambaran yang lebih akurat dari pusat data tanpa terpengaruh oleh nilai ekstrem. Kombinasi kedua metode ini memungkinkan kami untuk mempertahankan karakteristik penting dari data sambil mengatasi masalah nilai `NaN`, sehingga menghasilkan *train* maupun *test dataset* yang lebih bersih dan konsisten. Pendekatan ini memastikan bahwa fitur-fitur dengan distribusi tidak seimbang dan *skewness* tinggi tetap mempertahankan nilai yang paling representatif, sementara fitur numerik yang lainnya diisi dengan nilai yang tidak terpengaruh oleh *outlier*.

b) **Ketidakefektifan Iterative dan KNN Imputer**.

Selain dua metode tersebut, kami juga mengeksplorasi penggunaan MICE Iterative Imputer dan KNN Imputer untuk mengatasi masalah nilai `NaN`. Namun, hasilnya menunjukkan bahwa kedua metode ini kurang cocok untuk dataset yang kami gunakan.

MICE (Multiple Imputation by Chained Equations) Iterative Imputer merupakan metode yang mencoba mengisi nilai `NaN` secara iteratif dengan membangun model regresi untuk setiap fitur yang memiliki nilai hilang. Metode ini bekerja dengan mengisi nilai `NaN` awal dengan nilai prediksi yang diperoleh dari model yang dibuat berdasarkan fitur lainnya. Proses ini kemudian diulang beberapa kali untuk meningkatkan akurasi prediksi. MICE mengasumsikan bahwa data mengikuti distribusi normal multivariat, sehingga ketika data memiliki *skewness* tinggi, seperti yang terdapat dalam dataset kami, prediksi yang dihasilkan menjadi kurang akurat. Selain itu, proses imputasi menggunakan MICE juga bisa menjadi sangat lambat untuk dataset yang besar karena memerlukan fitting model berulang kali untuk setiap fitur.
KNN (K-Nearest Neighbors) Imputer, di sisi lain, bekerja dengan mengisi nilai yang hilang berdasarkan nilai-nilai dari k tetangga terdekat. Metode ini mencari k sampel dalam dataset yang memiliki jarak terdekat dengan sampel yang memiliki nilai `NaN` dan kemudian menghitung rata-rata dari fitur tersebut untuk mengisi nilai yang hilang. Meskipun KNN Imputer dapat menangani berbagai jenis data dan tidak membuat asumsi distribusi, metode ini memiliki kelemahan dalam hal skala dataset. Dalam kasus kami, terdapat ratusan ribu data dengan banyak nilai yang hilang, sehingga perhitungan jarak antara sampel menjadi sangat mahal secara komputasional. Akibatnya, proses imputasi menggunakan KNN menjadi sangat lambat, dan setelah 11 jam berjalan, proses ini belum selesai.

Dengan mempertimbangkan hasil dari metode MICE dan KNN, kami memutuskan bahwa metode imputasi *most frequent* dan median adalah pendekatan yang lebih efisien dan tepat untuk dataset ini. Penggunaan *most frequent* pada fitur dengan distribusi tidak seimbang dan *skewness* tinggi memungkinkan kami untuk mengisi nilai `NaN` dengan cara yang tetap mempertahankan karakteristik distribusi asli. Sementara itu, penggunaan median pada fitur numerik lainnya membantu kami meminimalkan pengaruh *outlier*, sehingga menjaga distribusi data tetap seimbang dan akurat. Kombinasi kedua metode ini memastikan bahwa data yang dihasilkan lebih bersih dan konsisten.

D. Feature Engineering

Dalam klasifikasi serangan jaringan, proses *feature engineering* berperan penting untuk menciptakan fitur-fitur yang dapat membedakan lalu lintas jaringan normal dari yang mencurigakan atau berbahaya. Berikut fitur-fitur yang sudah kami rekayasa:

1) **Pemisahan Segmen Alamat IP**.

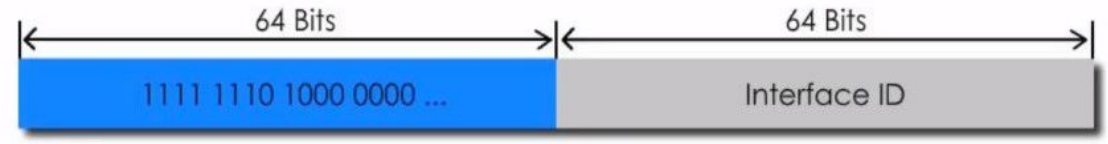
Sesuai dengan analisis data yang telah dilakukan, serta dengan pengetahuan umum mengenai alamat IP, masing-masing jenis alamat IP baik IPv4 dan IPv6 memiliki struktur yang berbeda. Alamat IPv4 terdiri dari 32 bit dan ditampilkan dalam notasi desimal yang dipisahkan oleh titik untuk setiap bloknya, dengan rentang setiap bloknya adalah angka desimal dari 0 hingga 255. Alamat IPv4 sendiri dibagi menjadi dua bagian yaitu network id yang menentukan jaringan spesifik di mana perangkat berada dan host id yang mengidentifikasi perangkat spesifik dalam jaringan tersebut [10]. Di sisi lain, IPv6 terdiri dari 128 bit dan dinyatakan dalam delapan grup atau blok dengan masing-masing blok terdiri dari empat digit heksadesimal, dipisahkan oleh tanda titik dua (colon). Grup nol berurutan dalam alamat IPv6 dapat disingkat menggunakan dua titik dua (::) untuk membuat notasi lebih singkat [5]. Berdasarkan informasi-informasi tersebut, kami pun melakukan pemrosesan fitur lebih lanjut yaitu dengan menguraikan alamat IP, baik itu IPv6 atau IPv4. Proses ini diharapkan dapat membantu dalam mengidentifikasi dan menganalisis setiap segmen atau blok alamat secara terpisah dan menemukan adanya pola-pola yang dapat membantu kinerja model nantinya.

