DETEKSI *DISTRIBUTED DENIAL OF SERVICES* (DDOS) PADA *SOFTWARE DEFINED NETWORK* (SDN) MENGGUNAKAN *FUZZY* TSUKAMOTO

SKRIPSI

Oleh :

FAHMI IDRIS
NIM. 200605110166

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2024

DETEKSI *DISTRIBUTED DENIAL OF SERVICES* (DDOS) PADA *SOFTWARE DEFINED NETWORK* (SDN) MENGGUNAKAN *FUZZY* TSUKAMOTO

SKRIPSI

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

Oleh :
FAHMI IDRIS
NIM. 200605110166

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM MALANG
2024

HALAMAN PERSETUJUAN

DETEKSI *DISTRIBUTED DENIAL OF SERVICES* (DDOS) PADA *SOFTWARE DEFINED NETWORK* (SDN) MENGGUNAKAN *FUZZY* TSUKAMOTO

SKRIPSI

Oleh :
FAHMI IDRIS
NIM. 200605110166

Telah Diperiksa dan Disetujui untuk Diuji:
Tanggal: 05 Desember 2024

Pembimbing I,	Pembimbing II,
Johan Wahyu Ericka P., M.Kom	Dr. M. Amin Hariyadi, M.T
NIP. 19831213 201903 1 004	NIP. 19670118 200501 1 001

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

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

HALAMAN PENGESAHAN

DETEKSI *DISTRIBUTED DENIAL OF SERVICES* (DDOS) PADA *SOFTWARE DEFINED NETWORK* (SDN) MENGGUNAKAN *FUZZY* TSUKAMOTO

SKRIPSI

Oleh :
FAHMI IDRIS
NIM. 200605110166

Telah Dipertahankan di Depan Dewan Penguji Skripsi
dan Dinyatakan Diterima Sebagai Salah Satu Persyaratan
Untuk Memperoleh Gelar Sarjana Komputer (S.Kom)
Tanggal: 23 Desember 2024

Susunan Dewan Penguji

Ketua Penguji	: Dr. Zainal Abidin, M.Kom NIP. 19760613 200501 1 004	( )
Anggota Penguji I	: Shoffin Nahwa Utama, M.T NIP. 19860703 202012 1 003	( )
Anggota Penguji II	: Johan Wahyu Ericka P., M.Kom NIP. 19831213 201903 1 004	( )
Anggota Penguji III	: Dr. M. Amin Hariyadi, M.T NIP. 19670118 200501 1 001	( )

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

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

PERNYATAAN KEASLIAN TULISAN

Saya yang bertanda tangan di bawah ini:

Nama	: Fahmi Idris
NIM	: 200605110166
Fakultas / Program Studi	: Sains dan Teknologi / Teknik Informatika
Judul Skripsi	: Deteksi *Distributed Denial of Services* (DDoS) Pada *Software Defined Network* (SDN) Menggunakan *Fuzzy* Tsukamoto

Menyatakan dengan sebenarnya bahwa skripsi yang saya tulis 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 Desember 2024
Yang membuat pernyataan,

Fahmi Idris
NIM.200605110166

MOTTO

"You'll never be ready, just start"

HALAMAN PERSEMBAHAN

Dengan penuh rasa syukur kepada Allah SWT atas limpahan rahmat dan kemudahan-Nya, akhirnya skripsi ini dapat terselesaikan dengan baik.

Penulis persembahkan karya ini kepada ibu Rukaidah dan bapak Muhammad sebagai tanda hormat dan terima kasih yang tidak terhingga atas doa dan dukungan yang senantiasa diberikan. Kepada segenap keluarga, dosen, dan teman-teman yang telah memberikan saran, arahan, dan dukungan selama perkuliahan.

KATA PENGANTAR

Assalamualaikum Warahmatullahi Wabarakatuh.

Alhamdulillah, segala puji dan syukur senantiasa penulis panjatkan pada Allah subhanahu wa ta'ala atas berkat rahmat, serta hidayah-Nya, sehingga penulis dapat menyelesaikan skripsi yang berjudul "Deteksi *Distributed Denial of Services* (DDoS) Pada *Software Defined Network* (SDN) Menggunakan *Fuzzy* Tsukamoto". Sholawat serta salam tetap tercurahkan kepada nabi Muhammad SAW. dan semoga kita semua mendapat syafaatnya di hari akhir kelak, amin.

Penulis mengucapkan rasa terima kasih yang begitu besar kepada seluruh pihak yang memberikan dukungan dan motivasi kepada penulis sehingga dapat menyelesaikan skripsi. Ucapan terima kasih ini penulis disampaikan kepada:

1. Prof. Dr. M. Zainuddin, M.A., selaku rektor Universitas Islam Negeri Maulana Malik Ibrahim Malang.
2. Prof. Dr. Hj. Sri Harini, M.Si., selaku dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.
3. Dr. Ir. Fachrul Kurniawan, M.MT, IPU., selaku ketua Program Studi Teknik Informatika Universitas Islam Negeri Maulana Malik Ibrahim Malang.
4. Johan Ericka Wahyu Prakasa, M. Kom, selaku dosen pembimbing I dan Dr. M. Amin Hariyadi, M.T, selaku dosen pembimbing II, yang telah memberikan saran, arahan, dan bimbingan dalam menyelesaikan skripsi.

5. Dr. Zainal Abidin, M.Kom, selaku dosen penguji I dan Shoffin Nahwa U., M.T, selaku dosen penguji II, yang telah memberikan saran dan arahan dalam menyelesaikan skripsi.
6. Juniardi Nur Fadila, M.T, selaku dosen wali.
7. Segenap dosen, laboran, dan jajaran staf serta karyawan Program Studi Teknik Informatika.
8. Kedua orang tua, Ibu Rukaidah dan Ayah Muhammad yang senantiasa mendukung penulis dalam menyelesaikan skripsi.
9. Kakak Silatul Laila beserta seluruh keluarga besar yang telah memberikan dukungan, sehingga penulis mampu menyelesaikan skripsi.
10. Teman-teman INTEGER 2020 yang telah berjuang bersama.
11. Seluruh pihak yang telah terlibat secara langsung maupun tidak langsung dari awal perkuliahan hingga akhir penulisan skripsi.

Penulis menyadari bahwa dalam penyusunan skripsi ini masih jauh dari kata sempurna. Penulis dengan senang hati menerima saran, kritik, dan masukan yang bersifat membangun sehingga skripsi ini dapat lebih dikembangkan. Penulis berharap semoga skripsi ini dapat memberikan manfaat untuk kedepannya.

Wassalamu'alaikum warahmatullahi wabarakatuh.

Malang, 16 Desember 2024

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Idris, Fahmi. 2024. **Deteksi *Distributed Denial of Services* (DDoS) Pada *Software Defined Network* (SDN) Menggunakan *Fuzzy* Tsukamoto.** Skripsi. Program Studi Teknik Informatika, Fakultas Sains dan Teknologi, Universitas Islam Negeri Maulana Malik Ibrahim Malang. Pembimbing: (I) Johan Ericka Wahyu P., M.Kom. (II) Dr. M. Amin Hariyadi, M.T.

Kata Kunci: DDoS, *Fuzzy* Tsukamoto, *Software Defined Network*

Perkembangan serangan siber semakin bervariasi seiring dengan perkembangan teknologi, salah satunya serangan terhadap jaringan komputer. Serangan yang umum ditemukan yaitu serangan DDoS. Serangan DDoS merupakan ancaman serius bagi ketersediaan layanan jaringan. Tidak hanya menimbulkan kerugian secara materi, tetapi juga non-materi bagi penyedia layanan. *Software Defined Network* (SDN) sebagai jaringan dengan arsitektur yang menyediakan manajemen jaringan inovatif masih rentan terhadap adanya serangan. Pencegahan DDoS menggunakan *firewall* dan IDS atau IPS yang masih tetap rentan terhadap kesalahan membutuhkan adanya pembaharuan metode deteksi untuk mengurangi kesalahan pada saat proses identifikasi. Penelitian ini mengusulkan sebuah sistem deteksi berbasis logika *fuzzy* Tsukamoto untuk mendeteksi serangan DDoS pada *Software Defined Network* (SDN). *Fuzzy* Tsukamoto dapat memberikan fleksibilitas dan toleransi terhadap ketidaktepatan data, sehingga memungkinkan untuk mendeteksi adanya serangan DDoS dengan mengenali pola *traffic* jahat dan normal pada suatu jaringan. Data yang digunakan adalah dataset "*DDoS Attack SDN*". Variabel-variabel yang digunakan meliputi durasi, *flow*, *packet-in*, jumlah paket, jumlah *byte*, *packet rate*, dan *port bandwidth*. Deteksi DDoS SDN dalam penelitian ini terdiri dari beberapa tahapan, yaitu pengumpulan data, penentuan semesta dan domain himpunan setiap variabel *fuzzy*, pembentukan aturan *fuzzy*, fuzzifikasi, inferensi, defuzzifikasi, keputusan hasil deteksi, dan evaluasi. Hasil uji coba menunjukkan bahwa model yang diusulkan mampu mendeteksi serangan DDoS dengan akurasi minimal sebesar 75,41%, maksimal sebesar 77,84%, dan rata-rata akurasi sebesar 76,60%.

ABSTRACT

Idris, Fahmi. 2024. **Detection of Distributed Denial of Services (DDoS) on Software Defined Network (SDN) Using Tsukamoto Fuzzy Model.** Thesis. Department of Computer Engineering, Faculty of Science and Technology, Maulana Malik Ibrahim State Islamic University of Malang. Supervisor: (I) Johan Ericka Wahyu P., M.Kom. (II) Dr. M. Amin Hariyadi, M.T.

Keywords : DDoS, Tsukamoto Fuzzy Model, Software Defined Network

The development of cyber attack is becoming increasingly diverse along with technological advancement, one of which is attack on computer network. A commonly encountered attack is the DDoS attack. DDoS attack pose a serious threat to the availability of network services. They not only cause material losses but also non-material losses for service providers. Software Defined Network (SDN) as a network with an architecture that provides innovative network management is still vulnerable to attacks. Preventing DDoS attacks using firewalls and IDS or IPS remains susceptible to errors, it needs the renewal of detection methods to reduce mistakes during the identification process. This research proposes a detection system based on Tsukamoto fuzzy logic to detect DDoS attacks on Software Defined Networks (SDN). Tsukamoto fuzzy logic can provide flexibility and tolerance to data inaccuracies, allowing for the detection of DDoS attacks by recognizing malicious and normal traffic patterns within a network. The data used is the "DDoS Attack SDN" dataset. The variables employed include duration, flow, packet-in, number of packets, number of bytes, packet rate, and port bandwidth. DDoS detection in SDN in this study consists of several stages, namely data collection, determining the universe and domain of the fuzzy set for each variable, forming fuzzy rules, fuzzification, inference, defuzzification, decision-making for detection results, and evaluation. The experimental result shows that the proposed model is capable of detecting DDoS attack with a minimum accuracy of 75.41%, a maximum accuracy of 77.84%, and an average accuracy of 76.60%.

البحث مستخلص

إدريس ، فهمي.٢٠٢٤.الكشف عن رفض الخدمات الموزع (DDoS) على الشبكة المعرفة بالبرمجيات (SDN) باستخدام تسوكاموتو الضبابي.اطروحه. برنامج دراسة هندسة المعلوماتية ،كلية العلوم والتكنولوجيا ، مولانا مالك إبراهيم جامعة الدولة الإسلامية ملانج. المشرف: (١) يوهان إريكا واهيو ب. الماجستير. (٢) دكتور. م. أمين حريادي، الماجستير

الكلمات الرئيسية: DDoS، غامض تسوكاموتو، الشبكة المعرفة بالبرمجيات

يتنوع تطور الهجمات السيبرانية بشكل متزايد جنبا إلى جنب مع التطورات التكنولوجية ، أحدها الهجمات على شبكات الكمبيوتر. الهجوم الشائع هو هجمات DDoS. تشكل هجمات DDoS تهديدا خطيرا لتوافر خدمات الشبكة. لا يتسبب ذلك في خسائر مادية فحسب ، بل يتسبب أيضا في خسائر غير مادية لمقدمي الخدمات. لا تزال الشبكة المعرفة بالبرامج (SDN) كشبكة ذات بنية توفر إدارة مبتكرة للشبكة عرضة للهجمات. تتطلب الوقاية من DDoS باستخدام جدران الحماية و IDS أو IPS التي لا تزال عرضة للأخطاء تحديث طرق الكشف لتقليل الأخطاء أثناء عملية تحديد الهوية. تقترح هذه الدراسة نظام كشف يعتمد على منطق تسوكاموتو الغامض للكشف عن هجمات DDoS على الشبكات المعرفة بالبرمجيات (SDN). يمكن أن يوفر Fuzzy Tsukamoto المرونة والتسامح مع عدم دقة البيانات ، مما يجعل من الممكن اكتشاف هجمات DDoS من خلال التعرف على أنماط حركة المرور الضارة والعادية على الشبكة. البيانات المستخدمة هي مجموعة بيانات "DDoS Attack SDN". تتضمن المتغيرات المستخدمة المدة والتدفق وإدخال الحزمة وعدد الحزم وعدد البايتات ومعدل الحزمة وعرض النطاق الترددي للمنفذ. يتكون اكتشاف SDN DDoS في هذه الدراسة من عدة مراحل ، وهي جمع البيانات ، وتحديد الكون ومجال مجموعة كل متغير ضبابي ، وتشكيل قواعد غامضة ، والتشويش ، والاستدلال ، وإزالة التشويش ، واتخاذ قرار بشأن نتائج الكشف ، والتقييم. تظهر نتائج الاختبار أن النموذج المقترح قادر على اكتشاف هجمات DDoS بدقة لا تقل عن 75.41٪ ، وبحد أقصى 77.84٪ ، ومتوسط دقة 76.60٪.

BAB I

PENDAHULUAN

1.1 Latar Belakang

Kebutuhan teknologi jaringan komputer dan perangkatnya meningkat dengan pesat di berbagai industri. Hal tersebut menyebabkan adanya kebutuhan akan kinerja jaringan yang lebih baik, penambahan konfigurasi jaringan menjadi lebih besar dan kompleks, dan pengendalian jaringan menjadi semakin rumit, sehingga pengelolaan jaringan menjadi sulit dan tidak fleksibel (Ahmed dkk., 2018).

Software Defined Network (SDN) merupakan sebuah paradigma jaringan yang mengubah arsitektur jaringan secara drastis. Jaringan SDN memungkinkan administrator dapat menyediakan jaringan dengan lebih cepat tanpa adanya konfigurasi secara manual. Pemisahan *control plane* dan *data plane* membuat administrator jaringan tidak perlu mengkonfigurasi perangkat jaringan satu per satu (Kandoi & Antikainen, 2015). Administrator mendapatkan kendali independen atas seluruh jaringan pada satu titik, sehingga dapat menyederhanakan operasi suatu jaringan.

Segala kemudahan yang disediakan dalam SDN tidak membuat jaringan jenis ini lepas dari ancaman keamanan. Semakin tinggi lalu lintas dalam suatu jaringan, semakin tinggi juga kemungkinan adanya ancaman keamanan. Salah satu serangan yang paling umum ditemukan adalah *Distributed Denial of Service* (DDoS).

DDoS umumnya dilakukan pada *layer* 3 (*application layer*) dan *layer* 4 (*transportation layer*). DDoS adalah jenis serangan yang memanfaatkan banyak IP untuk melakukan serangan pada jaringan dengan mengirimkan paket *request* yang membuat kemacetan lalu lintas pada suatu jaringan, sehingga kinerja perangkat menjadi lebih berat dari biasanya dan dapat mengakibatkan kerusakan pada perangkat jaringan (Bakker, 2017).

Dilansir dari stormwall.network jumlah total serangan DDoS di seluruh dunia pada tahun 2023 meningkat sebesar 63% dibandingkan pada tahun 2022. Faktor geopolitik menjadi pendorong utama peningkatan serangan DDoS. Hal tersebut semakin terlihat jelas pada kuartal ke-empat tahun 2023 sejak dimulainya kembali konflik antara Israel dan Palestina pada bulan Oktober. Dari aspek industri serangan DDoS berdampak pada beberapa sektor diantaranya; sektor keuangan sebesar 26% yang menjadi sektor yang paling banyak menjadi sasaran serangan DDoS, layanan pemerintah sebesar 21%, dan ritel sebesar 14%.

Umumnya pencegahan DDoS menggunakan dua langkah dasar, yaitu identifikasi dan *filtering* (Harto & Basuki, 2021). Proses identifikasi dilakukan secara manual atau menggunakan IDS (*Intrusion Detection System*) atau IPS (*Intrusion Prevention System*), sedangkan proses *filtering* menggunakan *firewall*. Serangan yang dapat dideteksi menggunakan *firewall* hanya 40% (Al-Shaer dkk., 2009). Dalam praktiknya *firewall* biasanya disandingkan dengan IDS atau IPS, namun penggunaan IDS atau IPS masih tetap rentan terhadap kesalahan. Pembaharuan secara berkala dibutuhkan untuk mengurangi kesalahan pada saat proses identifikasi (Hussain, 2018).

Oleh karena itu, diperlukan pengembangan metode yang kuat untuk menganalisis dan mengklasifikasikan serangan DDoS. Penelitian ini mengusulkan pendekatan deteksi DDoS menggunakan metode *Fuzzy* Tsukamoto. Metode ini memungkinkan untuk pemodelan fungsi kompleks dan non-linier (Brikh dkk., 2023), memberikan fleksibilitas, dan toleransi terhadap ketidaktepatan data (Bonato dkk., 2015). Penggunaan *fuzzy* tsukamoto memungkinkan untuk mendeteksi adanya serangan DDoS dengan mengenali pola *traffic* jahat dan normal pada suatu jaringan.

Islam melarang memasuki ranah milik orang lain tanpa sepengetahuan atau izin dari pemilik tersebut, salah satunya terdapat pada Al-Quran surah An-Nur ayat 27.

يٰٓاَيُّهَا الَّذِيْنَ اٰمَنُوْا لَا تَدْخُلُوْا بُيُوْتًا غَيْرَ بُيُوْتِكُمْ حَتّٰى تَسْتَأْنِسُوْا وَتُسَلِّمُوْا عَلٰٓى اَهْلِهَاۗ ذٰلِكُمْ خَيْرٌ لَّكُمْ لَعَلَّكُمْ تَذَكَّرُوْنَ

"Wahai orang-orang yang beriman, janganlah memasuki rumah yang bukan rumahmu sebelum meminta izin dan memberi salam kepada penghuninya. Demikian itu lebih baik bagimu agar kamu mengambil pelajaran." (QS. An-Nur [24]:27).

Berdasarkan tafsir Kemenag, ayat tersebut menjelaskan mengenai pentingnya adab dalam bersosialisasi. Penting untuk meminta izin sebelum memasuki rumah orang lain dan tidak lupa mengucapkan salam. Tindakan tersebut bukan hanya sekedar norma sosial, tetapi juga sebagai bentuk penghormatan terhadap privasi orang lain. Dengan demikian ayat ini mengajarkan untuk senantiasa membangun dan menjaga hubungan yang harmonis dalam bermasyarakat.

Tidak dibenarkan masuk ke dalam sistem atau komputer yang telah dilindungi dan dijaga dengan sistem keamanannya masing-masing karena si pemilik tidak menginginkan sembarang orang mengakses atau bahkan digunakan untuk tindakan kejahatan. Pelaku DDoS menggunakan komputer orang lain yang telah diinfeksi dengan *malware* (*botnet*) sebagai sumber dalam membanjiri server tujuan. Komputer-komputer tersebut digunakan untuk melakukan tindak kejahatan tanpa disadari oleh pemiliknya.

Dalam mendeteksi DDoS pada *Software Defined Network* (SDN), penulis menggunakan metode *fuzzy* Tsukamoto. Sistem akan mengidentifikasi *traffic* jahat (DDoS) dan normal berdasarkan pola serangan dan bukan serangan melalui aturan-aturan *fuzzy* yang telah ditentukan. Dengan penelitian ini, diharapkan dapat memudahkan proses identifikasi serangan DDoS pada lingkungan SDN, sehingga dapat segera dilakukan pencegahan.

1.2 Identifikasi Masalah

Berdasarkan latar belakang masalah yang dipaparkan, maka rumusan masalah pada penelitian ini adalah bagaimana cara mengidentifikasi *traffic* serangan DDoS berdasarkan *traffic Software Defined Network* (SDN) menggunakan algoritma *fuzzy* tsukamoto?

1.3 Tujuan Penelitian

Tujuan penelitian ini adalah mengidentifikasi serangan DDoS pada *traffic Software Defined Network* (SDN) menggunakan *fuzzy* Tsukamoto.

