INVESTIGASI LOG JARINGAN UNTUK DETEKSI SERANGAN *DISTRIBUTED DENIAL OF SERVICE* (DDOS) DENGAN MENGGUNAKAN METODE *GENERAL REGRESSION NEURAL NETWORK*

SKRIPSI

Oleh:
MUHAMMAD HILMI HAFID
NIM. 14650011

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2019

INVESTIGASI LOG JARINGAN UNTUK DETEKSI SERANGAN *DISTRIBUTED DENIAL OF SERVICE* (DDOS) DENGAN MENGGUNAKAN METODE *GENERAL REGRESSION NEURAL NETWORK*

SKRIPSI

Diajukan kepada:
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang
Untuk Memenuhi Salah Satu Persyaratan Dalam
Memperoleh Gelar Sarjana Komputer (S.Kom)

Oleh:
MUHAMMAD HILMI HAFID
NIM. 14650011

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM MALANG
2019

INVESTIGASI LOG JARINGAN UNTUK DETEKSI SERANGAN *DISTRIBUTED DENIAL OF SERVICE* (DDOS) DENGAN MENGGUNAKAN METODE *GENERAL REGRESSION NEURAL NETWORK*

SKRIPSI

Oleh:
MUHAMMAD HILMI HAFID
NIM. 14650011

Telah Diperiksa dan Disetujui untuk Diuji
Tanggal: 29 November 2019

Dosen Pembimbing I	Dosen Pembimbing II
Dr. M. Amin Hariyadi, M.T.	A'la Syauqi, M.Kom.
NIP. 19670118 200501 1 001	NIP. 19771201 200801 1 007

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

Dr. Cahyo Crysdian
NIP. 19740424 200901 1 008

INVESTIGASI LOG JARINGAN UNTUK DETEKSI SERANGAN *DISTRIBUTED DENIAL OF SERVICE* (DDOS) DENGAN MENGGUNAKAN METODE *GENERAL REGRESSION NEURAL NETWORK*

SKRIPSI

Oleh:
MUHAMMAD HILMI HAFID
NIM. 14650011

Telah Dipertahankan di Depan Dewan Penguji Skripsi
dan Dinyatakan Diterima sebagai Salah Satu Persyaratan
untuk Memperoleh Gelar Sarjana Komputer (S.Kom)
Tanggal: 23 Desember 2019

Susunan Dewan Penguji :			**Tanda Tangan**
Penguji Utama	:	Dr. Muhammad Faisal, M.T. NIP. 19740510 200501 1 007	( )
Ketua Penguji	:	Roro Inda Melani, S.Kom., M.Sc. NIP. 19780925 200501 2 008	( )
Sekertaris Penguji	:	Dr. M. Amin Hariyadi, M.T. NIP. 19670118 200501 1 001	( )
Anggota Penguji	:	A'la Syauqi, M.Kom.. NIP. 19771201 200801 1 007	( )

Mengetahui dan Mengesahkan,
Ketua Jurusan Teknik Informatika

Dr. Cahyo Crysdian
NIP. 19740424 200901 1 008

PERNYATAAN KEASLIAN TULISAN

Saya yang bertanda tangan di bawah ini,

Nama : Muhammad Hilmi Hafid

NIM : 14650011

Jurusan : Teknik Informatika

Fakultas : Sains dan Teknologi

menyatakan dengan sebenarnya bahwa skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri, bukan merupakan pengambilan 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 di kemudian hari terbukti atau dapat dibuktikan skripsi ini hasil jiplakan, maka saya bersedia menerima sanksi atas perbuatan saya tersebut.

Malang, 23 Desember 2019

Yang membuat pernyataan,

Muhammad Hilmi Hafid

NIM.14650011

MOTTO

"Work hard, trust God"

PERSEMBAHAN

Alhamdulillaah, puji syukur ke hadirat Allah yang telah memberikan kekuatan dan semangat tiada henti untuk menyelesaikan kewajiban belajar saya di kampus ini. Dengan ini saya persembahkan karya sederhana ini untuk:

Ayahanda Abdul Hafid dan Ibunda Siti Hartini yang senantiasa mengingatkan, mendoakan, dan memberikan dukungan moril maupun materil yang takkan pernah bisa terbalaskan.

Kakak Faizah dan Fauziyah yang selalu mengingatkan, menyemangati, dan menanyakan kapan skripsi ini selesai.

Keluarga besar tercinta lainnya yang setia menanti kepulanganku ke kampung halaman.

Squad Buk Lien, tim Asosiasi Aku Indonesia, teman-teman Biner 14, teman-teman IKA M2M Malang, teman-teman IKAMI SULSELBAR, dan teman-teman seperjuangan yang selalu ada di saat senang maupun susah.

Terkhusus Rina, Kak Hasbi, Mas Jefri, Kak Fadhil, Pepenk, Mohmud, Walax yang selalu menghibur dan bersedia direpotkan jika butuh bantuan.

Sahabat-sahabat dan teman-teman lainnya, serta siapapun yang berjasa dalam penyelesaian skripsi ini secara langsung maupun tidak langsung.

Terima kasih untuk semuanya, semoga Allah memberikan balasan yang terbaik. *Aamiin Allahumma aamiin*

KATA PENGANTAR

Assalamu'alaikum Wr. Wb.

Segala puji bagi Allah, Tuhan semesta alam yang telah memberikan rahmat dan karunia-Nya kepada penulis sehingga skripsi ini yang berjudul "**Investigasi Log Jaringan untuk Deteksi Serangan *Distributed Denial of Service* (DDoS) dengan Menggunakan Metode *General Regression Neural Network***" dapat diselesaikan dengan baik. Shalawat dan salam senantiasa tercurah kepada Rasulullah yang mengantarkan manusia dari zaman kegelapan ke zaman yang terang benderang ini.

Penyusunan skripsi ini dimaksudkan untuk memenuhi sebagian syarat-syarat guna mencapai gelar Sarjana Komputer di Universitas Islam Negeri Maulana Malik Ibrahim Malang. Penulis menyadari bahwa penulisan ini tidak dapat terselesaikan tanpa dukungan dari berbagai pihak baik moril maupun materiil. Oleh karena itu, penulis ingin menyampaikan ucapan terima kasih kepada semua pihak yang telah banyak membantu dalam penyusunan skripsi ini terutama kepada:

1. Prof. Dr. Abdul Haris, M.Ag., selaku Rektor Universitas Islam Negeri Maulana Malik Ibrahim Malang.
2. Dr. Sri Harini, M.Si, selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.
3. Dr. Cahyo Crysdian, selaku Ketua Jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.
4. Fatchurrahman, M.Kom., selaku dosen wali saya.

5. Dr. M. Amin Hariyadi, M.T. dan A'la Syauqi, M.Kom., selaku dosen pembimbing skripsi saya.
6. Segenap sivitas akademika jurusan Teknik Informatika, terutama seluruh dosen yang tidak dapat saya sebutkan satu per satu di halaman ini.
7. Ayahanda, Ibunda, kakak dan adik penulis, serta keluarga besar tercinta lainnya.
8. Sahabat-sahabat dan teman-teman seperjuangan di perantauan.
9. Semua pihak yang berjasa dalam penyelesaian skripsi ini.

Penulis menyadari bahwa skripsi ini masih jauh dari sempurna dikarenakan terbatasnya pengalaman dan pengetahuan yang dimiliki penulis. Oleh karena itu, penulis mengharapkan segala bentuk saran serta masukan bahkan kritik yang membangun dari berbagai pihak. Semoga skripsi ini dapat bermanfaat bagi para pembaca.

Wassalamu'alaikum Wr. Wb.

Malang, 23 Desember 2019
Penulis,

Muhammad Hilmi Hafid

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Hafid, Muhammad Hilmi. 2019. **Investigasi Log Jaringan untuk Deteksi Serangan Distributed Denial of Service (DDoS) dengan Menggunakan Metode *General Regression Neural Network***. Skripsi. Jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri (UIN) Maulana Malik Ibrahim Malang.
Pembimbing: (I) Dr. M. Amin Hariyadi, M.T. (II) A'la Syauqi, M.Kom.

Kata kunci: forensik jaringan, sistem deteksi intrusi, *machine learning, distributed denial of service*, *general regression neural network*, *random forest*.

Salah satu jenis serangan di dunia maya dengan intensitas yang cukup besar yaitu serangan *Distributed Denial of Service* (DDoS). Dibutuhkan sistem deteksi intrusi yang efektif dan akurat dalam mendeteksi serangan pada data intrusi jaringan untuk mengatasi hal tersebut. Oleh sebab itu, penelitian ini bertujuan untuk mengimplementasikan pendekatan baru dalam mendeteksi serangan pada data intrusi jaringan dengan tingkat akurasi yang baik. Metode yang diusulkan yaitu *General Regression Neural Network* yang dibantu dengan *Random Forest* dalam menyeleksi fitur sehingga mampu meningkatkan akurasi deteksi dan mempercepat waktu komputasi. Data latih yang digunakan yaitu CICIDS2017 dari Canadian Institute for Cybersecurity, sedangkan data uji yang digunakan yaitu log jaringan yang didapatkan dari simulasi serangan DDoS pada server web. Percobaan pertama menggunakan 69 fitur, diperoleh tingkat akurasi sebesar 66,41% dengan waktu pelatihan selama 1 jam 45 menit 6 detik. Adapun percobaan kedua menggunakan fitur terpilih yaitu sebanyak 20 fitur, diperoleh tingkat akurasi sebesar 97,21% dengan waktu pelatihan selama 42 menit 27 detik. Dari hasil percobaan tersebut, dapat disimpulkan bahwa *General Regression Neural Network* memiliki kemampuan deteksi dan klasifikasi yang cukup baik terhadap serangan DDoS pada data intrusi jaringan.

ABSTRACT

Hafid, Muhammad Hilmi. 2019. **Network Log Investigation to Detect Distributed Denial of Service (DDoS) Attack Using General Regression Neural Network**. Undergraduate Thesis. Department of Informatics Engineering, Faculty of Science and Technology, Maulana Malik Ibrahim State Islamic University of Malang.
Supervisor : (I) Dr. M. Amin Hariyadi, M.T. (II) A'la Syauqi, M.Kom.

One type of attack in cyberspace with a large enough intensity is the Distributed Denial of Service (DDoS) attack. To overcome this, an effective and accurate intrusion detection system needed to detect attacks on network intrusion data. Therefore, this study aims to implement a new approach in detecting attacks on network intrusion data with a good rate of accuracy. The proposed method is the General Regression Neural Network which is assisted by Random Forest in selecting features so as to improve detection accuracy and speed up computing time. The training data used is CICIDS2017 from the Canadian Institute for Cybersecurity, while the test data used are network logs obtained from a simulation of DDoS attacks on a web server. The first experiment using 69 features, obtained an accuracy rate of 66.41% with training time for 1 hour 45 minutes 6 seconds. As for the second experiment using selected features which is 20 features, obtained an accuracy rate of 97.21% with training time for 42 minutes 27 seconds. From the results of these experiments, it can be concluded that the General Regression Neural Network has a fairly good detection and classification ability against DDoS attacks on network intrusion data.

Keywords: network forensics, intrusion detection system, machine learning, distributed denial of service, general regression neural network, random forest.

خلاصة

حفيظ، محمد حلمي. 2019. **التحقيق في سجل الشبكة للكشف عن هجو Distributed Denial of Service باستخدام General Regression Neural Network**. أطروحة جامعية. قسم هندسة المعلوماتية. كلية العلوم والتكنولوجيا. مولانا مالك إبراهيم جامعة ولاية مالانج الإسلامية.

مشرف: (I) الدكتور محمد أمين هريدي الماجيستير (II) علا سيوكي الماجيستير

نوع الهجوم في الفضاء السيبراني بكثافة كبيرة بما فيه الكفاية هو الهجوم Distributed Denial of Service. للتغلب على هذا، هناك حاجة إلى نظام كشف اقتحام فعال ودقيق لاكتشاف الهجمات على بيانات اقتحام الشبكة. لذلك، تهدف هذه الدراسة إلى تطبيق نهج جديد في الكشف عن الهجمات على بيانات اقتحام الشبكة بمعدل دقة جيد. الطريقة المقترحة هي General Regression Neural Network التي تساعدها Random Forest في اختيار الميزات لتحسين دقة الكشف وتسريع وقت الحوسبة. بيانات التدريب المستخدمة هي CICIDS2017 من المعهد الكندي للأمن السيبراني، في حين أن بيانات الاختبار المستخدمة هي سجلات شبكة تم الحصول عليها من محاكاة هجمات Distributed Denial of Service على خادم ويب. أول تجربة باستخدام 69 ميزة ، حصلت على معدل دقة 66.41٪ مع وقت التدريب لمدة 1 ساعة 45 دقيقة 6 ثانية. أما بالنسبة للتجربة الثانية باستخدام الميزات المحددة وهي 20 ميزة ، فقد حصلت على معدل دقة 97.21٪ مع وقت تدريب لمدة 42 دقيقة و 27 ثانية. من نتائج هذه التجارب ، يمكن أن نستنتج أن General Regression Neural Network لديها قدرة جيدة على الكشف والتصنيف ضد هجمات Distributed Denial of Service على بيانات اقتحام الشبكة.

الكلمات المفتاحية :الطب الشرعي للشبكة، نظام كشف التسلل، التعلم الآلي،
distributed denial of service, general regression neural network, random forest.

BAB I

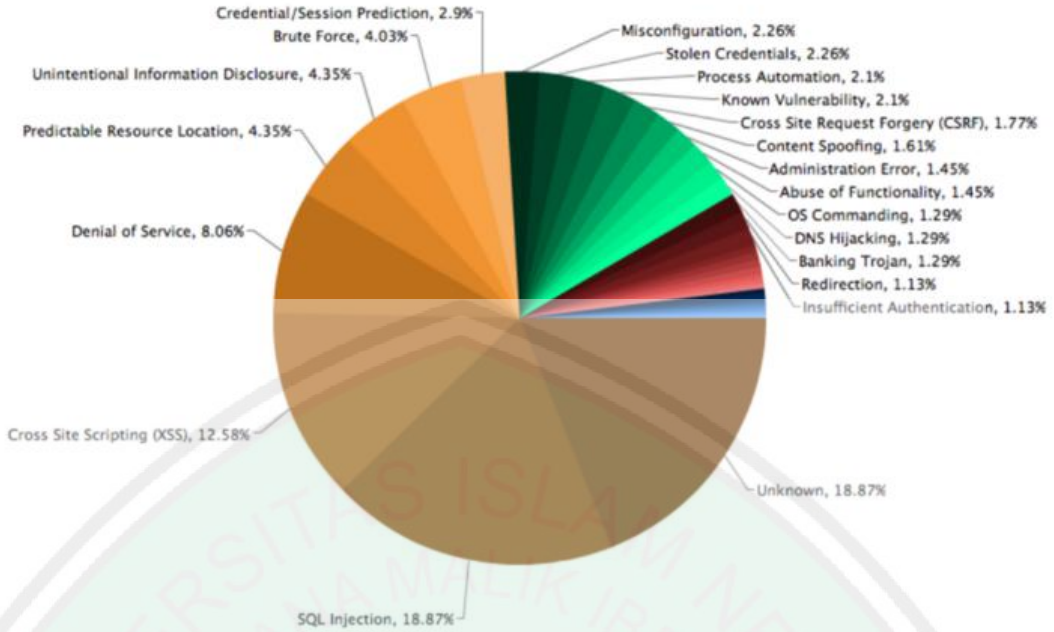
PENDAHULUAN

1.1 Latar Belakang Penelitian

Seiring dengan pesatnya perkembangan teknologi jaringan komputer dan internet, banyak institusi saat ini yang menyalurkan layanannya dengan memanfaatkan fasilitas intemet. Selain untuk mempermudah kinerja, institusi terkait juga dapat memperoleh keuntungan yang lebih karena di era saat ini internet sudah menjadi kebutuhan utama bagi sebagian orang. Internet di mana dan kapan pun dapat digunakan, seperti mengakses informasi, menghubungi kerabat, bahkan jual beli barang.

Dari hasil survei Asosiasi Penyelenggara Jasa Intemet Indonesia (APJII) di tahun 2019, menunjukkan bahwa jumlah pengguna intemet di Indonesia pada tahun 2017 yaitu sebanyak 143,26 juta dari 262 juta total populasi penduduk Indonesia atau sebesar 54,68%. Jauh berbeda dengan tahun 2018, terhitung sebanyak 171,17 juta pengguna internet dari 264,16 juta total populasi penduduk Indonesia atau sebesar 64.8%. Dapat disimpulkan bahwa penetrasi pengguna internet di Indonesia dari tahun ke tahun terus meningkat.

Dengan berkembang pesatnya pengguna internet di Indonesia, maka tidak heran kasus kejahatan di dunia maya juga terus meningkat. Ada beberapa kasus serangan yang sering terjadi khususnya pada web, seperti serangan *Distributed Denial of Service* (DDoS), *SQL Injection*, *Cross Site Scripting* (XSS), dan lain sebagainya. Gambar 1.2 menunjukkan diagram statistik jenis serangan yang sering dimanfaatkan oleh para peretas per tahun 2017.



Gambar 1.1 Jenis-Jenis Serangan di Dunia Maya
(Sumber: Web Hacking Incident Database, 2017)

Salah satu jenis serangan dengan intensitas yang cukup besar yaitu serangan *Denial of Service*. Serangan *Denial of Service* merupakan percobaan yang dilakukan oleh peretas untuk melumpuhkan sistem target dengan cara menghabiskan jaringan atau sumber daya dari sistem tersebut. Jika serangan ini dilakukan dengan lebih dari satu mesin, maka disebut dengan serangan *Distributed Denial of Service* (DDoS) (Mehic dkk, 2015). Serangan DDoS adalah salah satu ancaman paling serius untuk keamanan jaringan, dan jumlah korban serangan DDoS terus meningkat setiap harinya (Kato & Kluyev, 2014).

Salah satu serangan DDoS yang terbesar terjadi pada tanggal 5 Maret 2018 menyerang penyedia layanan internet yang berbasis di USA. Serangan ini berhasil dideteksi oleh peneliti dari Arbor Networks, penyedia layanan pencegahan serangan DDoS dengan analisis perkiraan data yang dikirimkan mencapai 1.7 Tbps (Goodin, 2018).

Dari kasus tersebut, sangat jelas bahwa kejahatan di dunia maya merupakan salah satu tindak kejahatan yang melanggar hukum. Pasal-pasal yang mengatur tentang kejahatan dunia maya khususnya serangan DDoS antara lain:

Pasal 32 ayat (1) dan Pasal 33 Undang-Undang Nomor 11 Tahun 2008 tentang Informasi dan Transaksi Elektronik.

Pasal 32 ayat (1), yaitu: "Setiap Orang dengan sengaja dan tanpa hak atau melawan hukum dengan cara apa pun mengubah, menambah, mengurangi, melakukan transmisi, merusak, menghilangkan, memindahkan, menyembunyikan suatu Informasi Elektronik dan/atau Dokumen Elektronik milik Orang lain atau milik publik".

Pasal 33, yaitu: "Setiap Orang dengan sengaja dan tanpa hak atau melawan hukum melakukan tindakan apa pun yang berakibat terganggunya Sistem Elektronik dan/atau mengakibatkan Sistem Elektronik menjadi tidak bekerja sebagaimana mestinya."

Perbuatan tersangka yang melakukan tindak pidana *cracking* melalui *botnet* telah memenuhi unsur subjektif dan unsur objektif Pasal 32 ayat (1) , maka berdasarkan Pasal 48 Undang-Undang Nomor 11 Tahun 2008 tentang Informasi dan Transaksi Elektronik: "Setiap Orang yang memenuhi unsur sebagaimana dimaksud dalam Pasal 32 ayat (1) dipidana dengan pidana penjara paling lama 8 (delapan) tahun dan/atau denda paling banyak Rp2.000.000.000,00 (dua miliar rupiah)."

Berdasarkan Pasal 49 Undang-Undang Nomor 11 Tahun 2008 tentang Informasi dan Transaksi Elektronik: "Setiap Orang yang memenuhi unsur sebagaimana dimaksud dalam Pasal 33, dipidana dengan pidana penjara paling

lama 10 (sepuluh) tahun dan/atau denda paling banyak Rp10.000.000.000,00 (sepuluh miliar rupiah)."

Selain melanggar hukum negara, kejahatan di dunia maya juga jelas melanggar hukum dan nilai-nilai Islam. Tindakan ini merupakan perbuatan zalim yang tidak disukai oleh Allah. Allah berfirman dalam Al-Qur'an:

وَاللَّهُ لَا يُحِبُّ الظَّالِمِينَ

"Dan Allah tidak menyukai orang-orang yang zalim." (Q.S. Ali 'Imran 57)

Dalam tafsir Jalalayn, disebutkan bahwa Allah tidak menyukai orang-orang yang aniaya, artinya Allah melaknat dan akan menyiksa mereka. Sangat jelas bahwa Allah sangat membenci orang-orang zalim seperti halnya melakukan serangan DDoS, karena perbuatan ini akan sangat merugikan korban.

Dari latar belakang tersebut, maka penelitian ini diharapkan mampu mengurangi kejahatan di dunia maya dengan menghasilkan persentase akurasi deteksi yang lebih baik dan mampu meningkatkan performa sistem deteksi intrusi jaringan dengan mengimplementasikan *General Regression Neural Network.*

1.2 Rumusan Masalah

Berdasarkan dari uraian latar belakang, dapat dirumuskan beberapa permasalahan sebagai berikut:

1. Bagaimana memanfaatkan *General Regression Neural Network* dalam menginvestigasi log jaringan untuk mendeteksi serangan DDoS.
2. Seberapa akurat *General Regression Neural Network* dalam mendeteksi serangan DDoS berdasarkan data latih dan data uji yang digunakan.

1.3 Tujuan Penelitian

Adapun beberapa poin tujuan dari penelitian ini adalah sebagai berikut:

1. Memanfaatkan *General Regression Neural Network* dalam menginvestigasi log jaringan untuk mendeteksi serangan DDoS.
2. Mengetahui tingkat akurasi *General Regression Neural Network* dalam mendeteksi serangan DDoS berdasarkan data latih dan data uji.

1.4 Batasan Masalah

Pada penelitian ini, penulis membatasi permasalahan terhadap jenis serangan yang dianalisis. Penelitian ini hanya difokuskan pada serangan DDoS saja, tidak untuk jenis serangan lain.

1.5 Manfaat Penelitian

Penelitian ini diharapkan menghasilkan persentase akurasi deteksi yang baik dengan memanfaatkan *General Regression Neural Network* dalam mendeteksi serangan yang ada pada jaringan, khususnya serangan DDoS. Dengan hasil yang baik, diharapkan *General Regression Neural Network* dapat diimplementasikan pada sistem deteksi intrusi jaringan sehingga mampu mengurangi angka kejahatan yang terjadi di dunia maya. Selain itu penelitian ini diharapkan dapat menjadi pembelajaran yang baik dan dapat menambah wawasan para pembaca.

1.6 Sistematika Penulisan

Sistematika penulisan memberikan gambaran dan kerangka yang jelas mengenai pokok bahasan di setiap bab. Berikut sistematika penulisan dan pembahasan pada masing-masing bab:

BAB I: PENDAHULUAN

Bab pendahuluan berisi latar belakang penelitian, rumusan masalah, tujuan penelitian, batasan masalah, manfaat penelitian, dan sistematika penulisan.

BAB II: STUDI PUSTAKA

Bab studi pustaka berisi penjelasan tentang penelitian terkait dan landasan -landasan teori yang berhubungan dengan permasalahan penelitian.

BAB III: METODOLOGI PENELITIAN

Bab metodologi penelitian berisi pembahasan tentang rancangan atau prosedur penelitian yang akan diimplementasikan dari awal hingga akhir.

BAB IV: UJI COBA DAN PEMBAHASAN

Bab uji coba dan pembahasan berisi pembahasan dari hasil setiap rancangan atau prosedur yang telah diimplementasikan sebelumnya.

BAB V: KESIMPULAN DAN SARAN

Bab terakhir yaitu bab kesimpulan dan saran. Kesimpulan dan saran uang dipaparkan berdasarkan hasil uji coba yang telah diperoleh sebagai bahan pertimbangan bagi pihak-pihak yang berkepentingan serta kemungkinan pengembangannya.

BAB II

STUDI PUSTAKA

2.1 Penelitian Terdahulu

Penelitian ini bukanlah penelitian pertama dalam bidang forensik jaringan khususnya yang terkait dengan sistem deteksi intrusi jaringan. Ada beberapa penelitian yang sebelumnya telah dipaparkan, diantaranya adalah:

(Resende & Drummond, 2018) mengusulkan sistem deteksi intrusi adaptif berbasis anomali menggunakan algoritma genetika dan *profiling*. Dalam hal ini, *profiling* adalah prosedur relevan yang digunakan untuk menetapkan garis dasar untuk perilaku normal. Sebuah pendekatan adaptif berdasarkan algoritma genetika digunakan untuk memilih fitur *profiling* dan parameter untuk metode deteksi intrusi berbasis anomali. Selain itu, dua metode berbasis anomali diperkenalkan untuk digabungkan dengan pendekatan yang diusulkan. Satu didasarkan pada statistik dasar dan yang lain didasarkan pada prosedur pengelompokan yang diproyeksikan. Dalam percobaan yang dilakukan pada set data CICIDS2017, hasil yang dicapai sebesar 92,85% untuk tingkat deteksi dan 0,69% tingkat kesalahan positif.

(Yulianto dkk, 2018) mempertimbangkan *Synthetic Minority Oversampling Technique* (SMOTE), *Principal Component Analysis* (PCA), dan *Ensemble Feature Selection* (EFS) untuk meningkatkan sistem deteksi intrusi berbasis AdaBoost dengan menggunakan set data CICIDS2017. Penelitian sebelumnya telah mengusulkan penggunaan AdaBoost untuk mengatasi set data baru. Namun karena beberapa masalah seperti ketidakseimbangan data pelatihan dan pemilihan metode klasifikasi yang tidak tepat, sehingga kinerja masih kurang

baik. Mereka bertujuan membangun peningkatan kinerja pendekatan deteksi intrusi untuk menangani ketidakseimbangan data pelatihan, SMOTE dipilih untuk mengatasi masalah tersebut. Selain itu, PCA dan EFS diterapkan sebagai pemilihan fitur untuk memilih atribut-atribut penting dari set data. Hasil evaluasi menunjukkan bahwa AdaBoost yang diusulkan menggunakan hasil PCA dan SMOTE sebesar 92% dan AdaBoost menggunakan EFS dan SMOTE menghasilkan *accuracy* 81,83%, *precision* 81,83%, *recall* 100%, dan *F1-score* 90,01%.

(Radford dkk, 2018) mengevaluasi metode untuk menerapkan deteksi anomali tanpa pengawasan untuk aplikasi keamanan siber pada data atau aliran lalu lintas jaringan komputer. Mereka meminjam dari literatur *Natural Language Processing* (NLP) dan mengkonseptualisasikan data atau aliran lalu lintas jaringan sebagai semacam bahasa yang diucapkan di antara mesin. Lima aturan agregasi urutan dievaluasi untuk kemampuannnya dalam menandai beberapa tipe serangan dalam set data CICIDS2017. Untuk pemodelan urutan, mereka mengandalkan *Long Short-Term Memory* (LSTM) dan *Recurrent Neural Network* (RNN). Selain itu, model berbasis frekuensi sederhana dijelaskan dan kinerjanya dibandingkan dengan model LSTM. Mereka menyimpulkan bahwa model berbasis frekuensi cenderung sebaik atau lebih baik daripada model LSTM untuk tugas-tugas yang dihadapi, dengan beberapa pengecualian. Adapun untuk model berbasis frekuensi diperoleh hasil berdasarkan *Area Under Curve* (AUC) yaitu sebesar 0,96.

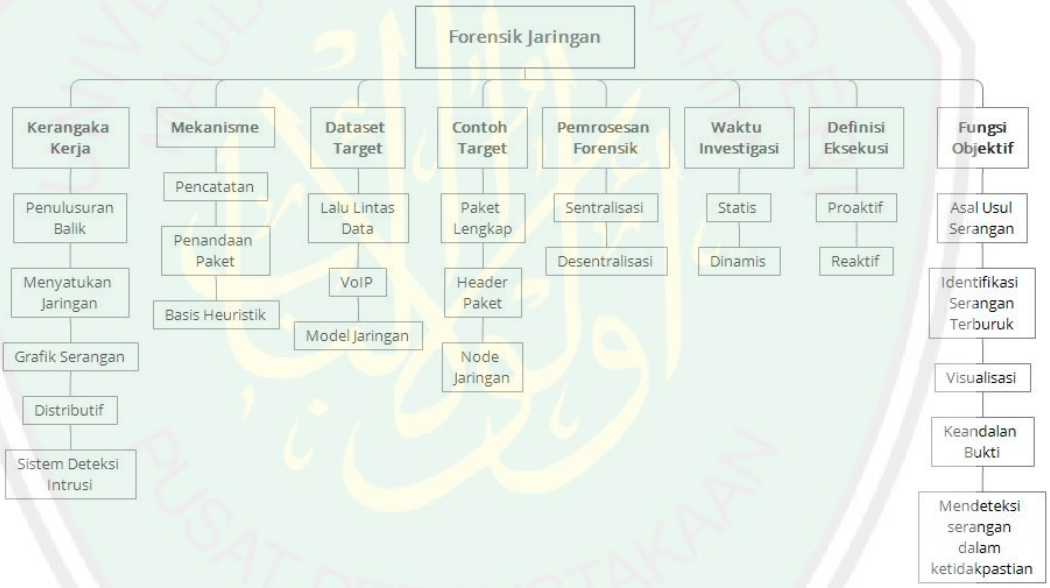
Ketiga penelitian tersebut memperoleh hasil yang cukup baik dengan sama-sama menggunakan set data CICIDS2017. Penelitian yang penulis lakukan menggunakan metode atau pendekatan yang berbeda, yaitu *General Regression*

Neural Network dan *Random Forest* dengan menggunakan set data yang sama yaitu CICIDS2017. Dengan kelebihan atau keunggulan yang dimiliki metode yang digunakan, diharapkan mampu memperoleh hasil yang lebih baik dari penelitian sebelumnya.