Untuk alamat IPv4, yang lebih sederhana dan umum, prosesnya melibatkan pemisahan blok-blok numerik yang dipisahkan oleh titik. Ini penting untuk memahami bagaimana alamat-alamat ini dialokasikan di dalam jaringan, serta untuk mengidentifikasi potensi subnet atau jaringan lokal. Sedangkan untuk alamat IPv6, yang memiliki struktur lebih kompleks dan panjang, setiap blok diuraikan dan dikonversi dari format heksadesimal ke desimal untuk memudahkan analisis. Hal ini penting karena beberapa segmen dari alamat IPv6 bisa menunjukkan informasi penting seperti jenis alamat, alokasi geografis, atau aplikasi spesifik. Selain itu, fitur khusus dari alamat IPv6, seperti kompresi blok nol yang menunjukkan penggunaan '::' untuk menyederhanakan notasi alamat, juga diidentifikasi.

2) **Kesamaan Subnet pada Alamat IP Server dan Klien**.
Berdasarkan penelitian yang dipublikasikan oleh Wiley dalam "Security and Communication Networks" [11] tahun 2017, analisis subnet telah terbukti efektif dalam mendeteksi pola trafik anomali yang mungkin menunjukkan upaya serangan-serangan siber. Penelitian ini juga menyebutkan bahwa segmentasi jaringan, berdasarkan subnet, secara signifikan dapat meningkatkan kemampuan deteksi intrusi atau serangan. Berdasarkan referensi ini, kami pun melakukan testing pembuatan fitur pengelompokan atau penanda apakah bagi kedua alamat IP server dan juga client berada dalam subnet yang sama. Fitur ini berguna karena komunikasi antara alamat dalam subnet yang sama biasanya dianggap normal, sementara serangan internal dapat lebih mudah diidentifikasi jika diketahui asal dan tujuan berada dalam subnet yang sama.

3) **Klasifikasi Alamat IP**.
Fitur selanjutnya yang kami buat mengklasifikasikan alamat IP berdasarkan struktur dan kriteria numeriknya. Berdasarkan sub bab ke-2 RFC 4291 [3], dibahas mengenai pengalamatan dan klasifikasi alamat IPv6 yang mengkategorikan alamat IPv6 berdasarkan prefix-nya menjadi alamat lokal, multicast, dan alamat lainnya. Serta diikuti dengan RFC 1918 [6] yang membahas pengaturan pengkategorian alamat IPv4 sebagai alamat yang private atau publik.
Untuk alamat IPv6, tahapan feature engineering ini akan mengkategorikan alamat IP berdasarkan prefix [3]: alamat lokal unik (FE80 atau FEC0), alamat multicast (FF00 atau lebih tinggi), dan alamat IPv6 lainnya. Untuk IPv4, fungsi ini memeriksa apakah alamat tersebut termasuk dalam rentang private yang umum digunakan dalam jaringan lokal (misalnya, 10.x.x.x, 192.168.x.x, 172.16.x.x hingga 172.31.x.x) [6] dan mengkategorikannya sesuai itu.

4) **Identifikasi Origin dan Response Port Umum**.
Dalam konteks keamanan jaringan dan analisis trafik, memahami jenis port yang digunakan dalam komunikasi jaringan sangatlah penting. Port-port yang secara umum dikenal dan sering digunakan biasanya terkait dengan layanan internet yang spesifik, seperti HTTP pada port 80 dan HTTPS pada port 443. Port-port ini sering menjadi target serangan karena penggunaannya yang luas. Dengan mengidentifikasi port-port ini sebagai port 'umum' atau 'common', diharapkan dapat



Gambar 10: Ilustrasi Pengelompokan Alamat IP Lokal pada IPv6

membantu model untuk lebih mendapatkan pola-pola serangan berdasarkan origin dan response port dari setiap aliran jaringan. Di sisi lain, port yang tidak umum atau jarang digunakan dapat menandakan aktivitas yang tidak biasa atau mencurigakan dalam jaringan. Oleh karena itu, mengkategorikan port dapat membantu untuk memberikan informasi terhadap model dalam mendeteksi pola trafik yang tidak standar.

Setelah mendalami beberapa sumber [1][8][4] terkait port-port yang umum digunakan untuk aliran jaringan serta mencocokannya dengan port-port yang dimiliki oleh dataset, kami pun menambahkan fitur dengan hasil binary-classification yang mengidentifikasi umum atau tidaknya masing-masing port origin dan response tersebut.

