KLASIFIKASI *MALWARE* PADA TRAFIK JARINGAN MENGGUNAKAN METODE *LOGISTIC REGRESSION*

SKRIPSI

Oleh :

CHARIS MAULANA SETYA ADI
NIM. 210605110083

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2025

KLASIFIKASI *MALWARE* PADA TRAFIK JARINGAN MENGGUNAKAN METODE *LOGISTIC REGRESSION*

SKRIPSI

Diajukan kepada:
Universitas Islam Negeri Maulana Malik Ibrahim Malang
Untuk memenuhi Salah Satu Persyaratan dalam
Memperoleh Gelar Sarjana Komputer (S.Kom)

Oleh :
CHARIS MAULANA SETYA ADI
NIM. 210605110083

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM MALANG
2025

HALAMAN PERSETUJUAN

KLASIFIKASI MALWARE PADA TRAFIK JARINGAN MENGGUNAKAN METODE *LOGISTIC REGRESSION*

SKRIPSI

Oleh :
CHARIS MAULANA SETYA ADI
NIM. 210605110083

Telah Diperiksa dan Disetujui untuk Diuji:
Tanggal: 16 Juni 2025

Pembimbing I,	Pembimbing II,
Ajib Hanani, M.T	Johan Ericka Wahyu Prakasa, M.Kom
NIP. 19840731 202321 1 013	NIP. 19831213 201903 1 004

Mengetahui,
Ketua Program Studi Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Ir. Fachrul Kurniawan, M.MT, IPU
NIP. 19771020 200912 1 001

HALAMAN PENGESAHAN

KLASIFIKASI *MALWARE* PADA TRAFIK JARINGAN MENGGUNAKAN METODE *LOGISTIC REGRESSION*

SKRIPSI

Oleh :
CHARIS MAULANA SETYA ADI
NIM. 210605110083

Telah Dipertahankan di Depan Dewan Penguji Skripsi
dan Dinyatakan Diterima Sebagai Salah Satu Persyaratan
Untuk Memperoleh Gelar Sarjana Komputer (S.Kom)
Tanggal: 16 Juni 2025

Susunan Dewan Penguji

Ketua Penguji	: Dr. M. Amin Hariyadi, M.T NIP. 19670018 200501 1 001	( )
Anggota Penguji I	: Shoffin Nahwa Utama, M.T NIP. 19860703 20201 2 1003	( )
Anggota Penguji II	: Ajib Hanani, M.T NIP. 19840731 202321 1 013	( )
Anggota Penguji III	: Johan Ericka Wahyu Prakasa, M.Kom NIP. 19831213 201903 1 004	( )

Mengetahui dan Mengesahkan,
Ketua Program Studi Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Ir. Fachrul Kurniawan, M.MT, IPU
NIP. 19771020 200912 1 001

PERNYATAAN KEASLIAN TULISAN

Saya yang bertanda tangan di bawah ini:

Nama	: Charis Maulana Setya Adi
NIM	: 210605110083
Fakultas / Program Studi	: Sains dan Teknologi / Teknik Informatika
Judul Skripsi	: Klasifikasi *Malware* Pada Trafik Jaringan Menggunakan Metode *Logistic Regression.*

Menyatakan dengan sebenarnya bahwa Skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri, bukan merupakan pengambil alihan data, tulisan, atau pikiran orang lain yang saya akui sebagai hasil tulisan atau pikiran saya sendiri, kecuali dengan mencantumkan sumber cuplikan pada daftar pustaka.

Apabila dikemudian hari terbukti atau dapat dibuktikan skripsi ini merupakan hasil jiplakan, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Malang, 16 Juni 2025
Yang membuat pernyataan,

Charis Maulana Setya Adi
NIM. 210605110083

MOTTO

Strive your best for

Allah is with you wherever you are.

...وَهُوَ مَعَكُمْ اَيْنَ مَا كُنْتُمْ...

Berikhtiarlah dengan sepenuh hati, serahkan hasilnya pada Ilahi

karena Allah selalu bersamamu di mana pun engkau berada.

HALAMAN PERSEMBAHAN

Dengan segenap rasa syukur kepada **Allah Subhanahu wa Ta'ala**,
Sang Maha Pencipta, atas limpahan rahmat, kesehatan, dan kekuatan yang tiada terhingga sehingga berkat izin dan kehendak-Nya, karya sederhana ini akhirnya dapat terselesaikan dengan seizin-nya. Penulis mempersembahkan karya ini kepada:

Ibunda dan Ayahanda tercinta,

Suci Hidayati & Moh. Fahrur Rozi,

Lentera hidup yang tak pernah redup,
dengan doa yang senantiasa mengiringi setiap langkah,
menjadi sumber kekuatan dalam diam,
dan teladan dalam ketulusan dan kesabaran.

Teruntuk Saudara dan Saudariku.

Achmad Afriza Fahrezi S & Adhelia Putri El-Maila

Teman tumbuh dan berbagi langkah,
yang hadir sebagai semangat dalam senyap
dan penyemangat di kala langkah mulai melemah.

Diri saya sendiri,

Charis Maulana Setya Adi, Aal

Yang telah memilih untuk terus melangkah meski tertatih, belajar untuk tidak menyerah di tengah keterbatasan,dan tetap berdiri ketika keadaan menguji batas kemampuan. Terima kasih telah bertahan sejauh ini,
dan semoga terus kuat menapaki perjalanan yang masih panjang ke depan.

KATA PENGANTAR

Bismillahirrahmaanirrahiim, Assalamualaikum Wr. Wb.

Segala puji dan syukur penulis panjatkan ke hadirat Allah Subhanahu wa Ta'ala atas segala limpahan rahmat, taufik, dan hidayah-Nya, sehingga penulis dapat menyelesaikan skripsi yang berjudul "*Klasifikasi Malware pada Trafik Jaringan Menggunakan Metode Logistic Regression"* . Skripsi ini disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer (S.Kom) pada Program Studi Teknik Informatika, Fakultas Sains dan Teknologi, Universitas Islam Negeri Maulana Malik Ibrahim Malang.

Penyusunan skripsi ini tidak terlepas dari bantuan dan dukungan dari berbagai pihak. Oleh karena itu, pada kesempatan ini penulis ingin menyampaikan ucapan terima kasih yang sebesar-besarnya kepada:

1. Prof. Dr. M. Zainuddin, M.A., selaku Rektor Universitas Islam Negeri Maulana Malik Ibrahim Malang, atas kebijakan yang inovatif dalam pengembangan sarana dan prasarana pembelajaran, sehingga mendukung tercapainya kualitas pendidikan yang optimal.
2. Prof. Dr. Hj. Sri Harini, M.Si., selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang, atas kepemimpinannya yang inspiratif dalam memajukan kualitas pendidikan di lingkungan fakultas.
3. Dr. Ir. Fachrul Kurniawan, M.MT., IPU., selaku Ketua Program Studi Teknik Informatika Universitas Islam Negeri Maulana Malik Ibrahim Malang dan

Wali Dosen, atas motivasi dan arahan yang diberikan selama proses perkuliahan.

4. Ajib Hanani, M.T., selaku Dosen Pembimbing 1 dan Johan Ericka Wahyu Prakasa, M.Kom., selaku Dosen Pembimbing 2 yang telah memberikan Bimbingan, saran, kesabaran dan dukungan yang sangat berarti bagi penulis selama proses penulisan skripsi dari awal hingga akhir.
5. Dr. M. Amin Hariyadi, M.T dan Shoffin Nahwa Utama, M.T, selaku Dosen Penguji, kritik dan saran yang sangat membantu dalam memperbaiki kualitas skripsi ini.
6. Seluruh staf dan dosen Program Studi Teknik Informatika, atas ilmu, dukungan, dan fasilitas yang telah diberikan kepada penulis selama masa studi.
7. Kedua orang tua dan adik tercinta serta keluarga besar Bani Mahfudz, Suci Hidayati, Moh. Fahrur Rozi, Achmad Afriza Fahrezi S, Adhelia Putri El-Maila, Sumiyati,.. atas cinta, doa, dan dukungan tiada henti yang menjadi kekuatan terbesar penulis dalam menyelesaikan studi ini.
8. Seseorang yang tak kalah penting kehadirannya, Sita Maulidia Alyatu Zahra. Terima kasih atas setiap doa, support, semangat dan segala kontribusinya dalam proses skripsi ini. Terima kasih telah menjadi peneduh di kala lelah, dan teman terbaik dalam setiap cerita. Terus berjuang bersama, hingga tiba waktunya esok.
9. Teman terbaik sekaligus saudara, Yusuf “Ndut”, Anas “Mbut”, Yatno, dan Adib “Vemble” dalam Nyxzone, yang pernah bersama-sama merancang impian membangun jasa joki ML, meski belum sempat terealisasi. Lalu Ikfin

atau yang biasa penulis panggil iik, yang telah bersama sejak bangku SD hingga saat ini Semoga kita semua selalu diberi kesehatan dan tetap semangat menjalani kerasnya hidup ini.

10. Seluruh rekan-rekan “MMS Satoe”, karya kecil ini kupersembahkan untuk kalian yang selalu menjadi sumber semangatku. Terima kasih sudah ingin berbagi tawa serta menjadi pelengkap cerita dalam setiap langkah.
11. Seluruh rekan Teknik Informatika terkhusus Angkatan 2021 “ASTER”, atas segala pengalaman berharga yang telah dibagikan. Semoga silaturahmi kita semakin erat dan kita semua dapat meraih impian kita semua.
12. Musisi-musisi ternama seperti Justin Bieber, Bring Me The Horizon, yang lewat karya-karyanya telah menjadi teman setia, menemani tiap malam panjang dan proses sunyi dalam menyelesaikan skripsi ini.

Penulis menyadari skripsi ini masih perlu penyempurnaan. Segala kritik dan saran sangat penulis harapkan untuk perbaikan. Semoga penelitian ini bermanfaat kepada pembaca dalam pengembangan ilmu kedepannya.

Malang, 16 Juni 2025

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Adi, Charis Maulana Setya**.** 2025**. Klasifikasi *Malware* pada Trafik Jaringan Menggunakan Metode *Logistic Regression***. Skripsi. Program Studi Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang. Pembimbing: (I) Ajib Hanani, M.T (II) Johan Ericka Wahyu Prakasa, M.Kom.

Kata Kunci: *Malware, Logistic Regression (LR)*, klasifikasi, Trafik Jaringan, *Cross Validation*

Peningkatan lalu lintas jaringan seiring dengan kemajuan teknologi informasi turut memperbesar risiko serangan *Malware*. Oleh karena itu, deteksi dini terhadap trafik jaringan yang mencurigakan menjadi langkah penting dalam mencegah kerugian akibat ancaman siber. Penelitian ini bertujuan untuk mengukur performa metode *Logistic Regression* dalam mengklasifikasikan *Malware* pada trafik jaringan, menggunakan data yang telah melalui tahapan *Preprocessing* berupa seleksi *Fitur*, *Encoding*, dan *Normalisasi*. Model dilatih dengan teknik 5-*Fold Cross Validation* guna mengukur konsistensi dan kemampuan generalisasi terhadap data baru. Penelitian menguji tiga skenario dengan perbedaan metode pembagian data, yakni manual *split* dengan *rasio* 80:20 dan *random split* dengan *rasio* 80:20 dan 70:30. Evaluasi dilakukan menggunakan metrik *akurasi*, *presisi*, *recall*, dan *F1-score* berdasarkan hasil *confusion matrix*. Hasil terbaik diperoleh pada Skenario 3 (*random split* 70:30) dengan nilai *akurasi* sebesar 92,93%, *presisi* 93,89%, *recall* 92,83%, dan *F1-score* 92,88%. *Visualisasi* nilai *Loss* dan *Learning curve* menunjukkan proses pemodelan yang stabil tanpa *overfitting*, serta kemampuan generalisasi yang baik terhadap data validasi. Temuan ini menunjukkan bahwa *Logistic Regression* merupakan pendekatan klasifikasi yang efektif dan layak diterapkan dalam sistem deteksi *Malware* berbasis trafik jaringan, karena mampu memberikan performa tinggi yang seimbang antara deteksi ancaman dan pengenalan trafik normal.

ABSTRACT

Adi, Charis Maulana Setya**.** 2025**.** ***Malware*** **Classification on Network Traffic Using** ***Logistic Regression*** **Method**. Undergraduate Thesis. Informatics Engineering Study Program, Faculty of Science and Technology, Universitas Islam Negeri Maulana Malik Ibrahim Malang. Supervisor: (I) Ajib Hanani, M.T (II) Johan Ericka Wahyu Prakasa, M.Kom.

Keywords : *Malware, Logistic Regression (LR), Classification, Network Traffic, Cross Validation*

The growth of information technology has led to an increase in network traffic, thereby escalating the risk of *Malware* attacks. Therefore, early detection of suspicious network traffic is crucial to prevent potential cyber-related *Loss* es. This study evaluates the performance of the *Logistic Regression* method in classifying *Malware* based on network traffic data. The dataset used for this research underwent pre-processing, including feature selection, *Encoding*, and normalisation. The model was trained using 5-*Fold* cross-validation to evaluate its consistency and generalisation capability. Three modelling scenarios were tested using the same number of *epoch*s, with variations in data *splitting* methods, including manual with ratio 80:20 and *random split*s with different ratios 80:20 and 70:30. The model was evaluated using the accuracy, *precision*, *recall*, and F1 score metrics derived from the *confusion matrix*. The best performance was achieved in Scenario 3 (*random split*, 70:30), with an accuracy of 92.93%, a *precision* of 93.89%, a *recall* of 92.83%, and an F1 score of 92.88%. Visualisation of the training *Loss* and *Learning curve* indicated a stable learning process without signs of *overfitting*, demonstrating strong generalisation to unseen data. These results confirm that *Logistic Regression* is an effective and reliable approach to *Malware* detection based on network traffic. It offers a balance between accuracy, *precision*, and generalisation performance that is suitable for real-world implementation in network security systems.

الملخص

أدي، شاريس مولانا سيتيا. 2025. تصنيف البرامج الضارة على حركة مرور الشبكة باستخدام طريقة الانحدار اللوجستي. أطروحة جامعية. برنامج دراسة هندسة المعلوماتية، كلية العلوم والتكنولوجيا، جامعة مولانا مالك إبراهيم مالانج الإسلامية الحكومية.
المشرفون: (I) أجيب حناني، M.T (II) يوهان إيريكا واهيو براكاسا، M.Kom.

الكلمات المفتاحية: البرمجيات الخبيثة، الانحدار اللوجستي (LR)، التصنيف، حركة مرور الشبكة، التحقق المتبادل

أدى الحجم المتزايد لحركة مرور الشبكة، إلى جانب التقدم التكنولوجي، إلى زيادة خطر هجمات البرامج الضارة. لذلك، يعد الكشف المبكر عن حركة المرور المشبوهة أمرًا ضروريًا لتقليل الخسائر الناجمة عن التهديدات السيبرانية. تهدف هذه الدراسة إلى تقييم أداء خوارزمية الانحدار اللوجستي في تصنيف البرامج الضارة بناءً على بيانات حركة مرور الشبكة، بعد معالجتها مسبقًا من خلال اختيار الميزات والترميز والتطبيع. تم تدريب النموذج باستخدام تقنية التحقق المتقاطع 5 أضعاف لقياس اتساق النموذج وقدرته على التعميم. تتضمن هذه الدراسة ثلاثة سيناريوهات بطرق مختلفة لتقسيم البيانات، وهي التقسيم اليدوي بنسبة 80:20 والتقسيم العشوائي بنسبة 80:20 و70:30. يتم تقييم أداء النموذج باستخدام مقاييس الدقة والنوعية والتذكر ودرجة F1 بناءً على مصفوفة الارتباك. يُظهر السيناريو الثالث (التقسيم العشوائي 70:30) أفضل أداء، محققًا دقة 92.93٪ وخصوصية 93.89٪ وتذكر 92.83٪ ودرجة F1 92.88٪. يُظهر منحنى التعلم ومخططات خسارة التدريب أيضًا تعلّمًا مستقرًا دون أي علامات على الإفراط في التجهيز، بالإضافة إلى قدرة تعميم قوية عند التعامل مع بيانات جديدة. تشير هذه النتائج إلى أن الانحدار اللوجستي طريقة فعالة وموثوقة للكشف عن البرامج الضارة استنادًا إلى حركة مرور الشبكة، مما يوفر توازنًا بين الدقة والموثوقية وقابلية التطبيق في أنظمة أمن الشبكات العملية.

BAB I

PENDAHULUAN

1.1 Latar Belakang

Seiring dengan kemajuan pesat dalam bidang teknologi informasi dan komunikasi, ancaman terhadap keamanan siber di Indonesia semakin berkembang, khususnya yang berkaitan dengan serangan *Malware*. Deteksi *Malware* menjadi salah satu tantangan utama dalam menjaga integritas dan kerahasiaan data dalam jaringan computer (Dola Ramalinda et al., 2024). Dengan semakin kompleksnya trafik jaringan dengan berbagai *protokol* komunikasi serta perangkat yang saling terhubung menjadikan proses deteksi serangan *Malware* secara manual menjadi semakin sulit. Oleh karena itu diperlukan penerapan teknik analisis data yang lebih canggih dan otomatis untuk mengatasi tantangan ini secara efektif (Manoppo et al., 2020).

Kejahatan dunia maya semakin beragam dan kompleks, dengan berbagai teknik, alat, dan metode baru yang digunakan oleh penyerang. Fenomena ini memungkinkan para penyerang untuk mengakses sistem yang lebih canggih dan terkelola dengan baik, menciptakan kerusakan tanpa jejak yang jelas. Berdasarkan statistik yang telah dilakukan oleh (Preetha et al., 2021), pada Januari 2021, Google tercatat menemukan sekitar 2 juta situs *web phishing* yang tersebar di internet. Selain itu, pada tahun 2019, sekitar 93,6% *Malware* yang diamati bersifat *polimorfik*, yang berarti *Malware* ini dapat terus mengubah dirinya untuk menghindari deteksi. Laporan dari FBI dan Pusat Pengaduan Kejahatan Internet

juga menunjukkan adanya peningkatan laporan kejahatan dunia maya yang dua kali lipat pada tahun 2020 dibandingkan tahun 2019. Bahkan, *International Data Corporation* (*IDC*) memperkirakan bahwa pengeluaran global untuk solusi keamanan siber akan mencapai $133,7 miliar pada tahun 2022, menunjukkan besarnya ancaman yang ada terhadap sistem di seluruh dunia.

Mengingat peningkatan ancaman ini, perlindungan terhadap sistem jaringan dan data menjadi prioritas utama. Keamanan siber bertujuan untuk menjaga sistem dan jaringan komputer dari kerusakan perangkat keras dan perangkat lunak, pencurian informasi, hingga gangguan layanan yang ada (Firmansyah, 2024). Untuk menghadapi ancaman yang semakin kompleks, dibutuhkan sistem deteksi yang lebih efektif dan efisien. Oleh karena itu teknologi pembelajaran mesin khususnya *Logistic Regression*, dapat dipertimbangkan sebagai solusi untuk mendeteksi serangan *Malware* dengan lebih efisien dan akurat.

Sistem klasifikasi untuk deteksi *Malware* menuntut kecepatan, *akurasi*, dan efisiensi tinggi guna mencegah dampak kerusakan lebih lanjut terhadap sistem dan data yang dimiliki. Namun, dalam implementasinya, metode deteksi yang ada masih menghadapi tantangan besar, terutama dalam mengelola volume trafik jaringan yang masif serta menghadapi keberagaman jenis *Malware* yang terus berkembang. Selain itu, proses pengawasan dan analisis trafik jaringan secara manual sering kali tidak mampu mendeteksi ancaman kompleks secara optimal.

Suricata dipilih sebagai sistem deteksi dan pencegahan intrusi berbasis *open-source*, telah menunjukkan efektivitas dalam memantau dan menganalisis trafik jaringan secara real-time untuk mengidentifikasi potensi ancaman,

sebagaimana ditunjukkan dalam studi oleh (Pinontoan & Sembiring, 2024). Namun demikian, pendekatan yang menggunakan *Suricata* umumnya bersifat *signature-based*, yang menjadikannya kurang adaptif dalam mengenali serangan baru serta memerlukan konfigurasi dan pembaruan aturan secara manual.

Keterbatasan tersebut menegaskan perlunya pengembangan pendekatan yang lebih canggih dan otomatis dalam mendeteksi *Malware,* khususnya melalui metode berbasis *machine learning*. Pendekatan ini dinilai lebih adaptif terhadap pola serangan baru, karena mampu melakukan proses klasifikasi berdasarkan pembelajaran dari data historis tanpa bergantung pada aturan yang statis.

Sebagai solusi untuk mengatasi permasalahan tersebut, berbagai teknik pembelajaran mesin telah dikembangkan untuk menciptakan sistem deteksi yang lebih otomatis, cepat, dan akurat. Salah satu pendekatan yang dapat dilakukan adalah menggunakan metode *Logistic Regression*, yang dikenal karena kemampuannya mengolah data dalam jumlah besar dan membuat prediksi klasifikasi berbasis probabilitas (Hadjar, 2018). Penelitian oleh (Farooq et al., 2022) *Logistic Regression* menempati posisi unggul dalam klasifikasi *Malware*, terutama karena sifat probabilistiknya yang mampu mengelola dataset berukuran besar secara efisien. Dalam studi tersebut menunjukkan bahwa *Logistic Regression* tidak hanya mampu mencapai *akurasi* tinggi secara mandiri, tetapi juga dapat menghasilkan performa maksimal hingga 99% *akurasi* ketika dikombinasikan dengan algoritma lain seperti *Naive Bayes, Random Forest, dan Support Vector Machine*. Keunggulan ini menunjukkan bahwa *Logistic Regression* masih sangat relevan untuk dikaji lebih lanjut. Oleh karena itu, penelitian ini bertujuan

mengeksplorasi kembali potensi *Logistic Regression* dalam mengklasifikasi *Malware* secara efektif dan adaptif terhadap berbagai pola serangan yang dinamis.

