Deteksi Serangan *Botnet* Pada Jaringan *Internet of Things* Menggunakan Algoritma *Random Forest* (RF)

1st Mohammad Sani Rafsanjani
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia
sanirafsanjani@students.telkomuniversity.ac.id

2nd Vera Suryani
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia
verasuryani@telkomuniversity.ac.id

3rd Rizka Reza Pahlevi
Fakultas Informatika
Universitas Telkom
Bandung, Indonesia
rizkarezapahlevi@telkomuniversity.ac.id

Abstrak

Perkembangan *Internet of Things* belakangan ini sangatlah pesat, hal tersebut terlihat dari meningkatnya jumlah pengguna berbagai perangkat *IoT* dari waktu ke waktu. *IoT* dapat menghubungkan berbagai *device* dan saling bertukar data melalui jaringan internet. Namun dalam pengimplementasian teknologi tersebut, terdapat berbagai macam ancaman. Salah satu ancaman serius pada teknologi *IoT* yaitu serangan *DDOS* melalui perantara *Botnet (Robot Network)*. Serangan tersebut telah menjadi penyebab risiko keamanan yang cukup serius terhadap jaringan Internet selama beberapa tahun ini. seperti masalah yang banyak terjadi pada privasi , keamanan, konfigurasi sistem , kontrol akses, dan verifikasi. Oleh karena itu, perlu adanya sistem pendeteksian serangan *botnet* dengan menggunakan algoritma *Random forest*. Dimana Random forest dipilih karena, algoritma tersebut sangat optimal dalam proses deteksi serangan dengan jumlah data yang besar dibanding algoritma lainnya. Pada tugas akhir ini, menggunakan dataset yang berasal dari *UNSW Canberra* yaitu *Bot-IoT UNSW*-2018 dataset dan *Algoritma Random forest* digunakan pada proses klasifikasi serangan *botnet*. Setelah dilakukan pengujian, Algoritma *Random Forest* dapat bekerja dengan baik dalam melakukan deteksi terhadap serangan botnet. Dimana pada fitur *attack* didapat nilai *accuracy* sebesar 99.27% sedangkan pada fitur *category* nilai *accuracy* sebesar 99.43% dan fitur *subcategory* nilai *accuracy* sebesar 98.86%, *category* 97.97% dan *subcategory* 83.77%, dengan pembagian data *train* 80%,data *test* 30% dan jumlah *n estimators* 200.

Kata Kunci : *Botnet, Internet of Things, Machine Learning, Random Forest*

Abstract

The development of the Internet of Things lately is very rapid, it can be seen from the increasing number of users of various IoT devices from time to time. IoT can connect various devices and exchange data with each other via the internet. However, in implementing this technology, there are various kinds of threats. One of the serious threats to IoT technology is DDOS attacks through Botnet intermediaries (Robot Network). These attacks have been the cause of a fairly serious security risk to the Internet network for several years. such as problems that often occur in privacy, security, system configuration, access control, and verification. Therefore, it is necessary to have a botnet attack detection system using the Random forest algorithm. Where Random forest was chosen because, the algorithm is very optimal in the process of detecting attacks with large amounts of data compared to other algorithms. In this final project, using a dataset originating from UNSW Canberra, namely the Bot-IoT UNSW-2018 dataset and the Random forest algorithm used in the botnet attack classification process. After testing, the Random Forest Algorithm can work well in detecting botnet attacks. Where in the attack feature, the accuracy value is 99.27%. Meanwhile, in the category feature, the accuracy value is 99.43%. and the subcategory feature has an accuracy value of 98.86%

Keywords: Botnet, Internet of Things, Machine Learning, Random Forest

I. PENDAHULUAN

A. Latar Belakang

Internet of Things (IoT) merupakan salah satu contoh perkembangan teknologi *modern* yang ada saat ini. Dengan adanya *IoT* setiap perangkat teknologi berbasis identifikasi seperti sensor, *RFID* dan lain sebagainya dapat saling terhubung ke internet berdasarkan protokol komunikasi standar [1]. Namun seiring dengan pesatnya teknologi *IoT* dalam kehidupan sehari-hari, juga menimbulkan berbagai macam masalah. Masalah yang timbul tersebut seperti masalah privasi, keamanan, konfigurasi sistem, kontrol akses, verifikasi dan lain sebagainya [2]. Selain itu *IoT* juga sangatlah rentan terhadap berbagai macam serangan yang mengakibatkan gangguan dalam proses komunikasi. Menurut kaspersky [3], telah terjadi lebih dari 100 juta serangan pada *IOT* di kuartal pertama 2019. Angka tersebut meningkat tujuh kali lebih banyak dibandingkan kuartal pertama tahun 2018.

Distributed Denial of Service (DDOS) merupakan salah satu ancaman serius yang ada pada jaringan *IoT*. Serangan *DDOS* dapat menyebabkan suatu *server* menjadi sibuk dengan banyaknya permintaan-permintaan sehingga pengguna yang sah atau normal tidak dapat mengakses jaringan tersebut. Dalam perkembangannya seorang *hacker* juga dapat menggunakan *botnet(Robot Network)* dalam melancarkan serangan *DDOS*. *Botnet* merupakan salah satu jenis *malware (Malicious Software)* yang dalam sistem kerjanya dapat dikendalikan oleh seorang *hacker (Botmaster)* secara teknis. Saat ini jenis *botnet* sangatlah beragam seperti *Zeus*, *Tricbot*, *Bonito*, *Virut*, *NSIS.ay*, *Mirai*, dan lain sebagainya. Berbagai penelitian mengenai *botnet* telah dilakukan