2.2 Landasan Teori

Landasan teori menjelaskan teori-teori yang digunakan dalam penelitian ini yang meliputi Forensik Jaringan, *Distributed Denial of Service*, *Random Forest*, dan *General Regression Neural Network.*

2.2.1 Forensik Jaringan



Gambar 2.1 Taksonomi Forensik Jaringan
(Sumber: Khan dkk, 2016)

Forensik jaringan adalah sebuah lingkup penelitian yang bertujuan untuk menemukan pengguna jahat dengan cara mengumpulkan dan menganalisa gangguan dan juga bukti pelanggaran dari kejahatan terhadap jaringan atau komputer seperti peretasan (Liao dkk, 2009).

Forensik jaringan mendukung kemampuan seperti identifikasi penyerang atau pelaku dan rekonstruksi serangan, dimana menyempurnakan deteksi gangguan yang tradisional dan teknik pertahanan dalam membangun sebuah mekanisme keamanan yang kuat. Dahulu dalam merekonstruksi serangan pada jaringan, forensik jaringan digunakan untuk menangkap lalu-lintas jaringan pada satu perangkat lalu ditransmisikan untuk perangkat lain untuk dianalis (Chen dkk, 2013). Namun dalam transmisi data yang besar dari satu perangkat ke perangkat yang lain dapat menyebabkan beban lebih pada saluran komunikasi yang ada sehingga menghasilkan waktu yang tertunda (Ibrahim & Abdullah, 2012).

2.2.1.1 Prinsip-Prinsip Forensik Jaringan

Forensik jaringan secara umum dapat didefinisikan sebagai disiplin ilmu yang bertujuan untuk menemukan dan mengambil informasi rahasia pada jaringan seperti kejahatan dan sejenisnya dengan sedemikan rupa sehingga dapat diterima secara hukum. Hal ini disebabkan karena semua teknik yang digunakan pada forensik jaringan harus memenuhi persyaratan hukum dan teknis.

Penting untuk menjamin apakah solusi forensik jaringan yang dikembangkan bersifat praktis dan cukup cepat untuk digunakan dalam jaringan berkecepatan tinggi dengan arsitektur dan perangkat jaringan yang heterogen. Yang lebih penting adalah solusi forensik jaringan yang dikembangkan tersebut harus memenuhi prinsip umum forensik seperti aturan dalam mengeluarkan bukti hasil investigasi. Berikut 5 aturan bukti-bukti tersebut sebagaimana mestinya menurut kriteria Daubert (Palmer, 2001, Whitcomb, 2002, Mocas, 2004):

a) Dapat diterima (*admissible*). Harus bisa digunakan dalam hukum atau di tempat lain.

b) Asli (*authentic*). Bukti yang di dapatkan dikaitkan dengan kejadian dengan cara yang relevan.

c) Lengkap (*complete*). Tidak ada bukti yang menguntungkan untuk alternatif tersangka.

d) Dapat diandalkan (*reliable*). Tidak diragukan lagi keaslian dan kebenarannya.

e) Dapat dipercaya (*believable*). Jelas, mudah dimengerti, dan dipercaya oleh hakim.

2.2.1.2 Pentingnya Forensik Jaringan

Munculnya industri IT dengan kekhawatiran dengan keamanannya merupakan salah satu faktor yang memotivasi adanya forensik jaringan. Organisasi sangat prihatin tentang keamanan jaringan dan data yang dimilikinya karena banyak serangan yang terjadi pada perusahaan yang berbeda (Zhu, 2011).

Dalam beberapa tahun terakhir, sejumlah serangan, tertuju pada jaringan sosia1 yang berbeda seperti Google, Blogger, Facebook dan Twitter (Thapliyal dkk, 2013). Jenis serangan yang menyerang jaringan sosial tersebut yaitu DDoS yang dilakukan oleh pengguna yang tidak puas untuk menyerang fungsi pada jaringan sosial tersebut. Selain itu ada juga jenis serangan lain yaitu *phising* yang digunakan untuk mendapatkan informasi rahasia dari pengguna. Contohnya untuk mendapatkan kata sandi dari rekening bank yang dapat mengubah penyusup menjadi seorang miliarder dalam beberapa menit (Li & Schmitz, 2009, Layton dkk, 2010).

2.2.1.3 Teknik Forensik Jaringan

Pada Gambar 2.1 dapat dilihat bahwa terdapat lima teknik dalam forensik jaringan (Liao dkk, 2009), yaitu:

a) Penelusuran Balik (*Traceback*)

Identifikasi asal usul paket dalam suatu jaringan disebut *traceback* atau penelusuran balik. Teknik ini digunakan untuk mengidentifikasi dari mana sumber paket dihasilkan dengan mengidentifikasi asal serangan. Penelusuran balik adalah salah satu teknik forensik jaringan yang tepat digunakan untuk mengidentifikasi asal paket dengan menyelidiki jalur serangan terutama untuk serangan DDoS dan IP *spoofing*. Penelusuran balik lebih signifikan karena serangan *botnet* dan DDoS yang disaksikan di jaringan terdistribusi yang berbeda. Sistem jaringan terdistribusi yang bekerja sama dengan internet menyediakan suasana dan menarik *bot-master* untuk menyerang. Untuk mengatasi serangan ini, perlu untuk menjaga sistem jaringan aman dengan memasukkan berbagai mekanisme *traceback* dengan cara yang efisien.

b) Jaringan Konvergen (*Converge Network*)

Bagian ini membahas identifikasi bukti digital dalam jaringan konvergen terutama dalam komunikasi VoIP. Komunikasi VoIP mewarisi ancaman keamanan, kerentanan, dan serangan dari jaringan data karena menggunakan media untuk komunikasinya. Dalam sinyal komunikasi VoIP dibagi menjadi kerangka yang tertanam dalam paket data sebagai kode suara dan dikirim pada jaringan IP sebagai paket suara normal. Paket suara mengirim dari satu pemanggil (pengirim) ke pemanggil lain (penerima) tanpa gangguan dan modifikasi oleh penyusup disebut paket suara normal. Biasanya, protokol SIP dan H.3231 digunakan untuk mengangkut paket suara pada jaringan IP. Paket suara membungkus alamat IP dan informasi *port* untuk membantu protokol komunikasi suara yang juga bertindak sebagai protokol kontrol sesi.

Informasi alamat IP dan port yang tertutup tidak dienkripsi karena persyaratan perangkat penerjemahan alamat jaringan untuk menerjemahkan lalu lintas suara. Dengan demikian, bidang yang tidak dienkripsi dalam paket suara meningkatkan kemungkinan untuk diserang oleh penyusup. Paket suara dieksploitasi oleh penyusup selama komunikasi suara yang mengubah paket suara normal ke paket jahat. Umumnya, paket jahat dihasilkan oleh penyusup dalam serangan yang berbeda termasuk *flooding*, panggilan pembajakan, *buffer overflow*, *man-in-the-middle*, penyadapan, kebocoran privasi, merusak integritas panggilan, dan eksploitasi berbagai perangkat VoIP.

c) Grafik Serangan (*Attack Graphs*)

Grafik serangan digunakan untuk mengidentifikasi semua jalur serangan dalam jaringan yang dilakukan penyusup selama serangannya dengan menganalisis host, jaringan, dan perangkat keamanan lainnya. Grafik serangan mengidentifikasi dan memvisualisasikan jalur serangan yang digunakan oleh penyusup selama serangan mereka. Grafik serangan berisi simpul yang mewakili simpul serangan dan tepi mengidentifikasi transisi antara simpul yang berbeda. Selain itu, grafik serangan digunakan untuk investigasi forensik jaringan yang memvisualisasikan jalur serangan dan menentukan jalur serangan terburuk untuk membantu administrator jaringan dalam pencegahan sebelum serangan terjadi. Grafik serangan digunakan untuk mengidentifikasi analisis dampak & terukur, mengidentifikasi serangan jaringan *multi stage*, pengumpulan bukti, dan pembekuan keamanan *cost-benefit*.

d) Distributif (*Distributive*)

Teknik forensik jaringan berbasis distributif memfasilitasi penyelidik jaringan dalam mengatasi masalah skalabilitas dengan mendistribusikan server forensik jaringan dan sistem agen data. Server forensik jaringan mengumpulkan data untuk analisis dari agen data dispersi terdistribusi di lokasi yang berbeda dalam jaringan. Teknik ini melakukan investigasi, bertindak atas tanggapan yang muncul, mengidentifikasi asal serangan, dan melakukan pengumpulan bukti. Namun, teknik distributif menciptakan *overhead* untuk menjaga server forensik teteap aman dari penyusup yang didistribusikan dalam jaringan.

e) Sistem Deteksi Intrusi (*Intrusion Detection System*)

Salah satu kerangaka kerja pada forensik jaringan yang akan digunakan pada penelitian ini yaitu sistem deteksi intrusi. Sistem deteksi intrusi adalah sebuah metode yang dapat digunakan untuk mendeteksi aktivitas yang mencurigakan dalam sebuah sistem atau jaringan. IDS dapat melakukan inspeksi terhadap lalu lintas jaringan *inbound* dan *outbound* dalam sebuah sistem atau jaringan, melakukan analisis dan mencari bukti dari percobaan intrusi (penyusupan).

Ada beberapa cara bagaimana IDS bekerja. Cara yang paling populer adalah dengan menggunakan pendeteksian berbasis *signature* (seperti halnya yang dilakukan oleh beberapa *antivirus*), yang melibatkan pencocokan lalu lintas jaringan dengan basis data yang berisi cara-cara serangan dan penyusupan yang sering dilakukan oleh penyerang. Sama seperti halnya *antivirus*, jenis ini membutuhkan pembaruan terhadap basis data *signature* IDS yang bersangkutan.

Metode selanjutnya adalah dengan mendeteksi adanya anomali, yang disebut sebagai *anomaly-based* IDS. Jenis ini melibatkan pola lalu lintas jaringan yang mungkin merupakan sebuah serangan yang sedang dilakukan oleh penyerang. Umumnya, dilakukan dengan menggunakan teknik statistik untuk membandingkan lalu lintas yang sedang dipantau dengan lalu lintas normal yang biasa terjadi. Metode ini menawarkan kelebihan dibandingkan *signature-based* IDS, yakni ia dapat mendeteksi bentuk serangan yang baru dan belum terdapat di dalam basis data *signature* IDS. Kelemahannya, adalah jenis ini sering mengeluarkan pesan *false positive*. Sehingga tugas administrator menjadi lebih rumit, dengan harus memilah-milah mana yang merupakan serangan yang sebenarnya dari banyaknya laporan *false positive* yang muncul.

Teknik lainnya yang digunakan adalah dengan memantau data-data sistem operasi, yakni dengan cara melihat apakah ada percobaan untuk mengubah beberapa data sistem operasi, utamanya data log. Teknik ini seringnya diimplementasikan di dalam *Host Intrusion Detection System* (HIDS), selain tentunya melakukan pemindaian terhadap log sistem untuk memantau apakah terjadi kejadian yang tidak biasa.

2.2.2 *Distributed Denial of Service*

Serangan *Denial of Service* (DoS) merupakan percobaan yang dilakukan oleh peretas untuk melumpuhkan sistem target dengan cara menghabiskan jaringan atau sumber daya dari sistem tersebut. Jika serangan ini dilakukan dengan lebih dari satu mesin, maka disebut dengan serangan *Distributed Denial of Service* (DDoS) (Mehic dkk, 2015). Serangan DDoS adalah salah satu ancaman

paling serius untuk keamanan jaringan, dan jumlah korban serangan DDoS terus meningkat setiap harinya (Kato & Kluyev, 2014).

Kesaksian pengadilan menunjukkan bahwa demonstrasi pertama serangan DoS dilakukan oleh Khan C. Smith pada tahun 1997 selama acara Def Con yang mengganggu akses internet ke jalur Las Vegas selama lebih dari satu jam. Dirilisnya contoh kode serangan pada acara tersebut menyebabkan serangan daring terhadap Sprint, EarthLink, E-Trade, dan beberapa perusahaan besar lainnya di tahun berikutnya (Smith, 2014).

Serangan DDoS terbesar yang baru-baru ini terjadi menyerang penyedia layanan internet yang berbasis di USA. Serangan ini berhasil dideteksi oleh peneliti dari Arbor Networks, penyedia layanan pencegahan serangan DDoS dengan analisis perkiraan data yang dikirimkan mencapai 1.7 Tbps (Goodin, 2018).

Selain itu serangan besar-besaran DDoS juga pernah terjadi pada bulan Oktober 2016 lalu yang dialami oleh salah satu perusahaan besar penyedia layanan DNS, yaitu Dyn. Tampaknya perangkat IoT juga memainkan peran penting dalam menurunkan layanan DNS Dyn, sebuah serangan yang analisis awalnya diperkirakan mencapai 1,2 Tbps, sekitar dua kali volume dari yang pernah dialami Krebs. Nama besar seperti Twitter, Amazon, AirBnB, Spotify, GitHub dan lainnya secara efektif menjadi offline karena serangan DDoS memblokir server DNS *anycast* milik Dyn. Serangan dimulai pada 21 Oktober 2016 dan terjadi. Dyn berhasil mengembalikan kendali dalam waktu singkat namun serangan kedua menyerangnya lagi selama beberapa jam. Dalam sebuah pernyataan yang dikeluarkan beberapa hari kemudian, Dyn memberi konfirmasi

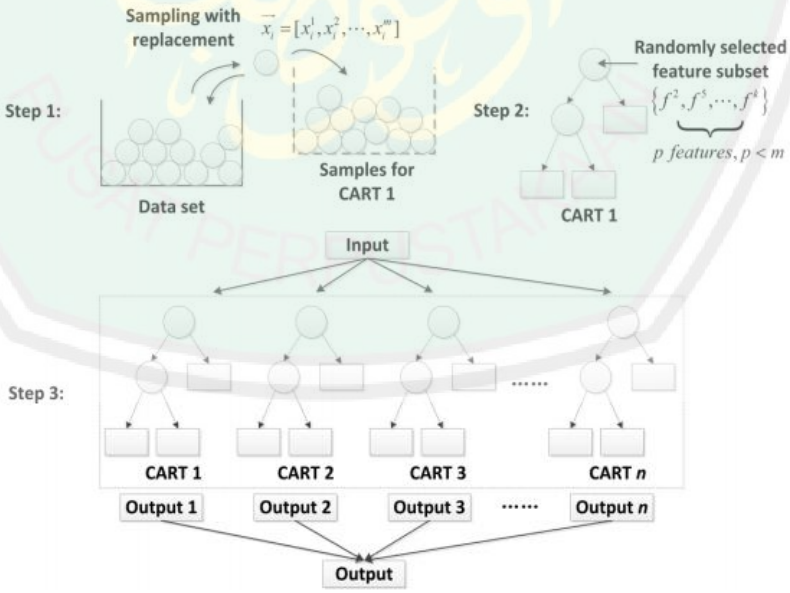
terhadap apa yang telah banyak dikabarkan: "Kami dapat memastikan bahwa sejumlah besar lalu lintas serangan berasal dari *botnet* berbasis Mirai." kata perusahaan tersebut (Hilton, 2016).

2.2.3 *Random Forest*

Random Forest adalah metode *ensemble learning* yang pertama kali diusulkan oleh Breiman pada tahun 2001. Maksud dari *ensemble learning* yaitu kumpulan dari metode pembelajaran menggunakan pohon keputusan sebagai basis klasifikasi yang dibangun dan dikombinasikan (Xiao dkk, 2012).

Algoritma ini mampu menghasilkan kesalahan yang lebih rendah, menghasilkan akurasi yang baik dalam klasifikasi, menangani data latih dalam jumlah yang besar, dan efektif dalam mengatasi data yang tidak lengkap (Breiman, 2001).

Berikut langkah-langkah rinci dari *Random Forest* seperti yang dapat dilihat pada Gambar 2.2 (Lin dkk, 2017).



Gambar 2.2 Langkah-langkah Algoritma *Random Forest*
(Sumber: Lin dkk, 2017)

1. Menghasilkan sebuah set-set data baru dari sampel acak dengan penggantian (*bootstrap*) dari set data yang asli.
2. Dibangun sebuah pohon untuk setiap set data baru dengan pemilihan fitur acak di setiap simpul pohon dan tanpa pemangkasan.
3. Setelah sejumlah besar pohon dihasilkan, data baru diprediksi dengan cara menggabungkan hasil semua pohon dengan strategi *voting* mayoritas.

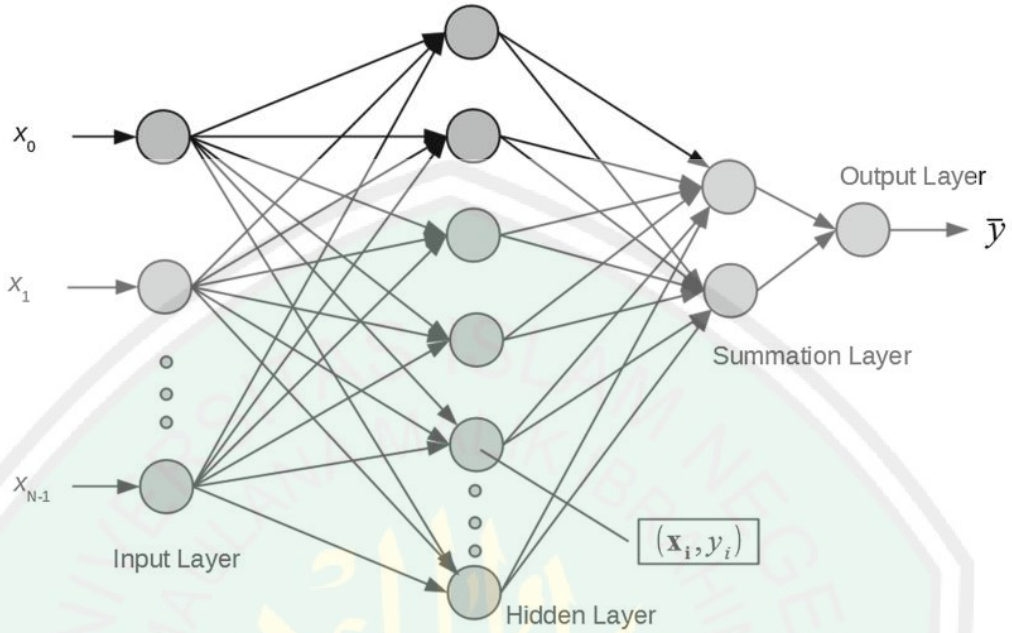
Random Forest juga dapat digunakan untuk menyeleksi fitur yang ada pada set data. Salah satu jenis ukuran pentingnya fitur yang umumnya digunakan pada *Random Forest* yaitu *Gini Importance Index*. *Gini Importance Index* secara langsung berasal dari indeks Gini yang digunakan sebagai ukuran ketidakmurnian simpul. Nilai penting fitur dalam satu pohon adalah jumlah dari pengurangan indeks Gini atas semua simpul yang digunakan untuk memisahkan fitur tertentu. Keseluruhan pentingnya fitur pada suatu pohon di hutan didefinisikan sebagai penjumlahan atau rata-rata nilai kepentingannya di antara semua pohon di hutan (Breiman, 2001).

2.2.4 *General Regression Neural Network*

General Regression Neural Network merupakan variasi dari *Radial Basis Neural Network* yang diusulkan pertama kali oleh D.F. Specht in 1991. *General Regression Neural Network* dapat digunakan pada kasus regresi, prediksi, maupun klasifikasi serta dapat menjadi solusi yang baik untuk sistem dinamis (Specht, 1991).

General Regression Neural Network memiliki empat lapisan unit pemrosesan. Lapisan-lapisan tersebut yaitu lapisan masukan (*input layer*), lapisan tersembunyi (*hidden layer*), lapisan penjumlahan (*summation layer*), dan yang

terakhir lapisan keluaran (*output layer*). Arsitektur *General Regression Neural Network* dapat dilihat pada Gamber 2.3 (Rizzo dkk, 2015).



Gambar 2.3 Arsitektur *General Regression Neural Network*
(Sumber: Rizzo dkk, 2015)

Setiap lapisan unit pemrosesan ditandai dengan suatu fungsi yang spesifik (https://www.dtreg.com/solution/view/22).

1. Lapisan masukan, menyimpan neuron masukan untuk setiap variabel prediktor tanpa adanya pemrosesan data yang dilakukan. Tidak ada pemrosesan data yang dilakukan pada neuron-neuron tersebut. Neuron masukan kemudian mengirimkan data ke lapisan kedua dari unit pemrosesan yang disebut lapisan tersembunyi.
2. Lapisan tersembunyi, menyimpan nilai-nilai dari variabel-variabel prediktor bersama dengan nilai target. Dalam hal ini, neuron pada lapisan tersembunyi sama dengan neuron pada lapisan masukan. Neuron pada lapisan tersembunyi menghitung matriks jarak dari nilai target sebagai titik pusat neuron,

kemudian menerapkan fungsi kernel *radial basis* menggunakan nilai sigma (σ).

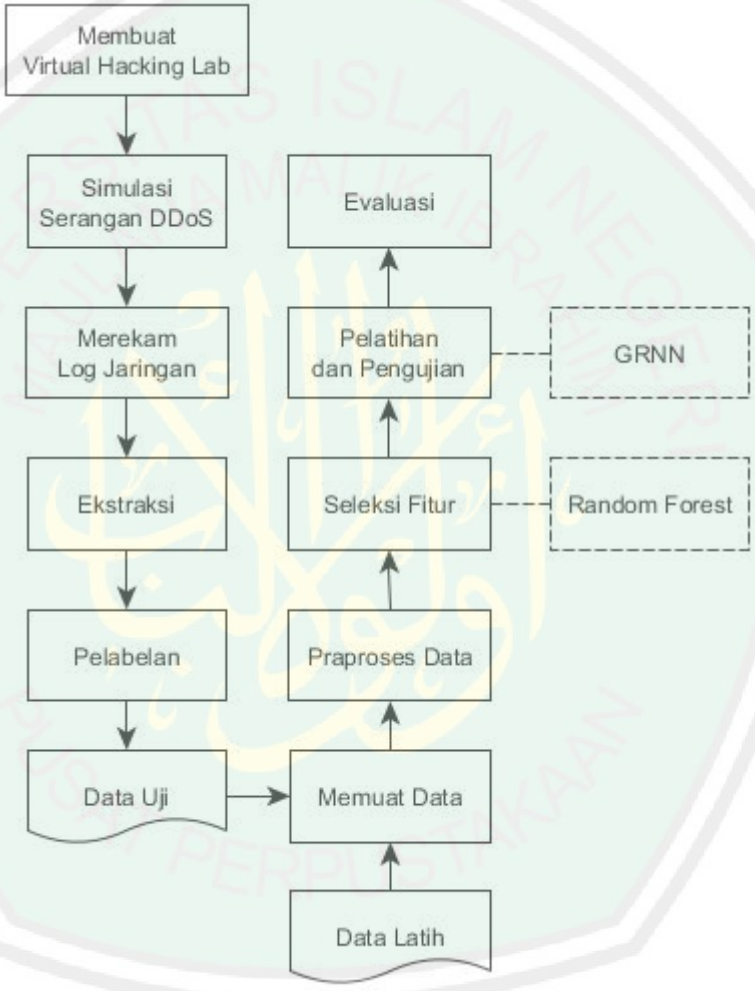
3. Lapisan penjumlahan, hanya memiliki dua neuron yaitu neuron penjumlahan penyebut dan neuron penjumlahan pembilang. Neuron penjumlahan penyebut menambah bobot yang berasal dari masing-masing neuron tersembunyi. Sedangkan neuron penjumlahan pembilang menambah nilai bobot dikalikan nilai target aktual untuk setiap neuron tersembunyi.
4. Lapisan keluaran, membagi nilai akumulasi dalam unit penjumlahan pembilang dengan nilai di unit penjumlahan penyebut yang kemudian hasilnya ditetapkan sebagai nilai terget prediksi.

BAB III

METODOLOGI PENELITIAN

3.1 Prosedur Penelitian

Pada penelitian ini, ada beberapa tahapan yang akan dilakukan seperti yang dapat dilihat pada Gambar 3.1.



Gambar 3.1 Prosedur Penelitian

Tahapan-tahapan tersebut yaitu sebagai berikut: 1) Membuat *virtual hacking lab* sederhana. 2) Melakukan simulasi serangan DDoS ke server web dengan menggunakan aplikasi *Low Orbit Ion Cannon* (LOIC) dan Metasploit. 3) Disaat serangan berlangsung, log jaringan dipantau dan direkam menggunakan

Wireshark. 4) Data dengan format PCAP yang didapatkan sebelumnya kemudian diekstrak menggunakan aplikasi CICFlowMeter 4.0. Dari hasil ekstraksi diperoleh data dengan format CSV yang akan digunakan sebagai data uji. 5) Dilakukan pelabelan untuk memberi kelas pada data uji berdasarkan skema waktu serangan yang dilakukan. 6) Tahap selanjutnya yaitu memuat data latih dari set data CICIDS2017 dan data uji dari simulasi DDoS yang dilakukan sebelumnya. 7) Setelah itu dilakukan praproses data yang meliputi penghapusan sebagian data yang tidak perlu dan normalisasi keseluruhan data. 8) Sebelum dilakukan pelatihan dan pengujian, perlu dilakukan pemilihan fitur untuk mengurangi waktu komputasi dan meningkatkan akurasi. Algoritma yang digunakan untuk pemilihan fitur yaitu *Random Forest.* 9) Setelah itu dilakukan analisis pelatihan dan pengujian menggunakan *General Regression Neural Network.* 10) Setelah itu dilakukan evaluasi model untuk mengetahui persentase *accuracy*, *precision*, *recall*, dan *f1-score* melalui *Confusion Matrix.*

3.1.1 Membuat *Virtual Hacking Lab*

Sebelum melakukan simulasi serangan, peneliti merancang dan mengimplementasikan sebuah *hacking lab* berbasis virtual. Adapun aplikasi yang digunakan yaitu Oracle VM VirtualBox sebagai mesin virtual, sistem operasi Windows dan Kali Linux sebagai penyerang, dan Metasploitable yang akan bertindak sebagai server web. Semua perangkat tersebut akan dikonfigurasikan sehingga dapat terhubung satu sama lain.

3.1.2 Simulasi Serangan DDoS

Salah satu aplikasi yang digunakan untuk simulasi serangan DDoS yaitu *Low Orbit Ion Cannon* (LOIC). Bisa dibilang jika LOIC adalah aplikasi yang

paling populer digunakan untuk melakukan serangan DoS. Aplikasi berbasis Windows ini efektif untuk mengirimkan banyak jumlah paket ICMP, TCP atau UDP ke target. LOIC telah teruji digunakan oleh 4chan dalam serangan *project chanology* terhadap web Church of Scientology di tahun 2009. Selain itu LOIC juga digunakan oleh Anonymous dalam *operation payback* melawan PayPal, Visa, and MasterCard karena memotong aliran dana sumbangan ke Wikileaks.

Selain LOIC, Metasploit juga digunakan untuk melakukan simulasi serangan DDoS. Metasploit adalah proyek keamanan komputer yang menyediakan informasi tentang kerentanan keamanan dan bantuan dalam pengujian penetrasi dan pengembangan sistem deteksi intrusi.

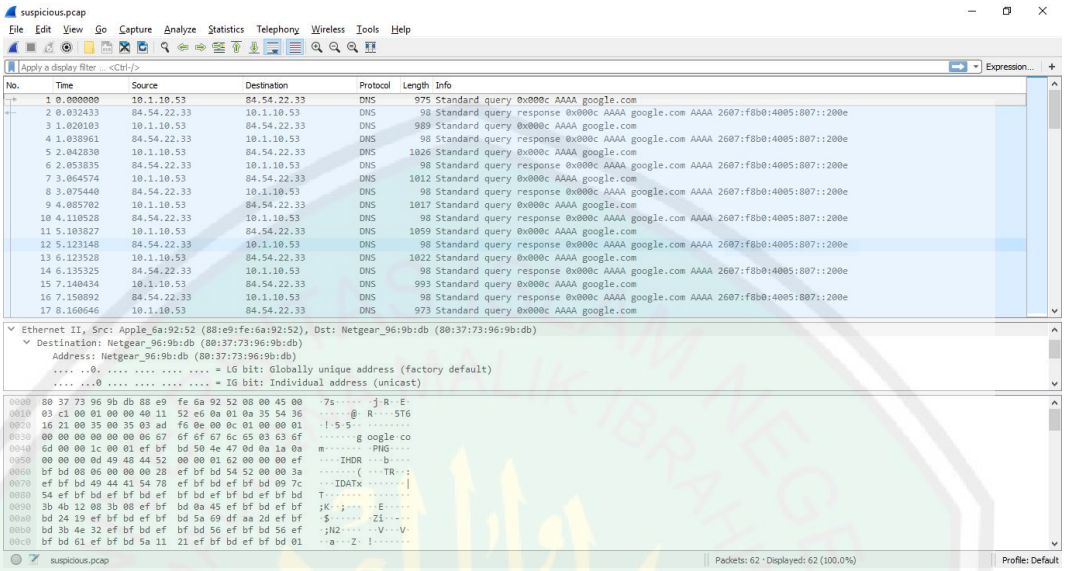
Kedua aplikasi tersebut digunakan dalam waktu bersamaan. Adapun simulasi serangan dilakukan dengan memperhatikan skema waktu serangan. Simulasi serangan dilakukan dengan selang waktu beberapa menit untuk penyerangan dan beberapa menit pula untuk menormalkan lalu lintas jaringan. Hal ini dilakukan berulang kali selama kurang lebih 20 menit.

3.1.3 Merekam Log Jaringan

Untuk merekam log jaringan, aplikasi yang digunakan yaitu Wireshark yang merupakan *network protocol analyzer* alias penganalisa protokol jaringan yang memiliki fitur lengkap. Aplikasi ini dapat menangkap semua paket yang lewat serta menyeleksi dan menampilkan data tersebut sedetail mungkin.