Penelitian ini juga dilihat melalui perspektif yang lebih luas, yaitu nilai-nilai moral dan etika yang terkandung dalam ajaran Islam. Integrasi islam yang berlandaskan pada Al-Qur'an Surah Al-Maidah ayat 100 menegaskan pentingnya membedakan antara hal yang baik dan buruk, yang berbunyi :

قُلْ لَّا يَسْتَوِى الْخَبِيْثُ وَالطَّيِّبُ وَلَوْ اَعْجَبَكَ كَثْرَةُ الْخَبِيْثِۚ فَاتَّقُوا اللّٰهَ يٰٓاُولِى الْاَلْبَابِ لَعَلَّكُمْ تُفْلِحُوْنَ ۝١٠٠

"Katakanlah (Nabi Muhammad), "Tidaklah sama yang buruk dengan yang baik meskipun banyaknya yang buruk itu menarik hatimu. Maka, bertakwalah kepada Allah wahai orang-orang yang berakal sehat agar kamu beruntung." (QS. Al-Maidah, 5:100)

Dalam tafsir Ibnu Katsir, ayat yang berbunyi قُلْ لَّا يَسْتَوِى الْخَبِيْثُ وَالطَّيِّبُ dimaknai sebagai penegasan bahwa hal yang buruk dan baik tidak pernah bisa disamakan. Dalam konteks klasifikasi trafik jaringan, ini dapat dipahami bahwa tidak semua trafik yang terlihat aktif (banyak) adalah baik, bisa jadi trafik tersebut merupakan bentuk serangan tersembunyi yang harus dikenali dan dipisahkan dari trafik yang sah. (Tafsir Ibnu Katsir Juz 7 : 98)

Oleh karena itu, upaya perlindungan terhadap sistem informasi menjadi hal yang penting. Penelitian ini, yang menerapkan metode klasifikasi berbasis *Logistic Regression* untuk mendeteksi dan mengelompokkan trafik jaringan menjadi *benign* dan *malicious*, selaras dengan nilai-nilai Islam dalam menjaga keamanan dan keselamatan manusia dari potensi kerusakan.

Pemilihan metode *Logistic Regression* dalam penelitian ini didasarkan pada sejumlah pertimbangan. Algoritma ini dikenal sebagai salah satu teknik *machine*

learning yang efektif untuk klasifikasi biner, seperti dalam konteks deteksi trafik *benign* dan *malicious*. Menurut (Farooq et al., 2022) dalam tinjauan sistematisnya, *Logistic Regression* menunjukkan performa yang unggul dalam mendeteksi *Malware* karena sifat probabilistiknya yang mampu memisahkan dua kelas secara jelas dan efisien serta bisa mencapai *akurasi* 99%. Keunggulan inilah yang menjadikan *Logistic Regression* relevan dan layak untuk digunakan dalam penelitian klasifikasi trafik jaringan berbasis *Malware*.

Selain itu, *Logistic Regression* memiliki keunggulan dalam hal kesederhanaan dan efisiensi komputasi, sehingga sangat sesuai untuk diterapkan dalam sistem yang membutuhkan proses deteksi yang cepat dan real-time. Penelitian yang dilakukan oleh (Rani & Dhavale, 2022) menunjukkan bahwa *Logistic Regression* mampu mencapai *akurasi* sebesar 98,21% dalam klasifikasi *Ransomware*, yang mengindikasikan efektivitas algoritma ini dalam mengidentifikasi ancaman keamanan siber

Dengan demikian, penelitian ini diharapkan dapat memberikan kontribusi dalam pengembangan sistem klasifikasi *Malware* berbasis *machine learning*, khususnya dalam menganalisis trafik jaringan. Dengan memanfaatkan metode *Logistic Regression*, penelitian ini bertujuan untuk meningkatkan efisiensi dalam mengidentifikasi apakah suatu data di jaringan tergolong berbahaya (*malicious*) atau aman (*benign*). *Logistic Regression* dipilih karena kemampuannya dalam menangani data berskala besar dan kompleks secara cepat, sehingga memungkinkan proses klasifikasi dilakukan secara efektif dan akurat dalam lingkungan jaringan yang dinamis.

1.2 Rumusan Masalah

Bagaimana mengukur performa metode *Logistic Regression* dalam proses klasifikasi *Malware* pada data trafik jaringan?

1.3 Batasan Masalah

Batasan masalah yang diterapkan dalam penelitian ini yaitu dataset yang digunakan berupa data trafik jaringan dari laboratorium Stratosphere, Republik Ceko. Dataset berasal dari https://www.stratosphereips.org/datasets-*IoT*23.

1.4 Tujuan Penelitian

Tujuan penelitian ini untuk mengukur performa metode *Logistic Regression* dalam mengklasifikasikan *Malware* pada trafik jaringan.

1.5 Manfaat Penelitian

Penelitian ini bertujuan untuk menghasilkan model klasifikasi *Malware* pada trafik jaringan menggunakan metode *Logistic Regression*. Riset ini diharapkan dapat memberikan manfaat di kemudian hari, yaitu:

1. Membantu praktisi keamanan jaringan dalam mendeteksi aktifitas *Malware* secara otomatis.
2. Menjadi referensi bagi peneliti atau pihak terkait yang tertarik dalam pengembangan sistem klasifikasi *Malware* berbasis *machine learning*.

BAB II

STUDI PUSTAKA

2.1 Penelitian Terkait

Penelitian-penelitian terdahulu menunjukkan berbagai pendekatan dalam penggunaan algoritma *Logistic Regression* (*LR*) untuk klasifikasi *Malware*. Salah satu studi yang relevan Studi oleh (Diana et al., 2022) meneliti klasifikasi file APK Android untuk klasifikasi *Malware* berdasarkan *Fitur* permission dan intent, menggunakan pendekatan analisis statis. Penelitian ini menggunakan tiga algoritma *machine learning* untuk membandingkan performa klasifikasi terhadap berbagai jenis *Malware*. Hasil evaluasi menunjukkan bahwa *Logistic Regression* memperoleh *akurasi* tertinggi sebesar 93% pada pembagian data latih dan uji 70:30, meskipun rata-ratanya adalah 88,75%. Sementara itu, *SVM* dan *Naïve Bayes* masing-masing memperoleh *akurasi* hingga 96,75% dan 97,75%. Penelitian ini juga menunjukkan bahwa kombinasi *Fitur* permission dan intent cukup efektif untuk mendeteksi *Malware*, serta mengindikasikan bahwa *Logistic Regression* dapat digunakan dalam konteks serupa, termasuk dalam pengolahan data trafik jaringan seperti pada penelitian ini.

Penelitian lainnya oleh (Ardelia et al., 2024) mengeksplorasi penggunaan *Logistic Regression* yang digabungkan dengan penyetelan *hyperparameter* untuk meningkatkan *akurasi* model dalam klasifikasi harga ponsel, yang kemudian diadaptasi dalam penelitian ini untuk klasifikasi *Malware*. Dengan pembagian data latih dan uji 90:10, model ini mencapai *akurasi* 98%, menunjukkan bahwa *Logistic*

Regression, dengan penyetelan yang tepat, dapat menjadi alat yang efektif dalam mengklasifikasi *Malware*.

Selain itu, terdapat studi dari (Handoyo et al., 2022) yang mengkaji implementasi metode *Logistic Regression* dalam bidang kesehatan masyarakat. Penelitian ini secara khusus meneliti efektivitas pendekatan feature selection berbasis uji *Chi-Square* untuk memilih *Fitur* yang relevan dan independen, serta membandingkan performa dua algoritma optimasi parameter, yaitu *Newton Raphson dan Gradient Descent*. Hasil penelitian menunjukkan bahwa model *Logistic Regression* yang dibangun dengan algoritma *Gradient Descent* menghasilkan performa yang lebih unggul dibandingkan *Newton Raphson*, dengan *akurasi* sebesar 98,54% dan *F1-score* 97,64%, sementara model dengan *Newton Raphson* hanya mencapai *akurasi* 86,34% dan *F1-score* 72,55%. Studi ini menunjukkan bahwa *Logistic Regression*, jika dioptimalkan dengan algoritma *Gradient Descent*, dapat mencapai performa tinggi dalam klasifikasi biner, termasuk berpotensi diterapkan pada klasifikasi trafik jaringan yang mengindikasikan *Malware*.

Penelitian lain oleh (Wijaya & Santoso, 2021) yang membandingkan performa algoritma *Naive Bayes* dan *Logistic Regression* dalam klasifikasi aplikasi Android sebagai *Malware* atau non-*Malware*, dengan pendekatan analisis statis menggunakan *Fitur* permission dan intent. Penelitian ini menggunakan teknik *validasi* silang 10-*Fold Cross Validation* dalam proses evaluasi model, dan hasilnya menunjukkan bahwa *Logistic Regression* secara konsisten menghasilkan *akurasi* lebih tinggi sebesar 91,8% dibandingkan *Naive Bayes* yang hanya mencapai 87,7%.

Hal ini membuktikan bahwa *Logistic Regression* tidak hanya unggul dalam *akurasi* mentah, tetapi juga lebih stabil dan andal saat diuji menggunakan teknik *Cross Validation*, menjadikannya pilihan yang lebih efektif dalam deteksi *Malware* berbasis *Fitur* statis.

Dan Terakhir, studi yang dilakukan oleh (Alharbi et al., 2022) secara khusus menggunakan dataset *IoT-23*, yang juga digunakan dalam penelitian ini, untuk membangun model prediksi perangkat lunak berbahaya (*Malware*) dalam lingkungan *Internet of Things* (*IoT*). Penelitian ini menggunakan berbagai algoritma klasifikasi supervised learning, seperti *Decision Tree (DT), Random Forest (RF), Support Vector Machine (SVM), Naïve Bayes (NB), dan K-Nearest Neighbor (KNN)*. Hasil eksperimen menunjukkan bahwa algoritma *Random Forest* memberikan *akurasi* tertinggi yaitu sebesar 98,48%, diikuti oleh *SVM* dan *KNN* dengan *akurasi* lebih dari 96%, sedangkan *Logistic Regression* tidak digunakan dalam studi ini. Meskipun metode yang digunakan berbeda, penelitian ini menunjukkan bahwa dataset *IoT-23* sangat relevan dan mampu mendukung pembangunan model prediksi *Malware* yang efektif, sehingga memperkuat landasan penggunaan dataset yang sama dalam penelitian ini.

Pada Tabel 2.1 berikut, menunjukkan daftar penelitian-penelitian yang berkaitan dengan metode dan penelitian yang sedang dilakukan penulis.

Tabel 2. 1 Penelitian Terkait

No	Peneliti (Tahun)	Judul	Metode	Hasil
1	Diana, Richardus Eko Indrajit, Erick Dazki (2022)	Komparasi Algoritma Naïve Bayes, *Logistic Regression* dan Support Vector Machine pada Klasifikasi File Application Package Kit Android *Malware*	*Naïve Bayes, Logistic Regression, SVM*	Hasil evaluasi Dari *Logistic Regression* memperoleh *akurasi* tertinggi sebesar 93% pada skenario 70:30, meskipun rata-ratanya adalah 88,75%. Sementara itu, SVM dan Naïve Bayes

No	Peneliti (Tahun)	Judul	Metode	Hasil
				memperoleh *akurasi* hingga 96,75% dan 97,75%.
2	Danika Najwa Ardelia, Hilda Desfianty Arifin, Sena Daniswara, Anggraini Puspita Sari (2024)	Klasifikasi Harga Ponsel Menggunakan Algoritma *Logistic Regression*	*Logistic Regression*	Evaluasi model pada perbandingan data 80:20 menghasilkan *akurasi* sebesar 97%, dan untuk perbandingan data *Split* 90:10 mencapai *akurasi* 98%.
3	Samingun Handoyo, Nandia Pradianti, Waego Hadi Nugroho, Yusnita Julyarni Akri (2022)	A Heuristic Feature Selection in *Logistic Regression* Modeling with Newton Raphson and *Gradien*t Descent Algorithm	*Logistic Regression*	*Logistic Regression* yang dibangun dengan algoritma *Gradient Descent* memperoleh *akurasi* sebesar 98,54% sementara dengan Newton Raphson hanya mencapai *akurasi* 86,34%.
4	Andreas Putra Wijaya, Handri Santoso (2021)	Komparasi Performansi Algoritma *Naive Bayes* dan *Logistic Regression* pada *Malware* Android	*Naïve Bayes*, *Logistic Regression*	*Logistic Regression* mencapai *akurasi* 91,37%, namun jika menggunakan *Cross Validation* meningkat menjadi 94,28%.
5	Abdulmohsen Alharbi, Md. Abdul Hamid, Husam Lahza (2022)	Predicting *Malicious* Software in *IoT* Environment Based on *Machine learning* and Data Mining Techniques	*Decision Tree, Random Forest, KNN, Naïve Bayes, SVM*	Penelitian ini menggunakan berbagai metode *machine learning* dan mendapatkan beberapa hasil *akurasi* dari dataset *IOT-23*.

Sebagai upaya untuk menyempurnakan celah yang masih ada dalam penelitian sebelumnya, studi ini mengimplementasikan metode *Logistic Regression* pada dataset yang sama karena belum diuji secara khusus dengan metode tersebut. Pendekatan ini dirancang secara sistematis untuk menilai kemampuan *Logistic Regression* dalam mendeteksi lalu lintas jaringan yang mengindikasikan aktivitas *Malware*. Validasi model dilakukan melalui teknik *Cross Validation* guna memastikan hasil yang stabil dan akurat secara empiris. Selain itu, model dioptimalkan menggunakan model *Gradient Descent* berdasarkan teori dan bukti yang menunjukkan keunggulannya dalam meningkatkan kinerja klasifikasi biner.

Penelitian ini juga secara khusus berfokus pada peningkatan *akurasi Logistic Regression* agar mampu bersaing dengan metode lain yang lebih unggul, melalui penerapan strategi pembagian data dan penyesuaian parameter yang dirancang secara logis dan relevan. Dengan demikian, penelitian ini tidak hanya bersifat aktual, tetapi juga memberikan kontribusi teoritis dan praktis dalam pengembangan sistem deteksi *Malware* berbasis pembelajaran mesin.

2.2 Klasifikasi

Klasifikasi merupakan salah satu teknik dalam pembelajaran mesin (machine learning) yang bertujuan untuk mengelompokkan data ke dalam kategori atau kelas tertentu berdasarkan pola atau ciri khas yang dimiliki oleh data tersebut. Dalam konteks keamanan jaringan, klasifikasi digunakan untuk membedakan antara trafik jaringan yang tergolong berbahaya (*malicious*) dan normal (*benign*).

Berbagai algoritma telah diterapkan dalam penelitian sebelumnya untuk tugas ini. (Nafis et al., 2024) membandingkan algoritma Decision Tree dan K-Nearest Neighbor (KNN) dalam mengklasifikasikan serangan pada jaringan IoT menggunakan dataset Edge-IIoTset. Hasil penelitian menunjukkan bahwa Decision Tree memiliki performa yang lebih baik dibandingkan KNN, dengan selisih nilai *presisi*, *recall*, *F1-score*, dan *akurasi* masing-masing sebesar 0.15, 0.18, 0.17, dan 0.08 dalam klasifikasi biner. Sementara itu, (Rijal et al., 2022) mengevaluasi efektivitas metode seleksi *Fitur* untuk deteksi aktivitas trojan pada jaringan. Penelitian ini mengkaji lima algoritma klasifikasi populer: *Decision Tree, Random Forest, KNN, Naïve Bayes, dan AdaBoost*. Hasil studi menunjukkan bahwa algoritma *Decision Tree* menghasilkan *akurasi* tertinggi sebesar 99% dan waktu

prediksi tercepat yaitu 0,0033 detik. Temuan ini menunjukkan bahwa algoritma klasifikasi yang tepat dapat meningkatkan efisiensi dan *akurasi* sistem deteksi *Malware* secara signifikan.

Dalam penelitian ini, pendekatan yang digunakan berbeda dengan studi sebelumnya, yaitu dengan menerapkan *Logistic Regression* yang dioptimasi menggunakan algoritma *Gradient Descent.* Pendekatan ini dipilih karena *Logistic Regression* memiliki keunggulan dalam interpretabilitas dan efisiensi komputasi, serta cocok untuk klasifikasi biner pada data trafik jaringan. Selain itu, penggunaan *Gradient Descent* memungkinkan optimasi parameter model secara efektif, terutama dalam skenario data berskala besar yang diproses menggunakan PySpark.

2.3 Trafik Jaringan

Trafik jaringan atau lalu lintas data merujuk pada aliran paket informasi yang dipertukarkan antarperangkat melalui jaringan. Lalu lintas ini memainkan peran krusial dalam proses pemantauan, kontrol, dan pertukaran informasi pada ekosistem IoT yang saling terhubung. Berdasarkan sifat dan tujuan pengirimannya, trafik data pada jaringan IoT dapat diklasifikasikan ke dalam dua kategori utama, yaitu *benign* trafik dan *malicious* trafik.

Benign trafik merupakan jenis trafik jaringan yang berasal dari aktivitas sah dan normal perangkat, seperti sensor suhu, kamera pengawas, atau perangkat pintar lainnya yang menjalankan fungsi sesuai dengan spesifikasinya. Trafik ini menunjukkan pola komunikasi reguler tanpa indikasi gangguan atau ancaman keamanan. Paket data *benign* tidak mengandung upaya merusak sistem, mencuri informasi, ataupun mengganggu kestabilan layanan.

Sebaliknya, *malicious* trafik adalah jenis lalu lintas jaringan yang dihasilkan oleh perangkat atau entitas yang telah terinfeksi *malicious software* (*Malware*). Paket data ini bertujuan untuk mengeksploitasi sistem, merusak jaringan, mencuri data, atau melumpuhkan layanan melalui berbagai metode serangan, seperti *port scanning*, *botnet communication*, atau *Distributed Denial of Service* (*DDoS*). Trafik ini menjadi indikator penting dalam mendeteksi adanya intrusi atau aktivitas siber berbahaya dalam sistem.

Salah satu contoh nyata dari *malicious* trafik adalah aktivitas yang ditimbulkan oleh *Malware Mirai*, yaitu perangkat lunak berbahaya yang secara khusus menargetkan perangkat *IoT* dengan kredensial lemah. Setelah berhasil menginfeksi, perangkat akan dikendalikan sebagai bagian dari botnet dan digunakan untuk mengirimkan lalu lintas data dalam jumlah besar ke target tertentu sebagai bagian dari serangan *DDoS*. Trafik ini tidak hanya membebani jaringan, tetapi juga berisiko mengganggu layanan *vital* secara *masif*. (Sandriana et al., 2022)

Oleh karena itu, deteksi dan klasifikasi trafik data pada jaringan IoT menjadi langkah strategis dalam menjaga stabilitas dan keamanan sistem. Penggunaan algoritma pembelajaran mesin, seperti *Logistic Regression*, menjadi pendekatan yang adaptif dan efisien dalam mengenali pola-pola trafik berbahaya serta membedakannya dari trafik normal. Penelitian dalam bidang ini terus berkembang guna mengantisipasi serangan siber yang semakin kompleks dan dinamis.

2.4 *Logistic Regression* (LR)

Logistic Regression (LR) merupakan salah satu algoritma pembelajaran mesin yang digunakan untuk menyelesaikan masalah klasifikasi, khususnya ketika

variabel target bersifat kategorikal. Berbeda dengan *regresi linear* yang digunakan untuk memprediksi nilai kontinu, *Logistic Regression* memodelkan hubungan linier antara variabel input dan probabilitas suatu kelas sebagai *output*. Algoritma ini banyak digunakan dalam berbagai bidang, seperti kesehatan, keuangan, dan keamanan siber termasuk dalam deteksi *Malware* berbasis analisis trafik jaringan.

Dalam konteks penelitian ini, *Logistic Regression* diterapkan untuk mengklasifikasikan trafik jaringan ke dalam dua kategori utama, yaitu *benign* (normal) dan *malicious* (berbahaya). Proses klasifikasi dilakukan dengan membentuk sebuah model matematis yang memproyeksikan input *Fitur* ke dalam *output probabilistik* melalui fungsi *aktivasi*. Karena dalam penelitian ini digunakan pendekatan klasifikasi multikelas (termasuk beberapa tipe serangan *Malware*), maka fungsi *aktivasi* yang digunakan *Softmax*.

Berbagai penelitian telah mengkaji efektivitas *Logistic Regression* dalam mendeteksi anomali pada sistem keamanan jaringan. Studi yang dilakukan oleh (Farooq et al., 2022) menunjukkan bahwa *Logistic Regression* memiliki *akurasi* yang tinggi yaitu sekitar 99% dalam mendeteksi *Malware* dibandingkan algoritma lain, seperti *Random Forest dan Support Vector Machine*. Selain itu, penelitian oleh (Primadya et al., 2024) mengungkapkan bahwa *Logistic Regression* dapat dioptimalkan untuk mendeteksi serangan *Denial of Service (DoS)* pada sistem keamanan *IoT,* dengan hasil yang cukup menjanjikan

Dalam konteks penggunaan metode *Logistic Regression (LR)* untuk klasifikasi, beberapa penelitian telah mengidentifikasi keterbatasan algoritma ini, khususnya dalam menangani dataset dengan hubungan non-linear antar *Fitur*.

Sebagai contoh, penelitian oleh Sutarman et al. (2024) menyoroti tantangan yang dihadapi *LR* saat diterapkan pada dataset dengan jumlah *Fitur* yang besar dan distribusi kelas yang tidak seimbang. Dalam penelitian tersebut, tiga dataset digunakan untuk mengevaluasi kinerja *LR*. Hasilnya menunjukkan bahwa *LR* mengalami penurunan performa ketika dihadapkan pada kompleksitas *Fitur* yang tinggi dan ketidakseimbangan kelas. Untuk mengatasi masalah ini, peneliti menerapkan teknik *Recursive Feature Elimination (RFE)* untuk seleksi *Fitur* dan *Synthetic Minority Over-sampling Technique (SMOTE)* untuk penyeimbangan kelas. Dengan kombinasi LR, RFE, dan SMOTE, model mencapai AUC sebesar 93%, yang lebih baik dibandingkan dengan model lain. Disamping itu penelitian yang dilakukan oleh (Rani & Dhavale, 2022) menggunakan metode optimasi seperti kombinasi *Logistic Regression* dengan *deep learning* yang sering digunakan untuk meningkatkan performa model dalam mendeteksi *Malware* yang lebih kompleks (Rani & Dhavale, 2022).

Secara matematis, metode *Logistic Regression* dalam klasifikasi multikelas bekerja dengan menghitung kombinasi linier antara *Fitur* input dan bobot, yang kemudian diteruskan ke dalam fungsi aktivasi *Softmax* untuk menghasilkan distribusi probabilitas terhadap setiap kelas.

Rumus dasar kombinasi linier dalam *Logistic Regression* ditulis pada rumus 2.1: (Le & Nguyen, 2024):

$$z = W^{T} X + b \tag{2.1}$$

Keterangan:
z : hasil kombinasi linier variabel input sebelum diterapkan fungsi *aktivasi*,
W : *vektor bobot* yang menunjukkan pengaruh masing-masing *Fitur*,
W^T : bentuk transpose dari *vektor bobot* W , yang digunakan untuk melakukan perkalian

matriks dengan X,
X : *vektor Fitur* input yang digunakan untuk klasifikasi,
b : *bias* yang berfungsi untuk mengoptimalkan keputusan model.

Setelah mendapatkan nilai z untuk seluruh kelas, fungsi aktivasi *Softmax* digunakan untuk mengonversi skor logit menjadi distribusi probabilitas yang dapat ditafsirkan pada rumus 2.2:

$$P(y_i|X) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \quad (2.2)$$

Keterangan:

$P(y_i|X)$: probabilitas suatu sampel X termasuk ke dalam kelas i ,
k : jumlah kelas dalam dataset,
e^{z_i} : eksponensial dari nilai logit untuk kelas i ,
$\sum_{j=1}^{k} e^{z_j}$: untuk Normalisasi probabilitas agar berada dalam rentang [0,1].