dalam beberapa tahun terakhir. Namun untuk tingkat akurasi beberapa algoritma machine learning masih rendah dan perlunya perbaikan. Pada penelitian sebelumnya [4], membahas bagaimana mengklasifikasikan *botnet* pada jaringan *IoT* berdasarkan *Network Traffic*. Pada penelitian [1], menggunakan dataset yang memiliki format *CTU-13*. dan menggunakan empat algoritma klasifikasi yaitu *Support Vector Machine (SVM),Neural Network,Naïve Bayes,dan Decision Tree*. Dari hasil penelitian yang telah dilakukan, klasifikasi dengan bantuan empat algoritma tersebut mendapatkan hasil yang cukup tinggi. Pada proses pengklasifikasian mendapatkan akurasi 99,8% untuk *Support Vector Machine (SVM)*, 54,5% untuk *Neural Network*, dan 98,4% untuk *Naive Bayes* dan 95,7 untuk *Decision tree*. Selanjutnya, pada penelitian [5] melakukan penelitian dengan menggunakan *Naive Bayes* dalam pendeteksian menggunakan dataset *CTU-13*. Untuk meningkatkan akurasi, metode tersebut digabungkan dengan teknik *Synthetic Minority Over-sampling Technique (SMOTE)* dan penambahan metode seleksi fitur untuk pemilihan fitur dari *noise feature* menggunakan metode *Best First Search*. Hasil akurasi yang didapatkan sebesar 96,68% Pada penelitian [6], *Algoritma Decision tree* digunakan dalam pengklasifikasian *botnet*. Dataset yang digunakan pada penelitian tersebut berasal dari *UNSW-NB15* Hasil akurasi yang didapatkan sebesar 92,23%.

Dengan adanya berbagai penelitian tersebut, menjadikan algoritma *machine learning* sangat penting untuk mendeteksi dan mengidentifikasi jenis serangan *botnet* lama maupun baru. Algoritma ini memiliki peran penting dalam menyediakan fungsi deteksi intrusi berbasis *anomali* dalam jaringan *IoT*. Pada penelitian [7] menyebutkan bahwa *algoritma Random Forest* adalah model sangat superior yang terbukti lebih kuat daripada model lainnya (*Logistic Regression, Naive Bayes, Support Vector Machine dan Neural Networks*) dan memiliki kinerja terbaik untuk deteksi serangan *botnet* [7]. Berlandaskan hal tersebut, tugas akhir ini memilih algoritma *random forest* dalam mendeteksi serangan *botnet* dan menggunakan Dataset yaitu dataset *Bot-IoT UNSW-2018*.

B. Topik dan Batasannya

Berdasarkan dari latar belakang diatas, maka rumusan masalah pada tugas akhir ini adalah

1. Bagaimana membangun sebuah model *Random Forest* untuk mendeteksi serangan *botnet* pada jaringan *Internet of Things(IoT)*?
2. Bagaimana tingkat performansi dari model *Random Forest* dalam mendeteksi serangan *botnet* pada jaringan *Internet of Things(IoT)*?

Adapun batasan masalah dari tugas akhir ini sebagai berikut

Tugas akhir ini dilakukan untuk deteksi serangan *Botnet* pada jaringan *IoT* dengan menggunakan dataset uji *Bot-IoT UNSW-2018*. Dimana hasil luaran berupa nilai *Accuracy*, *Precision*, *recall*, dan *F1-score* dari model algoritma *Random Forest.*

II. KAJIAN TEORI

Bab ini membahas tentang teori penunjang dan penelitian sebelumnya yang berhubungan dengan penerapan metode *Random Forest* dalam pendeteksian serangan *botnet.* Berikut merupakan *literature reviews* dari penelitian sebelumnya yang bersangkutan dengan deteksi *botnet* ditunjukan pada tabel 2.1.

A. Penelitian Terdahulu

Penelitian tentang pendeteksian botnet sebelumnya yang dilakukan oleh Kalaivani (2016) melakukan pendeteksian botnet pada flow jaringan dengan empat model klasifikasi, yaitu Support Vector Machine (SVM), Neural Network, Naïve Bayes dan Decision Tree. Metode SVM mendapat hasil tertinggi dibanding tiga metode lainnya, yaitu sebesar 99.8%. Pada Neural Network hasil sebesar 54.5%, Naïve Bayes sebesar 98.4% dan pada Decision Tree sebesar 95.7%.

Pada penelitian Didin Nizarul Fuadin(2018) melakukan penelitian dengan menggunakan *Naive Bayes Classifier* dalam pendeteksian menggunakan dataset *CTU-13*. Untuk meningkatkan akurasi, metode tersebut digabungkan dengan teknik *Synthetic Minority Over-sampling Technique (SMOTE)* dan penambahan metode seleksi fitur untuk pemilihan fitur dari *noise feature* menggunakan metode *Best First Search*. Hasil akurasi yang didapatkan sebesar 96,68%

Pada penelitian Nickilaos (2018) melakukan penelitian dengan menggunakan *ARM,Decision Tree, Naive Bayes, Artificial Neural Network (ANN)* dalam pendeteksian menggunakan dataset *UNSW-NB15*. dimana akurasi terbaik sebesar 97,4% dengan algoritma ANN

Pada penelitian Brendan Abraham(2018) menggunakan dataset yang memiliki format *CTU-13*. dan empat algoritma klasifikasi yaitu *Support Vector Machine (SVM),Neural Network,Naïve Bayes,dan Decision Tree*. Dari hasil penelitian yang telah dilakukan, klasifikasi dengan bantuan empat algoritma tersebut mendapatkan hasil yang cukup tinggi. Pada proses pengklasifikasian mendapatkan akurasi 99,8% untuk *Support Vector Machine (SVM)*, 54,5% untuk *Neural Network*, dan 98,4% untuk *Naive Bayes* dan 95,7 untuk *Decision tree*.

Pada penelitian Rizky Tri Wiyono(2018), *Algoritma Decision tree* digunakan dalam pengklasifikasian *botnet*. Dataset yang digunakan pada penelitian tersebut berasal dari *UNSW-NB15* Hasil akurasi yang didapatkan sebesar 92,23%.

Tabel 1: Penelitian terdahulu.