Wireshark merekam semua paket yang melewati *interface* yang dipilih. *Interface* adalah perangkat penghubung antar jaringan, bisa melalui *wifi* atau *ethernet*. Hasil rekaman tadi dapat dianalisa dengan memfilter protokol apa yang ingin ditampilkan seperti TCP, HTTP, UDP dan sebagainya serta dapat mencatat

cookie, *post* dan *request*. Log jaringan yang direkam dapat disimpan dalam beberapa format, salah satunya PCAP. Gambar 3.2 menunjukkan tampilan *user interface* dari Wireshark.



Gambar 3.2 *User Interface* Wireshark

3.1.4 Ekstraksi Log Jaringan

Untuk mengekstrak fitur lalu lintas jaringan, aplikasi yang digunakan yaitu CICFlowMeter yang merupakan ekstraktor fitur dan dapat mengekstrak lebih dari 80 fitur dari data dengan format PCAP. CICFlowMeter merupakan aplikasi sumber terbuka yang dikembangkan oleh Canadian Institute for Cybersecurity. Aplikasi ini telah banyak digunakan di banyak set data dalam bidang *cyber security.*

CICFlowMeter menghasilkan *Bidirectional Flows* (Biflow), di mana paket pertama menentukan arah maju (sumber ke tujuan) dan mundur (tujuan ke sumber), maka 84 fitur statistik seperti *Duration*, *Number of packets*, *Number of bytes*, *Length of packets*, dan lainnya juga dihitung secara terpisah dalam arah maju dan mundur. Output dari aplikasi ini adalah data dengan format CSV dengan

enam kolom berlabel untuk setiap aliran, yaitu *FlowID*, *SourceIP*, *DestinationIP*, *SourcePort*, *DestinationPort*, dan *Protocol* dengan lebih dari 80 fitur lalu lintas jaringan (http://www.netflowmeter.ca/).

3.1.5 Pelabelan

Proses ekstraksi menghasilkan data dengan format CSV yang berisi fitur-fitur yang sama seperti data latih. Tetapi dari hasil ekstraksi yang dilakukan, fitur *Label* tidak dapat diklasifikasikan secara otomatis, mana yang termasuk kelas normal ataupun kelas serangan. Meskipun demikian, pelabelan dilakukan secara manual dengan memperhatikan skema waktu simulasi serangan yang dilakukan sebelumnya.

3.1.6 Memuat Data Latih dan Data Uji

Sebelum memulai praproses data, data dimuat terlebih dahulu agar dapat terbaca oleh sistem. Selain itu, data latih dan data uji juga diberikan *custom header* untuk memudahkan dalam proses pengolahan. Berikut penjelasan tentang data latih dan data uji yang digunakan.

a) Data Latih

Data yang digunakan yaitu set data CICIDS2017 yang dikeluarkan oleh Canadian Institute for Cybersecurity. Set data ini sejak awal mulai menarik para peneliti untuk melakukan analisis serta pengembangan model dan algoritma baru (Panigrahi & Borah, 2018). Set data CICIDS2017 berisi kelas normal dan serangan terbaru pada jaringan dalam format PCAP. Selain itu tersedia juga data dengan format CSV yang sudah diekstrak menggunakan CICFlowMeter dengan tujuan pembelajaran *machine learning* atau *deep learning* yang dapat dengan bebas digunakan oleh peneliti.

Set data ini menyediakan data lengkap berisi berbagai jenis serangan pada jaringan dan juga data terpisah setiap serangan. Data latih yang digunakan pada penelitian ini yaitu data yang khusus berisi kelas normal dan kelas serangan DDoS saja. Deskripsi singkat dari data yang digunakan dapat dilihat pada Tabel 3.1. Adapun fitur-fitur yang ada pada data ini yaitu sebanyak 84 fitur seperti yang dapat dilihat pada Tabel 3.2.

Tabel 3.1 Deskripsi Singkat Data Latih yang Digunakan

Nama Data	Friday-WorkingHours-Afternoon-DDos.pcap_ISCX
Tahun Rilis	2017
Jenis Serangan	DDoS LOIT
Jumlah Data	225.745
Jumlah Fitur	84

Tabel 3.2 Fitur-Fitur Data Latih

No.	Nama Fitur	No.	Nama Fitur
1.	Flow ID	43.	Fwd Pkts/s
2.	Src IP	44.	Bwd Pkts/s
3.	Src Port	45.	Pkt Len Min
4.	Dst IP	46.	Pkt Len Max
5.	Dst Port	47.	Pkt Len Mean
6.	Protocol	48.	Pkt Len Std
7.	Timestamp	49.	Pkt Len Var
8.	Flow Duration	50.	FIN Flag Cnt
9.	Tot Fwd Pkts	51.	SYN Flag Cnt
10.	Tot Bwd Pkts	52.	RST Flag Cnt
11.	TotLen Fwd Pkts	53.	PSH Flag Cnt
12.	TotLen Bwd Pkts	54.	ACK Flag Cnt
13.	Fwd Pkt Len Max	55.	URG Flag Cnt
14.	Fwd Pkt Len Min	56.	CWE Flag Count
15,	Fwd Pkt Len Mean	57.	ECE Flag Cnt
16.	Fwd Pkt Len Std	58.	Down/Up Ratio
17.	Bwd Pkt Len Max	59.	Pkt Size Avg

18.	Bwd Pkt Len Min	60.	Fwd Seg Size Avg
19.	Bwd Pkt Len Mean	61.	Bwd Seg Size Avg
20.	Bwd Pkt Len Std	62.	Fwd Byts/b Avg
21.	Flow Byts/s	63.	Fwd Pkts/b Avg
22.	Flow Pkts/s	64.	Fwd Blk Rate Avg
23.	Flow IAT Mean	65.	Bwd Byts/b Avg
24.	Flow IAT Std	66.	Bwd Pkts/b Avg
25.	Flow IAT Max	67.	Bwd Blk Rate Avg
26.	Flow IAT Min	68.	Subflow Fwd Pkts
27.	Fwd IAT Tot	69.	Subflow Fwd Byts
28.	Fwd IAT Mean	70.	Subflow Bwd Pkts
29.	Fwd IAT Std	71.	Subflow Bwd Byts
30.	Fwd IAT Max	72.	Init Fwd Win Byts
31.	Fwd IAT Min	73.	Init Bwd Win Byts
32.	Bwd IAT Tot	74.	Fwd Act Data Pkts
33.	Bwd IAT Mean	75.	Fwd Seg Size Min
34.	Bwd IAT Std	76.	Active Mean
35.	Bwd IAT Max	77.	Active Std
36.	Bwd IAT Min	78.	Active Max
37.	Fwd PSH Flags	79.	Active Min
38.	Bwd PSH Flags	80.	Idle Mean
39.	Fwd URG Flags	81.	Idle Std
40.	Bwd URG Flags	82.	Idle Max
41.	Fwd Header Len	83.	Idle Min
42.	Bwd Header Len	84.	Label

b) Data Uji

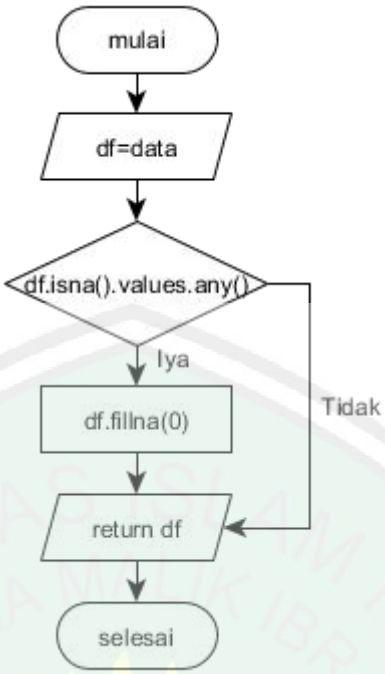
Data uji yang digunakan adalah hasil dari simulasi serangan DDoS terhadap server web. Bersamaan dengan dilakukannya simulasi serangan, lalu lintas jaringan dipantau dan direkam menggunakan Wireshark. Data yang didapatkan awalnya merupakan data log jaringan dengan format PCAP yang kemudian dilakukan ekstraksi dengan menggunakan CICFlowMeter 4.0 untuk mendapatkan format data yang sama dengan data latih.

3.1.7 Praproses Data

Praproses merupakan tahap penting yang bertujuan untuk memanipulasi data menjadi format yang dapat dimengerti oleh sistem. Kebanyakan data yang digunakan untuk dianalisis merupakan data yang tidak lengkap dan tidak konsisten. Oleh sebab itu perlu dilakukan praproses data yang bertujuan untuk meningkatkan kualitas data, hal ini akan membantu untuk meningkatkan akurasi dan efisiensi dai hasil yang didapatkan. Praproses data sangatlah penting dan vital dalam menganalisa lalu lintas jaringan karena polanya memiliki tipe data dan dimensi yang berbeda-beda.

a) Pembersihan Data

Tahap praproses yang pertama dilakukan yaitu pembersihan data. Pembersihan data perlu dilakukan karena sering didapati data yang hilang atau rusak. Penting untuk memahami sumber data yang hilang atau rusak tersebut. Bisa jadi kesalahan pada saat transfer data, kesalahan yang disebabkan oleh sistem ataupun disebabkan oleh permasalahan lainnya. Jika hal tersebut ada pada data yang digunakan, maka akan menghambat proses analisis yang dilakukan. Oleh sebab itu data tersebut perlu dihilangkan dengan cara mengganti data tersebut dengan nilai konstan atau dengan nilai tengah atau rata-rata dari data yang sekolom dengan data tersebut. *Flowchart* dari pembersihan data ditunjukkan pada Gambar 3.3.



Gambar 3.3 *Flowchart* Membersihkan Data yang Hilang atau Rusak

b) Normalisasi Atribut Numerik

Setelah dilakukan pembersihan data dari data yang hilang atau rusak, tahap selanjutnya yang perlu dilakukan adalah normalisasi atribut numerik maupun kategorik. Sebelum itu, perlu diketahui perbedaan dari atribut numerik dan atribut kategorik terlebih dahulu. Atribut numerik digunakan untuk data yang dapat diukur secara kuantitatif sehingga dapat menerima operasi matematik. Atribut numerik mencakup data interval dan data rasio. Atribut kategorik digunakan untuk data yang tidak dapat dihitung secara kuantitatif sehingga tidak dapat menerima operasi matematik seperti penjumlahan dan perkalian. Namun demikian, nilai-nilainya dapat dibedakan antara satu dengan lainnya. Atribut kagetorik terdiri dari data nominal, biner, dan ordinal.

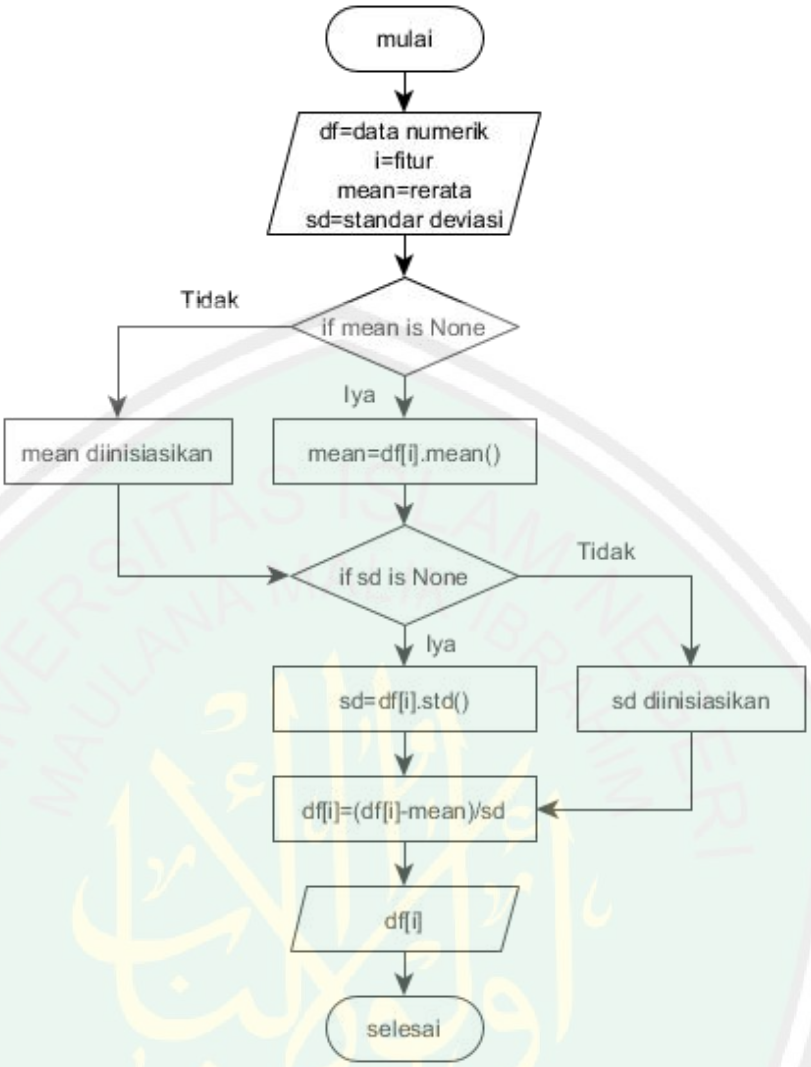
Untuk mengetahui yang termasuk atribut numerik dan atribut kategorik yang ada pada set data yang digunakan, maka perlu diketahui tipe data setiap kolom atau setiap fitur yang ada. Jika tipe data dari fitur berupa *integer* atau *float*,

maka fitur tersebut berupa fitur dengan atribut numerik, sedangkan jika tipe data fitur berupa *object* maka fitur tersebut berupa fitur dengan atribut kategorik. Namun perlu diperhatikan bahwa tidak semua atribut yang berupa angka itu merupakan atribut numerik. Misalnya level 1, 2, 3 dan seterusnya. Meskipun berupa angka, tetapi bisa dipastikan atribut tersebut bersifat kategorik.

CICIDS2017 sebagai set data yang digunakan, memiliki tiga jenis tipe data yaitu *integer*, *float*, dan *object.* Normalisasi atribut numerik dilakukan dengan melakukan standarisasi data pada semua atribut dengan tipe data *integer* dan *float.* Atribut numerik yang ada memiliki nilai dengan rentang berbeda-beda. Ada yang jumlahnya besar dan ada pula yang kecil. Rata-rata dan deviasi standar dihitung pada setiap fitur dan kemudian fitur dinormalisasikan berdasarkan pada persamaan 3.1:

$$\text{normalisasi}(x_i) = \frac{x_i - mean(x)}{stdev(x)} \tag{3.1}$$

Normalisasi atribut numerik ini dilakukan agar nilai tersebut menjadi lebih kecil dan seragam tanpa mengubah esensi dari nilai tersebut. Dengan demikian data dapat dianalisis dengan baik oleh model jaringan saraf tiruan yang digunakan. *Flowchart* normalisasi atribut numerik ditunjukkan pada Gambar 3.4.



Gambar 3.4 *Flowchart* Normalisasi Atribut Numerik

Berikut adalah contoh perhitungan manual dari proses normalisasi atribut numerik. Data yang digunakan adalah sampel data dari set data CICIDS2017 yang telah dibersihkan sebelumnya seperti yang dapat dilihat pada Tabel 3.3.

Tabel 3.3 Sampel Data untuk Normalisasi Atribut Numerik

No.	**Destination Port**	**Flow Duration**	**Total Fwd Packet**	**…**	**Idle Min**
1	54865	3	2	…	0
2	55054	109	1	…	0
3	55055	52	1	…	0
…	…	…	…	…	…
225745	61326	68	1	…	0

Sebagai contoh, data yang digunakan yaitu data dari fitur *Flow Duration* (x).

1. Menghitung rata-rata fitur.

$$\text{mean(x)} = \frac{\sum_{i=1}^{n} x_i}{n}$$

$$= \frac{3+109+52+...+68}{225745}$$

$$= \frac{3.66647\text{E}+12}{225745}$$

$$= 16241648.53$$

2. Menghitung deviasi standar fitur.

$$\text{stdev(x)} = \sqrt{\frac{\sum_{i=1}^{n}(x_i-\bar{x})^2}{n-1}}$$

$$= \sqrt{\frac{(3-16241648.53)^2+...+(68-16241648.53)^2}{225745-1}}$$

$$= \sqrt{\frac{(2.63791\text{E}+14)+...+(2.63789\text{E}+14)}{225744}}$$

$$= \sqrt{\frac{2.24341\text{E}+20}{225744}}$$

$$= 31524374.23$$

3. Normalisasi setiap atribut pada fitur. Perhitungan di bawah menunjukkan hasil normalisasi atribut pertama dari fitur *Flow Duration.*

$$\text{normalisasi}(x_i) = \frac{x_i - mean(x)}{stdev(x)}$$

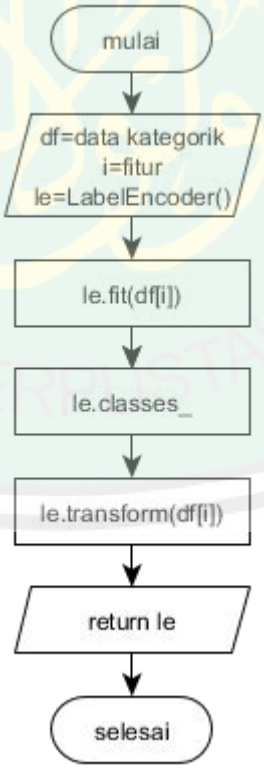
$$= \frac{3-16241648.53}{31524374.23}$$

$$= -0.515209133$$

4. Selanjutnya yaitu melakukan perhitungan yang sama pada semua atribut numerik yang ada pada set data CICIDS2017.

c) Normalisasi Atribut Kategorik

Atribut kategorik pada set data CICIDS2017 terdapat pada kolom Label. Pada kolom tersebut terdapat dua kategori yang ada yaitu kelas normal dan kelas serangan. Seperti yang diketahui, jaringan saraf tiruan tidak dapat mengenali data yang berbentuk teks yang dalam hal ini disebut atribut kategorik. Jadi sebelum jaringan saraf tiruan bekerja, atribut kategorik harus dinormalisasi menjadi atribut numerik terlebih dahulu. Dengan dilakukannya normalisasi, yang awalnya merupakan kelas normal akan berubah menjadi angka 0, sedangkan yang awalnya merupakan kelas serangan akan berubah menjadi angka 1. *Flowchart* normalisasi atribut kategorik ditunjukkan Gambar 3.5.



Gambar 3.5 *Flowchart* Normalisasi Atribut Kategorik

3.1.8 Seleksi Fitur

Dalam pemilihan fitur-fitur yang ada pada data latih dan data uji, algoritma yang digunakan yaitu *Random Forest*. Algoritma ini sering digunakan untuk pemilihan fitur karena strategi berbasis pohon yang digunakan secara alami memberi penilaian dengan seberapa baik mereka meningkatkan kemurnian simpul. Hal ini mengurangi rata-rata ketidakmurnian di semua pohon (disebut *Gini Impurity*). Simpul dengan penurunan ketidakmurnian terbesar terjadi di awal pohon, sementara catatan dengan penurunan ketidakmurnian paling sedikit terjadi di ujung pohon. Dengan demikian, dengan memangkas pohon di bawah simpul tertentu, kita dapat membuat bagian dari fitur yang paling penting.

Misalnya set data T berisi contoh-contoh dari n kelas, maka persamaan indeks Gini seperti pada persamaan 3.2.

$$Gini(T) = 1 - \sum_{j=1}^{n} (p_j)^2 \tag{3.2}$$

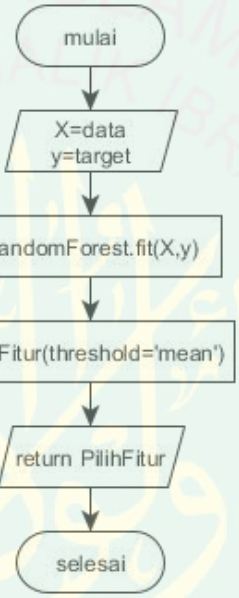
Jika set data T dibagi menjadi dua subset T1 dan T2 dengan ukuran masing-masing N1 dan N2, indeks Gini dari pemisahan data berisi contoh-contoh dari n kelas, maka indeks gini (T) didefinisikan seperti pada persamaan 3.3.

$$Gini(T) = \frac{N_1}{N} gini(T_1) + \frac{N_2}{N} gini(T_2) \tag{3.3}$$

Random Forest tidak menggunakan semua sampel pelatihan dalam konstruksi suatu pohon melainkan menyisakan seperangkat sampel *Out of Bag* (OOB), yang dapat digunakan untuk mengukur akurasi klasifikasi hutan. Untuk mengukur pentingnya fitur tertentu dalam struktur pohon, nilai-nilai fitur diacak dalam sampel OOB dan bandingkan akurasi klasifikasi antara sampel OOB yang

utuh dan sampel OOB dengan fitur tertentu yang dipermutasikan. (Hasan dkk, 2016)

Setelah menilai seberapa penting fitur yang ada pada set data dengan melakukan pelatihan pada setiap pohon, maka dipilih beberapa fitur yang dianggap penting. Fitur yang dipilih berdasarkan kaidah standar pemilihan fitur yaitu berdasarkan ambang batas rata-rata nilai dari setiap fitur. *Flowchart* seleksi fitur dengan menggunakan *Random Forest* ditunjukkan pada Gambar 3.6.



Gambar 3.6 *Flowchart* Seleksi Fitur dengan *Random Forest*

Berikut adalah contoh perhitungan manual dari seleksi fitur menggunakan *Gini Importance Index* pada *Random Forest*. Data yang digunakan yaitu data yang sebelumnya telah dinormalisasikan seperti yang dapat dilihat pada Tabel 3.4.

Tabel 3.4 Sampel Data untuk Seleksi Fitur

No.	Subflow Fwd Pkts	Fwd Act Data Pkts	Fwd Seg Size Avg	Fwd Pkt Len Max	Label
1	-0.186406	-0.188386	-0.314576	-0.285676	0
2	-0.251245	-0.269886	-0.314576	-0.285676	0
3	-0.251245	-0.269886	-0.314576	-0.285676	0
4	0.202627	0.219112	-0.312595	-0.278166	0

5	-0.121568	-0.106887	-0.309294	-0.278166	1
6	0.202627	0.219112	-0.312595	-0.278166	1

1. Misalnya fitur yang digunakan yaitu fitur *Fwd Pkt Len Max* (*T*). Data pada fitur tersebut dipisahkan. *T* memiliki 3 nilai (3/6) yang sama dengan -0.285676, 3 nilai (3/6) yang sama dengan -0.278166.
2. Untuk *T* == -0.285676 dan Label == 0, 3/3 nilai yang sama dengan 0.
3. Untuk *T* == -0.285676 dan Label == 1, 0/3 nilai yang sama dengan 1.
4. *Gini(T)* $= 1 - \left(\left(\frac{3}{3}\right)^2 + \left(\frac{0}{3}\right)^2\right)$

   $= 1 - (1 + 0)$

   $= 0$
5. Untuk *T* == -0.278166 dan Label == 0, 1/3 nilai yang sama dengan 0.
6. Untuk *T* == -0.278166 dan Label == 1, 2/3 nilai yang sama dengan 1.
7. *Gini(T)* $= 1 - \left(\left(\frac{1}{3}\right)^2 + \left(\frac{2}{3}\right)^2\right)$

   $= 1 - (0{,}11 + 0{,}44)$

   $= 1 - 0{,}55$

   $= 0{,}45$
8. Kemudian timbang dan jumlah masing-masing pemisahan berdasarkan pada proporsi data yang masing-masing terbagi.

   Gini(T) $= \left(\frac{3}{6} \cdot 0\right) + \left(\frac{3}{6} \cdot 0.45\right)$

   $= 0 + 0.225$

   $= 0.225$

9. Diperoleh nilai atau skor dari fitur *Fwd Pkt Len Max* berdasarkan perhitungan *Gini Importance Index* yaitu sebesar 0.225.
10. Selanjutnya yaitu melakukan perhitungan yang sama pada setiap fitur dari data yang digunakan.

3.1.9 Pelatihan dan Pengujian dengan GRNN

Dasar teoritis dari *General Regression Neural Network* adalah analisis regresi nonlinier. Inti dari analisis regresi, yang terdiri dari variabel independen Y dan variabel dependen X, adalah menghitung yang memiliki nilai probabilitas maksimum. Diasumsikan bahwa $f(x,y)$ adalah fungsi kepadatan probabilitas gabungan dari variabel acak x dan y, nilai yang diamati dari x adalah X. Dengan demikian, regresi y relatif terhadap X adalah:

$$\hat{Y} = E(y \mid x) = \frac{\int_{+\infty}^{+\infty} y f(x,y)\,dy}{\int_{+\infty}^{+\infty} f(x,y)\,dy} \tag{3.4}$$

Dimana $\hat{Y}$ adalah keluaran perkiraan Y ketika masukan adalah X.

Fungsi kerapatan $\hat{f}(X,y)$ dapat diperkirakan berdasarkan kumpulan data sampel $\{x_i,y_i\}_{i=1}^{n}$ menggunakan estimasi Parzen non-parametrik:

$$\hat{f}(X,y) = \frac{1}{n(2\pi)^{\frac{p+1}{2}}\sigma^{p+1}} \sum_{i=1}^{n} \exp\left[-\frac{(X-X_i)^T(X-X_i)}{2\sigma^2}\right] \exp\left[-\frac{(X-Y_i)^2}{2\sigma^2}\right] \tag{3.5}$$

Dimana X_i dan Y_i adalah nilai-nilai yang diamati dari variabel acak x dan y; n adalah ukuran sampel; p adalah dimensi x; dan σ adalah koefisien lebar fungsi Gaussian yang disebut simpangan baku atau deviasi standar (sigma).

$f(x,y)$ diganti dengan $\hat{f}(X,y)$ dalam persamaan 3.4, serta perintah integral dan penambahan ditukarkan:

$$\hat{Y}(X)=\frac{\sum_{i=1}^{n}\exp\left[-\frac{(X-X_i)^T(X-X_i)}{2\sigma^2}\right]\int_{+\infty}^{+\infty}y\exp\left[-\frac{(Y-Y_i)^2}{2\sigma^2}\right]dy}{\sum_{i=1}^{n}\exp\left[-\frac{(X-X_i)^T(X-X_i)}{2\sigma^2}\right]\int_{+\infty}^{+\infty}\exp\left[-\frac{(Y-Y_i)^2}{2\sigma^2}\right]dy} \quad (3.6)$$

Karena $\int_{-\infty}^{+\infty} ze^{-z^2}\,dz = 0$, keluaran $\hat{Y}(X)$ dari jaringan setelah dua integral dihitung dapat dilihat pada persamaan 3.7 yang merupakan persamaan dari *General Regression Neural Network* itu sendiri.

$$\hat{Y}(X)=\frac{\sum_{i=1}^{n}Y_i\exp\left[-\frac{(X-X_i)^T(X-X_i)}{2\sigma^2}\right]}{\sum_{i=1}^{n}\exp\left[-\frac{(X-X_i)^T(X-X_i)}{2\sigma^2}\right]} \quad (3.7)$$

Dimana:

1. X adalah sampel masukan dan X_i adalah sampel pengujian.
2. Keluaran dari sampel masukan *i* adalah Y_i.
3. $(X-X_i)^T(X-X_i)$ adalah jarak Euclidean dari X dan X_i.
4. $\exp\left[-\frac{(X-X_i)^T(X-X_i)}{2\sigma^2}\right]$ merupakan fungsi aktivasi *radial basis function*. Sederhananya, fungsi aktivasi ini secara teori merupakan bobot untuk sampel masukan.
5. Nilai dari $(X-X_i)^T(X-X_i)$ menandakan seberapa banyak sampel pelatihan dapat berkontribusi pada keluaran dari sampel uji tertentu.
6. Jika $(X-X_i)^T(X-X_i)$ memiliki yang nilai kecil, berarti akan memberikan kontribusi nilai lebih untuk keluaran. Tetapi jika memiliki nilai yang besar, berarti akan memberikan kontribusi lebih sedikit ke keluaran.

7. Adapun $\exp\left[-\frac{(X-X_i)^T(X-X_i)}{2\sigma^2}\right]$ menentukan seberapa banyak berat sampel pelatihan akan berkontribusi.
8. Jika $(X-X_i)^T(X-X_i)$ menghasilkan nilai kecil, maka $\exp\left[-\frac{(X-X_i)^T(X-X_i)}{2\sigma^2}\right]$ mengembalikan nilai yang relatif besar dan begitupun sebaliknya.
9. Jika $(X-X_i)^T(X-X_i)$ menghasilkan nilai 0, maka $\exp\left[-\frac{(X-X_i)^T(X-X_i)}{2\sigma^2}\right]$ mengembalikan nilai 1 yang berarti data uji sama dengan sampel pelatihan dan keluaran dari data uji juga merupakan keluaran dari sampel pelatihan.

a. Simulasi Menggunakan Matlab

Simulasi perhitungan dilakukan dengan menggunakan aplikasi Matlab. Data yang digunakan yaitu data yang digunakan pada tahap seleksi fitur. Misalnya terpilih tiga fitur yaitu Fwd Act Data Packets, Fwd Seg Size Avg, dan Fwd Pkt Len Max sehingga data menjadi seperti pada Tabel 3.5.

Tabel 3.5 Sampel Data Latih untuk Pelatihan

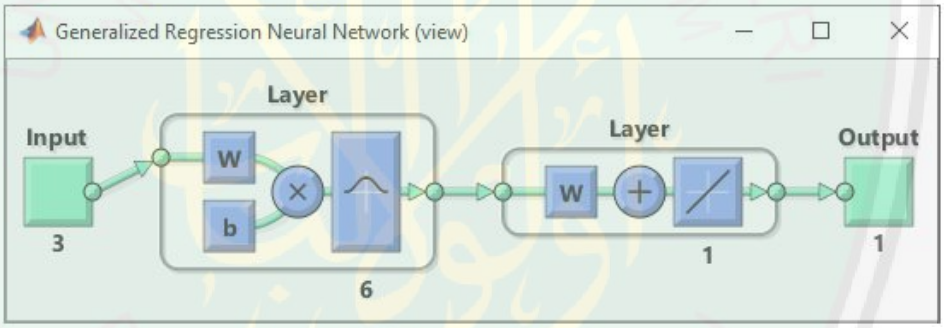
No.	Fwd Act Data Pkts	Fwd Seg Size Avg	Fwd Pkt Len Max	Label
1	-0.188386	-0.314576	-0.285676	0
2	-0.269886	-0.314576	-0.285676	0
3	-0.269886	-0.314576	-0.285676	0
4	0.219112	-0.312595	-0.278166	0
5	-0.106887	-0.309294	-0.278166	1
6	0.219112	-0.312595	-0.278166	1

Tabel 3.5 menunjukkan sampel data latih yang digunakan untuk pelatihan, sedangkan untuk sampel data uji yang digunakan untuk pengujian dapat dilihat pada Tabel 3.6.