Fungsi *Softmax* menjamin bahwa semua probabilitas berada dalam rentang [0, 1] dan totalnya bernilai 1. Kelas dengan probabilitas tertinggi akan menjadi hasil klasifikasi akhir oleh model (Zhang, 2024). Setelah model menghasilkan distribusi probabilitas menggunakan fungsi *Softmax*, langkah selanjutnya dalam proses pemodelan adalah mengoptimalkan parameter bobot dan *bias* untuk meminimalkan kesalahan prediksi. Proses ini dilakukan menggunakan metode *Gradient Descent*, yang bertujuan meminimalkan nilai dari fungsi biaya (*Loss function*) yang digunakan. Dalam penelitian ini, digunakan fungsi *Cross-Entropy Loss* yang sesuai untuk klasifikasi multikelas. Fungsi biaya tersebut dituliskan pada rumus 2.3:

$$J = -\frac{1}{m}\sum_{i=1}^{m}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)] \quad (2.3)$$

Keterangan:
J : nilai *Loss* yang harus diminimalkan,
m : jumlah sampel dalam dataset,
y_i : label sebenarnya dari sampel ke-i ,
$\hat{y}_i$: probabilitas prediksi untuk kelas positif.

Fungsi ini akan bernilai rendah jika probabilitas prediksi mendekati label sebenarnya, dan akan meningkat jika prediksi menyimpang dari label yang benar. Oleh karena itu, fungsi ini sangat tepat untuk mengukur performa prediksi probabilistik dari model klasifikasi multikelas seperti *Logistic Regression*.

Untuk mengurangi nilai *Loss* tersebut, parameter W dan b diperbarui secara iteratif menggunakan algoritma *Gradient Descent*. Adapun pembaruan *bobot* dan *bias* dilakukan dengan rumus 2.4 dan 2.5:

$$W \coloneqq W - \alpha \frac{1}{m} \sum \left(X^T \left(y_{pred} - y_{true} \right) \right) \tag{2.4}$$

$$b \coloneqq b - \alpha \frac{1}{m} \sum \left(y_{pred} - y_{true} \right) \tag{2.5}$$

dimana α adalah *learning rate*, yang menentukan seberapa besar perubahan parameter pada setiap iterasi. Proses ini dilakukan secara iteratif selama sejumlah *epoch* tertentu hingga nilai fungsi *Loss* stabil pada titik minimum atau mendekati nilai konvergen. Dengan demikian, model dapat memberikan prediksi yang optimal dan akurat terhadap data uji.

Sebelum model dievaluasi pada data uji akhir dilakukan proses *K-Fold Cross Validation* terlebih dahulu sebagai bentuk *validasi* internal. *Cross Validation* membantu menguji kemampuan *generalisasi* model terhadap berbagai subset data latih, serta mencegah *overfitting* yang mungkin terjadi jika model hanya dilatih dan diuji pada satu pembagian data saja. Tahap selanjutnya model akan dievaluasi menggunakan *Confusion matrix* dan *Classification report*.

Menurut (Sutarman et al., 2024) kelebihan dari *Logistic Regression* antara lain:

1. Mudah diinterpretasikan karena berbasis probabilitas.
2. Cepat dalam perhitungan dan efisien untuk dataset berukuran besar.

Sedangkan kekurangan dari *Logistic Regression* antara lain:

1. Kurang optimal dalam menangani hubungan non-linear antar *Fitur*.
2. Kurang efektif pada dataset yang tidak seimbang.

2.5 *Cross Validation*

Cross Validation merupakan teknik evaluasi model dalam pembelajaran mesin yang bertujuan untuk menilai kemampuan *generalisasi* model terhadap data yang belum pernah dilihat sebelumnya. Teknik ini meminimalkan risiko *overfitting*, yaitu kondisi di mana model terlalu menyesuaikan diri dengan data pemodelan sehingga performanya menurun saat dihadapkan pada data baru.

Salah satu metode *Cross Validation* yang paling umum digunakan adalah *K-Fold Cross Validation*. Dalam metode ini, dataset dibagi menjadi K subset atau *Fold* yang berukuran sama. Proses pemodelan dan *validasi* dilakukan sebanyak K (beberapa) kali, di mana pada setiap iterasi, satu *Fold* digunakan sebagai data *validasi*, sementara K-1*Fold* lainnya digunakan sebagai data pemodelan. Setelah semua iterasi selesai, hasil evaluasi dari setiap *Fold* dirata-rata untuk memberikan estimasi performa model yang lebih akurat dan stabil.

Metode *K-Fold Cross Validation* telah diterapkan dalam berbagai penelitian untuk mengevaluasi kinerja model klasifikasi. Sebagai contoh, dalam penelitian oleh (Wijiyanto et al., 2024), *K-Fold Cross Validation* digunakan untuk mengevaluasi kinerja model *Support Vector Machine (SVM), Naïve Bayes, Neural*

Network, dan Decision Tree dalam memprediksi kinerja mahasiswa berdasarkan faktor keluarga. Penelitian tersebut menunjukkan bahwa penggunaan *K-Fold Cross Validation* dapat memberikan estimasi *akurasi* yang lebih stabil dan mengurangi variansi hasil evaluasi model.

Penelitian lain yang dilakukan oleh (Wijaya & Santoso, 2021), *Penerapan K-Fold Cross Validation* dalam penelitian ini menggunakan teknik *validasi* silang 10-*Fold Cross Validation* untuk mengevaluasi kinerja model *Logistic Regression*. Hasil *validasi* menunjukkan bahwa *Logistic Regression* secara konsisten menghasilkan performa yang lebih baik dibandingkan *Naive Bayes* yaitu dengan *akurasi* 91,8% dari 87,7%., yang menunjukkan efektivitas metode tersebut dalam membangun sistem deteksi *Malware* Android.

Keunggulan utama dari *K-Fold Cross Validation* adalah kemampuannya dalam memanfaatkan seluruh data yang tersedia untuk proses pemodelan dan *validasi*, sehingga memberikan estimasi performa model yang lebih andal. Namun, pemilihan nilai K yang tepat sangat penting untuk memastikan keseimbangan antara *bias* dan variansi dalam evaluasi model. Nilai K yang terlalu kecil dapat menghasilkan estimasi yang *bias*, sementara nilai K yang terlalu besar dapat meningkatkan variansi dan beban komputasi.

Dengan demikian, *Cross Validation*, khususnya *K-Fold Cross Validation*, merupakan teknik evaluasi yang mendasar dan sangat penting dalam proses pengembangan model pembelajaran mesin, karena mampu memberikan gambaran performa model secara lebih menyeluruh dan objektif.

2.6 *Confusion matrix & Classification report*

Setelah proses pemodelan selesai, evaluasi kinerja model dilakukan menggunakan dua pendekatan utama, yaitu *Confusion matrix* dan *Classification report*. Kedua metode ini digunakan untuk menggambarkan performa model klasifikasi dalam membedakan antara trafik jaringan yang termasuk kategori *malicious* dan normal.

Confusion matrix merupakan sebuah Tabel yang menampilkan perbandingan antara label aktual dan hasil prediksi dari model. Dari matriks ini diperoleh empat komponen utama yang dijelaskan pada Tabel 2.2 berikut: (Ramadhani & Suryono, 2024)

Tabel 2. 2 Tabel *Confusion matrix*

Kelas	Prediksi Positif	Prediksi Negatif
Aktual Positif	True Positive (TP)	*False negative* (FN)
Aktual Negatif	*False positive* (FP)	True Negative (TN)

Keterangan:

- True Positive (TP) : Data aktual positif dan diprediksi sebagai positif.
- True Negative (TN) : Data aktual negatif dan diprediksi sebagai negatif.
- *False positive* (FP) : Data aktual negatif tetapi diprediksi sebagai positif.
- *False negative* (FN) : Data aktual positif tetapi diprediksi sebagai negatif.

Dari keempat nilai tersebut, beberapa metrik evaluasi dapat dihitung sebagai berikut: (Ramadhani & Suryono, 2024):

1. *Akurasi* (*Accuracy*): Mengukur persentase prediksi yang benar dibandingkan dengan total jumlah data uji.

$$Akurasi = \frac{TP+TN}{TP+TN+FP+FN} \quad (2.6)$$

Akurasi mengukur persentase prediksi yang benar dibandingkan dengan seluruh jumlah data uji. Ini mencakup prediksi benar terhadap kelas positif

(*Malware*) dan negatif (*benign*). *Akurasi* tinggi menunjukkan bahwa model dapat mengklasifikasikan data dengan benar dalam sebagian besar kasus.

2. *Presisi* (*Precision*): Mengukur ketepatan model dalam mengidentifikasi *Malware*, dihitung dengan rumus 2.7:

$$Presisi = \frac{TP}{TP+FP} \tag{2.7}$$

Presisi mengukur seberapa tepat model dalam mengidentifikasi *Malware*. Nilai ini menunjukkan proporsi prediksi positif (*Malware*) yang benar-benar positif.

Jika *presisi* tinggi, maka jumlah prediksi *false positive* (FP) rendah. Ini penting dalam konteks keamanan jaringan, karena kesalahan menandai koneksi yang aman sebagai *Malware* bisa mengganggu sistem.

3. *Recall* (*Sensitivity*): Menunjukkan seberapa banyak trafik *Malware* yang berhasil dideteksi oleh model, dihitung dengan rumus 2.8:

$$Recall = \frac{TP}{TP+FN} \tag{2.8}$$

Recall mengukur kemampuan model dalam menemukan semua trafik berbahaya (*Malware*) yang benar-benar ada di data. Artinya, dari semua yang seharusnya terdeteksi, berapa yang berhasil terdeteksi oleh model.

Recall tinggi berarti model tidak banyak melewatkan *Malware* (FN sedikit), sehingga sangat penting dalam konteks deteksi ancaman yang harus dihindari sebisa mungkin.

4. *F1-score*: Mengombinasikan *presisi* dan *recall* dalam satu metrik keseimbangan, dengan rumus 2.9:

$$F1\text{-}score = 2 \times \frac{Presisi \times Recall}{Presisi + Recall} \quad (2.9)$$

F1-score adalah harmonic mean (rata-rata harmonis) dari *presisi* dan *recall*. Nilai ini digunakan untuk mencari keseimbangan antara kedua metrik, terutama jika ada trade-off antara *presisi* dan *recall*.

Nilai *F1-score* tinggi menunjukkan bahwa model tidak hanya akurat dalam mendeteksi *Malware* (*presisi* tinggi), tetapi juga tidak banyak melewatkan ancaman (*recall* tinggi).

Evaluasi dengan *Confusion matrix* ini sangat penting dalam konteks klasifikasi *Malware* karena dapat menunjukkan apakah model cenderung sering melakukan kesalahan prediksi.

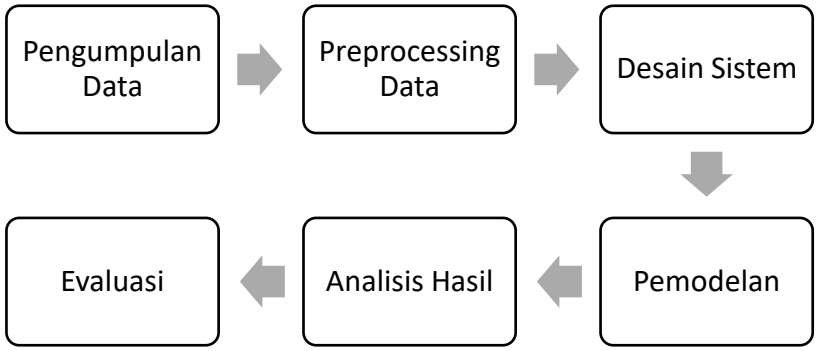
Sementara itu, *Classification report* adalah alat bantu evaluasi tambahan yang menyajikan nilai *Precision*, *recall*, *F1-score*, dan support untuk masing-masing kelas (normal dan *malicious*) dalam bentuk Tabel yang lebih terstruktur. Dalam penelitian ini, *Classification report* digunakan sebagai *Visualisasi* hasil dari *Confusion matrix* dan sebagai sarana untuk mempermudah analisis kinerja model secara menyeluruh.

BAB III

DESAIN DAN IMPLEMENTASI

3.1 Prosedur Penelitian

Pada penelitian ini, prosedur yang dilakukan untuk mendeteksi *Malware* dalam lalu lintas jaringan menggunakan model *Logistic Regression* melalui serangkaian tahapan yang terstruktur dan sistematis dalam gambar 3.1.



Gambar 3. 1 Prosedur Penelitian

Proses penelitian dimulai dengan pengumpulan data, dimana data yang digunakan merupakan data sekunder yang diperoleh dari dataset yang telah tersedia. Dataset ini mencakup berbagai parameter yang berkaitan dengan lalu lintas jaringan (Flynn & Olukoya, 2025). Seluruh data ini akan digunakan untuk membangun model yang mampu mengidentifikasi potensi serangan *Malware* dalam sistem jaringan.

Tahapan Selanjutnya *Preprocessing* data, yang akan dilakukan dengan beberapa langkah penting untuk memastikan data dalam kondisi yang siap digunakan dalam model pembelajaran mesin. Proses ini meliputi pembersihan data

dengan menghapus kolom-kolom yang tidak relevan, serta penanganan nilai yang hilang atau *null* yang terdapat pada dataset. Data yang sudah dibersihkan kemudian diproses lebih lanjut dengan transformasi *Fitur* menggunakan teknik *StringIndexer* dan *OneHotEncoder*. Selanjutnya, *Fitur-Fitur* yang ada di*Normalisasi* menggunakan *MinMaxScaler* untuk memastikan bahwa semua *Fitur* memiliki rentang yang sama, yang akan membantu model dalam melakukan pemodelan secara efisien.

Setelah data siap, tahap berikutnya adalah pemodelan. Pada tahap ini, model *Logistic Regression* diimplementasikan menggunakan fungsi *aktivasi Softmax* untuk klasifikasi multikelas. Pemodelan dilakukan dengan menggunakan algoritma *Gradient Descent*, yang digunakan untuk meminimalkan fungsi biaya (*Loss function*) dan mengoptimalkan parameter model, seperti *bobot* dan *bias*. Proses pemodelan dilakukan dalam beberapa *epoch* untuk memastikan model dapat belajar dengan baik dari data yang diberikan.

Setelah model dilatih, langkah selanjutnya tentunya analisis hasil dan evaluasi. Evaluasi dilakukan dengan menggunakan data uji, yang sebelumnya telah dipisahkan dari data latih dengan pembagian 80:20. Evaluasi ini dilakukan untuk mengukur kinerja model dengan menggunakan *Confusion matrix* dan *Classification report* yang mana untuk mem*Visualisasi*kan hasil evaluasi berupa skor-skor evaluasi seperti *akurasi*, *precision*, *recall*, dan *F1-score* untuk mengetahui seberapa baik model dalam mendeteksi *Malware* pada trafik jaringan.

Dengan mengikuti prosedur tersebut, penelitian ini bertujuan untuk mengembangkan model yang akurat dalam mendeteksi *Malware* berdasarkan pola

yang terdapat dalam lalu lintas jaringan. Semua tahapan, mulai dari pengumpulan data hingga analisi hasil dan evaluasi, bertujuan untuk memastikan bahwa model yang dihasilkan dapat diandalkan dan mampu mendeteksi *Malware* dengan tingkat *akurasi* yang baik.

3.1.1 Pengumpulan Data

Dataset yang digunakan dalam penelitian ini diperoleh dari situs resmi stratosphereips.org/datasets-*IoT*23. *Stratosphere Laboratory* merupakan bagian dari *Artificial Intelligence Center* (*AIC*) di Fakultas Teknik Elektro, Universitas Ceko Teknikal (CTU) di Praha. Dataset ini pertama kali dipublikasikan oleh (Garcia et al., 2020) dalam proyek *Stratosphere IPS*, yang bertujuan untuk menyediakan data lalu lintas jaringan *Internet of Things* (*IoT*) yang telah terinfeksi *Malware* maupun lalu lintas normal sebagai bahan studi keamanan jaringan.

Dataset yang digunakan berisi 10.390 data koneksi jaringan yang telah dilabeli, dengan komposisi seimbang antara lalu lintas *benign* dan *malicious* (50:50). Keseimbangan ini penting untuk menghindari *bias* dalam proses pemodelan model klasifikasi, serta memastikan bahwa model dapat belajar secara adil dari kedua kelas. Dataset mencerminkan pola-pola nyata dari serangan *Malware* terhadap perangkat *IoT*, khususnya dari jenis *Mirai*. Dataset ini sangat relevan dan representatif untuk tugas klasifikasi lalu lintas jaringan dengan pendekatan pembelajaran mesin. Dataset ini terdiri dari beberapa *Fitur* penting yang akan digunakan dalam model pembelajaran mesin. Tabel 3.1 menjelaskan *Fitur-Fitur* yang terdapat dalam dataset:

Tabel 3. 1 *Fitur-Fitur* dalam Dataset

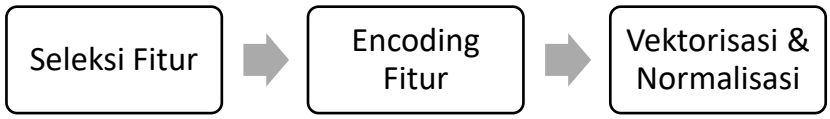
No	*Fitur*	Deskripsi
1	*ts*	*Timestamp* saat koneksi jaringan terjadi.
2	*uid*	Identifikasi unik dari setiap koneksi.
3	*id.orig_h*	Alamat IP asal dari koneksi.
4	*id.orig_p*	Port asal dari koneksi.
5	*id.resp_h*	Alamat IP tujuan dari koneksi.
6	*id.resp_p*	Port tujuan dari koneksi.
7	*proto*	*Protokol* jaringan yang digunakan (misalnya TCP, UDP, ICMP).
8	*service*	Jenis layanan yang digunakan dalam koneksi jaringan.
9	*duration*	Lama waktu koneksi berlangsung dalam satuan detik.
10	*orig_bytes*	Jumlah byte yang dikirim oleh pihak pengirim (originator).
11	*resp_bytes*	Jumlah byte yang dikirim oleh pihak penerima (responder).
12	*conn_state*	Status koneksi (misalnya S0, S1, REJ).
13	*local_orig*	Indikator apakah koneksi berasal dari jaringan lokal.
14	*local_resp*	Indikator apakah koneksi diterima oleh host lokal.
15	*missed_bytes*	Jumlah byte yang hilang selama koneksi.
16	*history*	Rangkaian status koneksi yang terjadi selama sesi berlangsung.
17	*orig_pkts*	Jumlah paket yang dikirim oleh pengirim.
18	*orig_ip_bytes*	Jumlah byte IP yang dikirim oleh pengirim.
19	*resp_pkts*	Jumlah paket yang diterima oleh penerima.
20	*resp_ip_bytes*	Jumlah byte IP yang diterima oleh penerima.
21	*tunnel_parents*	Indikasi apakah koneksi melalui sebuah tunnel.
22	*label*	Label utama yang menunjukkan apakah koneksi normal atau *malicious*.
23	*detailed-label*	Informasi lebih rinci mengenai jenis serangan *Malware* dalam koneksi.

Setelah memahami struktur dataset, tahap berikutnya dalam penelitian ini adalah *Preprocessing* data, yang mencakup Seleksi *Fitur*, *Encoding Fitur*, dan *Normalisasi* agar sesuai dengan kebutuhan model pembelajaran mesin. Proses ini dilakukan untuk memastikan bahwa data yang digunakan tidak mengandung nilai yang hilang, duplikat, atau informasi yang tidak relevan sehingga model yang

dikembangkan dapat bekerja secara optimal dalam mendeteksi ancaman *Malware* pada trafik jaringan.

3.1.2 *Preprocessing* Data

Tahap *Preprocessing* data memiliki peran krusial dalam penelitian ini, karena bertujuan untuk memastikan bahwa data yang digunakan dalam model memiliki kualitas yang optimal serta sesuai untuk dianalisis lebih lanjut. Proses ini dilakukan untuk menseleksi data dari nilai yang hilang, duplikasi, serta informasi yang tidak relevan, sekaligus mengonversinya ke dalam format yang dapat diproses oleh model *Logistic Regression*. Gambar 3.2 menggambarkan tahapan-tahapan yang dilakukan dalam proses *Preprocessing* data pada penelitian ini.

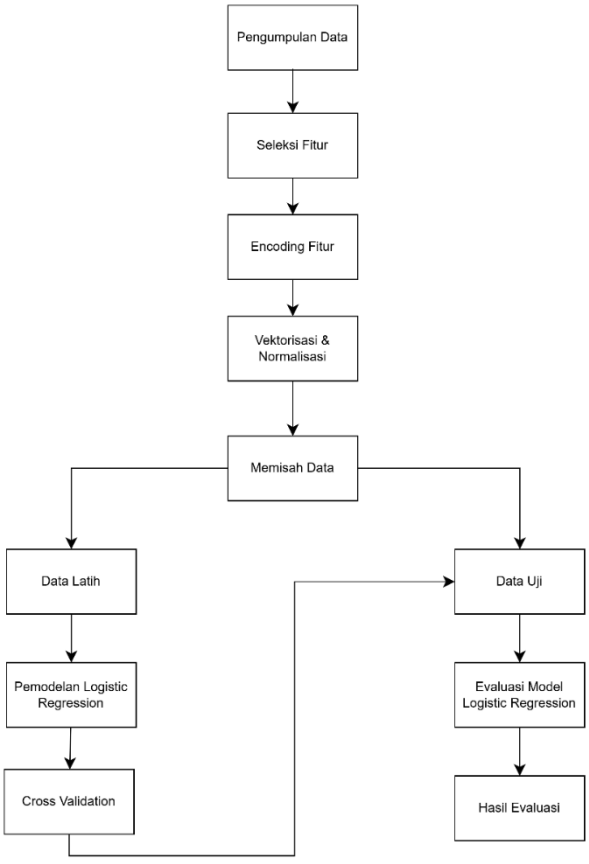


Gambar 3. 2 Proses *Preprocessing*

Proses *Preprocessing* dilakukan secara sistematis agar model dapat beroperasi secara optimal dalam mengklasifikasikan lalu lintas jaringan menjadi normal atau *malicious*. Tahap ini sangatlah penting karena kualitas data akan sangat berpengaruh terhadap hasil evaluasi performa model. Dengan data yang telah diproses dengan baik, diharapkan model dapat memberikan hasil klasifikasi yang lebih akurat dan dapat diandalkan dalam mendeteksi *Malware* pada lalu lintas jaringan.