No	Judul	Metode	Tahun	Jenis Dataset	Jenis *Botnet*	Akurasi	Penulis
1	*Mining Based Detection of botnet traffic in Network Flow*	*Support Vector Machine (SVM) Neural Network Naïve Bayes Decision Tree*	2016	*CTU-13*	*Virut Mentineris rbot Sogou Murlo NSIS.ay*	99,8% 54,5% 98,4% 95,7%	P. Kalaivani
2	*Deteksi Botnet Menggunakan Naive Bayes Classifier dengan SMOTE dan Metode BFS*	*Naive Bayes Classifier dengan SMOTE dan Metode BFS*	2018	*CTU-13*	-	96,68%	Didin Nizarul Fuadin
3	*Towards Developing Network forensic mechanism for Botnet Activities in the IoT based on Machine Learning Techniques [14]*	*ARM,Decision Tree, Naive Bayes, Artificial Neural Network (ANN)*	2018	*UNSW-NB15*	-	92,75% 92,3% 95% 97,4%	Nickilaos Koronioti, Nour Moustafa, Elena Sitnikova, Jill Slay
4	*A Comparison of Machine Learning Approaches to Detect Botnet Traffic*	*Algoritma LR Naive Bayes SVM Random Forest dan Neural Networks*	2018	*CTU-13*	*Trickbot dan Bonitu*	95% 59% 91% 99% 95%	Brenda n Abraham
5	*Network Forensics* untuk Aktifitas *Botnet* pada *Internet of Things* dengan *Machine Learning*	*Decision tree*	2020	*UNSW-NB15*	*Zeus mirai kaiten*	93,23%	Rizky Tri Wiyono

B. Robot Network (Network)

Botnet terdiri dari "*Bot*" dan "*Network*", Bot Didapatkan dari kata "*robot*" yang dimana merupakan sebuah proses otomatis yang berinteraksi dengan layanan jaringan lain[12]. *Bot* dapat digunakan untuk tujuan yang baik atau jahat. Jika digunakan untuk tujuan jahat, mereka bekerja sebagai *Zombie* yang disusupkan pada jaringan komputer menggunakan perangkat lunak yang bisa dijalankan, dipantau dan diperintahkan bereaksi oleh pembuatnya (*Botmaster*). *Botnet* juga berisi kegiatan menyusupkan program-program tertentu kepada *server-server* komputer dimana program-program tersebut biasanya disisipkan sebagai *Worms, Trojan horse, atau Backdoors,* tujuannya yaitu untuk mengganggu ataupun merusak suatu jaringan atau sistem operasi komputer yang berpotensi melumpuhkan jaringan *internet* secara luas. Yang lebih menjadi perhatian utama saat ini *Botnet* dapat digerakan dan dikendalikan oleh *Botmaster* dari tempat manapun dan kapanpun ia mau, jadi seperti *Zombie* yang dipasang pada *server-server* yang ditanam melalui *Malware*. *Malware*, berasal dari kata malicious dan software yang artinya perangkat lunak yang diciptakan untuk menyusup atau merusak sistem komputer atau jejaring komputer tanpa izin pemilik perangkat (*informed consent*).

C. Bentuk Serangan Botnet

Adapun bentuk serangan Botnet adalah [8]:

1) ***Distributed Denial of Service (DDOS) attacks***, adalah serangan kepada jaringan yang mengakibatkan hilangnya layanan kepada *user*, biasanya hilangnya konektivitas jaringan dengan mengkonsumsi *bandwidth* dari korban jaringan atau *overloading* sumber daya sistem komputasi korban.

2) ***Spamming***, beberapa *Bot* memiliki kemampuan untuk membuka *socks proxy*, sebuah *proxy generic* untuk tcp/ip berbasis aplikasi jaringan pada mesin yang diincar setelah *socks proxy* diaktifkan, mesin ini dapat digunakan untuk kegiatan jahat seperti mengirim *spam* atau *email phising*.

3) ***Sniffing Traffic***, *Bot* juga bisa dapat menggunakan *packet sniffer* (penyadap paket yang melewati jaringan) untuk melihat data yang menarik di komputer, sebagian besar *sniffer* mengambil informasi yang sensitif seperti *username* dan *password*.

4) ***Penyebaran Malware baru***, *Botnet* juga digunakan untuk menyebarkan *Bot* baru dan *Malware*, hal ini menjadi semakin mudah sejak *Bot* menerapkan mekanisme untuk mengunduh dan menjalankan file melalui http dan ftp, beberapa *Bot* dapat berpura-pura berperan sebagai *server* http atau ftp sebagai *Malware*.

5) ***Installing Advertisement Add-ons & Browser Helper Objects***, dengan membuat situs palsu, dengan membuat beberapa iklan dan mendaftar pay per klik di perusahaan penyedia iklan, *botmaster* akan mendapatkan pemasukan atau pendapatan.yaitu dengan bantuan dari sebuah *Botnet*, proses klik ini dapat dilakukan secara otomatis (klik penipuan) sehingga beberapa ribu *Bot* mengklik iklan tersebut.