1.4 Batasan Masalah

Adapun batasan masalah dalam penelitian ini, yaitu:

1. Menggunakan dataset DDoS SDN dari penelitian Nisha Ahuja, Gaurav Singal, dan Debajyoti Mukhopadhyay dengan judul "*Automated DDoS Attack Detection in Software Defined Networking*" yang dipublikasikan di situs web data.mendeley.com pada 27 september 2020.
2. Menggunakan variabel durasi, *flow*, *packet-in*, jumlah paket, jumlah *byte*, *packet rate*, dan *port bandwidth*.
3. Menggunakan *rule base* berdasarkan data yang digunakan.
4. Menggunakan representasi fungsi keanggotaan bentuk bahu dan segitiga.
5. Menggunakan data yang sudah ada sebagai data uji.
6. Menggunakan bahasa pemrograman python.

1.5 Manfaat Penelitian

Peneliti berharap penelitian ini dapat menjadi referensi dalam penelitian-penelitian selanjutnya yang berkaitan dengan algoritma kecerdasan buatan dalam mendeteksi serangan DDoS pada *Software Defined Network* (SDN).

BAB II

STUDI PUSTAKA

2.1 Penelitian Terdahulu

Sugianti dkk. (2020) menggunakan metode *fuzzy* sugeno untuk mendeteksi serangan DDoS berbasis HTTP. Penelitian ini bertujuan untuk membangun aplikasi yang mampu mendeteksi serangan DDoS sejak dini. Variabel input yang digunakan terdiri dari jumlah *user*, jumlah paket, jumlah panjang/*user*, dan panjang paket. Metode *fuzzy* sugeno digunakan untuk mengolah data dan menghasilkan tingkat akurasi dalam mendeteksi serangan DDoS. Hasil pengujian menunjukkan bahwa aplikasi yang dibangun dapat mendeteksi serangan DDoS berbasis HTTP dengan tingkat keakuratan mencapai 90%.

Tantriawan dkk. (2021) menerapkan logika *fuzzy* sugeno dalam mendeteksi serangan *Distributed Denial of Service* (DDoS). Data lalu lintas jaringan diambil menggunakan *Wireshark*. Variabel input yang digunakan terdiri dari jumlah *user*, panjang paket, *rate*, dan jumlah paket. Aturan *fuzzy* dibuat berdasarkan data yang dikumpulkan. Hasil pengujian menunjukkan bahwa metode pendeteksian menggunakan *fuzzy* sugeno dapat mengidentifikasi serangan DDoS dengan tingkat akurasi 70%.

Harto dan Basuki (2021) menggunakan algoritma *random forest* untuk mengklasifikasikan serangan DDoS pada *Software Defined Networks* (SDN). *Random forest* dipilih karena implementasinya yang sederhana dan kebutuhan sumber daya yang rendah. Data yang digunakan yaitu dataset Universitas

Muhammadiyah yang diambil oleh Oxicusa Gugi Housman pada tahun 2020 untuk melatih model *random forest*. Sistem ini dirancang untuk mencatat lalu lintas jaringan dan mengujinya berdasarkan model yang dilatih. Jumlah pohon keputusan optimal yang dihasilkan pada *random forest* adalah 15. Rata-rata waktu deteksinya juga tidak terlalu lama, yaitu 281 milidetik. Pengujian sistem ini mencapai akurasi rata-rata sebesar 92,8%.

De dkk. (2021) mendeteksi serangan *Reduction-of-Quality* (RoQ) DDoS menggunakan logika *fuzzy* dan algoritma machine learning. Peneliti mengevaluasi dan membandingkan empat algoritma machine learning, yaitu *multi-layer perceptron* dengan *backpropagation, k-nearest neighbors, support vector machine, multinomial naive bayes*. Peneliti juga menggunakan pendekatan *fuzzy logic* dan *euclidean distance*. Evaluasi dilakukan dengan membandingkan pendekatan tersebut menggunakan jejak *traffic* tiruan dan nyata. Hasil penelitian menunjukkan bahwa MLP memberikan hasil klasifikasi terbaik. MLP mencapai skor F1 sebesar 98,04% untuk *traffic* simulasi dan 99,30% untuk *traffic* normal.

Scaranti dkk. (2020) mengembangkan sistem deteksi anomali jaringan yang memanfaatkan *artificial immune system* dan logika *fuzzy*. Sistem ini mampu mendeteksi variasi *event* jaringan. Logika *fuzzy* digunakan untuk mengatasi ketidakpastian batas antara *traffic* normal dan serangan. Evaluasi sistem pada penelitian menggunakan data hasil simulasi dan dataset CiCDDoS2019. Hasil penelitian menunjukkan bahwa sistem ini dapat mengungguli klasifikasi *naive bayes* dan KNN dalam hal matriks *F-measure*. *F-measure* sistem deteksi mencapai 99,97% untuk data simulasi dan 92,28% untuk dataset CiCDDoS2019.

Tabel 2. 1 Tabel perbandingan dengan penelitian terdahulu

No	Peneliti	Objek	Metode	Hasil	Perbandingan Penelitian	
					Terdahulu	Usulan Penelitian
1	(Sugianti dkk., 2020)	DDoS berbasis HTTP	*Fuzzy* Sugeno	Akurasi aplikasi deteksi DDoS berbasis HTTP mencapai 90%.	Menggunakan *fuzzy* sugeno dalam mendeteksi serangan DDoS. Data diambil dari situs web dengan wireshark. Parameter yang digunakan terdiri dari jumlah user, jumlah paket, jumlah panjang/user, dan panjang paket.	Menggunakan *fuzzy* tsukamoto dengan dataset DDoS SDN sebagai objek penelitian. Variabel yang digunakan juga terdapat perbedaan yaitu variabel durasi, *packet-in*, *flow*, jumlah *byte*, *packet rate*, dan *port bandwidth.*
2	(Tantriawan dkk., 2021)	*Distributed Denial of Service* (DDoS)	*Fuzzy* Sugeno	Akurasi sistem deteksi DDoS mencapai 70%.	Menggunakan *fuzzy* sugeno dalam mendeteksi serangan DDoS. Data lalu lintas jaringan diambil menggunakan Wireshark. Parameter yang digunakan terdiri dari dari jumlah user, panjang paket, *rate*, dan jumlah paket.	Menggunakan *fuzzy* tsukamoto denga dataset DDoS SDN sebagai objek penelitian. Pada variabel penelitian juga terdapat perbedaaan yaitu variabel durasi, *packet-in*, *flow*, jumlah *byte*, dan *port bandwidth*.
3	(Harto & Basuki, 2021)	DDoS pada SDN	*Random Forest*	Rata-rata waktu deteksi sistem selama 281 milidetik dan akurasi deteksi rata-rata mencapai 92,8%.	Menggunakan *random forest* dalam mendeteksi serangan DDoS. Data yang digunakan adalah dataset Universitas Muhammadiyah yang diambil oleh Oxicusa Gugi Housman pada tahun 2020.	Menggunakan *fuzzy* tsukamoto sebagai sistem deteksi dengan dataset DDoS SDN sebagai objek penelitian.

4	(De dkk., 2021)	*Reduction-of-Quality* (RoQ) DDoS	*Multi-layer perceptron* dengan *backpropagation*, KNN, SVM, *multinomial naive bayes*, *fuzzy logic*, dan *euclidean distance*.	*Multi-layer perceptron*memberikan hasil klasifikasi terbaik dengan skor F1 sebesar 98,04% untuk *traffic* simulasi dan 99,30% untuk *traffic* normal.	*Multi-layer perceptron* dengan *backpropagation*, KNN, SVM, *multinomial naive bayes*, *fuzzy logic*, dan *euclidean distance* dalam mendeteksi RoQ DDoS. Data yang digunakan adalah data hasil simulasi, dataset CAIDA dan IFTO.	Menggunakan *fuzzy* tsukamoto sebagai algoritma sistem deteksi dengan dataset DDoS SDN sebagai objek penelitian.
5	(Scaranti dkk., 2020)	*Distributed Denial of Service* (DDoS)	*Artificial Immune Forest* dan *Fuzzy Logic*	F-meausre sistem deteksi mencapai 99,97%.	Menggunakan *artificial immune forest* dan *fuzzy logic* dalam mendeteksi serangan DDoS SDN. Data yang digunakan adalah data hasil simulasi dan dataset CiCDDoS2019.	Menggunakan *fuzzy* tsukamoto sebagai algoritma sistem deteksi dengan dataset DDoS *attack* SDN sebagai objek penelitian.

Perbedaan penelitian yang dilakukan oleh Sugianti dan kawan-kawan dengan penelitian ini terletak pada penggunaan metode, objek penelitian, data, dan beberapa variabel input. Penelitian sebelumnya menggunakan metode *fuzzy* sugeno dalam mendeteksi DDoS berbasis HTTP, dataset berasal dari simulasi dengan wireshark, dan menggunakan variabel jumlah *user*, jumlah paket, jumlah/panjang *user*, dan panjang paket, sedangkan penelitian ini menggunakan *fuzzy* tsukamoto dalam mendeteksi DDoS pada SDN, menggunakan dataset DDoS *attack* SDN, dan menggunakan variabel durasi, *packet-in*, *flow*, jumlah *byte*, *packet rate*, dan *port bandwidth*.

Perbedaan penelitian ini dengan penelitian Tantriawan dan kawan-kawan terletak pada penggunaan metode, data, dan beberapa variabel input. Penelitian sebelumnya menggunakan metode *fuzzy* sugeno dalam mendeteksi DDoS, menggunakan dataset yang diambil dengan wireshark, dan menggunakan variabel jumlah *user*, panjang paket, *rate*, dan jumlah paket, sedangkan penelitian ini menggunakan *fuzzy* tsukamoto dalam mendeteksi DDoS pada SDN, menggunakan dataset DDoS *attack* SDN, dan menggunakan variabel durasi, *packet-in*, *flow*, jumlah *byte*, dan *port bandwidth*.

Perbedaan penelitian Harto dan Basuki dengan penelitian ini terdapat pada penggunaan metode dan data penelitian. Penelitian sebelumnya menggunakan metode *random forest* dalam mendeteksi DDoS pada SDN, dan menggunakan dataset yang diambil oleh Gugi Housman, sedangkan penelitian ini menggunakan *fuzzy* tsukamoto dalam mendeteksi DDoS pada SDN, dan menggunakan dataset DDoS *attack* SDN yang diambil oleh Nisha Ahuja dan kawan-kawan.

Perbedaan penelitian De dan kawan-kawan dengan penelitian ini terdapat pada penggunaan metode, objek penelitian, dan data penelitian. Penelitian sebelumnya menggunakan metode *multi-layer perceptron* dengan *backpropagation*, KNN, SVM, *multinomial naive bayes*, *fuzzy logic*, dan *euclidean distance* dalam mendeteksi RoQ DDoS, dan menggunakan dataset hasil simulasi, dataset CAIDA dan IFTO, sedangkan penelitian ini menggunakan *fuzzy* tsukamoto dalam mendeteksi DDoS pada SDN, dan menggunakan dataset DDoS *attack* SDN.

Perbedaan penelitian Scaranti dan kawan-kawan dengan penelitian ini terdapat pada penggunaan metode dan data penelitian. Penelitian sebelumnya

menggunakan metode *artificial immune forest* dan *fuzzy logic* dalam mendeteksi serangan DDoS SDN, dan menggunakan dataset hasil simulasi dan dataset CiCDDoS2019, sedangkan penelitian ini menggunakan *fuzzy* tsukamoto dalam mendeteksi DDoS pada SDN, dan menggunakan dataset DDoS *attack* SDN.

2.2 Jaringan Komputer

Jaringan komputer adalah sekumpulan koneksi antara dua atau lebih komputer *autonomous* yang dihubungkan melalui kabel atau media transmisi nirkabel (Astuti, 2018). Disebut *autonomous*, apabila komputer tidak dapat mengendalikan komputer yang lain, seperti melakukan restart atau mematikan komputer yang lain. Dua atau lebih unit komputer dikatakan terhubung, jika antar komputer dapat bertukar data atau informasi.

Saat dua komputer atau lebih saling berkomunikasi atau bertukar data atau informasi, terdapat bagian jaringan komputer yang berperan menerima atau meminta layanan disebut *client* dan bagian yang berperan memberi atau mengirim disebut server. Model jaringan ini disebut sistem *client-server*. Berbeda dengan sistem *client-server*, sistem *peer-to-peer* adalah model jaringan komputer dimana komputer dapat berperan sebagai *client* dan server secara bersamaan.

Dalam sistem *client-server* terdapat layanan *file* server (penyimpanan file terpusat) dimana file atau data yang ada dalam server dapat dikelola dan dibagikan. Jaringan komputer juga memungkinkan penggunaan aplikasi dengan banyak pengguna (*multiuser*), serta memungkinkan untuk mendapatkan data terkini dengan lebih mudah dan cepat.

2.3 Protokol Jaringan

Dibutuhkan suatu mekanisme pengaturan bahasa yang dapat diketahui oleh dua buah sistem yang saling berkomunikasi (Mulyanta, 2005). Seperangkat aturan untuk mengatur proses komunikasi tersebut dinamakan protokol komunikasi data. Sebuah protokol menggambarkan format dan urutan pertukaran pesan antar dua atau lebih perangkat komunikasi serta mengatur perilaku pada saat transmisi dan penerimaan pesan. Protokol ini diterapkan berupa program komputer (*software*) dalam suatu komputer dan perangkat komunikasi data yang lain.

Badan International Organization for Standarization (IOS) membuat aturan baku yang dikenal dengan OSI (*Open System Interconnection*). Model OSI dibuat dengan tujuan mengatasi kendala *internetworking* yang disebabkan oleh perbedaan protokol dan arsitektur jaringan (Mardiana & Sahputra, 2017). Saat ini sudah banyak protokol-protokol jaringan yang diciptakan. Protokol-protokol yang banyak digunakan saat ini diantaranya, TCP/IP, *Ethernet*, dan lain-lain.

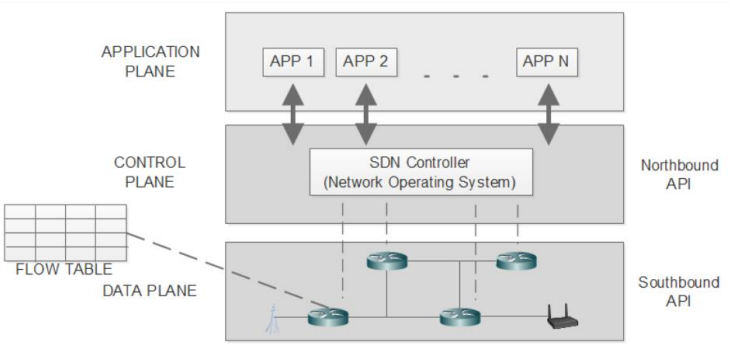
a. TCP/IP (*Transmission Control Protocol/Internet Protocol*): standar komunikasi data yang terdiri dari protokol-protokol dan digunakan oleh komunitas internet global.
b. UDP (*User Diagram Protocol*): Protokol pada lapisan transport TCP/IP yang menyokong komunikasi tidak andal (*unreliable*) dan tanpa koneksi (*connectionless*) antar host pada jaringan dengan protokol TCP/IP.
c. ICMP (*Internet Control Message Protocol*): Protokol yang dipakai oleh sistem operasi komputer pada jaringan dengan tujuan mengirim pesan kesalahan dalam jaringan.

2.4 *Software Defined Network* (SDN)

Software Defined Network (SDN) merupakan suatu paradigma baru dalam membangun jaringan komputer. SDN dapat menjadi solusi dari keterbatasan infrastruktur jaringan yang sudah ada. Dalam SDN, pusat kontrol (*control plane*) dan perangkat *forwarding* (*data plane*) disediakan secara terpisah. *Switch* hanya berperan sebagai *forwarding device* dan *logically centralized controller* sebagai pengontrol logika (Kreutz dkk., 2014).

Perbedaan tidak hanya terletak di sisi arsitektur, tetapi juga di sisi manajemen. Pada SDN, administrator jaringan dapat memprogram sendiri operasi suatu jaringan. *Controller* dapat dengan mudah diubah dan dimodifikasi programnya oleh admin melalui *middle box*. Pemrograman *controller* tidak terbatas pada *rules* dan *flow table*, tetapi juga untuk melakukan monitoring jaringan yang beroperasi (Alsmadi & Xu, 2015).

Arsitektur SDN diciptakan oleh *Open Networking Foundation* (ONF). Arsitektur SDN memiliki tiga lapisan utama yaitu; *Data Plane*, *Control Plane*, dan *Application Plane*. Setiap lapisan memiliki fungsi dan komponen spesifik masing-masing. Beberapa lapisan dalam arsitektur SDN yaitu; *Southbound API*, *SDN Controller* (*Network Operating System* (NOS)), *Northbound API*, dan aplikasi jaringan (Shaghaghi dkk., 2018).



Gambar 2. 1 Arsitektur *Software Defined Network* (SDN)

1. *Data Plane*

Pada *data plane* terdapat perangkat jaringan seperti *router* dan *switch* yang berfungsi untuk meneruskan paket. Perangkat pada lapisan ini hanya meneruskan paket secara sederhana tanpa adanya kemampuan yang lain. Melalui standar *interface* OpenFlow, perangkat pada lapisan ini dapat berkomunikasi dengan *controller*. *Interface* OpenFlow berfungsi untuk mengkonfigurasi, menyesuaikan (*compatibility*), dan interoperabilitas dalam komunikasi antar perangkat.

2. *Southbound* API

Southbound API merupakan salah satu komponen penting dalam arsitektur SDN. *Southbound* API berperan dalam menjembatani perangkat *forwarding* dengan *control plane*. *Controller* dapat mengendalikan perilaku jaringan dengan menjaga alur input perangkat-perangkat yang terhubung dengan *switch*. Komponen ini juga memungkinkan *router* untuk mengidentifikasi topologi jaringan, menentukan aliran jaringan, dan mengimplementasikan permintaan yang dikirim melalui *interface Southbound* API. *Southbound* API yang digunakan bisa berbeda-beda seperti OpenFLow, OpenState, dan OpFLex.

3. *Northbound* API

Berbeda dengan *Southbound* API, *Northbound* API memungkinkan komunikasi antar komponen *high-level*. *Interface northbound* menjadi penghubung antara aplikasi dan pengontrol SDN. Dengan adanya *northbound* aplikasi dapat memberi tahu jaringan mengenai sumber daya (data, penyimpanan, *bandwidth*, dan sebagainya) yang dibutuhkan dan jaringan dapat mengirimkan sumber daya tersebut atau mengkomunikasikan sumber daya yang dimiliki jaringan tersebut.

4. SDN *Controller*

SDN *controller* merupakan aplikasi dalam arsitektur SDN yang mengelola kontrol aliran jaringan. Komponen ini memungkinkan manajemen jaringan secara otomatis dan mempermudah integrasi dan pengelolaan aplikasi bisnis. SDN *controller* berjalan di server dan menggunakan protokol untuk memberi tahu *switch* tujuan pengiriman paket.

5. *Application Plane*

Application plane merupakan komponen paling atas yang menyediakan *interface* kepada pengguna dalam membuat program aplikasi. Komponen ini dapat mengkomunikasikan perilaku dan sumber daya yang dibutuhkan terhadap *controller*, sehingga penggunaan jaringan dapat optimal.

2.5 Mininet

Mininet merupakan emulator *Software Defined Network* (SDN) yang mampu melakukan simulasi jaringan pada suatu penelitian, riset, pengembangan, dan pengerjaan khususnya dalam bidang *Software Defined Network* (SDN) dan OpenFlow (Aprilianingsih dkk., 2017). Mininet dapat melakukan visualisasi

jaringan yang besar hanya dengan sumber daya yang terbatas. Mininet yang bersifat *open source* memungkinkan sebuah proyek dibangun oleh siapa saja yang dapat berupa *source code*, *script*, dan dokumentasi.

2.6 *Distributed Denial of Service* (DDoS)

Distributed Denial of Service (DDoS) merupakan serangan membanjiri sumber daya server dengan *request* untuk memenuhi kapasitas server. Hal tersebut dapat menyebabkan server tidak dapat menangani banyaknya *request* dan membuat server tidak bekerja dengan baik (Specht & Lee, 2003). Pada serangan DDoS, permintaan yang berlebih dikirimkan terhadap server atau jaringan dari banyak sumber yang berbeda-beda, sumber-sumber tersebut disebut dengan botnet. Botnet merupakan komputer-komputer yang sudah terinfeksi dengan malware dan komputer tersebut telah dikontrol tanpa sepengetahuan pengguna. Permintaan berlebih yang dikirimkan dari banyak sumber yang berbeda dapat mengakibatkan server atau jaringan tidak dapat diakses oleh pengguna yang sah (Hermawan, 2012).