3.1.3 Desain Sistem

Dalam penelitian ini, digunakan sistem yang dikembangkan bertujuan untuk mendeteksi *Malware* pada lalu lintas jaringan dengan mengimplementasikan metode *Logistic Regression*. *Logistic Regression* dipilih karena kemampuannya dalam menangani masalah klasifikasi biner, yang dalam konteks penelitian ini berarti mengklasifikasikan trafik jaringan sebagai normal atau *malicious* berdasarkan pola koneksi yang tercatat dalam dataset. Gambar 3.3 menggambarkan alur desain sistem dari penelitian ini.



Gambar 3. 3 Desain Sistem

3.1.3.1 Seleksi *Fitur*

Proses seleksi *Fitur* merupakan langkah krusial dalam tahapan *Preprocessing* yang bertujuan untuk menyaring atribut-atribut yang tidak relevan agar model klasifikasi dapat belajar dari informasi yang benar-benar penting. Dalam konteks penelitian ini, proses seleksi dilakukan untuk meningkatkan *akurasi*, efisiensi komputasi, serta mengurangi risiko *overfitting* dalam klasifikasi trafik jaringan menggunakan *Logistic Regression*. Dataset awal yang digunakan memiliki sejumlah atribut, Namun, tidak semua *Fitur* memberikan kontribusi signifikan terhadap proses klasifikasi. Oleh karena itu, dilakukan penghapusan terhadap beberapa *Fitur* yang dianggap tidak relevan atau memiliki kualitas data yang rendah.

Fitur uid dihapus karena merupakan identitas unik dari setiap koneksi dan tidak memiliki korelasi langsung dengan label kelas. Empat *Fitur* lainnya, yaitu *id.orig_h, id.orig_p, id.resp_h, dan id.resp_p* juga dihapus karena berisi informasi alamat dan port yang bersifat dinamis dan tidak generalisatif, sehingga berisiko menyebabkan *overfitting* jika tetap digunakan. Selanjutnya, *Fitur service, local_orig, dan local_resp* dieliminasi karena memiliki banyak nilai kosong atau tidak tersedia secara konsisten dalam seluruh dataset. Atribut *tunnel_parents* juga dihapus karena mayoritas nilainya kosong dan tidak memberikan kontribusi yang berarti dalam klasifikasi. Selain itu, *Fitur history* yang berisi kombinasi *flag* koneksi juga dihapus karena variasi nilainya sangat kompleks dan sulit diinterpretasi secara langsung oleh model. Terakhir, *Fitur detailed-label* tidak

digunakan karena hanya merupakan bentuk deskriptif dari *label*, yang sudah mencukupi sebagai target klasifikasi dua kelas, yaitu *benign* dan *malicious*.

Dengan menghapus *Fitur-Fitur* tersebut, proses pemodelan model menjadi lebih efisien karena hanya menggunakan data yang benar-benar informatif. Proses seleksi ini membantu menyederhanakan kompleksitas dataset dan memperkuat kemampuan model dalam mengenali pola dari trafik jaringan yang termasuk dalam kategori *benign* maupun *malicious*. Tabel 3.2 menunjukkan kondisi data sebelum seleksi *Fitur* dilakukan, sedangkan Tabel 3.3 menyajikan hasil akhir setelah *Fitur-Fitur* tidak relevan dihapus dari dataset.

Tabel 3. 2 Data Sebelum Seleksi *Fitur*

	ts	uid	**id_orig_h**	**id_orig_p**	**id_resp_h**	**id_resp_p**	***proto***	**service**	**duration**	**orig_bytes**	**…**	*conn_state*	missed_bytes	orig_pkts	orig_ip_bytes	resp_pkts	resp_ip_bytes	tunnel_parents	label	detailed_label
46	1,55E+15	CFCGvp4aEVGId8CkPh	192.168.100.113	53292	192.168.100.1	53	tcp	dns	5.005.152	78.0	…	SF	2896.0	93.0	5571.0	92.0	131755.0	-	Benign	-
47	1,55E+15	CvOvGB2WOKVl6Lo3Wf	192.168.1.195	123	147.231.100.5	123	tcp	-	0.000148 2	48.0	…	S0	0.0	3.0	180.0	0.0	0.0	-	Benign	-
48	1,55E+15	C44hHm17HZ3lMqs8Tb	192.168.100.113	123	81.2.254.224	123	tcp	-	NaN	NaN	…	S0	0.0	3.0	180.0	0.0	0.0	-	Malicious	None
49	1,55E+15	CFDwDE15VacFXqJUZ4	192.168.100.113	123	144.48.166.166	123	tcp	-	NaN	NaN	…	S0	0.0	1.0	60.0	0.0	0.0	-	Benign	-
50	1,55E+15	CmYYN04wo2sCcLG5Wk	192.168.1.195	65279	123.59.209.185	80	tcp	-	NaN	NaN	…	S0	0.0	1.0	60.0	0.0	0.0	-	Malicious	None
51	1,55E+15	CxNRRwNJhn2pAKXP1	192.168.1.195	65279	123.59.209.185	80	tcp	-	NaN	NaN	…	S0	0.0	1.0	60.0	0.0	0.0	-	Benign	-
52	1,55E+15	CcnxnK3X0qaNCFIFfj	192.168.100.113	123	5.1.56.123	123	tcp	-	NaN	NaN	…	S0	0.0	3.0	180.0	0.0	0.0	-	Malicious	None
53	1,55E+15	CHi0073vh0QyWm6v11	192.168.100.113	39665	192.168.100.1	53	udp	dns	0.000496	45.0	…	SF	0.0	1.0	76.0	1.0	76.0	-	Benign	-
54	1,55E+15	CrBy6k3Q58tC2vCBw5	192.168.100.113	123	147.231.100.5	123	tcp	-	NaN	NaN	…	S0	0.0	1.0	60.0	0.0	0.0	-	Malicious	None

Tabel 3. 3 Data Setelah Seleksi *Fitur*

	ts	*proto*	conn_state	missed_bytes	orig_pkts	orig_ip_bytes	resp_pkts	resp_ip_bytes	label
46	1,55E+15	tcp	SF	2896.0	93.0	5571.0	92.0	131755.0	*Benign*
47	1,55E+15	tcp	S0	0.0	3.0	180.0	0.0	0.0	*Benign*
48	1,55E+15	tcp	S0	0.0	3.0	180.0	0.0	0.0	*Malicious*
49	1,55E+15	tcp	S0	0.0	1.0	60.0	0.0	0.0	*Benign*
50	1,55E+15	tcp	S0	0.0	1.0	60.0	0.0	0.0	*Malicious*
51	1,55E+15	tcp	S0	0.0	1.0	60.0	0.0	0.0	*Benign*
52	1,55E+15	tcp	S0	0.0	3.0	180.0	0.0	0.0	*Malicious*
53	1,55E+15	udp	SF	0.0	1.0	76.0	1.0	76.0	*Benign*
54	1,55E+15	tcp	S0	0.0	1.0	60.0	0.0	0.0	*Malicious*
55	1,55E+15	tcp	S0	0.0	3.0	180.0	0.0	0.0	*Malicious*
56	1,55E+15	tcp	S0	0.0	1.0	60.0	0.0	0.0	*Benign*

Setelah proses seleksi *Fitur* selesai, dataset menjadi lebih terstruktur dan hanya mencakup informasi yang penting untuk pengembangan model *Logistic Regression*. Tahap berikutnya adalah melakukan *Encoding Fitur*, yang akan dijelaskan lebih detail pada tahap *Preprocessing* selanjutnya.

3.1.3.2 *Encoding Fitur*

Encoding Fitur adalah langkah penting dalam *Preprocessing* data, yang bertujuan untuk mengonversi variabel non-*numerik* menjadi format *numerik* agar dapat digunakan dalam model *Logistic Regression*. Dalam dataset yang digunakan dalam penelitian ini, terdapat beberapa *Fitur* dengan tipe data kategorikal, seperti *proto* dan *conn_state*. Model pembelajaran mesin tidak dapat memproses data dalam bentuk teks, sehingga *Fitur-Fitur* tersebut perlu diubah menjadi format

numerik dengan menggunakan teknik seperti Label *Encoding* atau *One-Hot Encoding*.

Pada penelitian ini, digunakan metode *One-Hot Encoding*, karena teknik ini dapat menghindari pemberian urutan *numerik* yang dapat mengarah pada interpretasi hierarkis yang salah pada *Fitur* kategorikal. *One-Hot Encoding* mengubah setiap kategori dalam *Fitur* menjadi *vektor* biner, di mana hanya satu elemen yang bernilai 1 (aktif), sementara elemen lainnya bernilai 0. Secara matematis, proses ini dapat digambarkan dengan rumus 3.1 :

$$X_{encoded} = [x_1, x_2, \dots, x_n] \quad (3.1)$$

Keterangan:
$X_{encoded}$: vektor hasil *Encoding*,
- x_i : bernilai 1 jika kategori Fitur sesuai dengan kolom tersebut, dan 0 jika tidak.

Sebagai ilustrasi, *Fitur proto* dalam dataset memiliki tiga nilai unik: TCP, UDP, dan ICMP. Dengan menggunakan *One-Hot Encoding*, *Fitur* ini akan diubah menjadi format biner yang dapat dilihat pada Tabel 3.4.

Tabel 3. 4 Konversi *Fitur proto*

proto	TCP	UDP	ICMP
TCP	1	0	0
UDP	0	1	0
ICMP	0	0	1

Proses serupa diterapkan pada *Fitur conn_state*, yang menggambarkan status koneksi jaringan. Jika dataset memiliki tiga kategori unik, seperti S0, S1, dan REJ, hasil *Encoding*-nya akan terlihat seperti pada Tabel 3.5

Tabel 3. 5 Konversi *Fitur conn_state*

conn_state	*S0*	*S1*	*REJ*
S0	*1*	*0*	*0*
S1	*0*	*1*	*0*
REJ	*0*	*0*	*1*

Hasil aktual dari proses *Encoding* ini dapat dilihat pada Tabel 3.7, adapun data sebelum di *Encoding* ada pada tabel 3.6. *Fitur proto* telah diubah menjadi dua bentuk *numerik* baru, yaitu *proto_indexed* dan *proto_oh_encoded*, dan *Fitur conn_state* menjadi *conn_state_indexed* dan *conn_state_oh_encoded*. *Fitur proto_indexed* dan *conn_state_indexed* merupakan hasil dari Label *Encoding*, dimana setiap kategori diberi label *numerik* sederhana (misalnya TCP = 0.0, UDP = 1.0, dll), sedangkan *proto_oh_encoded* dan *conn_state_oh_encoded* merupakan bentuk akhir hasil *One-Hot Encoding* yang digunakan dalam pemodelan model. Sementara itu, *encoded_label* merupakan hasil *Encoding* terhadap target klasifikasi (label) yang telah dikonversi ke dalam nilai *numerik*, di mana *benign* direpresentasikan dengan 2.0 dan *malicious* dengan 1.0.

Tabel 3. 6 Data Sebelum *Encoding*

	ts	*proto*	conn_ state	missed_ bytes	orig_ pkts	orig_ip_ bytes	resp_ pkts	resp_ip_ bytes	label
46	1,55E+15	tcp	SF	2896.0	93.0	5571.0	92.0	131755.0	*Benign*
47	1,55E+15	tcp	S0	0.0	3.0	180.0	0.0	0.0	*Benign*
48	1,55E+15	tcp	S0	0.0	3.0	180.0	0.0	0.0	*Malicious*
49	1,55E+15	tcp	S0	0.0	1.0	60.0	0.0	0.0	*Benign*
50	1,55E+15	tcp	S0	0.0	1.0	60.0	0.0	0.0	*Malicious*
51	1,55E+15	tcp	S0	0.0	1.0	60.0	0.0	0.0	*Benign*
52	1,55E+15	tcp	S0	0.0	3.0	180.0	0.0	0.0	*Malicious*
53	1,55E+15	udp	SF	0.0	1.0	76.0	1.0	76.0	*Benign*
54	1,55E+15	tcp	S0	0.0	1.0	60.0	0.0	0.0	*Malicious*
55	1,55E+15	tcp	S0	0.0	3.0	180.0	0.0	0.0	*Malicious*
56	1,55E+15	tcp	S0	0.0	1.0	60.0	0.0	0.0	*Benign*

Tabel 3. 7 Data Setelah *Encoding*

	ts	*pro to*	con n_ stat e	…	label	*prot o_* inde xed	*conn_st ate_* indexed	*proto_* oh_ encod ed	*conn_state*_oh_ encoded	encod ed_ label
4 6	1,55E +15	tcp	SF	…	Benig n	0.0	3.0	(1.0, 0.0)	(0.0, 0.0, 0.0, 1.0, 0.0, 0.0)	2.0
4 7	1,55E +15	tcp	S0	…	Benig n	0.0	1.0	(1.0, 0.0)	(0.0, 1.0, 0.0, 0.0, 0.0, 0.0)	2.0

	ts	*proto*	conn_state	…	label	*proto_*indexed	*conn_state_*indexed	*proto_*oh_encoded	*conn_state_*oh_encoded	encoded_label
48	1,55E+15	tcp	S0	…	Malicious	0.0	1.0	(1.0, 0.0)	(0.0, 1.0, 0.0, 0.0, 0.0, 0.0)	1.0
49	1,55E+15	tcp	S0	…	Benign	0.0	1.0	(1.0, 0.0)	(0.0, 1.0, 0.0, 0.0, 0.0, 0.0)	2.0
50	1,55E+15	tcp	S0	…	Malicious	0.0	1.0	(1.0, 0.0)	(0.0, 1.0, 0.0, 0.0, 0.0, 0.0)	1.0
51	1,55E+15	tcp	S0	…	Benign	0.0	1.0	(1.0, 0.0)	(0.0, 1.0, 0.0, 0.0, 0.0, 0.0)	2.0
52	1,55E+15	tcp	S0	…	Malicious	0.0	1.0	(1.0, 0.0)	(0.0, 1.0, 0.0, 0.0, 0.0, 0.0)	1.0
53	1,55E+15	udp	SF	…	Benign	1.0	3.0	(0.0, 1.0)	(0.0, 0.0, 0.0, 1.0, 0.0, 0.0)	2.0
54	1,55E+15	tcp	S0	…	Malicious	0.0	1.0	(1.0, 0.0)	(0.0, 1.0, 0.0, 0.0, 0.0, 0.0)	1.0
55	1,55E+15	tcp	S0	…	Malicious	0.0	1.0	(1.0, 0.0)	(0.0, 1.0, 0.0, 0.0, 0.0, 0.0)	1.0
56	1,55E+15	tcp	S0	…	Benign	0.0	1.0	(1.0, 0.0)	(0.0, 1.0, 0.0, 0.0, 0.0, 0.0)	2.0

Dengan penerapan metode *One-Hot Encoding*, *Fitur* telah berhasil diubah menjadi format *numerik* yang siap diproses oleh model *Logistic Regression*. Tahapan selanjutnya dalam *Preprocessing* data adalah *Vektorisasi* dan *Normalisasi*, yang bertujuan untuk menyatukan semua *Fitur numerik* ke dalam satu bentuk *vektor* berdimensi tetap serta menyamakan skala nilainya. Proses ini sangat penting agar algoritma *Logistic Regression* dapat melakukan pembelajaran dengan lebih efisien, stabil, dan tidak *bias* terhadap *Fitur* tertentu.

3.1.3.3 *Vektorisasi & Normalisasi*

Tahap *Vektorisasi* dan *Normalisasi*merupakan bagian penting dalam proses *Preprocessing* data sebelum dilakukan pemodelan menggunakan *Logistic Regression*. Proses ini bertujuan untuk menyusun data dalam format yang sesuai

dengan kebutuhan algoritma pembelajaran mesin serta memastikan skala antar *Fitur* bersifat seragam agar tidak terjadi *bias* dalam proses pemodelan model.

Vektorisasi dilakukan menggunakan teknik *Vector Assembler*, yang berfungsi untuk menggabungkan seluruh *Fitur numerik* hasil proses *Encoding* ke dalam satu representasi *vektor* berdimensi tetap yang disebut *feature_vector*. *Fitur-Fitur* yang telah melalui tahapan *One-Hot Encoding*, seperti *proto_oh_encoded* dan *conn_state_oh_encoded*, serta *Fitur numerik* asli lainnya seperti *duration, orig_bytes, resp_bytes, orig_pkts,* dan *resp_pkts,* dikombinasikan ke dalam sebuah *vektor* tunggal. Proses ini menghasilkan bentuk data yang siap diproses dalam format matriks, yang sesuai dengan kebutuhan perhitungan matematika pada *Logistic Regression*.

Sebagai hasil dari proses *Vektorisasi*, setiap baris data dalam dataset kini direpresentasikan sebagai satu *vektor* berdimensi tetap, di mana setiap elemen mewakili satu *Fitur* numerik atau hasil *Encoding* dari *Fitur* kategorikal. Representasi ini sangat penting karena algoritma *Logistic Regression* mengandalkan struktur data berbentuk matriks untuk melakukan optimasi parameter model secara efisien. Hasil akhir *Vektorisasi* disimpan dalam kolom baru bernama *feature_vector*, yang akan digunakan sebagai input langsung dalam proses pemodelan model.

Setelah proses Vektorisasi selesai, dilakukan *Normalisasi*data menggunakan teknik *Min-Max Scaling*. *Normalisasi*ini penting karena dalam dataset trafik jaringan, nilai antar *Fitur* memiliki rentang yang sangat bervariasi, seperti pada *Fitur orig_pkts, missed_bytes, hingga resp_ip_bytes*. Tanpa

Normalisasi, *Fitur* yang memiliki rentang nilai besar dapat mendominasi proses pembelajaran dan mengganggu konvergensi dari model. Dalam penelitian ini, *Min-Max Scaling* digunakan untuk melakukan *Normalisasi*, yang mengubah nilai *Fitur* ke dalam rentang [0,1]. Proses ini bertujuan untuk menyamakan skala antar *Fitur* sehingga setiap *Fitur* dapat memiliki pengaruh yang seimbang dalam proses pembelajaran. Teknik ini dipilih karena mampu mempertahankan distribusi data asli tanpa mengubah hubungan antara *Fitur*. Secara matematis, *Min-Max Scaling* dapat dirumuskan pada rumus 3.2 (Allorerung et al., 2024) :

$$X' = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \tag{3.2}$$

Keterangan :

a) X' : nilai hasil *Normalisasi*,
b) X : nilai asli dari Fitur,
c) $X_{\min}$: nilai minimum dari Fitur tersebut dalam dataset,
d) $X_{\max}$: nilai maksimum dari Fitur tersebut dalam dataset.

Penerapan rumus 3.2 dilakukan secara otomatis pada seluruh *Fitur numerik* yang terdapat dalam *feature_vector*. Misalnya, pada *Fitur orig_bytes* yang memiliki nilai asli sebesar 2896.0, dengan asumsi nilai minimum pada *Fitur* ini adalah 0.0 dan nilai maksimum adalah 5792.0, maka nilai hasil *Normalisasi* dihitung sebagai berikut:

$$X' = \frac{2896.0 - 0.0}{5792.0 - 0.0} = \frac{2896.0}{5792.0} = 0.5 \tag{3.3}$$

Hasil ini sesuai dengan nilai *0.5* yang terlihat pada kolom *normalized_feature_vector* baris ke-46 di Tabel 3.9. Contoh lain terdapat pada *Fitur orig_pkts* dengan nilai asli sebesar 93.0. Jika diketahui rentang nilai pada

Fitur tersebut adalah dari 1.0 hingga 101.0, maka *Normalisasi* dihitung menggunakan rumus:

$$X' = \frac{93.0-1.0}{101.0-1.0} = \frac{92.0}{100.0} = 0.92 \quad (3.4)$$

Dengan demikian, semua nilai *Fitur* yang terdapat pada *feature_vector* telah dikonversi ke dalam skala [0, 1] melalui proses ini. Setiap elemen *vektor* diubah dengan rumus yang sama, menggunakan nilai minimum dan maksimum dari masing-masing *Fitur* berdasarkan keseluruhan dataset. Proses ini dilakukan agar model *Logistic Regression* tidak *bias* terhadap *Fitur-Fitur* yang memiliki skala lebih besar, seperti *ts* (*timestamp*) atau *resp_ip_bytes*, yang rentang nilainya bisa sangat jauh dibandingkan *Fitur* lain seperti *missed_bytes*.

Aplikasi dari rumus ini dapat dilihat pada perbandingan antara Tabel 3.8 (sebelum *Vektorisasi* & *Normalisasi*) dan Tabel 3.9 (setelah *Vektorisasi* & *Normalisasi*), di mana seluruh nilai telah berada dalam rentang kontinu antara 0 hingga 1, menjamin kestabilan dan efisiensi proses pemodelan model.