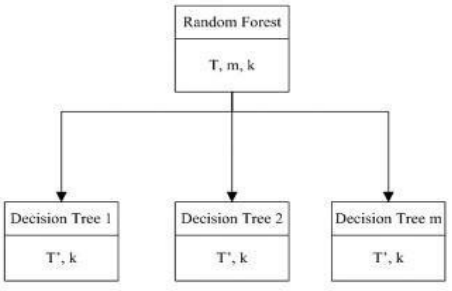
D. Machine Learning

Machine learning merujuk pada sebuah metode yang membuat komputer memiliki kemampuan dalam mempelajari dan melakukan sebuah pekerjaan secara otomatis. Proses *machine learning* dilakukan melalui algoritma tertentu, sehingga pekerjaan yang diperintahkan kepada komputer dapat dilakukan secara otomatis [9]. Mac*hine learning* dilakukan melalui 2 fase, yaitu fase *training* dan fase testing. Fase *training* adalah proses pemodelan dari algoritma yang digunakan akan dipelajari oleh sistem melalui *training* data, sedangkan fase *testing* adalah proses pemodelan yang telah dipelajari sistem melalui fase *training* akan digunakan untuk menghasilkan sebuah keputusan tertentu, dengan menggunakan *testing* data. *Machine learning* dapat dilakukan dengan dua cara, yaitu *supervised learning* dan *unsupervised learning. Unsupervised learning* adalah pemrosesan sampel data dilakukan tanpa mewajibkan hasil akhir memiliki bentuk yang sesuai dengan bentuk tertentu, dengan menggunakan beberapa sampel data sekaligus. Penerapan *unsupervised learning* dapat ditemukan pada proses visualisasi, atau eksplorasi data. *Supervised learning* adalah pemrosesan sampel data x akan diproses sedemikian rupa, sehingga menghasilkan *output* yang sesuai dengan hasil akhir y. *Supervised learning* dapat diterapkan pada proses deteksi [9].

E. Random forest

Random forest merupakan metode *bagging* yaitu metode yang membangkitkan sejumlah *tree* dari data sampel dimana pembuatan satu *tree* pada saat *training* tidak bergantung pada *tree* sebelumnya kemudian keputusan diambil berdasarkan *voting* terbanyak [10]. Dua konsep yang menjadi dasar dari *random forest* adalah membangun *ensemble* dari *tree* via *bagging* dengan *replacement* dan penyeleksian fitur secara acak untuk tiap *tree* yang dibangun. Pertama, setiap *sample* yang diambil dari dataset untuk *training tree* bisa dipakai lagi untuk *training tree* yang lain. Kedua, fitur yang digunakan pada saat *training* untuk tiap *tree* merupakan subset dari fitur yang dimiliki oleh dataset [10]).

Deteksi berbasis *ensemble* akan mempunyai performa yang maksimal jika antar *basic learner* mempunyai korelasi yang rendah. Sebuah *ensemble* harus membangun *basic learner* yang lemah, karena *learner* yang kuat kemungkinan besar akan mempunyai korelasi yang tinggi dan biasanya juga menyebabkan *overfit*, sedangkan *random forest* meminimalkan korelasi serta mempertahankan kekuatan deteksi dengan cara melakukan pengacakan pada proses *training*, yaitu dengan memilih sejumlah fitur secara acak dari semua fitur yang ada pada setiap melakukan *training tree*, kemudian menggunakannya menggunakan fitur-fitur yang terpilih untuk mendapatkan percabangan *tree* yang optimal. Berbeda dengan proses *training tree* pada *decision tree* biasa, proses *training tree* yang menjadi bagian dari *random forest* tidak menggunakan proses *pruning* akan tetapi percabangan akan terus dilakukan sampai ukuran batas *leaf* tercapai [10]. II-3 *Random forest* mempunyai dua parameter utama, yaitu: m jumlah *tree* yang akan dipakai dan k yaitu maksimal banyaknya fitur yang dipertimbangkan ketika proses percabangan. Semakin banyak nilai m maka semakin bagus hasil deteksi, sedangkan untuk nilai k direkomendasikan sebesar akar kuadrat atau logaritma dari jumlah total fitur [10].



Gambar 1. Random forest

Keterangan:
T : dataset
m : tree
k : banyaknya fitur yang dipakai

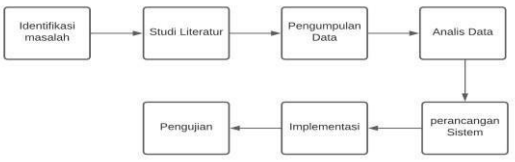
Gambar 1 menunjukan proses *training* untuk random forest menggunakan dataset T dengan sejumlah m *tree* sebagai *basic leaner* dan k fitur yang dipilih secara acak dari total fitur yang ada untuk percabangan pada setiap *tree*. Proses training pada setiap *tree*

menggunakan dataset T' yang merupakan hasil dari *bootstrap* dari database yang dijadikan parameter untuk *random forest*. *Bootstrap* merupakan proses memilih sampel dari dataset yang akan digunakan proses *training tree*. Metode *ensemble*, *bootstrap* merupakan proses *sampling* dengan *replacement*, sehingga sampel yang diambil untuk proses *training tree* yang satu masih bisa dipakai lagi untuk proses *training tree* yang lainnya [10].

F. SMOTE (*Syn Synthetic Minority Oversampling Technique*

Ketidakseimbangan data terjadi jika jumlah objek suatu kelas data lebih banyak dibandingkan dengan kelas lain. Kelas data yang objeknya lebih banyak disebut kelas mayor sedangkan lainnya disebut kelas minor. Pengaruh penggunaan data tidak seimbang untuk membuat model sangat besar pada hasil model yang diperoleh. Pengolahan algoritma yang tidak menghiraukan ketidakseimbangan data akan cenderung diliputi oleh kelas mayor dan mengacuhkan kelas minor [16]. Metode *SMOTE* diusulkan oleh [16] sebagai salah satu solusi dalam menangani data tidak seimbang dengan prinsip yang berbeda dengan Metode oversampling yang telah diusulkan sebelumnya. Bila Metode *oversampling* berprinsip memperbanyak pengamatan secara acak, Metode *SMOTE* menambah jumlah data kelas minor agar setara dengan kelas mayor dengan cara membangkitkan data buatan. Data buatan atau sintesis tersebut dibuat berdasarkan k-tetangga terdekat *(k-nearest neighbor)*. Jumlah k-tetangga terdekat ditentukan dengan mempertimbangkan kemudahan dalam melaksanakannya. Pembangkitan data buatan yang berskala numerik berbeda dengan kategorik. Data numerik diukur jarak kedekatannya dengan jarak *Euclidean* sedangkan data kategorik lebih sederhana yaitu dengan nilai modus. Perhitungan jarak antar contoh kelas minor yang peubahnya berskala kategorik dilakukan dengan rumus *Value Difference Metric (VDM).*

III. METODE



Gambar 2. Metodologi Penelitian

A. Identifikasi Masalah

Pada tahap ini merupakan langkah pertama yang dilakukan untuk menentukan topik dari tugas akhir ini dan mengidentifikasi dari permasalahan yang ada, identifikasi ini bermaksud untuk penegasan batasan-batasan dari permasalahan agar tugas akhir ini sesuai dengan tujuan.