Tabel 3.6 Sampel Data Uji untuk Pengujian

No.	Fwd Act Data Pkts	Fwd Seg Size Avg	Fwd Pkt Len Max	Label
1	-0.269886	-0.269886	-0.285676	0
2	-0.106887	-0.309294	-0.278166	1

Simulasi dilakukan dengan memanfaatkan fungsi *newgrnn()* yang merupakan fungsi *General Regression Neural Network* yang telah disediakan oleh Matlab. Dari fungsi tersebut, dihasilkan arsiektur *General Regression Neural Network* seperti yang dapat dilihat pada Gambar 3.7.



Gambar 3.7 *Arsitektur General Regression Neural Network* pada Matlab

Pada lapisan pertama yaitu lapisan masukan terdapat 3 neuron yang mana merupakan jumlah fitur kolom pada data kecuali fitur target. Pada lapisan kedua atau lapisan tersembunyi, *w* merupakan bobot, *b* merupakan bias, sedangkan 6 merupakan jumlah baris data. Lapisan ini memiliki fungsi *radial basis*. Setiap masukan tertimbang neuron adalah jarak antara vektor masukan dan vektor bobotnya, yang dihitung menggunakan persamaan jarak Euclidean. Setiap masukan neuron adalah produk dari masukan tertimbangnya dengan biasnya. Lapisan ketiga yaitu lapisan penjumlahan menyimpan bobot namun tidak

menyimpan bias tidak seperti pada lapisan sebelumnya. Lapisan ini memiliki fungsi *purelin.* Pada lapisan keluaran, setiap keluaran neuron adalah masukan bersihnya yang melewati fungsi *radial basis.* Jika vektor bobot neuron sama dengan vektor masukan (ditransposisikan), masukan tertimbangnya adalah 0, masukan bersihnya adalah 0, dan keluarannya adalah 1.

Dari proses pelatihan, diperoleh hasil prediksi yaitu [0,0,0,0,0,0] dengan akurasi klasifikasi sebesar 66,67%. Sedangkan dari proses pengujian, diperoleh hasil prediksi yaitu [0,0] dengan akurasi klasifikasi sebesar 50%.

b. Skema Pelatihan dan Pengujian

Pelatihan dilakukan dengan mengimplementasikan teknik *Cross Validation* untuk mendapatkan parameter sigma (σ) terbaik. Jumlah iterasi yang diterapkan yaitu sebanyak 15 kali dengan penambahan sigma sebesar 0.1 di setiap iterasi. Setelah mendapatkan sigma terbaik, maka selanjutnya dilakukan pengujian. Percobaan dilakukan sebanyak dua kali, yaitu pelatihan dan pengujian dengan fitur lengkap serta pelatihan dan pengujian dengan fitur terpilih.

3.1.10 Evaluasi

Evaluasi dilakukan dengan menggunakan *Confusion Matrix* yang merupakan metode yang biasanya digunakan dalam evaluasi model pada kasus klasifikasi untuk menghitung tingkat *accuracy*, *precision, recall*, dan *f1-Score.*

Accuracy mengukur proporsi jumlah total klasifikasi yang benar. Rumus dari *accuracy* ditunjukkan pada persamaan 3.8.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100\% \qquad (3.8)$$

Precision mengukur jumlah klasifikasi yang benar dibandingkan dengan jumlah klasifikasi yang salah. Rumus *precision* ditunjukkan pada persamaan 3.9.

$$Precision = \frac{TP}{TP + FP} \times 100\% \quad (3.9)$$

Recall mengukur jumlah klasifikasi yang benar dibandingkan dengan jumlah entri yang terlewat. Rumus dari *recall* ditunjukkan pada persamaan 3.10.

$$Recall = \frac{TP}{TP + FN} \times 100\% \quad (3.10)$$

F1-score mengukur rata-rata *precision* dan *recall*, yang berfungsi sebagai pengukuran efektivitas. Rumus dari *F1-score* ditunjukkan pada persamaan 3.11.

$$F1\text{-}score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \times 100\% \quad (3.11)$$

Untuk lebih mudah memahami, berikut adalah tabel kebenaran antara nilai sebenarnya dan nilai prediksi seperti yang ditunjukkan pada Tabel 3.5.

Tabel 3.7 Tabel Kebenaran Antara Nilai Sebenarnya dan Nilai Prediksi

Nilai Sebenarnya/ Nilai Prediksi	**Normal**	**Serangan**
Normal	TP	FP
Serangan	FN	TN

Dimana:

- *True Positive* (TP) adalah kelas serangan yang diklasifikasikan dengan benar sebagai serangan.
- *False Positive* (FP) adalah kelas normal yang salah diklasifikasikan sebagai serangan.
- *True Negative* (TN) adalah kelas normal yang diklasifikasikan dengan benar sebagai normal.
- *False Negative* (FN) adalah kelas serangan yang salah diklasifikasikan sebagai normal.

BAB IV

UJI COBA DAN PEMBAHASAN

4.1 Kebutuhan Perangkat

Sebelum dipaparkan proses dan hasil uji coba yang telah dilakukan, perlu diketahui spesifikasi perangkat keras maupun perangkat lunak yang digunakan, yaitu sebagai berikut:

a) Spesifikasi perangkat keras

- Sistem Operasi: Windows 10 Home 64-bit
- Prosesor: Intel(R) Core(TM) i3-6006U CPU @ 2.00GHz
- Memori: 8192 MB RAM

b) Spesifikasi perangkat lunak

- Google Colaboratory
  - Prosesor: Intel(R) Xeon(R) CPU @ 2.30GHz
  - Memori: 12617208 kB RAM
- Python 3
- Oracle VM VirtualBox
- Kali Linux 2
- Metasploitable 2
- Metasploit Freamework
- Low Orbit Ion Cannon (LOIC)
- Wireshark
- InjelliJ IDEA
- CICFlowMeter 4.0

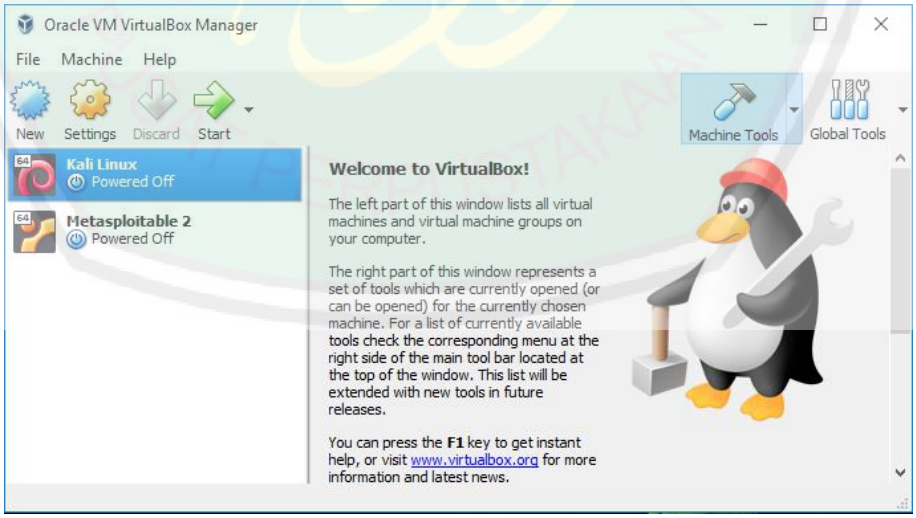
4.2 Uji Coba dan Pembahasan

Uji coba adalah proses pengerjaan atau implementasi dari setiap rancangan sistem yang telah dipaparkan pada bab sebelumnya. Pada sub bab ini akan dijelaskan proses uji coba yang disertai dengan pembahasan dari hasil yang didapatkan.

4.2.1 Implementasi Membuat *Virtual Hacking Lab*

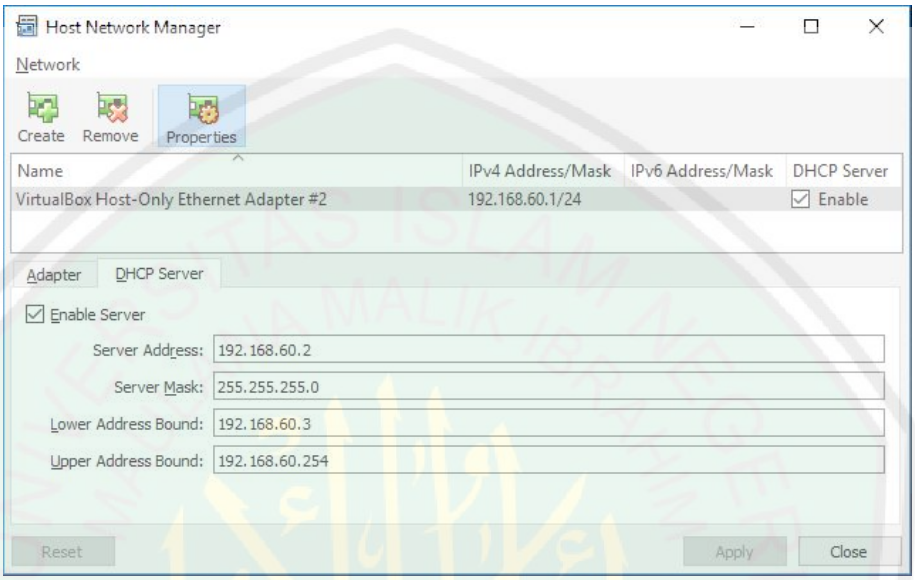
Sebelum melakukan simulasi serangan, dibutuhkan sebuah *virtual hacking lab* sederhana. *virtual hacking lab* adalah laboratorium pengujian penetrasi yang dirancang untuk mempelajari sisi praktis penilaian kerentanan dan pengujian penetrasi dalam lingkungan yang aman.

Pada penelitian ini, digunakan Oracle VM VirtualBox sebagai mesin virtual yang menghubungkan anatara sistem operasi penyerang dan target. Pada Gambar 4.1, menunjukkan sistem operasi Kali Linux 2 yang bertindak sebagai penyerang dan Metasploitable 2 yang bertindak sebagai server web atau target serangan yang telah terpasang pada Oracle VM VirtualBox.



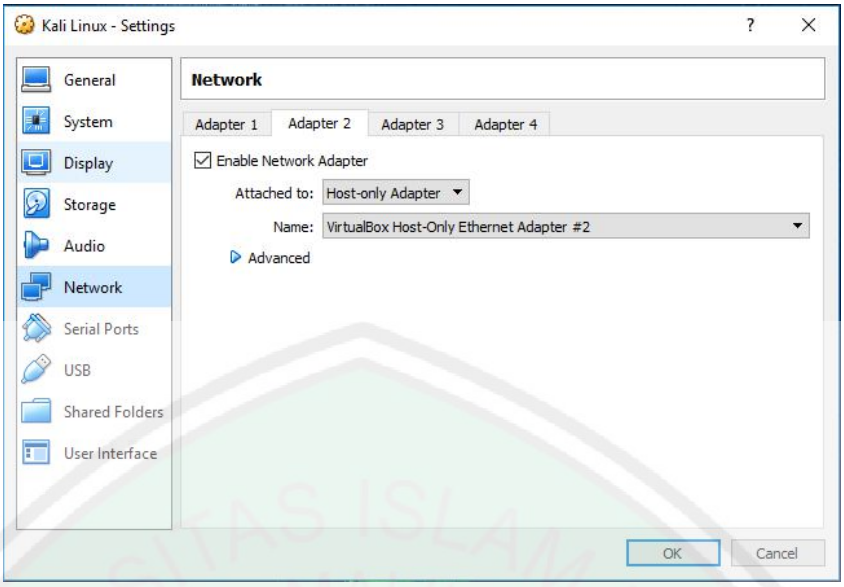
Gambar 4.1 Tampilan Awal Oracle VM VirtualBox

Sebelum menjalankan Kali Linux 2 dan Metasploitable 2, perlu dilakukan konfigurasi alamat IP terlebih dahulu agar penyerang dan target berada dalam satu jaringan dan saling terhubung oleh sebab itu dibutuhkan *Host-Only Ethernet Adapter* dengan konfigurasi seperti pada Gambar 4.2.



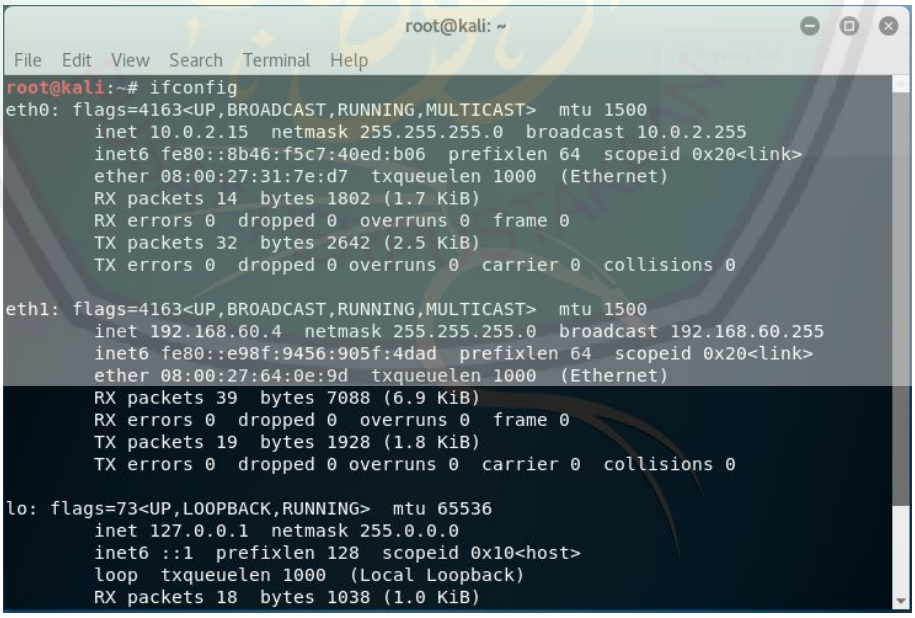
Gambar 4.2 Konfigurasi VirtualBox *Host-Only Ethernet Adapter*

Selanjutnya perlu mengaktifkan *network adapter* terlebih dahulu yang ada pada menu *Settings* > *Network* seperti yang terlihat pada Gambar 4.3. Pengaturan ini dilakukan pada keduanya, baik Kali Linux 2 dan Metasploitable 2.



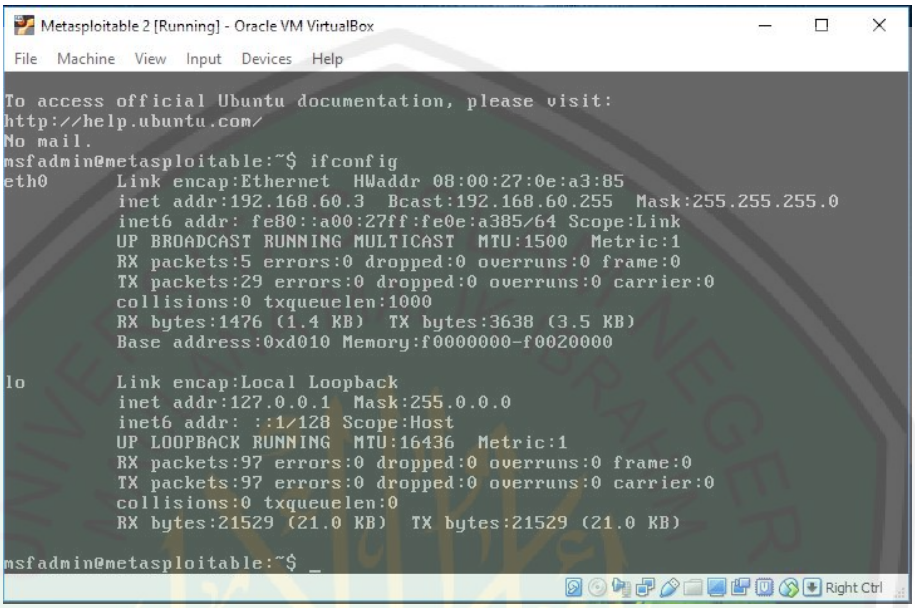
Gambar 4.3 Mengaktifkan *Network Adapter*

Ketika dijalankan, Kali Linux 2 tidak secara otomatis terhubung pada *Host-Only Ethernet Adapter*. Oleh karena dilakukan pengaturan pada koneksi dimana koneksi statis diganti menjadi dinamis (DHCP). Dengan demikian Kali Linux 2 secara otomatis terhubung dengan *Host-Only Ethernet Adapter* dan memperoleh alamat IP yaitu 192.168.60.4 seperti yang terlihat pada Gambar 4.4.



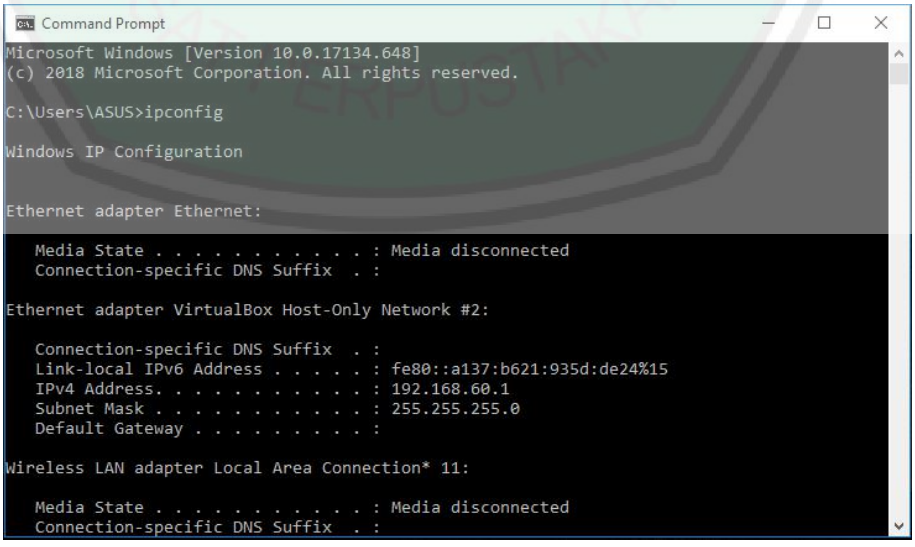
Gambar 4.4 Alamat IP Kali Linux 2

Berbeda dengan Kali Linux 2, Metasploitable 2 tidak memerlukan konfigurasi. Secara otomatis, Metasploitable 2 terhubung pada *Host-Only Ethernet Adapter* dengan memperoleh alamat IP yaitu 192.168.60.3 seperti yang terlihat pada Gambar 4.5.



Gambar 4.5 Alamat IP Metasploitable 2

Adapun komputer *host* dengan sistem operasi Windows 10, memperoleh alamat IP dari *Host-Only Network Adapter* yaitu 192.168.60.1 seperti yang terlihat pada Gambar 4.6.

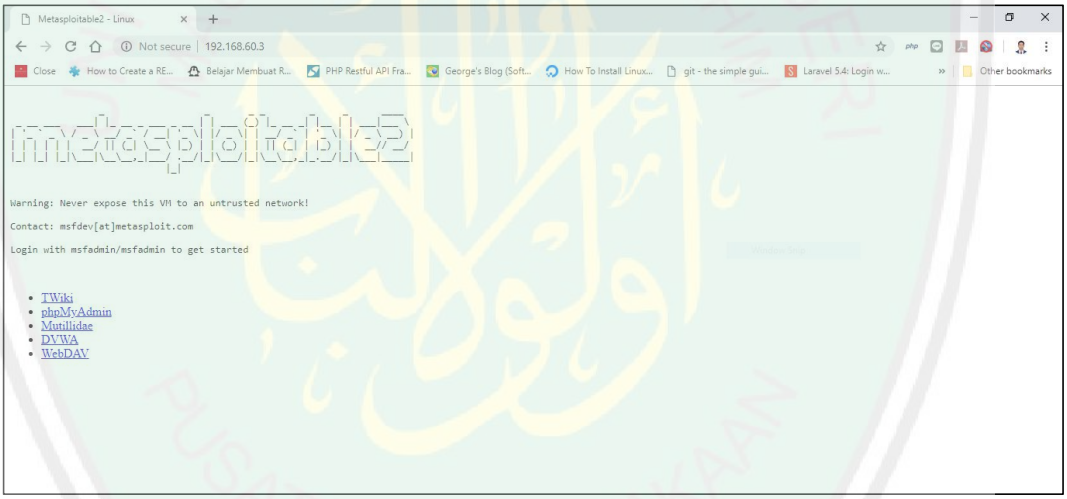


Gambar 4.6 Alamat IP Windows 10 Sebagai Komputer *Host*

Dengan demikian Windows 10 sebagai *host*, Kali Linux 2 sebagai penyerang, dan Metasploitable 2 sebagai server web alias target telah membentuk sebuah *virtual hacking lab* sederhana yang saling terhubung satu sama lain.

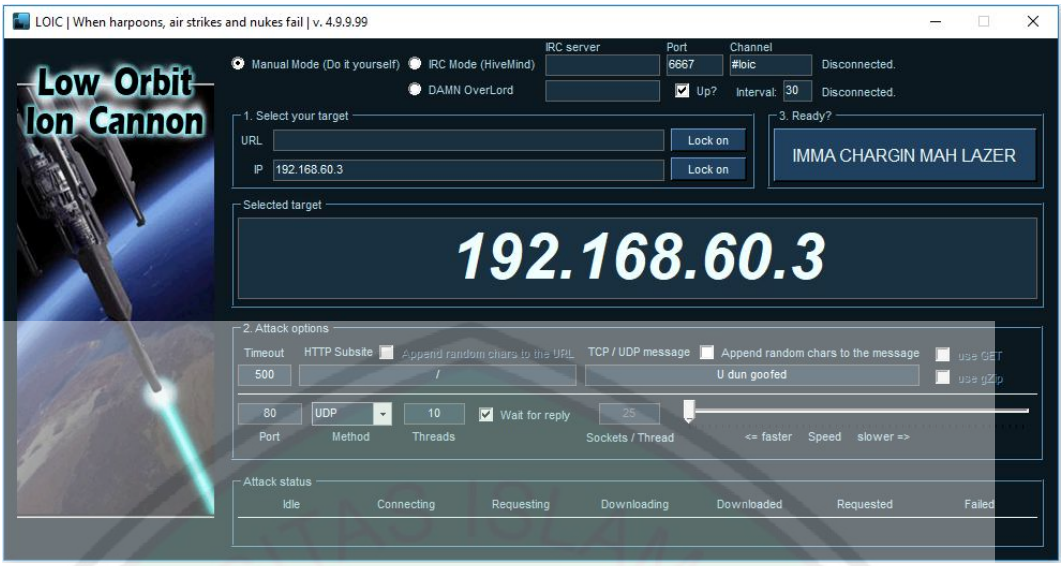
4.2.2 Implementasi Simulasi Serangan DDoS

Serangan DDoS dilakukan dengan menggunakan aplikasi Metasploit *Framework* yang di jalankan pada sistem operasi Kali Linux 2 dan *Low Orbit Ion Cannon* (LOIC) yang dijalankan pada sistem operasi Windows 10. Kedua aplikasi ini dijalankan secara bersamaan dengan menyerang target yang sama sehingga mampu mengirim dan membanjiri jaringan dengan paket yang sangat banyak. Gambar 4.7 menujukkan tampilan web sebelum serangan dilakukan.



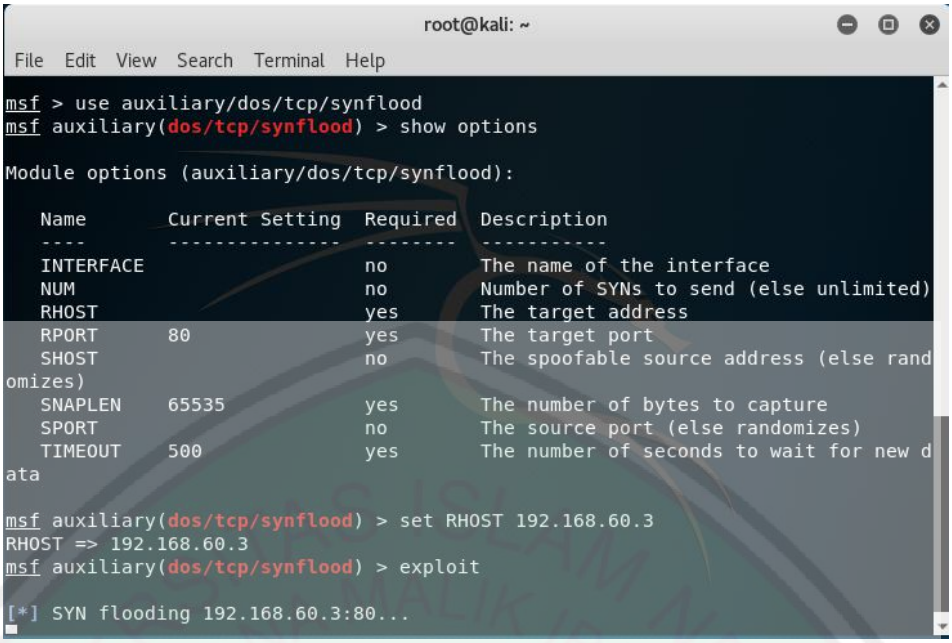
Gambar 4.7 Tampilan Web Target

Pada *Low Orbit Ion Cannon* (LOIC), yang pertama dilakukan ialah memasukkan alamat URL ataupun alamat IP target lalu kemudian klik *Lock on.* Setelah itu memilih *Method* dan *Port,* lalu memasukkan jumlah *Threads* dan *Timeout* seperti yang dapat dilihat pada Gambar 4.8. Setelah konfigurasi dilakukan, selanjutnya klik tombol *IMMA CHARGIN MAH LAZER* pada sisi kanan atas antarmuka untuk memulai serangan.



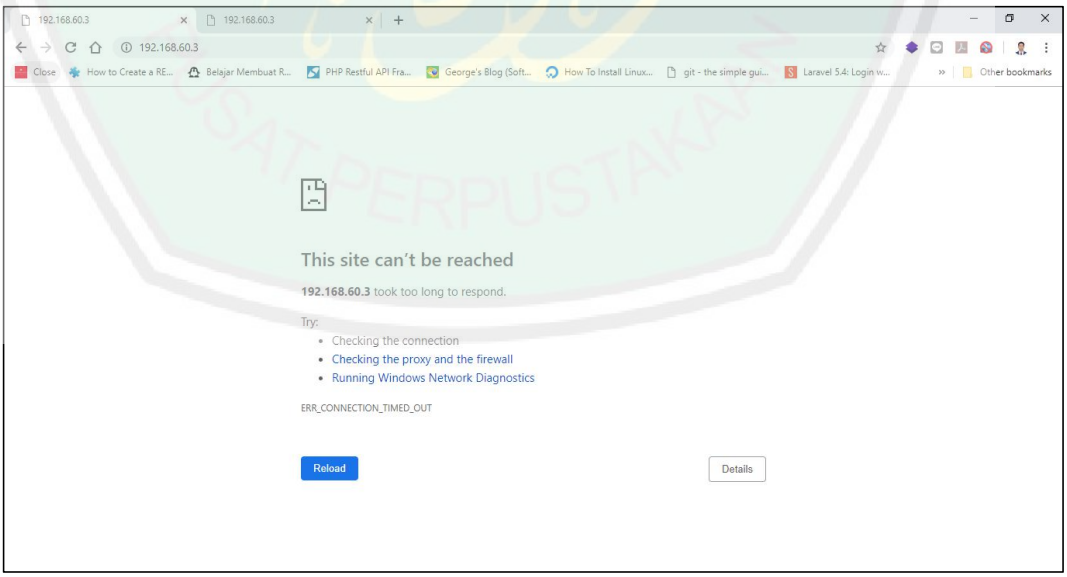
Gambar 4.8 Serangan DoS dengan *Low Orbit Ion Cannon* (LOIC)

Secara bersamaan, Metasploit *Framework* juga dijalankan untuk menyerang target yang sama. Yang pertama dilakukan ialah menjalankan Metasploit *Framework* dengan mengetik perintah *msfconsole* pada terminal Kali Linux 2. Pada *console* Metasploit *Framework*, ketik *use auxiliary/dos/tcp/synflood* kemudian menampilkan opsi parameter yang digunakan dalam melakukan serangan DoS, yaitu dengan cara mengetik perintah *show options*. Dalam hal ini, parameter yang perlu ditambahkan atau diisi adalah parameter RHOST karena parameter lainnya yang diperlukan sudah terisi otomatis dengan nilai standar. Setelah semua parameter yang diperlukan terisi, serangan siap dilakukan dengan mengetik *exploit* pada *console* lalu tekan enter untuk memulai serangan. Serangan pun diluncurkan ke target seperti yang dapat dilihat pada Gambar 4.9.