Tabel 3. 8 Data Sebelum *Vektorisasi & Normalisasi*

	ts	*proto*	conn_state	…	label	*proto_* indexed	*conn_state_* indexed	*proto_oh_* encoded	*conn_state*_oh_ encoded	encoded_ label
46	1,55E+15	tcp	SF	…	*Benign*	0.0	3.0	(1.0, 0.0)	(0.0, 0.0, 0.0, 1.0, 0.0, 0.0)	2.0
47	1,55E+15	tcp	S0	…	*Benign*	0.0	1.0	(1.0, 0.0)	(0.0, 1.0, 0.0, 0.0, 0.0, 0.0)	2.0
48	1,55E+15	tcp	S0	…	*Malicious*	0.0	1.0	(1.0, 0.0)	(0.0, 1.0, 0.0, 0.0, 0.0, 0.0)	1.0
49	1,55E+15	tcp	S0	…	*Benign*	0.0	1.0	(1.0, 0.0)	(0.0, 1.0, 0.0, 0.0, 0.0, 0.0)	2.0
50	1,55E+15	tcp	S0	…	*Malicious*	0.0	1.0	(1.0, 0.0)	(0.0, 1.0, 0.0, 0.0, 0.0, 0.0)	1.0
51	1,55E+15	tcp	S0	…	*Benign*	0.0	1.0	(1.0, 0.0)	(0.0, 1.0, 0.0, 0.0, 0.0, 0.0)	2.0
52	1,55E+15	tcp	S0	…	*Malicious*	0.0	1.0	(1.0, 0.0)	(0.0, 1.0, 0.0, 0.0, 0.0, 0.0)	1.0

	ts	*proto*	conn_state	…	label	*proto_* indexed	*conn_state_* indexed	*proto_oh_* encoded	*conn_state*_oh_ encoded	encoded_ label
53	1,55E+15	udp	SF	…	*Benign*	1.0	3.0	(0.0, 1.0)	(0.0, 0.0, 0.0, 1.0, 0.0, 0.0)	2.0
54	1,55E+15	tcp	S0	…	*Malicious*	0.0	1.0	(1.0, 0.0)	(0.0, 1.0, 0.0, 0.0, 0.0, 0.0)	1.0
55	1,55E+15	tcp	S0	…	*Malicious*	0.0	1.0	(1.0, 0.0)	(0.0, 1.0, 0.0, 0.0, 0.0, 0.0)	1.0
56	1,55E+15	tcp	S0	…	*Benign*	0.0	1.0	(1.0, 0.0)	(0.0, 1.0, 0.0, 0.0, 0.0, 0.0)	2.0

Tabel 3. 9 Data Setelah *Vektorisasi & Normalisasi*

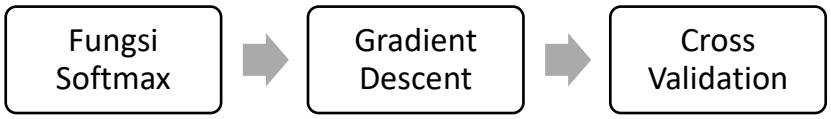
	feature_vector	*normalized_feature_vector*
46	(1545404408.582022, 2896.0, 93.0, 5571.0, 92.0…	(0.006848740506453515, 0.5, 0.0050422901756668…
47	(1545404409.888278, 0.0, 3.0, 180.0, 0.0, 0.0,…	(0.006863859723294655, 0.0, 0.0001626545217957…
48	(1545404410.924437, 0.0, 3.0, 180.0, 0.0, 0.0,…	(0.00687585271128174, 0.0, 0.00016265452179570…
49	(1545404417.061944, 0.0, 1.0, 60.0, 0.0, 0.0, …	(0.006946891079452752, 0.0, 5.421817393190197e…
50	(1545404418.101866, 0.0, 1.0, 60.0, 0.0, 0.0, …	(0.006958927621698982, 0.0, 5.421817393190197e…
51	(1545404425.622069, 0.0, 1.0, 60.0, 0.0, 0.0, …	(0.007045969957273958, 0.0, 5.421817393190197e…
52	(1545404426.417366, 0.0, 3.0, 180.0, 0.0, 0.0,…	(0.0070551750973745254, 0.0, 0.000162654521795…
53	(1545404378.459174, 0.0, 1.0, 76.0, 1.0, 76.0,…	(0.006500084613710292, 0.0, 5.421817393190197e…
54	(1545404433.622016, 0.0, 1.0, 60.0, 0.0, 0.0, …	(0.007138565075930232, 0.0, 5.421817393190197e…
55	(1545404436.419172, 0.0, 3.0, 180.0, 0.0, 0.0,…	(0.007170940665201295, 0.0, 0.0001626545217957…
56	(1545404442.262096, 0.0, 1.0, 60.0, 0.0, 0.0, …	(0.007238569391382427, 0.0, 5.421817393190197e…

Setelah melalui proses *Vektorisasi* dan *Normalisasi*, seluruh *Fitur numerik* dalam dataset telah disusun ke dalam bentuk *vektor* berdimensi tetap dan disesuaikan skalanya ke dalam rentang [0,1]. Dengan kondisi ini, model dapat melakukan proses pembelajaran secara lebih stabil dan efisien, tanpa

kecenderungan untuk memprioritaskan *Fitur* dengan nilai *numerik* yang lebih besar. Kombinasi kedua tahapan ini membantu meningkatkan kinerja model dalam mengklasifikasikan trafik jaringan secara objektif dan seimbang.

3.1.4 Pemodelan *Logistic Regression*

Proses pemodelan metode *Logistic Regression* ini bertujuan untuk membangun sistem klasifikasi yang mampu membedakan antara trafik jaringan yang bersifat normal dan trafik yang terindikasi sebagai *Malware*. Pemodelan ini dilakukan dengan beberapa tahapan sesuai dengan gambar 3.4.



Gambar 3. 4 Proses Pemodelan *Logistic Regression*

Model *Logistic Regression* dalam penelitian ini dirancang dengan menerapkan fungsi *aktivasi Softmax*, yang digunakan untuk menangani klasifikasi multikelas dengan cara mengubah hasil keluaran model menjadi nilai probabilitas untuk setiap kelas. Untuk mengoptimalkan parameter model, digunakan algoritma *Gradient Descent* yang bekerja dengan meminimalkan nilai kesalahan (*Loss function*) secara iteratif.

Selain itu, untuk meningkatkan keandalan model dan menghindari *overfitting*, dilakukan proses K-*Fold Cross Validation* sebanyak 5-*Fold*, di mana data latih dibagi ke dalam lima subset untuk memastikan bahwa setiap bagian data diuji secara adil. *Akurasi* dari masing-masing *Fold* dihitung dan dirata-ratakan untuk memperoleh gambaran performa model yang lebih objektif dan stabil.

Konfigurasi *K-Fold Cross Validation* dalam penelitian ini diimplementasikan dengan menggunakan library scikit-learn melalui objek *KFold*, dengan parameter *n_Splits=5*, *shuffle=True*, dan *random_state=42*. Parameter *n_Splits=5* menunjukkan bahwa dataset dibagi menjadi lima bagian yang sama besar, sehingga akan dilakukan lima kali iterasi pemodelan dan *validasi*. Parameter *shuffle=True* digunakan untuk mengacak urutan data sebelum dilakukan pembagian *Fold*, yang bertujuan agar distribusi data pada masing-masing *Fold* tidak *bias* atau bergantung pada urutan data. Sedangkan *random_state=42* digunakan untuk menjamin reproducibility, yaitu agar hasil pembagian *Fold* tetap konsisten setiap kali program dijalankan.

Setelah proses pemodelan selesai, model dievaluasi menggunakan berbagai metrik kinerja, di antaranya *Confusion matrix* dan *Classification report*. *Classification report* digunakan untuk menunjukkan metrik evaluasi seperti *akurasi, precision, recall,* dan *F1-score* untuk masing-masing kelas, yang memberikan informasi detail tentang kemampuan model dalam mendeteksi trafik yang *benign* maupun *malicious*. Evaluasi ini menjadi langkah penting untuk menilai sejauh mana model mampu meng*generalisasi* pola data serta untuk memastikan kesiapan model sebelum diterapkan dalam lingkungan nyata.

3.1.4.1 Fungsi *Softmax*

Dalam pemodelan *Logistic Regression*, fungsi *aktivasi* yang digunakan adalah *Softmax*, yang berperan dalam mengubah hasil *output* model menjadi probabilitas dalam rentang antara 0 hingga 1, memungkinkan penggunaan model

baik untuk klasifikasi biner maupun multikelas (Zhang, 2024). Persamaan matematis fungsi *Softmax* bisa dilihat dalam rumus 2.2.

Fungsi *Softmax* menjamin bahwa total probabilitas semua kelas dalam suatu sampel berjumlah 1, sehingga model dapat memilih kelas dengan probabilitas tertinggi sebagai prediksi akhir (Zhang, 2024).

Sebelum diterapkan fungsi *Softmax*, dataset telah melalui proses *Normalisasi* agar nilai-nilai *Fitur* berada dalam rentang yang sesuai. Misalnya, hasil *transformasi* dari *Fitur* yang telah *diNormalisasi* akan dimasukkan ke dalam persamaan model sebagai berikut:

Misalkan kita memiliki tiga kelas dalam dataset dengan skor *output* model sebagai rumus 3.5:

$$z_1 = 2.0, \quad z_2 = 1.0 \tag{3.5}$$

Untuk menghitung probabilitas menggunakan fungsi *Softmax*, kita gunakan rumus 3.6:

$$S(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}} \tag{3.6}$$

1. Hitung eksponensial dari setiap skor, dengan rumus 3.7

$$e^{z_1} = e^{2.0} \approx 7.389 \qquad e^{z_2} = e^{1.0} \approx 2.718 \tag{3.7}$$

2. Hitung jumlah total eksponensial pada rumus 3.8

$$\sum e^{z_j} = 7.389 + 2.718 = 10.107 \tag{3.8}$$

3. Hitung probabilitas masing-masing kelas seperti pada rumus 3.9

$$S(z_1) = \frac{7.389}{10.107} \approx 0.731 \quad S(z_2) = \frac{2.718}{10.107} \approx 0.269 \tag{3.9}$$

Dengan demikian, model memprediksi bahwa sampel kemungkinan besar termasuk ke dalam kelas pertama 73.1% dan kelas kedua 26.9%.

Fungsi *Softmax* memastikan bahwa total probabilitas semua kelas berjumlah 1, sehingga dapat digunakan untuk klasifikasi multikelas dengan memilih kelas yang memiliki probabilitas tertinggi sebagai prediksi akhir.

3.1.4.2 *Gradient Descent*

Gradient Descent adalah metode optimasi yang digunakan untuk meminimalkan nilai cost *function* dalam proses pemodelan model *Logistic Regression*. Konsep utamanya adalah memperbarui parameter model dalam hal ini bobot (W) dan *bias* (b) secara iteratif ke arah yang menurunkan nilai *Loss function*. Hal ini dilakukan dengan menghitung turunan dari fungsi *Loss* terhadap setiap parameter, lalu memperbaruinya ke arah negatif dari *gradien* (turunan) tersebut. Dengan pendekatan ini, model dapat belajar secara bertahap dari kesalahan prediksi dan menghasilkan *akurasi* yang lebih baik.

Pada implementasi program, fungsi cost yang digunakan adalah *Cross Entropy Loss*, karena klasifikasi yang dilakukan bersifat *biner* atau multikelas. Rumus dasar fungsi *Loss* tersebut (untuk multikelas) seperti yang ada di bab 2, Namun karena sudah menggunakan bentuk *multiclass* dan output model berupa *Softmax*, maka bentuk *Loss* -nya ada pada rumus 3.10.

$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k} y_{ij}\log\left(\widehat{y_{ij}}\right) \tag{3.10}$$

Keterangan :

- n = jumlah sampel
- k = jumlah kelas
- y_{ij} = label asli (one-hot)

- $\widehat{y_{ij}}$ = prediksi model (*Softmax* output)

Misalkan untuk satu sampel data yang memiliki label asli $y = [0,1,0]$ yang menunjukkan kelas ke-2, dan model memberikan prediksi model hasil dari *Softmax* $\hat{y} = [0.2,0.7,0.1]$. Maka fungsi *Loss* untuk sampel ini dihitung sebagai berikut :

$$\mathcal{L} = -(0 \cdot \log(0.2) + 1 \cdot \log(0.7) + 0 \cdot \log(0.1)) = -\log(0.7) \approx 0.3567$$

Perhitungan tersebut menunjukkan bahwa karena label sebenarnya berada pada indeks ke-2 (yaitu nilai 1 pada elemen kedua), maka hanya log dari probabilitas pada indeks tersebut yang dihitung, sementara elemen lainnya diabaikan (dikalikan dengan 0). Semakin kecil nilai *Loss* -nya, semakin baik prediksi model terhadap label sebenarnya. Jika proses ini diulang untuk seluruh data pemodelan, maka nilai-nilai *Loss* tersebut akan dijumlahkan dan dirata-ratakan untuk memberikan estimasi rata-rata kesalahan model pada seluruh data.

Setelah nilai *Loss* dihitung, langkah selanjutnya adalah melakukan pembaruan parameter model, yaitu bobot (W) dan *bias* (b), menggunakan algoritma *Gradient Descent*. Proses pembaruan dilakukan dengan memanfaatkan turunan dari fungsi *Loss* terhadap bobot dan *bias*. Rumus pembaruan bobot dan *bias* dapat dituliskan pada rumus 2.4 dan 2.5 diatas pada bab 2.

Sebagai ilustrasi proses pembaruan bobot, misalkan:

- α = 0.1
- Jumlah data (m) = 2
- X = $[[1,2],[3,4]]$
- Y = $[[1,0],[0,1]]$
- $\hat{y}$ = $[[0.7,0.3],[0.4,0.6]]$

Maka, selisih antara prediksi dan label aktual error $\hat{y}$ - y = $[[-0.3,0.3],[0.4,-0.4]]$. Selisih inilah yang kemudian digunakan dalam

perhitungan turunan fungsi *Loss* (3.10) dan (3.11) untuk menghitung pembaruan bobot dan *bias*, sehingga model dapat memperbaiki kesalahan prediksinya pada iterasi berikutnya. Proses ini akan terus berlangsung hingga nilai *Loss* mencapai titik minimum yang stabil, atau hingga mencapai jumlah *epoch* yang telah ditentukan.

Dengan cara ini, *Gradient Descent* memungkinkan model *Logistic Regression* mengoptimalkan prediksi klasifikasinya, hingga model mencapai konvergensi atau *Loss* yang stabil.

3.1.4.3 *Cross Validation*

Cross Validation merupakan salah satu teknik *validasi* internal yang bertujuan untuk mengevaluasi kemampuan model dalam menghasilkan prediksi yang konsisten dan tidak hanya bergantung pada subset data tertentu. Dalam penelitian ini, digunakan metode *K-Fold Cross Validation* sebanyak 5 *Fold*, di mana data latih dibagi menjadi lima bagian berukuran sama. Pada setiap iterasi, satu bagian digunakan sebagai data *validasi*, sedangkan empat bagian lainnya digunakan untuk pemodelan model. Proses ini memastikan bahwa seluruh data latih digunakan baik sebagai pemodelan maupun *validasi* secara bergantian.

Proses *Cross Validation* dimulai dengan *Encoding* label menjadi one-hot *Encoding*, kemudian model *Logistic Regression* dibangun ulang dari awal pada setiap *Fold*. Untuk menghitung keluaran prediksi, digunakan fungsi *aktivasi Softmax*, yang menghasilkan distribusi probabilitas antar kelas. Proses pemodelan memanfaatkan algoritma *Gradient Descent* dengan fungsi biaya *Cross-Entropy Loss* , dan parameter model diperbarui selama 14 *epoch*.

Setelah model selesai dilatih pada setiap *Fold*, dilakukan evaluasi terhadap performa model menggunakan empat metrik utama, yaitu *akurasi*, *Precision*, *recall*, dan *F1-score*. Evaluasi dilakukan menggunakan fungsi *accuracy*_score, *Precision*_score, *recall*_score, dan *f1_score* dari pustaka sklearn.metrics. Nilai metrik dari masing-masing *Fold* disimpan ke dalam sebuah Tabel hasil, dan pada akhir iterasi ditambahkan satu baris rata-rata dari seluruh metrik yang telah dihitung. Rumus *akurasi* dalam setiap *Fold* menggunakan rumus 3.11:

$$Akurasi = \frac{\text{Jumlah prediksi benar}}{\text{Total data } validasi} \tag{3.11}$$

Sebagai contoh hasil eksperimen, pada program yang dijalankan diperoleh hasil *akurasi*, *presisi*, *recall*, dan *F1-score* dari masing-masing *Fold*, seperti pada Tabel 3.10.

Tabel 3. 10 Hasil *Cross Validation*

Fold ke-	*Akurasi*	*Presisi*	*Recall*	*F1-score*
1	98.86	98.90	98.85	98.85
2	98.86	98.88	98.86	98.86
3	99.00	99.01	99.00	99.00
4	99.16	99.18	99.15	99.15
5	98.81	98.83	98.80	98.80
Rata-rata	98.94	98.96	98.93	98.93

Perhitungan rata-rata *akurasi* dilakukan dengan menjumlahkan hasil *akurasi* dari seluruh *Fold* dan membaginya dengan jumlah *Fold*, sebagaimana dituliskan dalam rumus 3.13.

$$\text{Rata-rata } Akurasi = \frac{A_1+A_2+A_3+A_4+A_5}{5}$$

$$= \frac{0.9886+0.9886+0.9900+0.9916+0.9881}{5} = 0.98938 \tag{3.13}$$

Dengan rata-rata *akurasi* sebesar 0.9894 atau 98.94%, serta nilai *presisi*, *recall*, dan *F1-score* yang tinggi dan stabil, model *Logistic Regression* dalam

penelitian ini terbukti memiliki kemampuan klasifikasi yang konsisten terhadap trafik jaringan. Hal ini menunjukkan bahwa model mampu mengenali pola dari data *benign* dan *malicious* dengan *akurasi* yang tinggi dan dapat diandalkan.

3.2 Skenario Pengujian

Skenario pengujian pada penelitian ini disusun untuk mengukur kinerja model *Logistic Regression* dalam mengklasifikasikan trafik jaringan ke dalam dua kategori, yaitu *malicious*(berbahaya) dan *benign* (tidak berbahaya). Untuk memastikan proses evaluasi berjalan secara objektif dan menghindari *overfitting*, dilakukan pemisahan data menjadi data latih dan data uji. Model akan dilatih menggunakan data latih, dan selanjutnya diuji menggunakan data uji untuk menilai kemampuannya dalam mengenali pola baru yang belum pernah dilihat sebelumnya. Rincian skenario pengujian dapat dilihat pada tabel 3.11.

Tabel 3. 11 Skenario Pengujian

Skenario	**Metode *Split* Data**	***Rasio Split***
1	Manual *Split*	80:20
2	*Random Split*	80:20
3	*Random Split*	70:30

BAB IV

HASIL DAN PEMBAHASAN

4.1 Hasil *Preprocessing* Data

Penelitian ini menggunakan data sekunder yang berasal dari stratosphereips.org/datasets-*IoT*23. Dataset tersebut berisikan data mentah traffik jaringan yang belum layak untuk model *Logistic Regression*. Tahapan *Preprocessing* data meliputi seleksi *Fitur* untuk menyeleksi *Fitur* apa yang dipakai dan tidak, jika tidak terpakai maka dihapus. Selanjutnya *Encoding Fitur* dimana *Fitur* kategorikal diubah menjadi format *numerik*, lalu tahapan akhir melakukan *Vektorisasi* dan *Normalisasi* pada data, yaitu penggabungan seluruh *Fitur numerik* menjadi satu *vektor* berdimensi tetap dan memastikan bahwa semua *Fitur* berada dalam skala yang sama, sehingga tidak ada *Fitur* yang mendominasi proses pemodelan *Logistic Regression*.

4.1.1 Hasil Seleksi *Fitur*

Seleksi *Fitur* merupakan salah satu tahapan krusial dalam proses pra-pemrosesan data. Tujuan utamanya adalah untuk menyaring *Fitur-Fitur* yang relevan, serta mengeliminasi *Fitur-Fitur* yang bersifat redundan atau mengandung noise, sehingga dapat meningkatkan performa dan efisiensi model klasifikasi yang digunakan. Sebelum dilakukan analisis awal terhadap dataset untuk menentukan *Fitur-Fitur* yang perlu dihapus, Tabel 4.1 menyajikan data mentah dari dataset sebelum melalui proses *Preprocessing*.

Tabel 4. 1 Data Mentah Sebelum Seleksi *Fitur*

ts	Uid	id_orig_h	id_resp_h	*proto*	service	. . .	local_resp	history	orig_pkts	orig_ip_bytes	resp_pkts	resp_ip_bytes	tunnel_parents	label	detailed_label
1,55E+15	CvOvGB2WOKVl6Lo3Wf	192.168.1.195	147.231.100.5	udp	-	. . .	-	Dd	1.0	76.0	1.0	76.0	-	*Benign*	-
1,54E+15	C44hHm17HZ3lMqs8Tb	1,92E+11	81.2.254.224	udp	-	. . .	-	D	1.0	76.0	0.0	0.0	-	*Benign*	-
1,54E+15	CFDwDE15VacFXqJUZ4	1,92E+11	144.48.166.166	udp	-	. . .	-	D	1.0	76.0	0.0	0.0	-	*Benign*	-
1,55E+15	CmYYN04wo2sCcLG5Wk	192.168.1.195	123.59.209.185	tcp	-	. . .	-	C	0.0	0.0	0.0	0.0	-	*Malicious*	None
1,55E+15	CxNRRwNJhn2pAKXP1	192.168.1.195	123.59.209.185	tcp	-	. . .	-	C	0.0	0.0	0.0	0.0	-	*Malicious*	None
1,54E+15	CcnxnK3X0qaNCFIFfj	1,92E+11	5.1.56.123	udp	-	. . .	-	D	1.0	76.0	0.0	0.0	-	*Benign*	-
1,54E+15	CHi0073vh0QyWm6v11	1,92E+11	192.168.100.1	udp	dns	. . .	-	Dd	1.0	73.0	1.0	73.0	-	*Benign*	-
1,54E+15	CrBy6k3Q58tC2vCBw5	1,92E+11	147.231.100.5	udp	-	. . .	-	D	1.0	76.0	0.0	0.0	-	*Benign*	-
1,55E+15	CoJGGr4w7i74b72Mp1	192.168.1.195	123.59.209.185	tcp	-	. . .	-	C	0.0	0.0	0.0	0.0	-	*Malicious*	None

Berdasarkan hasil analisis awal terhadap dataset, ditemukan bahwa beberapa *Fitur* mengandung banyak nilai kosong (*null*) serta simbol "-" yang menunjukkan ketidaktersediaan data. Temuan ini dapat dilihat pada Gambar 4.1, yang memperlihatkan distribusi data yang tidak lengkap pada sejumlah *Fitur*.

Column Name	Number of Null Values	Number of '-' Values
ts	0	NaN
uid	0	0.0
id_orig_h	0	0.0
id_orig_p	0	NaN
id_resp_h	0	0.0
id_resp_p	0	NaN
proto	0	0.0
service	0	7871.0
duration	6210	NaN
orig_bytes	6210	NaN
resp_bytes	6210	NaN
conn_state	0	0.0
local_orig	0	10390.0
local_resp	0	10390.0
missed_bytes	0	NaN
history	0	1.0
orig_pkts	0	NaN
orig_ip_bytes	0	NaN
resp_pkts	0	NaN
resp_ip_bytes	0	NaN
tunnel_parents	0	10390.0
label	0	0.0
detailed_label	5192	5195.0

Gambar 4. 1 Hasil Analisis Data Yang Kosong

Berdasarkan hasil, dilakukan penghapusan terhadap *Fitur-Fitur* tersebut karena dinilai tidak memberikan kontribusi signifikan terhadap model. Penghapusan ini bertujuan untuk meminimalkan potensi *bias* serta mempercepat proses pemodelan. Setelah proses seleksi *Fitur* dilakukan, diperoleh hasil akhir sebagaimana ditampilkan pada Gambar 4.2. Dataset tersisa 8 *Fitur* utama yang dinilai paling relevan dan siap digunakan dalam tahapan analisis selanjutnya.