Jenis-jenis serangan *Distributed Denial of Service* (DDoS), di antaranya:

1. TCP *SYN Flood*

TCP *SYN Flood* merupakan serangan dengan memanfaatkan kelemahan pada koneksi TCP (*Transmission Control Protocol*) yaitu *three way handshake*. Penyerang mengirimkan beberapa permintaan *SYN*, namun tidak menanggapi *SYN-ACK* dari server atau mengirimkan permintaan *SYN* menggunakan alamat IP yang palsu terhadap server. Server akan terus menunggu pengakuan untuk setiap *request* yang dikirim. Hal tersebut menyebabkan sumber daya pada server akan terbuang, sehingga koneksi baru tidak dapat dibuat dan penolakan layanan.

2. ICMP *Flood* (*Ping*)

ICMP *Flood* (*ping*) merupakan serangan dengan membanjiri server targer dengan paket *Echo Request* (*ping*). Server akan terus mencoba untuk merespon dengan paket ICMP *echo replay*, sehingga dapat menghabiskan *bandwitdh* masuk dan keluar. Hal ini dapat mengakibatkan sistem menjadi lambat secara signifikan.

3. UDP *Flood*

UDP *Flood* merupakan serangan membanjiri server target dengan paket *User Datagram Protocol* (UDP). Penyerang biasanya mengirimkan datagram UDP secara berurutan dengan IP palsu ke server dalam jaringan melalui berbagai *port* berbeda, sehingga memaksa server merespons dengan lalu lintas ICMP. Semakin banyak paket yang diterima dan direspon, maka server menjadi kebanjiran dan tidak dapat memproses permintaan yang sah.

4. HTTP *Flood*

Dalam HTTP *flood*, penyerang mengirimkan permintaan *GET* dan *POST* yang tampak sah pada suatu web server atau aplikasi. Serangan ini tidak menggunakan paket yang tidak valid, *spoofing*, ataupun teknik refleksi. Serangan ini hanya membutuhkan sedikit *bandwidth* untuk menumbangkan server tujuan. Hal tersebut dapat menyebabkan sumber daya server akan dialokasikan semaksimal mungkin untuk merespon banyaknya permintaan yang dikirimkan.

2.7 Logika *Fuzzy*

Fuzzy dapat diartikan sebagai sesuatu yang samar atau kabur, tidak jelas, dan membingungkan. Istilah sistem *fuzzy* tidak mengacu pada suatu sistem yang memiliki definisi, cara kerja, atau deskripsi yang tidak jelas (samar atau kabur),

namun sebuah sistem yang memiliki definisi, cara kerja, dan deskripsi jelas yang dibangun berdasarkan logika *fuzzy*. Logika *fuzzy* merupakan cara yang tepat dalam memetakan sebuah ruang input ke dalam sebuah ruang output (Fitriah & Maman, 2011).

Beberapa penyebab penggunaan logika *fuzzy*, diantaranya: (Kusumadewi & Purnomo, 2004)

1. Sederhana dan mudah dimengerti
2. Toleran terhadap data yang tidak tepat atau akurat
3. Fleksibel
4. Kemampuan dalam pemodelan fungsi non-linear yang cukup kompleks
5. Kemampuan dalam membangun dan mengimplementasikan pengalaman secara langsung dari para pakar
6. Kemampuan dalam berkombinasi dengan teknik kendali dengan model konvensional
7. Dibangun berdasarkan bahasa alami

Sistem *fuzzy* terdiri dari beberapa komponen, yaitu:

1. Variabel *Fuzzy* adalah variable-variabel yang akan digunakan dalam sistem *fuzzy*.
2. Himpunan *fuzzy* adalah kelompok yang mewakili suatu keadaan atau kondisi tertentu dalam sebuah variabel *fuzzy*. Contoh: himpunan dari variabel durasi: rendah dan tinggi. Terdapat dua atribut dalam himpunan *fuzzy*, yaitu:
    a. Himpunan linguistik merupakan penamaan suatu kelompok yang mewakili suatu keadaan atau kondisi tertentu dengan menggunakan bahasa alami.

b. Himpunan numeris merupakan sebuah nilai (angka) yang mewakili ukuran dari suatu variabel.

3. Semesta pembicara merupakan seluruh nilai yang akan dioperasikan dalam sebuah variabel *fuzzy*. Contoh: semesta pembicara variabel durasi: [0, 140].
4. Domain himpunan *fuzzy* adalah seluruh nilai dalam semesta pembicara yang dioperasikan dalam suatu himpunan *fuzzy*. Contoh: domain himpunan dari variabel durasi: rendah [0, 70], tinggi [70, 140].

2.8 Operasi Himpunan *Fuzzy*

Operasi himpunan *fuzzy* digunakan dalam proses penalaran atau inferensi. Operasi himpunan *fuzzy* menghasilkan derajat keanggotaan. Derajat keanggotaan merupakan hasil dari operasi dua buah himpunan *fuzzy* yang disebut dengan α-predikat atau *fire strength*. Menurut Kusumadewi dan Purnomo (2004), terdapat tiga operator dasar himpunan *fuzzy*, yaitu:

1. Operator *And*

Operator *and* berkaitan dengan operasi irisan terhadap himpunan (*intersection*). α-predikat yang dihasilkan pada operator ini merupakan nilai keanggotaan yang paling kecil (minimal) antar keanggotaan himpunan A dan himpunan B yang dituliskan dalam persamaan:

$$\mu A \cap B \; = \; min(\mu A[x], \mu B[y]) \qquad (2.1)$$

2. Operator *Or*

Operator *or* adalah operasi *union*. α-predikat yang dihasilkan pada operator ini merupakan nilai keanggotaan yang paling besar antar keanggotaan himpunan A dan himpunan B yang dituliskan dalam persamaan:

$$\mu AUB = min(\mu A[x], \mu B[y]) \quad (2.2)$$

3. Operator *Not*

Operator *not* adalah operasi *complement*. α-predikat yang dihasilkan pada operator ini merupakan selisih nilai keanggotaan terhadap nilai 1 yang dituliskan dalam persamaan:

$$\mu A = 1 - \mu A[x] \quad (2.3)$$

2.9 Fungsi Keanggotaan

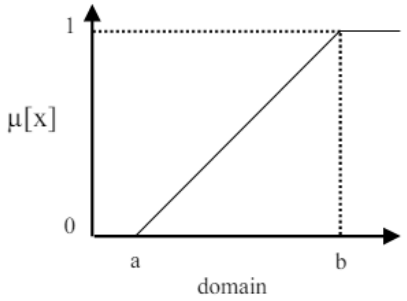
Sebuah kurva yang menginterpretasikan pemetaan titik-titik masukan data terhadap nilai keanggotaan dengan interval antara 0 hingga 1 (Kusumadewi & Purnomo, 2004). Pendekatan fungsi keanggotaan yang digunakan dapat digunakan untuk memperoleh nilai keanggotaan. Fungsi keanggotaan *fuzzy* yang umum digunakan meliputi:

a. Representasi Linear

Representasi linear menggambarkan pemetaan *input* terhadap derajat keanggotaan sebagai sebuah garis lurus. Representasi jenis ini merupakan bentuk representasi yang paling sederhana dan opsi yang bagus sebagai pendekatan suatu konsep yang samar atau kurang jelas. Terdapat dua kondisi himpunan *fuzzy linear*.

1. Linear Naik

Himpunan linear naik dimulai dari nilai domain dengan derajat keanggotaan bernilai 0 menuju nilai domain dengan derajat keanggotaan yang nilainya lebih tinggi. Himpunan linear naik dapat direpresentasikan seperti pada gambar 2.2.



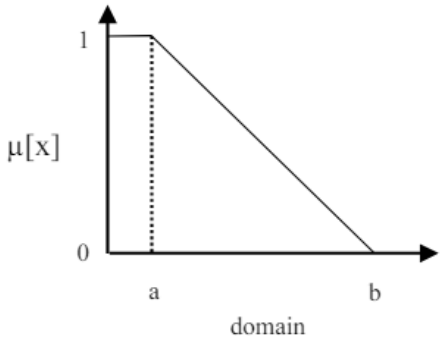
Gambar 2. 2 Linear Naik

Fungsi keanggotaan:

$$\mu(x) = \begin{cases} 0 & ;x \leq a \\ \dfrac{x-a}{b-a} & ;a \leq x \leq b \\ 1 & ;x \geq b \end{cases} \quad (2.4)$$

2. Linear Turun

Himpunan linear turun dimulai dari domain yang memiliki nilai paling tinggi bergerak turun terhadap nilai kodomain dengan derajat keanggotaan yang lebih rendah. Representasi himpunan linear turun ditunjukkan pada gambar 2.3.



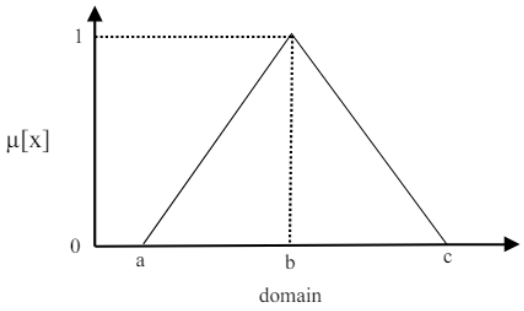
Gambar 2. 3 Linear Turun

Dengan:

$$\mu(x) = \begin{cases} 1 & ;x \leq a \\ \dfrac{b-x}{b-a} & ;a \leq x \leq b \\ 0 & ;x \geq b \end{cases} \tag{2.5}$$

b. Representasi Kurva Segitiga

Fungsi keanggotaan segitiga terdiri dari tiga parameter, yaitu $a, b, c \in R$ dengan $a < b < c$ yang representasikan dengan segitiga. Himpunan kurva segitiga dapat direpresentasikan seperti pada gambar 2.4.



Gambar 2. 4 Kurva Segitiga

Dengan:

$$\mu(x) = \begin{cases} 0 & ; x \leq a \; atau \; x \geq c \\ \dfrac{x-a}{b-a} & ; a \leq x \leq b \\ \dfrac{c-x}{c-b} & ; b \leq x \leq c \end{cases} \quad (2.6)$$

2.10 Implifikasi *Fuzzy*

Dalam aplikasinya, proposi *fuzzy* yang umum digunakan adalah implikasi *fuzzy*. Umumnya implikasi *fuzzy* berbentuk:

$$Jika \; x \; adalah \; A, maka \; y \; adalah \; B$$

x dan *y* merupakan variabel linguistik yang nilainya berada pada himpunan *X* dan *Y*, sedangkan *A* dan *B* merupakan predikat fuzzy yang berhubungan dengan himpunan fuzzy *A* dan *B* dalam semesta *X* dan *Y*. Proposisi yang diikutkan kata "jika" disebut anteseden. Proposisi yang diikutkan kata "maka" disebut konsekuen (Kusumadewi & Purnomo, 2004).

Umumnya terdapat dua fungsi implikasi yang digunakan, yaitu:

a. Min

Fungsi minimum merupakan pengambilan keputusan dengan mencari nilai minimum yang didasarkan pada aturan ke-*i*. Berikut persamaan fungsi minimum:

$$a_i = \mu A_i(x) \cap \mu B_i(x) = min\{\mu A_i(x), \mu B_i(x)\} \quad (2.7)$$

Dengan:
a_i = nilai minimum dari himpunan *A* dan *B* pada *rule* ke-*i*
$\mu A_i(x)$ = derajat keanggotaan *x* dari himpunan *A* pada rule ke-*i*
$\mu B_i(x)$ = derajat keanggotaan *x* dari himpunan *B* pada *rule* ke-i

b. Hasil Kali

Fungsi hasil kali merupakan pengambilan keputusan berdasarkan pada aturan ke-i. Fungsi ini menggunakan persamaan:

$$a_i \cdot \mu C_i(Z) \tag{2.8}$$

Keterangan:
a_i = nilai minimum dari himpunan *A* dan *B* pada *rule* ke-*i*
$\mu C_i(Z)$ = derajat keanggotaan konsekuen dari himpunan *C* pada *rule* ke-*i*

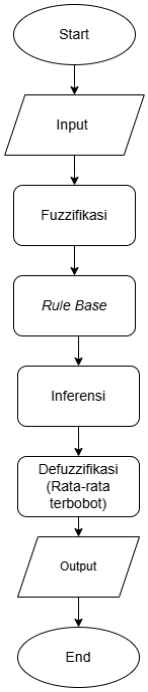
2.11 *Fuzzy Inference System* (FIS)

Proses menggabungkan aturan-aturan berdasarkan data yang digunakan. Sistem pakar menggunakan mesin inferensi dalam melakukan proses inferensi. Dalam menarik kesimpulan pada *IF-THEN rule*, terdapat dua pendekatan yaitu *forward chaining* dan *backward chaining* (Turban dkk., 2005). Sistem inferensi *fuzzy* digunakan untuk mengambil keputusan dari proses tertentu menggunakan aturan inferensi berdasarkan teori logika *fuzzy*.

2.12 *Fuzzy* Tsukamoto

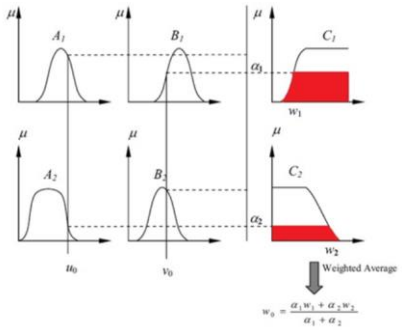
Metode tsukamoto merupakan salah satu metode yang dapat diimplementasikan dalam sistem inferensi *fuzzy*. Fungsi keanggotaan yang digunakan pada metode ini bersifat monoton. Setiap masukan dan himpunan fuzzy akan diproses melalui basis pengetahuan yang didalamnya terdapat aturan-aturan berbentuk *if-then*, kemudian dicari α-predikat dan nilai konsekuen (z) dari aturan-aturan tersebut. Dilanjutkan dengan proses defuzzifikasi dengan metode rata-rata terbobot (*Weighted Average*) untuk menghasilkan output yang berupa himpunan *crisp* (Kusumadewi & Purnomo, 2004).

Alur proses metode fuzzy Tsukamoto ditunjukkan pada gambar 2.5.



Gambar 2. 5 Flowchart Fuzzy Tsukamoto

Alur inferensi yang digunakan dalam memperoleh nilai *crisp* (tegas) ditunjukkan pada gambar 2.6.



Gambar 2. 6 Inferensi *Fuzzy* Tsukamoto (Maseleno & Hasan, 2015)

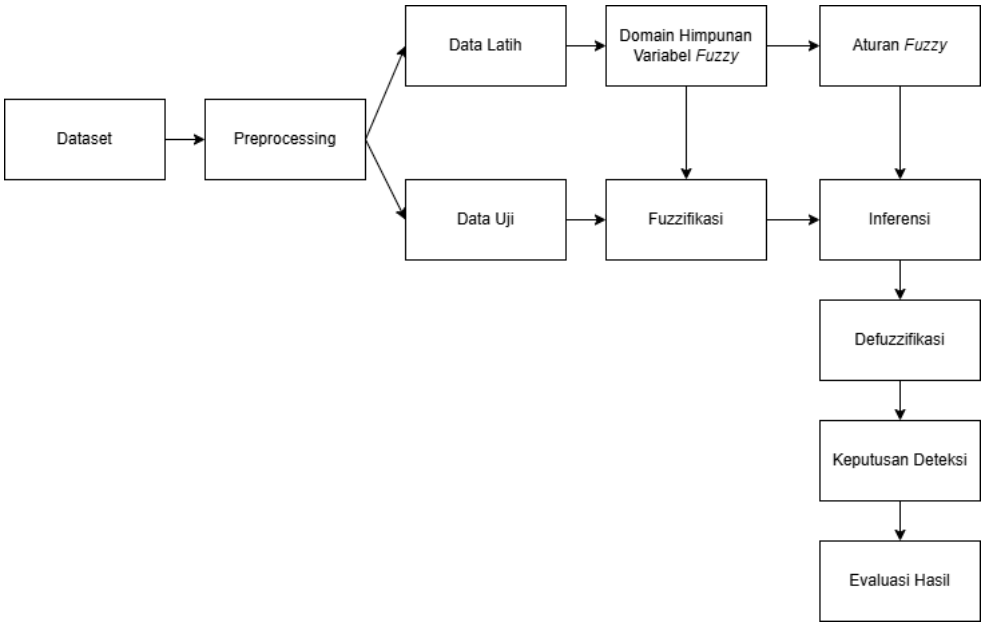
Nilai α-predikat dan z yang dihasilkan dari masing-masing aturan akan ditentukan hasil akhirnya melalui defuzzifikasi dengan menggunakan persamaan rata-rata terbobot.

BAB III

METODOLOGI PENELITIAN

3.1 Desain Sistem

Desain sistem adalah tahap yang akan diimplementasikan oleh peneliti dalam sistem yang akan dibangun. Sistem deteksi DDoS pada SDN terdiri dari beberapa tahapan, yaitu pengumpulan data, penentuan semesta dan domain himpunan setiap variabel *fuzzy*, pembentukan aturan *fuzzy*, fuzzifikasi, inferensi, defuzzifikasi, keputusan hasil deteksi, dan evaluasi. Desain sistem ditunjukkan pada gambar 3.1.



Gambar 3. 1 Desain Sistem

3.2 Pengambilan Data

Data diambil dari penelitian Nisha Ahuja, Gaurav Singal, dan Debajyoti Mukhopadhyay dengan judul "*Automated DDoS Attack Detection in Software Defined Networking*" yang dipublikasikan di situs web data.mendeley.com pada 27

september 2020. Dataset dihasilkan melalui simulasi mininet selama 200 menit menggunakan 10 topologi jaringan dengan *switch* yang terhubung ke satu Ryu *controller*. Data yang digunakan terdiri dari 33588 baris data dan 23 fitur yang mencakup hasil ekstrak dari *switch*, hasil perhitungan, dan label yang menunjukkan *traffic* normal atau jahat, tetapi hanya 7 fitur yang dipakai, yaitu durasi, *flow*, *packet-in*, jumlah paket, jumlah *byte*, *packet rate*, dan *port bandwidth*. Deskripsi fitur data ditunjukkan pada tabel 3.1.

Tabel 3. 1 Deskripsi data DDoS pada SDN

Fitur	**Penjelasan**
dt	*Timestamp* pengambilan data
switch	Id alur data (datapath) pada topologi jaringan
src	Alamat sumber
dst	Alamat tujuan
Pktcount	Paket yang ada dalam *flow*
bytecount	Byte yang dikirimkan dalam *flow*
duration_sec	Durasi flow dalam detik
duration_nsec	Durasi *flow* dalam nano detik
total_duration	Total durasi *flow* (jumlah *duration*_sec dan *duration*_nsec)
flow	Jumlah *flow* yang tercatat
packetins	Jumlah pesan paket yang diterima
pktperflow	Jumlah paket rata-rata per *flow*
byteperflow	Jumlah *byte* rata-rata per *flow*
pktrate	Jumlah paket yang dikirim tiap detik (jumlah paket dibagi interval monitoring)
Pairflow	Pairflow terkait
protocol	Protokol jaringan yang digunakan dalam mengirim paket
port_no	Nomor *port* dalam *flow*
tx_bytes	Jumlah *byte* yang dikirim di *switch port*
rx_bytes	Jumlah *byte* yang diterima di *switch port*
tx_kbps	Kecepatan transmisi data dalam kbps
rx_kbps	Kecepatan penerimaan data dalam kbps
tot_kbps	Jumlah dari tx_kbps dan rx_kbps (*port bandwith*)
Label	Label kelas *traffic* (normal atau jahat). *Traffic* normal memiliki label 0. *Traffic* jahat memiliki label 1.

Contoh data DDoS attack SDN ditunjukkan pada tabel 3.2.