Gambar 4.9 Serangan DoS dengan Menggunakan Metasploit *Framework*

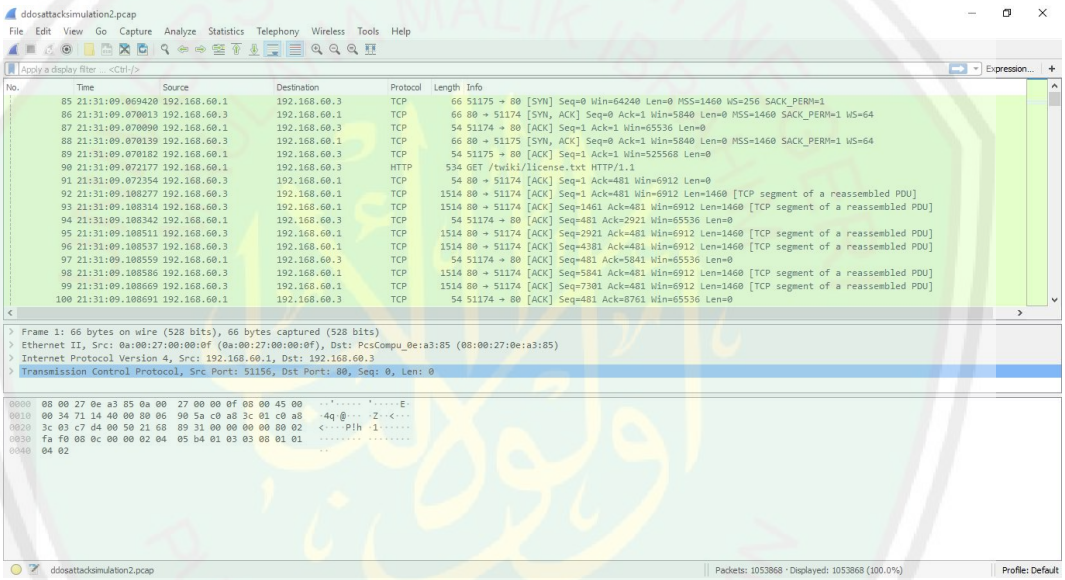
Dengan diluncurkannya serangan DoS dari dua sistem dan perangkat yang berbeda selama kurang lebih 20 menit, sehingga menyebabkan serangan DDoS yang melumpuhkan server serta membuat server menjadi tidak responsif dengan seketika. Gambar 4.10 menunjukkan web tidak dapat diakses dan menandakan serangan DDoS yang dilakukan berhasil.



Gambar 4.10 Server Lumpuh dan Web Tidak Dapat Diakses

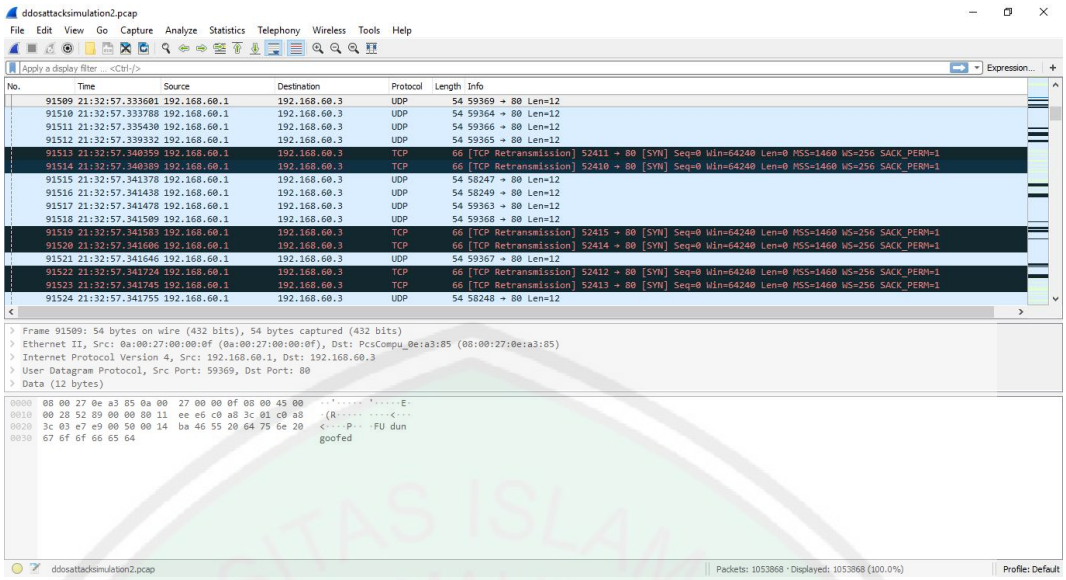
4.2.3 Implementasi Merekam Log Jaringan

Untuk mendapatkan log jaringan, dibutuhkan sebuah aplikasi bernama Wireshark. Ketika menjalankan Wireshark, yang dilakukan pertama kali yaitu memilih *interface* jaringan dimana dalam hal ini *interface* yang dipilih yaitu *VirtualBox Host-Only Network* sehingga lalu lintas jaringan yang direkam yaitu hanya lalu lintas jaringan yang melalui *interface* tersebut saja. Sebelum serangan DDoS dimulai, Wireshark sudah berjalan terlebih dahulu dan merekam lalu lintas normal pada jaringan seperti yang terlihat pada Gambar 4.11.



Gambar 4.11 Lalu Lintas Jaringan Sebelum Serangan Dilakukan

Ketika serangan dimulai yang bersumber dari Metasploit *Framework* dan *Low Orbit Ion Cannon* (LOIC), seketika Wireshark mulai sibuk merekam lalu lintas jaringan yang menerima puluhan bahkan ratusan paket perdetiknya seperti yang rafik



Gambar 4.12 Lalu Lintas Jaringan Disaat Serangan Dilakukan

Perekaman dilakukan dengan durasi kurang lebih 20 menit dan merekam lalu lintas normal maupun lalu lintas serangan. Dengan durasi selama itu, baris data yang terekam yaitu sebanyak 1.053.868 baris. Setelah itu perekaman dihentikan lalu datanya disimpan dalam format PCAP. Tabel 4.1 menunjukkan hasil perekaman yang dilakukan dengan skema waktu serangan yang telah ditetapkan .

Tabel 4.1 Hasil Perekaman Lalu Lintas Jaringan

Menit	Kelas	Baris Data
2	Normal	796
1	Serangan	153913
2	Normal	1460
1	Serangan	143520
2	Normal	143
1	Serangan	153605
2	Normal	3335
1	Serangan	156429
2	Normal	1638
1	Serangan	150243
2	Normal	1526

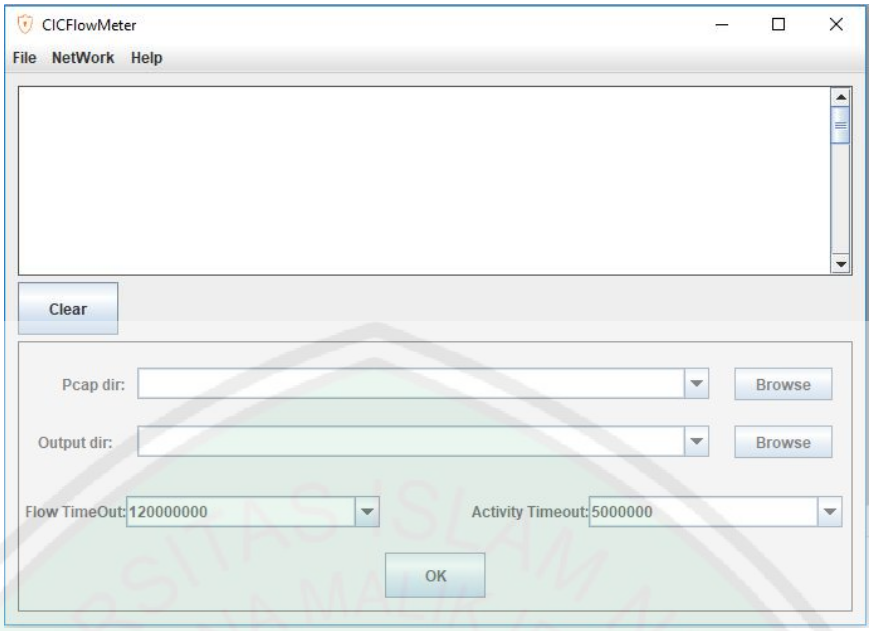
1	Serangan	285687
2	Normal	1573

4.2.4 Implementasi Ekstraksi Log Jaringan

Ekstraksi dilakukan dengan tujuan menghasilkan data baru dari data log jaringan yang didapatkan sebelumnya, menjadi data yang dapat diolah oleh *Machine Learning.* Sebelum menjalankan CICFlowMeter 4.0, terlebih dahulu perlu lakukan instalasi jnetpcap *local repo* dengan perintah seperti pada Gambar 4.13.

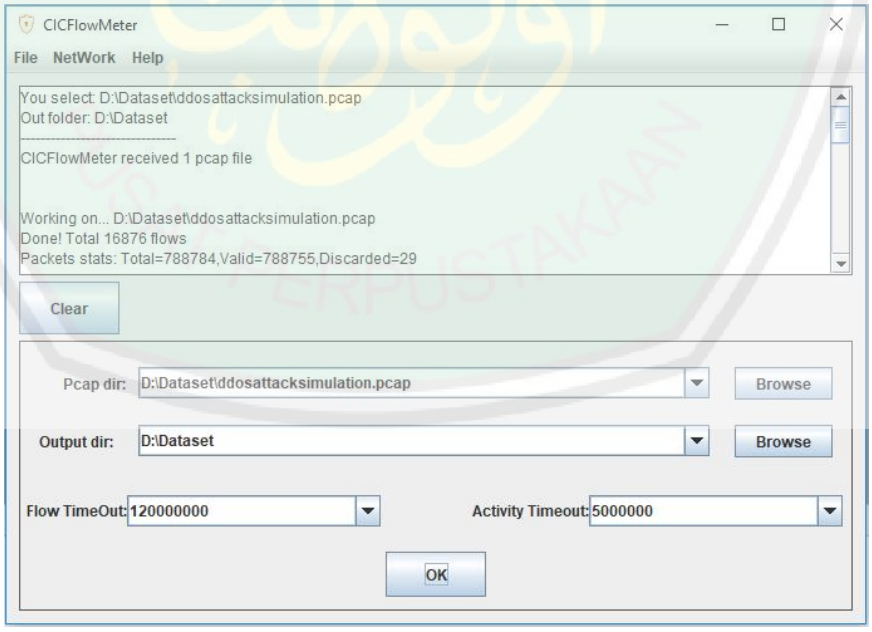
Gambar 4.13 Instalasi Jnetpcap *Local Repo*

Setelah instalasi jnetpcap selesai, CICFlowMeter 4.0 yang telah diunduh dari situs Github kemudian jalankan perintah *gradlew execute* melalui terminal IntelliJ IDEA, lalu klik *Network* > *Offline* sehingga muncul antarmuka seperti pada Gambar 4.14.



Gambar 4.14 Antarmuka CICFlowMeter 4.0

Klik *Browse Pcap dir* dan cari data PCAP untuk diekstrak lalu klik *Browse Output dir* untuk menentukan direktori keluaran dari hasil ekstraksi dengan format data yaitu CSV. Setelah proses ekstrasi telah selesai, maka muncul pemberitahuan di *dialog box* seperti pada Gambar 4.15.



Gambar 4.15 Proses Ekstraksi dengan CICFlowMeter 4.0

Dari proses ekstraksi yang dilakukan, didapatkan data dengan format CSV. Data tersebut memiliki 84 fitur sama seperti fitur-fitur yang ada pada data latih dengan 31167 baris data. Tabel 4.2 menunjukkan detail hasil ekstraksi log jaringan yang didapatkan sebelumnya.

Tabel 4.2 Hasil Ekstraksi Log Jaringan

Menit	Kelas	Baris Data	Hasil Ekstraksi
2	Normal	796	41
1	Serangan	153913	3309
2	Normal	1460	399
1	Serangan	143520	20
2	Normal	143	17
1	Serangan	153605	6454
2	Normal	3335	798
1	Serangan	156429	3819
2	Normal	1638	585
1	Serangan	150243	5297
2	Normal	1526	404
1	Serangan	285687	9509
2	Normal	1573	515

4.2.5 Implementasi Pelabelan

Dari hasil ekstraksi yang sebelumnya dilakukan, didapatkan data baru dengan format CSV dengan jumlah fitur sebanyak 84 dan jumlah baris data sebanyak 31167. Ekstraksi yang dilakukan tidak mengklasifikasikan baris data, mana yang termasuk kelas normal ataupun kelas serangan. Oleh sebab itu perlu dilakukan pelabelan.

Pelabelan dilakukan secara manual dengan berpatokan pada skema waktu simulasi serangan. Sebanyak 2779 (8.9%) baris data untuk kelas normal dan sebanyak 28388 (91.1%) baris data untuk kelas serangan. Karena jumlah antara data kedua kelas terpaut jauh, maka dilakukan reduksi agar data kedua kelas

tersebut menjadi seimbang. Setelah dilakukan reduksi, didapatkan data untuk kelas normal sebanyak 2779 (53.1%) baris data dan data untuk kelas serangan sebanyak 3147 (46.9%) baris data. Kelas normal kemudian diberi label BENIGN dan kelas serangan diberi label DDoS, seperti halnya yang ada pada data latih.

4.2.6 Implementasi Memuat Data Latih dan Data Uji

Variabel *datatrain* diinisiasikan untuk data latih dan variabel *datatest* diinisiasikan untuk data uji. Karena kedua data yang digunakan belum memiliki *header,* maka kedua data diberikan nama kolom yang sama yang dijadikan sebagai *header.* Agar sistem dapat membaca data dalam format CSV, maka digunakan *library* Pandas dengan fungsi *read_csv()* sehingga membentuk sebuah *dataframe*. Lebih jelasnya dapat dilihat pada Gambar 4.16.

```
datatrain = 'ddoscicids2017.csv'
datatest = 'ddosattacksimulationfix_manual.csv'

col_names = [
    'flow_id','source_ip','source_port','destination_ip','destination_port','protocol','timestamp','flow_duration',
    'total_fwd_packets','total_bwd_packets','total_length_of_fwd_packets','total_length_of_bwd_packets',
    'fwd_packet_length_max','fwd_packet_length_min','fwd_packet_length_mean','fwd_packet_length_std',
    'bwd_packet_length_max','bwd_packet_length_min','bwd_packet_length_mean','bwd_packet_length_std',
    'flow_bytes','flow_packets','flow_iat_mean','flow_iat_std','flow_iat_max','flow_iat_min',
    'fwd_iat_total','fwd_iat_mean','fwd_iat_std','fwd_iat_max','fwd_iat_min','bwd_iat_total','bwd_iat_mean',
    'bwd_iat_std','bwd_iat_max','bwd_iat_min','fwd_psh_flags','bwd_psh_flags','fwd_urg_flags','bwd_urg_flags',
    'fwd_header_length','bwd_header_length','fwd_packets','bwd_packets','min_packet_length','max_packet_length',
    'packet_length_mean','packet_length_std','packet_length_variance','fin_flag_count','syn_flag_count',
    'rst_flag_count','psh_flag_count','ack_flag_count','urg_flag_count','cwe_flag_count','ece_flag_count',
    'down_up_ratio','average_packet_size','avg_fwd_segment_size','avg_bwd_segment_size','fwd_avg_bytes_bulk',
    'fwd_avg_packets_bulk','fwd_avg_bulk_rate','bwd_avg_bytes_bulk','bwd_avg_packets_bulk','bwd_avg_bulk_rate',
    'subflow_fwd_packets','subflow_fwd_bytes','subflow_bwd_packets','subflow_bwd_bytes','init_win_bytes_forward',
    'init_win_bytes_backward','act_data_pkt_fwd','min_seg_size_forward','active_mean','active_std','active_max',
    'active_min','idle_mean','idle_std','idle_max','idle_min','label'
]

df_train = pd.read_csv(datatrain, names = col_names, skiprows = 1)
df_test = pd.read_csv(datatest, names = col_names, skiprows = 1)

print('Dimensi Data Latih:', df_train.shape)
print('Dimensi Data Uji:', df_test.shape)
```

```
Dimensi Data Latih: (225745, 84)
Dimensi Data Uji: (5926, 84)
```

Gambar 4.16 Memuat Data Latih dan Data Uji

Tabel 4.3 Penamaan Ulang Fitur

No.	Nama Fitur	Penamaan Ulang
1.	Flow ID	flow_id
2.	Src IP	source_ip
3.	Src Port	source_port
4.	Dst IP	destination_ip

5.	Dst Port	destination_port
6.	Protocol	protocol
7.	Timestamp	timestamp
8.	Flow Duration	flow_duration
9.	Tot Fwd Pkts	total_fwd_packets
10.	Tot Bwd Pkts	total_bwd_packets
11.	TotLen Fwd Pkts	total_length_of_fwd_packets
12.	TotLen Bwd Pkts	total_length_of_bwd_packets
13.	Fwd Pkt Len Max	fwd_packet_length_max
14.	Fwd Pkt Len Min	fwd_packet_length_min
15,	Fwd Pkt Len Mean	fwd_packet_length_mean
16.	Fwd Pkt Len Std	fwd_packet_length_std
17.	Bwd Pkt Len Max	bwd_packet_length_max
18.	Bwd Pkt Len Min	bwd_packet_length_min
19.	Bwd Pkt Len Mean	bwd_packet_length_mean
20.	Bwd Pkt Len Std	bwd_packet_length_std
21.	Flow Byts/s	flow_bytes
22.	Flow Pkts/s	flow_packets
23.	Flow IAT Mean	flow_iat_mean
24.	Flow IAT Std	flow_iat_std
25.	Flow IAT Max	flow_iat_max
26.	Flow IAT Min	flow_iat_min
27.	Fwd IAT Tot	fwd_iat_total
28.	Fwd IAT Mean	fwd_iat_mean
29.	Fwd IAT Std	fwd_iat_std
30.	Fwd IAT Max	fwd_iat_max
31.	Fwd IAT Min	fwd_iat_min
32.	Bwd IAT Tot	bwd_iat_total
33.	Bwd IAT Mean	bwd_iat_mean
34.	Bwd IAT Std	bwd_iat_std
35.	Bwd IAT Max	bwd_iat_max
36.	Bwd IAT Min	bwd_iat_min
37.	Fwd PSH Flags	fwd_psh_flags
38.	Bwd PSH Flags	bwd_psh_flags
39.	Fwd URG Flags	fwd_urg_flags
40.	Bwd URG Flags	bwd_urg_flags

41.	Fwd Header Len	fwd_header_length
42.	Bwd Header Len	bwd_header_length
43.	Fwd Pkts/s	fwd_packets
44.	Bwd Pkts/s	bwd_packets
45.	Pkt Len Min	min_packet_length
46.	Pkt Len Max	max_packet_length
47.	Pkt Len Mean	packet_length_mean
48.	Pkt Len Std	packet_length_std
49.	Pkt Len Var	packet_length_variance
50.	FIN Flag Cnt	fin_flag_count
51.	SYN Flag Cnt	syn_flag_count
52.	RST Flag Cnt	rst_flag_count
53.	PSH Flag Cnt	psh_flag_count
54.	ACK Flag Cnt	ack_flag_count
55.	URG Flag Cnt	urg_flag_count
56.	CWE Flag Count	cwe_flag_count
57.	ECE Flag Cnt	ece_flag_count
58.	Down/Up Ratio	down_up_ratio
59.	Pkt Size Avg	average_packet_size
60.	Fwd Seg Size Avg	avg_fwd_segment_size
61.	Bwd Seg Size Avg	avg_bwd_segment_size
62.	Fwd Byts/b Avg	fwd_avg_bytes_bulk
63.	Fwd Pkts/b Avg	fwd_avg_packets_bulk
64.	Fwd Blk Rate Avg	fwd_avg_bulk_rate
65.	Bwd Byts/b Avg	bwd_avg_bytes_bulk
66.	Bwd Pkts/b Avg	bwd_avg_packets_bulk
67.	Bwd Blk Rate Avg	bwd_avg_bulk_rate
68.	Subflow Fwd Pkts	subflow_fwd_packets
69.	Subflow Fwd Byts	subflow_fwd_bytes
70.	Subflow Bwd Pkts	subflow_bwd_packets
71.	Subflow Bwd Byts	subflow_bwd_bytes
72.	Init Fwd Win Byts	init_win_bytes_forward
73.	Init Bwd Win Byts	init_win_bytes_backward
74.	Fwd Act Data Pkts	act_data_pkt_fwd
75.	Fwd Seg Size Min	min_seg_size_forward
76.	Active Mean	active_mean

77.	Active Std	active_std
78.	Active Max	active_max
79.	Active Min	active_min
80.	Idle Mean	idle_mean
81.	Idle Std	idle_std
82.	Idle Max	idle_max
83.	Idle Min	idle_min
84.	Label	label

Fungsi *head()* digunakan untuk menampilkan lima baris awal dari data. Tujuannya yaitu untuk memastikan bahwa data benar-benar telah terbaca oleh sistem atau tidak. Gambar 4.17 dan Gambar 4.18 menunjukkan sebagian data latih dan data uji yang ditampilkan dengan menggunakan fungsi *head()*.

```
df_train.head()
```

	flow_id	source_ip	source_port	destination_ip	destination_port	protocol	timestamp	flow_duration	total_fwd_packets	total_bwd_packets	total
0	192.168.10.5-104.16.207.165-54865-443-6	104.16.207.165	443	192.168.10.5	54865	6	7/7/2017 3:30	3	2	0	
1	192.168.10.5-104.16.28.216-55054-80-6	104.16.28.216	80	192.168.10.5	55054	6	7/7/2017 3:30	109	1	1	
2	192.168.10.5-104.16.28.216-55055-80-6	104.16.28.216	80	192.168.10.5	55055	6	7/7/2017 3:30	52	1	1	
3	192.168.10.16-104.17.241.25-46236-443-6	104.17.241.25	443	192.168.10.16	46236	6	7/7/2017 3:30	34	1	1	
4	192.168.10.5-104.19.196.102-54863-443-6	104.19.196.102	443	192.168.10.5	54863	6	7/7/2017 3:30	3	2	0	

Gambar 4.17 Sebagian Isi Data Latih

```
df_test.head()
```

	flow_id	source_ip	source_port	destination_ip	destination_port	protocol	timestamp	flow_duration	total_fwd_packets	total_bwd_packets	total_len
0	192.168.60.1-192.168.60.3-51165-80-6	192.168.60.3	80	192.168.60.1	51165	6	6/7/2019 21:30	38	0	2	
1	192.168.60.1-224.0.0.252-52305-5355-17	192.168.60.1	52305	224.0.0.252	5355	17	6/7/2019 21:32	409783	1	1	
2	192.168.60.1-192.168.60.3-51192-80-6	192.168.60.3	80	192.168.60.1	51192	6	6/7/2019 21:31	34	0	2	
3	192.168.60.1-192.168.60.3-51168-80-6	192.168.60.3	80	192.168.60.1	51168	6	6/7/2019 21:30	36	0	2	
4	192.168.60.1-192.168.60.3-51184-80-6	192.168.60.3	80	192.168.60.1	51184	6	6/7/2019 21:31	43	0	2	

Gambar 4.18 Sebagian Isi Data Uji

4.2.7 Implementasi Praproses Data

a. Explorasi Data

Sebelum praproses data dilakukan, data perlu diketahui lebih rinci terlebih dahulu. Dengan menggunakan fungsi *describe()* dari Pandas, dapat diketahui jumlah data, rata-rata, deviasi standar, data terendah, data 25%, data 50%, data 75%, dan data tertinggi dari setiap fitur seperti yang dapat dilihat pada Gambar 4.19 dan Gambar 4.20.

```
df_train.describe()
```

	source_port	destination_port	protocol	flow_duration	total_fwd_packets	total_bwd_packets	total_length_of_fwd_packets	total_length_of_bwd_p
count	225745.000000	225745.00000	225745.000000	2.257450e+05	225745.000000	225745.000000	225745.000000	2.25745
mean	38257.568402	8879.61946	7.600288	1.624165e+07	4.874916	4.572775	939.463346	5.96047
std	23057.302075	19754.64740	3.881586	3.152437e+07	15.422874	21.755356	3249.403484	3.92183
min	0.000000	0.00000	0.000000	-1.000000e+00	1.000000	0.000000	0.000000	0.00000
25%	18990.000000	80.00000	6.000000	7.118000e+04	2.000000	1.000000	26.000000	0.00000
50%	49799.000000	80.00000	6.000000	1.452333e+06	3.000000	4.000000	30.000000	1.64000
75%	58296.000000	80.00000	6.000000	8.805237e+06	5.000000	5.000000	63.000000	1.16010
max	65534.000000	65532.00000	17.000000	1.199999e+08	1932.000000	2942.000000	183012.000000	5.17234

Gambar 4.19 Deskripsi Data Latih

```
df_test.describe()
```

	source_port	destination_port	protocol	flow_duration	total_fwd_packets	total_bwd_packets	total_length_of_fwd_packets	total_length_of_bwd_packe
count	5926.000000	5926.000000	5926.000000	5.926000e+03	5926.000000	5926.000000	5926.000000	5926.0000
mean	48600.502700	9208.431151	6.079987	8.216515e+06	44.502194	1.935707	527.431488	247.9326
std	21627.892509	21146.795653	1.014169	1.181130e+07	815.368133	6.947476	9785.775434	7282.3398
min	0.000000	0.000000	0.000000	1.300000e+01	0.000000	1.000000	0.000000	0.0000
25%	51605.750000	80.000000	6.000000	4.962500e+02	0.000000	1.000000	0.000000	0.0000
50%	55826.500000	80.000000	6.000000	8.999517e+06	2.000000	1.000000	0.000000	0.0000
75%	61297.000000	80.000000	6.000000	9.000654e+06	2.000000	2.000000	0.000000	0.0000
max	65523.000000	65243.000000	17.000000	1.160009e+08	25726.000000	344.000000	308712.000000	474039.0000

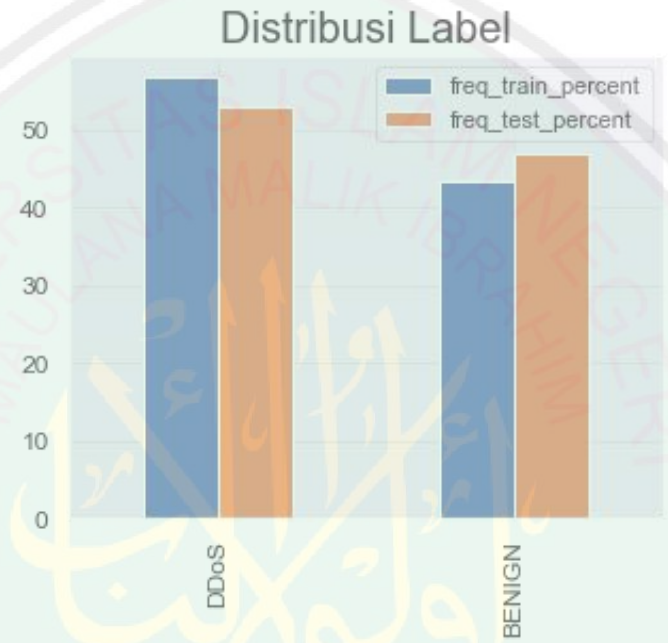
Gambar 4.20 Deskripsi Data Uji

Fungsi *describe()* hanya menunjukkan detail atribut numerik. Oleh sebab itu fitur label yang merupakan fitur dengan atribut kategorik tidak dapat dirincikan. Untuk distribusi kelas BENIGN dan kelas DDoS pada fitur label data latih dan data uji dapat dilihat pada Tabel 4.4.

Tabel 4.4 Distribusi Label Data Latih dan Data Uji

Label	**Data Latih**		**Data Uji**	
DDoS	128027	56.71%	3147	53.1%
BENIGN	97718	43.39%	2779	46.9%

Grafik distribusi kelas BENIGN dan kelas DDoS pada fitur label data latih dan data uji dapat dilihat pada Gambar 4.21.



Gambar 4.21 Grafik Dristribusi Label Data Latih dan Data Uji

b. Membersihkan Data

Dari fungsi *describe()* juga dapat diketahui fitur apa saja yang tidak menyimpan atau tidak memiliki data. Karena fitur-fitur tersebut tidak memengaruhi pada saat pelatihan dan pengujian data, maka digunakan fungsi *drop()* untuk menghapus fitur-fitur tersebut. Adapun fitur-fitur yang tidak perlu dan tidak digunakan dalam proses analisis, dapat dilihat pada Tabel 4.5.

Tabel 4.5 Fitur-Fitur yang Tidak Perlu

No.	Nama Fitur
1.	flow_id
2.	source_ip
3.	source_port
4.	destination_ip
5.	timestamp
6.	fwd_urg_flags
7.	bwd_urg_flags
8.	cwe_flag_count
9.	fwd_avg_bytes_bulk
10.	fwd_avg_packets_bulk
11.	fwd_avg_bulk_rate
12.	bwd_avg_bytes_bulk
13.	bwd_avg_packets_bulk
14.	bwd_avg_bulk_rate

Selanjutnya yaitu membersihkan data yang rusak. Dalam hal ini yaitu data yang bernilai NaN dan *Infinity*. Apabila data memiliki nilai NaN dan *Infinity*, maka praproses tidak dapat dilanjutkan. Untuk mengecek keberadaan kedua nilai tersebut, fungsi yang digunakan yaitu *isna().any()* dari Pandas. Fungsi ini mengecek keseluruhan fitur dan menunjukkan keluaran *True* atau *False*. Keluaran *True* menunjukkan bahwa fitur memiliki nilai NaN atau *Infinity*, sedangkan nilai *False* menunjukkan sebaliknya. Tabel 4.6 dan Tabel 4.7 menunjukkan hasil pengecekan nilai NaN dan *Infinity* yang ada pada data latih dan data uji.