Column Name	Number of Null Values	Number of '-' Values
ts	0	NaN
proto	0	0.0
conn_state	0	0.0
missed_bytes	0	NaN
orig_pkts	0	NaN
orig_ip_bytes	0	NaN
resp_pkts	0	NaN
resp_ip_bytes	0	NaN
label	0	0.0

Gambar 4. 2 Hasil Analisis Setelah Seleksi *Fitur*

Tabel 4.1 menampilkan dataset lengkap yang terdiri dari data pada *Fitur-Fitur* terseleksi. Dataset inilah yang digunakan sebagai dasar dalam proses klasifikasi menggunakan metode *Logistic Regression.*

Tabel 4. 2 Data Hasil Seleksi *Fitur*

ts	*proto*	*conn_state*	missed_bytes	orig_pkts	orig_ip_bytes	resp_pkts	resp_ip_bytes	label
1,55E+15	udp	SF	0.0	1.0	76.0	1.0	76.0	*Benign*
1,54E+15	udp	S0	0.0	1.0	76.0	0.0	0.0	*Benign*
1,54E+15	udp	S0	0.0	1.0	76.0	0.0	0.0	*Benign*
1,55E+15	tcp	OTH	0.0	0.0	0.0	0.0	0.0	*Malicious*
1,55E+15	tcp	OTH	0.0	0.0	0.0	0.0	0.0	*Malicious*
1,54E+15	udp	S0	0.0	1.0	76.0	0.0	0.0	*Benign*
1,54E+15	udp	SF	0.0	1.0	73.0	1.0	73.0	*Benign*
1,54E+15	udp	S0	0.0	1.0	76.0	0.0	0.0	*Benign*
1,55E+15	tcp	OTH	0.0	0.0	0.0	0.0	0.0	*Malicious*

4.1.2 Hasil *Encoding Fitur*

Setelah proses seleksi *Fitur* dilakukan, tahap selanjutnya adalah melakukan *Encoding* terhadap *Fitur-Fitur* kategorikal. *Encoding* diperlukan untuk mengubah nilai-nilai non-*numerik* menjadi bentuk *numerik* agar dapat diproses oleh algoritma *Logistic Regression* yang hanya menerima input *numerik.*

Dalam penelitian ini, proses *Encoding* dilakukan menggunakan metode Label *Encoding*, yang menggantikan setiap nilai kategorikal dengan angka indeks tertentu berdasarkan urutan kemunculannya. *Fitur-Fitur* yang dikonversi melalui proses ini antara lain *proto, conn_state, dan label*. Pada Tabel 4.3 dan Tabel 4.4 merupakan pemetaan kategori ke indeks untuk masing-masing *Fitur*.

Tabel 4. 3 Hasil Konversi *Fitur proto*

proto	TCP	UDP	ICMP
TCP	1	0	0
UDP	0	1	0
ICMP	0	0	1

Tabel 4. 4 Hasil Konversi *Fitur conn_state*

conn_state	S0	OTH	SF	S3	RSTR	S2
S0	1	0	0	0	0	0
OTH	0	1	0	0	0	0
SF	0	0	1	0	0	0
S3	0	0	0	1	0	0
RSTR	0	0	0	0	1	0
S2	0	0	0	0	0	1

Sebelum masuk ke hasil proses *Encoding*, data hasil seleksi *Fitur* yang belum melalui proses *Encoding* disajikan terlebih dahulu pada Tabel 4.5. Tabel ini menampilkan nilai asli dari *Fitur* kategorikal seperti *proto* dan *conn_state*, yang masih dalam format string. Nilai-nilai tersebut belum dapat langsung digunakan dalam pemodelan karena belum berbentuk *numerik*.

Tabel 4. 5 Data Sebelum di *Encoding*

ts	*proto*	*conn_state*	missed_bytes	orig_pkts	orig_ip_bytes	resp_pkts	resp_ip_bytes	label
1,55E+15	udp	SF	0.0	1.0	76.0	1.0	76.0	*Benign*
1,54E+15	udp	S0	0.0	1.0	76.0	0.0	0.0	*Benign*
1,54E+15	udp	S0	0.0	1.0	76.0	0.0	0.0	*Benign*
1,55E+15	tcp	OTH	0.0	0.0	0.0	0.0	0.0	*Malicious*
1,55E+15	tcp	OTH	0.0	0.0	0.0	0.0	0.0	*Malicious*
1,54E+15	udp	S0	0.0	1.0	76.0	0.0	0.0	*Benign*
1,54E+15	udp	SF	0.0	1.0	73.0	1.0	73.0	*Benign*
1,54E+15	udp	S0	0.0	1.0	76.0	0.0	0.0	*Benign*
1,55E+15	tcp	OTH	0.0	0.0	0.0	0.0	0.0	*Malicious*

Hasil aktual dari proses *Encoding* dapat dilihat pada Tabel 4.6, di mana *Fitur proto* telah diubah menjadi dua bentuk representasi *numerik*, yaitu menjadi *Fitur proto_indexed,* dan *proto_oh_encoded,* dan untuk *Fitur conn_state* diubah menjadi *Fitur conn_state_indexed, dan conn_state_oh_encoded*. Kolom *_indexed merupakan hasil dari proses *Label Encoding*, yaitu pemberian label *numerik* sederhana untuk setiap kategori. Sementara itu, kolom *_oh_encoded menunjukkan hasil dari *One-Hot Encoding*, yang menghasilkan representasi *vektor* biner dan digunakan sebagai input dalam pemodelan model klasifikasi.

Tabel 4. 6 Data Setelah *Encoding*

ts	*proto*	*conn_state*	missed_bytes	orig_pkts	orig_ip_bytes	resp_pkts	resp_ip_bytes	label	*proto*_indexed	*conn_state*_indexed	*proto*_oh_encoded	*conn_state*_oh_encoded	encoded_label
1,55E+15	udp	SF	0.0	1.0	76.0	1.0	76.0	*Benign*	1.0	2.0	(0.0, 1.0, 0.0)	(0.0, 0.0, 1.0, 0.0, 0.0, 0.0)	0.0
1,54E+15	udp	S0	0.0	1.0	76.0	0.0	0.0	*Benign*	1.0	0.0	(0.0, 1.0, 0.0)	(1.0, 0.0, 0.0, 0.0, 0.0, 0.0)	0.0
1,54E+15	udp	S0	0.0	1.0	76.0	0.0	0.0	*Benign*	1.0	0.0	(0.0, 1.0, 0.0)	(1.0, 0.0, 0.0, 0.0, 0.0, 0.0)	0.0
1,55E+15	tcp	OTH	0.0	0.0	0.0	0.0	0.0	*Malicious*	0.0	1.0	(1.0, 0.0, 0.0)	(0.0, 1.0, 0.0, 0.0, 0.0, 0.0)	1.0
1,55E+15	tcp	OTH	0.0	0.0	0.0	0.0	0.0	*Malicious*	0.0	1.0	(1.0, 0.0, 0.0)	(0.0, 1.0, 0.0, 0.0, 0.0, 0.0)	1.0
1,54E+15	udp	S0	0.0	1.0	76.0	0.0	0.0	*Benign*	1.0	0.0	(0.0, 1.0, 0.0)	(1.0, 0.0, 0.0, 0.0, 0.0, 0.0)	0.0
1,54E+15	udp	SF	0.0	1.0	73.0	1.0	73.0	*Benign*	1.0	2.0	(0.0, 1.0, 0.0)	(0.0, 0.0, 1.0, 0.0, 0.0, 0.0)	0.0
1,54E+15	udp	S0	0.0	1.0	76.0	0.0	0.0	*Benign*	1.0	0.0	(0.0, 1.0, 0.0)	(1.0, 0.0, 0.0, 0.0, 0.0, 0.0)	0.0
1,55E+15	tcp	OTH	0.0	0.0	0.0	0.0	0.0	*Malicious*	0.0	1.0	(1.0, 0.0, 0.0)	(0.0, 1.0, 0.0, 0.0, 0.0, 0.0)	1.0
1,54E+15	udp	S0	0.0	2.0	134.0	0.0	0.0	*Benign*	1.0	0.0	(0.0, 1.0, 0.0)	(1.0, 0.0, 0.0, 0.0, 0.0, 0.0)	0.0

4.1.3 Hasil *Vektorisasi* & *Normalisasi*

Pada tahap ini, dilakukan proses *Vektorisasi* dan *Normalisasi* data sebagai bagian dari *tahapan Preprocessing* dalam pengembangan model klasifikasi *Logistic Regression*. *Vektorisasi* diperlukan untuk menggabungkan berbagai *Fitur numerik* yang relevan menjadi satu representasi terstruktur dalam bentuk *vektor* berdimensi tetap, sehingga dapat langsung digunakan sebagai input ke dalam model pembelajaran mesin.

Sebelum dilakukan proses *Vektorisasi*, data yang digunakan masih berada dalam bentuk tabel standar dengan *Fitur-Fitur numerik* yang tersebar di sejumlah kolom terpisah, seperti *missed_bytes, orig_pkts, orig_ip_bytes, resp_pkts,* dan lainnya. Variasi skala antar *Fitur* sangat mencolok, terutama pada *Fitur ts* (*timestamp*) yang memiliki nilai jauh lebih besar dibandingkan *Fitur* lainnya. Struktur data mentah sebelum proses *Vektorisasi* dan *Normalisasi* dapat dilihat pada Tabel 4.6.

Vektorisasi dilakukan menggunakan metode *Vector Assembler*. Proses ini bertujuan untuk menggabungkan seluruh *Fitur numerik* hasil seleksi (kecuali atribut target dan *Fitur* kategorikal yang telah di*Encoding*) ke dalam satu kolom *vektor* bernama *feature_vector*. Dengan kata lain, *Vector Assembler* menyatukan *Fitur* seperti *missed_bytes, orig_pkts, orig_ip_bytes*, dan sebagainya ke dalam sebuah kolom baru bernama *feature_vector*.

Selanjutnya, dilakukan proses *Normalisasi* menggunakan metode *MinMaxScaler*, yang berfungsi untuk menskalakan nilai setiap *Fitur* dalam rentang [0, 1]. Teknik ini sangat bermanfaat untuk menghindari dominasi *Fitur* tertentu yang memiliki nilai absolut besar, seperti *Fitur* waktu (*timestamp*), yang berpotensi menyebabkan *bias* pada proses pemodelan.

Hasil dari proses ini menghasilkan dua kolom baru, yaitu:

- *feature_vector*: representasi *vektor* dari semua *Fitur numerik* dalam satu kolom terintegrasi.
- *normalized_feature_vector*: hasil *Normalisasi* dari *feature_vector* yang telah disesuaikan skalanya agar berada dalam rentang yang seragam.

Tabel 4.7 menampilkan contoh hasil *Vektorisasi* dan *Normalisasi* dari sepuluh entri data. Nilai awal pada kolom *feature_vector* yang memiliki perbedaan skala besar, setelah dinormalisasi menjadi lebih proporsional pada kolom *normalized_feature_vector*. Hal ini mendukung efektivitas model dalam mengenali pola distribusi trafik jaringan yang mengindikasikan adanya aktivitas *Malware*.

Tabel 4. 7 Data Setelah *Vektorisasi* dan *Normalisasi*

feature_vector	*normalized_feature_vector*
(1545433108.456638, 0.0, 1.0, 76.0, 1.0, 76.0,...	(0.9917451351542612, 0.0, 0.000169635284139100...
(1538582545.658519, 0.0, 1.0, 76.0, 0.0, 0.0, ...	(0.0013866695488011737, 0.0, 0.000169635284139...
(1538573455.664419, 0.0, 1.0, 76.0, 0.0, 0.0, ...	(7.256545624349914e-05, 0.0, 0.000169635284139...
(1545466964.126317, 0.0, 0.0, 0.0, 0.0, 0.0, 1...	(0.9966395139174883, 0.0, 0.0, 0.0, 0.0, 0.0, ...
(1545466984.133378, 0.0, 0.0, 0.0, 0.0, 0.0, 1...	(0.9966424062582815, 0.0, 0.0, 0.0, 0.0, 0.0, ...
(1538613756.676788, 0.0, 1.0, 76.0, 0.0, 0.0, ...	(0.005898721634714338, 0.0, 0.0001696352841391...
(1538613675.676472, 0.0, 1.0, 73.0, 1.0, 73.0,...	(0.005887011742961941, 0.0, 0.0001696352841391...
(1538602919.668338, 0.0, 1.0, 76.0, 0.0, 0.0, ...	(0.004332058664589728, 0.0, 0.0001696352841391...
(1545466977.969652, 0.0, 0.0, 0.0, 0.0, 0.0, 1...	(0.9966415151930517, 0.0, 0.0, 0.0, 0.0, 0.0, ...
(1538573274.744539, 0.0, 2.0, 134.0, 0.0, 0.0,...	(4.641059277480936e-05, 0.0, 0.000339270568278...

4.2 Hasil *Splitting* Data

Proses pemisahan data dilakukan untuk membedakan antara data pelatihan yang digunakan dalam pembuatan model klasifikasi dan data pengujian yang berfungsi untuk

mengevaluasi performa model tersebut. Rincian masing-masing skenario pembagian data dapat dilihat pada tabel 3.11.

4.3 Hasil Pemodelan *Logistic Regression*

Pada bagian ini, akan dibahas hasil pemodelan dengan menggunakan data latih serta implementasi dari metode *Logistic Regression* yang telah dibangun menggunakan rumus dasar *Logistic Regression* dengan persamaan 2.1 pada bab 2.

Dalam proses pembangunan model ini, digunakan beberapa komponen penting, seperti fungsi Softmax untuk menghitung probabilitas prediksi kelas, *Cross Entropy Loss* untuk menghitung kesalahan model, serta algoritma *Gradient Descent* sebagai metode optimasi parameter.

Sebelum masuk ke tahap pemodelan, data *Fitur* x*_train dan x_test* serta *label y_train dan y_test* dikonversi terlebih dahulu ke dalam bentuk *NumPy array* agar dapat diproses lebih lanjut dalam pelatihan model. Proses pemodelan dilakukan secara manual dengan pendekatan K-*Fold Cross Validation* sebanyak 5 *Fold*, menggunakan *learning rate* (α) sebesar 0.1 dan *epoch* sebanyak 14 iterasi. Di tahap ini, dilakukan empat skenario pemodelan model seperti pada tabel 3.11.

Pada setiap *Fold*, data latih digunakan untuk melatih model menggunakan fungsi *gradient_descent*, lalu hasilnya diuji pada data *validasi* guna menghitung metrik evaluasi dari setiap lipatan dalam *Cross Validation*.

4.3.1 Skenario 1

Pada Skenario 1, model *Logistic Regression* dilatih menggunakan data latih dengan perbandingan *rasio split* data 80:20 secara manual sesuai urut baris data. Untuk

mengukur kemampuan *generalisasi* model lebih lanjut, digunakan teknik 5-*Fold Cross Validation*. Hasil dari setiap *Fold* disajikan dalam tabel 4.8 :

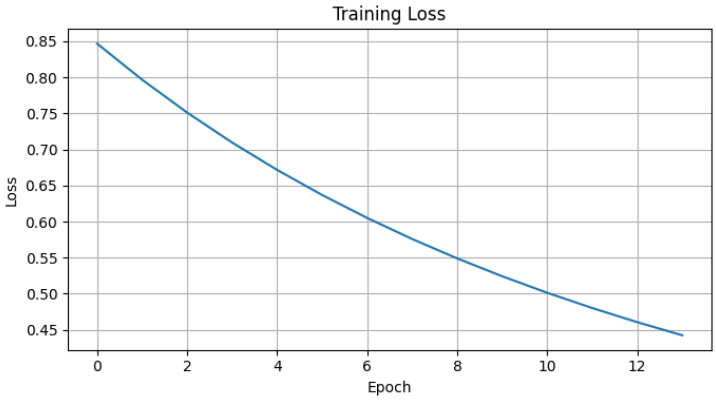
Tabel 4. 8 Hasil Evaluasi *Cross Validation* – Skenario 1

***Fold* ke-**	***Akurasi***	***Presisi***	***Recall***	***F1-score***
1	90.98	92.39	90.98	90.93
2	91.04	92.40	91.04	90.99
3	89.95	91.61	89.95	89.83
4	91.03	92.41	91.03	90.97
5	98.38	98.43	98.38	98.37
Rata-rata	**92.28**	**93.45**	**92.28**	**92.22**

Tabel tersebut menunjukkan bahwa model mampu memberikan hasil yang cukup stabil pada setiap *Fold*, dengan *akurasi* yang berkisar antara 89,95% hingga 98,38%. Rata-rata *akurasi* yang dicapai sebesar 92,28% menunjukkan bahwa model memiliki kemampuan sangat baik dalam mengklasifikasikan trafik jaringan. Nilai *presisi* yang tinggi, yaitu 93,45%, menunjukkan bahwa sebagian besar prediksi terhadap trafik yang dianggap sebagai *Malware* memang benar-benar merupakan *Malware*. Artinya, tingkat kesalahan deteksi terhadap trafik yang aman tergolong sangat rendah. Sementara itu, *recall* sebesar 92,28% menandakan bahwa sebagian besar *Malware* dalam data berhasil dikenali dengan baik oleh model. Kombinasi dari *presisi* dan *recall* tersebut menghasilkan nilai *F1-score* sebesar 92,22%, yang menunjukkan bahwa model memiliki performa klasifikasi yang sangat seimbang. Secara keseluruhan, hasil ini menunjukkan bahwa meskipun pemodelan dilakukan hingga 14 *epoch*, model sudah menunjukkan kemampuan klasifikasi yang kuat dan konsisten.

Untuk menggambarkan proses pembelajaran model dalam skenario ini, Gambar 4.3 menunjukkan grafik penurunan nilai *Loss* selama proses pemodelan berlangsung.

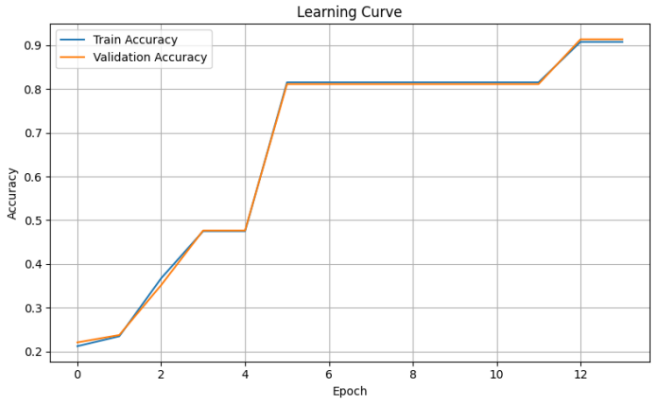
Grafik ini memberikan gambaran bagaimana model secara bertahap melakukan penyesuaian terhadap parameter bobot guna meminimalkan kesalahan prediksi.



Gambar 4. 3 *Visualisasi* Nilai *Loss* – Skenario 1

Dari grafik tersebut terlihat bahwa nilai *Loss* awal berada di kisaran 0.85, dan secara bertahap mengalami penurunan hingga mencapai nilai sekitar 0.44 pada akhir *epoch* ke-14. *Loss* sendiri merupakan indikator dari besarnya kesalahan prediksi model terhadap data aktual. Penurunan nilai *Loss* secara konsisten menggambarkan bahwa model berhasil menyesuaikan bobot-bobot internalnya untuk meminimalkan kesalahan selama proses pemodelan. Hal ini menandakan bahwa proses pembelajaran berjalan efektif, dan model mampu menyerap informasi dari data dengan baik.

Selain melihat penurunan *Loss* , performa model pada Skenario 1 juga dianalisis melalui *Visualisasi Learning curve* yang ditunjukkan pada Gambar 4.4. Grafik ini membandingkan *akurasi* model pada data pemodelan (training accuracy) dan data validasi (validation accuracy) sepanjang proses pemodelan selama 14 *epoch*.



Gambar 4. 4 *Visualisasi Learning curve* – Skenario 1

Dari grafik *Learning curve* tersebut terlihat bahwa *akurasi* awal model berada di angka sekitar 22%. Namun, terjadi peningkatan yang signifikan terutama setelah *epoch* ke-4. Pada *epoch* ke-5, *akurasi* langsung naik tajam hingga mencapai sekitar 81%, dan terus meningkat hingga akhir *epoch* ke-14 dengan *akurasi* pelatihan dan validasi keduanya berada di angka 91%. Kenaikan yang seimbang antara kedua *akurasi* ini menunjukkan bahwa model mampu belajar secara bertahap dari data tanpa mengalami *overfitting*. Artinya, model tidak hanya hafal terhadap data latih, tetapi juga mampu melakukan prediksi yang baik terhadap data baru.

4.3.2 Skenario 2

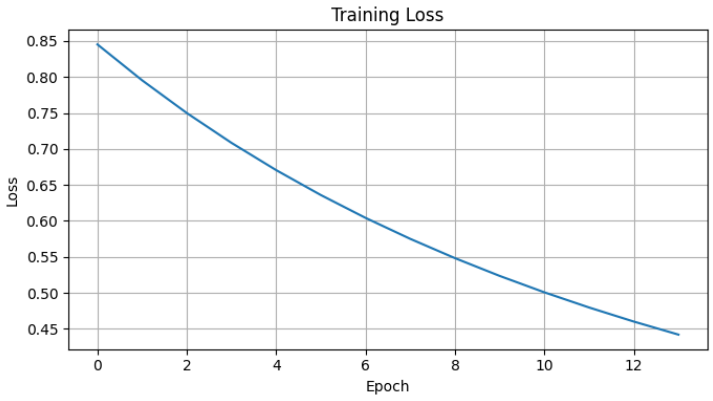
Pada Skenario 2, model *Logistic Regression* dilatih ulang dengan jumlah *epoch* menjadi 14 dan learning rate sebesar 0.1. Pengujian dilakukan menggunakan teknik 5-*Fold Cross Validation* untuk mengukur konsistensi performa model. Hasil evaluasi ditampilkan pada Tabel 4.9 berikut:

Tabel 4. 9 Hasil Evaluasi *Cross Validation* – Skenario 2

***Fold* ke-**	***Akurasi***	***Presisi***	***Recall***	***F1-score***
1	90.88%	92.33%	90.88%	90.83%
2	91.35%	92.66%	91.35%	91.31%
3	91.17%	92.47%	91.17%	91.08%
4	91.77%	92.92%	91.77%	91.73%
5	89.62%	91.31%	89.62%	89.47%
Rata-rata	**90.96%**	**92.34%**	**90.96%**	**90.88%**

Secara umum, model menunjukkan performa yang konsisten dan stabil di tiap *Fold*, dengan *presisi* dan *recall* yang tinggi. Nilai rata-rata *presisi* sebesar 92.34% mengindikasikan bahwa prediksi *Malware* oleh model umumnya benar, dengan risiko *false positive* yang rendah. Sementara itu, *recall* sebesar 90.96% menandakan bahwa sebagian besar *Malware* berhasil dikenali oleh model. Kombinasi ini menghasilkan *F1-score* rata-rata 90.88%, mencerminkan keseimbangan yang baik antara *akurasi* deteksi dan ketepatan prediksi.

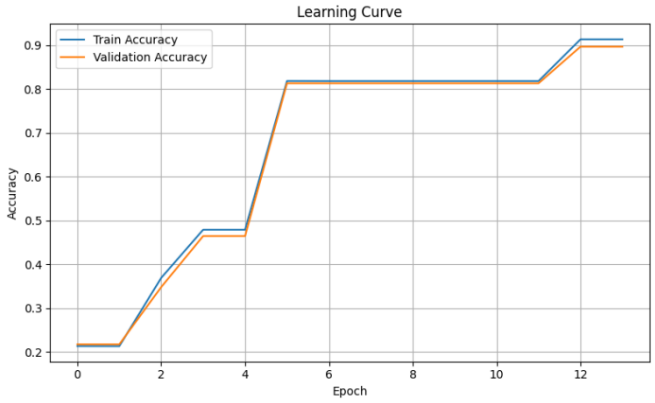
Untuk menggambarkan proses pembelajaran model selama training, Gambar 4.5 menunjukkan *Visualisasi* penurunan nilai *Loss* serta *Learning curve akurasi* terhadap data latih dan validasi.



