TESIS – TE142599

DETEKSI BOTNET MENGGUNAKAN NAÏVE BAYES *CLASSIFIER* DENGAN SMOTE DAN METODE BFS

DIDIN NIZARUL FUADIN
2215206711

DOSEN PEMBIMBING
Dr. Surya Sumpeno, ST., M.Sc.
Dr. Supeno Mardi Susiki Nugroho, ST., MT.

PROGRAM MAGISTER
BIDANG KEAHLIAN TEKNIK TELEMATIKA - CIO
DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI ELEKTRO
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2017

TESIS – TE142599

DETEKSI BOTNET MENGGUNAKAN NAÏVE BAYES *CLASSIFIER* DENGAN SMOTE DAN METODE BFS

DIDIN NIZARUL FUADIN
2215206711

DOSEN PEMBIMBING
Dr. Surya Sumpeno, ST., M.Sc.
Dr. Supeno Mardi Susiki Nugroho, ST., MT.

PROGRAM MAGISTER
BIDANG KEAHLIAN TEKNIK TELEMATIKA - CIO
DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI ELEKTRO
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2017

LEMBAR PENGESAHAN

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Teknik (M.T)
di
Institut Teknologi Sepuluh Nopember

oleh:

Didin Nizarul Fuadin
NRP. 2215206711

Tanggal Ujian : 5 Juni 2017
Periode Wisuda : September 2017

Disetujui oleh:

1. Dr. Surya Sumpeno, ST., M.Sc. (Pembimbing I)
   NIP. 19690613 199702 1 001
2. Dr. Supeno Mardi S.N, ST., MT. (Pembimbing II)
   NIP. 19700313 199512 1 001
3. Prof. Dr. Ir. Yoyon Kusnendar S, M.Sc. (Penguji)
   NIP: 19540925 197803 1 001
4. Dr. Ir. Endroyono, DEA (Penguji)
   NIP: 19650404 199102 1 001
5. Dr. Eko Mulyanto Yuniarto, ST., MT. (Penguji)
   NIP: 19680601 199512 1 009
6. Dr. Istas Pratomo, ST., MT. (Penguji)
   NIP: 19790325 200312 1 001

Dekan Fakultas Teknologi Elektro

Dr. Tri Arief Sardjono, S.T., M.T.
NIP. 197002121995121001

Halaman ini sengaja dikosongkan

PERNYATAAN KEASLIAN TESIS

Dengan ini saya menyatakan bahwa isi keseluruhan Tesis saya dengan judul "**DETEKSI BOTNET MENGGUNAKAN NAÏVE BAYES *CLASSIFIER* DENGAN SMOTE DAN METODE BFS**" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, Mei 2017

Didin Nizarul Fuadin

NRP. 2215207711

Halaman ini sengaja dikosongkan

DETEKSI BOTNET MENGGUNAKAN NAÏVE BAYES *CLASSIFIER* DENGAN SMOTE DAN METODE BFS

Nama Mahasiswa : Didin Nizarul Fuadin
NRP : 2215206711
Pembimbing : 1. Dr. Surya Sumpeno, ST., M.Sc.
2. Dr. Supeno Mardi, S.N., ST., MT.

ABSTRAK

Tingginya pertumbuhan kejahatan siber merupakan suatu tantangan untuk analis keamanan informasi. Terutama identifikasi anomali pada jaringan, yang mana merupakan langkah awal terpenting untuk mengetahui kemungkinan adanya potensi ancaman. Hal ini menjadi sulit dikarenakan beberapa perilaku dari fitur serangan sangatlah mirip dengan aktifitas jaringan normal. Seorang peneliti ataupun Analis dalam bidang keamanan informasi, untuk mengkaji lebih dalam pengetahuan dan wawasan tentang serangan dapat menggunakan data log, *network flow* atau paket data jaringan. Penelitian dengan klasifikasi intrusi yang sudah ada dipandang masih perlu dilakukan optimasi sebagai bentuk perkembangan penelitian dan proaktif dalam identifikasi dengan kemunculan intrusi baru. Pada penelitian ini fokus pada improvisasi deteksi Botnet, penelitian ini merupakan bagian kecil dari sistem besar *Intrusion Prevention System*, Metode yang diusulkan adalah Naïve Bayes yang digunakan untuk klasifikasi data *network flow* yang terdiri dari data normal dan Botnet. Untuk kendala *imbalance data* digunakanlah teknik *Synthetic Minority Over-sampling Technique* (SMOTE) untuk menanganani ketidakseimbangan kelas dan seleksi fitur untuk pemilihan fitur dari *noise feature* menggunakan metode *Best First Search*. Berdasarkan penelitian yang telah dilakukan, diketahui bahwa tingkat akurasi hasil deteksi menggunakan Naïve Bayes classifier dengan SMOTE dan metode BFS meningkat signifikan dibandingkan dengan hanya menggunakan Naïve Bayes yaitu dari tingkat akurasi **96,68%** meningkat menjadi **99,19%** atau peningkatan **2,51%** akurasi tertinggi dengan eksperimen SMOTE 40% *oversampling data*.

Kata kunci: *Securiy*, Botnet, SMOTE, *Best First Searh*, Naïve Bayes *classifier*.

Halaman ini sengaja dikosongkan

BOTNET DETECTION USING NAÏVE BAYES CLASSIFIER WITH SMOTE AND BFS METHOD

By : Didin Nizarul Fuadin
Student Identity Number : 2215206711
Supervisor(s) : 1. Dr. Surya Sumpeno, ST., M.Sc.
2. Dr. Supeno Mardi, S.N., ST., MT.

ABSTRACT

The high growth of cybercrime is a challenge for information security analysts, especially the identification of anomalies on the network, which is the most important first step in knowing potential threats. This is becomes difficult because some of the behavior of attack acitvity is very similar with normal activity. A researches or analyst in the domain of information security can assess knowledge and insight about the attack can use log data, network flow or network packets data. Research with an existing intrusion detection is considered to still be optimized as actual of research and proactive development in the identification with the appear of new intrusions. In this study focus on improving Botnet detection, this propose method is a small part of the large system of Intrusion Prevention System, The proposed method is using Naïve Bayes for network flow classification, where the traffic is consisting of normal or Botnet. For the data imbalance we used the Synthetic Minority Over-sampling Technique (SMOTE) technique to handle it and feature selection for reducing of noise feature using Best First Search (BFS) method. Empirical result show that Naïve Bayes classifier with SMOTE and BFS Method gives better performance to handle imbalance data and noise features. It can be seen that the accuracy of detection results using Naïve Bayes classifier with SMOTE and BFS method increases significantly compared with Naïve Bayes only ie from 96.68% accuracy rate increases to 99.19% or increased of 2.51% accuracy with SMOTE scenario 40 % Oversampling data.

Key words: *Securiy*, Botnet, SMOTE, *Best First Searh*, Naïve Bayes *classifier*

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Alhamdulillahirobbil'alamin, puji syukur atas segala limpahan nikmat dan karunia Allah SWT, Tuhan yang Maha Kuasa. Hanya dengan petunjuk, rahmat dan ridho-Nya, sehingga penulis dapat menyelesaikan tesis dengan judul "**DETEKSI BOTNET MENGGUNAKAN NAÏVE BAYES *CLASSIFIER* DENGAN SMOTE DAN METODE BFS**".

Ucapan terima kasih yang sebesar-besarnya dan penghargaan yang setinggi-tingginya saya sampaikan kepada yang terhormat Dr. Surya Sumpeno, ST., M.Sc. selaku pembimbing pertama dan Dr. Supeno Mardi S. N, ST., MT. selaku pembimbing kedua, yang dengan penuh perhatian, dan kesabaran selalu meluangkan waktu, memberikan pengarahan dan motivasi serta semangat dalam penulisan tesis ini.

Penulis dapat menyelesaikan tesis ini, juga tidak terlepas dari bantuan dan kerjasama dari berbagai pihak, maka perkenankan saya dengan sepenuh hati menyampaikan terima kasih yang tak terhingga kepada:

1. Prof. Ir. Joni Hermana, M.Sc.Es, Ph.D., selaku Rektor Institut Teknologi Sepuluh Nopember Surabaya, yang telah memberikan kesempatan dan fasilitas kepada saya untuk mengikuti dan menyelesaikan pendidikan pada Program Magister, Jurusan Teknik Elektro, Bidang Keahlian Telematika / *Chief Information Officer* (CIO), Institut Teknologi Sepuluh Nopember Surabaya.
2. Kementerian Komunikasi dan Informatika Republik Indonesia yang telah memberikan kesempatan mendapatkan beasiswa Program Magister Jurusan Teknik Elektro, Bidang Keahlian Telematika / *Chief Information Officer* (CIO) pada Institut Teknologi Sepuluh Nopember Surabaya.
3. Dr. Surya Sumpeno ST., M.Sc., selaku pembimbing tesis sekaligus kepala Laboratorium *Human Center Computing and Visualization* (HCCV).
4. Dr. Adhi Dharma Wibawa, S.T., M.T., selaku Koordinator Bidang Keahlian Telematika / *Chief Information Officer* (CIO) sekaligus Dosen Pembimbing Akademik Program Magister (S2) Jurusan Teknik Elektro, Bidang Keahlian

Telematika / *Chief Information Officer* (CIO) Angkatan Tahun 2015, Fakultas Teknologi Elektro, Institut Teknologi Sepuluh Nopember atas arahan, bimbingan dan motivasinya dalam menyelesaikan perkuliahan maupun penulisan tesis ini.

5. Seluruh Pengajar dan staf Program Studi Magister (S2) Jurusan Teknik Elektro, Bidang Keahlian Telematika / *Chief Information Officer* (CIO), yang telah mentransfer ilmu pengetahuannya melalui kegiatan perkuliahan maupun praktikum serta membantu kelancaran pengurusan administrasi perkuliahan dan penyelesaian tesis ini.
6. Orang tua Penulis Bapak Abdul Madjid dan Ibu Choiriyah terimakasih atas segala do'a dan dukungannya sehingga penulis dapat menyelesaikan tesis ini tepat waktu.
7. Istriku tersayang Novitasari Windhiarto dan anak-anakku yang dengan penuh kesabaran, mendukung dan mendoakan demi selesainya studi ini. Semoga kita semua selalu mendapat ridlo-Nya dalam keberkahan. Aamiin.
8. Mahasiswa Program Studi Magister (S2) Telematika / *Chief Information Officer* (CIO) Angkatan 2015 yang selalu kompak dan saling mendukung, saling mendoakan baik dalam perkuliahan maupun dalam penyelesaian penulisan tesis ini.
9. Laskar Lab HCCV (Atyanta, Pak Kamid, Mbak Fula, Mas Mamad, Mas adi) terima kasih atas support, kekompakan dan bantuannya. Semoga Sukses selalu.

Semoga Allah SWT membalas kebaikan semua pihak yang telah memberi kesempatan, dukungan dan bantuan dalam menyelesaikan tesis ini. Penulis menyadari bahwa tesis ini masih jauh dari sempurna, oleh karena itu saran dan kritik yang membangun sangat diharapkan demi kesempurnaan tulisan ini, sehingga tesis ini memberikan manfaat yang baik bagi agama, bangsa dan negara.

Surabaya, 12 Juni 2017

Penulis

DAFTAR ISI

DAFTAR GAMBAR

Halaman ini sengaja dikosongkan

DAFTAR TABEL

DAFTAR NOTASI

$\Delta(x_i, y_i)$	:	jarak antara amatan x dengan y
$x_i \; dan \; y_i$	:	nilai dari variabel ke-i pada titik x dan y
x_{new}	:	data buatan baru (*synthetic minority*)
α	:	angka acak dari 0 dan 1
x_t	:	amatan yang dipilih dari kelas minoritas
x_s	:	amatan lain pada kelas minoritas
M_s	:	Relevansi subset fitur
F_t	:	Jumlah fitur
$\overline{r_{cf}}$	:	Rata-rata koefisien korelasi linear antara fitur dan kelas
$\overline{r_{ff}}$	:	Rata-rata koefisien korelasi linier antar fitur yang berbeda
$f(n)$	:	Fungsi evaluasi
$g(n)$	:	Merupakan *cost* dari *initial state* ke *current state.*
$h(n)$	:	Perkiraan *cost* dari *current state* ke *goal state*.
P(Y|X)	:	probabilitas data dengan vektor X pada kelas Y
P(Y)	:	probabilitas awal kelas.
P(X)	:	probabilitas data dengan vector X
X_{raw}	:	Variabel untuk data mentah
X_{fitur}	:	Variabel untuk nama fitur
X_{data}	:	Variabel untuk isi data
n	:	Jumlah data
TP	:	Jumlah pengujian postif yang diklasifikasikan benar
TN	:	Jumlah pengujian postif yang diklasifikasikan salah
FP	:	Jumlah pengujian negatif yang diklasifikasikan benar
FN	:	Jumlah pengujian negatif yang diklasifikasikan salah
Total_populasi		Jumlah keseluruhan populasi atau semua sampel
Akurasi	:	Variabel untuk hasil akurasi
Presisi	:	Variabel untuk hasil presisi
Recall	:	Variabel untuk hasil *recall*
$F_{measure}$	:	Variabel untuk bobot harmonik *mean* presisi dan *recall*

$\bar{X}_{Presisi}$	:	Variabel hasil rata-rata presisi
$\bar{X}_{Recall}$	:	Variabel hasil rata-rata *recall*
i	:	Iterasi; i = 1,2,…,n

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Meningkatnya kebutuhan sistem Teknologi Informasi dan Komunikasi (TIK) di instansi swasta, organisasi atau Pemerintahan dan dalam rangka meningkatkan efisiensi sumber daya (manusia, finansial, infrastruktur TIK). Pengelolaan TIK merupakan suatu kewajiban untuk menjaga dan mendukung layanan publik yang prima. Sistem pelayanan untuk pihak luar dengan pengguna jumlah besar, dan persyaratan ketersediaan sistem yang tinggi.

Salah satu hal yang sangat penting dan mendasar dalam permasalahan keamanan informasi yang cukup signifikan dan berisiko terhadap data pada Data Center adalah masuknya *user* dan program (misalnya : *bot, worm, trojan horse, virus, Dos Attack, malware*) yang tidak sah sehingga dapat merusak sistem. Meskipun telah terpasang perangkat keamanan informasi berupa *Intrusion Detection System* (IDS), tidak menutup kemungkinan serangan itu bisa masuk ke dalam jaringan kita. Sistem strategis tentu akan menjadi sasaran menarik oleh para *attacker*. Dengan mencoba-coba untuk menyusup atau memang ada unsur kesengajaan berniat jahat mencoba menyusup ke dalam sistem.

Salah satu bentuk *malicious software* (malware) yang berbahaya adalah *robot network* (Botnet), Botnet merupakan sebuah *zombie* didalam jaringan dari ribuan atau jutaan perangkat yang tersambung ke Internet seperti *Personal Computer* (PC), *smartphone*, *tablet*, *router*, perangkat cerdas, atau *gadget* lainnya. Dimana *Bot* tersebut diinfeksi dengan malware khusus sehingga dapat dikendalikan oleh *Cybercriminal* untuk memberikan serangan siber. Tujuan dari pada kegiatan penggunaan *Bot* itu sendiri adalah untuk mencuri informasi-informasi penting yang nanti dari informasi penting tersebut akan disalah gunakan atau bahkan tujuan dari *hacking* sendiri selain untuk mencuri informasi bisa jadi untuk merusak sistem yang telah dibuat sehingga sistem tidak dapat berjalan dengan semestinya.

Pada dasarnya terdapat 3 komponen sebuah Botnet dapat berjalan yaitu *Botmaster, Command and Control* (C&C) Server dan *bot*. C&C merupakan server

pusat yang digunakan untuk merekam atau mengendalikan suatu komputer yang terinfeksi, dan *Botmaster* berperan sebagai pengendali dari C&C server tersebut. Hal ini memungkinkan Botnet dapat dikendalikan oleh *hacker* dari jarak jauh untuk meluncurkan berbagai serangan jaringan, seperti serangan DDoS, spam, *click fraud*, pencurian identitas dan *phishing*. Sehingga Botnet telah menjadi alat yang populer dan produktif di balik banyaknya serangan siber. Karakteristik Botnet dalam komunikasi dengan C&C yang berkembang saat ini adalah menggunakan protokol HTTP (Vania, et al., 2013). Sehingga dalam deteksinya dapat menggunakan algoritma *Data Mining*, untuk otomatisasi dalam mendeteksi karakteristik Botnet dalam jumlah yang besar.