Tabel 3. 2 Contoh data DDoS *attack* SDN

No.	src	dst	Pkt count	Byte count	Durasi	Durasi (*nano_second*)	Total durasi	Packet_in	Pkt perflow	Byte perflow	Rate	label
1	10.0.0.2	10.0.0.8	90333	96294978	200	744000000	201931000000	1943	13534	14427244	451	0
2	10.0.0.1	10.0.0.7	32914	330586324	73	2413306000	732413306000	1931	13385	14268410	446	0
3	10.0.0.1	10.0.0.8	45304	48294064	100	7113306000	101931000000	1943	13535	14428310	451	0
4	10.0.0.1	10.0.0.8	126395	134737070	280	734000000	281931000000	1943	13531	14424046	451	0
5	10.0.0.2	10.0.0.8	117399	125147334	260	719430000	261931000000	1943	13533	14426178	451	0
6	10.0.0.1	10.0.0.8	112864	120313024	200	732000000	251931000000	1943	13537	14430442	451	0
7	10.0.0.2	10.0.0.8	36282	38676612	80	7313306000	80700000000	1306	13536	14429376	451	0
8	10.0.0.1	10.0.0.8	85676	91330616	190	7213306000	191931000000	1943	13306	14184196	443	0
9	10.0.0.2	10.0.0.8	49807	53094262	113	739000000	111931000000	1790	13525	144176305	290	0
10	10.0.0.2	10.0.0.8	130706	139332596	290	754000000	291931000000	1943	13307	14185262	443	0
11	10.0.0.1	10.0.0.8	1330536	143948376	310	738000000	311931000000	1943	8641	9211306	288	1
12	10.0.0.1	10.0.0.8	4777	30592282	10	711931000	10700000000	1790	0	0	0	1
13	10.0.0.10	10.0.0.8	34232	35669744	113	689000000	111931000000	1943	9256	9644752	308	1
14	10.0.0.10	10.0.0.8	24976	26024992	80	6813306000	80700000000	1943	9242	9630164	308	1
15	10.0.0.10	10.0.0.8	6374	6641708	20	681931000	20700000000	1943	0	0	0	1
16	10.0.0.10	10.0.0.8	113012	114632004	380	759000000	381931000000	1943	6795	7080390	226	1
17	10.0.0.3	10.0.0.7	10454	11143964	23	101943000	23193100000	1086	0	0	0	1
18	10.0.0.10	10.0.0.8	15734	16394828	305	682000000	3057000000000	1943	9360	9753120	312	1
19	10.0.0.10	10.0.0.8	129991	135290622	470	773000000	471931000000	1943	6701	6982442	223	1
20	10.0.0.10	10.0.0.8	95620	99636040	320	709000000	321931000000	1943	8012	83483054	267	1

Berikut indikasi adanya serangan DDoS pada SDN dari variabel durasi, *flow*, *packet-in*, jumlah paket, jumlah *byte*, *packet rate*, dan *port bandwidth*:(Ahuja dkk., 2021)

- Jumlah paket: jumlah paket akan berkurang jika terjadi serangan dan meningkat apabila tidak terjadi serangan karena jumlah paket yang dikirim dalam serangan lebih sedikit. Penyerang hanya bertujuan membanjiri *flow table* dan tidak mengirimkan paket data.
- Jumlah *byte*: sama halnya dengan jumlah paket, jumlah *byte* juga menurun jika terjadi serangan.
- *Flow*: *flow* merupakan pengiriman paket antara host pengirim dan penerima dalam jaringan. Jika terjadi serangan, *switch* yang terhubung ke *host* target akan memiliki jumlah entri *flow* yang banyak.
- Durasi: *Traffic* berstatus DDoS memiliki durasi yang lebih lama dibandingkan dengan *traffic* normal. Durasi bernilai tinggi jika terjadi serangan dibandingkan dengan *traffic* normal.
- *Packet-in*: Pesan *packet_in* dihasilkan oleh host dan dikirim ke *controller* ketika paket yang diterima tidak memiliki deskripsi *flow* yang cocok di *flow table*. Ketika terjadi serangan, sejumlah besar pesan *packet_in* dilaporkan ke *controller*. Sebagai tanggapan, *controller* mengirim pesan *packet_out* (paket dikembalikan ke *switch* untuk dikeluarkan melalui *port* tertentu). Jumlah pesan *packet_in* meningkat jika terjadi serangan.

- *Packet rate*: Lalu lintas normal dan DDoS memiliki *rate* 290 paket per detik, tetapi dalam dataset ini terjadi penurunan *packet rate* pada *traffic* serangan karena penggunaan alamat IP palsu dalam melakukan serangan DDoS.
- *Port bandwidth*: Jumlah dari *byte* yang diterima dan *byte* yang dikirim. Penyerang yang mengirim lebih banyak permintaan dan lebih sedikit data menyebabkan *port bandwidth* menjadi lebih rendah, sehingga menunjukkan adanya serangan.

3.3 Pra-pemrosesan Data

Pra-pemrosesan data diawali dengan menghapus duplikasi nilai yang ada dalam dataset. Data dari variabel yang digunakan akan diambil dari dataset utama, kemudian dilakukan pembersihan outlier data menggunakan metode *Interquartile Range* (IQR). Data yang berada di luar batas nilai yang ditentukan akan dianggap sebagai outlier dan dihapus dari dataset. Dataset yang sudah dibersihkan nilai ouliernya tersebut akan dilakukan data sampling. Proses pembersihan outlier data ditunjukkan pada tabel 3.3.

Tabel 3. 3 Proses pembersihan outlier data

Algoritma 1: Identifikasi dan pembersihan outlier pada dataset
Input: dataset DDoS SDN **Output**: dataset DDoS SDN yang telah bersih dari nilai outlier
Inisialisasi: Q1, Q3, batas atas outlier,batas bawah outlier 1. Hitung kuartil pertama (Q1) dari data 2. Hitung kuartil ketiga (Q3) dari data 3. Hitung rentang interkuartil (IQR): IQR = Q3 - Q1 4. Hitung batas bawah outlier: batas_bawah = Q1 - 1.5 * IQR 5. Hitung batas atas outlier: batas_atas = Q3 + 1.5 * IQR 6. Buat dataset kosong untuk menyimpan data yang bersih (data_clean) 7. Untuk setiap baris dalam data cek setiap nilai dalam baris tersebut: 8. **if** nilai < batas_bawah atau nilai > batas_atas: 9. Lompat ke baris berikutnya (nilai ini adalah outlier) 10. **if** tidak ada nilai dalam baris yang merupakan outlier: 11. Tambahkan baris tersebut ke data_clean 12. **return** data_clean

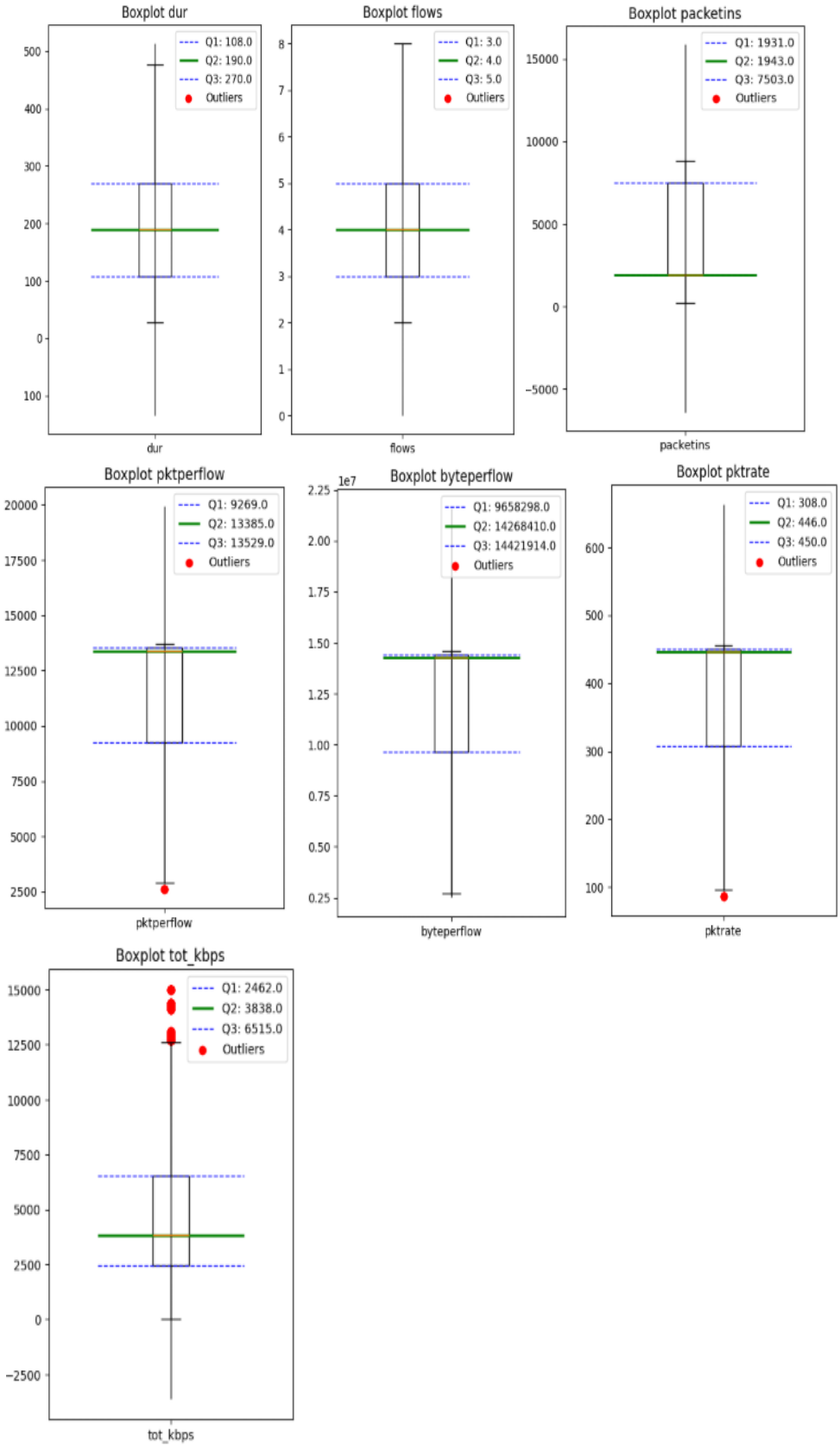
3.4 Penentuan Domain Himpunan Variabel *Fuzzy*

Pada data latih akan ditentukan nilai minimal, Q1, tengah (median), Q3, dan maksimal dari data masing-masing variabel masukan. Nilai-nilai tersebut yang dijadikan sebagai acuan dalam menentukan nilai domain himpunan *fuzzy* masing-masing variabel masukan. Nilai minimal sebagai batas bawah dan maksimal sebagai batas atas dari data latih masing-masing variabel masukan. Penentuan nilai domain himpunan dan semesta pembicara variabel *fuzzy* dari data latih ditunjukkan pada tabel 3.4.

Tabel 3. 4 Proses penentuan nilai domain himpunan dan semesta variabel *fuzzy*

Algoritma 2: Penentuan nilai minimal, Q1, tengah (median), Q3, dan maksimal
Input: data latih tiap variabel **Output**: nilai minimal, Q1, tengah (median), Q3, dan maksimal dari data tiap variabel
Inisialisasi: rentang, median 1. Untuk setiap kolom dalam data: 2. Temukan nilai minimal, Q1, tengah (median), Q3, dan maksimal dari data 3. Untuk setiap kolom dalam tabel rentang: 4. Buat daftar angka dari nilai terendah hingga nilai tertinggi dengan selisih 1. 5. Tambahkan daftar angka ke dalam variabel semesta pembicara.

Hasil dari penentuan nilai domain dan semeseta himpunan variabel dapat divisualisasikan dengan *boxplot*. Titik-titik merah menunjukkan nilai outlier, nilai minimum dari data direpresentasikan dengan garis horizontal paling bawah, garis biru putus-putus pertama dari bawah merepresentasikan nilai q1, nilai q2 atau median direpresentasikan dengan garis warna hijau, garis biru putus-putus kedua dari bawah merepresentasikan nilai q3, dan nilai maksimum direpresentasikan dengan garis horizontal hitam paling atas. *Boxplot* data ditunjukkan pada gambar 3.2.



Gambar 3. 2 *Boxplot* data

Berdasarkan gambar 3.2 dapat diketahui nilai minimal, q1, q2, q3, dan maksimal yang ditunjukkan pada tabel 3.5.

Tabel 3. 5 Nilai q1, q2, dan q3 dari data

Variabel	Q1	Q2	Q3
Durasi	108	190	270
Flow	3	4	5
Packet-in	1931	1943	7503
Jumlah paket	9269	13385	13529
Jumlah *byte*	9658298	14268410	14421914
Rate	308	446	450
Port bandwidth	2462	3838	6515

Berdasarkan gambar 3.2 dapat dibentuk domain himpunan *fuzzy* yang ditunjukkan pada tabel 3.6.

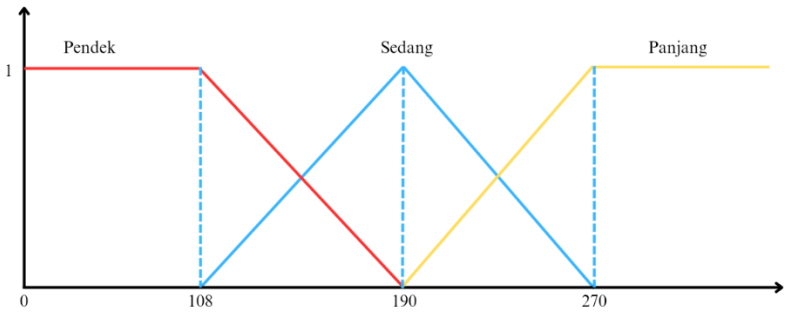
Tabel 3. 6 Domain himpunan *fuzzy*

Fungsi	Variabel	Domain Himpunan
Input	Durasi	pendek: [28, 190] sedang: [108, 270] panjang: [190, 477]
	Flow	sedikit: [2, 4] sedang: [3, 5] banyak: [4, 8]
	Packet-in	kecil: [219, 1943] sedang: [1931, 7503] besar: [1943, 8803]
	Jumlah paket	sedikit: [2604, 13385] sedang: [9269, 13529] banyak: [13385, 13685]
	Jumlah *byte*	sedikit: [2713368, 14268410] sedang: [9658298, 14421914] banyak: [14268410, 14588210]
	Rate	kecil: [86, 446] sedang: [308, 450] besar: [446, 456]
	Port bandwidth	rendah: [0, 3838] sedang: [2462, 6515] tinggi: [3838, 14993]
Output	Status	normal: [0, 0,6] DDoS: [0,4, 1]

Berikut representasi fungsi keanggotaan tujuh variabel masukan *fuzzy* berdasarkan penentuan domain dan semesta *fuzzy*:

a. Variabel Durasi

Variabel durasi didefinisikan dalam tiga himpunan yaitu pendek, sedang, dan panjang. Fungsi keanggotaan variabel durasi direpresentasikan dengan kurva bentuk bahu linear turun untuk himpunan pendek, kurva bentuk segitiga untuk himpunan sedang, dan kurva bentuk bahu linear naik untuk himpunan panjang. Fungsi keanggotaan variabel durasi ditunjukkan pada gambar 3.3.



Gambar 3. 3 Fungsi Keanggotaan Variabel Durasi

Fungsi keanggotaan dari variabel durasi pada gambar 3.3 dapat dituliskan dengan persamaan:

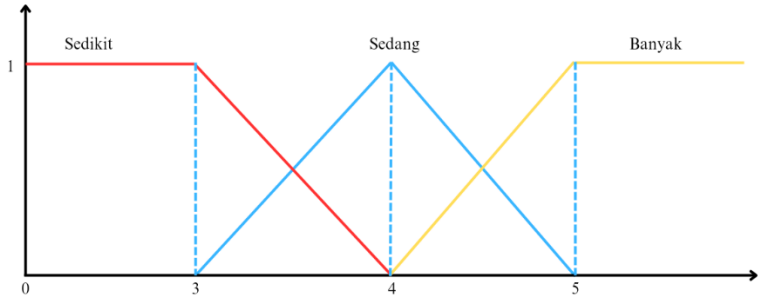
$$\mu_{\text{pendek}}(x) = \begin{cases} 1 & ; x \leq 108 \\ \dfrac{108 - x}{190 - 108} & ; 108 \leq x \leq 190 \\ 0 & ; x \geq 190 \end{cases} \quad (3.1)$$

$$\mu_{\text{sedang}}(x) = \begin{cases} 0 & ; x \leq 109 \text{ atau } x \geq 270 \\ \dfrac{x - 108}{190 - 108} & ; 108 \leq x \leq 190 \\ \dfrac{270 - x}{270 - 190} & ; 190 \leq x \leq 270 \end{cases} \quad (3.2)$$

$$\mu_{\text{panjang}}(x) = \begin{cases} 0 & ; x \leq 190 \\ \dfrac{x - 190}{270 - 190} & ; 190 \leq x \leq 270 \\ 1 & ; x \geq 270 \end{cases} \quad (3.3)$$

b. Variabel *Flow*

Variabel *flow* didefinisikan dalam tiga himpunan yaitu sedikit, sedang, dan banyak. Fungsi keanggotaan variabel *flow* direpresentasikan dengan kurva bentuk bahu linear turun untuk himpunan sedikit, kurva bentuk segitiga untuk himpunan sedang, dan kurva bentuk bahu linear naik untuk himpunan banyak. Fungsi keanggotaan variabel *flow* ditunjukkan pada gambar 3.4.



Gambar 3. 4 Fungsi Keanggotaan Variabel *Flow*

Fungsi keanggotaan dari variabel *flow* pada gambar 3.4 dapat dituliskan dengan persamaan:

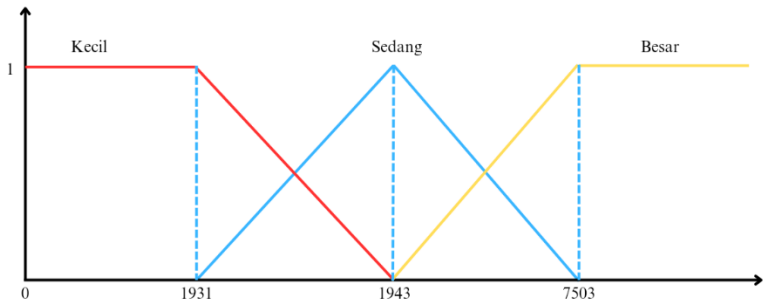
$$\mu_{\text{sedikit}}(x) = \begin{cases} 1 & ; x \leq 3 \\ \dfrac{3-x}{4-3} & ; 3 \leq x \leq 4 \\ 0 & ; x \geq 4 \end{cases} \tag{3.4}$$

$$\mu_{\text{sedang}}(x) = \begin{cases} 0 & ; x \leq 3 \text{ atau } x \geq 5 \\ \dfrac{x-3}{4-3} & ; 3 \leq x \leq 4 \\ \dfrac{5-x}{5-4} & ; 4 \leq x \leq 5 \end{cases} \tag{3.5}$$

$$\mu_{\text{banyak}}(x) = \begin{cases} 0 & ; x \leq 4 \\ \dfrac{x-4}{5-4} & ; 4 \leq x \leq 5 \\ 1 & ; x \geq 5 \end{cases} \tag{3.6}$$

c. Variabel *Packet-in*

Variabel *packet-in* didefinisikan dalam tiga himpunan yaitu kecil, sedang, dan besar. Fungsi keanggotaan variabel *packet-in* direpresentasikan dengan kurva bentuk bahu linear turun untuk himpunan kecil, kurva bentuk segitiga untuk himpunan sedang, dan kurva bentuk bahu linear naik untuk himpunan besar. Fungsi keanggotaan variabel *packet-in* ditunjukkan pada gambar 3.5.



Gambar 3. 5 Fungsi Keanggotaan Variabel *Packet-in*

Fungsi keanggotaan dari variabel *packet-in* pada gambar 3.5 dapat dituliskan dengan persamaan:

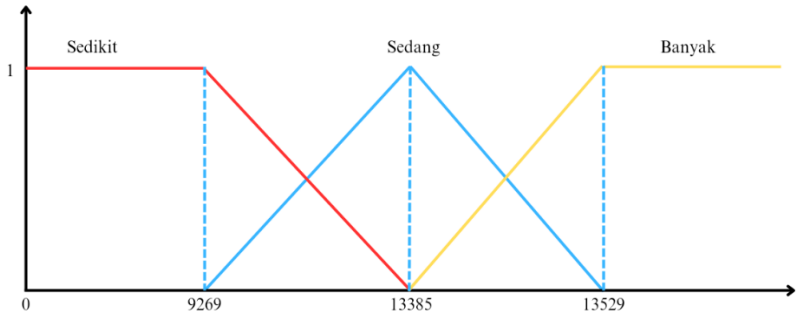
$$\mu_{\text{kecil}}(x) = \begin{cases} 1 & ; x \leq 1931 \\ \dfrac{1943 - x}{1943 - 1931} & ; 1931 \leq x \leq 1943 \\ 0 & ; x \geq 1943 \end{cases} \tag{3.7}$$

$$\mu_{\text{sedang}}(x) = \begin{cases} 0 & ; x \leq 1931 \text{ atau } x \geq 1943 \\ \dfrac{x - 1931}{1943 - 1931} & ; 1931 \leq x \leq 1943 \\ \dfrac{7503 - x}{7503 - 1943} & ; 1943 \leq x \leq 7503 \end{cases} \tag{3.8}$$

$$\mu_{\text{besar}}(x) = \begin{cases} 0 & ; x \leq 1943 \\ \dfrac{x - 1943}{7503 - 1943} & ; 1943 \leq x \leq 7503 \\ 1 & ; x \geq 7503 \end{cases} \tag{3.9}$$

d. Variabel Jumlah Paket

Variabel jumlah paket didefinisikan dalam tiga himpunan yaitu sedikit, sedang, dan banyak. Fungsi keanggotaan variabel jumlah paket direpresentasikan dengan kurva bentuk bahu linear turun untuk himpunan sedikit, kurva bentuk segitiga untuk himpunan sedang, dan kurva bentuk bahu linear naik untuk himpunan banyak. Fungsi keanggotaan variabel jumlah paket ditunjukkan pada gambar 3.6.