Gambar 4. 5 *Visualisasi* Nilai *Loss* – Skenario 2

Grafik ini hampir sama dengan skenario 1 yang menunjukkan penurunan *Loss* dari sekitar 0.85 di awal hingga sekitar 0.44 di akhir *epoch* ke-14, yang menandakan bahwa model berhasil meminimalkan kesalahan secara stabil dan efektif selama proses pelatihan.

Selain nilai *Loss* , Gambar 4.6 berikut menyajikan *Learning curve*, yaitu grafik *akurasi* model terhadap data pemodelan dan data *validasi* selama proses pemodelan berlangsung.



Gambar 4. 6 *Visualisasi Learning curve* – Skenario 2

Dari grafik learning curve, *akurasi* model meningkat signifikan setelah *epoch* ke-4, dan pada akhir pelatihan tercapai *akurasi* sekitar 91% pada data latih dan 90% pada data validasi. Perbedaan yang sangat kecil antara keduanya menunjukkan bahwa model tidak mengalami *overfitting* dan memiliki kemampuan generalisasi yang sangat baik.

Secara keseluruhan, Skenario 2 memberikan hasil yang hampir sama dengan skenario 1. Yang membedakan hanya terdapat pada *epoch* 1 ke *epoch* 5 dan pada akhir *epoch* 12 ke 14.

4.3.3 Skenario 3

Pada Skenario 3, model *Logistic Regression* dilatih menggunakan metode *random split* dengan *rasio* data latih dan uji sebesar 70:30. Tujuan dari skenario ini adalah untuk mengevaluasi apakah variasi proporsi data pelatihan dan pengujian yang lebih seimbang namun tetap berbasis *random split* dapat mendorong peningkatan *akurasi* model secara keseluruhan, dibandingkan skenario 2.

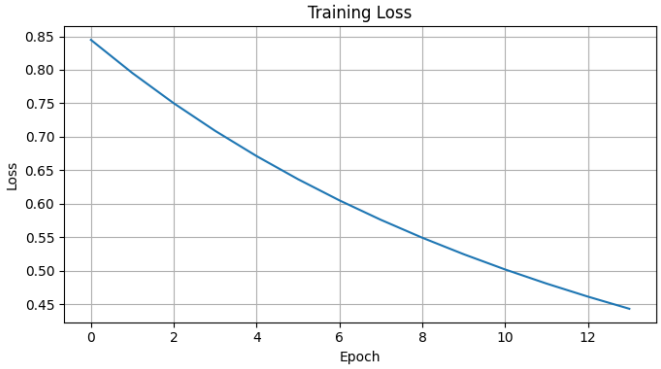
Evaluasi performa dilakukan menggunakan teknik 5-*Fold Cross Validation*, sebagaimana ditampilkan dalam Tabel 4.10 berikut:

Tabel 4. 10 Hasil Evaluasi *Cross Validation* – Skenario 3

***Fold* ke-**	***Akurasi***	***Presisi***	***Recall***	***F1-score***
1	90.92%	92.36%	90.92%	90.87%
2	91.66%	92.85%	91.66%	91.60%
3	90.31%	91.84%	90.31%	90.22%
4	90.37%	91.97%	90.37%	90.31%
5	94.03%	94.62%	94.03%	94.00%
Rata-rata	**91.46%**	**92.73%**	**91.46%**	**91.40%**

Hasil tersebut menunjukkan bahwa model menunjukkan peningkatan *akurasi* dibandingkan skenario sebelumnya (khususnya Skenario 1 dan 2), dengan rata-rata *akurasi* mencapai 91.46% dan nilai *precision*, *recall*, serta *F1-score* semuanya berada di atas 90%. Rentang performa antar *Fold* cukup stabil, dengan fluktuasi yang wajar dan tidak ekstrem, yang menandakan konsistensi kinerja model terhadap variasi data pelatihan.

Untuk memahami proses pembelajaran selama pemodelan, Gambar 4.7 berikut menunjukkan grafik nilai *Loss* terhadap jumlah *epoch*:

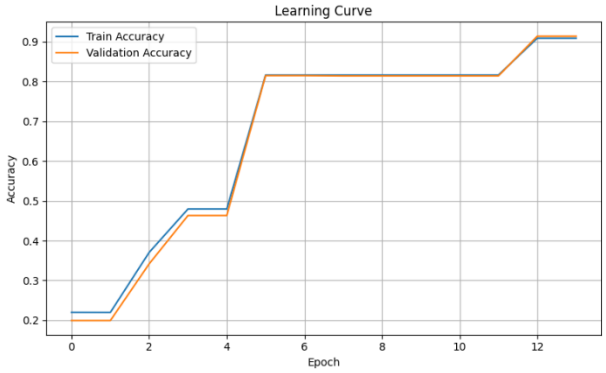


Gambar 4. 7 *Visualisasi* Nilai *Loss* – Skenario 3

Grafik ini menunjukkan bahwa nilai *Loss* mengalami penurunan secara bertahap dari awal pelatihan hingga akhir *epoch* ke-14. Dimulai dari sekitar 0.85, *Loss* terus menurun hingga mencapai sekitar 0.44, yang menunjukkan bahwa model secara konsisten mampu memperbaiki kesalahan prediksi selama proses pelatihan. Pola penurunan ini mirip dengan tren pada skenario sebelumnya, di mana tidak terjadi lonjakan atau fluktuasi besar, menandakan bahwa proses pembelajaran berjalan secara stabil.

Meskipun grafik ini menyerupai skenario sebelumnya, hal yang membedakan adalah setting pembagian data yang lebih bervariasi (*random split* 70:30), yang memungkinkan model mengoptimalkan bobotnya berdasarkan sebaran data yang lebih luas. Oleh karena itu, meskipun bentuk kurva serupa, konteks dan dampaknya terhadap generalisasi model menjadi lebih signifikan pada Skenario 3.

Selain grafik *Loss* , Gambar 4.8 berikut menampilkan grafik *learning curve*, yaitu perkembangan *akurasi* model terhadap data pemodelan dan data *validasi* selama proses pemodelan berlangsung:



Gambar 4. 8 *Visualisasi Learning curve* – Skenario 3

Grafik Learning *curve* pada skenario ini memperlihatkan pola yang hampir sama dengan skenario sebelumnya, namun hanya berbeda pada akhir *epoch* bahwa pada skenario ini perbandingan antara train *akurasi* dan validation *akurasi* hanya berbeda sedikit saja. Ini mengindikasikan bahwa model tidak mengalami *overfitting* serta memiliki kemampuan generalisasi yang sangat baik terhadap data uji.

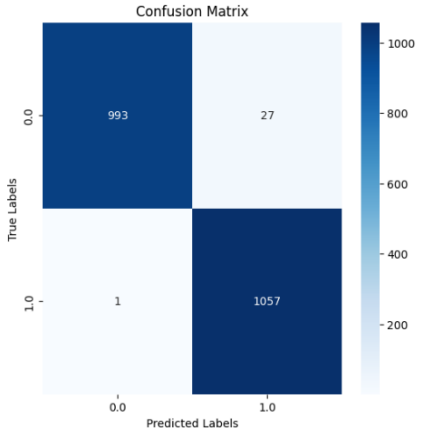
Kemampuan model untuk tetap menjaga konsistensi antara *akurasi* pelatihan dan validasi pada skenario ini membuktikan bahwa konfigurasi *random split* yang lebih ketat sekalipun tidak menurunkan stabilitas pembelajaran. Hal ini memperkuat keyakinan bahwa model dapat diandalkan dalam menghadapi data baru di lingkungan nyata.

4.4 Hasil Pengujian Model *Logistic Regression*

Pada bagian ini akan dibahas hasil evaluasi model *Logistic Regreesion*. Hasil evaluasi performa model didapatkan berdasarkan perbandingan antara nilai *y_test_pred* (data aktual) dan *y_test_pred_labels* (data hasil prediksi) sebagai tolak ukur tingkat keberhasilan metode.

4.4.1 Skenario 1

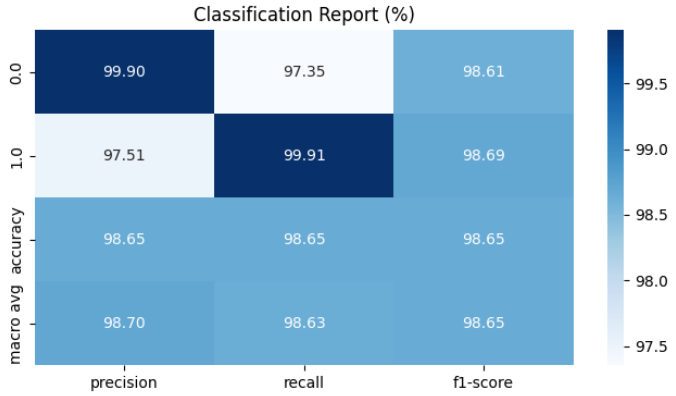
Pengujian model *Logistic Regression* pada Skenario 1 dilakukan dengan menggunakan data uji dari hasil pemisahan manual dengan *rasio* 80:20 untuk menilai kemampuan prediksi model terhadap data yang belum pernah dilihat. Evaluasi pertama divisualisasikan melalui *Confusion matrix* sebagaimana ditunjukkan pada Gambar 4.9:



Gambar 4. 9 Hasil *Confusion matrix* – Skenario 1

Confusion matrix menunjukkan bahwa model berhasil mengklasifikasikan 993 data *benign* (kelas 0) dan 1.057 data *Malware* (kelas 1) dengan benar. Namun, masih terdapat 27 data *benign* yang diklasifikasikan salah sebagai *Malware* (*false positive*) dan 1 data *Malware* yang diklasifikasikan sebagai *benign* (false negative). Meskipun jumlah kesalahan cukup kecil, hal ini tetap memiliki implikasi penting, terutama pada sistem deteksi di dunia nyata yang membutuhkan *presisi* tinggi untuk menghindari alarm palsu atau kesalahan deteksi.

Evaluasi metrik secara kuantitatif ditampilkan dalam Gambar 4.10 melalui *Classification report* yang berisi nilai *precision*, *recall*, dan *F1-score* untuk masing-masing kelas.



Gambar 4. 10 Hasil *Classification report* – Skenario 1

Dari *Classification report*, *precision* untuk kelas *benign* (0) mencapai 99.90% dan *recall*-nya sebesar 97.35%, yang berarti sebagian besar prediksi terhadap trafik normal sangat akurat dan hanya sedikit data *benign* yang salah diklasifikasikan. Untuk kelas *Malware* (1), *precision*-nya sebesar 97.51% dan *recall* mencapai 99.91%, menandakan kemampuan model yang hampir sempurna dalam mengenali *Malware*, serta sangat minim dalam menghasilkan false negative.

Nilai rata-rata keseluruhan metrik menunjukkan hasil yang sangat tinggi, yaitu *precision* sebesar 98.70%, *recall* sebesar 98.63%, dan *F1-score* sebesar 98.65%. Hasil ini tampak sangat baik dan mencerminkan model yang mampu melakukan klasifikasi dengan tingkat kesalahan sangat rendah, serta hasil *akurasi* model yang tinggi 98.65%.

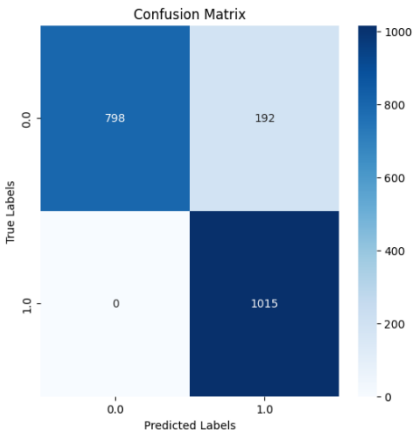
Namun, jika dibandingkan dengan hasil evaluasi saat proses training menggunakan 5-*Fold Cross Validation* (rata-rata *akurasi* 92.28%, *precision* 93.45%, *recall* 92.28%, dan *F1-score* 92.22%), terlihat adanya perbedaan performa yang cukup signifikan. Meskipun hasil pengujian akhir lebih tinggi, hal ini justru menunjukkan adanya ketidaksesuaian atau ketidakstabilan akibat metode pembagian data secara manual yang mungkin mengandung bias atau tidak mewakili sebaran data sebenarnya.

Performa tinggi pada data uji yang sangat kontras dengan hasil training (cross-validation) mengindikasikan bahwa model bisa jadi mengalami *overfitting* terhadap data uji karena proses pembagian data yang tidak acak. Oleh karena itu, penting untuk melakukan pengujian tambahan menggunakan metode pembagian data yang lebih merata, seperti *random split*, agar hasil evaluasi benar-benar mencerminkan performa model yang general dan tidak hanya unggul pada subset tertentu dari data.

Dengan temuan ini, maka pengujian dilanjutkan pada Skenario 2, menggunakan *random split* untuk memastikan kestabilan dan keakuratan model dalam mendeteksi trafik jaringan secara lebih realistis.

4.4.2 Skenario 2

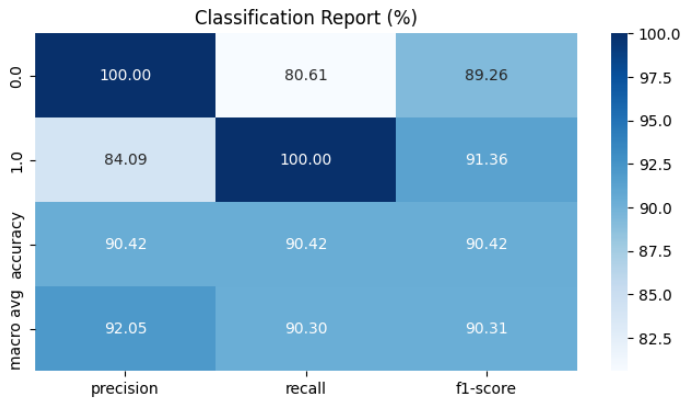
Pengujian pada Skenario 2 dilakukan untuk mengevaluasi apakah perubahan metode pembagian data dari manual menjadi *random split* dengan *rasio* 80:20 dapat memberikan dampak positif terhadap performa model klasifikasi, khususnya dalam mengurangi kesalahan klasifikasi terhadap trafik *benign* (*false positive*) tanpa menurunkan kemampuan dalam mendeteksi trafik *Malware*. Hasil pengujian divisualisasikan melalui *Confusion matrix* pada Gambar 4.11 berikut:



Gambar 4. 11 Hasil *Confusion matrix* – Skenario 2

Berdasarkan *Confusion matrix* tersebut, model berhasil mengklasifikasikan 798 data *benign* (kelas 0) secara benar dan 1.015 data *Malware* (kelas 1) juga diklasifikasikan secara tepat. Meskipun demikian, masih terdapat 192 data *benign* yang salah diklasifikasikan sebagai *Malware* (*false positive*), dan tidak ditemukan satu pun false negative, yang berarti setiap sampel *Malware* berhasil dideteksi. Hal ini mengindikasikan bahwa kemampuan model dalam mendeteksi serangan (*recall* untuk kelas *Malware*) tetap berada pada level maksimal, yaitu 100%, sebagaimana halnya pada Skenario 1. Namun yang membedakan, jumlah kesalahan dalam mengenali trafik *benign* mengalami penurunan signifikan, yaitu dari 390 menjadi 192 kasus, yang menandakan adanya peningkatan kemampuan model dalam membedakan antara trafik normal dan trafik berbahaya.

Evaluasi performa model secara kuantitatif ditunjukkan melalui *Classification report* yang disajikan pada Gambar 4.12:



Gambar 4. 12 Hasil *Classification report* – Skenario 2

Berdasarkan *Classification report* tersebut, diketahui bahwa *precision* untuk kelas 0 (*benign*) mencapai 100.00%, artinya seluruh prediksi *benign* yang dihasilkan oleh model adalah benar. Sementara itu, *precision* untuk kelas 1 (*Malware*) berada di angka 84.09%, menunjukkan bahwa sebagian kecil prediksi *Malware* merupakan kesalahan

(*false positive*). Dari sisi *recall*, terjadi peningkatan signifikan untuk kelas 0 (*benign*) menjadi 80.61% dibandingkan skenario sebelumnya, yang mengindikasikan bahwa model kini lebih andal dalam mengenali trafik *benign*. Sementara itu, *recall* untuk kelas 1 tetap berada di angka 100.00%, menandakan tidak adanya data *Malware* yang gagal dikenali.

Kombinasi antara *precision* dan *recall* menghasilkan *F1-score* sebesar 89.26% untuk *benign* dan 91.36% untuk *Malware*. Secara keseluruhan, model mencapai *akurasi* makro rata-rata sebesar 90.42%, serta nilai *precision* rata-rata sebesar 92.05%, nilai *recall* rata-rata 90.30%, dan *f1-score* rata-rata 90.31% yang menunjukkan kestabilan performa model dalam mengklasifikasikan kedua kelas secara proporsional.

Hal menarik dari Skenario 2 adalah bahwa model tidak hanya menunjukkan performa baik pada data uji, tetapi juga konsisten selama proses pelatihan. Hasil evaluasi melalui 5-*Fold Cross Validation* menghasilkan nilai *akurasi* rata-rata 90.96%, *precision* 92.34%, *recall* 90.96%, dan *F1-score* 90.88%. Nilai-nilai ini sangat mendekati hasil evaluasi akhir pada data uji (*akurasi* 90.42%, *precision* 92.05%, *recall* 90.30%, *F1-score* 90.31%), yang menunjukkan bahwa model memiliki kestabilan dan generalisasi yang baik.

Konsistensi ini berbeda dengan temuan pada Skenario 1, di mana terdapat selisih performa yang cukup mencolok antara hasil training (cross-validation) dan hasil pengujian akhir. Pada Skenario 1, metode manual *split* menyebabkan hasil pada data uji tampak lebih tinggi dari rata-rata cross-validation, yang bisa mengindikasikan adanya bias atau ketidakseimbangan dalam pembagian data. Hal tersebut menimbulkan potensi *overfitting* terhadap subset tertentu dari data uji, sehingga performa model tampak tinggi tetapi tidak mencerminkan kemampuan generalisasi yang sesungguhnya.

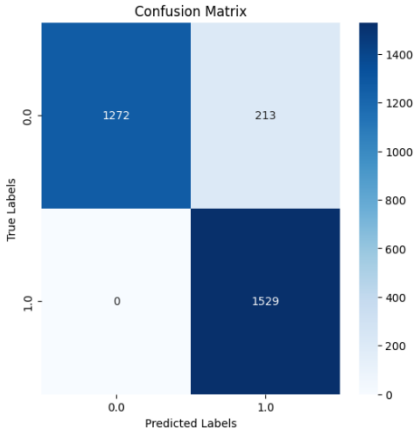
Sebaliknya, pada Skenario 2, penggunaan metode *random split* yang membagi data secara acak memberikan sebaran yang lebih representatif dan merata terhadap seluruh distribusi data. Hal ini memungkinkan model untuk belajar dari variasi data yang lebih luas dan mengurangi kemungkinan bias. Dengan demikian, performa pada tahap pelatihan dan pengujian menjadi lebih selaras, yang menjadi indikator penting dari kestabilan model dalam konteks penerapan nyata.

Temuan ini memperkuat bahwa konfigurasi Skenario 2 tidak hanya akurat, tetapi juga lebih dapat diandalkan dalam berbagai kondisi data, menjadikannya lebih layak untuk diterapkan dalam sistem klasifikasi trafik jaringan secara real-time. Untuk menguji lebih lanjut kemampuan adaptasi model terhadap perubahan proporsi data latih dan uji, dilakukan pengujian lanjutan pada Skenario 3 dengan *rasio* pembagian data yang berbeda, yaitu 70:30.

4.4.3 Skenario 3

Skenario 3 merupakan lanjutan dari pengujian sebelumnya, dengan pendekatan utama berupa perubahan *rasio* pembagian data menggunakan metode *random split* 70:30. Tujuan dari skenario ini adalah untuk menguji konsistensi dan kemampuan generalisasi model ketika dihadapkan pada proporsi data uji yang lebih besar. Hal ini menjadi penting untuk menilai keandalan model dalam konteks nyata, di mana data baru yang belum pernah dilatih sebelumnya dapat muncul dalam jumlah yang signifikan. Tidak terdapat perubahan jumlah *epoch* dibandingkan skenario sebelumnya, sehingga segala perbedaan performa murni berasal dari variasi dalam metode pembagian data.

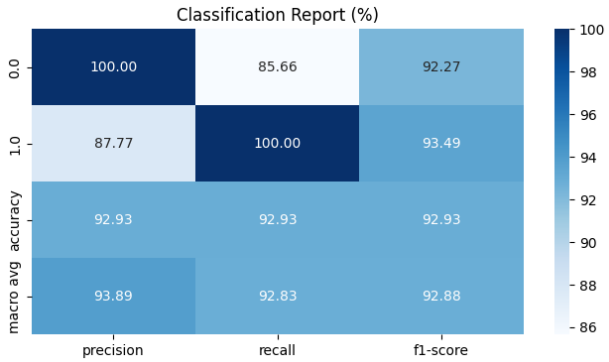
Evaluasi performa model ditunjukkan melalui *Confusion matrix* yang disajikan pada Gambar 4.13 berikut:



Gambar 4. 13 Hasil *Confusion matrix* – Skenario 3

Berdasarkan hasil *confusion matrix*, model berhasil mengklasifikasikan 1.272 data *benign* (kelas 0) dengan tepat, serta seluruh 1.529 data *Malware* (kelas 1) juga berhasil diidentifikasi secara akurat. Namun, masih ditemukan 213 data *benign* yang salah diklasifikasikan sebagai *Malware* (*false positive*), sementara tidak terdapat false negative pada kelas *Malware*. Hal ini menandakan bahwa model mempertahankan kemampuan deteksi maksimal terhadap trafik berbahaya (*recall* = 100%), sekaligus memperlihatkan adanya peningkatan jumlah kesalahan dalam mengidentifikasi trafik normal dibandingkan skenario sebelumnya.

Penilaian lebih lanjut terhadap performa model ditampilkan pada *Classification report* dalam Gambar 4.14:



Gambar 4. 14 Hasil *Classification report* – Skenario 3

Dari laporan klasifikasi tersebut, diketahui bahwa *precision* untuk kelas *benign* (0) mencapai 100.00%, mengindikasikan bahwa semua prediksi yang menyatakan suatu trafik sebagai *benign* adalah benar. Sementara itu, *precision* untuk kelas *Malware* (1) berada pada angka 87.77%, menunjukkan bahwa terdapat sejumlah prediksi *Malware* yang keliru karena merupakan *false positive*. Dari sisi *recall*, kelas *benign* mencatatkan nilai 85.66%, yang berarti sebagian data *benign* masih terklasifikasi salah, sedangkan kelas *Malware* mempertahankan *recall* sebesar 100.00%, tanpa ada satu pun data yang terlewat. Kombinasi dari *precision* dan *recall* menghasilkan nilai *F1-score* sebesar 92.27% untuk kelas *benign* dan 93.49% untuk kelas *Malware*. Dan untuk *akurasi* keseluruhan model mencapai 92.93% meningkat sekitar 2% dibanding skenario sebelumnya.