Penelitian dengan deteksi intrusi, malware atau Botnet yang sudah ada dipandang masih perlu dilakukan pengembangan sebagai bentuk aktif penelitian dan proaktif dalam identifikasi dengan kemunculan intrusi baru. Pada penelitian ini fokus pada improvisasi pengolahan data untuk deteksi Botnet, penelitian ini merupakan bagian kecil dari sistem besar *Intrusion Prevention System*.

Metode yang diusulkan adalah Naïve Bayes yang digunakan untuk klasifikasi data *network flow* yang terdiri dari data normal dan Botnet. Untuk deteksi trafik Botnet sebagai sasaran utama dimana jika dibandingkan dengan jumlah trafik *background*/normal secara keseluruhan adalah sangat kecil. Untuk meningkatkan akurasi deteksi pada penelitian ini akan menerapkan teknik teknik *Synthetic Minority Over-sampling Technique* (SMOTE) (Chawla & Bowyer, 2002) atau dapat disebut teknik untuk kendala ketidakseimbangan kelas (*imbalanced data*) dan selanjutnya penambahan metode seleksi fitur untuk pemilihan fitur dari *noise feature* menggunakan metode *Best First Search.*

1.2 Rumusan Masalah

Klasifikasi merupakan bentuk pendekatan untuk mengenali atau deteksi trafik jaringan Botnet, trafik jaringan Botnet tidak menggunakan keseluruhan utilitas jaringan yang tersedia, trafik Botnet akan menyusup kecil bersamaan dengan trafik *background*/normal. Karena risiko dari serangan Botnet yang berbahaya maka sebuah sistem deteksi harus memiliki kriteria akurat dan tepat dalam mengenali trafik Botnet. Dilihat dari perbandingannya, trafik Botnet

merupakan kelas kecil diantara trafik *background*/normal, maka perlu adanya inovasi dan solusi sehingga didapatkan peningkatan keakurasian dan ketepatan pada sistem deteksi Botnet.

1.3 Tujuan Penelitian

Karakteristik Botnet adalah mampu menyediakan platform yang dapat didistribusikan pada kegiatan ilegal seperti *spam, phishing, click fraud,* dan pencurian *password* . Dalam melakukan aksinya Botnet tidak membanjiri penuh utilitas jaringan yang ada, Botnet hanya menggunakan trafik kecil yang terjadi komunikasi antara Bot dengan C&C server , sehingga pada saat *capture* trafik untuk deteksi Botnet maka hal yang terjadi adalah *capture* keseluruhan trafik yang berjalan. Dari hasil *capture* didapatkan data *background* trafik yang besar (*mayor*) dan trafik botnet yang kecil (*minor*), terjadi *imbalance data.* Hal ini apabila dilakukan klasifikasi akan mempengaruhi hasil akurasi. Sehingga dengan implementasi teknik *oversampling data* menggunakan *Synthetic Minority Over-sampling Technique* (SMOTE) untuk *imbalance data* dan seleksi fitur menggunakan metode *Best First Search* (BFS) ini diharapkan akurasi dari deteksi Botnet untuk menjamin kebersihan trafik tanpa disusupi Botnet meningkat dibandingkan dengan deteksi Botnet tanpa menggunakan teknik SMOTE dan metode BFS.

1.4 Batasan Masalah

Penelitian ini dilakukan untuk deteksi Botnet berbasis *packet flows* dengan menggunakan dataset uji CTU-13 Skenario data 1 (Sebastian, 2017) (Garcia, et al., 2014), Botnet yang dijalankan di *Czech Technical University* (CTU). Dimana karakteristik Bot (*neris.exe*) adalah berbasis HTTP dengan kanal C&C, mengirim *spam,* dan beberapa tindakan *click-fraud*.

1.5 Kontribusi

Kontribusi dari penelitian ini adalah dapat memberikan *alternative* metode untuk sistem deteksi Botnet yang menitikberatkan pada teknik deteksi dengan *over-sampling* pada *imbalance data* dan pemilihan fitur *packet flow* jaringan Botnet.

1.6 Sistematika Penulisan

BAB 1 : Bab ini merupakan bagian yang akan menguraikan latar belakang pengambilan judul penelitian, rumusan masalah yang diangkat, tujuan penelitian, batasan masalah, kontribusi dan metodologi skema penulisan penelitian.

BAB 2 : Bab ini berisi tentang tinjauan pustaka yang digunakan sebagai acuan dalam membangun sistem untuk menyelesaikan permasalahan yang diuraikan dalam bab pertama.

BAB 3 : Bab ini berisi tentang uraian alur metodologi dan metode yang digunakan untuk membangun secara rinci dan metode pengujian yang digunakan dalam penelitian ini.

BAB 4 : Bab ini menyajikan pembahasan hasil yang diperoleh dan model deteksi menggunakan klasifikasi yang dibangun dalam penelitian ini. Serta evaluasi dan pengujian sistem yang telah dihasilkan untuk melihat kesesuaian dengan tujuan yang diharapkan.

BAB 5 : Bab ini berisi kesimpulan dari hasil penelitian yang sudah dilakukan dan saran untuk perbaikan penelitian ini kedepannya.

BAB 2
KAJIAN PUSTAKA

Pada bab ini akan dibahas secara singkat tentang gambaran umum yang berkaitan dengan penelitian yang dilakukan, meliputi kajian pustaka dan dasar teori. Kajian pustaka didasarkan pada penelitian yang telah dilakukan sebelumnya, selain itu dijelaskan pula beberapa dasar teori yang menunjang penelitian ini.

.

2.1 Kejahatan Siber (*Cybercrime*)

Banyak nama kejahatan Siber (*Cybercrime*) (Sharma, et al., 2016) dimana tindakan kriminal tersebut dilakukan dengan menggunakan teknologi komputer sebagai alat kejahatan utama. *Cybercrime* merupakan kejahatan yang memanfaatkan perkembangan teknologi komputer khususnya internet. *Cybercrime* didefinisikan sebagai perbuatan melanggar hukum yang memanfaatkan teknologi komputer yang berbasis pada kecanggihan perkembangan teknologi internet.

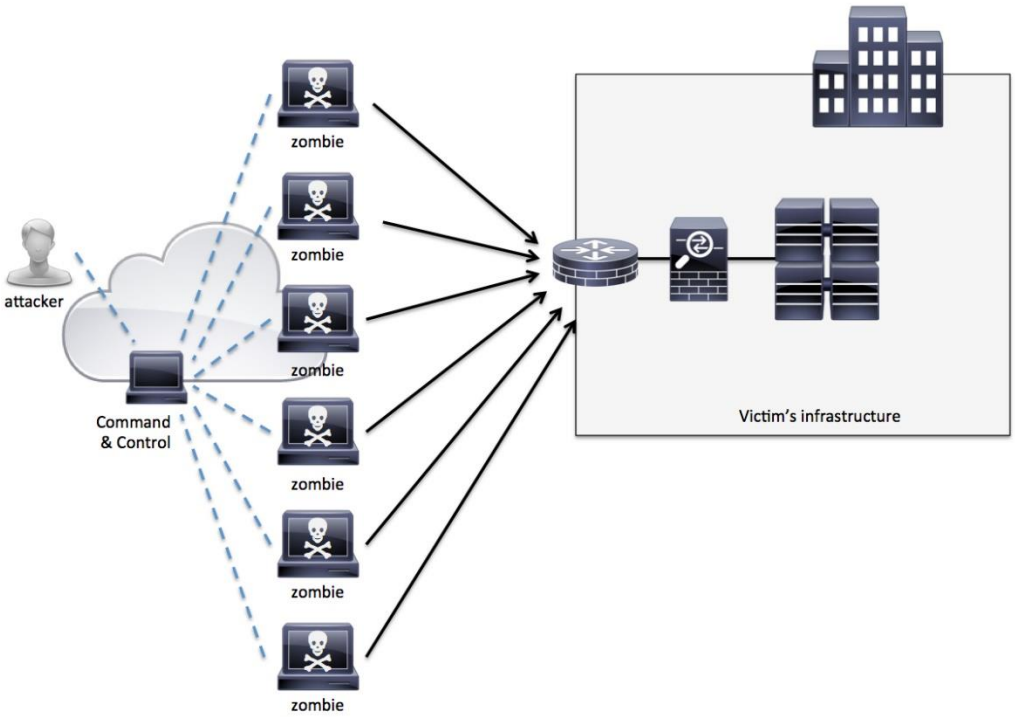
Beberapa karakteristik dilihat dari macam-macam kejahatannya maka untuk mempermudah penanganannya *Cybercrime* dapat diklasifikasikan sebagai berikut :

1) *Cyberpiracy* : Penggunaan teknologi komputer untuk mencetak ulang software atau informasi, lalu mendistribusikan informasi atau software tersebut lewat teknologi komputer.
2) *Cybertrespass* : Penggunaan teknologi komputer untuk meningkatkan akses pada sistem komputer suatu organisasi atau individu.
3) *Cybervandalism* : Penggunaan teknologi komputer untuk membuat program yang menganggu proses transmisi elektronik, dan menghancurkan data dikomputer.

2.1.1 Robot Network (Botnet)

Botnet terdiri dari “Bot” dan “Network”, Bot Didapatkan dari kata "robot" yang dimana merupakan sebuah proses otomatis yang berinteraksi dengan layanan jaringan lain. Bot dapat digunakan untuk tujuan yang baik atau jahat. Jika

digunakan untuk tujuan jahat, mereka bekeja sebagai *Zombie* yang disusupkan pada jaringan komputer menggunakan perangkat lunak yang bisa dijalankan, dipantau dan diperintahkan bereaksi oleh pembuatnya (*Master refer / Botmaster*).



Gambar 2.1 Diagram Botnet[1]

Botnet juga berisi kegiatan menyusupkan program-program tertentu kepada server-server komputer dimana program-program tersebut biasanya disusupkan sebagai *Worms*, *Trojan horse*, atau *Backdoors*, seperti pada Gambar 2.1. di bawah perintah *Botmaster/attacker* dan dikendalikan dengan sebuah *remote*, sehingga program tersebut dapat bekerja kapanpun dengan *Master* tadi yang tujuannya untuk mengganggu ataupun merusak suatu jaringan atau sistem operasi komputer (kejahatan) 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

[1] Sumber gambar : http://www.cisco.com/c/en/us/about/security-center/guide-ddos-defense.html

yang diciptakan untuk menyusup atau merusak system komputer atau jejaring komputer tanpa izin pemilik perangkat (*informed consent*).

2.1.2 Bentuk Serangan Botnet

Adapun bentuk serangan Botnet (Zao, et al., 2013) adalah :

1) ***Distributed Denial of Service (DDOS) attacks***, adalah serangan kepada jaringan yang mengakibatkan hilangnya layanan kepada *user*, biasanya hilangnya konektifitas 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 paket *sniffer* (penyadap paket yang melewati jaringan) untuk melihat data yang menarik di komputer, sebagian besar *sniffer* mengambil informasi yang *sensitive* 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 Adversitement Add-ons & Browser Helper Objects***, dengan membuat situs palsu, dengan membuat bebrapa iklan dan mendaftar *pay per cliks* 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.

2.2 Definisi Deteksi Botnet

Definisi deteksi adalah usaha menemukan dan menentukan keberadaan, anggapan, atau kenyataan. Sehingga deteksi Botnet dapat diartikan usaha

menemukan sebuah Botnet dengan parameter tertentu didalam jaringan dengan menggunakan teknik atau metode tertentu

Ada dua pendekatan utama dalam deteksi Botnet yaitu dengan metode *Honeypot* dan *Passive Network Monitoring* (Vania, et al., 2013).

2.2.1 Honeypot

Honeynet merupakan suatu jaringan yang terdiri dari satu *Honeypot* ataupun lebih. Sedangkan *Honeypot* sendiri adalah perangkap yang bertindak sebagai umpan untuk memikat *client* yang berpotensi sebagai penyerang yang mecoba masuk secara paksa ke dalam suatu sistem. *Honeypot* digunakan untuk memantau dan mempelajari metode yang digunakan *hacker* untuk menembus suatu sistem. Informasi yang didapatkan akan digunakan sebagai tindakan preventif dimasa mendatang.

2.2.2 Passive Monitoring

Pendekatan lain untuk deteksi Botnet dilakukan melalui pendekatan *passive network monitoring*. Teknik ini dilakukan dengan memonitor lalu lintas yang melewati suatu jaringan dan kemudian dilakukan analisis untuk mengidentifikasi keberadaan dan memahami karakteristik Botnet. Teknik ini dapat diklasifikasikan menjadi *signature-based*, *anomaly-based*, DNS-*based*, dan *mining-based*.

1) *Signatured Based Detection*

Teknik *signature based* mengidentifikasi paket yang melewati suatu jaringan dan kemudian mencocokannya dengan informasi-informasi yang diperoleh dari penelitian yang sebelumnya. Bagian paket yang diteliti dapat berupa *signature*, perilaku maupun ukuran dari paket. Dalam hal ini, *signature* adalah kode ataupun perilaku yang secara unik mengidentifikasikan Botnet tertentu. Kelemahan dari teknik ini adalah tidak dapat diterapkan untuk mendeteksi Botnet yang belum diketahui.

2) *Anomaly Based*

Anomaly based detection melakukan pendekatan untuk mendeteksi Botnet berdasarkan pada beberapa anomali lalu lintas jaringan seperti latensi jaringan yang tinggi, volume lalu lintas yang tinggi, lalu lintas yang tidak biasa di *port*, dan perilaku sistem yang tidak biasa yang dapat mengindikasikan

potensi ancaman adanya Bot berbahaya dalam jaringan. Pendeteksian model ini didasarkan pada perubahan dalam pola pemakaian atau kelakuan sistem.

3) *DNS Based*

DNS *based detection* dilakukan berdasarkan informasi DNS yang dihasilkan oleh sebuah Botnet. Teknik DNS-*based* mirip dengan teknik *anomaly-based* sebagaimana menerapkan algoritma deteksi anomali untuk diterapkan pada deteksi DNS. Bot biasanya melakukan koneksi dengan C&C server untuk mendapatkan perintah ataupun *update*. Untuk mengakses C&C server, bot melakukan *query* DNS untuk menemukan lokasi C&C server. Jadi, sangatlah memungkinkan untuk mendeteksi Botnet dengan memantau dan mendeteksi lalu lintas DNS.

4) *Mining Based*

Data mining umumnya mengacu pada model proses otomatisasi untuk pengauditan data dalam jumlah besar. Perkembangan dalam proses data mining telah diterapkan untuk algoritma pada *machine learning*, *pattern recognition machine* dan lainnya. Keuntungan utama dari pendekatan ini adalah bahwa penggalian informasi terhadap kumpulan data yang besar dapat dilakukan secara otomatis untuk dapat menghasilkan model deteksi ringkas dan akurat. Tujuan utama dari teknik ini adalah untuk mendeteksi penyebaran Botnet dan menemukan C&C server berdasarkan *logfiles* yang telah dikumpulkan.

2.3 Data Mining untuk Deteksi

Sebenarnya pekerjaan *data mining* dapat dibagi menjadi empat kelompok, yaitu model prediksi (*prediction modelling*), analis kelompok (*cluster anlysis*), analisis asosiasi (*association analysis*), dan deteksi anomali (*anomaly detection*).