Gambar 3. 6 Fungsi Keanggotaan Variabel Jumlah Paket

Fungsi keanggotaan dari variabel jumlah paket pada gambar 3.6 dapat dituliskan dengan persamaan:

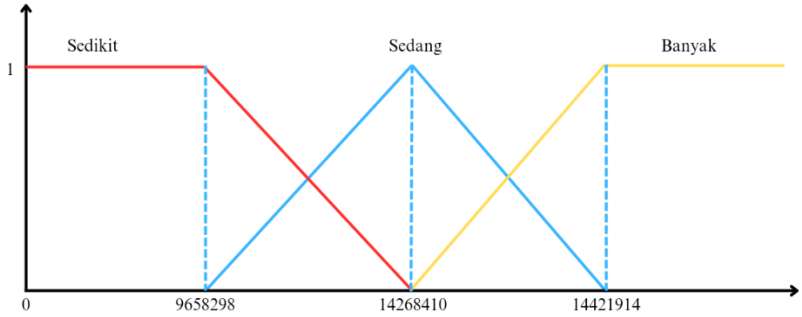
$$\mu_{\text{sedikit}}(x) = \begin{cases} 1 & ; x \leq 9269 \\ \dfrac{13385 - x}{13385 - 9269} & ; 9269 \leq x \leq 13385 \\ 0 & ; x \geq 13385 \end{cases} \quad (3.10)$$

$$\mu_{\text{sedang}}(x) = \begin{cases} 0 & ; x \leq 9269 \text{ atau } x \geq 13529 \\ \dfrac{x - 9269}{13385 - 9269} & ; 9269 \leq x \leq 13385 \\ \dfrac{13529 - x}{13529 - 13385} & ; 13385 \leq x \leq 13529 \end{cases} \quad (3.11)$$

$$\mu_{\text{banyak}}(x) = \begin{cases} 0 & ; x \leq 13385 \\ \dfrac{x - 13385}{13529 - 13385} & ; 13385 \leq x \leq 13529 \\ 1 & ; x \geq 13529 \end{cases} \quad (3.12)$$

e. Variabel Jumlah *Byte*

Variabel jumlah *byte* didefinisikan dalam tiga himpunan yaitu sedikit, sedang, dan banyak. Fungsi keanggotaan variabel jumlah *byte* direpresentasikan dengan kurva bentuk bahu linear turun untuk himpunan sedikit, kurva bentuk segitiga untuk himpunan sedang, dan kurva bentuk bahu linear naik untuk himpunan banyak. Fungsi keanggotaan variabel jumlah *byte* ditunjukkan pada gambar 3.7.



Gambar 3. 7 Fungsi Keanggotaan Variabel Jumlah *Byte*

Fungsi keanggotaan dari variabel jumlah *byte* pada gambar 3.7 dapat dituliskan dengan persamaan:

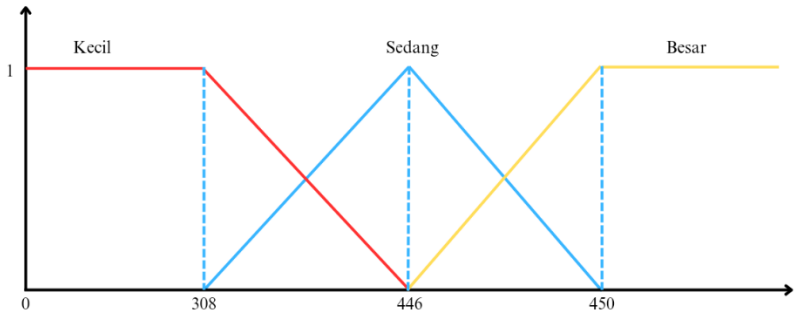
$$\mu_{sedikit}(x) = \begin{cases} 1 & ; x \leq 9658298 \\ \dfrac{14268410 - x}{14268410 - 9658298} & ; 9646836 \leq x \leq 14268410 \\ 0 & ; x \geq 14268410 \end{cases} \quad (3.13)$$

$$\mu_{sedang}(x) = \begin{cases} 0 & ; x \leq 9658298 \text{ atau } x \geq 13306 \\ \dfrac{x - 9658298}{14268410 - 9658298} & ; 9658298 \leq x \leq 14268410 \\ \dfrac{14421914 - x}{14421914 - 14268410} & ; 14268410 \leq x \leq 14421914 \end{cases} \quad (3.14)$$

$$\mu_{banyak}(x) = \begin{cases} 0 & ; x \leq 14268410 \\ \dfrac{x - 14268410}{14421914 - 14268410} & ; 14268410 \leq x \leq 14421914 \\ 1 & ; x \geq 14421914 \end{cases} \quad (3.15)$$

f. Variabel *Packet Rate*

Variabel *packet rate* didefinisikan dalam tiga himpunan yaitu kecil, sedang, dan besar. Fungsi keanggotaan variabel *packet rate* direpresentasikan dengan kurva bentuk bahu linear turun untuk himpunan kecil, kurva bentuk segitiga untuk himpunan sedang, dan kurva bentuk bahu linear naik untuk himpunan besar. Fungsi keanggotaan variabel *packet rate* ditunjukkan pada gambar 3.8.



Gambar 3. 8 Fungsi Keanggotaan Variabel *Packet Rate*

Fungsi keanggotaan dari variabel *packet rate* pada gambar 3.8 dapat dituliskan dengan persamaan:

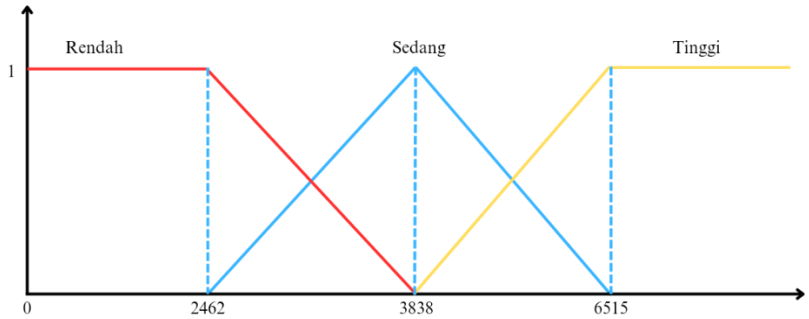
$$\mu_{\text{kecil}}(x) = \begin{cases} 1 & ; x \leq 308 \\ \dfrac{446 - x}{446 - 305} & ; 308 \leq x \leq 446 \\ 0 & ; x \geq 446 \end{cases} \tag{3.16}$$

$$\mu_{\text{sedang}}(x) = \begin{cases} 0 & ; x \leq 308 \text{ atau } x \geq 450 \\ \dfrac{x - 308}{446 - 308} & ; 308 \leq x \leq 446 \\ \dfrac{450 - x}{450 - 446} & ; 446 \leq x \leq 450 \end{cases} \tag{3.17}$$

$$\mu_{\text{besar}}(x) = \begin{cases} 0 & ; x \leq 446 \\ \dfrac{x - 446}{450 - 446} & ; 446 \leq x \leq 450 \\ 1 & ; x \geq 450 \end{cases} \tag{3.18}$$

g. Variabel *Port Bandwidth*

Variabel *port bandwidth* didefinisikan dalam tiga himpunan yaitu sedikit, sedang, dan banyak. Fungsi keanggotaan variabel *port bandwidth* direpresentasikan dengan kurva bentuk bahu linear turun untuk himpunan rendah, kurva bentuk segitiga untuk himpunan sedang, dan kurva bentuk bahu linear naik untuk himpunan tinggi. Fungsi keanggotaan variabel *port bandwidth* ditunjukkan pada gambar 3.9.



Gambar 3. 9 Fungsi Keanggotaan Variabel *Port Bandwidth*

Fungsi keanggotaan dari variabel *port bandwidth* pada gambar 3.9 dapat dituliskan dengan persamaan:

$$\mu_{\text{rendah}}(x) = \begin{cases} 1 & ; x \leq 2462 \\ \dfrac{3838 - x}{3838 - 2462} & ; 2462 \leq x \leq 3838 \\ 0 & ; x \geq 3838 \end{cases} \quad (3.19)$$

$$\mu_{\text{sedang}}(x) = \begin{cases} 0 & ; x \leq 2462 \text{ atau } x \geq 6515 \\ \dfrac{x - 2462}{3838 - 2462} & ; 2462 \leq x \leq 3838 \\ \dfrac{6515 - x}{6515 - 3838} & ; 3838 \leq x \leq 6515 \end{cases} \quad (3.20)$$

$$\mu_{\text{tinggi}}(x) = \begin{cases} 0 & ; x \leq 3838 \\ \dfrac{x - 3838}{6515 - 3838} & ; 3838 \leq x \leq 6515 \\ 1 & ; x \geq 6515 \end{cases} \quad (3.21)$$

3.5 Pembentukan Aturan *Fuzzy*

Aturan *fuzzy* digunakan untuk memperoleh hasil yang berupa nilai tegas (*crisp*). Aturan *fuzzy* yang digunakan pada penelitan ini adalah aturan *if-then* dengan operator *and* sebagai operasi antar variabel input. Pernyataan yang mengikuti kata "jika" disebut antiseden, sedangkan pernyataan yang mengikuti kata "maka" disebut konsekuen.

Pada tahap ini, peneliti melakukan fuzzifikasi terlebih dahulu pada data latih untuk mengetahui keanggotaan (*membership*) dari data masing-masing variabel masukan. Himpunan dengan nilai derajat keanggotaan tertinggi yang akan dijadikan acuan dalam membentuk aturan *fuzzy*. Langkah berikutnya adalah dengan menggabungkan data keanggotaan (membership) dari variabel input dan output, sehingga membentuk aturan *fuzzy*. Penggabungan data keanggotaan untuk membentuk aturan *fuzzy* ditunjukkan pada tabel 3.7.

Tabel 3. 7 Proses pembentukan aturan *fuzzy*

Algoritma 3: Pembentukan aturan *fuzzy*
Input: data himpunan keanggotaan seluruh variabel masukan **Output**: aturan *fuzzy*
Inisialisasi: himpunan keanggotaan seluruh variabel input, dataframe aturan *fuzzy* 1. Untuk setiap baris *dataframe* aturan *fuzzy*: 2. Tambahkan kolom data himpunan keanggotaan durasi 3. Tambahkan kolom data himpunan keanggotaan *flow* 4. Tambahkan kolom data himpunan keanggotaan *packet-in* 5. Tambahkan kolom data himpunan keanggotaan jumlah paket 6. Tambahkan kolom data himpunan keanggotaan jumlah *byte* 7. Tambahkan kolom data himpunan keanggotaan *rate* 8. Tambahkan kolom data himpunan keanggotaan *port bandwith* 9. Tambahkan kolom data himpunan keanggotaan label 10. simpan aturan *fuzzy* 11. **return** aturan *fuzzy*

Aturan *fuzzy* dapat dituliskan sebagai pernyataan "jika-maka" diantaranya sebagai berikut:

[R5] Jika durasi “panjang” dan *flow* “banyak” dan *packet-in* “besar” dan jumlah paket “sedikit” dan jumlah *byte* “sedikit” dan *rate* “kecil” dan *port bandwidth* “sedang”, maka status “DDoS”.

[R25] Jika durasi “panjang” dan *flow* “banyak” dan *packet-in* “sedang” dan jumlah paket “sedang” dan jumlah *byte* “sedang” dan *rate* “sedang” dan *port bandwidth* “sedang”, maka status “normal”.

[R50] Jika durasi “panjang” dan *flow* “sedang” dan *packet-in* “besar” dan jumlah paket “banyak” dan jumlah *byte* “banyak” dan *rate* “besar” dan *port bandwidth* “tinggi”, maka status “normal”.

[R145] Jika durasi “pendek” dan *flow* “sedang” dan *packet-in* “kecil” dan jumlah paket “sedikit” dan jumlah *byte* “sedikit” dan *rate* “kecil” dan *port bandwidth* “rendah”, maka status “DDoS”.

3.6 Fuzzifikasi

Proses mengubah nilai input *crisp* (tegas) menjadi nilai input *fuzzy*. Nilai tegas (*crisp*) dimasukkan ke dalam fungsi pengaburan, sehingga menghasilkan nilai input *fuzzy*. Terdapat 4 input dan 1 output yang digunakan dalam mengidentifikasi serangan DDoS pada *traffic* SDN. Input yang digunakan berupa durasi, *packet-in*, jumlah paket, dan *rate*. Output yang digunakan berupa status yang menunjukkan *traffic* normal dan DDoS. Dari input dan output tersebut akan difuzzifikasi ke dalam himpunan *fuzzy*.

Jika variabel durasi memiliki nilai masukan 423 berdasarkan fuangsi keanggotaan pada gambar 3.3, maka:

$$\mu_{\text{pendek}}(423) = 0$$

$$\mu_{\text{sedang}}(423) = 0$$

$$\mu_{\text{panjang}}(423) = 1$$

Jika variabel *flow* memiliki nilai masukan 3 berdasarkan fungsi keanggotaan pada gambar 3.4, maka:

$$\mu_{\text{sedikit}}(3) = 1$$

$$\mu_{\text{sedang}}(3) = 0$$

$$\mu_{\text{banyak}}(3) = 0$$

Jika variabel *packet-in* memiliki nilai masukan 1931 berdasarkan fungsi keanggotaan pada gambar 3.5, maka:

$$\mu_{\text{kecil}}(1931) = 1$$

$$\mu_{\text{sedang}}(1931) = 0$$

$$\mu_{\text{besar}}(1931) = 0$$

Jika variabel jumlah paket memiliki nilai masukan 7106 berdasarkan fungsi keanggotaan pada gambar 3.6, maka:

$$\mu_{\text{sedikit}}(7106) = 1$$

$$\mu_{\text{sedang}}(7106) = 0$$

$$\mu_{\text{banyak}}(7106) = 0$$

Jika variabel jumlah *byte* memiliki nilai masukan 7404452 berdasarkan fungsi keanggotaan pada gambar 3.7, maka:

$$\mu_{\text{sedikit}}(7404452) = 1$$

$$\mu_{\text{sedang}}(7404452) = 0$$

$$\mu_{\text{banyak}}(7404452) = 0$$

Jika variabel *rate* memiliki nilai masukan 236 berdasarkan fungsi keanggotaan pada gambar 3.8, maka:

$$\mu_{\text{kecil}}(236) = 1$$

$$\mu_{\text{sedang}}(236) = 0$$

$$\mu_{\text{besar}}(236) = 0$$

Jika variabel *port bandwidth* memiliki nilai masukan 423 berdasarkan fungsi keanggotaan pada gambar 3.9, maka:

$$\mu_{\text{rendah}}(423) = 0$$

$$\mu_{\text{sedang}}(423) = 0$$

$$\mu_{\text{tinggi}}(423) = 1$$

3.7 Inferensi

Tahap mengubah masukan *fuzzy* menjadi keluaran *fuzzy* berdasarkan aturan-aturan *fuzzy* yang telah ditentukan. Setiap aturan adalah sebuah pernyataan implikasi. Fungsi implikasi yang digunakan adalah fungsi implikasi min, sehingga, operator yang digunakan adalah operator *and*. Nilai yang diambil adalah nilai keanggotaan terkecil diantara elemen pada himpunan aturan yang terkait. Hasil fungsi implikasi dari setiap aturan disebut α-predikat atau ditulis α. Fungsi implikasi min dapat dihitung dengan persamaan 2.7. Pada proses ini tidak hanya menentukan nilai α-predikat, tetapi juga menentukan nilai z.

Misalkan nilai masukan variabel durasi = 423, *flow* = 3, *packet-in* = 1931, jumlah paket = 7106, jumlah *byte* = 7404452, *rate* = 236, dan *port bandwidth* = 2384. Berikut perhitungan α-predikat dan nilai z berdasarkan aturan *fuzzy*:

[R5] Jika durasi "panjang" dan *flow* "banyak" dan *packet-in* "besar" dan jumlah paket "sedikit" dan jumlah *byte* "sedikit" dan *rate* "kecil" dan *port bandwidth* "sedang", maka status "DDoS".

$$a_{predikat5} = \mu_{panjang} \cap \mu_{banyak} \cap \mu_{besar} \cap \mu_{sedikit} \cap \mu_{sedikit} \cap \mu_{kecil} \cap \mu_{sedang}$$
$$= \min(1; 0; 0; 1; 1; 1; 0)$$
$$= 0$$

$$\mu_{DDoS} = \frac{z - 0{,}4}{0{,}6 - 0{,}4}$$
$$0 = \frac{z - 0{,}4}{0{,}2}$$
$$z - 0{,}4 = 0$$
$$z = 0{,}4$$

[R25] Jika durasi "panjang" dan *flow* "banyak" dan *packet-in* "sedang" dan jumlah paket "sedang" dan jumlah *byte* "sedang" dan *rate* "sedang" dan *port bandwidth* "sedang", maka status "normal".

$$a_{predikat25} = \mu_{panjang} \cap \mu_{banyak} \cap \mu_{sedang} \cap \mu_{sedang} \cap \mu_{sedang} \cap \mu_{sedang} \cap \mu_{sedang}$$
$$= \min(1; 0; 0; 0; 0; 0; 0)$$
$$= 0$$

$$\mu_{normal} = \frac{0{,}6 - z}{0{,}6 - 0{,}4}$$
$$0 = \frac{0{,}6 - z}{0{,}2}$$
$$0 = 0{,}6 - z$$
$$z = 0{,}6$$

[R50] Jika durasi "panjang" dan *flow* "sedang" dan *packet-in* "besar" dan jumlah paket "banyak" dan jumlah *byte* "banyak" dan *rate* "besar" dan *port bandwidth* "tinggi", maka status "normal".

$$a_{predikat50} = \mu_{panjang} \cap \mu_{sedang} \cap \mu_{besar} \cap \mu_{banyak} \cap \mu_{banyak} \cap \mu_{besar} \cap \mu_{tinggi}$$

$$\mu_{normal} = \frac{0{,}6 - z}{0{,}6 - 0{,}4}$$
$$0 = \frac{0{,}6 - z}{0{,}2}$$

$= \min(1; 0; 0; 0; 0; 0; 1)$ $\quad 0 = 0{,}6 - z$

$= 0$ $\quad z = 0{,}6$

[R145] Jika durasi "pendek" dan *flow* "sedang" dan *packet-in* "kecil" dan jumlah paket "sedikit" dan jumlah *byte* "sedikit" dan *rate* "kecil" dan *port bandwidth* "rendah", maka status "DDoS".

$a_{predikat145} = \mu_{pendek} \cap \mu_{sedang} \cap \mu_{kecil} \cap \mu_{sedikit} \cap \mu_{sedikit} \cap \mu_{kecil} \cap \mu_{rendah}$ $\quad \mu_{DDoS} = \frac{z - 0{,}4}{0{,}6 - 0{,}4}$

$= \min(0; 0; 1; 1; 1; 1; 0)$ $\quad 0 = \frac{z - 0{,}4}{0{,}2}$

$= 0$ $\quad z - 0{,}4 = 0$

$z = 0{,}4$

3.8 Defuzzifikasi

Defuzzifikasi merupakan proses mengubah nilai output *fuzzy* menjadi nilai tegas (*crisp*). Metode *fuzzy* tsukamoto menggunakan rata-rata terbobot sebagai proses defuzzifikasi.

$$\mathrm{Z} = \frac{\sum z_i \cdot a_i}{\sum a_i}, i = 1,2,3, \ldots \tag{3.22}$$

Z : nilai rata-rata terbobot
z_i : nilai konsekuen pada aturan ke-i
a_i : nilai α-predikat pada aturan ke-i

Berdasarkan contoh fuzzifikasi dan inferensi sebelumnya, maka dapat dilakukan defuzzifikasi dengan persamaan 3.22 sebagai berikut:

$$\mathrm{Z} = \frac{z_1 * a_{pred1} + z_2 * a_{pred2} + z_3 * a_{pred3} + \ldots + z_n * a_n}{a_{pred1} + a_{pred2} + a_{pred3} + \ldots + a_n}$$

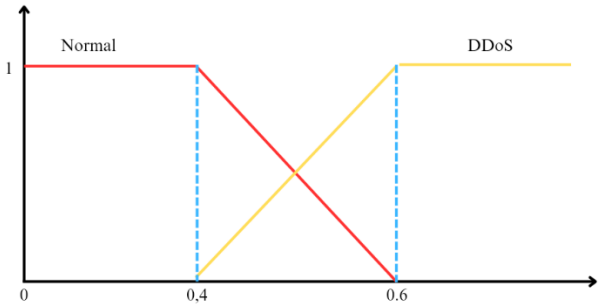
$$Z = \frac{0{,}6(0) + 0{,}6(0) + 0{,}6(0) + \ldots + 0{,}4(0)}{0 + 0 + 0 + \ldots + 0}$$

$$Z = 0{,}6$$

Dari hasil defuzzifikasi dapat diketahui bahwa nilai input setiap variabel dari contoh perhitungan menghasilkan output berupa nilai tegas sebesar 0,6.