B. Studi Literatur

Pada tahap studi literatur, penulis mengumpulkan dan mempelajari berbagai literatur baik dari jurnal, buku, maupun artikel penunjang yang berhubungan dengan *Botnet*, *Internet of Things (IoT)*, *Imbalance data*, *Random forest algorithm*, *Confusion Matrix*, *ROC Curve*, *python programming language* dan lain-lain. Untuk mendapatkan pemahaman tentang apa yang akan dilakukan berdasarka**n s**tudi literatur terdahulu yang dapat menunjang dalam pengerjaan tugas akhir ini.

C. Analisis Data

Pada tahap ini melakukan analisis data dari dataset *UNSW 2018 Bot IoT*, dengan melihat berapa jumlah data yang ada, jumlah serangan botnet, jumlah bukan serangan dan menentukan pemilihan *algoritma machine learning* terbaik dalam proses deteksi serangan botnet berdasarkan studi literatur yang ada.

D. Perancangan Sistem

Pada tahapan ini akan dijelaskan langkah-langkah metodologi penelitian secara sistematis dan terarah yang akan dijadikan acuan dalam kerangka penelitian yang membahas tentang deteksi *Botnet* menggunakan *Random Forest* alur perancangan sistem yang akan dilakukan ditunjukan pada gambar 3 sebagai berikut:

Gambar 3. Flowchart rancangan system

E. Pengumpulan data

Dataset *Bot-IoT UNSW-2018* dibuat dengan merancang lingkungan jaringan yang realistis di *Cyber Range Lab UNSW Canberra*. Lingkungan jaringan menggabungkan kombinasi lalu lintas normal dan *botnet*. File sumber dataset disediakan dalam format yang berbeda, termasuk file pcap asli, file argus yang dihasilkan, dan file csv. File dipisahkan, berdasarkan kategori *attack*, *category*, dan *subcategory* untuk lebih membantu dalam proses pelabelan. File pcap yang diambil berukuran 69,3 GB, dengan lebih dari 72.000.000 catatan. Arus lalu lintas yang diekstrak, dalam format csv berukuran 16,7 GB. Dataset ini mencakup serangan *category* botnet seperti serangan *DDoS, DoS, OS dan Service Scan, Keylogging and Data exfiltration* dan serangan *subcategory UDP, TCP, Service Scan, OS Fingerprint, HTTP, Keylogging, Data Exfiltration.* Serangan *DDOS* pada dataset ini seperti U*DP flooding, SYN flooding, Remote Controlled Attack, dan Ping Of Deat*h. Pada tugas akhir ini menggunakan

sekitar 5% dari data asli yang telah disediakan dalam bentuk csv dan terbagi menjadi 2 yaitu dataset pelatihan 352 MB, sedangkan Dataset Pengujian sebesar 88 MB. Tidak ada yang hilang*(null data)* nilai dalam dataset pelatihan maupun pengujian yang telah disediakan. Berikut ini 19 atribut dari dataset yang digunakan:

Tabel 2: Deskripsi fitur dataset Bot-Iot UNSW 2018

No	Fitur	Deskripsi
1	*pkSeqID*	*Row Identifier*
2	*Proto*	*Textual representation of tran protocols present in networ*
3	*Saddr*	*Source IP address*
4	*Sport*	*Source port number*
5	*Daddr*	*Destination IP address*
6	*Dport*	*Destination port number*
7	*Seq*	*sequence number*
8	*Stddev*	*Standard deviation of aggregated records*
9	*N_IN_Conn_P_SrcIP*	*Number of inbound connections per source IP*
10	*Min*	*Minimum duration of aggregate records*
11	*state_number*	*Numerical representation of feature state*
12	*Mean*	*Average duration of aggregated records*
13	*N_IN_Conn_P_DstIP*	*Number of inbound connections per destination IP*
14	*Drate*	*Destination-to-source packets per second*
15	*Srate*	*Source-to destination packets per second*
16	*Max*	*Maximum duration of aggregated records*
17	*Attack*	*Class label: 0 for Normal traffic, 1 for Attack*
18	*Category*	*Traffic category*
19	*Subcategory*	*Traffic subcategory*

F. *Preprocessing data*

Pada tahap ini, data akan dilakukan pemrosesan data, dimana melakukan pengecekan jumlah data kosong, melakukan penyamarataan data dengan fitur scaling dan melakukan seleksi fitur dari 19 atribut seperti *state number, drate, srate, proto, seq, stddev, min, N_IN_Conn_P_SrcIP, N_IN_Conn_P_DstIP, mean, max, attack, category. dan subcategory* dengan metode *extratreesclassifier*.