Dalam penelitian yang dikerjakan disini akan masuk pada kelompok pekerjaan deteksi anomali, pekerjaan deteksi anomali berkaitan dengan pengamatan sebuah data dari sejumlah data yang secara signifikan mempunyai karakteristik yang berbeda dari sisa data yang lain. Data-data yang karakteristiknya menyimpang (berbeda) dari data yang lain disebut *outlier*. Algoritma deteksi anomali yang baik harus mempunyai laju deteksi yang tinggi dan laju error yang rendah. Deteksi anomali dapat diterapkan pada sistem jaringan untuk mengetahui pola data yang

memasuki jaringan sehingga penyusupan bisa ditemukan jika pola kerja yang datang berbeda. Perilaku kondisi cuaca yang mengalami anomaly juga dapat dideteksi dengan algoritma ini.

2.3.1 Konsep Anomali Data

Dataset terdiri atas sejumlah data dimana setiap data mempunyai jumlah fitur yang mendeskripsikan karakter data tersebut. Dalam *data mining* set data tersebut dapat diolah untuk menghasilkan informasi yang berguna, seperti klasifikasi, pengelompokan, prediksi dan sebagaianya. Tidak jarang data-data yang akan diolah ditemukan data yang karakteristiknya menyimpang/berbeda dengan data pada umumnya. Data yang menyimpang tersebut dinamakan *outlier* (Prasetyo, 2012).

Outlier biasanya dianggap sebagai objek/data yang jumlahnya sangat kecil jika dibandingkan dengan data normal lainnya, misalnya probabilitas kemunculannya satu dari seribu data, tetapi bisa menjadi seribu jika data sudah berjumlah satu juta. Dengan demikian, deteksi *outlier* pada data yang menyimpang merupakan pekerjaan yang penting untuk berbagai keperluan dalam *data mining.*

Penerapan deteksi anomali dapat ditemukan pada sejumlah bidang kegunaan, yaitu seperti deteksi kecurangan (*fraud detection),* deteksi penyusupan (*intrusion detection*), gangguan ekosistem, kesehatan masyarakat dan kedokteran.

Pada prinsipnya, data berkarakter menyimpang akan memiliki perbedaan yang besar (kemiripan yang kecil) terhadap data lain pada umumnya, sehingga dengan mengukur tingkat kemiripan (jarak) semua data, maka bisa dapat mengetahui karakter menyimpang tersebut.

2.4 Definisi *Imbalance Class*

Imbalance class adalah sebuah keadaan yang menggambarkan tidak seimbangnya porsi data antara sebuah kelas dengan kelas yang lain. Permasalahan seperti ini menjadi penting dikarenakan pada beberapa aplikasi *data mining*, akurasi model prediksi terhadap kelas minor lebih menarik / lebih penting daripada akurasi model prediksi terhadap kelas mayor, padahal data kelas mayor akan lebih terlatih daripada data kelas minor karena *classifier* biasa cenderung akan membiaskan

prediksi kelas minor ke kelas mayor. Hal ini mengakibatkan terjadinya *misclassification* yang mengakibatkan akurasi untuk prediksi kelas minor cenderung buruk serta memungkinkan kelas minor hanya dianggap sebagai *outlier*. *Undersampling.*

Terdapat 4 cara untuk menangani imbalance class, (He, et al., 2009) yaitu metode *sampling, cost sensitive method, kernel based and active learning method,* dan *additional method*.

2.4.1 Metode Sampling (*Sampling based Approaches*)

Umumnya penggunaan metode *sampling* dalam aplikasi pembelajaran yang tidak seimbang terdiri dari modifikasi data yang tidak seimbang dengan menggunakan teknik tertentu sehingga memberikan solusi distribusi data yang seimbang.

Dalam teknik *sampling* sendiri terdapat 2 metode untuk menyelesaikan permasalahan data tidak seimbang, yaitu dengan *undersampling* dan *oversampling*.

1) *Undersampling Data*

Undersampling adalah membuat *instance class minoritas* menjadi lebih sedikit. Kekurangan cara ini dapat membuat kehilangan informasi penting, karena pengurangan *instance.*

2) *Oversampling Data*

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 (Chawla & Bowyer, 2002).

Metode *Synthetic Minority Oversampling Technique* (SMOTE) diusulkan oleh (Chawla & Bowyer, 2002) 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.

Algoritma SMOTE terdiri dari dua bagian utama, bagian pertama berisi perulangan untuk mencari *k*-tetangga terderkat dan bagian kedua untuk membuat data sintetis dari kelas minor. Untuk menguraikan metode SMOTE, berikut dijelaskan algoritma SMOTE dalam bentuk *pseudocode* (Chawla & Bowyer, 2002).

1. **Masukan :** Jumlah kelas minoritas (T), jumlah SMOTE (N%), jumlah *k*-tetangga terdekat (*k*)
2. **Keluaran :** (N/100) * T

   (* Jika N kurang dari 100%, maka sampel acak kelas minoritas sesuai persentasenya yang akan dibangkitkan *)

   a. ***If*** N<100
   b. ***Then*** acak dari sampel T kelas minoritas
   c. T = (N/100) * T
   d. N = 100
   e. ***End if***
   f. N = (int)(N/100) (* Jumlah SMOTE diasumsikan berada dalam kelipatan integral 100 *)
   g. *k* adalah jumlah tetangga terdekat
   h. *numattrs* adalah jumlah atribut
   i. *Sample [ ][ ]* adalah *array* dari sampel asli kelas minoritas
   j. *newindex* adalah membuat hitungan jumlah sampel sintetik yang dihasilkan, diinisialisasi menjadi 0
   k. *Synthetic [ ][ ]* adalah *array* untuk sampel sintetik (* menghitung *k* tetangga terderkat untuk masing-masing hanya sampel kelas minoritas *)
   l. ***For*** i ← 1 ***to*** T
   m. hitung *k* tetangga terdekat untuk *i*, dan simpan *index* di *nnarray*
   n. Populasi (N, *i*, *nnarray*) (* fungsi untuk menghasilkan sintetik *)

o. ***Endfor***

p. ***While*** N ≠ 0

q. Memilih angka acak antara 1 dan *k*, yang disebut *nn*. Pada tahap ini memilih 1 *k* tetangga terdekat dari *i*

r. ***for*** attr ← 1 ***to*** numattrs

s. Menghitung → *dif = Sample[nnarray[nn][attr] – Sample[i][attr]*

t. Menghitung → *gap = angka acak antara 0 dan 1*

u. *Synthetic[newindex][attr] = Sample[i][attr] + gap * dif*

v. ***endfor***

w. *newindex++*

x. N = N-1

y. ***Endwhile***

z. ***Return*** (* akhir *pseudocode* *)

Pembangkitan data buatan yang berskala numerik berbeda dengan kategorik. Data numerik diukur jarak kedekatannya dengan *Euclidean Distance* sedangkan data kategorik lebih sederhana yaitu dengan nilai modus. Perhitungan jarak antar contoh kelas minor yang peubahnya berskala kategorik dilakukan dengan persamaan *Value Difference Metric* (VDM) (Barro, et al., 2013). Untuk pengukuran jarak kedekatan dua elemen vector dengan *Euclidean Distance Metric*. Persamaan untuk menghitung jarak *Euclidean* antara titik $x = (x_1, x_2, \dots, x_n)$ dan titik $y = (y_1, y_2, \dots, y_n)$ adalah pada persamaan (1) (Tsai & Yu, 2016).

$$\Delta(x_i, y_i) = \sqrt{\sum_{1 \le i \le n} (x_i - y_i)^2} \qquad (1)$$

Dimana :

$\Delta(x_i, y_i)$: jarak antara amatan x dengan y

n : adalah jumlah variable

$x_i \; dan \; y_i$: nilai dari variabel ke-i pada titik x dan y

Prosedur pembangkitan data buatan dapat dilakukan dengan 2 (dua) opsi yaitu data numerik dan data kategorik. Berikut prosedur tersebut :

1) Data Numerik

a) Hitung perbedaan antar vektor utama dengan *k*-tetangga terdekatnya.

b) Kalikan perbedaan dengan angka yang diacak diantara 0 dan 1.
c) Tambahkan perbedaan tersebut ke dalam nilai utama pada vektor utama asal sehingga diperoleh vektor utama baru.

2) Data Ketegorik
a) Pilih mayoritas antara vektor utama yang dipertimbangkan dengan *k*-tetangga terdekatnya untuk nilai nominal. Jika terjadi nilai samamaka pilih secara acak.
b) Jadikan nilai tersebut data contoh kelas buatan baru.

Adapun persamaan untuk pembangkitan data buatan baru (Xie, et al., 2015) adalah pada persamaan (2)

$$x_{new} = \alpha * x_s + (1 - \alpha) * x_t \quad (2)$$

Dimana :

x_{new} : data buatan baru (*synthetic minority*)
α : angka acak dari 0 dan 1
x_t : amatan yang dipilih dari kelas minoritas
x_s : amatan lain pada kelas minoritas

Amatan yang dipilih x_t dari sampel lain x_s pada kelas minor dengan hitungan menggunakan *Euclidean Distance,* pada umumnya *k* bernilai 5. Lalu hitung perbedaan antara sampel minoritas dan tetangga terdekat yang dipilih secara acak dari *k*-minoritas. Kalikan perbedaan dengan angka yang diacak diantara 0 dan 1. Kemudian tambahkan perbedaan tersebut ke dalam nilai utama pada amatan utama asal sehingga diperoleh vektor amatan baru (x_{new}). Amatan baru ditambahkan kedalam data minor atau kelas minor.

2.4.2 Cost-Sensitive Method

Penanganan dengan memberikan *cost* yang berbeda untuk *misclassification* ke kelas minor dan kelas mayor, *cost* kelas minor lebih tinggi daripada kelas mayor.

2.5 *Correlation Based Feature Selection* (CBFS)

Seleksi atribut melibatkan pencarian melalui semua kemungkinan kombinasi antar atribut dalam data untuk menemukan subset dari atribut mana yang paling sesuai untuk pengenalan atau prediksi. Untuk itu diperlukan evaluator atribut dan metode pencarian (*search method*). Evaluator menentukan metode apa yang digunakan untuk menetapkan nilai untuk setiap subset atribut. Metode pencarian menentukan gaya pencarian yang dilakukan. Metode pencarian sendiri terdapat beberapa cara diantaranya *Best First Method* dan *Greedy Search*.

2.5.1 Evaluasi Fitur

CFS adalah pendekatan heuristik untuk mengevaluasi nilai atau manfaat subset fitur. Ini adalah salah satu teknik untuk menilai relevansi fitur dengan mengukur korelasi antara fitur dan kelas dan antara fitur dan fitur lainnya. Dengan jumlah fitur F_t dan kelas *c*, CFS mendefinisikan relevansi fitur subset dengan menggunakan persamaan (3) korelasi *Pearson* (Karthikeyan & Thangaraju, 2015).

$$M_s = \frac{F_t \overline{r_{cf}}}{\sqrt{F_t + (F_t - 1)\,\overline{r_{ff}}}} \qquad (3)$$

Dimana :

M_s : Relevansi subset fitur

F_t : Jumlah fitur

$\overline{r_{cf}}$: Rata-rata koefisien korelasi linier antara fitur dan kelas

$\overline{r_{ff}}$: Rata-rata koefisien korelasi linier antar fitur yang berbeda.

Dalam keadaan tertentu, CFS menambahkan (*forward*) atau menghapus (*backward*) satu fitur dalam satu waktu.

2.5.2 Metode Best First Search (BFS)

Metode ini merupakan kombinasi dari metode *depth-first search* dan *breadth-first search*. Pada metode BFS, pencarian diperbolehkan mengunjungi node yang ada di *level* yang lebih rendah (*backward*) , jika ternyata *node* pada *level*

yang lebih tinggi (*foreward*) ternyata memiliki nilai heuristic yang lebih buruk. Secara iteratif berdasarkan nilai heuristik terbaik untuk node yang berdekatan untuk setiap node saat ini (Karthikeyan & Thangaraju, 2015).

Untuk menggunakan *Best-First-Search*, maka memerlukan dua daftar simpul, yaitu :

1) OPEN

Berisi simpul yang dihasilkan dari fungsi heuristik tapi belum dievaluasi, memiliki antrian prioritas dimana elemen dengan prioritas tertinggi adalah yang memiliki nilai paling baik yang dihasilkan fungsi heuristik.

2) CLOSED

Berisi simpul yang sudah dievaluasi. Kita perlu tetap menyimpan simpul-simpul ini dalam memori jika kita ingin melakukan *search* pada *Graph*, sehingga jika kita menemui suatu simpul kita bisa memeriksa apakah simpul ini sudah pernah dieavaluasi atau belum

Tabel 2.1 Algoritma *Best First Search*

1. Mulai dengan OPEN hanya berisi *initial state*
2. Sampai *goal* ditemukan atau tidak ada lagi simpul yang tersisa dalam OPEN, lakukan :
    a. Pilih simpul terbaik dalam OPEN
    b. Telusuri *successor*-nya
    c. Untuk tiap *successor*, lakukan :
        i. Jika belum pernah ditelusuri sebelumnya, evaluasi simpul ini, tambahkan kedalam OPEN dan catat *parentnya*.
        ii. Jika sudah pernah ditelusuri, ganti *parent* nya jika jalur baru lebih baik dari sebelumnya.

Fungsi Heuristik yang digunakan merupakan prakiraan (estimasi) *cost* dari initial state ke *goal state*, yang dinyatakan dengan persamaan (4).

$$f(n) = g(n) + h(n) \quad (4)$$

Dimana :

$f(n)$: Fungsi evaluasi

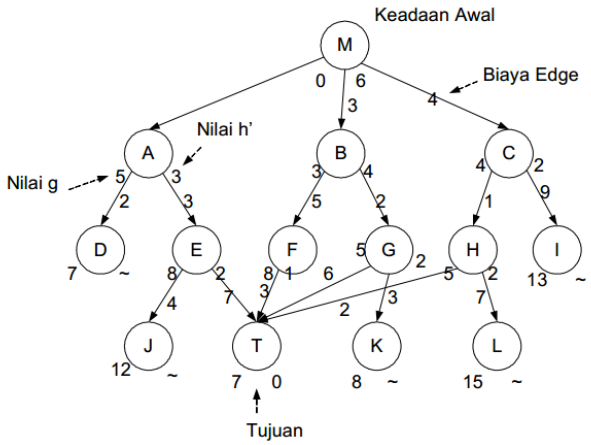
$g(n)$: Merupakan *cost* dari *initial state* ke *current state.*

$h(n)$ *:* Perkiraan *cost* dari *current state* ke *goal state*.

Sebagai contoh Misalkan kita memiliki ruang pencarian seperti pada Gambar 2.1 dimana *Node M* merupakan keadaan awal dan *node T* merupakan tujuannya. Biaya *edge* yang menghubungkan *node M* dengan *node A* adalah biaya yang dikeluarkan untuk bergerak dari kota *M* ke kota *A.* Nilai *g* diperoleh berdasarkan biaya *edge* minimal. Sedangkan nilai *h* di node *A* merupakan hasil perkiraan terhadap biaya yang diperlukan dari *node A* untuk sampai ke tujuan. *h(n)* bernilai ~ jika sudah jelas tidak ada hubungan antara *node n* dengan *node* tujuan (jalan buntu). Kita bisa mengurut nilai untuk setiap *node* seperti pada Tabel 2.2.

Tabel 2.2 Penyelesaian dan Tabel Status tiap Node

Node(n)	**g(n)**	**h(n)**	**f(n)**
M	0	6	6
A	5	3	8
B	3	4	7
C	4	2	6
D	7	~	~
E	8	2	10
F	8	1	9
G	5	2	7
H	5	2	7
I	13	~	~
J	12	~	~
K	8	~	~
L	15	~	~
T	17	0	7



Gambar 2. 1 Contoh pencarian jarak terdekat

2.5.3 Metode Greedy Search

Gready Search (GS) atau *Greedy Hill Climbing Search* akan mencari ruang subset fitur dalam batasan waktu tertentu jika diperlukan algoritma seleksi fitur dan ini beroperasi pada data dengan jumlah fitur yang besar.

Algoritma Greedy adalah algoritma yang mengikuti pemecahan masalah langkah per langkah, dengan cara:

1) Mengambil pilihan terbaik yang diperoleh pada saat itu tanpa memperhatikan konsekuensi ke depan (Prinsip "*take what you can get now*!")
2) Membuat pilihan optimum lokal pada setiap langkah dengan mengharapkan optimum global.