3.9 Keputusan Hasil Deteksi

Nilai keluaran yang berupa nilai tegas (crisp) akan dihitung nilai derajat keanggotaan statusnya untuk menentukan status dari *traffic* data. Jika nilai derajat keanggotaan status normal lebih besar dari nilai keanggotaan DDoS, maka data *traffic* teridentifikasi sebagai *traffic* dengan status normal dan sebaliknya. Fungsi keanggotaan variabel status direpresentasikan dengan kurva bentuk bahu linear turun untuk himpunan normal dan kurva bentuk bahu linear naik untuk himpunan DDoS. Fungsi keanggotaan status ditunjukkan pada gambar 3.10.



Gambar 3. 10 Fungsi Keanggotaan Status

Jika nilai output crisp yang dihasilkan adalah 0,6, maka dapat dihitung derajat keanggotaan normal dan DDoS:

$$\mu_{normal}[0{,}6] = \frac{0{,}6 - z}{0{,}6 - 0{,}4} \qquad \mu_{DDoS}[0{,}6] = \frac{z - 0{,}4}{0{,}6 - 0{,}4}$$

$$= \frac{0,6 - 0,6}{0,6 - 0,4} \qquad = \frac{0,6 - 0,4}{0,6 - 0,4}$$

$$= 0 \qquad = 1$$

Dapat diketahui dari hasil perhitungan bahwa nilai tegas 0,6 memiliki derajat keanggotaan DDoS lebih besar dari keanggotaan normal, maka durasi = 423, *flow* = 3, *packet-in* = 1931, jumlah paket = 7106, jumlah *byte* = 7404452, *rate* = 236, dan *port bandwidth* = 2384 teridentifikasi sebagai *traffic* DDoS.

3.10 Desain Eksperimen

Desain eksperimen dalam melakukan identifikasi serangan DDoS pad SDN mencakup pra-pemrosesan data, pelatihan model, pengujian model, dan evaluasi.

1. Pra-pemrosesan Data

Setelah berhasil mengumpulkan data, langkah berikutnya adalah pra-pemrosesan data. Pra-pemrosesan data pada penelitian ini mencakup pengambilan data pada parameter yang digunakan, *data cleaning*, dan pembagian data menjadi data uji dan data latih. Data hasil prapemrosesan berjumlah 4174 data dari 33588 data.

2. Pelatihan Model

Pelatihan model dilakukan dengan menggunakan data latih untuk menentukan batas-batas nilai linguistik dari variabel-variabel yang digunakan dan derajat keanggotaan dari nilai input. Derajat keanggotaan dari masing-masing data akan membentuk pola. Pola tersebut yang akan dijadikan acuan dalam membentuk aturan *fuzzy*.

3. Pengujian Model

Pada tahap ini dilakukan analisis hasil dari model yang dibuat menggunakan data uji. Pada pengujian model terdapat proses fuzzifikasi, inferensi, dan defuzzifikasi. Output dari proses-proses tersebut yang berupa nilai tegas (*crisp*) kemudian digunakan untuk memutuskan hasil deteksi.

4. Evaluasi Hasil

Jika hasil pengujian menunjukkan tingkat akurasi model masih rendah, maka akan diperbaiki batas nilai linguitsik, basis aturan, atau bentuk fungsi keanggotaan. Evaluasi hasil dilakukan dengan menghitung nilai akurasi dari hasil deteksi yang dihasilkan sistem menggunakan tabel *confussion matrix* yang ditunjukkan pada tabel 3.8.

Tabel 3. 8 *Confussion matrix*

Actual / ***Prediction***	***Positive***	***Negative***
Positive	*True Positive (TP)*	*False Negative (FN)*
Negative	*False Positive (FP)*	*True Negative (TN)*

Menghitung akurasi dengan menggunakan *confussion matrix* dapat menggunakan persamaan:

$$Akurasi = \frac{True\ Positive + True\ Negative}{Total\ Instances} \times 100\% \qquad (3.23)$$

3.11 Skenario Uji Coba

Uji coba dilakukan untuk mengetahui akurasi dari sistem dalam mendeteksi *traffic* normal dan DDoS. Untuk mengetahui akurasi dari sistem yang dibuat, peneliti menghitung nilai akurasi dengan persamaan 3.23. Pengujian diawali

dengan menghitung nilai derajat keanggotaan masing-masing variabel input (fuzzifikasi), kemudian menghitung nilai α-predikat dan nilai z dari masing-masing aturan *fuzzy* (inferensi). Nilai α-predikat dan z tersebut kemudian digunakan untuk menghitung nilai tegas dengan metode *weighted average* (defuzzifikasi). Nilai tegas kemudian digunakan untuk menghitung derajat nilai keanggotaan status *traffic*. Jika nilai derajat keanggotaan status normal lebih besar dari nilai keanggotaan status DDoS, maka hasil deteksi *traffic* adalah normal. Sebaliknya, jika nilai derajat keanggotaan status DDoS lebih besar dari nilai keanggotaan status normal, maka hasil deteksi *traffic* adalah DDoS. Uji coba sistem dilakukan menggunakan *k-fold cross validation* dengan k=5. Data yang digunakan dalam pengujian sistem sebanyak 4174. *Fold* 1-4 terdiri dari 3267 data latih dan 907 data uji, sedangkan *fold* 5 terdiri dari 3268 data latih dan 906 data uji.

3.12 Komponen Perancangan Sistem

Sistem deteksi DDoS pada SDN diimplementasikan dengan pemrograman python melalui Google Colab. Komponen-komponen dalam perancangan ini mencakup perangkat keras dan perangkat lunak. Berikut perangkat-perangkat yang digunakan, diantaranya:

a. Perangkat Keras
    1. Laptop ASUS X441M
    2. RAM 4 GB
    3. SSD 1 TB
b. Perangkat Lunak
    1. Sistem Operasi Windows 10

2. Google Colab
3. Web Browser
4. Bahasa pemrograman Python

BAB IV

HASIL DAN PEMBAHASAN

4.1 Fuzzifikasi

Fuzzifikasi atau pengaburan merupakan tahap mengubah data *input* tegas (*crisp*) menjadi input *fuzzy*. Penelitian ini menggunakan 7 variabel dalam mendeteksi DDoS pada SDN, yaitu variabel durasi, *flow*, *packet-in*, jumlah paket, jumlah *byte*, *packet rate*, dan *port bandwidth* sebagai variabel input dan variabel status sebagai variabel output.

Berikut representasi fungsi keanggotaan empat variabel masukan *fuzzy* yang digunakan:

a. Variabel Durasi

Variabel durasi didefinisikan dalam tiga himpunan yaitu pendek, sedang, dan panjang. Domain untuk himpunan pendek: 28-190, sedang: 108-270, dan panjang: 190-477. Penghitungan derajat keanggotaan nilai masukan variabel durasi ditunjukkan pada tabel 4.1.

Tabel 4. 1 Proses fuzzifikasi variabel durasi

Algoritma 4: Fuzzifikasi variabel durasi
Input: nilai variabel durasi **Output**: nilai derajat keanggotaan data variabel durasi
Inisialisasi: dataframe fuzzifikasi durasi, domain tiap himpunan fuzzy variabel durasi 1. Untuk setiap nilai durasi: 2. Hitung derajat keanggotaan nilai tersebut pada himpunan fuzzy pendek. 3. Hitung derajat keanggotaan nilai tersebut pada himpunan fuzzy sedang. 4. Hitung derajat keanggotaan nilai tersebut pada himpunan fuzzy panjang. 5. Simpan hasil perhitungan derajat keanggotaan dalam dataframe variabel durasi. 6. **return** hasil fuzzifikasi variabel durasi.

Output fuzzifikasi variabel durasi ditunjukkan pada tabel 4.2.

Tabel 4. 2 Hasil fuzzifikasi variabel durasi

No.	Input	Derajat Keanggotaan		
	Durasi	Pendek	Sedang	Panjang
1	390	0	0	1
2	138	0.64	0.36	0
3	187	0.04	0.96	0
4	390	0	0	1
5	76	1	0	0
6	72	1	0	0
7	255	0	0.19	0.81
8	197	0	0.91	0.09
9	107	1	0	0
10	113	0.95	0.05	0
11	318	0	0	1
12	450	0	0	1
13	97	1	0	0
14	127	0.78	0.22	0
15	312	0	0	1

b. Variabel *Flow*

Variabel *flow* didefinisikan dalam tiga himpunan yaitu sedikit, sedang, dan banyak. Domain untuk himpunan sedikit: 2-4, sedang: 3-5, dan banyak: 4-8. Penghitungan derajat keanggotaan nilai masukan variabel jumlah *flow* ditunjukkan pada tabel 4.3.

Tabel 4. 3 Proses fuzzifikasi variabel *flow*

Algoritma 5: Fuzzifikasi variabel *flow*
Input: nilai variabel *flow* **Output**: nilai derajat keanggotaan data variabel *flow*
Inisialisasi: dataframe fuzzifikasi *flow*, domain tiap himpunan fuzzy variabel *flow* 1. Untuk setiap nilai *flow*: 2. Hitung derajat keanggotaan nilai tersebut pada himpunan fuzzy sedikit. 3. Hitung derajat keanggotaan nilai tersebut pada himpunan fuzzy sedang. 4. Hitung derajat keanggotaan nilai tersebut pada himpunan fuzzy banyak. 5. Simpan hasil perhitungan derajat keanggotaan dalam dataframe variabel *flow*. 6. **return** hasil fuzzifikasi variabel *flow*.

Output fuzzifikasi variabel jumlah *flow* ditunjukkan pada tabel 4.4.

Tabel 4. 4 Hasil fuzzifikasi variabel *flow*

No.	Input	Derajat Keanggotaan		
	Flow	**Sedikit**	**Sedang**	**Banyak**
1	3	1	0	0
2	7	0	0	1
3	2	1	0	0
4	3	1	0	0
5	3	1	0	0
6	4	0	1	0
7	4	0	1	0
8	2	1	0	0
9	2	1	0	0
10	3	1	0	0
11	6	0	0	1
12	3	1	0	0
13	5	0	0	1
14	4	0	1	0
15	3	1	0	0

c. Variabel *Packet-in*

Variabel *packet-in* didefinisikan dalam tiga himpunan yaitu kecil, sedang, dan besar. Domain untuk himpunan kecil: 219-1943, sedang: 1931-7503, dan besar: 1943-8803. Penghitungan derajat keanggotaan nilai masukan *packet-in* durasi ditunjukkan pada tabel 4.5.

Tabel 4. 5 Proses fuzzifikasi variabel *pakcet-in*

Algoritma 6: Fuzzifikasi variabel *packet-in*
Input: nilai variabel *packet-in* **Output**: nilai derajat keanggotaan data variabel *packet-in*
Inisialisasi: *dataframe* fuzzifikasi *packet-in*, domain tiap himpunan fuzzy variabel *packet-in* 1. Untuk setiap nilai *packet-in*: 2. Hitung derajat keanggotaan nilai tersebut pada himpunan *fuzzy* kecil. 3. Hitung derajat keanggotaan nilai tersebut pada himpunan *fuzzy* sedang. 4. Hitung derajat keanggotaan nilai tersebut pada himpunan *fuzzy* besar. 5. Simpan hasil perhitungan derajat keanggotaan dalam dataframe variabel *packet-in*. 6. **return** hasil fuzzifikasi variabel *packet-in*.

Output fuzzifikasi variabel *packet-in* ditunjukkan pada tabel 4.6.

Tabel 4. 6 Hasil fuzzifikasi variabel *packet-in*

No.	Input	Derajat Keanggotaan		
	Packet-in	**Kecil**	**Sedang**	**Besar**
1	1943	0	1	0
2	1790	1	0	0
3	7916	0	0	1
4	1943	0	1	0
5	8803	0	0	1
6	1931	1	0	0
7	7916	0	0	1
8	7916	0	0	1
9	7503	0	0	1
10	2375	0	0.92	0.08
11	1943	0	1	0
12	1943	0	1	0
13	2375	0	0.92	0.08
14	2385	0	0.92	0.08
15	1931	1	0	0

d. Variabel Jumlah Paket

Variabel jumlah paket didefinisikan dalam tiga himpunan yaitu sedikit, sedang, dan banyak. Domain untuk himpunan sedikit: 2604-13385, sedang: 9269-13529, dan banyak: 13385-13685. Penghitungan derajat keanggotaan nilai masukan variabel jumlah paket ditunjukkan pada tabel 4.7.

Tabel 4. 7 Proses fuzzifikasi variabel jumlah paket

Algoritma 7: Fuzzifikasi variabel jumlah paket
Input: nilai variabel jumlah paket **Output**: nilai derajat keanggotaan data variabel jumlah paket
Inisialisasi: dataframe fuzzifikasi jumlah paket, domain tiap himpunan *fuzzy* variabel jumlah paket 1. Untuk setiap nilai jumlah paket: 2. Hitung derajat keanggotaan nilai tersebut pada himpunan *fuzzy* sedikit. 3. Hitung derajat keanggotaan nilai tersebut pada himpunan *fuzzy* sedang. 4. Hitung derajat keanggotaan nilai tersebut pada himpunan *fuzzy* banyak. 5. Simpan hasil perhitungan derajat keanggotaan dalam dataframe variabel jumlah paket. 6. **return** hasil fuzzifikasi variabel jumlah paket.

Output fuzzifikasi variabel jumlah paket ditunjukkan pada tabel 4.8.

Tabel 4. 8 Hasil fuzzifikasi variabel jumlah paket

No.	Input	Derajat Keanggotaan		
	Jumlah Paket	Sedikit	Sedang	Banyak
1	7952	1	0	0
2	13528	0	0.01	0.99
3	13536	0	0	1
4	7952	1	0	0
5	13581	0	0	1
6	13385	0	1	0
7	13526	0	0.02	0.98
8	9208	1	0	0
9	13548	0	0	1
10	13443	0	0.60	0.40
11	4288	1	0	0
12	2604	1	0	0
13	13529	0	0	1
14	13466	0	0.44	0.56
15	7570	1	0	0

e. Variabel Jumlah *Byte*

Variabel jumlah *byte* didefinisikan dalam tiga himpunan yaitu sedikit, sedang, dan banyak. Domain untuk himpunan sedikit: 2713368-14268410, sedang: 9658298-14421914, dan banyak: 14268410-14588210. Penghitungan derajat keanggotaan nilai masukan variabel jumlah *byte* ditunjukkan pada tabel 4.9.

Tabel 4. 9 Proses fuzzifikasi variabel jumlah *byte*

Algoritma 8: Fuzzifikasi variabel jumlah *byte*
Input: nilai variabel *jumlah byte* **Output**: nilai derajat keanggotaan data variabel *jumlah byte*
Inisialisasi: *dataframe* fuzzifikasi *jumlah byte*, domain tiap himpunan *fuzzy* variabel jumlah *byte* 7. Untuk setiap nilai jumlah *byte*: 8. Hitung derajat keanggotaan nilai tersebut pada himpunan *fuzzy* sedikit. 9. Hitung derajat keanggotaan nilai tersebut pada himpunan *fuzzy* sedang. 10. Hitung derajat keanggotaan nilai tersebut pada himpunan *fuzzy* banyak. 11. Simpan hasil perhitungan derajat keanggotaan dalam dataframe variabel jumlah *byte*. 12. **return** hasil fuzzifikasi variabel jumlah *byte*.

Output fuzzifikasi variabel jumlah *byte* ditunjukkan pada tabel 4.10.

Tabel 4. 10 Hasil fuzzifikasi variabel jumlah *byte*

No.	Input	Derajat Keanggotaan		
	Jumlah *Byte*	Sedikit	Sedang	Banyak
1	8285984	1	0	0
2	14420848	0	0.01	0.99
3	14429376	0	0	1
4	8285984	1	0	0
5	14477346	0	0	1
6	14268410	0	1	0
7	14418716	0	0.02	0.98
8	9594736	1	0	0
9	14442168	0	0	1
10	14330238	0	0.60	0.40
11	4571008	1	0	0
12	2713368	1	0	0
13	14421914	0	0	1
14	14354756	0	0.44	0.56
15	8069620	1	0	0

f. Variabel *Packet Rate*

Variabel *packet rate* didefinisikan dalam tiga himpunan yaitu kecil, sedang, dan besar. Domain untuk himpunan kecil: 86-446, sedang: 308-450, dan besar: 446-456. Penghitungan derajat keanggotaan nilai masukan variabel *packet rate* ditunjukkan pada tabel 4.11.

Tabel 4. 11 Proses fuzzifikasi variabel *packet rate*

Algoritma 9: Fuzzifikasi variabel *packet rate*
Input: nilai variabel *packet rate* **Output**: nilai derajat keanggotaan data variabel *packet rate*
Inisialisasi: *dataframe* fuzzifikasi *packet rate*, domain tiap himpunan *fuzzy* variabel *packet rate* 13. Untuk setiap nilai packet rate: 14. Hitung derajat keanggotaan nilai tersebut pada himpunan *fuzzy* kecil. 15. Hitung derajat keanggotaan nilai tersebut pada himpunan *fuzzy* sedang. 16. Hitung derajat keanggotaan nilai tersebut pada himpunan *fuzzy* besar. 17. Simpan hasil perhitungan derajat keanggotaan dalam dataframe variabel *packet rate*. 18. **return** hasil fuzzifikasi variabel *packet rate*.

Output program fuzzifikasi variabel *rate* ditunjukkan pada tabel 4.12.

Tabel 4. 12 Hasil fuzzifikasi variabel *rate*

No.	Input	Derajat Keanggotaan		
	Rate	Kecil	Sedang	Besar
1	451	0	0	1
2	451	0	0	1
3	451	0	0	1
4	451	0	0	1
5	443	0	1	0
6	443	0	1	0
7	446	0	0.57	0.43
8	446	0	0.57	0.43
9	446	0	0.57	0.43
10	451	0	0	1
11	451	0	0	1
12	451	0	0	1
13	451	0	0	1
14	451	0	0	1
15	451	0	0	1

g. Variabel *Port Bandwidth*

Variabel *port bandwidth* didefinisikan dalam tiga himpunan yaitu kecil, sedang, dan besar. Domain untuk himpunan rendah: 0-3838, sedang: 2462-6515, dan tinggi: 3838-14993. Penghitungan derajat keanggotaan nilai masukan variabel *port bandwidth* ditunjukkan pada tabel 4.13.

Tabel 4. 13 Proses fuzzifikasi variabel *port bandwidth*

Algoritma 10: Fuzzifikasi variabel *port bandwidth*
Input: nilai variabel *port bandwidth* **Output**: nilai derajat keanggotaan data variabel *port bandwidth*
Inisialisasi: dataframe fuzzifikasi *port bandwidth*, domain tiap himpunan fuzzy variabel *port bandwidth* 19. Untuk setiap nilai *port bandwidth*: 20. Hitung derajat keanggotaan nilai tersebut pada himpunan fuzzy rendah. 21. Hitung derajat keanggotaan nilai tersebut pada himpunan fuzzy sedang. 22. Hitung derajat keanggotaan nilai tersebut pada himpunan fuzzy tinggi. 23. Simpan hasil perhitungan derajat keanggotaan dalam dataframe variabel *port bandwidth*. 24. **return** hasil fuzzifikasi variabel *port bandwidth*.

Output program fuzzifikasi variabel *port bandwidth* ditunjukkan pada tabel 4.14.

Tabel 4. 14 Hasil fuzzifikasi variabel *port bandwidth*

No.	Input	Derajat Keanggotaan		
	Port Bandwidth	**Panjang**	**Sedang**	**Pendek**
1	451	0	0	1
2	451	0	0	1
3	451	0	0	1
4	451	0	0	1
5	443	0	1	0
6	443	0	1	0
7	446	0	0.57	0.43
8	446	0	0.57	0.43
9	446	0	0.57	0.43
10	451	0	0	1
11	451	0	0	1
12	451	0	0	1
13	451	0	0	1
14	451	0	0	1
15	451	0	0	1

4.2 Inferensi

Penelitian ini menggunakan fungsi implikasi min (oprator “and”), sehingga nilai terkecil diantara himpunan *rule* yang akan diambil. Fungsi implikasi dari setiap aturan menghasilkan α-predikat dan nilai z dari masing-masing aturan *fuzzy*. Proses inferensi *fuzzy* ditunjukkan pada table 4.15.