5) **Pengembangan Fitur Berdasarkan Analisis *Traffic Benign***. Berdasarkan analisis sebelumnya yang menunjukkan kecenderungan ”*benign*” untuk *traffic* dengan kombinasi *host* asal ’103.255.15.150’, *response_host* ’128.199.242.104’, dan *response port* 443, penelitian ini berlanjut dengan pengembangan fitur untuk memperkuat model prediksi keamanan jaringan. Kami mengidentifikasi lima metrik kunci yang berkorelasi dengan karakteristik *traffic benign*, yaitu: durasi aliran jaringan (*flow_duration*), laju pengiriman paket dari *client* ke *server* per detik (*forward_packets_per_sec*), laju pengiriman paket dari *server* kembali ke *client* per detik (*backward_packets_per_sec*), laju keseluruhan paket yang dikirim per detik dalam seluruh aliran (*flow_packets_per_sec*), dan rata-rata waktu aktif dalam aliran (*active*).

   Untuk setiap *traffic* yang memenuhi kriteria *host* dan *port* yang telah ditetapkan, kami melakukan proses identifikasi *traffic* serupa dengan membandingkan kelima metrik tersebut. Proses ini dilakukan dengan mengelompokkan data berdasarkan *port* asal dan menerapkan algoritma yang membandingkan setiap sesi terhadap semua sesi lain dalam grup berdasarkan vektor fitur yang telah ditentukan. Kesamaan antar sesi dihitung berdasarkan jumlah metrik yang memiliki nilai yang sama atau sangat dekat, dengan toleransi yang telah ditetapkan untuk setiap metrik.

   Setiap sesi yang diidentifikasi memiliki *traffic* yang serupa diberi penanda (penanda *binary*) dalam *dataset* dan dihitung jumlah kesamaannya. Hal ini memungkinkan kami untuk tidak hanya mengidentifikasi pola *traffic benign* tetapi juga untuk mengobservasi frekuensi kemunculan pola tersebut dalam *dataset*. Penanda ini juga berfungsi sebagai fitur prediktif yang menambahkan dimensi data dalam pemodelan kami.

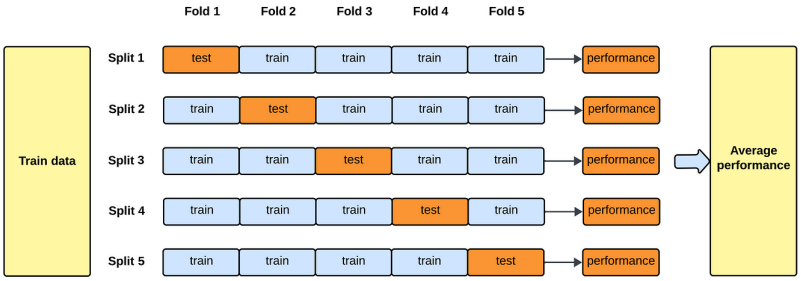
6) **active_greater_than_idle**

   Fitur ini adalah fitur biner yang menunjukkan apakah waktu aktif (‘active‘) lebih besar daripada waktu idle (‘idle‘). Jika ‘active‘ lebih besar, fitur ini akan bernilai 1 (True), dan jika tidak, akan bernilai 0 (False). Fitur ini dapat membantu mengidentifikasi aktivitas jaringan yang berkelanjutan dibandingkan dengan periode tidak aktif, yang mungkin menunjukkan pola aktivitas mencurigakan.

7) **flow_duration_x_payload**

   Fitur ini merupakan hasil perkalian antara ‘flow_duration‘ dan ‘payload_bytes_per_sec‘. ‘flow_duration‘ mengukur berapa lama aliran data berlangsung, sementara ‘payload_bytes_per_sec‘ menunjukkan jumlah data yang ditransfer per detik. Dengan mengalikan keduanya, fitur ini dapat menangkap hubungan antara durasi aliran dan ukuran muatan data, yang bisa menjadi indikator penting dalam mendeteksi aktivitas abnormal, seperti transfer data besar dalam waktu singkat.

8) **active_greater_than_idle**



Gambar 11: Skema *K-Fold Cross Validation*

Fitur ini adalah fitur biner yang menunjukkan apakah waktu aktif (‘active‘) lebih besar daripada waktu idle (‘idle‘). Jika ‘active‘ lebih besar, fitur ini akan bernilai 1 (True), dan jika tidak, akan bernilai 0 (False). Fitur ini dapat membantu mengidentifikasi aktivitas jaringan yang berkelanjutan dibandingkan dengan periode tidak aktif, yang mungkin menunjukkan pola aktivitas mencurigakan.

9) **flow_duration_x_payload**
Fitur ini merupakan hasil perkalian antara ‘flow_duration‘ dan ‘payload_bytes_per_sec‘. ‘flow_duration‘ mengukur berapa lama aliran data berlangsung, sementara ‘payload_bytes_per_sec‘ menunjukkan jumlah data yang ditransfer per detik. Dengan mengalikan keduanya, fitur ini dapat menangkap hubungan antara durasi aliran dan ukuran muatan data, yang bisa menjadi indikator penting dalam mendeteksi aktivitas abnormal, seperti transfer data besar dalam waktu singkat.