Dengan demikian, Skenario 3 memperlihatkan bahwa perubahan *rasio* pembagian data menjadi 70:30 melalui metode *random split* berhasil menjaga performa model dalam level yang tinggi, baik dari segi kemampuan deteksi *Malware* maupun kestabilan metrik klasifikasi. Temuan ini mendukung teori bahwa distribusi data yang lebih representatif dapat meningkatkan kemampuan model dalam melakukan generalisasi terhadap data yang sebelumnya tidak terlihat, sehingga relevan untuk diterapkan dalam sistem klasifikasi trafik jaringan pada skala operasional nyata.

4.5 Pembahasan Hasil Pengujian

Pengujian terhadap model *Logistic Regression* dilakukan melalui tiga skenario berbeda dengan pendekatan yang berfokus pada metode pembagian data (data *split*). Meskipun jumlah *epoch* yang digunakan pada ketiga skenario adalah sama, perbedaan performa model dianalisis berdasarkan variasi metode dan *rasio split* data: Skenario 1 menggunakan manual *split* 80:20, Skenario 2 menggunakan *random split* 80:20, dan

Skenario 3 menggunakan *random split* 70:30. Evaluasi dilakukan dengan mengukur empat metrik utama, yaitu *akurasi*, *precision*, *recall*, dan *F1-score*, karena keempatnya merepresentasikan keseimbangan antara kemampuan deteksi ancaman (*Malware*) dan kemampuan mengenali trafik normal (*benign*). Tabel 4.11 berikut merangkum hasil evaluasi model pada ketiga skenario:

Tabel 4. 11 Hasil Pengujian Model

Skenario	***Split* Data**	***Akurasi***	***Precision***	***Recall***	***F1-score***
1	Manual 80:20	98.65%	98.70% (avg)	98.63% (avg)	98.65% (avg)
2	*Random* 80:20	90.42%	92.05% (avg)	90.30% (avg)	90.31% (avg)
3	*Random* 70:30	92.93%	93.89% (avg)	92.83% (avg)	92.88% (avg)

Pada Skenario 1, model menunjukkan performa evaluasi yang sangat tinggi pada data uji, dengan *precision* untuk kelas *benign* (0) sebesar 99.90% dan *recall* sebesar 97.35%. Untuk kelas *Malware* (1), *precision* tercatat 97.51% dan *recall* mencapai 99.91%. Nilai rata-rata metrik juga sangat tinggi, yaitu *precision* 98.70%, *recall* 98.63%, dan *F1-score* 98.65%. Meskipun hasil ini tampak sangat baik, performa ini justru berbeda signifikan dengan hasil saat pelatihan menggunakan 5-*Fold Cross Validation*, yang hanya mencatat rata-rata *akurasi* 92.28%, *precision* 93.45%, *recall* 92.28%, dan *F1-score* 92.22%.

Perbedaan mencolok ini mengindikasikan bahwa model pada Skenario 1 mengalami *overfitting* terhadap data uji, akibat metode pembagian data secara manual yang cenderung bias dan tidak mewakili distribusi data secara menyeluruh. Artinya, model tampil sangat baik hanya pada subset data tertentu, namun kurang stabil dan tidak general ketika dihadapkan pada data baru. Oleh karena itu, metode manual *split* tidak disarankan dalam proses pelatihan model klasifikasi trafik jaringan karena hasilnya tidak konsisten dan kurang dapat diandalkan dalam implementasi nyata.

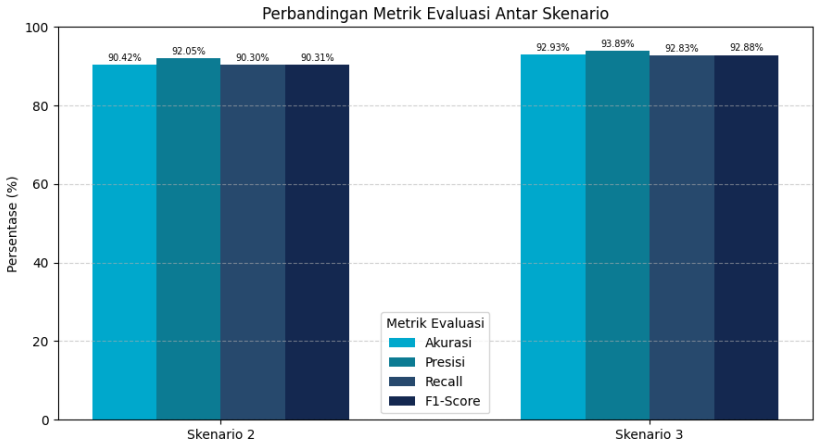
Skenario 2 menggunakan pendekatan *random split* dengan *rasio* data yang sama (80:20), dan hasil pengujian menunjukkan performa yang lebih realistis dan stabil. *Precision* untuk kelas *benign* mencapai 100.00%, dan *precision* untuk kelas *Malware* sebesar 84.09%, dengan *recall* untuk kelas *Malware* tetap di 100.00%. *Recall* untuk kelas *benign* juga meningkat menjadi 80.61%, menghasilkan *F1-score* sebesar 89.26% untuk kelas *benign* dan 91.36% untuk *Malware*.

Performa model pada data uji selaras dengan hasil evaluasi *Cross Validation*, yang mencatat *akurasi* rata-rata 90.96%, *precision* 92.34%, *recall* 90.96%, dan *F1-score* 90.88%. Kedekatan nilai ini menunjukkan bahwa model memiliki kemampuan generalisasi yang baik dan tidak hanya unggul pada subset data tertentu. Ini membuktikan bahwa metode *random split* memberikan distribusi data yang lebih merata dan representatif, sehingga menghasilkan model yang lebih seimbang dan andal.

Pada Skenario 3, *rasio* data uji ditingkatkan menjadi 30% untuk menguji ketahanan model terhadap data baru yang lebih besar. Hasil pengujian tetap tinggi dan konsisten, dengan *precision* kelas *benign* 100.00% dan kelas *Malware* 87.77%. *Recall* untuk *benign* mencapai 85.66%, dan *Malware* tetap di 100.00%, menghasilkan *F1-score* sebesar 92.27% (*benign*) dan 93.49% (*Malware*). Nilai *akurasi* keseluruhan mencapai 92.93%, meningkat sekitar 2% dibandingkan Skenario 2.

Performa ini menunjukkan bahwa perubahan *rasio* pembagian data tidak menurunkan kualitas klasifikasi, bahkan dalam kondisi data uji yang lebih besar. Hal ini mendukung teori bahwa pembagian data secara acak dengan proporsi yang lebih luas memungkinkan model untuk belajar dari variasi data yang lebih kompleks, serta tetap menjaga *akurasi* dan stabilitas.

Perbandingan metrik evaluasi antar skenario divisualisasikan pada Gambar 4.15 berikut:



Gambar 4. 15 *Visualisasi* Perbandingan antar Skenario

Gambar 4.15 menampilkan perbandingan empat metrik evaluasi utama *akurasi*, *presisi*, *recall*, dan *F1-score* antara Skenario 2 dan Skenario 3. *Visualisasi* ini dibuat untuk memperlihatkan tren peningkatan performa model ketika menggunakan *rasio* data uji yang lebih besar dengan metode pembagian data yang tetap menggunakan *random split*.

Berdasarkan grafik, terlihat bahwa Skenario 3 secara konsisten unggul pada seluruh metrik evaluasi dibandingkan Skenario 2. Pada Skenario 2, model mencatat *akurasi* 90.42%, *precision* 92.05%, *recall* 90.30%, dan *F1-score* 90.31%. Sementara itu, pada Skenario 3, performa meningkat menjadi *akurasi* 92.93%, *precision* 93.89%, *recall* 92.83%, dan *F1-score* 92.88%. Kenaikan ini memperlihatkan bahwa model tidak hanya mampu mempertahankan kestabilan meskipun porsi data uji diperbesar (dari 20% menjadi 30%), tetapi juga menunjukkan kemampuan generalisasi yang lebih baik dan konsistensi klasifikasi yang lebih akurat terhadap data yang belum pernah dilihat.

Penting untuk dicatat bahwa Skenario 1 tidak disertakan dalam grafik ini. Hal ini dilakukan secara sengaja karena performa model pada skenario tersebut, meskipun tinggi secara angka evaluasi akhir (98.65%), tidak mencerminkan performa yang stabil dan realistis. Perbedaan yang besar antara hasil evaluasi data uji dan hasil *Cross Validation* menunjukkan adanya indikasi *overfitting*, yang disebabkan oleh metode pembagian data manual yang tidak representatif. Oleh karena itu, Skenario 1 dinilai tidak efisien dan tidak layak dijadikan tolok ukur untuk *Visualisasi* performa model yang sebenarnya, terutama dalam konteks penerapan sistem klasifikasi trafik jaringan di dunia nyata.

Dengan demikian, *Visualisasi* ini menegaskan bahwa metode *random split* yang digunakan dalam Skenario 2 dan 3 memberikan hasil yang lebih akurat, stabil, dan dapat digeneralisasikan, dengan performa terbaik ditunjukkan oleh Skenario 3. Pendekatan ini relevan dan layak diterapkan pada sistem deteksi *Malware* yang menuntut *presisi* tinggi dan keandalan klasifikasi terhadap trafik jaringan yang dinamis.

Dalam penelitian ini, digunakan pendekatan klasifikasi yang berakar pada perspektif Islam, di mana prinsip-prinsipnya diperkuat oleh nilai-nilai Al-Qur'an. Landasan spiritual yang mendukung konsep klasifikasi ini terdapat dalam Surah Fussilat ayat 34, yang berbunyi:

وَلَا تَسْتَوِى الْحَسَنَةُ وَلَا السَّيِّئَةُۗ اِدْفَعْ بِالَّتِيْ هِيَ اَحْسَنُ فَاِذَا الَّذِيْ بَيْنَكَ وَبَيْنَهٗ عَدَاوَةٌ كَاَنَّهٗ وَلِيٌّ حَمِيْمٌ ۝

"Tidaklah sama kebaikan dengan kejahatan. Tolaklah (kejahatan) dengan perilaku yang lebih baik sehingga orang yang ada permusuhan denganmu serta-merta menjadi seperti teman yang sangat setia.'" (QS. Al-Fussilat, 34:34)

Dalam tafsir Ibnu Katsir, ayat ini menunjukkan perbedaan hakiki antara dua hal yang berlawanan: kebaikan dan kejahatan. Maka, secara konseptual, proses klasifikasi dalam penelitian ini yang membedakan trafik baik dan jahat merupakan refleksi nyata

dari prinsip yang diajarkan ayat tersebut yaitu untuk mampu mengidentifikasi dan memperlakukan keburukan dengan pendekatan yang terbaik, tanpa mencampuradukkan antara yang benar dan yang salah. (Tafsir Ibnu Katsir Jilid 7 : 215)

Dengan demikian, integrasi nilai Islam ke dalam penelitian ini bukan hanya simbolis, melainkan memberi dimensi etis dan spiritual dalam penerapan teknologi. Klasifikasi terhadap trafik jaringan menjadi bentuk nyata dalam mewujudkan nilai-nilai Al-Qur'an dalam teknologi, yakni menjaga kebaikan, mencegah kerusakan, dan bertindak berdasarkan prinsip kebenaran.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Hasil penelitian menunjukkan bahwa model *Logistic Regression* mampu memberikan performa klasifikasi yang cukup efektif dalam membedakan antara trafik *benign* dan *Malware*. Meskipun Skenario 1 menunjukkan nilai evaluasi yang sangat tinggi pada data uji (*akurasi* 98.65%), hasil ini tidak konsisten dengan hasil cross-validation, sehingga mengindikasikan adanya *overfitting* akibat metode pembagian data manual yang cenderung bias dan tidak representatif. Sebaliknya, pada Skenario 2 dan Skenario 3, penggunaan metode *random split* mampu menghasilkan model yang lebih stabil dan general terhadap data yang belum pernah dilihat sebelumnya.

Kinerja terbaik diperoleh pada Skenario 3, yaitu ketika model dilatih menggunakan metode *random split* dengan *rasio* data 70:30. Pada skenario ini, model mencapai *akurasi* 92.93%, *precision* 93.89%, *recall* 92.83%, dan *F1-score* 92.88%, serta menunjukkan konsistensi antara hasil pelatihan dan pengujian. Temuan ini membuktikan bahwa kombinasi antara *Logistic Regression* dengan strategi pembagian data yang tepat dapat menghasilkan model klasifikasi yang akurat, seimbang, dan dapat diandalkan.

Selain memberikan kontribusi teknis yang signifikan, penelitian ini juga mengintegrasikan nilai-nilai Islam sebagai landasan etis. Prinsip membedakan antara kebaikan dan keburukan, seperti tercermin dalam Surah Fussilat ayat 34 dan

Surah Al-Maidah ayat 100. Dengan demikian, model klasifikasi yang dikembangkan tidak hanya unggul secara empiris dan ilmiah, tetapi juga selaras dengan nilai-nilai keagamaan, menjadikannya solusi keamanan jaringan yang komprehensif dan berkelanjutan.

5.2 Saran

Berdasarkan hasil penelitian yang telah dilakukan, terdapat beberapa saran yang dapat dijadikan acuan untuk penelitian selanjutnya maupun pengembangan sistem deteksi *Malware* berbasis klasifikasi trafik jaringan:

1. Pemanfaatan metode pembagian data secara manual (*manual split*) dibandingkan dengan metode acak (*random split*) menunjukkan adanya perbedaan performa. Hasil optimal diperoleh pada skenario 2 dan 3 yang menerapkan random split, karena mampu menghindari *overfitting* pada model. Meski demikian, eksplorasi terhadap kombinasi teknik lainnya tetap perlu dilakukan guna meningkatkan akurasi dan generalisasi model.
2. Penelitian ini menggunakan algoritma *Logistic Regression* sebagai metode klasifikasi utama. Untuk penelitian selanjutnya, disarankan untuk mengevaluasi performa algoritma lain seperti *Naïve Bayes, Support Vector Machine (SVM),* atau metode klasifikasi lainnya, dengan harapan dapat memperoleh hasil yang lebih akurat dan andal dalam menklasifikasikan malware pada trafik jaringan.

DAFTAR PUSTAKA

Alharbi, A., Hamid, Md. A., & Lahza, H. (2022). Predicting Malicious Software in IoT Environment Based on Machine Learning and Data Mining Techniques. International Journal of Advanced Computer Science and Applications, 13(8). https://doi.org/10.14569/IJACSA.2022.0130857

Allorerung, P. P., Erna, A., Bagussahrir, M., & Alam, S. (2024). Analisis Performa NormalisasiData untuk Klasifikasi K-Nearest Neighbor pada Dataset Penyakit. JISKA (Jurnal Informatika Sunan Kalijaga), 9(3), 178–191. https://doi.org/10.14421/jiska.2024.9.3.178-191

Ardelia, D. N., Arifin, H. D., Daniswara, S., & Sari, A. P. (2024). Klasifikasi Harga Ponsel Menggunakan Algoritma *Logistic Regression*. 01.

Diana, D., Indrajit, R. E., & Dazki, E. (2022). Komparasi Algoritma Naïve Bayes, *Logistic Regression* Dan Support Vector Machine pada Klasifikasi File Application Package Kit Android Malware. Jutisi : Jurnal Ilmiah Teknik Informatika dan Sistem Informasi, 11(1), 109. https://doi.org/10.35889/jutisi.v11i1.815

Dola Ramalinda, Jayadi, & Agung Rachmat Raharja. (2024). Strategi Perlindungan Data Menggunakan Sistem Kriptografi Dalam Keamanan Informasi. Journal of International Multidisciplinary Research, 2(6), 665–671. https://doi.org/10.62504/jimr679

Farooq, M. S., Akram, Z., Alvi, A., & Omer, U. (2022). Role of *Logistic Regression* in Malware Detection: A Systematic Literature Review. VFAST Transactions on Software Engineering, 10(2), 36–46. https://doi.org/10.21015/vtse.v10i2.963

Firmansyah, B. (2024). Cybersecurity Fundamentals: In B. Gupta (Ed.), Advances in Computational Intelligence and Robotics (pp. 280–320). IGI Global. https://doi.org/10.4018/979-8-3693-3860-5.ch009

Flynn, R., & Olukoya, O. (2025). Using approximate matching and machine learning to uncover malicious activity in logs. Computers & Security, 151, 104312. https://doi.org/10.1016/j.cose.2025.104312

Garcia, S., Parmisano, A., & Erquiaga, M. J. (2020). IoT-23: A labeled dataset with malicious and benign IoT network traffic (Version 1.0.0) [Dataset]. Zenodo. https://doi.org/10.5281/ZENODO.4743746

Hadjar, I. (2018). REGRESI LOGISTIK: MENAKSIR PROBABILITAS PERISTIWA VARIABEL BINARI. Phenomenon : Jurnal Pendidikan MIPA, 7(2), 137–163. https://doi.org/10.21580/phen.2017.7.2.1385

Handoyo, S., Pradianti, N., Nugroho, W. H., & Akri, Y. J. (2022). A Heuristic Feature Selection in *Logistic Regression* Modeling with Newton Raphson and Gradient Descent Algorithm. International Journal of Advanced Computer Science and Applications, 13(3). https://doi.org/10.14569/IJACSA.2022.0130317

Le, M. T., & Nguyen, Q. A. (2024). Predicting learning status of informatics students at Dong Thap University based on machine learning models. Dong Thap University Journal of Science, 13(5), 45–52. https://doi.org/10.52714/dthu.13.5.2024.1287

Manoppo, V. A., Lumenta, A. S. M., & Karouw, S. D. S. (2020). Analisa Malware Menggunakan Metode Dynamic Analysis Pada Jaringan Universitas Sam Ratulangi.

M, M Abdul Ghoffar E., Abdurrahim Mu'thi, Abu Ihsan Al-Atsari. (2004). Tafsir Ibnu Katsir. Bogor: Pustaka Imam Asy-Syafi'i, 8 jilid.

Nafis, Z. M. J., Nazilla, R., Nugraha, R., & Shofwatul 'Uyun, S. 'Uyun. (2024). Perbandingan Algoritma Decision Tree dan K-Nearest Neighbor untuk Klasifikasi Serangan Jaringan IoT. Komputika : Jurnal Sistem Komputer, 13(2), 245–252. https://doi.org/10.34010/komputika.v13i2.12609

Pinontoan, P. Y., & Sembiring, I. (2024). IMPLEMENTASI DAN ANALISIS DETEKSI SERANGAN JARINGAN PADA WEB SERVER NFT MENGGUNAKAN SURICATA. 4.

Preetha, S., Lalasa, P., Department, department of ISE, B.M.S. College of Engineering, VTU, Bengaluru (Karnataka), India., R, P., & Department of ISE, B.M.S. College of Engineering, VTU, Bengaluru (Karnataka), India. (2021). A Comprehensive Overview on Cybersecurity Threats and Attacks. International Journal of Innovative Technology and Exploring Engineering, 10(8), 98–106. https://doi.org/10.35940/ijitee.H9242.0610821

Primadya, N. D., Nugraha, A., Luthfiarta, A., & Fahrezi, S. Y. (2024). Optimasi *Logistic Regression* untuk Deteksi Serangan DoS pada Keamanan IoT. Jurnal Eksplora Informatika, 13(2), 245–252. https://doi.org/10.30864/eksplora.v13i2.1065

Ramadhani, B., & Suryono, R. R. (2024). Komparasi Algoritma Naïve Bayes dan *Logistic Regression* Untuk Analisis Sentimen Metaverse. JURNAL MEDIA

INFORMATIKA BUDIDARMA, 8(2), 714. https://doi.org/10.30865/mib.v8i2.7458

Rani, N., & Dhavale, S. V. (2022). Leveraging Machine Learning for Ransomware Detection (No. arXiv:2206.01919). arXiv. https://doi.org/10.48550/arXiv.2206.01919

Rijal, M., Ilham, A. A., & Paundu, A. W. (2022). Evaluasi Algoritma Klasifikasi dengan Berbagai Metode *Seleksi Fitur* untuk Mendeteksi Aktivitas Trojan. Jurnal Pekommas, 7(2). https://doi.org/10.56873/jpkm.v7i2.4842

Sandriana, A., Rianto, R., & Maulana, F. (2022). Klasifikasi serangan Malware terhadap lalu lintas jaringan Internet of Things menggunakan Algoritma K-Nearest Neighbour (K-NN). E-JOINT (Electronica and Electrical Journal Of Innovation Technology), 3(1), 12–22. https://doi.org/10.35970/e-joint.v1i3.1336

Sutarman, Siringoringo, R., Arisandi, D., Kurniawan, E., & Nababan, E. B. (2024). Model Klasifikasi Dengan *Logistic Regression* Dan Recursive Feature Elimination Pada Data Tidak Seimbang. Jurnal Teknologi Informasi dan Ilmu Komputer, 11(4), 735–742. https://doi.org/10.25126/jtiik.1148198

Wijaya, A. P., & Santoso, H. (2021). Komparasi Performansi Algoritma Naive Bayes dan *Logistic Regression* pada Malware Android. 4.

Wijiyanto, W., Pradana, A. I., Sopingi, S., & Atina, V. (2024). Teknik K-Fold *Cross Validation* untuk Mengevaluasi Kinerja Mahasiswa. Jurnal Algoritma, 21(1). https://doi.org/10.33364/algoritma/v.21-1.1618

Zhang, Z. (2024). An Application of *Logistic Regression* in Bank Lending Prediction: A Machine Learning Perspective. Highlights in Business, Economics and Management, 45, 342–347. https://doi.org/10.54097/20qcm2

LAMPIRAN

Link Dataset Resmi :
stratosphereips.org/datasets-*IoT*23

Link Data Pemodelan :
https://bit.ly/DataPemodelanLR

Link Data Uji Coba 1 :
https://bit.ly/DataUjiCobaLR

Link Data Uji Coba 2 :
https://bit.ly/DataUjiCoba2LR

Link Kode Program Skenario 1:
https://colab.research.google.com/drive/1tYLQ7Nzh-vUwpUEJ_IehUkJHYaM6ey3o

Link Kode Program Skenario 2:
https://colab.research.google.com/drive/1YfHjE0KVVCMzh15mmaEx1s_RxPhtAR47

Link Kode Program Skenario 3:
https://colab.research.google.com/drive/1QpJ5UnV6QHynMC74pCMzs87SWlZr5yFz