Tabel 4.6 Hasil Pengecekan Data yang Rusak Pada Data Latih

Nama Fitur	Hasil	Nama Fitur	Hasil
destination_port	False	fwd_packets	False
protocol	False	bwd_packets	False
flow_duration	False	min_packet_length	False
total_fwd_packets	False	max_packet_length	False
total_bwd_packets	False	packet_length_mean	False
total_length_of_fwd_packets	False	packet_length_std	False

total_length_of_bwd_packets	False	packet_length_variance	False
fwd_packet_length_max	False	fin_flag_count	False
fwd_packet_length_min	False	syn_flag_count	False
fwd_packet_length_mean	False	rst_flag_count	False
fwd_packet_length_std	False	psh_flag_count	False
bwd_packet_length_max	False	ack_flag_count	False
bwd_packet_length_min	False	urg_flag_count	False
bwd_packet_length_mean	False	ece_flag_count	False
bwd_packet_length_std	False	down_up_ratio	False
flow_bytes	True	average_packet_size	False
flow_packets	True	avg_fwd_segment_size	False
flow_iat_mean	False	avg_bwd_segment_size	False
flow_iat_std	False	subflow_fwd_packets	False
flow_iat_max	False	subflow_fwd_bytes	False
flow_iat_min	False	subflow_bwd_packets	False
fwd_iat_total	False	subflow_bwd_bytes	False
fwd_iat_mean	False	init_win_bytes_forward	False
fwd_iat_std	False	init_win_bytes_backward	False
fwd_iat_max	False	act_data_pkt_fwd	False
fwd_iat_min	False	min_seg_size_forward	False
bwd_iat_total	False	active_mean	False
bwd_iat_mean	False	active_std	False
bwd_iat_std	False	active_max	False
bwd_iat_max	False	active_min	False
bwd_iat_min	False	idle_mean	False
fwd_psh_flags	False	idle_std	False
bwd_psh_flags	False	idle_max	False
fwd_header_length	False	idle_min	False
bwd_header_length	False	label	False

Tabel 4.7 Hasil Pengecekan Data yang Rusak Pada Data Uji

Nama Fitur	**Hasil**	**Nama Fitur**	**Hasil**
destination_port	False	fwd_packets	False
protocol	False	bwd_packets	False
flow_duration	False	min_packet_length	False
total_fwd_packets	False	max_packet_length	False
total_bwd_packets	False	packet_length_mean	False
total_length_of_fwd_packets	False	packet_length_std	False
total_length_of_bwd_packets	False	packet_length_variance	False
fwd_packet_length_max	False	fin_flag_count	False
fwd_packet_length_min	False	syn_flag_count	False
fwd_packet_length_mean	False	rst_flag_count	False
fwd_packet_length_std	False	psh_flag_count	False
bwd_packet_length_max	False	ack_flag_count	False
bwd_packet_length_min	False	urg_flag_count	False
bwd_packet_length_mean	False	ece_flag_count	False

bwd_packet_length_std	False	down_up_ratio	False
flow_bytes	False	average_packet_size	False
flow_packets	False	avg_fwd_segment_size	False
flow_iat_mean	False	avg_bwd_segment_size	False
flow_iat_std	False	subflow_fwd_packets	False
flow_iat_max	False	subflow_fwd_bytes	False
flow_iat_min	False	subflow_bwd_packets	False
fwd_iat_total	False	subflow_bwd_bytes	False
fwd_iat_mean	False	init_win_bytes_forward	False
fwd_iat_std	False	init_win_bytes_backward	False
fwd_iat_max	False	act_data_pkt_fwd	False
fwd_iat_min	False	min_seg_size_forward	False
bwd_iat_total	False	active_mean	False
bwd_iat_mean	False	active_std	False
bwd_iat_std	False	active_max	False
bwd_iat_max	False	active_min	False
bwd_iat_min	False	idle_mean	False
fwd_psh_flags	False	idle_std	False
bwd_psh_flags	False	idle_max	False
fwd_header_length	False	idle_min	False
bwd_header_length	False	label	False

Dari hasil pengecekan, didapati bahwa fitur *flow_bytes* dan *flow_packets* pada data latih memiliki nilai NaN dan *Infinity*, sedangkan pada data uji, tidak didapati nilai NaN dan Infinity. Karena fitur *flow_bytes* dan *flow_packets* pada data latih memiliki nilai NaN dan *Infinity*, sehingga menyebabkan tipe data dari kedua fitur tersebut menjadi ambigu. Oleh sebab itu tipe datanya perlu diganti menjadi *float64* terlebih dahulu, kemudian mengganti nilai NaN dan *Infinity* dengan nilai 0 (nilai konstan).

Karena telah banyak perubahan pada data, seperti menghapus fitur yang tidak perlu dan praproses lainnya, sehingga dimensi pada data menjadi berubah, dimana sekarang hanya memiliki 69 fitur dari sebelumnya yang berjumlah 84 fitur seperti yang dapat dilihat pada Gambar 4.22.

```
print('Dimensi Data Train:', df_train.shape)
print('Dimensi Data Test:', df_test.shape)
```

```
Dimensi Data Train: (225745, 70)
Dimensi Data Test: (5926, 70)
```

Gambar 4.22 Perubahan Dimensi Setelah Pembersihan Data

c. Normalisasi Atribut Numerik

Praproses selanjutnya adalah normalisasi atribut numerik dengan menggunakan fungsi *StandardScaler()* dari Scikit-Learn. Standarisasi dilakukan untuk mengubah data menjadi data dengan skala yang wajar dan serupa agar data yang digunakan tidak memiliki penyimpangan yang besar.

Sebelum data dinormalisasikan agar data memiliki rata-rata nol dan varian unit, atribut numerik diekstrak dan diseleksi terlebih dahulu sehingga fitur label tidak ikut distandarisasikan karena merupakan fitur dengan atribut kategorik. Dari proses standarisasi, atribut numerik kemudian berubah seperti yang dapat dilihat pada Gambar 4.23 dan Gambar 4.24.

```
sc_dftrain.head()
```

	destination_port	protocol	flow_duration	total_fwd_packets	total_bwd_packets	total_length_of_fwd_packets	total_length_of_bwd_packets	fwd_packet_lengtl
0	2.327831	-0.412278	-0.515210	-0.186406	-0.210191	-0.285426	-0.151982	-0.2
1	2.337398	-0.412278	-0.515207	-0.251245	-0.164225	-0.287273	-0.151829	-0.2
2	2.337449	-0.412278	-0.515209	-0.251245	-0.164225	-0.287273	-0.151829	-0.2
3	1.891022	-0.412278	-0.515209	-0.251245	-0.164225	-0.287273	-0.151829	-0.2
4	2.327730	-0.412278	-0.515210	-0.186406	-0.210191	-0.285426	-0.151982	-0.2

Gambar 4.23 Hasil Normalisasi Atribut Numerik Pada Data Latih

```
sc_dftest.head()
```

	destination_port	protocol	flow_duration	total_fwd_packets	total_bwd_packets	total_length_of_fwd_packets	total_length_of_bwd_packets	fwd_packet_lengtl
0	1.984230	-0.078876	-0.695704	-0.054584	0.009255	-0.053902	-0.034049	-0.(
1	-0.182238	10.768363	-0.661010	-0.053357	-0.134694	-0.051654	-0.031027	0.3
2	1.985507	-0.078876	-0.695704	-0.054584	0.009255	-0.053902	-0.034049	-0.(
3	1.984372	-0.078876	-0.695704	-0.054584	0.009255	-0.053902	-0.034049	-0.(
4	1.985129	-0.078876	-0.695704	-0.054584	0.009255	-0.053902	-0.034049	-0.(

Gambar 4.24 Hasil Normalisasi Atribut Numerik Pada Data Uji

d. Normalisasi Atribut Kategorik

Selain atribut numerik, atribut kategorik juga perlu dinormalisasikan. Untuk menormalisasi atribut kategorik, fungsi yang digunakan yaitu

LabelEncoder() dari Scikit-Learn. Atribut kategorik yang dinormalisasikan dalam hal ini yaitu yang ada pada fitur label. Atribut teks yang merupakan atribut kategorik diekstrak dan diseleksi terlebih dahulu. Kemudian data yang ada di fitur label berubah menjadi angka 0 dan 1. Angka 0 merepresentasikan kelas BENIGN, sedangkan angka 1 merepresentasikan kelas DDoS. Hasil normalisasi atribut kategorik menggunakan *LabelEncoder()* dapat dilihat pada Gambar 4.25 dan Gambar 4.26.

```
en_ytrain.values.reshape(1,-1)
```

```
array([[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

Gambar 4.25 Hasil Normalisasi Atribut Kategorik Pada Data Latih

```
en_ytest.values.reshape(1,-1)
```

```
array([[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
```

Gambar 4.26 Hasil Normalisasi Atribut Kategorik Pada Data Uji

4.2.8 Implementasi Seleksi Fitur

a. Resampling

Resampling adalah proses *sampling* ulang terhadap sebuah data. *Resampling* dilakukan untuk mengatasi ketidakseimbangan kelas yang ada pada data (*class imbalance*). *Class imbalance* atau ketidakseimbangan kelas merupakan salah satu permasalahan pada *data mining* dengan kondisi dimana kelas minoritas jauh lebih kecil atau lebih jarang dari kelas mayoritas ataupun sebaliknya sehingga menghasilkan akurasi yang kurang baik. Dalam hal ini, *resampling* dilakukan untuk mengoptimalkan hasil dari seleksi fitur.

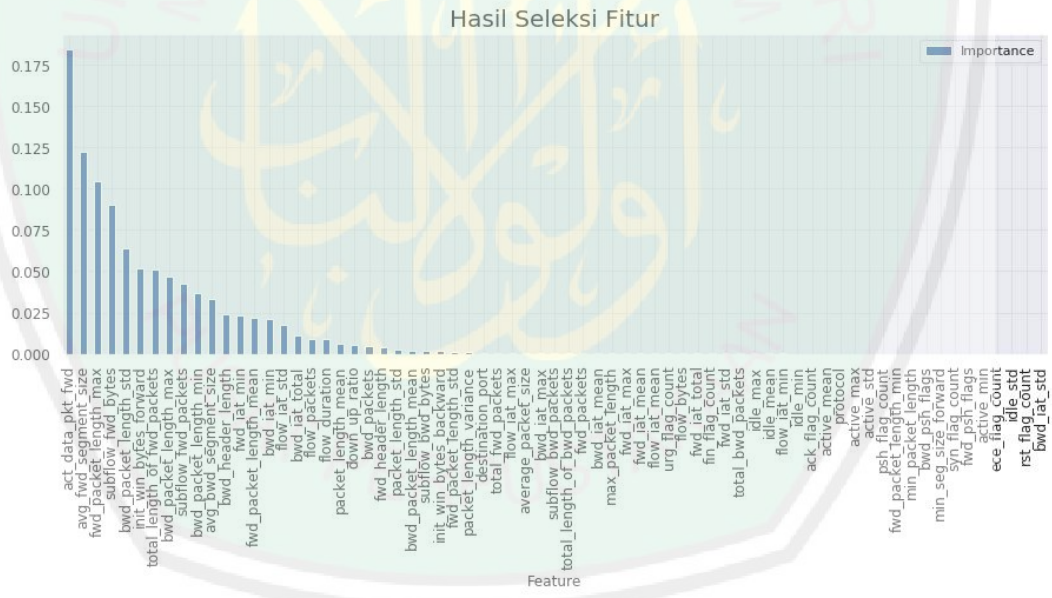
Pendekatan *resampling* dibagi menjadi tiga kategori, yaitu metode *over-sampling*, *under sampling*, dan hibrida yang menggabungkan kedua pendekatan *sampling* (Jian dkk, 2016). Adapun pendekatan yang digunakan yaitu *over-sampling* yang bertujuan untuk meningkatkan sampel kelas minoritas (BENIGN) sampai sama dengan kelas mayoritas (DDoS) dengan cara menduplikasi sampel kelas BENIGN secara acak.

Pada penelitian ini, digunakan fungsi *RandomOverSampler()* dari imblearn untuk melakukan *resampling*. Tipe data kolom ditentukan dan diekstrak terlebih dahulu sebelum pengambilan sampel secara acak dilakukan. Langkah selanjutnya yaitu membentuk ulang kolom target atau fitur label dalam bentuk *array* satu dimensi kemudian menjalankan fungsi *RandomOverSampler()*. Setelah pengambilan sampel dilakukan, kelas yang ada pada fitur label data latih menjadi sama dan seimbang, yang sebelumnya berjumlah 97718 baris data untuk kelas BENIGN dan 128027 baris data untuk kelas DDoS, menjadi 128027 baris data untuk kedua kelas.

b. Seleksi Fitur dengan *Random Forest*

Seleksi fitur bertujuan untuk memilih fitur-fitur mana saja yang dianggap penting sebelum dilakukan pelatihan dan pengujian. Setelah memberi penilaian setiap fitur dengan melakukan pelatihan pada setiap pohon, maka dipilih beberapa fitur yang dianggap penting. Fitur yang dipilih berdasarkan kaidah standar pemilihan fitur yaitu berdasarkan ambang batas rata-rata nilai dari setiap fitur.

Dengan menggunakan fungsi *RandomForestClassifier()* dari Scikit-Learn, fitur-fitur yang ada pada data latih kemudian dihitung nilainya lalu divisualisasikan dalam bentuk grafik untuk memudahkan mengetahui hasil dari seleksi fitur yang dilakukan. Gambar 4.27 menunjukkan hasil penilaian fitur dari *Random Forest.*



Gambar 4.27 Hasil Seleksi Fitur dengan *Random Forest*

Total skor dari seluruh fitur yaitu 0.999999 yang kemudian dirata-ratakan dengan jumlah fitur yang ada untuk dijadikan nilai ambang batas pemilihan fitur. Dari nilai ambang batas yang didapatkan, dipilih 20 fitur yang diurutkan berdasarkan skor tertinggi seperti yang dapat dilihat pada Tabel 4.8.

Tabel 4.8 Fitur Terpilih Hasil Seleksi Fitur

No.	Fitur Terpilih	Skor
1.	act_data_pkt_fwd	0.18469
2.	avg_fwd_segment_size	0.12274
3.	fwd_packet_length_max	0.10458
4.	subflow_fwd_bytes	0.09058
5.	bwd_packet_length_std	0.06399
6.	init_win_bytes_forward	0.05179
7.	total_length_of_fwd_packets	0.05094
8.	bwd_packet_length_max	0.04691
9.	subflow_fwd_packets	0.04265
10.	bwd_packet_length_min	0.03666
11.	avg_bwd_segment_size	0.03316
12.	bwd_header_length	0.02401
13.	fwd_iat_min	0.02331
14.	fwd_packet_length_mean	0.02165
15.	bwd_iat_min	0.02105
16.	flow_iat_std	0.01787
17.	bwd_iat_total	0.01131
18.	flow_packets	0.00915
19.	flow_duration	0.00901
20.	packet_length_mean	0.0058

4.2.9 Implementasi Pelatihan dan Pengujian dengan GRNN

Sebelum pelatihan dan pengujian dilakukan, data yang sebelumnya berbentuk *array* ditransformasikan kembali menjadi *dataframe* untuk memudahkan proses. Dimensi data akhir yang diperoleh sebelum partisi dilakukan yaitu untuk data latih sebanyak (256054, 70) dan data uji sebanyak (5926, 70).

Data tersebut kemudian dipartisi menjadi beberapa variabel, yaitu X_train, X_test, X_train_fs, X_test_fs, y_train, dan y_test seperti yang dapat dilihat pada Gambar 4.28. X_train yaitu data latih dengan fitur lengkap, X_test yaitu data uji dengan fitu lengkap, X_train_fs yaitu data latih dengan fitur terpilih, X_test_fs

yaitu data uji dengan fitur terpilih, y_train yaitu data target pelatihan, dan y_test yaitu data target pengujian.

```
X_train = traindf[traincols].values
X_test = testdf[traincols].values

X_train_fs = traindf[selected_features].values
X_test_fs = testdf[selected_features].values

ytrain = traindf[['label']].copy()
#y_train = ytrain.values
r, c = ytrain.values.shape
y_train = ytrain.values.reshape(r,)
y_train = y_train.astype(np.int64)

ytest = testdf[['label']].copy()
#y_test = ytest.values
r, c = ytest.values.shape
y_test = ytest.values.reshape(r,)
y_test = y_test.astype(np.int64)
```

Gambar 4.28 Partisi Data Latih dan Data Uji

Untuk dimensi data untuk masing-masing variabel yaitu X_train sebanyak (256054, 69), X_test sebanyak (5926, 69), X_train_fs sebanyak (256054, 20), X_test_fs sebanyak (5926, 20), y_train sebanyak (256054,), dan y_test sebanyak (5926,).

Pelatihan dilakukan dengan mengimplementasikan teknik *Cross Validation* untuk mendapatkan parameter sigma terbaik. Jumlah iterasi yang diterapkan yaitu sebanyak 15 kali dengan penambahan sigma sebesar 0.1 di setiap iterasi. Setelah mendapatkan sigma terbaik, maka selanjutnya dilakukan pengujian. Percobaan dilakukan dua kali, yaitu pelatihan dan pengujian dengan fitur lengkap serta pelatihan dan pengujian dengan fitur terpilih.

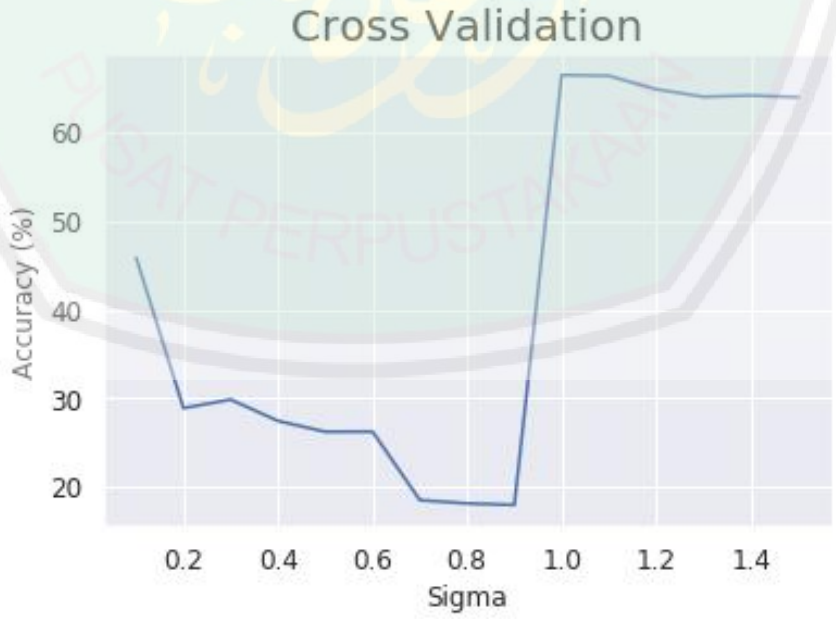
a. Pelatihan dan Pengujian dengan Fitur Lengkap

Data yang digunakan untuk pelatihan dan pengujian ini yaitu X_train, X_test, y_train, dan y_test. Waktu yang dibutuhkan untuk pelatihan yaitu selama 1 jam 45 menit 6 detik. Adapun hasil dari pelatihan dengan fitur lengkap dapat dilihat pada Tabel 4.9.

Tabel 4.9 Hasil Pelatihan dengan Fitur Lengkap

Iterasi	Sigma	Akurasi
1	0.1	45.8%
2	0.2	28.9%
3	0.3	29.8%
4	0.4	27.4%
5	0.5	26.2%
6	0.6	26.2%
7	0.7	18.5%
8	0.8	18.1%
9	0.9	18.0%
10	1.0	66.4%
11	1.1	66.3%
12	1.2	64.8%
13	1.3	64.0%
14	1.4	64.1%
15	1.5	63.9%

Sedangakan hasil dari pelatihan dengan fitur lengkap dalam bentuk grafik dapat dilihan pada Gambar 4.29.



Gambar 4.29 Grafik Hasil Pelatihan dengan Fitur Lengkap

Dari hasil pelatihan yang telah dilakukan, diperoleh parameter nilai sigma atau deviasi standar terbaik yaitu sebesar 1 yang menunjukkan data uji terdistribusi dengan cukup baik dimana titik data uji cukup mirip atau mendekati terhadap data latih pada sigma 1 yang masih terbilang normal. Dengan demikian, sigma 1 digunakan sebagai masukan untuk pengujian. Gambar 4.30 menunjukkan hasil pengujian dengan fitur terpilih.

```
predictions = grnn.predict(X_test, sigma=1)
print(predictions)
```

```
[0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 1. 0. 1. 0. 1. 1. 0. 0. 0. 1. 1. 0. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 0. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 0.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0.
 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 0. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 0. 1. 1. 1. 1. 0. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1.
 1. 0. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1.
```

Gambar 4.30 Hasil Pengujian dengan Fitur Lengkap

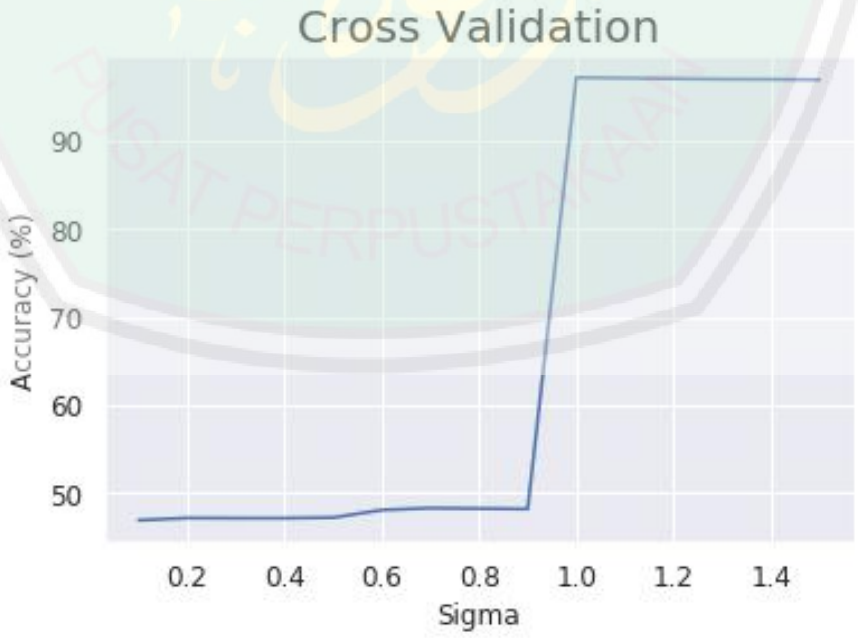
b. Pelatihan dan Pengujian dengan Fitur Terpilih

Data yang digunakan untuk pelatihan dan pengujian ini yaitu X_train_fs, X_test_fs, y_train, dan y_test. Waktu yang dibutuhkan untuk pelatihan yaitu selama 42 menit 27 detik. Waktu yang dibutuhkan menjadi lebih singkat karena fitur yang digunakan lebih sedikit sehingga dimensi data pun berkurang. Oleh karena itu waktu komputasi yang dibutuhkan untuk pelatihan dan pengujian menjadi lebih singkat. Adapun hasil dari pelatihan dengan fitur terpilih dapat dilihat pada Tabel 4.10.

Tabel 4.10 Hasil Pelatihan dengan Fitur Terpilih

Iterasi	Sigma	Akurasi
1	0.1	46.9%
2	0.2	47.2%
3	0.3	47.1%
4	0.4	47.1%
5	0.5	47.2%
6	0.6	48.1%
7	0.7	48.3%
8	0.8	48.2%
9	0.9	48.2%
10	1.0	97.2%
11	1.1	97.1%
12	1.2	97.1%
13	1.3	97.0%
14	1.4	97.0%
15	1.5	97.0%

Sedangakan hasil dari pelatihan dengan fitur terpilih dalam bentuk grafik dapat dilihan pada Gambar 4.31.



Gambar 4.31 Grafik Hasil Pelatihan dengan Fitur Terpilih

Dari hasil pelatihan yang telah dilakukan, diperoleh parameter nilai sigma atau deviasi standar terbaik yaitu sebesar 1 yang menunjukkan data uji terdistribusi dengan cukup baik dimana titik data uji cukup mirip atau mendekati terhadap data latih pada sigma 1 yang masih terbilang normal. Dengan demikian, sigma 1 digunakan sebagai masukan untuk pengujian. Gambar 4.32 menunjukkan hasil pengujian dengan fitur terpilih.

```
predictions2 = grnn2.predict(X_test_fs, sigma=1)
print(predictions2)
```

```
[0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.
 1. 1. 1. 1. 1. 0. 0. 0. 1. 1. 0. 0. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
```

Gambar 4.32 Hasil Pengujian dengan Fitur Terpilih

4.2.10 Implementasi Evaluasi Model

Evaluasi model dilakukan dengan menggunakan *Confusion Matrix* untuk mengukur *accuracy, precision, recall,* dan *f1-score* dari model yang digunakan.

a. Evaluasi Model dengan Fitur Lengkap

Hasil evaluasi model dengan fitur lengkap dapat dilihat pada Gambar 4.33.

```
Model Accuracy:
 0.6641916976037799

Confusion matrix:
 [[ 947 1832]
 [ 158 2989]]

Classification report:
              precision    recall  f1-score   support

           0       0.86      0.34      0.49      2779
           1       0.62      0.95      0.75      3147

    accuracy                           0.66      5926
   macro avg       0.74      0.65      0.62      5926
weighted avg       0.73      0.66      0.63      5926
```

Gambar 4.33 Hasil Evaluasi Model dengan Fitur Lengkap

Dari hasil evaluasi yang dilakukan, diperoleh *accuracy* sebesar 66.41%; *precision* sebesar 73,85%; *recall* sebesar 64,52%; dan *f1-score* sebesar 61,89%.

b. Evaluasi Model dengan Fitur Terpilih

Hasil evaluasi dengan fitur terpilih dapat dilihat pada Gambar 4.34.

```
Model Accuracy:
 0.9721565980425245

Confusion matrix:
 [[2690   89]
 [  76 3071]]

Classification report:
              precision    recall  f1-score   support

           0       0.97      0.97      0.97      2779
           1       0.97      0.98      0.97      3147