3.6.1 Normalisasi Data

Tahap ini menghilangkan data yang kosong atau data dengan nilai atribut yang tidak memenuhi syarat dan tidak dapat digunakan digunakan dalam proses atau tahap berikutnya. Tahap ini bersamaan dengan tahap ekstraksi fitur, dimana ketika membaca, transformasi dan pengecekan atribut. Jika atribut itu kosong maka data tidak dapat dilakukan deteksi, sehingga dihapus atau tidak masuk dalam data final untuk proses berikutnya

3.6.2 Seleksi Fitur dengan *ExtraTreeClassifier*

Seleksi fitur *ExtraTreeClassifier* dengan digunakan untuk menghilangkan fitur yang tidak relevan dan akan menyebabkan tingkat akurasi rendah(menghasilkan model yang buruk). Dimana metode seleksi fitur ini akan mencari atribut yang paling berpengaruh terhadap fitur uji dari *attack,category* dan *subccategory*. Berdasarkan percobaan yang telah dilakukan, terdapat 5 atribut yang tidak berpengaruh terhadap fitur uji, seperti : *PkSeqId, Saddr, Daddr, Dport, Sport,* dan *Drate.* Dan *10* atribut yang paling berpengaruh terhadap fitur uji seperti*: proto, state number, srate, seq, stddev, min, N_IN_Conn_P_SrcIP,N_IN_Conn_P_DstIP, mean dan max.*

3.6.3 Oversampling Data Minor dengan *SMOTE*

Pada tahap ini dilakukan oversampling dengan metode *smote* pada fitur uji, dimana jumlah data pada kelas minoritas ditambahkan sebanyak kelas mayoritas. Teknik oversampling pada *SMOTE* lebih dipilih dibanding undersampling karena perbedaan jumlah data yang terlalu jauh, dan akan menghasilkan hasil yang efektif dalam pembobotan *confusion matriks.*

G. *Partitioning data*

Pada tahap ini dataset akan dipecah menjadi dua bagian yaitu data *train*, data *test*. dimana untuk pembagian data *train* 80%, data *test* 20% dan pembagian dataset *train* 70%, data *test* 30%.

H. Evaluasi Model

Tugas akhir ini melakukan evaluasi menggunakan metode *confusion matrix. Confusion matrix* digunakan untuk menganalisis seberapa baik *classifier* mengenali *tuple* dari kelas yang berbeda [11]. Nilai dari *True-Positive* dan *True-Negative* memberikan informasi ketika *classifier* melakukan klasifikasi data bernilai benar, sedangkan *False-Positive* dan *False-Negative* memberikan informasi ketika *classifier* salah dalam melakukan klasifikasi data [11]. Confusion matrix ditunjukkan pada Tabel 3.1 sebagai berikut:

Prediksi / Aktual	Normal	Botnet
Normal	TP	FN
Botnet	FP	TN

Gambar 4**:** Confusion Matrix.

True-Positive (TP) yaitu jumlah prediksi benar dari *flow attack*. *False-Positive (FP)* yaitu jumlah kesalahan prediksi *flow normal* pada f*low attack*. *False- Negative (FN)* yaitu jumlah kesalahan prediksi *flow attack* pada *flow normal*. *True-Negative (TN)* yaitu jumlah prediksi benar pada *flow normal*.

Hasil *confusion matrix* digunakan untuk mengukur *accuracy*, *precision*, *recall* dan *F1-score* untuk menganalisis kinerja dari algoritma dalam melakukan klasifikasi untuk mendeteksi *botnet* dengan persamaan (1) (2) (3) (4):

$$\text{Akurasi} = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

$$\text{Recall} = \frac{TP}{TP+FN} \quad (2)$$

$$\text{Precision} = \frac{TP}{TP+FP} \quad (3)$$

$$\text{F1-score} = 2 \text{ x } Precision \ x \ Recall \ Precision + Recall \quad (4)$$

Akurasi yaitu kedekatan antara nilai prediksi dan nilai aktual. *Precision* yaitu tingkat ketepatan antara informasi yang diminta oleh pengguna dengan jawaban yang diberikan oleh sistem. *Recall* adalah tingkat keberhasilan sistem dalam menemukan kembali sebuah informasi. *F1-score* merupakan salah satu perhitungan evaluasi yang mengkombinasikan *recall* dan *precision*.

Keterangan:

- True Positive (TP): Hasil diprediksi dan data positif benar / terdeteksi serangan dan trafik botnet.
- True Negative (TN): Hasil diprediksi dan data negatif benar / terdeteksi serangan dan trafik normal.
- False Positive (FP): Hasil diprediksi positif namun data negatif / tidak terdeteksi serangan dan trafik botnet.
- False Negative (FN): Hasil diprediksi negatif namun data positif / tidak terdeteksi serangan dan trafik normal.

I. Skenario Pengujian

Pada skenario pengujian tugas akhir ini, dilakukan skenario pengujian berdasarkan pembagian jumlah dataset. Dimana menggunakan 70% data train, 30% data test dan 80% data train, 20% data test. kemudian setiap dataset diuji dengan *n_estimators* = 20, 40, 60, 80, 100, 120, 140, 160, 180, dan 200 selanjutnya di tes terhadap 3 fitur uji seperti fitur *attack*, *category* dan *subcategory*.

IV. HASIL DAN PEMBAHASAN

A. Hasil Pengujian *confusion matrix* dan kurva ROC

Dalam tugas akhir ini, Pengujian dilakukan dengan proses pembagian data, pengujian kinerja matrix berdasarkan jumlah pohon keputusan (n_estimator) yang dibuat. Dimana penerapan algoritma *random forest* untuk menghitung tingkat performansi dalam mendeteksi serangan botnet, Dimana menggunakan dataset *Bot-IoT UNSW 2018* dan dengan 10 fitur berpengaruh seperti : *proto*, *state number*, *srate*, *seq*, *stddev*, *min*, *N_IN_Conn_P_SrcIP*, *N_IN_Conn_P_DstIP*, *mean*, *max*.

Dalam pengujian ini, model random forest menggunakan pembagian 80% data *train* dan 20% data *test*, dan dengan berbagai pohon keputusan(*n_estimators)*. Berdasarkan pengujian yang telah dilakukan, didapat nilai *confusion matrix optimum*, *yaitu accuracy 99.27%*, *precision* 99.99%, *recall* 98.55%, *f1 score* 99.27% dan *n_estimators* = 80.