Algoritma greedy merupakan metode yang paling populer untuk memecahkan persoalan optimasi.

2.6 Konsep Klasifikasi

Klasifikasi merupakan suatu pekerjaan untuk menilai objek data untuk memasukkannya kedalam kelas tertentu dari sejumlah kelas yang tersedia (Prasetyo, 2012). Dalam klasifikasi terdapat dua pekerjaan utama yang akan dilakukan yaitu :

1) Pembangunan model sebagai *prototype* untuk disimpan sebagai memori, dan

2) Penggunaan model tersebut untuk melakukan pengenalan/ klasifikasi/ prediksi/ deteksi pada suatu objek data lain agar diketahui dikelas mana objek data tersebut dalam model yang sudah disimpannya.

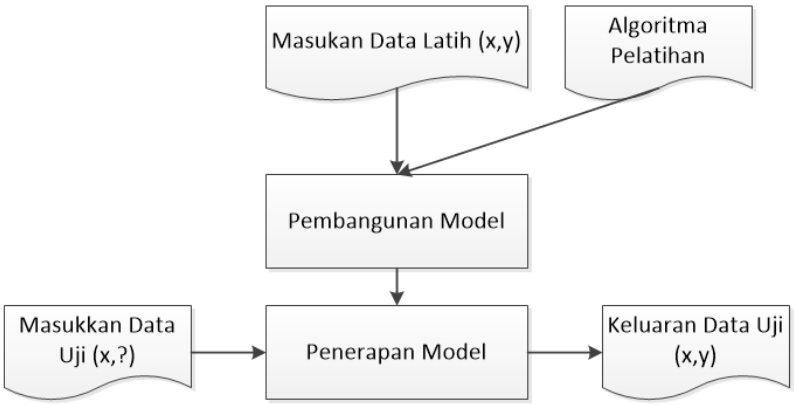
Contoh aplikasi yang sering ditemui adalah pengklasifikasian jenis hewan, yang mempunyai sejumlah atribut. Dengan atribut tersebut, jika ada hewan baru, kelas hewannya bisa langsung diketahui. Contoh lain adalah bagaimana melakukan diagnosi penyakit kulit kanker melanoma (Amaliyah, 2011), yaitu dengan melakukan pembangunan model berdasarkan data latih yang ada, kemudian menggunakan model tersebut untuk mengidentifikasi penyakit pasien baru sehingga diketahui apakah pasien tersebut menderita kanker atau tidak.

2.6.1 Model untuk Klasifikasi

Klasifikasi dapat didefinisikan sebagai pekerjaan yang melakukan pelatihan/pembelajaran terhadap fungsi target *f* yang memetakan setiap atribut (fitur) *x* ke satu dari sejumlah label kelas *y* yang tersedia. Pekerjaan pelatihan tersebut akan menghasilkan suatu model yang kemudia disimpan sebagai memori.

Model dalam klasifikasi mempunyai arti yang sama dengan kotak hitam, dimana ada suatu model yang menerima masukan, kemudian mampu melakukan pemikiran terhadap masukan tersebut. Dan memberikan jawaban sebagai keluaran dari hasil pemikirannya. Sebuah kerangka kerja (*framework*) klasifikasi ditunjukkan pada Gambar 2.2. pada gambar tersebut disediakan sejumlah data latih (x,y) untuk digunakan sebagai data pembangun model. Model tersebut kemudian dipakai untuk memprediksi kelas dari data uji (x,?) sehingga diketahui kelas *y* yang sesungguhnya.

Model yang dibangun pada saat pelatihan kemudian dapat digunakan untuk memprediksi label kelas data baru yang belum diketahui (Prasetyo, 2012). Dalam pembangungan model selama proses pelatihan tersebut diperlukan algoritma untuk membangunnya, atau bisa disebut dengan algoritma pelatihan (*learning algorithm*).



Gambar 2.2 Proses Pekerjaan Klasifikasi

Ada banyak algoritma pelatihan yang sudah dikembangkan oleh para peneliti, diantaranya adalah *K-Nearest Neighbor*, *Artificial Neural Nework*, *Naïve Bayes*, *Support Vector Machine* dan sebagainya. Setiap algoritma mempunyai kelebihan dan kekurangan, tetapi semua algoritma mempunyai prinsip sama, yaitu melakukan suatu pelatihan sehingga diakhir pelatihan, model dapat memetkan (memprediksi) setiap vector masukan ke label kelas keluaran dengan benar.

2.6.2 Naïve Bayes *Classifier*

Naive Bayes merupakan sebuah teknik untuk klasifikasi menggunakan probabilistik sederhana yang menghitung sekumpulan probabilitas dengan menjumlahkan frekuensi dan kombinasi nilai dari dataset yang diberikan. Algoritma mengunakan teorema Bayes dan mengasumsikan semua atribut independen atau tidak saling ketergantungan yang diberikan oleh nilai pada variabel kelas (Supriyanti, et al., 2016). Definisi lain mengatakan Naive Bayes merupakan teknik klasifikasi dengan metode probabilitas dan statistik yang dikemukan oleh ilmuwan Inggris Thomas Bayes, yaitu memprediksi peluang di masa depan berdasarkan pengalaman dimasa sebelumnya (Bustami, 2013).

Naive Bayes didasarkan pada asumsi penyederhanaan bahwa nilai atribut secara kondisional saling bebas jika diberikan nilai output. Dengan kata lain, diberikan nilai luaran, probabilitas mengamati secara bersama adalah produk dari probabilitas individu. Keuntungan penggunaan Naive Bayes adalah bahwa metode ini hanya membutuhkan jumlah data pelatihan (*Training Data*) yang kecil untuk

menentukan estimasi paremeter yang diperlukan dalam proses pengklasifikasian. Naive Bayes sering bekerja jauh lebih baik dalam kebanyakan situasi dunia nyata yang kompleks dari pada yang diharapkan (Pattekari & Parveen, 2012).

Kelebihan yang dimiliki oleh *Naive Bayes* adalah dapat menangani data kuantitatif dan data diskrit, *Naive Bayes* kokoh terhadap *noise*, *Naive Bayes* hanya memerlukan sejumlah kecil data pelatihan untuk mengestimasi parameter yang dibutuhkan untuk klasifikasi, *Naive Bayes* dapat menangani nilai yang hilang dengan mengabaikan instansiasi selama perhitungan estimasi peluang.

2.6.3 Persamaan Teorema Bayes

Persamaan (5) merupakan persamaan dari teorema Bayes (Prasetyo, 2012) adalah :

$$P(Y|X) = \frac{P(Y)\prod_{i=1}^{q} P(X_i|Y)}{P(X)} \quad (5)$$

Dimana :

P(Y|X) : probabilitas data dengan *vector* X pada kelas Y.

P(Y) : probabilitas awal kelas.

P(X) : probabilitas data dengan *vector* X

Nilai P(X) selalu tetap sehingga dalam perhitungan prediksi nantinya tinggal menghitung bagian $P(Y)\prod_{i=1}^{q} P(X_i|Y)$ dengan memilih yang tersebar sebagai kelas yang dipilih sebagai hasil prediksi. Sementara probabilitas independen $\prod_{i=1}^{q} P(X_i|Y)$ tersebut merupakan pengaruh semua fitur dari data terhadap setiap kelas Y.

Umumnya, Bayes mudah dihitung untuk fitur bertipe kategoris seperti kasus klasifikasi hewan dengan fitur “penutup kulit” dengan nilai {bulu, rambut, cangkang}, atau kasus fitur “jenis kelamin” dengan nilai {pria, wanita}. Namun untuk fitur dengan tipe numerik (kontinu). Ada perlakuan khusu sebelum dimasukkan dalam Naïve Bayes. Caranya adalah sebagai berikut :

1. Melakukan diskretisasi pada setiap fitur kontinu dan mengganti nilai fitur kontinu tersebut dengan nilai interval diskret. Pendekatan ini dilakukan dengan mentransformasi fitur kontinu dengan fitur ordinal.

2. Mengasumsikan bentuk tertentu dari distribusi probabilitas untuk fitur kontinu dan memperkirakan parameter distribusi dengan data pelatihan. Distribusi Gaussian biasanya dipilih untuk mempresentasikan probabilitas bersyarat dari fitur kontinu pada sebuah kelas $P(X_i|Y)$, sedangkan distribusi Gaussian dikarakteristikan dengan dua parameter mean, μ dan varian σ^2.

2.6.4 Karakteristik Naïve Bayes

Klasifikasi dengan Naïve Bayes bekerja berdasarkan teori probabilitas yang memandang semua fitur dari data sebagai bukti dalam probabilitas. Hal ini memberikan karakteristik Naïve Bayes adalah sebagai berikut :

1. Metode Naïve Bayes teguh terhadap data-data yang terisolasi yang biasanya merupkan data dengan karakteristik berbeda (*outlier*). Naïve Bayes juga bisa menangani nilai atribut yang salah dengan mengabaikan data latih selama proses pembangunan model dan prediksi.
2. Tangguh menghadapi atribut yang tidak relevan.
3. Atribut yang mempunyai korelasi bisa mendegradasi kinerja klasifikasi Naïve Bayes karena asumsi independensi atribut tersebut sudah tidak ada.

2.7 Kajian Penelitian Terkait

Jignesh Vania (Vania, et al., 2013) didalam penelitiannya menjelaskan beberapa teknik dalam deteksi Botnet, diantaranya adalah menggunakan *Honeypots* dan *Honeynet,* merujuk *intrusion detection system* seperti *signature based* dan *anomaly based*, berbasis DNS dan terakhir menggunakan teknik *Data Mining*.

Sebastian Garcia (Garcia, et al., 2014) Telah melakukan penelitian deteksi Botnet, dengan melakukan kajian tentang metode yang lebih baik dalam mendeteksi Botnet, memperbaiki algoritma, membuat dataset yang lebih baik dan membangun metodologi perbandingan.

Sudipta (Chowdhury, et al., 2017) menggunakan *dataset* CTU-13 untuk deteksi Botnet dengan berbasis *graph* dan dengan teknik fitur klasterisasi.

Pada penelitian lain adalah yang dilakukan oleh (Bijalwan, et al., 2016) menggunakan *Ensemble classifier algorithm*. Hasil penelitiannya menunjukkan

bahwa dengan menggunakan metode voting ketepatan klasifikasi berbasis *ensemble* meningkat hingga 96,41% dari 93,37%.

Pada tahun 2011 juga telah dilakukan penelitian tentang "*Detecting P2P Botnets through Network Behavior Analysis and Machine Learning"* (Saad, et al., 2011) yang fokus penilitan pada deteksi P2P Botnet dengan menggunakan 5 (lima) metode pada *machine learning* yang berbeda, dengan menggunakan dataset *Information security and object technology* (ISOT).

Duc Le (Le, et al., 2016) juga telah melakukan penelitian deteksi Botnet berbasis perilaku (*behavior based*) menggunakan *Self Organizing Map* (SOM) dengan teknik *unsupervised learning* sebagai analisis data.

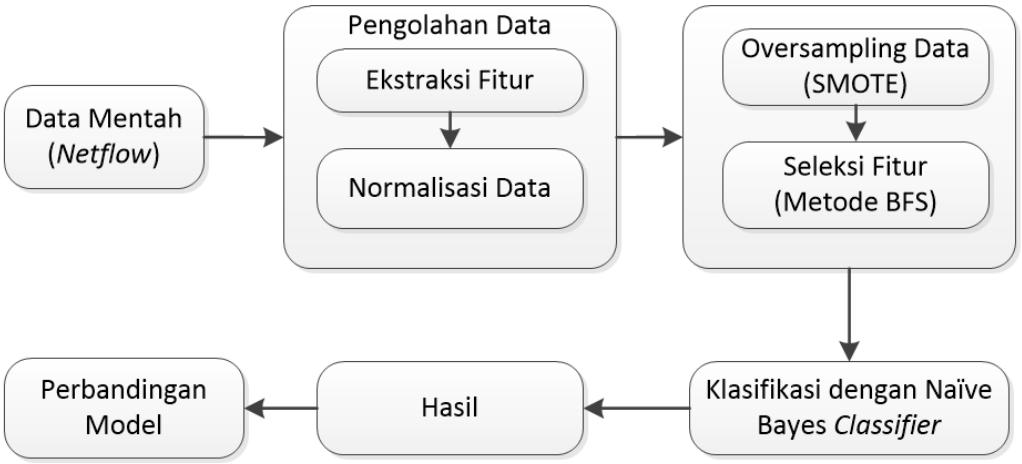
Ahmad Azab (Azab, et al., 2016) Penelitiannya tentang Botnet, di mana mencari kumpulan fitur lalu lintas jaringan menggunakan *Classification of Network Information Flow Analysis* (CONIFA) untuk menangkap komunikasi C&C dan *traffic* yang berbahaya.

Kannan (R. & A.V, 2014) Telah meneliti juga dengan deteksi Botnet berdasarkan *flow* dengan mengklasifikasikan paket *capture* menggunakan *decision tree*. Pemilihan atribut terutama didasarkan pada atribut paket dan tidak mempertimbangkan bagian data. Pendekatannya adalah untuk mendeteksi aktivitas Botnet tanpa melihat aliran jaringan yang lengkap tetapi hanya mengklasifikasikan perilaku berdasarkan interval waktu.

Halaman ini sengaja dikosongkan

BAB 3
METODOLOGI PENELITIAN

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 Naive Bayes *Classifier* dengan SMOTE dan Metode BFS. Berikut pada Gambar 3.1 tentang diagram penelitian.

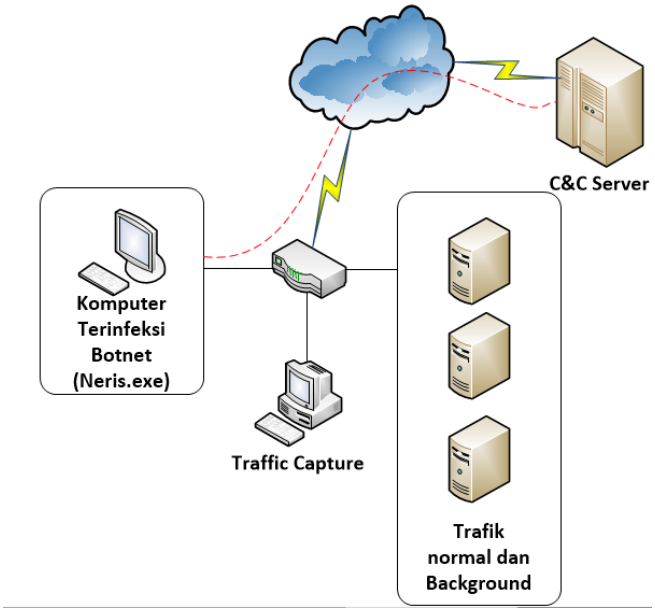


Gambar 3.1 Diagram Penelitian

3.1 Pengumpulan Data

Data mentah yang digunakan adalah dari dataset CTU-13. CTU-13 berisi 13 kelompok skenario yang di *capture* pada lingkungan jaringan aktual, *capture* meliputi Botnet, trafik normal dan trafik *background*. Trafik Botnet berasal dari komputer/*host* yang terinfeksi, trafik normal dan *background* dari *host* diseluruh kampus yang menggunakan infrastruktur *Czech Technical University* (CTU). Skenario di dataset CTU-13 dapat didefinisikan sebagai infeksi tertentu dari mesin virtual menggunakan Malware tertentu. Periode pengumpulan data untuk setiap skenario secara signifikan berbeda satu sama lain. Durasi data *NetFlow* yang direkam bervariasi yaitu antara 0,26 sampai 66,85 jam, dan selanjutnya jumlah data *NetFlow* juga bervariasi.

Pada Gambar 3.2 merupakan skenario *capture* dataset, dimana host yang terinfeksi bot dengan trafik normal dan background berada pada intranet jaringan kampus, sedangkan C&C server berada di internet. Proses *capture* berada pada perangkat *backbone*, karena penempatan system deteksi nantinya berada pada *backbone* infrastrukur kampus yang mendeteksi komunikasi bot dengan C&C server.