10) **flow_packets_x_payload**
Fitur ini adalah hasil perkalian antara ‘flow_packets_per_sec‘ dan ‘payload_bytes_per_sec‘. ‘flow_packets_per_sec‘ mengukur jumlah paket data yang dikirim per detik, sedangkan ‘payload_bytes_per_sec‘ mengukur ukuran data yang dikirim per detik. Fitur ini membantu memahami bagaimana frekuensi pengiriman paket terkait dengan ukuran data yang dikirim, yang dapat memberikan wawasan lebih dalam tentang karakteristik aliran jaringan.

E. Modelling

Stratified K-Fold sebagai Validator.

KFold Cross-Validation adalah teknik evaluasi model yang membagi dataset menjadi K subset atau folds. Model dilatih pada K-1 fold dan diuji pada fold yang tersisa, diulang sebanyak K kali sehingga setiap fold berfungsi sebagai data uji sekali. Hasil dari tiap iterasi dirata-rata untuk memberikan evaluasi kinerja model yang lebih akurat dan memaksimalkan penggunaan data.

Pada dataset ini dimana terjadinya *imbalance label class*, kami menggunakan variasi dari KFOld, yaitu Stratified KFold. Stratified KFold adalah variasi KFold yang menjaga distribusi kelas dalam setiap fold sama seperti distribusi asli dataset. Ini sangat penting untuk klasifikasi dengan distribusi kelas yang tidak seimbang, memastikan setiap fold merepresentasikan proporsi kelas dengan lebih baik, sehingga hasil evaluasi lebih konsisten dan tidak bias terhadap kelas mayoritas.

Mengatasi Ketidakseimbangan Kelas pada Dataset.

Berdasarkan *train dataset*, dapat dilihat bahwa data memiliki label `traffic` yang sangat *imbalance* terutama pada kelas *Benign* dan *Background*. Ketidakseimbangan ini dapat menyebabkan model menjadi bias, di mana model akan lebih cenderung

memprediksi label yang memiliki jumlah sampel terbanyak (*majority class*) tanpa mementingkan label dengan jumlah sampel yang lebih sedikit (*minority class*). Akibatnya, model mungkin akan memiliki performa yang buruk dalam mendeteksi kelas minoritas yang seringkali lebih penting dalam konteks tertentu. Misalnya, dalam kasus deteksi intrusi jaringan, kelas minoritas mungkin mewakili serangan berbahaya yang harus segera dikenali, meskipun jarang terjadi. Mengatasi masalah ini harus melibatkan penanganan kelas mayoritas untuk memastikan bahwa model tidak menjadi bias. Jika model hanya berfokus pada memprediksi kelas mayoritas dengan benar, ia akan mengabaikan kelas minoritas, yang dapat menyebabkan tingginya *False Negatives* untuk kelas minoritas. Misalnya, dalam konteks keamanan jaringan, ini berarti model mungkin gagal mendeteksi serangan berbahaya, yang dapat memiliki konsekuensi serius.

Untuk mengevaluasi performa model dalam kondisi ketidakseimbangan ini, kami menggunakan metrik *Balanced Accuracy*. *Balanced Accuracy* adalah metrik yang memberikan bobot yang sama kepada setiap kelas, dengan rumus sebagai berikut:

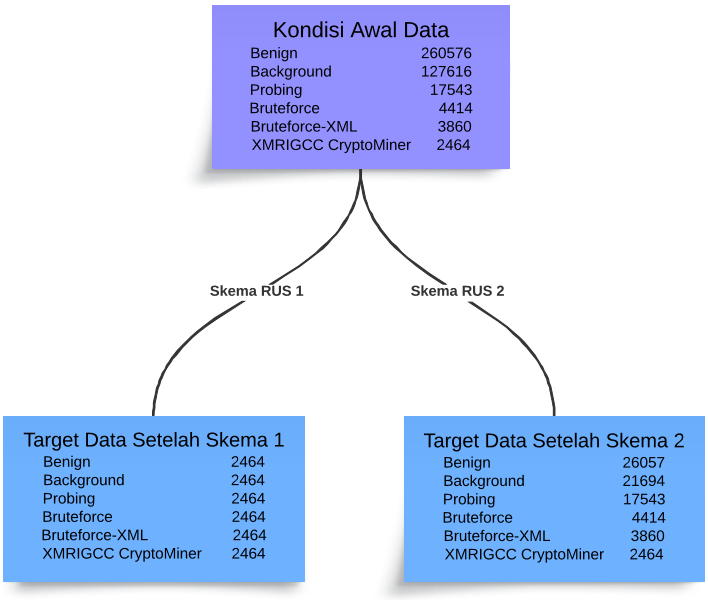
$$\text{Balanced Accuracy} = \frac{1}{2}\left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right) \tag{1}$$

di mana TP adalah *True Positives*, FN adalah *False Negatives*, TN adalah *True Negatives*, dan FP adalah *False Positives*.

Metrik ini memastikan bahwa hasil prediksi harus seimbang di setiap kelasnya, dan tidak boleh ada kecenderungan hasil prediksi yang terlalu terpusat pada satu label. Dengan menggunakan *Balanced Accuracy*, kita dapat memastikan bahwa model memberikan perhatian yang cukup pada kelas minoritas, sehingga performa model tidak hanya diukur berdasarkan kemampuan memprediksi kelas mayoritas saja.