Tabel 4. 15 Inferensi *fuzzy*

Algoritma 11: Inferensi
Input: nilai fuzzifikasi setiap variabel input **Output**: nilai α-predikat dan z dari setiap aturan *fuzzy*
Inisialisasi: derajat keanggotaan seluruh variabel masukan, aturan *fuzzy* 1. Untuk setiap aturan *fuzzy*: 2. Hitung nilai minimal dari keempat derajat keangggotaan variabel masukan (α-predikat). 3. Hitung nilai konsekuen (z). 4. Simpan hasil perhitungan α-predikat dan z dari setiap aturan *fuzzy*. 5. **return** hasil inferensi.

4.3 Defuzzifikasi

Pada proses defuzzifikasi dilakukan proses perubahan nilai *output fuzzy* menjadi nilai tegas. Penelitian ini menggunakan metode rata-rata terbobot sebagai proses defuzzifikasi. Proses defuzzifikasi ditunjukkan pada 4.16.

Tabel 4. 16 Proses defuzzifikasi

Algoritma 12: Defuzzifikasi menggunakan *weighted average*
Input: nilai α-predikat dan z dari setiap aturan fuzzy **Output**: nilai rata-rata terbobot
Inisialisasi: α-predikat, konsekuen (z) 1. Untuk setiap aturan fuzzy: 2. Kalikan α-predikat dan konsekuen (z). 3. num = Jumlahkan hasil kali α-predikat dan konsekuen (z) 4. denum = Jumlahkan seluruh nilai α-predikat 5. **end** 6. defuzz = num/denum 7. Simpan hasil perhitungan rata-rata terbobot. 8. **return** hasil defuzzifikasi.

4.4 Keputusan Hasil Deteksi

Proses fuzzifikasi nilai keluaran untuk mengetahui derajat keanggotaan status *traffic* dan penentuan hasil deteksi ditunjukkan pada table 4.17.

Tabel 4. 17 Proses penentuan hasil deteksi

Algoritma 13: Menentukan hasil deteksi
Input: nilai defuzzifikasi **Output**: nilai derajat keangggotaan status dan hasil deteksi
Inisialisasi: Z (rata-rata terbobot), fuzzifikasi status 1. α-predikat normal = (0,6 - Z) / 0,6 – 0,4 2. α-predikat DDoS = (Z – 0,4) / 0,6 – 0,4 3. **if** α-predikat normal > α-predikat DDoS: 4. status traffic = normal 5. **if** α-predikat DDoS > α-predikat normal: 6. status traffic = DDoS 7. Simpan hasil deteksi 8. **return** hasil deteksi.

Output dari sistem deteksi DDoS pada SDN ditunjukkan pada tabel 4.18.

Tabel 4. 18 Hasil deteksi

No.	Durasi	*Flow*	*Packet-in*	Jumlah Paket	Jumlah *Byte*	*Rate*	*Port Banwidth*	Label Aktual	Label Prediksi
1	4	7503	13548	14442168	451	977.0	0	0	0
2	53	4	7503	13548	14442168	451	0.0	0	0
3	53	4	7503	13548	14442168	451	3838.0	0	0
4	226	5	7916	8849	9220658	294	2518.0	1	0
5	192	4	7894	13531	14424046	451	10313.0	0	0
6	144	4	7894	13531	14424046	451	14152.0	0	0
7	162	4	7894	13421	14306786	447	9234.0	0	1
8	347	2	4440	9168	9553056	305	8912.0	1	1
9	347	2	4440	9168	9553056	305	2543.0	1	1
10	114	4	7894	13420	14305720	447	13071.0	0	0
11	377	2	4440	7901	8232842	263	3690.0	1	1
12	377	2	4440	7901	8232842	263	0.0	1	1
13	54	3	7503	13548	14442168	451	977.0	0	0
14	102	3	7503	13548	14442168	451	7676.0	0	0
15	54	3	7503	13548	14442168	451	7676.0	0	0

4.5 Analisa Uji Coba

Tahap selanjutnya adalah pengujian sistem. Pengujian menggunakan *k-fold cross validation* dengan k=5. Hasil pengujian pada *fold* 1 dapat dipetakan ke dalam tabel *matrix confussion* yang ditunjukkan pada tabel 4.19.

Tabel 4. 19 Tabel *confussion matrix fold* 1

Aktual / **Prediksi**	**Normal**	**DDoS**
Normal	395	121
DDoS	82	309

Dari tabel pemetaan diperoleh nilai *True Positive* (TP) = 394 dari data yang teridentifikasi normal dari prediksi dan aktual. Diperoleh nilai *True Negative* (TN) = 306 dari data yang teridentifikasi DDoS dari prediksi dan aktual. Dari hasil tersebut dapat dihitung nilai akurasi sebagai berikut:

$$Akurasi = \frac{395 + 309}{907} \times 100\%$$

$$= \frac{704}{907} \times 100\%$$

$$= 77{,}62\%$$

Berdasarkan perhitungan akurasi dari sistem deteksi DDoS pada *fold* 1 menunjukkan tingkat akurasi deteksi mencapai 77,62%.

Hasil pengujian pada *fold* 2 dapat dipetakan ke dalam tabel *matrix confussion* yang ditunjukkan pada tabel 4.20.

Tabel 4. 20 Tabel *confussion matrix fold* 2

Aktual / **Prediksi**	**Normal**	**DDoS**
Normal	388	129
DDoS	72	318

Dari tabel pemetaan diperoleh nilai *True Positive* (TP) = 388 dari data yang teridentifikasi normal dari prediksi dan aktual. Diperoleh nilai *True Negative* (TN) = 318 dari data yang teridentifikasi DDoS dari prediksi dan aktual. Dari hasil tersebut dapat dihitung nilai akurasi sistem deteksi DDoS sebagai berikut:

$$Akurasi = \frac{388 + 318}{907} \times 100\%$$

$$= \frac{706}{907} \times 100\%$$

$$= 77{,}84\%$$

Berdasarkan perhitungan akurasi dari sistem deteksi DDoS pada *fold* 2 menunjukkan tingkat akurasi deteksi mencapai 77,84%.

Hasil pengujian pada *fold* 3 dapat dipetakan ke dalam tabel *matrix confussion* yang ditunjukkan pada tabel 4.21.

Tabel 4. 21 Tabel *confussion matrix fold* 3

Aktual / **Prediksi**	**Normal**	**DDoS**
Normal	371	143
DDoS	79	314

Dari tabel pemetaan diperoleh nilai *True Positive* (TP) = 371 dari data yang teridentifikasi normal dari prediksi dan aktual. Diperoleh nilai *True Negative* (TN) = 314 dari data yang teridentifikasi DDoS dari prediksi dan aktual. Dari hasil tersebut dapat dihitung nilai akurasi sistem deteksi DDoS sebagai berikut:

$$Akurasi = \frac{371 + 314}{907} \times 100\%$$

$$= \frac{685}{907} \times 100\%$$

$$= 75{,}52\%$$

Berdasarkan perhitungan akurasi dari sistem deteksi DDoS pada *fold* 3 menunjukkan tingkat akurasi deteksi mencapai 75,52%.

Hasil pengujian pada *fold* 4 dapat dipetakan ke dalam tabel *matrix confussion* yang ditunjukkan pada tabel 4.22.

Tabel 4. 22 Tabel *confussion matrix fold* 4

Aktual / **Prediksi**	**Normal**	**DDoS**
Normal	359	122
DDoS	101	325

Dari tabel pemetaan diperoleh nilai *True Positive* (TP) = 359 dari data yang teridentifikasi normal dari prediksi dan aktual. Diperoleh nilai *True Negative* (TN) = 325 dari data yang teridentifikasi DDoS dari prediksi dan aktual. Dari hasil tersebut dapat dihitung nilai akurasi sistem deteksi DDoS sebagai berikut:

$$Akurasi = \frac{359 + 325}{907} \times 100\%$$

$$= \frac{684}{907} \times 100\%$$

$$= 75{,}41\%$$

Berdasarkan perhitungan akurasi dari sistem deteksi DDoS pada *fold* 4 menunjukkan tingkat akurasi deteksi mencapai 75,41%.

Hasil pengujian pada *fold* 5 dapat dipetakan ke dalam tabel *matrix confussion* yang ditunjukkan pada tabel 4.23.

Tabel 4. 23 Tabel *confussion matrix fold* 5

Aktual / **Prediksi**	**Normal**	**DDoS**
Normal	382	132
DDoS	80	312

Dari tabel pemetaan diperoleh nilai *True Positive* (TP) = 382 dari data yang teridentifikasi normal dari prediksi dan aktual. Diperoleh nilai *True Negative* (TN) = 312 dari data yang teridentifikasi DDoS dari prediksi dan aktual. Dari hasil tersebut dapat dihitung nilai akurasi sistem deteksi DDoS sebagai berikut:

$$Akurasi = \frac{382 + 312}{906} \times 100\%$$

$$= \frac{694}{906} \times 100\%$$

$$= 76{,}60\%$$

Berdasarkan perhitungan akurasi dari sistem deteksi DDoS pada *fold* 5 menunjukkan tingkat akurasi deteksi mencapai 76,60%.

Hasil perhitungan akurasi dari 5 *fold* menunjukkan bahwa akurasi sistem deteksi DDoS menggunakan *fuzzy* Tsukamoto pada penelitian ini menghasilkan akurasi minimal sebesar 75,41%, maksimal sebesar 77,84%, dan rata-rata akurasi sebesar 76,60%.

Akurasi sistem yang bagus dapat memungkinkan sistem memiliki peluang besar dalam mendeteksi dan mengidentifikasi serangan DDoS dengan tepat. Islam sendiri menganjurkan untuk memastikan kebenaran atau kejelasan tentang sesuatu sebelum disebarkan. Anjuran tersebut terkandung pada Al-Quran surah Al-Hujurat ayat 6 yang berbunyi:

يٰٓاَيُّهَا الَّذِيْنَ اٰمَنُوْٓا اِنْ جَاۤءَكُمْ فَاسِقٌۢ بِنَبَاٍ فَتَبَيَّنُوْٓا اَنْ تُصِيْبُوْا قَوْمًاۢ بِجَهَالَةٍ فَتُصْبِحُوْا عَلٰى مَا فَعَلْتُمْ نٰدِمِيْنَ

"Wahai orang-orang yang beriman, jika seorang fasik datang kepadamu membawa berita penting, maka telitilah kebenarannya agar kamu tidak mencelakakan suatu kaum karena ketidaktahuan(-mu) yang berakibat kamu menyesali perbuatanmu itu." (QS. Al-Hujurat [49]:6)

Berdasarkan tafsir Kemenag, ayat tersebut menjelaskan bahwa kaum muslim harus berhati-hati dalam menerima berita, terutama dari orang fasik. Penting untuk melakukan *tabayyun* (klarifikasi) dan memastikan kebenaran berita tersebut sebelum menyebarkannya. Hal tersebut dilakukan untuk menghindari penyesalan dan fitnah yang dapat ditimbulkan.

Tabayyun dalam konteks penelitian ini berkaitan dengan kejelasan atau kebenaran adanya serangan DDoS pada suatu jaringan. Hal tersebut dapat dilakukan dengan mengidentifikasi serangan DDoS berdasarkan data *traffic* pada suatu jaringan dengan tepat. Semakin bagus akurasi pada sistem deteksi, maka semakin akurat atau jelas informasi yang dihasilkan.

Serangan DDoS dapat menyebabkan kerugian jika tidak segera dilakukan pencegahan atau mitigasi karena menyebabkan layanan tidak dapat diakses dengan normal. Tidak hanya menyebabkan kerugian secara materi, tetapi juga non-materi. Kerugian secara materi dapat ditimbulkan karena terlanggarnya SLA (*Service Level Agreement*) atau batasan minimal layanan yang disediakan kepada pelanggannya. Hal tersebut dapat menyebabkan penyedia layanan harus membayar kompensasi atau ganti rugi kepada pengguna layanan. Kerugian secara non-materi dapat ditimbulkan karena reputasi yang kurang baik dari penyedia layanan, sehingga kurang dapat dipercaya oleh pengguna. Islam melarang perbuatan yang merugikan orang lain, salah satunya terdapat pada Al-Quran surah Al-Baqarah ayat 205.

وَاِذَا تَوَلّٰى سَعٰى فِى الْاَرْضِ لِيُفْسِدَ فِيْهَا وَيُهْلِكَ الْحَرْثَ وَالنَّسْلَۗ وَاللّٰهُ لَا يُحِبُّ الْفَسَادَ

"Dan apabila dia berpaling (dari engkau), dia berusaha untuk berbuat kerusakan di bumi, serta merusak tanam-tanaman dan ternak, sedang Allah tidak menyukai kerusakan." (QS. Al-Baqarah [2]:205)

Berdasarkan tafsir Kemenag, ayat tersebut menjelaskan bahwa orang yang berpaling dari ajaran Allah dan rasul-Nya akan berusaha merusak di muka bumi dengan menghancurkan tanaman dan ternak. Allah tidak menyukai perbuatan kerusakan tersebut, sehingga akan diberikan balasan setimpal terhadap orang-orang tersebut.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Penelitian ini berhasil membangun sistem deteksi serangan *Distributed Denial of Service* (DDoS) pada *Software Defined Network* (SDN) menggunakan metode *fuzzy* Tsukamoto. Berdasarkan penelitian yang telah dilakukan, dapat diambil kesimpulan sebagai berikut:

1. Deteksi serangan *Distributed Denial of Service* (DDoS) pada *Software Defined Network* (SDN) menggunakan metode *fuzzy* Tsukamoto dalam penelitian ini terdiri dari beberapa tahapan, yaitu pengumpulan data, penentuan semesta dan domain himpunan setiap variabel *fuzzy*, pembentukan aturan *fuzzy*, fuzzifikasi, inferensi, defuzzifikasi, keputusan hasil deteksi, dan evaluasi. Variabel yang digunakan dalam mendeteksi serangan DDoS pada SDN meliputi durasi, *flow*, *packet-in*, jumlah paket, jumlah *byte*, *packet rate*, dan *port bandwidth* sebagai variabel input dan variabel status sebagai variabel output.
2. Sistem deteksi DDoS pada SDN pada penelitian ini mampu mencapai akurasi deteksi sebesar 77,84%.

5.2 Saran

Berdasarkan penelitian yang telah dilakukan, berikut saran untuk penelitian selanjutnya:

1. Menggunakan parameter-parameter lain yang tersedia dalam dataset.

2. Menggunakan dataset dengan jumlah yang lebih banyak agar menghasilkan model sistem deteksi yang lebih optimal dan akurat.
3. Mengembangkan sistem deteksi menggunakan metode *fuzzy* yang lain seperti *fuzzy* mamdani dan sugeno.
4. Mengembangkan sistem mitigasi yang dapat secara otomatis menanggapi serangan DDoS yang terdeteksi.

DAFTAR PUSTAKA

Ahmed, K., Blech, J. O., Gregory, M. A., & Schmidt, H. W. (2018). Software defined networks in industrial automation. *Journal of Sensor and Actuator Networks*, *7*(3). https://doi.org/10.3390/jsan7030033

Ahuja, N., Singal, G., Mukhopadhyay, D., & Kumar, N. (2021). Automated DDOS attack detection in software defined networking. *Journal of Network and Computer Applications*, *187*. https://doi.org/10.1016/j.jnca.2021.103108

Al-Shaer, E., El-Atawy, A., & Samak, T. (2009). Automated pseudo-live testing of firewall configuration enforcement. *IEEE Journal on Selected Areas in Communications*, *27*(3).

Alsmadi, I., & Xu, D. (2015). Security of Software Defined Networks: A survey. *Computers and Security*, *53*, 79–108. https://doi.org/10.1016/j.cose.2015.05.006

Aprilianingsih, E. P., Primananda, R., & Suharsono, A. (2017). *Analisis Fail Path Pada Arsitektur Software Defined Network Menggunakan Dijkstra Algorithm* (Vol. 1, Issue 3). http://j-ptiik.ub.ac.id

Astuti, I. K. (2018). *Jaringan Komputer*.

Bakker, J. N. (2017). *Intelligent Traffic Classification for Detecting DDoS Attacks using SDN/OpenFlow*.

Bonato, J., Mrak, Z., & Badurina, M. (2015). Speed Regulation in Fan Rotation Using Fuzzy Inference System. In *Scientific Journal of Maritime Research* (Vol. 29).

Brikh, L., Guenounou, O., & Bakir, T. (2023). Selection of Minimum Rules from a Fuzzy TSK Model Using a PSO–FCM Combination. *Journal of Control, Automation and Electrical Systems*, *34*(2), 384–393. https://doi.org/10.1007/s40313-022-00975-2

De, V., Rios, M., Inácio, P. R. M., Magoni, D., & Freire, M. M. (2021). *Detection of reduction-of-quality DDoS attacks using Fuzzy Logic and machine learning algorithms*. 186. https://doi.org/10.1016/j.comnet.2020.107792ï

Fitriah, A., & Maman, A. A. (2011). *Aplikasi Model Neuro Fuzzy Untuk Prediksi Tingkat Inflasi Di Indosenia*.

Harto, M. K., & Basuki, A. (2021). *Deteksi Serangan DDoS Pada Jaringan Berbasis SDN Dengan Klasifikasi Random Forest* (Vol. 5, Issue 4). http://j-ptiik.ub.ac.id

Hermawan, R. (2012). *ANALISIS KONSEP DAN CARA KERJA SERANGAN KOMPUTER DISTRIBUTED DENIAL OF SERVICE (DDOS)* (Vol. 5, Issue 1).

Hussain, A. (2018). USE OF FIREWALL AND IDS TO DETECT AND PREVENT NETWORK ATTACKS. *International Journal of Technical Research & Science*, *3*(IX). https://doi.org/10.30780/ijtrs.v3.i9.2018.002

Kandoi, R., & Antikainen, M. (2015, May). Denial-of-service attacks in OpenFlow SDN networks. *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*.

Kreutz, D., Ramos, F. M. V., Verissimo, P., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S. (2014). *Software-Defined Networking: A Comprehensive Survey*. http://arxiv.org/abs/1406.0440

Kusumadewi, S., & Purnomo, H. (2004). *Aplikasi Logika Fuzzy untuk Pendukung Keputusan* (1st ed.). Graha Ilmu.

Mardiana, Y., & Sahputra, J. (2017). Analisa Performansi Protokol TCP, UDP dan SCTP Pada Lalu Lintas Multimedia. *Jurnal Media Infotama*, *13*(2).

Maseleno, A., & Hasan, Md. M. (2015). Finding Kicking Range of Sepak Takraw Game: A Fuzzy Logic Approach. *TELKOMNIKA Indonesian Journal of Electrical Engineering*, *14*(3). https://doi.org/10.11591/telkomnika.v14i3.7833

Mulyanta, E. S. (2005). *Pengenalan Protokol Jaringan Wireless Komputer*.

Prakasa, J. E. W. (2020). Peningkatan Keamanan Sistem Informasi Melalui Klasifikasi Serangan Terhadap Sistem Informasi. *Jurnal Ilmiah Teknologi Informasi Asia*, *14*(2), 75–84. https://doi.org/10.32815/jitika.v14i2.452

Scaranti, G. F., Carvalho, L. F., Barbon, S., & Proenca, M. L. (2020). Artificial Immune Systems and Fuzzy Logic to Detect Flooding Attacks in Software-Defined Networks. *IEEE Access*, *8*, 100172–100184. https://doi.org/10.1109/ACCESS.2020.2997939

Shaghaghi, A., Kaafar, M. A., Buyya, R., & Jha, S. (2018). *Software-Defined Network (SDN) Data Plane Security: Issues, Solutions and Future Directions*. http://arxiv.org/abs/1804.00262

Specht, S., & Lee, R. (2003). *Taxonomies of Distributed Denial of Service Networks, Attacks, Tools, and Countermeasures*. www.princeton.edu

Sugianti, N., Galuh, Y., Fatia, S., Fahmi, K., & Holle, H. (2020). Deteksi Serangan Distributed Denial of Services (DDOS) Berbasis HTTP Menggunakan Metode Fuzzy Sugeno. In *JANUARI* (Vol. 4, Issue 3).

Tantriawan, H., Agung Yunmar, R., Setiawan, A., & Suryadi, M. (2021). *Detection Distributed Denial of Service (DDoS) Using Sugeno's Fuzzy Logic Deteksi Distributed Denial of Service (DDoS) Menggunakan Fuzzy Logic Sugeno*. *1*, 144–154.

Turban, E., Aronson, J. E., & Liang, T.-P. (2005). *Decision Support Systems and Intelligent Systems*.