Gambar 3.2 Topologi *Capture Dataset*

Beberapa jenis bot yang sama ditemukan dalam skenario berbeda. Sebagian besar skenario hanya memiliki satu bot (skenario 1-8 dan 13), sedangkan skenario 9-12 memiliki beberapa bot di dalamnya. Persentase Botnet sangat kecil yaitu kurang dari 2%, dibandingkan dengan *NetFlow* secara keseluruhan total tiap-tiap skenario. Ciri khas lain dari dataset CTU-13 adalah bahwa setiap skenario pada *netflow* telah dianalisis dan diberi label secara manual. Tabel 3.1 memberikan ringkasan jumlah data pada skenario 1 yang berisi total paket, jumlah dan persentase Botnet, dan durasi *capture dataset*.

Tabel 3.1 Dataset CTU-13 Skenario 1 (Neris.exe)

Dataset	**Durasi (Jam)**	***NetFlows***	**Ukuran (GB)**	**Bot**	**Jumlah**	**Persentase Bot**
1	6.15	2,824,637	52	Neris	1	39933(1.41%)

Adapun dataset CTU-13 (Garcia, et al., 2014) dirancang dengan tujuan adalah sebagai berikut :

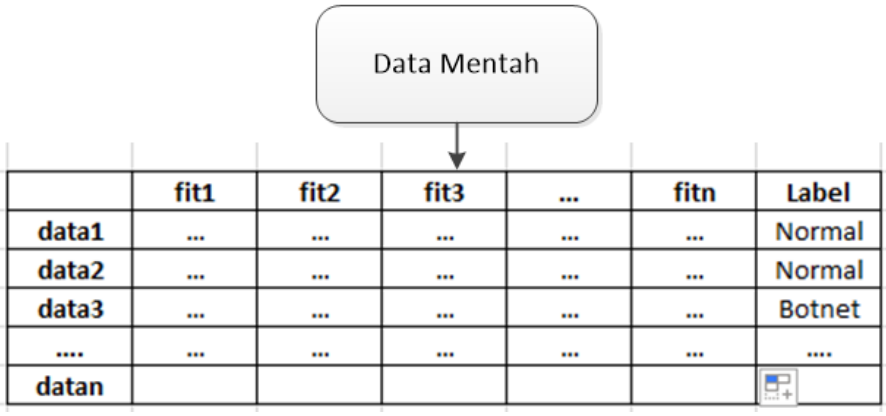
1. Dataset memiliki serangan Botnet yang sebenarnya, bukan simulasi.
2. Terdapat trafik aktual, artinya ketika *capture* trafik kondisi infrastruktur berjalan seperti biasanya, tidak dikondisikan seperti simulasi.
3. Memiliki dasar dalam pelabelan untuk pelatihan dan evaluasi pada metode yang dibahas di (Garcia, et al., 2014). Artinya *host* yang telah terinfeksi bot pada proses *capture* telah dilabeli sebagai trafik Botnet.
4. Pada skenario 1 CTU-13 terdapat detail perangkat pada topologi infrastruktur Gambar 3.2 yaitu :
    a. Komputer terinfeksi dengan *ip address* 145.32.84.165 dengan sistem operasi windows XP, untuk jumlah *flow* adalah 40961.
    b. Untuk komputer normal terdapat 6 *host* dengan *ip address* 147.32.84.170, 147.32.84.164, 147.32.84.134, 147.32.87.36, 147.32.80.9 dan 147.32.87.11 dengan aktifitas sebagai trafik normal, webserver, DNS server dan Matlab Server.

3.2 Pengolahan Data

Pada tahap ini akan dilakukan pengolahan data mentah ke dalam data final. Data mentah berupa data *flow* dari *netflow*. Pada tahap pengolahan data terdapat proses ekstraksi fitur, transformasi dan normalisasi data, membangkitkan data sintetis (SMOTE) dan proses seleksi fitur.

3.2.1 Ekstraksi Fitur

Adapun algoritma pengolahan data diawali dengan membaca file *dataset* (data mentah) dengan variabel X_{raw}. kemudian mengambil fitur perkolom dengan nama fitur *netflow*, nama-nama fitur terdapat pada Tabel 3.3. setelah terbaca fitur kemudian membaca data secara iterasi perbaris pada *dataset* untuk direkam dan disimpan sesuai fiturnya (X_{Fitur}). Keseluruhan data akan disimpan dalam variabel X_{data}. Secara lengkap algoritma pengolahan data mentah terdapat pada Tabel 3.2:



Gambar 3.3 Proses data mentah ke data final

Gambar 3.3 merupakan alur proses pengolahan data dari algoritma pengolahan data Tabel 3.2, dimana $X_{Fitur} = split(x_{fit1,}x_{fit2}, \dots \dots \dots x_{fitn})$ adalah fitur yang ada di *Netflow*, dan $X_{data} = split(X_{raw})$ yang berisi isi data *Netflow*.

Tabel 3.2 Algoritma Pengolahan Data

1. Proses diawali dengan membaca file dengan format .*binetflow*

   $$X_{raw} = [x_{data1}, x_{data2}, \dots \dots \dots \dots x_{datan}]$$

2. Mengambil nama fitur, dengan membaca baris pertama dari data mentah.

   $$X_{Fitur} = split(x_{fit1,}x_{fit2}, \dots \dots \dots x_{fitn})$$

3. Merekam data dan memasukkan sesuai fitur.

   $$X_{data} = split(X_{raw})$$

4. Jika fitur berupa huruf (*text*), harus ditransformasikan ke dalam numerik, agar dapat diproses ke tahap berikutnya.
5. Pembersihan data dari item data yang tidak diperlukan (*irrelevan data*), Berdasarkan item fitur *protocol*.
6. Menyimpan hasil ekstraksi keseluruhan data (fitur dan data) kedalam file dengan format .*csv*.

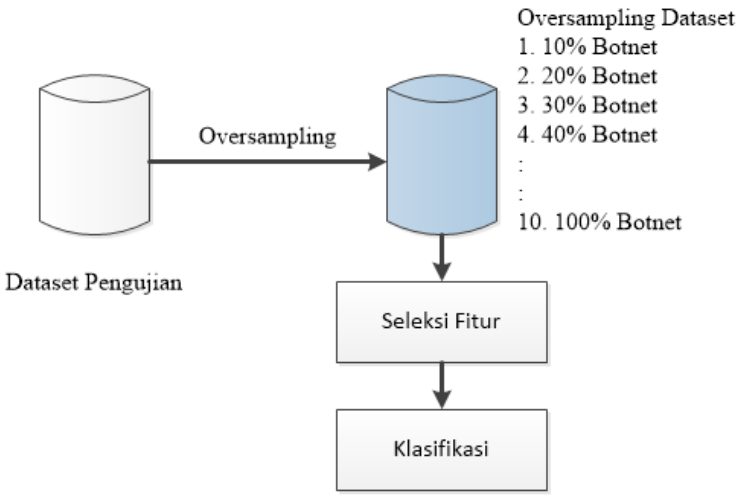
3.2.2 Normalisasi Data

Normalisasi data adalah tahap 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 ekstrasksi fitur, dimana ketika membaca, transformasi dan merekam data

didalamnnya terdapat proses pengecekan atribut. Jika atribut itu kosong maka data tidak dapat dilakukan deteksi, sehingga dihapus atau tidak masuk dalam data final untuk proses berikutnya.

3.2.3 Oversampling Data Minor dengan SMOTE

Pada tahap *oversampling data* adalah menggunakan teknik SMOTE, Gambar 3.4 merupakan 10 skenario membangkitkan data kelas minor, yaitu 10%, 20%, 30% sampai dengan 100% data buatan.



Gambar 3.4 Proses oversampling data SMOTE

Dengan langkah algoritma SMOTE pada tahap pengolahan data ini adalah sebagai berikut :

1. Menghitung jarak antar amatan pada kelas minor menggunakan rumus (1) VDM pada bagian 2.4 Kajian pustaka.
2. Menentukan nilai *k* yaitu 5 dan persentase oversampling sesuai skenario yaitu 10%, 20%, 30% sampai dengan 100% secara bertahap.
3. Dipilih satu contoh kelas minor secara acak.
4. Menentukan amatan *k* tetangga terdekat dengan mengurut jarak contoh terpilih dengan semua amatan pada kelas minor.
5. Data sintesis dibuat dengan menentukan nilai per peubah penjelasnya. Nilai tersebut diperoleh dari mayoritas nilai pada *k* tetangga terdekat. Jika semua peubah telah dibuat maka diperoleh satu amatan baru.

6. Langkah 3 hingga 5 dilakukan berulang hingga banyaknya *oversampling* yang diinginkan telah tercapai.

3.2.4 Seleksi Fitur dengan Metode Best First Search (BFS)

BFS dipakai untuk melakukan pencarian secara sistematis, dimana untuk mencapai tujuan dengan bergerak maju (*forward*) secara iteratif berdasarkan nilai heuristik terbaik untuk node yang berdekatan untuk setiap node saat ini.

Pada tahap seleksi fitur *netflow* ini, proses algoritma BFS diterapkan untuk mencari fitur terbaik dalam *netflow*, hasil pencarian fitur merupakan representasi fitur sebuah *flow* yang paling dominan, fitur tersebut dijadikan sebagai alat mengenali yang efektif dan efisian kedalam sebuah kelas. Tabel 3.3 merupakan keseluruhan fitur.

Dalam Seleksi atribut atau seleksi fitur, metode yang digunakan adalah *search method BFS*, dan untuk *attribute evaluator* menggunakan Naïve Bayes. Sehingga diharapkan mendapatkan luaran hasil seleksi fitur yang optimal.

Tabel 3.3 Fitur *Network Flow*

No	Fitur	Keterangan Fitur
1	Duration	Durasi dari *flow* data (*record last time – record start time*)
2	Protocol	Transaksi protokol yang digunakan
3	Source Address	Asal alamat IP
4	Source Port	Asal port
5	Destination Address	Tujuan alamat IP
6	Destination Port	Tujuan port
7	State	*Transaction state*
8	sTos	Nilai TOS *byte* asal
9	dTos	Nilai TOS *byte* tujuan
10	Total Packets	Keseluruhan transaksi dalam *packet*
11	Total Bytes	Keseluruhan transaksi dalam *bytes*
12	Source Bytes	Transaksi dalam *bytes* asal ke tujuan

3.3 Tahap Klasifikasi

Dataset yang telah melalui proses normalisasi dan telah ditinjau dalam bentuk lain, selanjutnya dilakukan proses klasifikasi menggunakan metode Naive Bayes dengan mengasumsikan bahwa tiap atribut variabel masing-masing data

bersifat bebas (*independence*). sehingga dihasilkan model klasifikasi sebagai metode untuk deteksi Botnet.

Dalam proses klasifikasi, skema yang digunakan untuk pencarian model alokasi terbaik adalah validasi silang sepuluh lipatan/ 10-*folds cross validation* pada dataset final setelah proses SMOTE dan seleksi fitur dengan BFS.

3.4 Skenario Pengujian

Pada penelitian ini dilakukan 21 (dua puluh satu) eksperimen untuk membandingkan dan melihat hasil model usulan yaitu deteksi Botnet menggunakan Naïve Bayes *classifier* dengan SMOTE dan metode BFS. Adapun tool untuk melakukan pengujian adalah menggunakan Weka (*Waikato Environtment for Knowledge Analysis*) dengan versi 3.6.13.

Pada tahap eksperimen pertama akan dilakukan pengujian data dengan menggunakan klasifikasi menggunakan model algoritma Naïve Bayes. Adapun prosesnya adalah sebagai berikut :

1. Membaca dataset final
2. Menentukan classifier, yang dipilih adalan Naïve Bayes *Classifier.*
3. Untuk opsi pengujian model (test option) yang dipilih adalah *cross validation* dengan *folds* 10.

Pada tahap eksperimen kedua akan menggunakan model Naïve Bayes *classifier* dengan oversampling data SMOTE **10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%** dan **100%** tanpa Seleksi Fitur. Adapun proses untuk tahap kedua ini adalah sebagai berikut :

1. Membaca dataset final
2. Pada *Filter* dipilih *resample*, kemudian untuk besaran data buatan pada *sampleSizePercent* dimasukkan angka sesuai skenario oversampling dari 10% - 100%.
3. Menentukan classifier, yang dipilih adalan Naïve Bayes *Classifier.*
4. Untuk opsi pengujian model (test option) yang dipilih adalah *cross validation* dengan *folds* 10.

Pada tahap eksperimen ketiga akan menggunakan Naïve Bayes *classifier* dengan oversampling data berurutan dengan besaran **10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%** dan **100%** dari data minor dan seleksi fitur menggunakan metode BFS. Adapun proses untuk tahap ketiga ini adalah sebagai berikut :

1. Membaca dataset final
2. Pada *Filter* dipilih *resample*, kemudian untuk besaran data buatan pada *sampleSizePercent* dimasukkan angka sesuai skenario oversampling dari 10% - 100%.
3. Pada menu *Select Attributes.* Untuk *Attribute Evaluator* menggunakan *ClassifierSubsetEval*, kemudian pada *classifier* dipilih menggunakan Naïve Bayes *classifier*. Pada *Search Method* dipilih *Best First Search*, kemudian *attribute selection mode* menggunakan keseluruhan dataset (*use full training set*).
4. Kemudian kembali ke menu Preproses, memilih fitur/atribut hasil seleksi fitur dengan metode BFS.
5. Menentukan classifier, yang dipilih adalan Naïve Bayes *Classifier*.
6. Untuk opsi pengujian model (test option) yang dipilih adalah *cross validation* dengan *folds* 10.

3.5 Evaluasi Model

Teknik Pengujian yang akan digunakan pada model klasifikasi yang terbentuk dari metode Naive Bayes, yaitu *Confusion matrix*. *Confusion matrix* merupakan metode yang menggunakan matrik sebagaimana pada Tabel 3.4. teknik evaluasi ini juga dilakukan untuk deteksi Botnet oleh (Garcia, et al., 2014).

Tabel 3.4 Confusion Matriks

Kelas Aktual	**Kelas Prediksi**	
	Prediksi = Yes	Prediksi = No
Aktual = Yes	TP	FN
Aktual = No	FP	TN

Jika data set hanya terdiri dari dua kelas, kelas yang satu dianggap sebagai positif dan yang lainnya negatif.

True Positif (TP) merupakan jumlah hasil pengujian positif yang diklasifikasikan **benar** oleh model *classifier*, false positif (FP) adalah jumlah hasil pengujian negatif yang diklasifikasikan **benar**. sedangkan *false negatives* (FN) jumlah hasil pengujian negatif yang diklasifikasikan **salah**, *true negatives* adalah jumlah hasil pengujian positif yang diklasifikasikan **salah** oleh model *classifier.*

Kemudian hasil pengujian dimasukan pada tabel, maka dapat dihitung nilai-nilai dari *sensitivity* (*recall*), *specificity*, presisi dan akurasi. Deteksi Botnet ini menghasilkan *binary class* oleh sebab itu digunakan perhitungan keakurasian menggunakan persamaan (6).

$$Akurasi = \frac{\sum TP + \sum TN}{\sum Total\_Populasi} \quad (6)$$

jadi akurasi merupakan perbandingan antara semua sampel yang *true positif* berbanding terhadap jumlah *true positif* dan *false positif*, sedangkan untuk perhitungan presisi dari sebuah kelas sebagaimana persamaan (7)

$$Presisi = \frac{\sum TP}{\sum TP + \sum FP} \quad (7)$$

untuk *recall* menggunakan persamaan (10) yaitu perbandingan antara *true positif* terhadap jumlah *true positif* dan *false negatif*

$$Recall = \frac{\sum TP}{\sum TP + \sum FN} \quad (8)$$

sehingga untuk $F_{measure}$ (F1-*Score*) dihitung berdasarkan persamaan (11), dimana merupakan salah satu perhitungan evaluasi dalam temu kembali informasi yang mengkombinasikan *recall* dan presisi. Nilai *recall* dan presisi pada suatu keadaan dapat memiliki bobot yang berbeda. Ukuran yang menampilkan timbal balik antara *recall* dan presisi adalah $F_{measure}$ yang merupakan bobot harmonik *mean* dari *recall* dan presisi.

$$F_{measure} = 2 * \frac{Presisi * Recall}{Presisi + Recall} \quad (9)$$

Pengukuran *respond time* terhadap performa deteksi juga dicatat, untuk mengetahui rata-rata kemampuan klasifikasi *naive bayes* terhadap data fitur dari Botnet.