Oleh karena itu, kami menerapkan beberapa skema penelitian untuk menangani masalah ketidakseimbangan data ini, yaitu

1) **Random Under Sampling (RUS)**: Dalam penelitian ini, saya menerapkan dua skema penggunaan RUS. Pada skema pertama, saya mencoba menggunakan RUS untuk menyeimbangkan jumlah sampel di setiap kelas sehingga proporsinya menjadi 1:1:1:1:1:1. Dengan kata lain, semua kelas selain dari kelas mayoritas akan memiliki jumlah sampel yang sama. Pendekatan ini diharapkan memungkinkan model untuk memprediksi dengan baik tanpa perlu melakukan pengaturan bobot (*weighting*) untuk setiap label.
   Skema kedua yang saya terapkan adalah menggunakan RUS secara selektif hanya pada label *Benign* dan *Background*. Tujuan dari pendekatan ini adalah untuk mengurangi jumlah sampel pada kedua label tersebut agar tidak terlalu jauh berbeda dari jumlah sampel pada label 'Probing'. Dalam hal ini, data dari label 'Benign' diambil secara acak sebesar 10% dari total data yang ada, sedangkan data dari label 'Background' diambil secara acak sebesar 17% dari total data yang ada. Dengan melakukan under sampling ini, diharapkan distribusi data menjadi lebih seimbang dan model dapat melakukan prediksi dengan lebih akurat tanpa kehilangan informasi penting dari kelas minoritas.
2) **LightGBM dengan Weighting Auto Balance**: LightGBM memiliki parameter yang dapat secara otomatis memberikan bobot pada kelas berdasarkan proporsi data. Dengan mengatur parameter 'class_weight="balanced"', model akan menyesuaikan bobot untuk setiap kelas sehingga memberikan perhatian lebih pada kelas minoritas selama pelatihan. Hal ini sangat berguna dalam kasus ketidakseimbangan kelas, di mana model cenderung bias terhadap kelas mayoritas jika bobot tidak diatur dengan benar.
   Selain itu, model LightGBM ini telah di-tuning dengan berbagai parameter untuk meningkatkan performa dalam memprediksi kelas yang tidak seimbang. Misalnya, penggunaan 'max_depth' yang lebih tinggi memungkinkan model untuk menangkap interaksi kompleks antara fitur, yang dapat membantu dalam mendeteksi pola khusus dari kelas minoritas. Parameter 'lambda_l1' dan 'lambda_l2' digunakan untuk regularisasi, mengurangi overfitting dengan menghukum kompleksitas model,



Gambar 12: Skema Eksperimen *Random Under Sampling*

terutama ketika berhadapan dengan fitur yang mungkin tidak terlalu relevan.

Parameter seperti ‘min_split_gain‘ membantu dalam mencegah pemecahan pohon yang tidak diperlukan kecuali ada keuntungan yang signifikan, yang dapat menjaga integritas informasi penting dari kelas minoritas. Pengaturan ‘min_child_weight‘ mencegah pemecahan simpul yang mengandung sedikit sampel, sehingga menghindari overfitting pada kelas mayoritas.

Penggunaan ‘colsample_bytree‘ dan ‘subsample‘ memungkinkan model untuk bekerja dengan subset data dan fitur secara acak selama pelatihan, yang dapat meningkatkan generalisasi dan kinerja pada kelas minoritas. Dengan men-tuning parameter seperti ‘reg_alpha‘ dan ‘reg_lambda‘, model ini mengontrol regularisasi tambahan untuk menangani ketidakseimbangan kelas dan meningkatkan kemampuan prediksi.

Selain itu, ‘n_estimators‘ diatur pada jumlah iterasi tertentu untuk memastikan model memiliki cukup pohon untuk belajar dengan baik tanpa overfitting. Melalui fine-tuning parameter ini, model dapat mencapai keseimbangan yang baik antara bias dan varians, sehingga mampu memprediksi kelas minoritas dengan lebih akurat meskipun dataset tidak seimbang.

3) **CatBoost dengan Auto Balance Weight**: Seperti LightGBM, CatBoost juga memiliki fitur yang memungkinkan penyeimbangan bobot otomatis antar kelas melalui pengaturan parameter ‘auto_class_weights‘. Fitur ini memastikan bahwa model memberikan perhatian yang cukup pada kelas minoritas selama proses pelatihan, yang sangat penting dalam konteks klasifikasi serangan jaringan di mana sering terjadi ketidakseimbangan antara jumlah sampel normal dan serangan.

   Pada model ini, kami menggunakan `CatBoostClassifier` dengan sejumlah iterasi pohon keputusan yang telah dioptimalkan, yakni sekitar 1500 iterasi. Jumlah ini dipilih berdasarkan optimasi menggunakan Optuna, dengan tujuan mencapai keseimbangan antara akurasi dan waktu komputasi. Dengan iterasi sebanyak ini, model mampu menangkap pola kompleks tanpa mengorbankan efisiensi pelatihan.