LAMPIRAN

Lampiran I. Kode Program

Training

```
# import library
import numpy as np
import pandas as pd
import skfuzzy as fuzz
from skfuzzy import control as ctrl
import matplotlib.pyplot as plt
from sklearn import metrics

# import data
from google.colab import files
uploaded = files.upload()

datasett = pd.read_csv("dataset_sdna.csv")
dataset=datasett.dropna()

datas = dataset.copy()
datas2 = datas[['dur', 'flows', 'packetins', 'pktperflow', 'byteperflow',
'pktrate', 'tot_kbps', 'label']]
duplicate_mask = datas2.duplicated()
duplicates = datas2[duplicate_mask]

data = datas2.drop_duplicates()

def identify_outliers(data):
    q1, q3 = data.quantile(0.25), data.quantile(0.75)
    iqr = q3 - q1
    cut_off = iqr * 1.5
    data_clean = data[~((data < (q1 - cut_off)) |(data > (q3 +
cut_off))).any(axis=1)]
    return data_clean
data = identify_outliers(data)

#menentukan nilai maks dan minim
def get_range(data):
    data_range = pd.DataFrame(columns = data.columns, index = ['max',
'min'])
    data_range.loc['max'] = [data[col].max() for col in data.columns]
    data_range.loc['min'] = [data[col].min() for col in data.columns]
    return data_range

range_df = get_range(data)
```

```
#menentukan semesta pembicara
universe = []
for col in range_df.columns:
        universe_col = np.arange(np.floor(range_df[col].min()),
np.ceil(range_df[col].max()+1))
        universe.append(universe_col)
#menentukan median
def med(l, u):
    return np.median(np.arange(l, u))

# inisiasi fungsi keanggotaan variabel
# masukkan q1, q2, dan q3
dura_lo  = fuzz.trapmf(universe[0], [0, 0, 109, 190])
dura_mid = fuzz.trimf(universe[0], [109, 190, 270])
dura_hi  = fuzz.trapmf(universe[0], [190, 270, 477, 477])

flow_lo  = fuzz.trapmf(universe[1], [0, 0, 3, 4])
flow_mid = fuzz.trimf(universe[1], [3, 4, 5])
flow_hi  = fuzz.trapmf(universe[1], [4, 5, 8, 8])

pktin_lo  = fuzz.trapmf(universe[2], [0, 0, 1931, 1943])
pktin_mid = fuzz.trimf(universe[2], [1931, 1943, 7503])
pktin_hi  = fuzz.trapmf(universe[2], [1943, 7503, 8803, 8803])

jpkt_lo  = fuzz.trapmf(universe[3], [0, 0, 9258, 13385])
jpkt_mid = fuzz.trimf(universe[3], [9258, 13385, 13529])
jpkt_hi  = fuzz.trapmf(universe[3], [13385, 13529, 13685, 13685])

jbyte_lo  = fuzz.trapmf(universe[4], [0, 0, 9646836, 14268410])
jbyte_mid = fuzz.trimf(universe[4], [9646836, 14268410, 14421914])
jbyte_hi  = fuzz.trapmf(universe[4], [14268410, 14421914, 14588210,
14588210])

rt_lo  = fuzz.trapmf(universe[5], [0, 0, 308, 446])
rt_mid = fuzz.trimf(universe[5], [308, 446, 450])
rt_hi  = fuzz.trapmf(universe[5], [446, 450, 456, 456])

totkbps_lo  = fuzz.trapmf(universe[6], [0, 0, 2385, 3838])
totkbps_mid = fuzz.trimf(universe[6], [2385, 3838, 6515])
totkbps_hi  = fuzz.trapmf(universe[6], [3838, 6515, 14993, 14993])

lbel_lo  = fuzz.trimf(universe[7], [0, 0, med(0,1)])
lbel_mid = fuzz.trimf(universe[7], [0, med(0,1), 1])
lbel_hi  = fuzz.trimf(universe[7], [med(0,1), 1, 1])
```

```
# Pembentukan rule
def fuzzify_dura(dura_val):
    dura_df = pd.DataFrame(dura_val)
    dura_df['low'] = fuzz.interp_membership(universe[0], dura_lo,
dura_val)
    dura_df['mid'] = fuzz.interp_membership(universe[0], dura_mid,
dura_val)
    dura_df['high'] = fuzz.interp_membership(universe[0], dura_hi,
dura_val)

    dura_df['membership'] = dura_df.loc[:, ['low', 'mid',
'high']].idxmax(axis = 1)
    dura_df['degree'] = dura_df.loc[:, ['low', 'mid', 'high']].max(axis =
1)
    return dura_df

dura_val = train_data.iloc[:,0]
dura_df = fuzzify_dura(dura_val)

def fuzzify_flow(flow_val):
    flow_df = pd.DataFrame(flow_val)
    flow_df['low'] = fuzz.interp_membership(universe[1], flow_lo,
flow_val)
    flow_df['mid'] = fuzz.interp_membership(universe[1], flow_mid,
flow_val)
    flow_df['high'] = fuzz.interp_membership(universe[1], flow_hi,
flow_val)

    flow_df['membership'] = flow_df.loc[:, ['low', 'mid',
'high']].idxmax(axis = 1)
    flow_df['degree'] = flow_df.loc[:, ['low', 'mid', 'high']].max(axis =
1)
    return flow_df

flow_val = train_data.iloc[:,1]
flow_df = fuzzify_flow(flow_val)

def fuzzify_pktin(pktin_val):
    pktin_df = pd.DataFrame(pktin_val)
    pktin_df['low'] = fuzz.interp_membership(universe[2], pktin_lo,
pktin_val)
    pktin_df['mid'] = fuzz.interp_membership(universe[2], pktin_mid,
pktin_val)
```

```
    pktin_df['high'] = fuzz.interp_membership(universe[2], pktin_hi,
pktin_val)

    pktin_df['membership'] = pktin_df.loc[:, ['low', 'mid',
'high']].idxmax(axis = 1)
    pktin_df['degree'] = pktin_df.loc[:, ['low', 'mid', 'high']].max(axis
= 1)
    return pktin_df

pktin_val = train_data.iloc[:,2]
pktin_df = fuzzify_pktin(pktin_val)

def fuzzify_jpkt(jpkt_val):
    jpkt_df = pd.DataFrame(jpkt_val)
    jpkt_df['low'] = fuzz.interp_membership(universe[3], jpkt_lo,
jpkt_val)
    jpkt_df['mid'] = fuzz.interp_membership(universe[3], jpkt_mid,
jpkt_val)
    jpkt_df['high'] = fuzz.interp_membership(universe[3], jpkt_hi,
jpkt_val)

    jpkt_df['membership'] = jpkt_df.loc[:, ['low', 'mid',
'high']].idxmax(axis = 1)
    jpkt_df['degree'] = jpkt_df.loc[:, ['low', 'mid', 'high']].max(axis =
1)
    return jpkt_df

jpkt_val = train_data.iloc[:,3]
jpkt_df = fuzzify_jpkt(jpkt_val)

def fuzzify_jbyte(jbyte_val):
    jbyte_df = pd.DataFrame(jbyte_val)
    jbyte_df['low'] = fuzz.interp_membership(universe[4], jbyte_lo,
jbyte_val)
    jbyte_df['mid'] = fuzz.interp_membership(universe[4], jbyte_mid,
jbyte_val)
    jbyte_df['high'] = fuzz.interp_membership(universe[4], jbyte_hi,
jbyte_val)

    jbyte_df['membership'] = jbyte_df.loc[:, ['low', 'mid',
'high']].idxmax(axis = 1)
    jbyte_df['degree'] = jbyte_df.loc[:, ['low', 'mid', 'high']].max(axis
= 1)
    return jbyte_df
```

```
jbyte_val = train_data.iloc[:,4]
jbyte_df = fuzzify_jbyte(jbyte_val)

def fuzzify_rt(rt_val):
    rt_df = pd.DataFrame(rt_val)
    rt_df['low'] = fuzz.interp_membership(universe[5], rt_lo, rt_val)
    rt_df['mid'] = fuzz.interp_membership(universe[5], rt_mid, rt_val)
    rt_df['high'] = fuzz.interp_membership(universe[5], rt_hi, rt_val)

    rt_df['membership'] = rt_df.loc[:, ['low', 'mid',
'high']].idxmax(axis = 1)
    rt_df['degree'] = rt_df.loc[:, ['low', 'mid', 'high']].max(axis = 1)
    return rt_df

rt_val = train_data.iloc[:,5]
rt_df = fuzzify_rt(rt_val)

def fuzzify_totkbps(totkbps_val):
    totkbps_df = pd.DataFrame(totkbps_val)
    totkbps_df['low'] = fuzz.interp_membership(universe[6], totkbps_lo,
totkbps_val)
    totkbps_df['mid'] = fuzz.interp_membership(universe[6], totkbps_mid,
totkbps_val)
    totkbps_df['high'] = fuzz.interp_membership(universe[6], totkbps_hi,
totkbps_val)

    totkbps_df['membership'] = totkbps_df.loc[:, ['low', 'mid',
'high']].idxmax(axis = 1)
    totkbps_df['degree'] = totkbps_df.loc[:, ['low', 'mid',
'high']].max(axis = 1)
    return totkbps_df

totkbps_val = train_data.iloc[:,6]
totkbps_df = fuzzify_totkbps(totkbps_val)

def fuzzify_lbel(lbel_val):
    lbel_df = pd.DataFrame(lbel_val)
    lbel_df['low'] = fuzz.interp_membership(universe[7], lbel_lo,
lbel_val)
    lbel_df['mid'] = fuzz.interp_membership(universe[7], lbel_mid,
lbel_val)
    lbel_df['high'] = fuzz.interp_membership(universe[7], lbel_hi,
lbel_val)
```

```
    lbel_df['membership'] = lbel_df.loc[:, ['low', 'mid',
'high']].idxmax(axis = 1)
    lbel_df['degree'] = lbel_df.loc[:, ['low', 'mid', 'high']].max(axis =
1)
    return lbel_df

lbel_val = train_data.iloc[:,7]
lbel_df = fuzzify_lbel(lbel_val)

# menggabungkan kolom keanggotaan tiap variabel untuk membentuk rule
fuzzy
def merge_rule(df, dura_df, flow_df, pktin_df, jpkt_df, jbyte_df, rt_df,
totkbps_df, lbel_df):
    rule_df = df.copy()
    rule_df['dur'] = dura_df['membership']
    rule_df['flows'] = flow_df['membership']
    rule_df['packetins'] = pktin_df['membership']
    rule_df['pktperflow'] = jpkt_df['membership']
    rule_df['byteperflow'] = jbyte_df['membership']
    rule_df['pktrate'] = rt_df['membership']
    rule_df['tot_kbps'] = totkbps_df['membership']
    rule_df['label'] = lbel_df['membership']
    rule_df['degree'] =
dura_df['degree']*flow_df['degree']*pktin_df['degree']*jpkt_df['degree']*
jbyte_df['degree']*rt_df['degree']*totkbps_df['degree']*lbel_df['degree']
    return rule_df

rule_df = merge_rule(train_data, dura_df, flow_df, pktin_df, jpkt_df,
jbyte_df, rt_df, totkbps_df, lbel_df)

rule_df.drop_duplicates(inplace=True)
rule_df.reset_index(drop=True, inplace=True)

rule_fuzzy = rule_df.groupby(['dur', 'flows', 'packetins', 'pktperflow',
'byteperflow', 'pktrate', 'tot_kbps']).max()
rule_fuzzy = rule_fuzzy.reset_index()
rule_fuzzy_deg = rule_fuzzy.pop('degree')
rule_fuzzy
```

Testing

```
# inisiasi fungsi keanggotaan
xdura_lo  = fuzz.trapmf(universe[0], [0, 0, 108, 190])
xdura_mid = fuzz.trimf(universe[0], [108, 190, 270])
xdura_hi  = fuzz.trapmf(universe[0], [190, 270, 477, 477])

xflow_lo  = fuzz.trapmf(universe[1], [0, 0, 3, 4])
xflow_mid = fuzz.trimf(universe[1], [3, 4, 5])
xflow_hi  = fuzz.trapmf(universe[1], [4, 5, 8, 8])

xpktin_lo  = fuzz.trapmf(universe[2], [0, 0, 1931, 1943])
xpktin_mid = fuzz.trimf(universe[2], [1931, 1943, 7503])
xpktin_hi  = fuzz.trapmf(universe[2], [1943, 7503, 8803, 8803])

xjpkt_lo  = fuzz.trapmf(universe[3], [0, 0, 9269, 13385])
xjpkt_mid = fuzz.trimf(universe[3], [9269, 13385, 13529])
xjpkt_hi  = fuzz.trapmf(universe[3], [13385, 13529, 13685, 13685])

xjbyte_lo  = fuzz.trapmf(universe[4], [0, 0, 9658298, 14268410])
xjbyte_mid = fuzz.trimf(universe[4], [9658298, 14268410, 14421914])
xjbyte_hi  = fuzz.trapmf(universe[4], [14268410, 14421914, 14588210,
14588210])

xrt_lo  = fuzz.trapmf(universe[5], [0, 0, 308, 446])
xrt_mid = fuzz.trimf(universe[5], [308, 446, 450])
xrt_hi  = fuzz.trapmf(universe[5], [446, 450, 456, 456])

xtotkbps_lo  = fuzz.trapmf(universe[6], [0, 0, 2462, 3838])
xtotkbps_mid = fuzz.trimf(universe[6], [2462, 3838, 6515])
xtotkbps_hi  = fuzz.trapmf(universe[6], [3838, 6515, 14993, 14993])
```

```
inp1, inp2, inp3, inp4, inp5, inp6, inp7 = [],[],[],[],[],[],[]

## Rules dan inferensi
# normal = 0.6 - (apred)*0.2
# ddos = 0.2*apred + 0.4

apred1,apred2,apred3,apred4,apred5,..apredn =[],[],[],[],[],[]

z1,z2,z3,z4,z5,..zn =[],[],[],[],[],[]

z = []
put = []
lb_deteksi = []
```

```
lb_pred = []
lb_aktual = []

from sklearn.model_selection import KFold

# inisiasi fold
k = 5

# data dan variabel target
x = data.drop('label', axis=1)
y = data['label']

# inisialisasi k-fold cross-validation
kf = KFold(n_splits=k, shuffle=True, random_state=42)

cnt = 1
# perulangan folds
for train_index, test_index in kf.split(x):
  x_train, x_test = x.iloc[train_index], x.iloc[test_index]
  y_train, y_test = y.iloc[train_index], y.iloc[test_index]
  print(f'Fold:{cnt}, Train set: {len(train_index)}, Test
set:{len(test_index)}')
  cnt+=1


  x_test = x_test.rename(columns={'dur':'durasi', 'flows':'flow',
'packetins':'packet_in', 'pktperflow':'jumlah paket',
'byteperflow':'jumlah byte', 'pktrate':'packet rate', 'tot_kbps':'port
bandwidth'})
  x_test.reset_index(drop=True, inplace=True)
  y_test.reset_index(drop=True, inplace=True)

  test_dura_df = x_test.iloc[:,0]

  test_flow_df = x_test.iloc[:,1]

  test_pktin_df = x_test.iloc[:,2]

  test_jpkt_df = x_test.iloc[:,3]

  test_jbyte_df = x_test.iloc[:,4]

  test_rt_df = x_test.iloc[:,5]
```

```
  test_totkbps_df = x_test.iloc[:,6]

  #reset
  inp1, inp2, inp3, inp4, inp5, inp6, inp7 = [],[],[],[],[],[],[]
  apred1,apred2,apred3,apred4,apred5,..apredn =[],[],[],[],[],[]
  z1,z2,z3,z4,z5,..zn =[],[],[],[],[],[]

  z = []
  put = []
  lb_deteksi = []

  len(x_test)

  for i in range(len(x_test)):

    ## fuzzifikasi Tsukamoto low(0), mid(1), high(2)
    inp1.append([fuzz.interp_membership(universe[0], xdura_lo,
test_dura_df[i]), fuzz.interp_membership(universe[0], xdura_mid,
test_dura_df[i]), fuzz.interp_membership(universe[0], xdura_hi,
test_dura_df[i])])

    inp2.append([fuzz.interp_membership(universe[1], xflow_lo,
test_flow_df[i]), fuzz.interp_membership(universe[1], xflow_mid,
test_flow_df[i]), fuzz.interp_membership(universe[1], xflow_hi,
test_flow_df[i])])

    inp3.append([fuzz.interp_membership(universe[2], xpktin_lo,
test_pktin_df[i]), fuzz.interp_membership(universe[2], xpktin_mid,
test_pktin_df[i]), fuzz.interp_membership(universe[2], xpktin_hi,
test_pktin_df[i])])

    inp4.append([fuzz.interp_membership(universe[3], xjpkt_lo,
test_jpkt_df[i]), fuzz.interp_membership(universe[3], xjpkt_mid,
test_jpkt_df[i]), fuzz.interp_membership(universe[3], xjpkt_hi,
test_jpkt_df[i])])

    inp5.append([fuzz.interp_membership(universe[4], xjbyte_lo,
test_jbyte_df[i]), fuzz.interp_membership(universe[4], xjbyte_mid,
test_jbyte_df[i]), fuzz.interp_membership(universe[4], xjbyte_hi,
test_jbyte_df[i])])

    inp6.append([fuzz.interp_membership(universe[5], xrt_lo,
test_rt_df[i]), fuzz.interp_membership(universe[5], xrt_mid,
test_rt_df[i]), fuzz.interp_membership(universe[5], xrt_hi,
test_rt_df[i])])
```

```
    inp7.append([fuzz.interp_membership(universe[6], xtotkbps_lo,
test_totkbps_df[i]), fuzz.interp_membership(universe[6], xtotkbps_mid,
test_totkbps_df[i]), fuzz.interp_membership(universe[6], xtotkbps_hi,
test_totkbps_df[i])])


## inferensi
    apred1.append(np.fmin(np.fmin(np.fmin(inp1[i][2], inp2[i][2]),
np.fmin(inp3[i][2], inp4[i][2])), np.fmin(np.fmin(inp5[i][2],
inp6[i][2]), inp7[i][2])))
    z1.append(0.6 - (apred1[i]*0.2))

    apred2.append(np.fmin(np.fmin(np.fmin(inp1[i][2], inp2[i][2]),
np.fmin(inp3[i][2], inp4[i][2])), np.fmin(np.fmin(inp5[i][2],
inp6[i][2]), inp7[i][0])))
    z2.append(0.6 - (apred2[i]*0.2))

    apred3.append(np.fmin(np.fmin(np.fmin(inp1[i][2], inp2[i][2]),
np.fmin(inp3[i][2], inp4[i][2])), np.fmin(np.fmin(inp5[i][2],
inp6[i][2]), inp7[i][1])))
    z3.append(0.6 - (apred3[i]*0.2))

    apred4.append(np.fmin(np.fmin(np.fmin(inp1[i][2], inp2[i][2]),
np.fmin(inp3[i][2], inp4[i][0])), np.fmin(np.fmin(inp5[i][0],
inp6[i][0]), inp7[i][2])))
    z4.append((0.2*apred4[i]) + 0.4)

    apred5.append(np.fmin(np.fmin(np.fmin(inp1[i][2], inp2[i][2]),
np.fmin(inp3[i][2], inp4[i][0])), np.fmin(np.fmin(inp5[i][0],
inp6[i][0]), inp7[i][1])))
    z5.append((0.2*apred5[i]) + 0.4)
                                    .
                                    .
                                    .
       apredke-n

# Defuzifikasi
    z.append((apred1[i]*z1[i] + apred2[i]*z2[i] + apred3[i]*z3[i] +
apred4[i]*z4[i] + apred5[i]*z5[i] + apredn[i]*zn[i]/(apred1[i] +
apred2[i] + apred3[i] + apred4[i] + apred5[i] + apredn[i]))

  # fuzzifikasi nilai output(crisp)
    member_normal = (0.6 - z[i]) / 0.2
    member_ddos   = (z[i] - 0.4) / 0.2
```

```
  # keputusan deteksi
    if member_normal > member_ddos:
      lb_deteksi.append(0)
      put.append("Normal")
    else:
      lb_deteksi.append(1)
      put.append("DDoS")

  lb_aktual.append(y_test.iloc[:])
  lb_pred.append(lb_deteksi)

# akurasi
all_akurasi = []
for j in range(5):
  y = lb_aktual[j]
  y_pred = lb_pred[j]
  confusion_matrix = metrics.confusion_matrix(y, y_pred)

  cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix =
confusion_matrix, display_labels = [0, 1])

  cm_display.plot()
  plt.show()

  Accuracy = metrics.accuracy_score(y, y_pred)
  akurasi = Accuracy*100
  all_akurasi.append(akurasi)
  print(Accuracy)
  print("Akurasi :",akurasi,"%")
```