Tabel 3: Hasil pengujian confusion matriks pada fitur attack

Data		n_estimators	Precision	Recall	F1 score	Accuracy
train	test					
80 %	20 %	20	99.99 %	94.93%	97.40 %	97.46 %
		40	99.99 %	95.44%	97.66 %	97.72 %
		60	99.99 %	97.82%	98.89 %	98.91 %
		80	99.99 %	98.55%	99.27 %	99.27 %
		100	99.99 %	98.52%	99.25 %	99.26 %
		120	99.99 %	98.53%	99.26 %	99.26 %
		140	99.99 %	98.53%	99.26 %	99.26 %
		160	99.99 %	98.53%	99.26 %	99.26 %
		180	99.99 %	98.53%	99.26 %	99.26 %
		200	99.99 %	96.99%	98.47 %	98.49 %
Rata- rata			99.99 %	97.64%	98.8 %	98.82 %

Dalam pengujian ini, model random forest menggunakan pembagian 70% data *train* dan 30% data *test*, dan dengan berbagai pohon keputusan(*n_estimators)*. Berdasarkan pengujian yang telah dilakukan, didapat nilai *confusion matrix optimal*, *yaitu accuracy* 99.26%, *precision* 99.99%, *Recall* 98.53%, *f1 score* 99.26% dan *n_estimators* = 100.

Tabel 4: Hasil pengujian confusion matriks pada fitur attack

Data		n_estimators	Precision	Recall	F1 score	Accuracy
train	test					

70 %	30 %	20	99.91 %	94.48 %	97.12 %	97.2 %
		40	99.91 %	94.99 %	97.39 %	97.45 %
		60	99.91 %	98.54 %	99.22 %	99.22 %
		80	99.91 %	98.54 %	99.22 %	99.22 %
		100	99.99 %	98.53 %	99.26 %	99.26 %
		120	99.91 %	98.54 %	99.22 %	99.22 %
		140	99.91 %	98.54 %	99.22 %	99.22 %
		160	99.91 %	98.54 %	99.22 %	99.22 %
		180	99.91 %	98.54 %	99.22 %	99.22 %
		200	99.91 %	98.54 %	99.22 %	99.22 %
Rata- rata			99.92 %	97.78 %	98.83 %	98.95 %

Dalam pengujian ini, model random forest menggunakan pembagian 80% data *train* dan 20% data *test*, dan dengan berbagai pohon keputusan(*n_estimators)*. Berdasarkan pengujian yang telah dilakukan, didapat nilai *confusion matrix optimal*, *yaitu accuracy* 99.43%, *precision* 99.43%, *recall* 99.43%, *f1 score* 99.43% dan *n_estimators* = 120.

Tabel 5: Hasil pengujian fitur category

Data		n_estimators	Precision	Recall	F1 score	Accuracy
train	test					
80 %	20 %	20	98.96 %	98.95 %	98.95 %	98.95 %
		40	99.36 %	99.36 %	99.36 %	99.36 %
		60	98.99 %	98.98 %	98.98 %	98.98 %
		80	99%	99%	99%	99%
		100	99%	99%	99%	99%
		120	99.43 %	99.43 %	99.43 %	99.43 %
		140	99.39 %	99.39 %	99.39 %	99.39 %
		160	99.41 %	99.41 %	99.41 %	99.41 %
		180	99.35 %	99.35 %	99.35 %	99.35 %
		200	99.37 %	99.37 %	99.37 %	99.37 %
Rata- rata			99.22 %	99.22 %	99.22 %	99.22 %

Dalam pengujian ini, model random forest menggunakan pembagian 70% data *train* dan 30% data *test*, dan dengan berbagai pohon keputusan(*n_estimators)*. Berdasarkan pengujian yang telah dilakukan, didapat nilai *confusion matrix optimal*, *yaitu accuracy* 99.15%, *precision* 99.15%, *recall* 99.15%, *f1 score* 99.15% dan *n_estimators* = 180.

Tabel 6: Hasil pengujian confusion matriks fitur category

Data		n_estimators	Precision	Recall	F1 score	Accuracy
train	test					
70 %	30 %	20	97.74 %	97.62 %	97.6 %	97.62 %
		40	97.71 %	97.6 %	97.5 %	97.6 %
		60	97.61 %	97.49 %	97.47 %	97.49 %
		80	97.67 %	97.57 %	97.67 %	97.57 %
		100	97.67 %	97.55 %	97.54 %	97.55 %
		120	97.67 %	97.55 %	97.54 %	97.55 %
		140	97.67 %	97.55 %	97.54 %	97.55 %
		160	98.97 %	98.97 %	98.95 %	98.95 %
		180	99.15 %	99.15 %	99.15 %	99.15 %
		200	98.06 %	97.97 %	97.97 %	97.97 %
Rata- rata			97.99 %	97.9 %	97.89 %	97.9 %

Dalam pengujian ini, model random forest menggunakan pembagian 80% data *train* dan 20% data *test*, dan dengan berbagai pohon keputusan(*n_estimators)*. Berdasarkan pengujian yang telah dilakukan, didapat nilai *confusion matrix optimal*, *yaitu accuracy* 98.864%, *precision* 98.52%, *recall* 98.86%, *f1 score* 98.53% dan *n_estimators* = 120.

Tabel 7: Hasil pengujian confusion matriks fitur subcategory

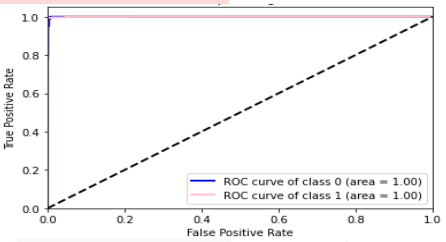
Data		n_estimators	Precision	Recall	F1 score	Accuracy
train	test					
80 %	20 %	20	98.51 %	98.84 %	98.5 %	98.84 %
		40	98.51 %	98.85 %	98.51 %	98.85 %
		60	98.51 %	98.85 %	98.51 %	98.85 %
		80	98.52 %	98.86 %	98.52 %	98.86 %
		100	98.52 %	98.86 %	98.53 %	98.86 %
		120	98.52 %	98.86 %	98.53 %	98.86 4%
		140	98.52 %	98.86 %	98.53 %	98.86 %
		160	98.52 %	98.86 %	98.52 %	98.86 %
		180	98.52 %	98.86 %	98.52 %	98.86 %
		200	98.52 %	98.86 %	98.53 %	98.86 %
Rata- rata			98.52 %	98.86 %	98.52 %	98.86 %

Dalam pengujian ini, model random forest menggunakan pembagian 80% data *train* dan 20% data *test*, dan dengan berbagai pohon keputusan(*n_estimators)*. Berdasarkan pengujian yang telah dilakukan, didapat nilai *confusion matrix optimal*, *yaitu accuracy* 98.85*%*, *precision* 98.51%, *Recal*l 98.84%, *recall* 98.51% dan *n_estimators* = 140.