   Parameter `learning_rate` dipilih dalam kisaran yang memungkinkan model untuk belajar secara efektif, sekitar 0.2. Dengan nilai ini, model dapat menyesuaikan diri dengan cepat pada setiap iterasi, tetapi tetap mempertahankan generalisasi yang baik untuk menghindari *overfitting*. Nilai yang terlalu tinggi dapat menyebabkan model terlalu sensitif terhadap data

pelatihan, sedangkan nilai yang terlalu rendah dapat menyebabkan pelatihan menjadi terlalu lambat.

Selain itu, `l2_leaf_reg` digunakan untuk memberikan regulasi pada daun pohon keputusan, dengan nilai di kisaran menengah hingga tinggi. Regulasi ini membantu mencegah *overfitting* dengan menambahkan penalti pada kompleksitas model. Melalui optimasi, nilai yang optimal dalam kisaran ini dapat menjaga keseimbangan antara bias dan varians, memastikan bahwa model tidak terlalu kompleks namun cukup kuat untuk menangkap pola yang ada.

Untuk menangani ketidakseimbangan kelas, kami mengatur parameter `auto_class_weights="Balanced"`. Hal ini memungkinkan model untuk secara otomatis memberikan bobot lebih pada kelas minoritas, memastikan bahwa model tetap sensitif terhadap kelas-kelas yang kurang terwakili dalam data.

Dalam pemilihan fitur kategori, kami menggunakan kolom seperti `origin_host`, `response_host`, `origin_port`, `response_port`, serta fitur-fitur gabungan seperti `concat_1`, `concat_2`, dan `concat_3`. Fitur-fitur ini dikategorikan karena mereka mewakili nilai diskret yang lebih bermakna secara kontekstual daripada numerik, sehingga CatBoost dapat menangani fitur-fitur ini dengan cara yang meningkatkan kinerja model.

Kami tidak perlu secara eksplisit mengatur parameter seperti `depth` atau `border_count`, karena CatBoost memiliki kemampuan adaptif bawaan untuk menyesuaikan kedalaman pohon dan pengaturan batas fitur selama pelatihan. Fitur ini mengurangi kebutuhan untuk penyetelan manual dan memungkinkan model untuk secara dinamis menyesuaikan strukturnya guna mencapai performa optimal.

Perbedaan Waktu CPU vs GPU.

Pada awalnya, kami melakukan seluruh proses eksperimen menggunakan CPU. Eksperimen tersebut selanjutnya akan kami uji performa metriksnya menggunakan *Stratified K-Fold*. Ternyata, untuk melakukan *Stratified 5-Fold* pada satu skema *CatBoost* model memakan waktu pelatihan dan validasi hingga 5 jam. Sumber daya waktu yang terambil sangat banyak ini membuat kami memutuskan untuk menggunakan GPU sebagai infrastruktur pelatihan model *CatBoost* utama. Perubahan infrastruktur dari CPU menjadi GPU tersebut membuat waktu pelatihan model menurun jauh hingga menjadi setengah jam saja.

F. Hasil Analisis

Berdasarkan beberapa skema eksperimen yang telah dijabarkan pada bagian *modelling*, kami melakukan ujicoba setiap skema terhadap performa metriks `balanced_accuracy`. Kami membandingkan skor setiap skema dengan menggunakan tiga validator yaitu *Stratified K-Fold*, *public leaderboard*, serta hasilnya pada *private leaderboard*. Hasil eksperimen dan penelitian tersebut kami jabarkan pada tabel 5.

Skema Eksperimen	***Private Leaderboard***	**Public Leaderboard**	**Stratified K-Fold**
LGBM + RUS Skema 1	$0,80839$	$0,80769$	$0,8206$
LGBM + RUS Skema 2	$0,79874$	$0,79851$	$0,8198$
LGBM Auto Balance Weighting	$0,82060$	$0,82240$	$0,82170$
CatBoost Auto Balance Weighting dengan CPU	$0,87270$	$0,87257$	$0,87201$
CatBoost Auto Balance Weighting dengan GPU	$0,84221$	$0,84337$	$0,84566$
CatBoost + RUS Skema 2 dengan CPU	$0,85140$	$0,85504$	$0,85387$
CatBoost + RUS Skema 2 dengan GPU	$0,82230$	$0,82224$	0.81240