Halaman ini sengaja dikosongkan

BAB 4
HASIL DAN PEMBAHASAN

Pada bab ini akan membahas mengenai hasil dari penelitian tentang model Deteksi Botnet, terdapat 3 (tiga) model deteksi yaitu deteksi Botnet menggunakan Naive Bayes *Classifier,* deteksi Botnet menggunakan Naive Bayes *Classifier* dengan SMOTE dan Deteksi Botnet menggunakan Naive Bayes Classifier dengan SMOTE dan Metode BFS.

4.1 Data Pengujian

Data untuk penelitian ini menggunakan *raw dataset Bidirectional netflow* CTU-13 yang dihasilkan oleh *Argus* (*Audit Record Generation and Utilization System*). Dataset Botnet tersebut telah tersedia dan di publikasikan oleh *Czech Technical University* (CTU). Pada *dataset* CTU-13 terdapat 13 sampel skenario, pada masing-masing skenario tersebut terdapat aksi *malware* dengan tindakan yang berbeda. Dalam penelitian ini *dataset* yang digunakan adalah dataset CTU-13 pada skenario 1. Seperti yang disajikan Pada Tabel 4.1 dengan rincian data diantaranya durasi pengambilan data, jumlah paket, jumlah *netflow*, jenis *bot* dan jumlah komputer yang terinfeksi *bot*.

Tabel 4.1 Dataset pengujian

Id	Durasi (jam)	Packets	Netflows	Jenis Bot	Jumlah Bots
1	6.15	71.971.482	2.824.637	Neris	1

Dari hasil dataset pengujian tersebut terdapat 2 jenis data yaitu dalam format *packet capture* (pcap) dan dalam format *netflow* (berbasis teks). Untuk penelitian ini data yang digunakan adalah data dalam format *netflow*. Dimana data *netflow* berisi dengan informasi karakteristik aliran data (*network flow*) dan bukan isi data (*payload*). D*ataset netflow* dalam data pengujian telah dilakukan analisa dan pelabelan. Pada Tabel 4.2 merupakan karakteristik *dataset* CTU-13 pada skenario 1. Dapat diketahui bahwa total data dalam bentuk baris *flow* yaitu sebanyak 2.824.636, dari total data itu terdapat data *flow* abnormal yaitu Botnet dan

C&C. Terdapat juga data *flow background* dan normal. Dalam tahap selanjutnya, data tersebut akan dikelompokkan menjadi 2 data yaitu Normal dan Botnet.

Tabel 4.2 Label data

Skenario	Total Flows	Botnet Flows	Normal Flows	C&C Flows	Background Flows
1	2.824.637	39.933	30.387	1.026	2.753.290

4.2 Tahap Pengolahan Data

Dari data mentah pada Tabel 4.2, maka akan dilakukan ekstraksi fitur dan normalisasai. Ekstraksi fitur merupakan suatu pengambilan ciri / *feature* dari suatu bentuk yang nantinya nilai yang didapatkan akan dapat analisis untuk proses selanjutnya. Kemudian proses normalisasi data, dimana data sampel yang ada tidak memiliki informasi lengkap atau *missing value* (nilai yang tidak lengkap atau tidak mempunyai arti), sehingga data yang tidak diperlukan tersebut akan dihapus atau tidak dipakai dalam proses berikutnya, maka akan dihasilkan *dataset* yang layak untuk dilakukan proses klasifikasi.

4.2.1 Hasil Ekstraksi Fitur

Data mentah *netflow* adalah berekstensi *.biargus* dan berbentuk teks, sehingga data diekstraksi dalam format *.csv* agar pada tahap berikutnya isi data dapat dikenali dan diproses. Pada masing-masing data *flow* terdapat 12 fitur yang di ekstraksi (*Duration, Protocol, Source Address, Source Port, Destination Address, Destination Port, State, sTos, dTos, Total Packets, Total Bytes, Source Bytes*) Gambar 4.1 merupakan contoh sampel data mentah *netflow*.

```
StartTime,Dur,Proto,SrcAddr,Sport,Dir,DstAddr,Dport,State,sTos,dTos,TotPkts,TotBytes,SrcBytes
2011/08/10 09:46:59.607825,1.026539,tcp,94.44.127.113,1577,    ->,147.32.84.59,6881,S_RA,0,0,4
2011/08/10 09:47:00.634364,1.009595,tcp,94.44.127.113,1577,    ->,147.32.84.59,6881,S_RA,0,0,4
2011/08/10 09:47:48.185538,3.056586,tcp,147.32.86.89,4768,    ->,77.75.73.33,80,SR_A,0,0,3,182
2011/08/10 09:47:48.230897,3.111769,tcp,147.32.86.89,4788,    ->,77.75.73.33,80,SR_A,0,0,3,182
2011/08/10 09:47:48.963351,3.083411,tcp,147.32.86.89,4850,    ->,77.75.73.33,80,SR_A,0,0,3,182
2011/08/10 09:47:58.806814,3.097288,tcp,147.32.86.89,4866,    ->,77.75.73.33,80,SR_A,0,0,3,182
2011/08/10 09:51:34.450457,1.048908,tcp,213.200.244.217,47908,    ->,147.32.84.59,6881,S_RA,0,
2011/08/10 09:54:55.231320,4.373526,tcp,75.105.28.60,1419,    ->,147.32.84.59,6881,S_RA,0,0,4,
2011/08/10 09:57:13.352114,4.827912,tcp,75.105.28.60,1491,    ->,147.32.84.59,6881,S_RA,0,0,4,
2011/08/10 09:58:43.301515,0.049697,tcp,178.111.79.115,41752,    ->,147.32.84.229,13363,SR_SA,
```

Gambar 4.1 Raw Data Network Flow

Dari Gambar 4.1 dapat dilihat beberapa fitur dengan dengan format alamat *ip* dan huruf. Agar semua fitur dapat dikenali untuk diproses selanjutnya maka fitur tersebut dilakukan proses transformasi data.

4.2.2 Transformasi Data

Fitur yang masih berupa huruf dan *ip address* perlu dilakukan transformasi data, sehingga data semula yang tidak dapat dikenali dirubah menjadi numerik. Hal ini perlu dilakukan agar data yang memiliki ciri sebuah fitur yang keseluruhan datanya telah menjadi numerik dapat di proses ke tahapan selanjutnya. Adapaun data yang harus ditransformasikan adalah *source address, destination address, protocol*, dan *state*.

1) Transformasi Data Huruf

Terdapat 2 (dua) buah fitur data yang berbentuk huruf yaitu fitur *protocol* dan *state*. Fitur *protocol* berisi informasi jenis transaksi yang digunakan, pada Tabel 4.3 merupakan kamus pemetaan jenis *protocol* yang di transformasikan ke dalam numerik, yang terdapat 15 macam jenisnya.

Tabel 4.3 Hasil pemetaan fitur protocol ke numerik

Protocol Dictionary			
Protokol	**Nilai**	**Protokol**	**Nilai**
arp	5	rarp	14
unas	13	ipv6-icmp	9
udp	1	rtp	2
rtcp	7	ipv6	10
pim	3	ipx/spx	6
udt	11	icmp	4
esp	12	igmp	8
tcp	0		

Fitur *state* memiliki informasi status transaksi pada aliran data, terdapat 232 jenis status dalam bentuk huruf (Anon., 2014). Tabel 4.4 merupakan kamus hasil pemetaan jenis *state* yang akan di transformasikan ke dalam kode numerik yang telah ditentukan.

Tabel 4.4 Pemeteaan State ke numerik

No	Status	Kode	No	Status	Kode	No	Status	Kode
1	‘ ‘	1	36	RPAC_PA	1	73	PAC_PA	5
2	FSR_SA	30	37	FRPA_R	1	74	SRPA_SPA	9646
3	_FSA	296	38	SPA_SPA	2989	75	SRPA_FSRA	13
4	FSRPA_FSA	77	39	PA_RA	3	76	FPA_FRPA	49
5	SPA_SA	31	40	SPA_SRPA	4185	77	SRA_SPA	10
6	FSA_SRA	1181	41	RA_FA	8	78	SA_SRA	838
7	FPA_R	46	42	FSPAC_SRPA	1	79	PA_PA	5979
8	SPAC_SPA	37	43	SPA_FSA	1	80	FPA_RPA	27
9	FPAC_FPA	2	44	FPA_FSRPA	3	81	SR_RA	10
10	_R	1	45	SRPA_FSA	379	82	RED	4579
11	FPA_FPA	784	46	FPA_FRA	7	83	CON	2190507
12	FPA_FA	66	47	S_SRA	81	84	FSRPA_FSPA	13547
13	_FSRPA	1	48	FSA_SA	6	85	FSPA_FPA	4
14	URFIL	431	49	State	1	86	FAU_R	2
15	FRPA_PA	5	50	SRA_SRA	38	87	ECO	2877
16	_RA	2	51	S_FA	2	88	FRPA_FPA	72
15	SA_A	2	52	FSRPAC_SPA	7	89	FSAU_SRA	1
16	SA_RA	125	53	SRPA_FSPA	35460	90	FRA_FA	8
17	FA_FPA	17	54	FPA_A	1	91	FSPA_FSPA	216341
18	FA_RA	14	55	FSA_FPA	3	92	SEC_RA	19
19	PA_FPA	48	56	FRPA_RA	1	93	ECR	3316
20	URHPRO	380	57	FSAU_SA	1	94	SPAC_FSPA	12
21	FSRPA_SRA	8	58	FSPA_FSRPA	10560	95	SR_A	34
22	R_	541	59	SA_FSA	358	96	SEC_	5
23	DCE	5	60	FA_FRA	8	97	FSAU_FSRA	3
24	SA_R	1674	61	FSRPA_SPA	2807	98	FSRA_FSRPA	11
25	SA_	4295	62	FSRPA_FSRA	32	99	SRC	13
26	RPA_FSPA	4	63	FRA_FPA	6	100	A_RPA	1
27	FA_A	17	64	FSRA_FSRA	3	101	FRA_PA	3
28	FSPA_FSPAC	7	65	SPAC_FSRPA	1	102	A_RPE	1
29	RA_	2230	66	FS_	40	103	RPA_FRPA	20
30	FSRPA_SA	255	67	FSPA_FSRA	798	104	_SRA	74
31	NNS	47	68	FSAU_FSA	13	105	SRA_FSPA	293
32	SRPA_FSPAC	1	69	A_R	36	106	FPA_	118
33	RPA_FPA	42	70	FSRPAE_FSPA	1	107	FSRPAC_FSRPA	2
34	FRA_R	10	71	SA_FSRA	4	108	_FA	1
35	FSPAC_FSPA	86	72	PA_PAC	3	109	DNP	1

Tabel 4.5 Lanjutan Tabel 4.4

No	Status	Kode	No	Status	Kode	No	Status	Kode
109	RPA_R	3	130	FSA_FSRA	279	151	FSRPA_FSRPA	379
110	_FPA	5	131	A_A	68	152	FSRA_SRA	14
111	SREC_SA	1	132	REQ	892	153	_FRPA	1
112	URN	339	133	FA_R	124	154	SR_	59
113	URO	6	134	FSRPA_SRPA	97	155	FSPA_SPA	517
114	URH	3593	135	FSPAC_FSRA	20	156	FRPA_FSPA	1
115	MRQ	4	136	FRPA_RPA	7	157	PA_A	159
116	SR_FSA	1	137	FSRA_SPA	8	158	PA_SRA	1
117	SPA_SRPAC	1	138	INT	85813	159	FPA_RA	5
118	URP	23598	139	FRPA_FRPA	6	160	S_	68710
119	RPA_A	1	140	SRPAC_FSPA	4	161	SA_FSRPA	4
120	FRA_	351	141	SPA_SRA	808	162	FSA_FSRPA	1
121	FSPA_SRA	91	142	SA_SRPA	1	163	SA_SPA	4
122	FSA_FSA	26138	143	SPA_FSPA	2118	164	RA_A	5
123	PA_	149	144	FSRAU_FSA	2	165	_SRPA	9
124	FSRA_FSPA	798	145	RPA_PA	171	166	S_FRA	156
125	FSPAC_FSA	11	146	_SPA	268	167	FA_FRPA	1
126	SRPA_SRPA	176	147	A_PA	47	168	PA_R	72
127	SA_SA	33	148	SPA_FSRA	416	169	FSRPAEC_FSPA	1
128	FSPAC_SPA	1	149	FSPA_FSRPAC	2	170	_PA	7
129	SRA_RA	78	150	FSRPAC_FSPA	139	171	RA_S	1

2) Transformasi Data Alamat IP

Dalam penelitian ini untuk alamat asal (*source address*) dan alamat tujuan (*destination address*) dari format yang memiliki titik (contoh: 123.45.67.89) ditransformasikan kedalam 32-*Bit Binary* dan kemudian ditransformasikan lagi ke dalam bentuk desimal. Tabel 4.7 merupakan tabel transformasi *ip address* ke desimal. Adapun *code* untuk transformasi ip address ke dalam desimal di *python* adalah seperti pada Tabel 4.6.

Tabel 4.6 Kode transformasi ip addresss

```
Sip = socket.inet_aton(Sip)
Sip = struct.unpack("!L", Sip)
Dip = socket.inet_aton(Dip)
Dip = struct.unpack("!L", Dip)
```

Tabel 4.7 Hasil transformasi data alamat IP ke Desimal

IP Address	32-Bit Binary	Desimal
147.32.84.165	10010011001000000101010010100101	2468369573
147.32.84.191	10010011001000000101010010111111	2468369599
147.32.84.192	10010011001000000101010011000000	2468369600
147.32.84.193	10010011001000000101010011000001	2468369601
147.32.84.204	10010011001000000101010011001100	2468369612
147.32.84.205	10010011001000000101010011001101	2468369613
147.32.84.206	10010011001000000101010011001110	2468369614
147.32.84.207	10010011001000000101010011001111	2468369615
147.32.84.208	10010011001000000101010011010000	2468369616
147.32.84.209	10010011001000000101010011010001	2468369617

4.2.3 Hasil Normalisasi Data

Data *cleaning* dilakukan untuk menghilangkan data yang kosong atau data dengan nilai atribut yang tidak memenuhi syarat dan tidak dapat digunakan digunakan dalam proses atau tahap berikutnya. Adapun fitur yang dilakukan pengecekan data yang kosong adalah *source port (Sport)* dan *destination port (Dport)* dengan iterasi sesuai jumlah set data. Sehingga didapatkan *dataset* yang telah dibersihkan. Tabel 4.8 merupakan tabel berisi informasi hasil jumlah data yang belum normalisasi dan sudah di normalisasi.

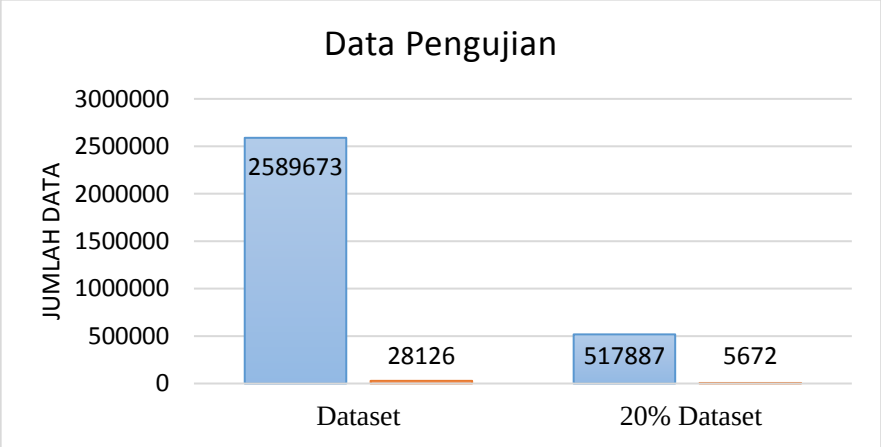
Tabel 4.8 Set data yang telah dibersihkan

Skenario Data	Total Flow Data	Normal	Botnet	Data Berkurang	Data Bersih
Skenario 1	2.824.636	2.783.677	40.959	206.837	2.617.799

Dari data final sejumlah 2.617.799 akan diambil 20% berimbang antar kuantitas kelas data dari data tersebut menggunakan Weka sebagai data pengujian. Tabel 4.9 dan Gambar 4.2 hasil *resample* 20% menggunakan Weka dengan jumlah dan persentase data normal 517.887 (98,92%), data Botnet 5.672(1,09%).