Tabel 9: Hasil pengujian confusion matriks fitur subcategory

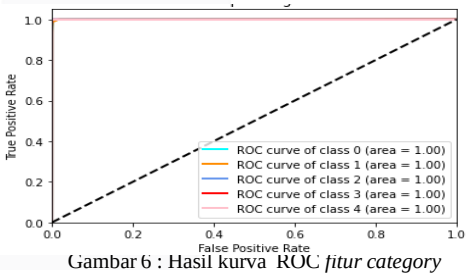
Data		n_estimators	Precision	Recall	F1 score	Accuracy
train	test					
70 %	30 %	20	98.35 %	98.75 %	98.42 %	98.75 %
		40	98.5 %	98.83 %	98.49 %	98.83 %
		60	98.51 %	98.84 %	98.5 %	98.84 %
		80	98.51 %	98.84 %	98.5 %	98.84 %
		100	98.51 %	98.84 %	98.5 %	98.84 %
		120	98.51 %	98.84 %	98.51 %	98.84 %
		140	98.51 %	98.84 %	98.51 %	98.85 %
		160	98.51 %	98.84 %	98.51 %	98.84 %
		180	98.51 %	98.84 %	98.51 %	98.84 %
		200	98.51 %	98.84 %	98.51 %	98.84 %
Rata- rata			98.49 %	98.83 %	98.5 %	98.83 %

Kurva ROC pada fitur *attack* menunjukan kinerja yang baik dari model yang telah dibuat. Dimana nilai kurva yang berada di titik 1.0 atau diatas garis diagonal.



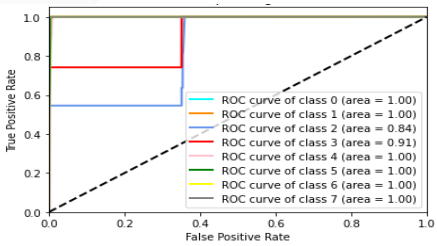
Gambar 5. Hasil kurva ROC *fitur attack*

Kurva ROC pada fitur *category* menunjukan kinerja yang baik dari model yang telah dibuat. Dimana nilai kurva yang berada di titik 1.0 atau diatas garis diagonal.



Gambar 6 : Hasil kurva ROC *fitur category*

Kurva ROC pada fitur *subcategory* menunjukan kinerja yang baik dari model yang telah dibuat. Dimana nilai kurva yang berada diatas garis diagonal.



Gambar 7 : Hasil kurva ROC *fitur subcategory*

B. Analisi Hasil Pengujian

Berdasarkan pengujian yang telah dilakukan**,** nilai confusion matrix terbaik untuk setiap fitur didapat dengan pembagian dataset, 80% data train dan 20% data test. Dimana pada fitur attack jumlah pohon optimal adalah 80 dengan nilai

accuracy 99.27%, *precision* 99.99%, *recall* 98.55%, *dan f1 score* 99.27%. Pada fitur category jumlah pohon optimal adalah 120 dengan nilai *accuracy* 99.43%, *precision* 99.43%, *recall* 99.43%, dan *f1 score* 99.43%. Pada fitur subcategory jumlah pohon optimal adalah 120 dengan *accuracy* 98.864%, *precision* 98.52%, *recall* 98.86%, dan *f1 score* 98.53%. serta nilai kurva ROC setiap fitur yang lebih dari 0.5(jauh dari garis diagonal) menghasilkan performa model yang baik[15]. Berdasarkan hasil pengujian tersebut, dapat disimpulkan bahwa jumlah pohon yang di-*generate* adalah 200. Hal ini karena setelah 200 grafik kinerja landa. waktu pengambilan keputusan yang dibutuhkan juga tidak terlalu panjang, jika pohon yang di*generate* ditambah lagi, kemungkinan beban kerja yang dihasilkan juga akan bertambah dari sisi waktu dan sumber daya. Selain itu model *random forest* dengan penyeimbangan data *SMOTE* memaksimalkan kinerja algoritma *random forest* dalam pengukuran tingkat akurasi dan menambah nilai dari *False Positif*. Sehingga model yang di bangun tidak terjadi *overfit*.

V. KESIMPULAN

Berdasarkan pengujian dapat disimpulkan bahwa, kinerja *Random forest* dalam melakukan deteksi serangan *botnet* dengan cukup baik. Dimana hasil pengujian, dengan pembagian data dan *n_estimators* mendapatkan nilai *accuracy, precision, recall, f1 score* setiap fitur yang lebih dari 97% serta nilai kurva *ROC* setiap fitur yang lebih dari 0.5 (jauh dari garis diagonal) menghasilkan performa model yang baik. selain itu, hasil *running* algoritma *random forest* terbilang cukup cepat dengan jumlah dataset yang lebih dari 6 juta dan hemat dalam sumber daya. Untuk Saran dalam percobaan selanjutnya adalah melakukan teknik *cross validation* pada model, dan menggunakan metode *oversampling* lainnya seperti *SMOTE-NC*, *ADA-SYN*. Ataupun penggabungan teknik *oversampling* dan *undersampling* guna memaksimalkan hasil akurasi yang didapat, dan dapat juga menggunakan variasi algoritma dari *deep learning.*

REFERENSI