Tabel 5: Hasil Skema Eksperimen

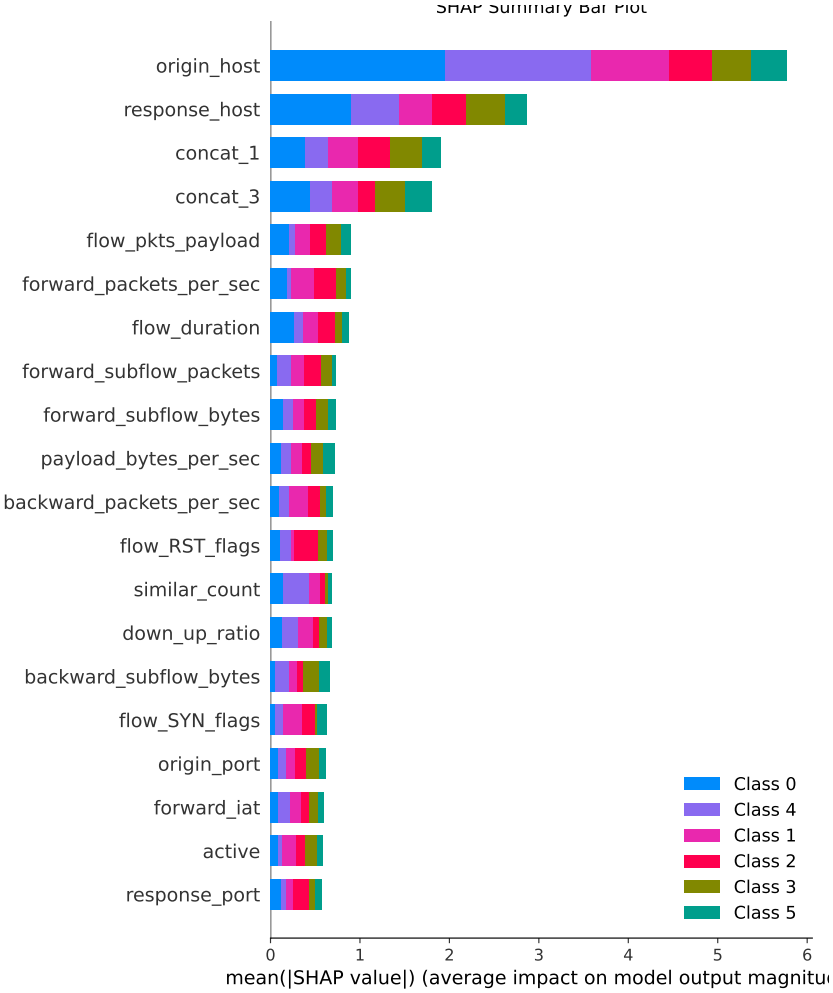
Performa Model Terbaik dengan CPU. Tabel hasil skema eksperimen di atas menunjukkan bahwa CatBoost Auto Balance Weighting yang di train menggunakan CPU mencatatkan skor metrik paling tinggi. Menariknya, ternyata performa skor yang

dicatatkan berbeda antara CPU dan GPU. Meskipun proses eksperimen dan model yang digunakan sama, model yang dilatih di CPU mencatatkan performa skor yang lebih baik dibandingkan model yang dilatih di GPU. Hal ini disebabkan adanya nilai *default* dan *range hyperparameter* yang berbeda antara CPU dan GPU. Salah satu contohnya adalah *hyperparameter* `border_count`. Parameter ini memiliki nilai *default* yang berbeda untuk CPU dan GPU, dengan CPU sebanyak 254 *splits* dan GPU hanya sebanyak 128 *splits*. Bahkan, pada fitur tertentu seperti `tree_depth`, beberapa GPU memiliki batasan kedalaman yaitu 8, berbeda dibandingkan menggunakan CPU dimana `tree_depth` memiliki *range* dari 1 hingga 16.

Weighting vs RUS. Perbandingan performa model yang telah dilakukan menunjukkan superioritas weighting terhadap RUS dalam mengatasi ketidakseimbangan kelas. Hal ini bisa terjadi dikarenakan kemungkinan bahwa RUS menghilangkan informasi penting dari data kelas mayoritas yang dapat berdampak negatif pada kinerja model. Karena RUS secara acak mengurangi jumlah sampel dari kelas mayoritas, ada risiko bahwa beberapa sampel yang dihilangkan mungkin mengandung fitur-fitur penting yang membantu model dalam melakukan generalisasi. Hal ini dapat menyebabkan model menjadi kurang akurat atau kurang mampu menangkap variasi dalam data, terutama jika informasi yang dihilangkan tersebut relevan untuk membedakan antara kelas-kelas yang berbeda. Berbeda dengan RUS, *balanced weighting* yang dilakukan model telah diatur sedemkian rupa sehingga secara otomatis model memberikan perhatian lebih pada kelas minoritas tanpa membuang informasi yang ada pada kelas mayoritas.

Mengatasi Masalah Besarnya Waktu Latih. Dalam proses pelatihan model selama ujicoba, didapati bahwa model yang dilatih menggunakan CPU memakan waktu yang relatif lama, bahkan hingga berjam-jam. Konsumsi waktu yang sangat lama ini disebabkan oleh besarnya dataset pelatihan yang mencapai hingga ratusan ribu data. Mengingat model terbaik berdasarkan performa adalah model yang dilatih menggunakan CPU, perlu ditentukan cara supaya pelatihan model ini tidak memakan waktu yang sangat lama. Dalam mengatasi hal tersebut, **PCA (*Principal Component Analysis*)** dapat diterapkan. PCA adalah salah satu adalah teknik reduksi dimensi yang digunakan untuk mengurangi jumlah fitur dalam dataset sambil mempertahankan informasi yang paling penting. PCA bekerja dengan mentransformasikan fitur asli menjadi sejumlah principal components, yang merupakan kombinasi linear dari fitur-fitur awal. Komponen-komponen ini dipilih berdasarkan variansi tertinggi, sehingga fitur yang memiliki variabilitas terbesar akan diutamakan. Dengan PCA, kita dapat mengurangi kompleksitas model, mempercepat proses komputasi, dan mengurangi overfitting, tanpa terlalu banyak kehilangan informasi penting.