Tabel 4.9 Data Eksperimen

Nama	Normal	Botnet
Data Final	2.589.673(98,93)	28.126 (1,08)
20 % Data Pengujian	517.887(98,92%)	5.672 (1,09%)



Gambar 4.2 Grafik perbandingan jumlah dataset keseluruhan dengan pengambilan 20% dataset.

4.2.4 Hasil Pengolahan Data

Adapun sampel hasil dari pengolahan data *netflow* adalah dalam format *.csv*, seperti pada Tabel 4.10 dan Tabel 4.11 *dataset* tersebut selanjutnya dipakai untuk data latih dan data uji model deteksi Botnet menggunakan Naïve Bayes *classifier* dengan SMOTE dan metode BFS.

Tabel 4.10 Hasil Pengolahan data

Duration	Protocol	Src_Port	Dst_Port	Src_IP	Dst_IP
0	0	6881	1904	2468369467	3161474632
0.939583	0	42572	6881	1442330664	2468369467
0.99076	0	42650	6881	1442330664	2468369467
2.716508	0	2242	13363	1296007107	2468369637
2.775121	0	2247	80	1296007107	2468369637
2.568064	0	1041	2012	2468369573	1019141963
0.559377	1	1025	53	2468369573	2468368393
2.807242	0	1050	81	2468369573	1033069491
0.829001	0	1051	80	2468369573	1581225492

Tabel 4.11 Tabel lanjutan dari tabel 4.10

Tot_Packet	Tot_Byte	SrcByte	sTos	dTos	State	Label
1	60	60	0	0	2230	Normal
4	244	124	0	0	61105	Normal
4	244	124	0	0	61105	Normal
3	184	122	0	0	3119	Normal
3	184	122	0	0	3119	Normal
72	60278	1897	0	0	216341	Botnet
2	218	77	0	0	2190507	Botnet
48	37425	1441	0	0	9646	Botnet
19	10327	837	0	0	216341	Botnet

4.3 Pengujian Model

Dataset dengan 12 fitur dari hasil pengolahan pada tabel 4.10 adalah sebagai data uji klasifikasi dengan 3 model. Model pertama menggunakan klasifikasi menggunakan Naïve Bayes *classifier* kemudian melakukan evaluasi. Model kedua menggunakan Naïve Bayes classifier dengan SMOTE dan Model ketiga menggunakan Naïve Bayes classifier dengan SMOTE dan metode BFS sebagai seleksi fitur, pada model pertama hanya dengan 1 skenario dan model kedua dan ketiga akan dibagi kedalam 10 skenario pengujian data.

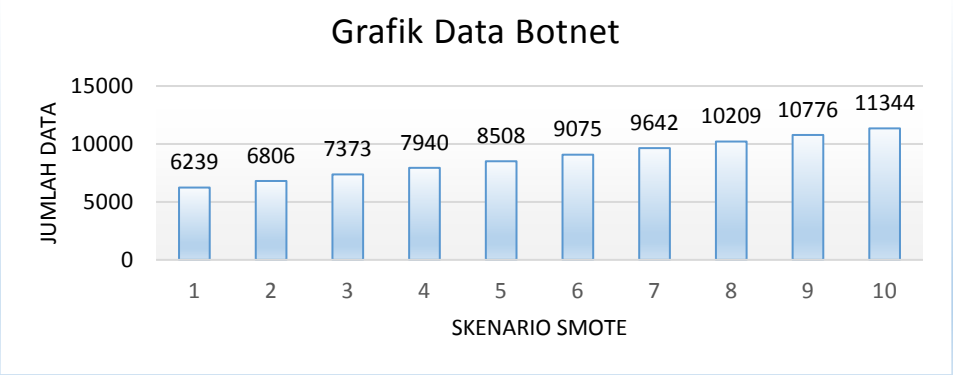
Pada Tabel 4.12 dataset yang akan digunakan untuk pengujian, dataset tersebut hasil dari generate SMOTE dengan 10 skenario berdasarkan persentase besaran data untuk *oversampling*. Pada Tabel 4.13 keseluruhan fitur yang di ujikan untuk evaluasi seleksi fitur.

Pada Tabel 4.12 dapat dilihat data awal adalah 517.887 *flow* normal dan 5.672 *flow* Botnet. SMOTE hanya dilakukan pada data minor, dalam hal ini kelas yang akan dibangkitkan data buatan dengan SMOTE adalah kelas “Botnet”.

Tabel 4.12 Dataset Hasil SMOTE

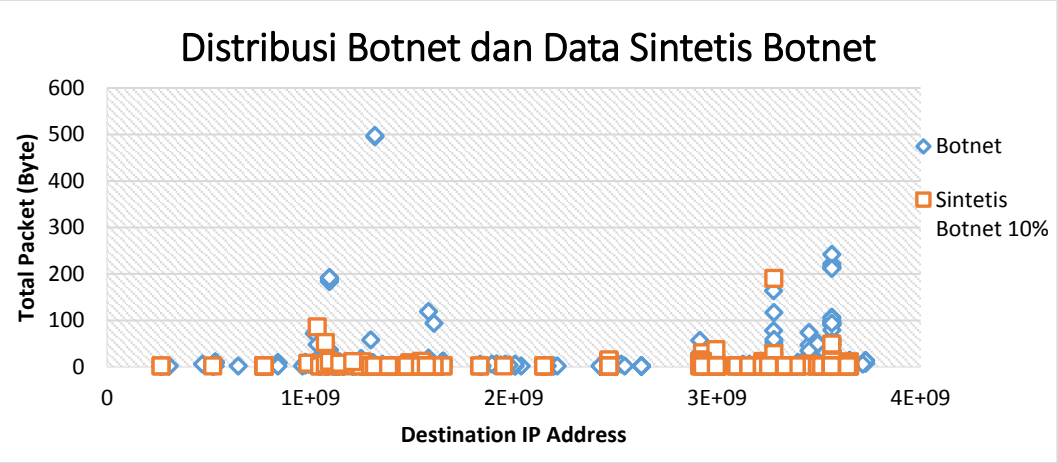
Skenario SMOTE		Jumlah Data (*Flows*)		Total Dataset
		Normal	Botnet	
1	Naïve Bayes (NB)	517.887 (98,92%)	5.672 (1,08%)	523.559
2	SMOTE (10%)	517.887 (98,81%)	6.239 (1,19%)	524.126
3	SMOTE (20%)	517.887 (98,70%)	6.806 (1,30%)	524.693
4	SMOTE (30%)	517.887 (98.60%)	7.373 (1,40%)	525.260
5	SMOTE (40%)	517.887 (98,49%)	7.940 (1,51%)	525.827
6	SMOTE (50%)	517.887 (98,38%)	8.508 (1,62%)	526.395
7	SMOTE (60%)	517.887 (98,28%)	9.075 (1,72%)	526.962
8	SMOTE (70%)	517.887 (98,17%)	9.642 (1,83%)	527.529
9	SMOTE (80%)	517.887 (98,07%)	10.209 (1,93%)	528.096
10	SMOTE (90%)	517.887 (97,96%)	10.776 (2,04%)	528.663
11	SMOTE (100%)	517.887 (97,86%)	11.344 (2,14%)	529.231

Skenario 1 SMOTE 10% pada Tabel 4.12 dapat meningkatkan kelas Botnet menjadi 6239, naik 567 *flow*. Grafik peningkatan kelas data buatan Botnet dapat dilihat pada Gambar 4.3 dimana dari skenario ke skenario berikutnya data kelas minor menjadi meningkat jumlahnya.



Gambar 4.3 Grafik peningkatan kelas dengan tiap Skenario SMOTE

Distribusi data untuk data sintetis Botnet, sebagaimana dilakukan menggunakan algoritma SMOTE dapat di lihat pada Gambar 4.4, data yang diambil adalah fitur destination ip address dan total packet dengan berdasarkan kelompok data Botnet dan data sintetis 10% data Botnet. Dari hasil sebaran data sintetis pada Gambar 4.4 dapat dilihat data sintetis memberikan kedekatan antar *instance/* vector.



Gambar 4.4 Distribusi Data Botnet

Tahap seleksi fitur dari seluruh fitur pada Tabel 4.13 menggunakan BFS adalah dengan *tool* Weka. Pilih menu *Select Attributes*, untuk *Attribute Evaluator* menggunakan *ClassifierSubsetEval*, kemudian pada *classifier* dipilih menggunakan Naïve Bayes *classifier*. Pada *Search Method* dipilih *Best First Search*, kemudian *attribute selection mode* menggunakan keseluruhan dataset (*use full training set*).

Tabel 4.13 Keseluruhan fitur yang di ujikan untuk evaluasi seleksi fitur

Fitur	**Kode**
Duration	1
Protocol	2
Source Address	3
Source Port	4
Dest. Address	5
Dest. Port	6
State	7
sTos	8
dTos	9
Total Packets	10
Total Bytes	11
Source Bytes	12

Tabel 4.14 Hasil Seleksi Fitur dengan Metode BFS

Skenario		Hasil Seleksi Fitur dengan Metode BFS
1	Naïve Bayes (NB)	0
2	SMOTE (10%)	2,3,4,6
3	SMOTE (20%)	2,3,4,6
4	SMOTE (30%)	2,3,4,6
5	SMOTE (40%)	2,3,4,6
6	SMOTE (50%)	2,3,4,6
7	SMOTE (60%)	2,3,4,6
8	SMOTE (70%)	2,3,4,6
9	SMOTE (80%)	2,3,4,6
10	SMOTE (90%)	2,3,4,6
11	SMOTE (100%)	2,3,4,6

Langkah seleksi fitur dilakukan pada tiap-tiap skenario setelah proses SMOTE dijalankan. Tabel 4.14 menampilkan hasil dari seleksi fitur menggunakan metode BFS untuk dataset dengan SMOTE dari 10% sampai dengan 100%.

Dari keseluruhan tahap seleksi fitur pada tiap skenario didapatkan bahwa hasil fitur yang terpilih tetap yaitu fitur 2,3,4, dan 6. Dimana fitur tersebut adalah *protocol, Src_address, Dst_port, dan Dst_IP*. Selanjutnya digunakan dengan Naïve Bayes *classifier*. Dari skenario 2 sampai dengan skenario 11 mendapatkan hasil seleksi fitur sama yaitu {2,3,4 dan 6}.

Dalam skenario pengujian dilakungan penghitungan nilai akurasi, presisi, *recall* dan $F_{measure}$. Dengan persamaan (10)(11)(12)(13) dimana *i* adalah skenario ujicoba :

$$Akurasi_i = \frac{\sum TP_i + \sum TN_i}{\sum Total\_Populasi_i} \quad (10)$$

$$Presisi_i = \frac{\sum TP_i}{\sum TP_i + \sum FP_i} \quad (11)$$

$$Recall_i = \frac{\sum TP_i}{\sum TP_i + \sum FN_i} \quad (12)$$

$$F_{Measure_i} = 2 * \frac{Presisi_i * Recall_i}{Presisi_i + Recall_i} \quad (13)$$

Pada Bab 3.4 metodologi penelitian telah dijelaskan beberapa skenario yang dilakukan untuk pengujian, pengujian 21 skenario tersebut dibagi menjadi 3 model adalah sebagai berikut :

4.3.1 Pengujian Model Naïve Bayes (NB) *Classifier*

Pada tahap eksperimen ini dilakukan pengujian data dengan menggunakan klasifikasi menggunakan model algoritma Naïve Bayes *classifier*, kemudian dengan mengevaluasi klasifikasi secara *cross-validation* terhadap 10 *fold* subset.

Tabel 4.15 Confusion matrix dari cross-validation 10 fold Skenario 1

Confusion matrix		***Kelas Prediksi***	
		a	***b***
Kelas Aktual	**a = Normal**	500608	17279
	b = Botnet	78	5594

Berdasarkan *confusion matrix* pada Tabel 4.15. dapat dihitung tingkat akurasi, presisi dan *recall*/*sensitivity* untuk hasil *cross-validation 10 folds*, sebagai berikut :

$$Akurasi = \frac{(500608) + (5594)}{523559} x100\% = 96{,}68\%$$

$$Presisi(kelas\ a) = \frac{500608}{500608 + (78)} x100\% = 99{,}98\%$$

$$Presisi(kelas\ b) = \frac{5594}{5594 + (17279)} x100\% = 24{,}45\%$$

Maka rata-rata presisi dari dataset dari semua kelas adalah :

$$\bar{X}_{Presisi} = \frac{99{,}98 + 24{,}45}{2} = 62.2206\%$$

Tingkat *recall* selanjutnya juga dihitung, untuk menemukan nilai $F_{measure}$, sebagai berikut :

$$Recall\ (kelas\ a) = \frac{500608}{500608 + 17279} x100\% = 96{,}67\%$$

$$Recall\ (kelas\ b) = \frac{5594}{5594 + 78} x100\% = 98{,}63\%$$

Maka rata-rata *recall* dari semua kelas dapat dihitung menjadi sebagai berikut :

$$\bar{X}_{Recall} = \frac{96{,}67 + 98{,}63}{2} = 97{,}64\%$$

Sehingga selanjutnya dapat dihitung $F_{measure}$ dari hasil klasifikasi pada skenario 1 sebagai berikut :

$$F_{measure} = 2 * \left(\frac{62.2206 * 97{,}64}{62.2206 + 97{,}64}\right) = 76{,}00646\%$$

4.3.2 Pengujian Model NB dengan SMOTE

Pada tahap eksperimen kedua ini menggunakan model Naïve Bayes *classifier* dengan *oversampling data* SMOTE **10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%** dan **100%** tanpa Seleksi Fitur.

1. **Pengujian dengan Skenario 2**

Tabel 4.16 adalah merupakan hasil dari *cross-validation* terhadap 10 *fold* subset pada skenario 2 dengan SMOTE 10%.

Tabel 4.16 Confusion matrix dari cross-validation 10 fold Skenario 2

Confusion matrix		***Kelas Prediksi***	
		a	***b***
Kelas Aktual	**a = Normal**	500752	17135
	b = Botnet	94	6145

Berdasarkan *confusion matrix* pada Tabel 4.16 dapat dihitung tingkat akurasi, presisi dan *recall*/sensitivity untuk hasil *cross-validation 10 folds*, sebagai berikut :

$$Akurasi = \frac{(500752) + (6145)}{524126} x100\% = 96{,}71\%$$

$$Presisi(kelas\ a) = \frac{500752}{500752 + (94)} x100\% = 99{,}98\%$$

$$Presisi(kelas\ b) = \frac{6145}{6145 + (17135)} x100\% = 26{,}39\%$$

Maka rata-rata presisi dari dataset dari semua kelas adalah :

$$\bar{X}_{Presisi} = \frac{99{,}98 + 26{,}39}{2} = 63.19\%$$

Tingkat *recall* selanjutnya juga dihitung, untuk menemukan nilai F*measure*, sebagai berikut :

$$Recall\ (kelas\ a) = \frac{500752}{500752 + 17135} x100\% = 96{,}69\%$$

$$Recall\ (kelas\ b) = \frac{6145}{6145 + 94} x100\% = 98{,}49\%$$

Maka rata-rata *recall* dari semua kelas dapat dihitung menjadi sebagai berikut :

$$\bar{X}_{Recall} = \frac{96{,}69 + 98{,}49}{2} = 97{,}59\%$$

Sehingga selanjutnya dapat dihitung F*measure* dari hasil klasifikasi pada skenario 2 sebagai berikut :

$$F_{measure} = 2 * \left(\frac{63{,}19 * 97{,}59}{63{,}19 + 97{,}59}\right) = 76{,}71\%$$

2. Pengujian dengan Skenario 3

Tabel 4.17 adalah merupakan hasil dari *cross-validation* terhadap 10 *fold* subset pada skenario 3 dengan SMOTE 20%.