    accuracy                           0.97      5926
   macro avg       0.97      0.97      0.97      5926
weighted avg       0.97      0.97      0.97      5926
```

Gambar 4.34 Hasil Evaluasi Model dengan Fitur Terpilih

Dari hasil evaluasi yang dilakukan, diperoleh *accuracy* sebesar 97,21%; *precision* sebesar 97,21%; *recall* sebesar 97,19%; dan *f1-score* sebesar 97,2%.

4.3 Integrasi dengan Islam

Selain melanggar hukum negara, kejahatan di dunia maya juga jelas melanggar hukum dan nilai-nilai Islam. Tindakan ini merupakan perbuatan zalim yang tidak disukai oleh Allah. Allah berfirman dalam Al-Qur'an:

وَاللهُ لاَ يُحِبُّ الظَّالِمِينَ

"Dan Allah tidak menyukai orang-orang yang zalim." (Q.S. Ali 'Imran 57)

Dalam tafsir Jalalayn, disebutkan bahwa Allah tidak menyukai orang-orang yang aniaya, artinya Allah melaknat dan akan menyiksa mereka. Sangat jelas bahwa Allah sangat membenci orang-orang zalim seperti halnya melakukan serangan DDoS, karena perbuatan ini akan sangat merugikan korban.

Dalam sebuah hadits hasan yang diriwayatkan oleh Ibnu Majah, Daruquthni dan lain-lainnya, disebutkan bahwa:

عنْ أَبِي سَعِيدٍ سَعدِ بنِ مَالِك بنِ سِنَانٍ الخُدرِيِّ رضي الله عنه أَنَّ رَسُولَ الله صلى الله عليه وسلم قَالَ :لاَ ضَرَرَ وَلاَ ضِرَارَ

"Dari Abu Sa'id, Sa'ad bin Malik bin Sinan Al Khudri radhiyallahu anhu, sesungguhnya Rasulullah Shallallahu 'alaihi wa Sallam telah bersabda: Janganlah engkau membahayakan dan saling merugikan." [Ibnu Majah no. 2341, Daruquthni no. 4/228, Imam Malik (Muwaththo 2/746)]

Para ahli fiqih dalam hal ini mempunyai berbagai pendapat, namun pendapat yang benar ialah seseorang tidak boleh membahayakan saudaranya baik hal itu merugikan atau tidak, namun dia berhak untuk diberi pembelaan dan pelakunya diberi hukuman sesuai dengan ketentuan hukum. Hal ini tidak

dikatakan zalim atau membahayakan selama sesuai dengan ketentuan yang dibenarkan oleh Sunnah.

Dari kedua dalil tersebut, maka penelitian ini diharapkan mampu mengurangi kejahatan di dunia maya dengan menghasilkan persentase akurasi deteksi yang lebih baik dan mampu meningkatkan performa sistem deteksi intrusi jaringan dengan mengimplementasikan *General Regression Neural Network.*

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan dari hasil penelitian yang telah dilakukan, dapat disimpulkan bahwa *General Regression Neural Network* mampu mendeteksi serangan DDoS dengan baik. Analisis dilakukan dengan menggunakan data latih dari set data terbaru untuk deteksi intrusi yaitu CICIDS2017 yang merupakan pengembangan dari set data yang telah ada sebelumnya. Sedangkat untuk data uji diperoleh dari hasil simulasi serangan DDoS ke server web yang diekstrak menggunakan CICFlowMeter 4.0 sehingga menghasilkan data dengan format yang sama seperti data latih.

Dua percobaan berbeda dilakukan pada penelitian ini, pada percobaan pertama, pelatihan dan pengujian dilakukan dengan menggunakan semua fitur yang ada pada data latih dan data uji yaitu sebanyak 69 fitur. Percobaan ini membutuhkan waktu selama 1 jam 45 menit 6 detik untuk pelatihan. Dari pelatihan tersebut diperoleh sigma terbaik yaitu 1 yang kemudian dijadikan masukan untuk pengujian dan evaluasi. Dari hasil evaluasi ini, diperoleh *accuracy* sebesar 66.41%; *precision* sebesar 73,85%; *recall* sebesar 64,52%; dan *f1-score* sebesar 61,89%.

Sementara itu pada percobaan kedua, pelatihan dan pengujian dilakukan dengan memanfaatkan sebagian fitur yang telah diseleksi menggunakan algoritma *Random Forest*. Dari hasil seleksi fitur, diperoleh sebanyak 20 fitur terpilih. Percobaan ini membutuhkan waktu selama 42 menit 27 detik untuk pelatihan. Dari pelatihan tersebut diperoleh sigma terbaik yaitu 1 yang kemudian dijadikan

masukan untuk pengujian dan evaluasi. Dari hasil evaluasi ini, diperoleh *accuracy* sebesar 97,21%; *precision* sebesar 97,21%; *recall* sebesar 97,19%; dan *f1-score* sebesar 97,2%.

Dari kedua percobaan yang telah dilakukan, percobaan kedua menghasilkan *accuracy*, *precision*, *recall*, dan *f1-score* yang jauh lebih baik dari percobaan pertama. Dapat disimpulkan bahwa optimalisasi parameter sigma merupakan hal yang sangat penting pada saat evaluasi model. Hal ini disebabkan karena sigma secara langsung mampu mempengaruhi sebaran data pada data, apabila sigma terlalu besar maka hasil yang diperoleh pun akan tidak optimal. Seleksi fitur juga memiliki peran penting dalam memperoleh *accuracy*, *precision*, *recall*, dan *f1-score* yang optimal. Hal ini disebabkan karena fitur-fitur yang terpilih adalah fitur yang memiliki pengaruh penting terhadap *y* atau variabel dependen. Selain itu seleksi fitur juga dapat memangkas waktu komputasi menjadi lebih singkat.

5.2 Saran

Peneliti menyadari bahwa penelitian ini masih belum sempurna. Dengan demikian perlu adanya pengembangan untuk mendapatkan hasil yang lebih baik. Adapun beberapa saran dari peneliti antara lain:

a) Melakukan pembuktian dengan pendekatan yang berbeda.

b) Mengoptimalkan waktu komputasi dengan memanfaatkan algoritma lain.

c) Melakukan percobaan dengan set data yang berbeda.

d) Memvisualisasikan hasil ke dalam GUI seperti website atau lainnya.

e) Mengimplementasikan ke dalam bentuk sistem deteksi intrusi sederhana untuk mendeteksi serangan DDoS secara *realtime.*

DAFTAR PUSTAKA

Breiman, L. 2001. *Random Forests*. Machine Learning, 45: 5-32.

Canadian Institute for Cybersecurity, [online], (http://www.netflowmeter.ca/, diakses Maret 2019).

Chen, L.M., Chen, M.C., Liao, W., Sun Y.S. 2013. *A scalable network forensics mechanism for stealthy self-propagating*. Computer Communications 36, 1471-1484.

DTERG Predictive Modelling Software, [online], (https://www.dtreg.com/solution/view/22, diakses Desember 2019).

Goodin, D. 2018. *US service provider survives the biggest recorded DDoS in history,* [online], (https://arstechnica.com/information-technology/2018/03/us-service-provider-survives-the-biggest-recorded-ddos-in-history/, diakses Mei 2018)

Hasan, M.A.M., Nasser, M., Ahmad, S., Molla, K.I. 2016. *Feature Selection for Intrusion Detection Using Random Forest.* Journal of Information Security, 2016, 7, 129-140.

Hilton, S. 2016. *Dyn analysis summary of Friday October 21 attack*, [online], (http://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/, diakses Februari 2018).

Ibrahim, M., Abdullah, M.T., Dehghantanha, A. 2012. *VoIP evicence model: A new forensic method for investigating VoIP malicious attacks*. Cyber Security, Cyber Warfare and Digital Forensic (Cyber Sec), International Conference on: IEEE: 2012. 201-6.

Kato, K. & Klyuev, V. 2014. *Large-scale network packet analysis for intelligent DDoS attack detection development*. 9th Inernational Conference for Internet Technology and Secure Transansactions. ICITST 2014, 360-365.

Khan, S., Gani, A., Wahab A.W.A., Shiraz, M., Khan, I.A. 2016. *Network forensics: Review, taxonomy, and open challenges*. Journal of Network and Computer Applications, S1084-8045(16)30012-1

Layton, R., Watters, P., Dazeley, R. 2010. *Automatically determining phishing campaigns using the USCAP methodology*. eCrime Researchers Summit (eCrime), 2010: IEEE. 1-8.

Li, S., Schmitz, R. 2009. *A novel anti-phishing framework based on honeypots*. IEEE.

Liao, N., Tian, S., Wang, T. 2009. *Network forensics based on fuzzy logic and expert system*. Computer Communications 32, 1881-1892.

Lin, L., Wang, F., Xie, X., Zhong, S. 2017. *Random forests-based extreme learning machine ensemble for multiregime time series prediction*. Expert Systems with Applications, 83, 164-176.

Mehic, M., Slachta, J., Voznak, M. 2016. *Whispering through DDoS attack*. Perspectives in Science Vol 7, 95-100.

Mocas, S. 2004. *Building theoretical underpinnings for digital forensics research.* Digital Investigation, Vol. 1,61-68.

Panigrahi, R. & Borah, S. 2018. *A Detailed Analysis of CICIDS2017 Dataset for Designing Intrusion Detection Systems.* International Journal of Engineering & Technology, 7 (3.24) : 479-482.

Palmer, G. 2001. *A road map for digital forensic research*. Digital Forensic Research Workshop (DFRWS), Final Report.

Radford, B.J., Richardson, B.D., Davis, S.E. 2018. *Sequence Aggregation Rules for Anomaly Detection in Computer Network Traffic.* arXiv:1805.03735, 2018.

Rizzo, R., Fiannaca, A., La Rosa, M., Urso, A. 2015. *The General Regression Neural Network to Classify Barcode and mini-barcode DNA.* CIBB 2014, LNCS 8623, pp. 142–155.

Resende, P.A.A. & Drummond, A.C. 2018. *Adaptive anomaly-based intrusion detection system using genetic algorithm and profiling*. Security Privacy. 2018;1:e36.

Sergey, B. May 2016. *Intrusion Detection System and Intrusion Prevention System with Snort provided by Security Onion.* Bachelor's Thesis of Information Technology.

Specht, D. F. 1991. A *General Regression Neural Network*. IEEE Transactions On Neural Networks. Vol. 2 . No. 6. November 1991.

Smith. S. 20 November 2014, *5 Famous Botnets that Held the Internet Hostage*, [online], (https://tqaweekly.com/episodes/season5/tqa-se5ep11.php, diakses 11 Maret 2018).

Thapliyal, M., Bijalwan, A., Garg, N., Pilli, E.S. 2013. *A Generic Process Model for Botnet Forensic Analysis*.

Whitcomb, C.M. 2002. *An historical perspective of digital evidence: A forensic scientist's view*. IJDE.

Wu, J., Peng, D., Li, Z., Zhao, L., Ling, H. 2015. *Network Intrusion Detection Based on a General Regression Neural Network Optimized by an Improved Artificial Immune Algorithm*. PLoS ONE 10 (3): e0120976.

Xiao, J., Xie, L., He, C., Jiang, X. 2012. *Dynamic classifier ensemble model for customer classification with imbalanced class distribution*. Expert Systems with Applications, 39(3), 3668–3675.

Yulianto, A., Sukarno, P., Suwastika, N.A. 2018. *Improving AdaBoost-based Intrusion Detection System (IDS) Performance on CIC IDS 2017 Dataset*. IOP Conf. Series: Journal of Physics: Conf. Series 1192 (2019) 012018.

Zhu, Y. 2011. *Attack Pattern Discovery in Forensic Investigation of Network Attacks*. Selected Areas in Communications, IEEE Journal 29. 1349-57.

LAMPIRAN – LAMPIRAN

Data Latih

Nama Fitur	Jumlah
Flow ID	86421 (38%)
Source IP	2067 (0%)
Source Port	50697 (22%)
Destination IP	2554 (1%)
Destination Port	23950 (10%)
Protocol	[6:85.41%,17:14.56%,0:0.02%]
Timestamp	(0%)
Flow Duration	187752 (83%)
Total Fwd Packets	297 (0%)
Total Backward Packets	367 (0%)
Total Length of Fwd Packets	3831 (1%)
Total Length of Bwd Packets	6760 (2%)
Fwd Packet Length Max	1891 (0%)
Fwd Packet Length Min	151 (0%)
Fwd Packet Length Mean	7401 (3%)
Fwd Packet Length Std	9555 (4%)
Bwd Packet Length Max	1945 (0%)
Bwd Packet Length Min	343 (0%)
Bwd Packet Length Mean	8655 (3%)
Bwd Packet Length Std	9650 (4%)
Flow Bytes/s	204602 (90%)
Flow Packets/s	196730 (87%)
Flow IAT Mean	193666 (85%)
Flow IAT Std	159622 (70%)
Flow IAT Max	139745 (61%)
Flow IAT Min	11093 (4%)
Fwd IAT Total	78049 (34%)
Fwd IAT Mean	95095 (42%)
Fwd IAT Std	91184 (40%)
Fwd IAT Max	77978 (34%)
Fwd IAT Min	8820 (3%)
Bwd IAT Total	101942 (45%)
Bwd IAT Mean	117311 (51%)
Bwd IAT Std	113859 (50%)
Bwd IAT Max	101379 (44%)
Bwd IAT Min	3787 (1%)
Fwd PSH Flags	[0:96.68%, 1:3.32%]
Bwd PSH Flags	[0:100.0%]
Fwd URG Flags	[0:100.0%]
Bwd URG Flags	[0:100.0%]
Fwd Header Length	714 (0%)
Bwd Header Length	806 (0%)
Fwd Packets/s	192230 (85%)
Bwd Packets/s	144444 (63%)

Min Packet Length	109 (0%)
Max Packet Length	2352 (1%)
Packet Length Mean	11680 (5%)
Packet Length Std	13502 (5%)
Packet Length Variance	13502 (5%)
FIN Flag Count	[0:99.73%,1:0.27%]
SYN Flag Count	[0:96.68%,1:3.32%]
RST Flag Count	[0:99.99%,1:0.01%]
PSH Flag Count	[0:64.88%,1:35.12%]
ACK Flag Count	[1:50.45%,0:49.55%]
URG Flag Count	[0:85.92%,1:14.08%]
CWE Flag Count	[0:100.0%]
ECE Flag Count	[0:99.99%,1:0.01%]
Down/Up Ratio	[0:43.11%,1:39.42%,2:9.6%,3:0.33%,4:0.08%,5:4.42%,6:2.75%,7:0.29%]
Average Packet Size	11270 (4%)
Avg Fwd Segment Size	7401 (3%)
Avg Bwd Segment Size	8655 (3%)
Fwd Header Length	714 (0%)
Fwd Avg Bytes/Bulk	[0:100.0%]
Fwd Avg Packets/Bulk	[0:100.0%]
Fwd Avg Bulk Rate	[0:100.0%]
Bwd Avg Bytes/Bulk	[0:100.0%]
Bwd Avg Packets/Bulk	[0:100.0%]
Bwd Avg Bulk Rate	[0:100.0%]
Subflow Fwd Packets	297 (0%)
Subflow Fwd Bytes	3831 (1%)
Subflow Bwd Packets	367 (0%)
Subflow Bwd Byte	6760 (2%)
Init_Win_bytes_forward	1804 (0%)
Init_Win_bytes_backward	1922 (0%)
act_data_pkt_fwd	234 (0%)
min_seg_size_forward	[0:0.02%,20:88.19%,24:0.0%,28:0.27%,32:10.47%,40:1.02%,44:0.02%,52:0.0%]
Active Mean	40409 (17%)
Active Std	5669 (2%)
Active Max	40313 (17%)
Active Min	38393 (17%)
Idle Mean	35285 (15%)
Idle Std	5857 (2%)
Idle Max	33002 (14%)
Idle Min	48018 (21%)
Label	[DDoS:56.71%,BENIGN:43.29%]

Data Uji

Nama Fitur	Jumlah
Flow ID	3822 (64%)
Source IP	[192.168.60.1:83.88%,192.168.60.3:16.01%,8.6.0.1:0.1%]
Source Port	3557 (60%)
Destination IP	[192.168.60.3:83.5%,192.168.60.1:16.01%,239.255.255.250:0.22%,8.0.6.4:0.1%,224.0.0.252:0.08%,224.0.0.22:0.05%,192.168.60.255:0.03%]
Destination Port	566 (9%)
Protocol	[6:99.04%,17:0.81%,0:0.15%]
Timestamp	[6/7/2019 21:41:15.51%,6/7/2019 21:44:13.15%,6/7/2019 21:48:10.85%,6/7/2019 21:32:10.7%,6/7/2019 21:39:10.43%,6/7/2019 21:45:8.27%,6/7/2019 21:38:6.19%,6/7/2019 21:34:4.34%,6/7/2019 21:42:4.18%,6/7/2019 21:49:3.86%,6/7/2019 21:47:3.39%,6/7/2019 21:40:2.68%,6/7/2019 21:33:2.24%,6/7/2019 21:43:1.38%,6/7/2019 21:46:0.88%,6/7/2019 21:50:0.62%,6/7/2019 21:35:0.49%,6/7/2019 21:31:0.32%,6/7/2019 21:30:0.27%,6/7/2019 21:36:0.22%,6/7/2019 21:37:0.02%]
Flow Duration	3776 (63%)
Total Fwd Packets	[2:50.42%,0:28.08%,1:16.44%,3:3.14%,4:0.71%,5:0.32%,6:0.1%,9:0.07%,14:0.05%,8:0.05%,12:0.03%,11:0.03%,7:0.03%,10:0.03%,14182:0.02%,14463:0.02%,13946:0.02%,14811:0.02%,176:0.02%,45:0.02%,60:0.02%,36:0.02%,24:0.02%,16:0.02%,17:0.02%,13681:0.02%,14442:0.02%,15241:0.02%,14840:0.02%,14804:0.02%,151:0.02%,13748:0.02%,13660:0.02%,25726:0.02%,86:0.02%,14466:0.02%,62:0.02%,58:0.02%,42:0.02%,14207:0.02%,13189:0.02%,14714:0.02%,14838:0.02%]
Total Backward Packets	[1:55.4%,2:31.35%,3:6.48%,4:4.84%,6:0.71%,5:0.67%,14:0.08%,9:0.05%,12:0.05%,23:0.05%,15:0.03%,16:0.02%,80:0.02%,104:0.02%,344:0.02%,8:0.02%,24:0.02%,159:0.02%,13:0.02%,69:0.02%,111:0.02%,10:0.02%,42:0.02%,118:0.02%,286:0.02%,27:0.02%,77:0.02%]
Total Length of Fwd Packets	[0:98.38%,522:0.15%,48:0.07%,33:0.07%,523:0.07%,12:0.07%,685:0.05%,452:0.03%,3936:0.03%,394:0.03%,173592:0.02%,128

	8:0.02%,1384:0.02%,173556:0.02%,15826 8:0.02%,525:0.02%,1564:0.02%,164976:0. 02%,2184:0.02%,437:0.02%,513:0.02%,17 3304:0.02%,916:0.02%,1040:0.02%,904:0. 02%,812:0.02%,520:0.02%,516:0.02%,504: 0.02%,428:0.02%,4522:0.02%,164172:0.0 2%,100:0.02%,163920:0.02%,60:0.02%,36: 0.02%,24:0.02%,182892:0.02%,1711:0.0 2%,881:0.02%,866:0.02%,4685:0.02%,170 484:0.02%,427:0.02%,2258:0.02%,170184: 0.02%,2222:0.02%,178080:0.02%,178056: 0.02%,177732:0.02%,177648:0.02%,30871 2:0.02%,1366:0.02%,958:0.02%,942:0.0 2%,642:0.02%,911:0.02%,542:0.02%,478: 0.02%,470:0.02%,446:0.02%,176568:0.0 2%,422:0.02%,2307:0.02%,250:0.02%,214 7:0.02%,22:0.02%,167352:0.02%,1069:0.0 2%,1057:0.02%,1029:0.02%,897:0.02%]
Total Length of Bwd Packets	[0:98.36%,12:0.47%,174:0.13%,1124:0.0 7%,13590:0.07%,33:0.07%,137:0.05%,521: 0.05%,10289:0.03%,656:0.03%,1065:0.0 3%,50:0.03%,1618:0.02%,4809:0.02%,419: 0.02%,27062:0.02%,26608:0.02%,124616: 0.02%,9664:0.02%,3509:0.02%,903:0.02%, 14520:0.02%,474039:0.02%,967:0.02%,477 59:0.02%,4674:0.02%,30089:0.02%,1511: 0.02%,29017:0.02%,13463:0.02%,5712:0.0 2%,426:0.02%,22020:0.02%,103864:0.0 2%,13692:0.02%,7517:0.02%,23253:0.0 2%,2547:0.02%,53261:0.02%,28468:0.0 2%,223363:0.02%,28286:0.02%,22:0.02%, 6145:0.02%,12252:0.02%,24414:0.02%,119 50:0.02%]
Fwd Packet Length Max	[0:98.38%,12:0.47%,174:0.13%,523:0.07%, 33:0.07%,520:0.05%,137:0.05%,50:0.03%, 394:0.03%,470:0.03%,656:0.03%,642:0.0 3%,452:0.03%,513:0.03%,516:0.02%,504: 0.02%,525:0.02%,492:0.02%,664:0.02%,10 16:0.02%,460:0.02%,456:0.02%,428:0.0 2%,437:0.02%,453:0.02%,461:0.02%,591: 0.02%,529:0.02%,490:0.02%,515:0.02%,47 9:0.02%,427:0.02%,662:0.02%,542:0.02%, 538:0.02%,522:0.02%,486:0.02%,865:0.0 2%,478:0.02%,458:0.02%,446:0.02%,422: 0.02%,418:0.02%,22:0.02%,527:0.02%,86 9:0.02%]
Fwd Packet Length Min	[0:99.19%,12:0.47%,174:0.13%,33:0.07%,1 37:0.05%,50:0.03%,656:0.03%,22:0.02%]

Fwd Packet Length Mean	[0.0:98.38%,12.0:0.47%,174.0:0.13%,33.0:0.07%,137.0:0.05%,656.0:0.03%,113.0:0.03%,50.0:0.03%,58.11111111:0.03%,67.75:0.02%,173.0:0.02%,47.54545455:0.02%,128.4705882:0.02%,68.42857143:0.02%,109.25:0.02%,37.66666667:0.02%,47.71111111:0.02%,183.2:0.02%,75.36666667:0.02%,117.5:0.02%,111.5:0.02%,128.25:0.02%,25.837209299999998:0.02%,31.02649007:0.02%,162.4:0.02%,74.28571429:0.02%,107.33333329999999:0.02%,22.0:0.02%,107.0:0.02%,82.81818182:0.02%,39.77586207:0.02%,75.5:0.02%,342.2:0.02%,7.761363600000005:0.02%,57.77777778:0.02%,66.8125:0.02%,96.22222222:0.02%,75.0:0.02%,106.75:0.02%,17.83333333:0.02%,134.5714286:0.02%,21.35714286:0.02%,102.9:0.02%,88.1:0.02%,25.22580645:0.02%,87.0:0.02%,70.33333333:0.02%,87.16666667:0.02%,78.8:0.02%,64.5:0.02%,188.16666669999998:0.02%,168.0:0.02%,98.5:0.02%,119.5:0.02%]
Fwd Packet Length Std	[0.0:99.19%,174.33333330000002:0.03%,226.0:0.03%,104.6463917:0.02%,157.07476830000002:0.02%,182.4335495:0.02%,173.33333330000002:0.02%,218.5:0.02%,195.5329934:0.02%,468.67387379999997:0.02%,251.00039840000002:0.02%,96.69257833:0.02%,235.0:0.02%,188.83099019999997:0.02%,214.0:0.02%,256.5:0.02%,191.91775080000002:0.02%,185.87477859999998:0.02%,157.6904332:0.02%,172.2807786:0.02%,229.8708747:0.02%,174.95833100000002:0.02%,184.9036604:0.02%,130.9684632:0.02%,132.0883011:0.02%,59.38243979:0.02%,290.9845357:0.02%,145.5969531:0.02%,176.2021566:0.02%,216.93237549999998:0.02%,191.1967428:0.02%,213.513855899999998:0.02%,198.4313483:0.02%,222.53718790000002:0.02%,191.62593769999998:0.02%,107.0:0.02%,213.5:0.02%,213.10560759999998:0.02%,230.2272986:0.02%,128.0063404:0.02%,239.9613064:0.02%,183.7285021:0.02%,241.32113399999997:0.02%,197.0:0.02%,235.5353026:0.02%,239.0:0.02%,223.0:0.02%]
Bwd Packet Length Max	[0:98.36%,1460:0.56%,12:0.47%,174:0.13%,33:0.07%,1124:0.07%,521:0.07%,137:

	0.05%,544:0.05%,50:0.03%,656:0.03%,22:0.02%,887:0.02%,426:0.02%,1450:0.02%,419:0.02%,903:0.02%]
Bwd Packet Length Min	[0:99.19%,12:0.47%,174:0.13%,33:0.07%,137:0.05%,50:0.03%,656:0.03%,22:0.02%]
Bwd Packet Length Mean	[0.0:98.36%,12.0:0.47%,174.0:0.13%,970.7142857000001:0.07%,281.0:0.07%,33.0:0.07%,137.0:0.05%,130.25:0.03%,213.0:0.03%,656.0:0.03%,50.0:0.03%,857.4166667000002:0.03%,269.6666667:0.02%,601.125:0.02%,322.3333333:0.02%,225.49152540000003:0.02%,225.75:0.02%,835.222222:0.02%,1404.798742:0.02%,841.4375:0.02%,968.0:0.02%,584.8333332999999:0.02%,1047.6296300000001:0.02%,362.7125:0.02%,104.2:0.02%,435.72027970000005:0.02%,509.4:0.02%,682.7777778:0.02%,22.0:0.02%,1127.583333:0.02%,1011.0:0.02%,104.75:0.02%,1061.478261:0.02%,1348.883117:0.02%,459.22115379999997:0.02%,106.5:0.02%,359.53846150000004:0.02%,412.5797101:0.02%,816.8:0.02%,966.4:0.02%,634.6666666667000001:0.02%,1268.119048:0.02%,978.0:0.02%,1378.0203490000001:0.02%,957.3913043:0.02%,995.8333332999998:0.02%,377.75:0.02%,271.0720721:0.02%]
Bwd Packet Length Std	[0.0:99.17%,690.0637015:0.07%,562.0:0.07%,291.7755987:0.03%,260.5:0.03%,725.4349311000001:0.02%,587.0726876:0.02%,688.5997386:0.02%,232.9982833:0.02%,504.7994016:0.02%,481.57475980000004:0.02%,604.1964879:0.02%,255.2859508:0.02%,663.712714:0.02%,702.2473312999999:0.02%,677.1028782000001:0.02%,534.0346714:0.02%,447.4681624:0.02%,726.8545906:0.02%,252.4444488:0.02%,743.8685666:0.02%,744.8832072:0.02%,696.4216928:0.02%,213.0:0.02%,451.5:0.02%,256.5537405:0.02%,346.1718836:0.02%,676.7229686000001:0.02%,525.5079168999999:0.02%,507.1362766:0.02%,281.6564101:0.02%,630.9314809:0.02%,481.80737989999994:0.02%,706.9153061000001:0.02%,209.5:0.02%,743.3669649:0.02%,709.8815394:0.02%,420.66835710000004:0.02%,561.2696546:0.02%,677.0074302:0.02%,649.6204486:0.02%,662.3472331:0.02%]
Flow Bytes/s	[0.0:98.36%,86.79493345:0.02%,160.56675

	2:0.02%,100.0242059:0.02%,231.88645290000002:0.02%,231.76460440000002:0.02%,5994.666044:0.02%,4534.795818:0.02%,161.943005:0.02%,553.3755797:0.02%,2487.077804:0.02%,1002.511617:0.02%,5980.38402:0.02%,1260.978578:0.02%,2264.1951480000002:0.02%,4537.326065:0.02%,3029.6139949999997:0.02%,3.7492250819999997:0.02%,2.726962845:0.02%,5425.805964:0.02%,6028.916945:0.02%,371.57306439999996:0.02%,511.98177230000005:0.02%,901.6511531:0.02%,2021.26568:0.02%,6038.634639:0.02%,5571.070350999999:0.02%,6042.541656:0.02%,231.86536349999997:0.02%,6056.027333:0.02%,91.46671747:0.02%,231.83933530000002:0.02%,1627.512498:0.02%,231.85771659999997:0.02%,1664.488941:0.02%,107.5352348:0.02%,3659.445029:0.02%,1.411469709000000001:0.02%,7112.15395:0.02%,160.7814955:0.02%,1.260757333:0.02%,758.0914124:0.02%,7.999016121:0.02%,231.7967915:0.02%,11435.83227:0.02%,2.307098141:0.02%,54.74638139:0.02%,257.8953901:0.02%,6039.736252000001:0.02%,2999.294434:0.02%,6036.697768:0.02%,13299.55715:0.02%,6031.291684000001:0.02%,160.7008505:0.02%,3398.862114:0.02%,6034.757729999999:0.02%,34.44430769:0.02%,87173.55993999999:0.02%,1361.601111:0.02%,1121.7408779999998:0.02%,216.52483430000004:0.02%,196.5004712:0.02%,105.69759209999998:0.02%,2578.757188:0.02%,1050.056419:0.02%,6125.355070000005:0.02%,6053.571929:0.02%,10215.55967:0.02%,54.77154071:0.02%,187963.378:0.02%,2.570870876:0.02%,617.2655298:0.02%,753.1530952:0.02%,3227.009627:0.02%,25977.75615:0.02%,1717.395638:0.02%,2698.7367489999997:0.02%,231.687068:0.02%,1.4114380830000002:0.02%,1460.74078:0.02%,2253.378179:0.02%,54.76700835:0.02%,6041.761453:0.02%,2979.396561:0.02%,5958.363579:0.02%,12137.58203:0.02%,107.3739028:0.02%,547.2908537000001:0.02%,37798.84005:0.02%,175.1206065:0.02%,2.6661977119999998:0.02%,55867.55979:0.02%,4.798660214:0.02%,161.0144

	889:0.02%,231.8184089:0.02%,6061.587836:0.02%,9892.928001:0.02%,10670.139009999999:0.02%]
Flow Packets/s	3886 (65%)
Flow IAT Mean	3496 (58%)
Flow IAT Std	2737 (46%)
Flow IAT Max	2810 (47%)
Flow IAT Min	2733 (46%)
Fwd IAT Total	1682 (28%)
Fwd IAT Mean	1719 (29%)
Fwd IAT Std	300 (5%)
Fwd IAT Max	1678 (28%)
Fwd IAT Min	1710 (28%)
Bwd IAT Total	1051 (17%)
Bwd IAT Mean	1105 (18%)
Bwd IAT Std	412 (6%)
Bwd IAT Max	1037 (17%)
Bwd IAT Min	848 (14%)
Fwd PSH Flags	[0:100.0%]
Bwd PSH Flags	[0:99.97%,1:0.03%]
Fwd URG Flags	[0:100.0%]
Bwd URG Flags	[0:100.0%]
Fwd Header Length	[64:50.22%,0:28.23%,20:13.48%,32:2.82%,84:1.62%,96:0.81%,72:0.52%,116:0.39%,40:0.19%,80:0.17%,24:0.15%,8:0.15%,148:0.13%,180:0.07%,160:0.05%,120:0.05%,280:0.05%,100:0.05%,104:0.03%,240:0.03%,16:0.03%,136:0.03%,140:0.03%,52:0.03%,48:0.03%,200:0.03%,60:0.02%,168:0.02%,220:0.02%,112:0.02%,121928:0.02%,105512:0.02%,109280:0.02%,113656:0.02%,113456:0.02%,111568:0.02%,118720:0.02%,118704:0.02%,118488:0.02%,118432:0.02%,3520:0.02%,109984:0.02%,117712:0.02%,3020:0.02%,109448:0.02%,1720:0.02%,320:0.02%,1240:0.02%,1200:0.02%,1160:0.02%,115728:0.02%,115704:0.02%,205808:0.02%,900:0.02%,115536:0.02%,840:0.02%,720:0.02%,480:0.02%,340:0.02%,244:0.02%]
Bwd Header Length	[32:49.38%,40:22.49%,64:8.86%,20:5.06%,84:3.43%,128:2.53%,116:2.14%,60:1.84%,96:1.21%,8:0.81%,180:0.56%,160:0.54%,0:0.15%,104:0.15%,192:0.12%,304:0.08%,148:0.07%,204:0.05%,484:0.05%,124:0.05%,264:0.05%,324:0.03%,144:0.03%,5744:0.02%,1404:0.02%,3204:0.02%,2384:0.02%,2

	244:0.02%,2104:0.02%,1624:0.02%,1564:0.02%,80:0.02%,564:0.02%,864:0.02%,184:0.02%,504:0.02%,284:0.02%,136:0.02%,224:0.02%,344:0.02%,6904:0.02%]
Fwd Packets/s	3039 (51%)
Bwd Packets/s	3780 (63%)
Min Packet Length	[0:99.19%,12:0.47%,174:0.13%,33:0.07%,137:0.05%,50:0.03%,656:0.03%,22:0.02%]
Max Packet Length	[0:98.36%,1460:0.56%,12:0.47%,174:0.13%,33:0.07%,1124:0.07%,521:0.07%,544:0.05%,137:0.05%,50:0.03%,656:0.03%,887:0.02%,22:0.02%,1450:0.02%,428:0.02%,427:0.02%,903:0.02%]
Packet Length Mean	[0.0:98.36%,12.0:0.47%,174.0:0.13%,33.0:0.07%,137.0:0.05%,588.0416667000001:0.03%,656.0:0.03%,50.0:0.03%,175.11111110000002:0.03%,101.66666670000001:0.02%,391.4117647:0.02%,667.6842105000001:0.02%,808.4328357999999:0.02%,564.5714286:0.02%,22.0:0.02%,274.6226415:0.02%,416.6875:0.02%,639.2894737:0.02%,157.33333330000002:0.02%,298.4:0.02%,180.0909091:0.02%,178.0:0.02%,542.8076923:0.02%,266.61538459999997:0.02%,201.22674419999998:0.02%,515.7619048:0.02%,170.63636359999998:0.02%,496.1875:0.02%,155.64640880000002:0.02%,582.92:0.02%,582.5454545:0.02%,247.33333300000002:0.02%,696.3333332999999:0.02%,107.44444440000001:0.02%,277.4166667:0.02%,873.0083332999999:0.02%,326.2:0.02%,307.15337420000003:0.02%,355.46666669999996:0.02%,94.88888889:0.02%,269.6666667:0.02%,94.0:0.02%,235.5:0.02%,513.3571429:0.02%,295.2077626:0.02%,510.80769230000004:0.02%,514.5238095:0.02%,917.0121951:0.02%,607.2894737:0.02%,566.0555555999999:0.02%,667.1590909:0.02%,83.18181818:0.02%,203.5:0.02%,587.9166667000001:0.02%,912.4856046:0.02%]
Packet Length Std	[0.0:99.17%,385.9573955:0.03%,702.6950281000001:0.03%,702.7073679:0.02%,674.7120765999999:0.02%,480.764591:0.02%,478.7701877:0.02%,188.2899655:0.02%,693.9186417999999:0.02%,684.002412:0.02%,187.233981:0.02%,411.39838210000005:0.02%,690.9031150000001:0.02%,693.66

	77258999999:0.02%,240.21918019999995:0.02%,538.8993309:0.02%,685.9051227000001:0.02%,388.3786812:0.02%,680.3891258:0.02%,638.2712818:0.02%,584.2947735:0.02%,684.4867857999999:0.02%,691.6675857000001:0.02%,467.4347513:0.02%,692.6856152:0.02%,327.0691823:0.02%,214.0263483:0.02%,458.8885739:0.02%,186.5361895:0.02%,594.7857798:0.02%,701.2648073:0.02%,485.5893621:0.02%,665.7530251000001:0.02%,652.8992582999999:0.02%,683.0912037:0.02%,661.9633776000001:0.02%,259.4268683:0.02%,684.9335545:0.02%,257.78576380000004:0.02%,411.633332:0.02%,204.22169330000003:0.02%,699.057667:0.02%,396.24740519999995:0.02%,251.0842307:0.02%,420.04090690000004:0.02%,382.78377539999997:0.02%,714.3668149:0.02%,693.9793671:0.02%]
Packet Length Variance	[0.0:99.17%,148963.1111:0.03%,493780.3025:0.03%,407390.2292:0.02%,470465.8374:0.02%,169248.6288:0.02%,235797.0286:0.02%,477347.1144:0.02%,290412.4889:0.02%,45807.277780000004:0.02%,106974.25:0.02%,210578.7232:0.02%,41706.5:0.02%,353770.1238:0.02%,468522.16:0.02%,57705.25455:0.02%,229220.8927:0.02%,462929.3625:0.02%,481607.3619:0.02%,35453.111110000005:0.02%,443227.0905:0.02%,510319.9462:0.02%,466613.5926:0.02%,63043.29091:0.02%,146523.4187:0.02%,455236.3862:0.02%,481174.9139:0.02%,488681.6218:0.02%,34795.75:0.02%,481523.0815:0.02%,176434.3635:0.02%,491772.33:0.02%,66453.5:0.02%,479813.3615:0.02%,438195.5132:0.02%,467859.2997:0.02%,426277.4415:0.02%,469133.974:0.02%,35056.56364:0.02%,169442.0:0.02%,341400.3824:0.02%,231134.592:0.02%,150838.0:0.02%,478404.0491:0.02%,218495.2467:0.02%,67302.3:0.02%,493797.6449:0.02%,157012.0062:0.02%]
FIN Flag Count	[0:75.68%,1:24.32%]
SYN Flag Count	[1:69.63%,0:30.37%]
RST Flag Count	[0:100.0%]
PSH Flag Count	[0:99.97%,1:0.03%]
ACK Flag Count	[0:62.42%,1:37.58%]
URG Flag Count	[0:100.0%]

CWE Flag Count	[0:100.0%]
ECE Flag Count	[0:100.0%]
Down/Up Ratio	[0:78.32%,2:9.82%,1:9.11%,4:2.14%,3:0.61%]
Average Packet Size	[0.0:98.36%,217.5:0.13%,18.0:0.07%,49.5:0.07%,14.4:0.07%,159.83333330000002:0.05%,613.6086957:0.03%,749.7142857000001:0.03%,197.0:0.03%,202.40350880000003:0.02%,58.33333333:0.02%,12.00083085:0.02%,33.0:0.02%,880.3445377999999:0.02%,12.00087706:0.02%,12.0007873:0.02%,302.63636360000004:0.02%,12.00080868:0.02%,349.5:0.02%,532.3703704:0.02%,198.1:0.02%,12.00046644:0.02%,277.23809520000003:0.02%,12.00081054:0.02%,12.00081549:0.02%,14.0:0.02%,16.0:0.02%,541.55:0.02%,444.46666669999996:0.02%,920.755102:0.02%,12.00081015:0.02%,106.75:0.02%,15.0:0.02%,66.66666667:0.02%,200.25:0.02%,277.28:0.02%,12.00084608:0.02%,531.24:0.02%,914.2403846000001:0.02%,610.2857142999999:0.02%,249.312:0.02%,156.5111111:0.02%,309.0493827:0.02%,12.00080857:0.02%,12.00087841:0.02%,303.375:0.02%,12.00087279:0.02%,12.00084459:0.02%,12.00082965:0.02%,120.875:0.02%,682.6744186000001:0.02%,564.52:0.02%,282.6:0.02%,331.55555560000005:0.02%,91.5:0.02%,613.4782609:0.02%,415.875:0.02%,105.75:0.02%,12.0008604:0.02%,380.85714289999993:0.02%,656.567567676:0.02%,177.0:0.02%,114.375:0.02%,623.7027027:0.02%,585.4814815000001:0.02%,187.7:0.02%,540.25:0.02%,12.000909978:0.02%,820.6818182:0.02%,232.5714286:0.02%,295.8832952:0.02%,607.2083332999999:0.02%,685.7297297:0.02%,599.3529412:0.02%,529.2666667:0.02%,713.3170732000001:0.02%,12.00082947:0.02%]
Avg Fwd Segment Size	[0.0:98.38%,12.0:0.47%,174.0:0.13%,33.0:0.07%,137.0:0.05%,656.0:0.03%,113.0:0.03%,50.0:0.03%,58.11111111:0.03%,67.75:0.02%,173.0:0.02%,47.54545455:0.02%,128.4705882:0.02%,68.42857143:0.02%,109.25:0.02%,37.66666667:0.02%,47.71111111:0.02%,183.2:0.02%,75.36666667:0.02%,117.5:0.02%,111.5:0.02%,128.25:0.02%,25.837209299999998:0.02%,31.02649007:0.0

	2%,162.4:0.02%,74.28571429:0.02%,107.3 3333329999999:0.02%,22.0:0.02%,107.0:0. 02%,82.81818182:0.02%,39.77586207:0.0 2%,75.5:0.02%,342.2:0.02%,7.7613636360 000005:0.02%,57.77777778:0.02%,66.812 5:0.02%,96.22222222:0.02%,75.0:0.02%,10 6.75:0.02%,17.83333333:0.02%,134.57142 86:0.02%,21.35714286:0.02%,102.9:0.02%, 88.1:0.02%,25.22580645:0.02%,87.0:0.0 2%,70.33333333:0.02%,87.16666667:0.0 2%,78.8:0.02%,64.5:0.02%,188.166666699 99998:0.02%,168.0:0.02%,98.5:0.02%,119. 5:0.02%]
Avg Bwd Segment Size	[0.0:98.36%,12.0:0.47%,174.0:0.13%,970.7 142857000001:0.07%,281.0:0.07%,33.0:0.0 7%,137.0:0.05%,130.25:0.03%,213.0:0.0 3%,656.0:0.03%,50.0:0.03%,857.41666670 00002:0.03%,269.6666667:0.02%,601.125: 0.02%,322.3333333:0.02%,225.491525400 00003:0.02%,225.75:0.02%,835.2222222:0. 02%,1404.798742:0.02%,841.4375:0.02%,9 68.0:0.02%,584.8333332999999:0.02%,104 7.6296300000001:0.02%,362.7125:0.02%,1 04.2:0.02%,435.72027970000005:0.02%,50 9.4:0.02%,682.7777778:0.02%,22.0:0.02%, 1127.583333:0.02%,1011.0:0.02%,104.75: 0.02%,1061.478261:0.02%,1348.883117:0. 02%,459.22115379999997:0.02%,106.5:0.0 2%,359.53846150000004:0.02%,412.57971 01:0.02%,816.8:0.02%,966.4:0.02%,634.66 66667000001:0.02%,1268.119048:0.02%,9 78.0:0.02%,1378.0203490000001:0.02%,95 7.3913043:0.02%,995.8333332999998:0.0 2%,377.75:0.02%,271.0720721:0.02%]
Fwd Avg Bytes/Bulk	[0:100.0%]
Fwd Avg Packets/Bulk	[0:100.0%]
Fwd Avg Bulk Rate	[0:100.0%]
Bwd Avg Bytes/Bulk	[0:100.0%]
Bwd Avg Packets/Bulk	[0:100.0%]
Bwd Avg Bulk Rate	[0:100.0%]
Subflow Fwd Packets	[2:50.42%,0:28.08%,1:16.44%,3:3.14%,4:0. 71%,5:0.32%,6:0.1%,9:0.07%,14:0.05%,8: 0.05%,12:0.03%,11:0.03%,7:0.03%,10:0.0 3%,14182:0.02%,14463:0.02%,13946:0.0 2%,14811:0.02%,176:0.02%,45:0.02%,60: 0.02%,36:0.02%,24:0.02%,16:0.02%,17:0.0 2%,13681:0.02%,14442:0.02%,15241:0.0 2%,14840:0.02%,14804:0.02%,151:0.02%,

	13748:0.02%,13660:0.02%,25726:0.02%,86:0.02%,14466:0.02%,62:0.02%,58:0.02%,42:0.02%,14207:0.02%,13189:0.02%,14714:0.02%,14838:0.02%]
Subflow Fwd Bytes	[0:98.38%,522:0.15%,48:0.07%,33:0.07%,523:0.07%,12:0.07%,685:0.05%,452:0.03%,3936:0.03%,394:0.03%,173592:0.02%,1288:0.02%,1384:0.02%,173556:0.02%,158268:0.02%,525:0.02%,1564:0.02%,164976:0.02%,2184:0.02%,437:0.02%,513:0.02%,173304:0.02%,916:0.02%,1040:0.02%,904:0.02%,812:0.02%,520:0.02%,516:0.02%,504:0.02%,428:0.02%,4522:0.02%,164172:0.02%,100:0.02%,163920:0.02%,60:0.02%,36:0.02%,24:0.02%,182892:0.02%,1711:0.02%,881:0.02%,866:0.02%,4685:0.02%,170484:0.02%,427:0.02%,2258:0.02%,170184:0.02%,2222:0.02%,178080:0.02%,178056:0.02%,177732:0.02%,177648:0.02%,308712:0.02%,1366:0.02%,958:0.02%,942:0.02%,642:0.02%,911:0.02%,542:0.02%,478:0.02%,470:0.02%,446:0.02%,176568:0.02%,422:0.02%,2307:0.02%,250:0.02%,2147:0.02%,22:0.02%,167352:0.02%,1069:0.02%,1057:0.02%,1029:0.02%,897:0.02%]
Subflow Bwd Packets	[1:55.4%,2:31.35%,3:6.48%,4:4.84%,6:0.71%,5:0.67%,14:0.08%,9:0.05%,12:0.05%,23:0.05%,15:0.03%,16:0.02%,80:0.02%,104:0.02%,344:0.02%,8:0.02%,24:0.02%,159:0.02%,13:0.02%,69:0.02%,111:0.02%,10:0.02%,42:0.02%,118:0.02%,286:0.02%,27:0.02%,77:0.02%]
Subflow Bwd Byte	[0:98.36%,12:0.47%,174:0.13%,1124:0.07%,13590:0.07%,33:0.07%,137:0.05%,521:0.05%,10289:0.03%,656:0.03%,1065:0.03%,50:0.03%,1618:0.02%,4809:0.02%,419:0.02%,27062:0.02%,26608:0.02%,124616:0.02%,9664:0.02%,3509:0.02%,903:0.02%,14520:0.02%,474039:0.02%,967:0.02%,47759:0.02%,4674:0.02%,30089:0.02%,1511:0.02%,29017:0.02%,13463:0.02%,5712:0.02%,426:0.02%,22020:0.02%,103864:0.02%,13692:0.02%,7517:0.02%,23253:0.02%,2547:0.02%,53261:0.02%,28468:0.02%,223363:0.02%,28286:0.02%,22:0.02%,6145:0.02%,12252:0.02%,24414:0.02%,11950:0.02%]
Init_Win_bytes_forward	[-1:100.0%]

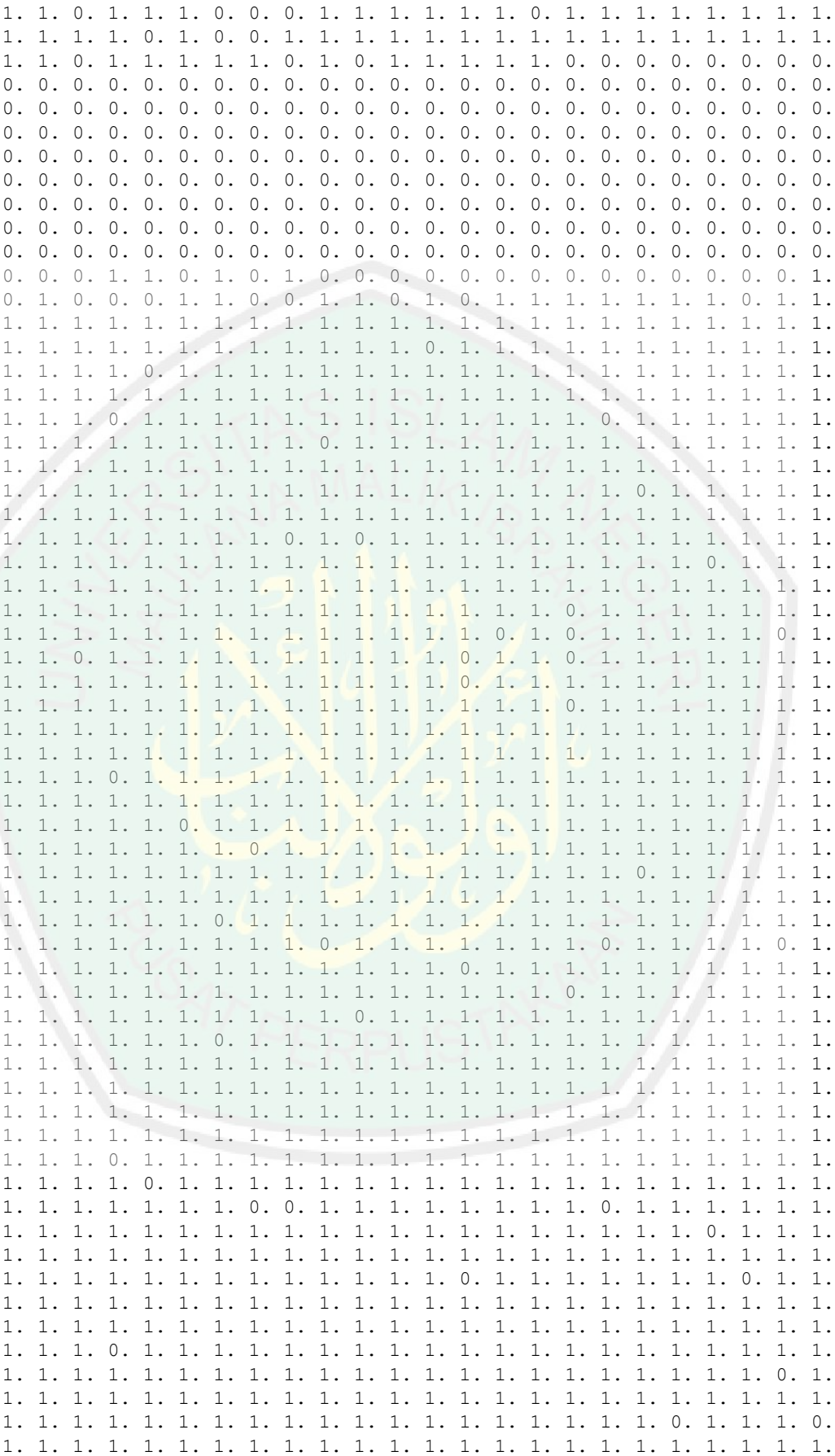
Init_Win_bytes_backward	[64240:49.38%,0:18.8%,5840:11.29%,92:7.98%,2053:7.39%,256:2.89%,-1:0.96%,108:0.35%,125:0.22%,2051:0.13%,2049:0.1%,2048:0.1%,2052:0.05%,142:0.05%,2050:0.05%,175:0.03%,176:0.02%,2129:0.02%,112:0.02%,164:0.02%,2047:0.02%,244:0.02%,252:0.02%,129:0.02%,146:0.02%,250:0.02%,155:0.02%,167:0.02%,109:0.02%]
act_data_pkt_fwd	[0:98.38%,1:0.54%,2:0.29%,3:0.2%,5:0.13%,4:0.1%,6:0.05%,14466:0.02%,8:0.02%,14463:0.02%,14811:0.02%,13946:0.02%,14182:0.02%,14442:0.02%,13681:0.02%,14714:0.02%,13189:0.02%,14207:0.02%,15241:0.02%,25726:0.02%,13660:0.02%,13748:0.02%,14804:0.02%,14840:0.02%,14838:0.02%]
min_seg_size_forward	[0:100.0%]
Active Mean	1863 (31%)
Active Std	[0.0:97.98%,0.7071067809999999:0.15%,2.121320344:0.15%,1.4142135619999998:0.12%,2.828427125:0.07%,3.535533906000004:0.07%,0.577350269:0.05%,2.081665999:0.03%,4.949747468:0.03%,4.242640687:0.03%,3.0:0.03%,1.0:0.03%,1.732050808:0.03%,2.309401077:0.03%,4.725815626:0.03%,2.0:0.03%,2.886751346:0.03%,17.785762100000003:0.02%,8.326663997999999:0.02%,6.3639610310000005:0.02%,5.859465277000001:0.02%,19.15724406:0.02%,17.67295486:0.02%,9.899494937:0.02%,14.8492424:0.02%,2544427.0919999997:0.02%,18.24828759:0.02%,4.509249753:0.02%,2946323.038:0.02%,4.041451884:0.02%,139513.5821:0.02%,2965717.563:0.02%,2603496.458:0.02%,414448.0124:0.02%,21.501937899999998:0.02%,2873392.752:0.02%,7.778174592999999:0.02%,6.350852961:0.02%,2749050.853:0.02%,8.485281374:0.02%,2.516611478:0.02%,2235670.849:0.02%,16.26345597:0.02%,15.62049935:0.02%,2742760.4310000003:0.02%,19.79898987:0.02%,7.637626158:0.02%,2500432.136:0.02%,1.527525232:0.02%,2961483.4080000003:0.02%,6.557438524:0.02%,2793729.395:0.02%,2083878.535:0.02%,13.42882472:0.02%,8.717797887:0.02%,2936881.042:0.02%,2431339.7630000003:0.02%,2971476.9480000003:0.02%,5.291502622:

	0.02%,4.618802154:0.02%,2782092.539:0.02%,10.58300524:0.02%,19.09188309:0.02%,9.192388155:0.02%,3.055050463:0.02%,22.36813209:0.02%,2787868.187:0.02%,2192312.133:0.02%,17.34935157:0.02%,2160442.687:0.02%,7.094598885:0.02%,20.0748599:0.02%,3.2145502539999997:0.02%,2689039.4080000003:0.02%,2339257.017:0.02%,2931179.64:0.02%,28.68797658:0.02%,2944190.4039999996:0.02%,9.073771726:0.02%,9.291573243:0.02%,3069107.8880000003:0.02%]
Active Max	1846 (31%)
Active Min	1806 (30%)
Idle Mean	1635 (27%)
Idle Std	125 (2%)
Idle Max	1573 (26%)
Idle Min	1648 (27%)
Label	[DDoS:53.1%,BENIGN:46.9%]

Hasil Pengujian dengan Fitur Lengkap