Analisis SHAP Value. Berdasarkan hasil analisis SHAP yang ditampilkan, terlihat ada top 20 fitur yang memiliki pengaruh lebih ke dataset dibandingkan fitur-fitur lainnya. Fitur origin_host merupakan fitur yang paling penting dalam membedakan antar kelas (Class 0 hingga Class 5). Fitur ini memberikan kontribusi yang signifikan di hampir semua kelas, terutama pada Class 0 dan Class 4. Selain itu, response_host juga berperan penting, meskipun dampaknya sedikit lebih kecil dibandingkan dengan origin_host. Kontribusi terbesar response_host terlihat pada Class 0 dan Class 4. Fitur-fitur seperti concat_1 dan concat_3 juga menunjukkan pengaruh yang cukup besar, terutama pada Class 1 dan Class 3. Hal ini menunjukkan bahwa kedua fitur tersebut relevan dalam membantu model mengklasifikasikan data ke kelas-kelas tertentu. Jika dilihat lebih dalam, fitur-fitur yang memberikan pengaruh besar adalah fitur-fitur yang terkait dengan identitas pelaku dan penerima aliran trafik. Di sisi lain, beberapa fitur lain seperti active, response_port, dan forward_iat memiliki kontribusi yang lebih kecil. Fitur-fitur ini memberikan pengaruh yang relatif rendah terhadap keputusan klasifikasi model, sehingga bisa dianggap tidak terlalu penting dalam membedakan antar kelas. Meskipun begitu, fitur-fitur tersebut masih memberikan pengaruh yang lebih baik dibandingkan fitur-fitur yang tidak terlihat dalam Top 20 SHAP fitur.



Gambar 13: Shap Value Catboost CPU

III. KESIMPULAN DAN SARAN

Kesimpulan.

Penelitian ini berhasil menunjukkan pentingnya analisis dan pemodelan lalu lintas jaringan untuk deteksi intrusi dan aktivitas mencurigakan. Dengan menggabungkan teknik pembelajaran mesin dan eksplorasi data yang mendalam, model yang dikembangkan mampu mengidentifikasi pola lalu lintas yang normal dan anomali secara akurat. Hasil eksperimen menunjukkan bahwa metode Auto Balance Weighting dengan algoritma CatBoost memberikan performa terbaik, terutama pada pengujian menggunakan CPU. Metode ini berhasil mengatasi ketidakseimbangan data antar kelas, yang umumnya menjadi kendala dalam deteksi intrusi jaringan.

Saran.

Adapun saran-saran untuk pengembangan dari penelitian ini adalah:

1) Pengembangan Model Berbasis GPU: Meskipun CPU menunjukkan hasil yang lebih baik, eksplorasi lebih lanjut menggunakan GPU perlu dilakukan dengan penyesuaian hyperparameter yang lebih optimal. Hal ini dapat mempercepat waktu komputasi

dan memberikan performa yang setara atau lebih baik.

2) Pengujian di Lingkungan Jaringan Nyata: Hasil penelitian dapat lebih diperkaya dengan menerapkan model ini pada lingkungan jaringan yang nyata dan dinamis, yang mungkin memiliki karakteristik berbeda dibandingkan dengan dataset yang digunakan dalam penelitian.
3) Peningkatan Skema Eksperimen: Implementasi variasi skema seperti kombinasi RUS dengan Weighted Sampling bisa dijadikan bahan penelitian lebih lanjut untuk mengetahui metode terbaik dalam menangani ketidakseimbangan data kelas. Dapat juga dilakukan eksperimen dengan fitur-fitur tertentu saja.

REFERENSI

[1] GeeksforGeeks. 50 common ports you should know. Web, GeeksforGeeks, 2024. URL https://www.geeksforgeeks.org/50-common-ports-you-should-know/.

[2] S. Harris. *Network Security Essentials*. McGraw-Hill Education, 4th edition, 2016.

[3] R. Hinden dan S. Deering. Ip version 6 addressing architecture. RFC 4291, IETF, 2006. URL https://datatracker.ietf.org/doc/html/rfc4291.

[4] NetworkProGuide. Common ports cheat sheet. Web, NetworkProGuide, 2024. URL https://networkproguide.com/common-ports-cheat-sheet/.

[5] M. Pietroforte. Ipv6 tutorial – part 4: Ipv6 address syntax. https://4sysops.com/archives/ipv6-tutorial-part-4-ipv6-address-syntax/, 2011. Accessed: [Date you accessed the URL].

[6] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, dan E. Lear. Address allocation for private internets. RFC 1918, IETF, 1996. URL https://www.ietf.org/rfc/rfc1918.txt.

[7] P. G. Starkweather, M. R. Allen. Analyzing network traffic patterns. *Journal of Computer Networks*, 52(4):1234–1247, 2008. doi: 10.1016/j.jocn.2008.01.005.

[8] StationX. Common ports cheat sheet: The ultimate list. Web, StationX, 2024. URL https://www.stationx.net/common-ports-cheat-sheet/.

[9] A. S. Tanenbaum. *Computer Networks*. Prentice Hall, 5th edition, 2011.

[10] Y. Tian dan J. Gao. *Network Addressing Architecture*. Springer, 2023. doi: 10.1007/978-981-99-5648-7_6.

[11] J. Wiley. Security and communication networks. *Security and Communication Networks*, 2017. ISSN 1939-0122.