```
[0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 1. 0. 1. 0. 1. 1. 0. 0. 0. 1. 1. 0. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 0. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 0.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0.
 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1.
 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 0. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 0. 1. 1. 1. 1. 0. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1.
 1. 0. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1.
 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0.
 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 0. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 0. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0.
 0. 0. 1. 1. 0. 1. 1. 1. 0. 1. 1. 0. 0. 1. 1. 0. 1. 1. 1. 0. 1. 1. 0. 1.
 0. 1. 1. 0. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 0. 1. 0. 1. 1. 1. 1. 1. 1. 1.
 1. 0. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 0. 1. 0. 1. 1. 1. 1. 1. 1. 1. 0.
```

```
1. 1. 1. 1. 1. 0. 0. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1.
1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 0. 1. 1. 1. 0. 1. 1. 1. 1. 1.
0. 1. 0. 0. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 0. 0. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 1. 1. 1. 0. 0. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0.
0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 1.
1. 0. 0. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 0. 1. 0. 0. 0. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 0. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1.
1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 0. 1. 1. 0. 0. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1.
1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 0. 0. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 0. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 0.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 0.
1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1.
1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 0.
1. 1. 1. 1. 1. 0. 1. 1. 0. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 1. 0.
1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1.
1. 1. 1. 0. 1. 1. 1. 1. 0. 1. 1. 0. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 1. 1. 1. 0.
1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 0. 0. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 0. 1. 0. 1. 1. 1. 1. 1. 1.
1. 1. 0. 1. 0. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
```

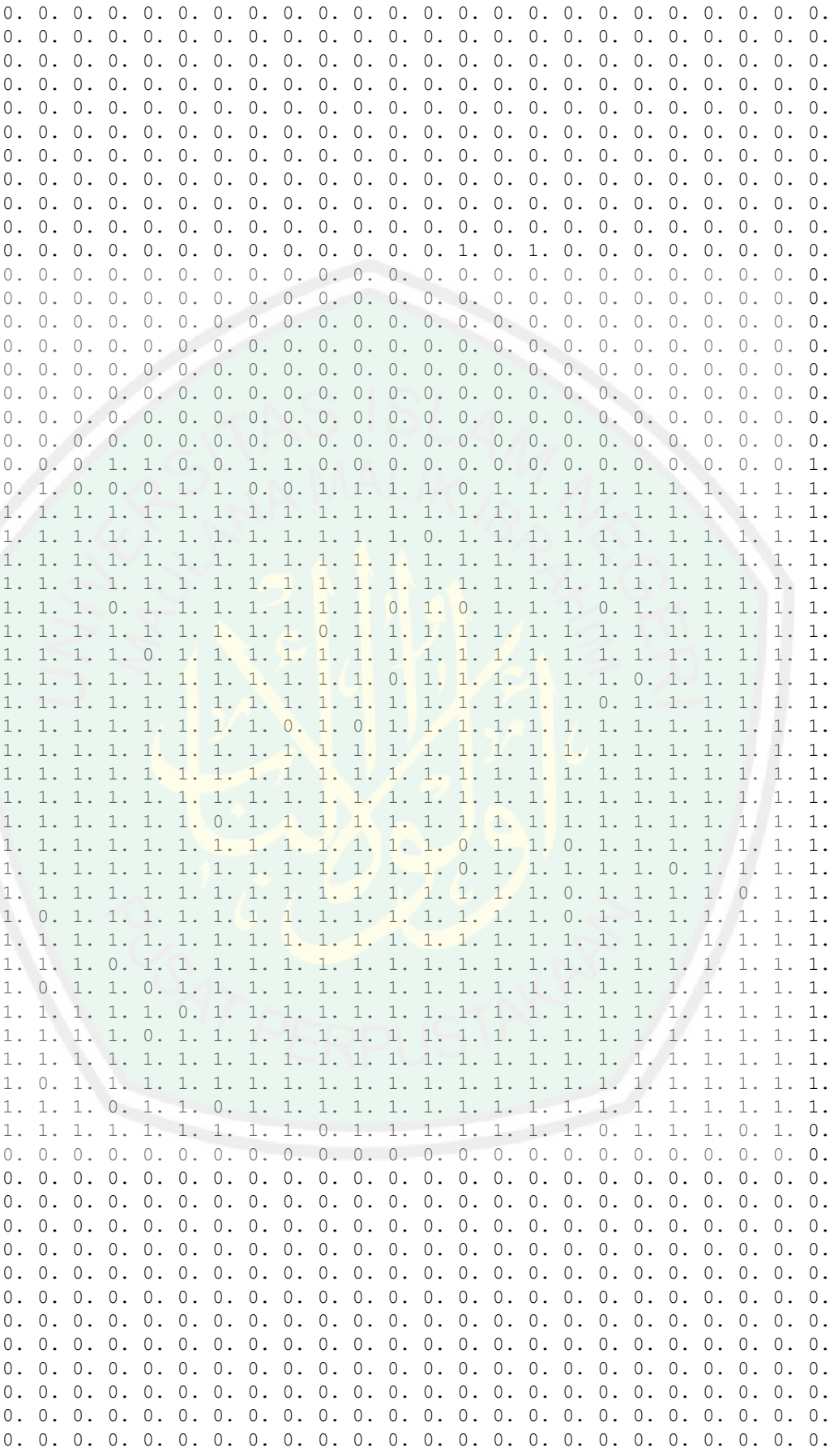


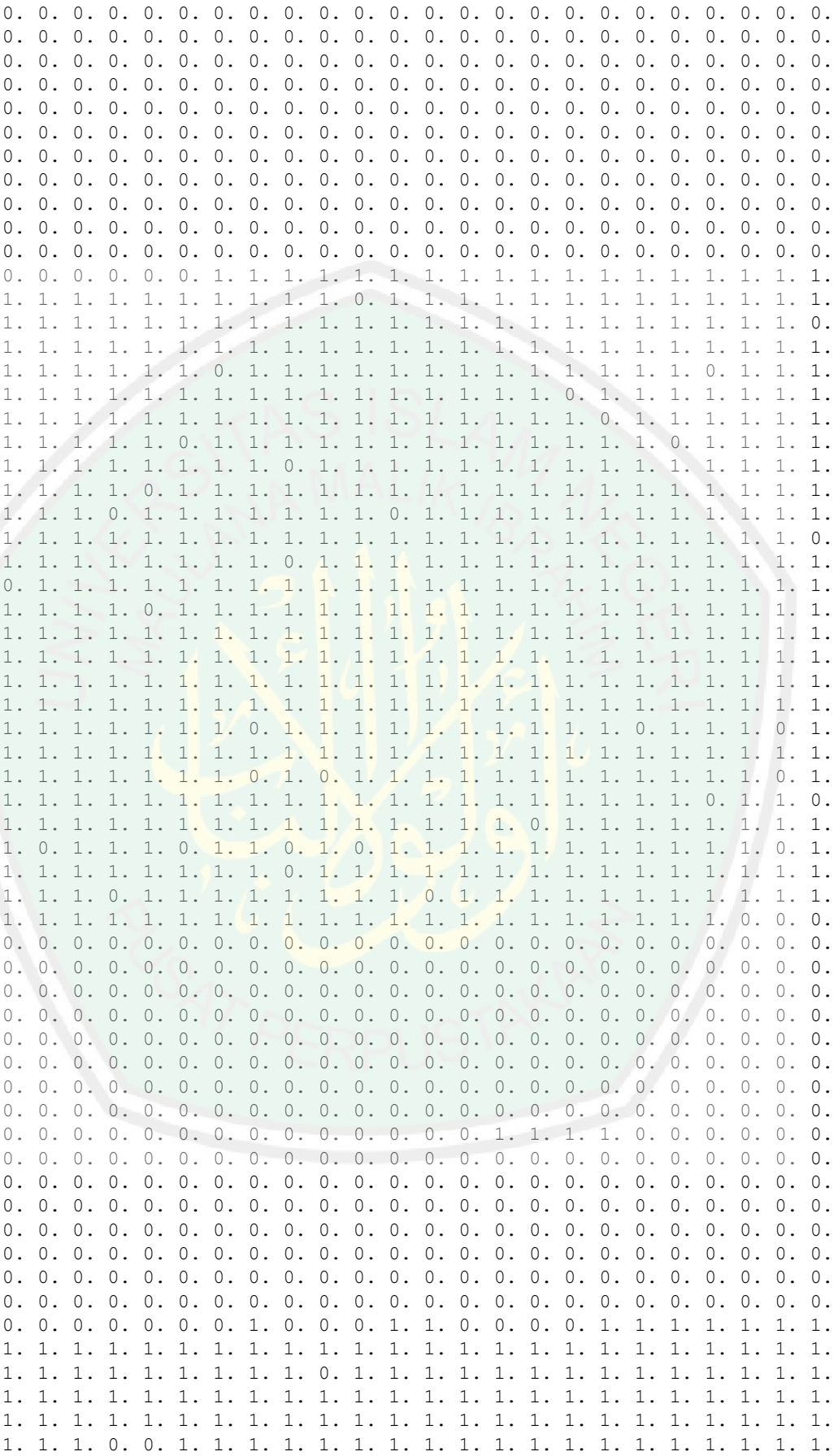
```
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 0. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 0.
1. 0. 1. 1. 1. 1. 1. 1. 0. 1. 1. 0. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0.
1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1.
1. 0. 1. 1. 1. 1. 0. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 0.
0. 1. 1. 1. 1. 0. 1. 1. 1. 0. 0. 1. 1. 1. 0. 1. 1. 1. 0. 1. 1. 0. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 0. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 0. 0. 1. 1. 1. 1. 1. 1. 1. 1.
0. 1. 1. 0. 1. 1. 1. 1. 0. 1. 1. 0. 0. 0. 1. 1. 0. 1. 0. 1. 1. 0. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 0. 1. 1. 1. 0. 1. 1. 1. 0. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 1. 0. 1. 0. 1. 1. 0. 0. 0. 0. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 1.
1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 0. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1.
1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 0. 1. 1.
```

```
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1.
 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 0. 1.
 0. 1. 1. 0. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 0. 1. 1. 1. 1. 1. 1.
 1. 0. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 0. 1. 1. 1. 1. 1. 0. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 0. 1. 1.
 0. 1. 1. 0. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 0. 1. 1. 0. 1. 1. 0. 1. 1. 0.
 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 0. 1. 1. 1. 0. 1. 1. 1.
 1. 1. 0. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 0. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 0. 1. 0. 1. 1. 1. 1. 1. 0. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 0.
 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 0. 1. 0. 0. 1. 1. 1. 1. 0.
 0. 0. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 0. 1. 1. 0. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0.
 0. 1. 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

Hasil Pengujian dengan Fitur Terpilih

```
[0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.
 1. 1. 1. 1. 1. 0. 0. 0. 1. 1. 0. 0. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
```





```
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0.
1. 1. 1. 1. 1. 0. 0. 1. 1. 1. 1. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1.
1. 1. 1. 1. 1. 1. 1. 1. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
```