Tabel 4.17 Confusion matrix dari cross-validation 10 fold Skenario 3

Confusion matrix		***Kelas Prediksi***	
		a	***b***
Kelas Aktual	**a = Normal**	500851	17036
	b = Botnet	113	6693

Berdasarkan *confusion matrix* pada Tabel 4.17 dapat dihitung tingkat akurasi, presisi dan *recall*/*sensitivity* untuk hasil *cross-validation 10 folds*, sebagai berikut :

$$Akurasi\ = \frac{(500851) + (6693)}{524693} x100\% = 96{,}73\%$$

$$Presisi(kelas\ a) = \frac{500851}{500851 + (113)} x100\% = 99{,}97\%$$

$$Presisi(kelas\ b) = \frac{6693}{6693 + (17036)} x100\% = 20{,}21\%$$

Maka rata-rata presisi dari dataset dari semua kelas adalah :

$$\bar{X}_{Presisi} = \frac{99{,}97 + 20{,}21}{2} = 64.09\%$$

Tingkat *recall* selanjutnya juga dihitung, untuk menemukan nilai F*measure*, sebagai berikut :

$$Recall\ (kelas\ a) = \frac{500851}{500851 + 17036} x100\% = 96{,}71\%$$

$$Recall\ (kelas\ b) = \frac{6693}{6693 + 113} x100\% = 98{,}33\%$$

Maka rata-rata *recall* dari semua kelas dapat dihitung menjadi sebagai berikut :

$$\bar{X}_{Recall} = \frac{96{,}71 + 98{,}33}{2} = 97{,}52\%$$

Sehingga selanjutnya dapat dihitung F*measure* dari hasil klasifikasi pada skenario 3 sebagai berikut :

$$F\_measure = \ 2 * ((64.09 * 97{,}52)/(64{,}09 + 97{,}52)) = 77{,}35\%$$

3. Pengujian dengan Skenario 4

Tabel 4.18 adalah merupakan hasil dari *cross-validation* terhadap 10 *fold* subset pada skenario 4 dengan SMOTE 30%.

Tabel 4.18 Confusion matrix dari cross-validation 10 fold Skenario 4

Confusion matrix		***Kelas Prediksi***	
		a	***b***
Kelas Aktual	**a = Normal**	501234	16653
	b = Botnet	129	7244

Berdasarkan *confusion matrix* pada Tabel 4.18 dapat dihitung tingkat akurasi, presisi dan *recall*/*sensitivity* untuk hasil *cross-validation 10 folds*, sebagai berikut :

$$Akurasi\ = \frac{(501234) + (7244)}{525260} x100\% = \ 96{,}81\%$$

$$Presisi(kelas\ a) = \frac{501234}{501234 + (129)} x100\% = 99{,}97\%$$

$$Presisi(kelas\ b) = \frac{7244}{7244 + (16653)} x100\% = 30{,}31\%$$

Maka rata-rata presisi dari dataset dari semua kelas adalah :

$$\bar{X}_{Presisi} = \frac{99{,}97 + 30{,}31}{2} = 65.14\%$$

Tingkat *recall* selanjutnya juga dihitung, untuk menemukan nilai F*measure*, sebagai berikut :

$$Recall\ (kelas\ a) = \frac{501234}{501234 + 16653} x100\% = 96{,}78\%$$

$$Recall\ (kelas\ b) = \frac{7244}{7244 + 129} x100\% = 98{,}25\%$$

Maka rata-rata *recall* dari semua kelas dapat dihitung menjadi sebagai berikut :

$$\bar{X}_{Recall} = \frac{96{,}78 + 98{,}25}{2} = 97{,}51\%$$

Sehingga selanjutnya dapat dihitung *F*$_{measure}$ dari hasil klasifikasi pada skenario 4 sebagai berikut :

$$F_{measure} = 2 * \left(\frac{65{,}14 * 97{,}51}{65{,}14 + 97{,}51}\right) = 78{,}10\%$$

4. Pengujian dengan Skenario 5

Tabel 4.19 adalah merupakan hasil dari *cross-validation* terhadap 10 *fold* subset pada skenario 5 dengan SMOTE 40%.

Tabel 4.19 Confusion matrix dari cross-validation 10 fold Skenario 5

Confusion matrix		***Kelas Prediksi***	
		a	***b***
Kelas Aktual	**a = Normal**	501536	16351
	b = Botnet	152	7788

Berdasarkan *confusion matrix* pada Tabel 4.19 dapat dihitung tingkat akurasi, presisi dan *recall*/*sensitivity* untuk hasil *cross-validation 10 folds*, sebagai berikut :

$$Akurasi\ = \frac{(501536) + (7788)}{525827} x100\% = 96{,}86\%$$

$$Presisi(kelas\ a) = \frac{501536}{501536 + (152)} x100\% = 99{,}96\%$$

$$Presisi(kelas\ b) = \frac{7788}{7788 + (16351)} x100\% = 32{,}26\%$$

Maka rata-rata presisi dari dataset dari semua kelas adalah :

$$\bar{X}_{Presisi} = \frac{99{,}96 + 32{,}26}{2} = 66{,}11\%$$

Tingkat *recall* selanjutnya juga dihitung, untuk menemukan nilai $F_{measure}$, sebagai berikut :

$$Recall\ (kelas\ a) = \frac{501536}{501536 + 16351} x100\% = 96{,}84\%$$

$$Recall\ (kelas\ b) = \frac{7788}{7788 + 152} x100\% = 98{,}08\%$$

Maka rata-rata *recall* dari semua kelas dapat dihitung menjadi sebagai berikut :

$$\bar{X}_{Recall} = \frac{96{,}84 + 98{,}08}{2} = 97{,}46\%$$

Sehingga selanjutnya dapat dihitung $F_{measure}$ dari hasil klasifikasi pada skenario 5 sebagai berikut :

$$F_{measure} = 2 * \left(\frac{66{,}11 * 97{,}46}{66{,}11 + 97{,}46}\right) = 78{,}78\%$$

5. Pengujian dengan Skenario 6

Tabel 4.20 adalah merupakan hasil dari *cross-validation* terhadap 10 *fold* subset pada skenario 6 dengan SMOTE 50%.

Tabel 4.20 Confusion matrix dari cross-validation 10 fold Skenario 6

Confusion matrix		***Kelas Prediksi***	
		a	***b***
Kelas Aktual	**a = Normal**	501622	16265
	b = Botnet	155	8353

Berdasarkan *confusion matrix* pada Tabel 4.20 dapat dihitung tingkat akurasi, presisi dan *recall*/*sensitivity* untuk hasil *cross-validation 10 folds*, sebagai berikut :

$$Akurasi\ = \frac{(501622) + (8353)}{526395} x100\% = 96{,}88\%$$

$$Presisi(kelas\ a) = \frac{501622}{501622 + (155)} x100\% = 99{,}96\%$$

$$Presisi(kelas\ b) = \frac{8353}{8353 + (16265)} x100\% = 33{,}93\%$$

Maka rata-rata presisi dari dataset dari semua kelas adalah :

$$\bar{X}_{Presisi} = \frac{99{,}96 + 33{,}93}{2} = 66{,}94\%$$

Tingkat *recall* selanjutnya juga dihitung, untuk menemukan nilai $F_{measure}$, sebagai berikut :

$$Recall\ (kelas\ a) = \frac{501622}{501622 + 16265} x100\% = 96{,}85\%$$

$$Recall\ (kelas\ b) = \frac{8353}{8353 + 155} x100\% = 98{,}17\%$$

Maka rata-rata *recall* dari semua kelas dapat dihitung menjadi sebagai berikut :

$$\bar{X}_{Recall} = \frac{96{,}85 + 98{,}17}{2} = 97{,}51\%$$

Sehingga selanjutnya dapat dihitung $F_{measure}$ dari hasil klasifikasi pada skenario 6 sebagai berikut :

$$F_{measure} = 2 * \left(\frac{66{,}94 * 97{,}51}{66{,}94 + 97{,}51}\right) = 79{,}39\%$$

6. Pengujian dengan Skenario 7

Tabel 4.21 adalah merupakan hasil dari *cross-validation* terhadap 10 *fold* subset pada skenario 7 dengan SMOTE 60%.

Tabel 4.21 Confusion matrix dari cross-validation 10 fold Skenario 7

Confusion matrix		***Kelas Prediksi***	
		a	***b***
Kelas Aktual	**a = Normal**	501526	16361
	b = Botnet	164	8911

Berdasarkan *confusion matrix* pada Tabel 4.21 dapat dihitung tingkat akurasi, presisi dan *recall*/*sensitivity* untuk hasil *cross-validation 10 folds*, sebagai berikut :

$$Akurasi\ = \frac{(501526) + (8911)}{526962} x100\% = 96{,}86\%$$

$$Presisi(kelas\ a) = \frac{501526}{501526 + (164)} x100\% = 99{,}96\%$$

$$Presisi(kelas\ b) = \frac{8911}{8911 + (16361)} x100\% = 35{,}26\%$$

Maka rata-rata presisi dari dataset dari semua kelas adalah :

$$\bar{X}_{Presisi} = \frac{99{,}96 + 35{,}26}{2} = 67{,}61\%$$

Tingkat *recall* selanjutnya juga dihitung, untuk menemukan nilai F*measure*, sebagai berikut :

$$Recall\ (kelas\ a) = \frac{501526}{501526 + 16361} x100\% = 96{,}84\%$$

$$Recall\ (kelas\ b) = \frac{8911}{8911 + 164} x100\% = 98{,}19\%$$

Maka rata-rata *recall* dari semua kelas dapat dihitung menjadi sebagai berikut :

$$\bar{X}_{Recall} = \frac{96{,}84 + 98{,}19}{2} = 97{,}51\%$$

Sehingga selanjutnya dapat dihitung *Fmeasure* dari hasil klasifikasi pada skenario 7 sebagai berikut :

$$F_{measure} = \ 2 * \left(\frac{67{,}61 * 97{,}51}{67{,}61 + 97{,}51}\right) = 79{,}85\%$$

7. Pengujian dengan Skenario 8

Tabel 4.22 adalah merupakan hasil dari *cross-validation* terhadap 10 *fold* subset pada skenario 8 dengan SMOTE 70%.

Tabel 4.22 Confusion matrix dari cross-validation 10 fold Skenario 8

Confusion matrix		***Kelas Prediksi***	
		a	***b***
Kelas Aktual	**a = Normal**	501552	16335
	b = Botnet	175	9467

Berdasarkan *confusion matrix* pada Tabel 4.22 dapat dihitung tingkat akurasi, presisi dan *recall*/*sensitivity* untuk hasil *cross-validation 10 folds*, sebagai berikut :

$$Akurasi\ = \frac{(501552) + (9467)}{527529} x100\% = \ 96{,}87\%$$

$$Presisi(kelas\ a) = \frac{501552}{501552 + (175)} x100\% = 99{,}96\%$$

$$Presisi(kelas\ b) = \frac{9467}{9467 + (16335)} x100\% = 36{,}69\%$$

Maka rata-rata presisi dari dataset dari semua kelas adalah :

$$\bar{X}_{Presisi} = \frac{99{,}96 + 36{,}69}{2} = 68{,}32\%$$

Tingkat *recall* selanjutnya juga dihitung, untuk menemukan nilai $F_{measure}$, sebagai berikut :

$$Recall\ (kelas\ a) = \frac{501552}{501552 + 16335} x100\% = 96{,}84\%$$

$$Recall\ (kelas\ b) = \frac{9467}{9467 + 175} x100\% = 98{,}18\%$$

Maka rata-rata *recall* dari semua kelas dapat dihitung menjadi sebagai berikut :

$$\bar{X}_{Recall} = \frac{96{,}84 + 98{,}18}{2} = 97{,}51\%$$

Sehingga selanjutnya dapat dihitung $F_{measure}$ dari hasil klasifikasi pada skenario 8 sebagai berikut :

$$F_{measure} = 2 * \left(\frac{68{,}32 * 97{,}51}{68{,}32 + 97{,}51}\right) = 80{,}35\%$$

8. Pengujian dengan Skenario 9

Tabel 4.23 adalah merupakan hasil dari *cross-validation* terhadap 10 *fold* subset pada skenario 9 dengan SMOTE 80%.

Tabel 4.23 Confusion matrix dari cross-validation 10 fold Skenario 9

Confusion matrix		***Kelas Prediksi***	
		a	***b***
Kelas Aktual	**a = Normal**	501432	16455
	b = Botnet	182	10027

Berdasarkan *confusion matrix* pada Tabel 4.23 dapat dihitung tingkat akurasi, presisi dan *recall*/*sensitivity* untuk hasil *cross-validation 10 folds*, sebagai berikut :

$$Akurasi\ = \frac{(501432) + (10027)}{528096} x100\% = 96{,}85\%$$

$$Presisi(kelas\ a) = \frac{501432}{501432 + (182)} x100\% = 99{,}96\%$$

$$Presisi(kelas\ b) = \frac{10027}{10027 + (16455)} x100\% = 37{,}86\%$$

Maka rata-rata presisi dari dataset dari semua kelas adalah :

$$\bar{X}_{Presisi} = \frac{99{,}96 + 37{,}86}{2} = 68{,}91\%$$

Tingkat recall selanjutnya juga dihitung, untuk menemukan nilai $F_{measure}$, sebagai berikut :

$$Recall\ (kelas\ a) = \frac{501432}{501432 + 16455} x100\% = 96{,}82\%$$

$$Recall\ (kelas\ b) = \frac{10027}{10027 + 182} x100\% = 98{,}21\%$$

Maka rata-rata recall dari semua kelas dapat dihitung menjadi sebagai berikut :

$$\bar{X}_{Recall} = \frac{96{,}82 + 98{,}21}{2} = 97{,}51\%$$

Sehingga selanjutnya dapat dihitung $F_{measure}$ dari hasil klasifikasi pada skenario 9 sebagai berikut :

$$F_{measure} = 2 * \left(\frac{68{,}91 * 97{,}51}{68{,}91 + 97{,}51}\right) = 80{,}75\%$$

9. Pengujian dengan Skenario 10

Tabel 4.24 adalah merupakan hasil dari *cross-validation* terhadap 10 *fold* subset pada skenario 10 dengan SMOTE 90%.

Tabel 4.24 Confusion matrix dari cross-validation 10 fold Skenario 10

Confusion matrix		***Kelas Prediksi***	
		a	***b***
Kelas Aktual	**a = Normal**	501449	16438
	b = Botnet	195	10581

Berdasarkan *confusion matrix* pada Tabel 4.24 dapat dihitung tingkat akurasi, presisi dan *recall*/*sensitivity* untuk hasil *cross-validation 10 folds*, sebagai berikut :

$$Akurasi = \frac{(501449) + (10581)}{528663} x100\% = 96{,}85\%$$

$$Presisi(kelas\ a) = \frac{501449}{501449 + (195)} x100\% = 99{,}96\%$$

$$Presisi(kelas\ b) = \frac{10581}{10581 + (16438)} x100\% = 39{,}16\%$$

Maka rata-rata presisi dari dataset dari semua kelas adalah :

$$\bar{X}_{presisi} = \frac{99{,}96 + 39{,}16}{2} = 69{,}56\%$$

Tingkat recall selanjutnya juga dihitung, untuk menemukan nilai $F_{measure}$, sebagai berikut :

$$Recall\ (kelas\ a) = \frac{501449}{501449 + 16438} x100\% = 96{,}82\%$$

$$Recall\ (kelas\ b) = \frac{10581}{10581 + 195} x100\% = 98{,}19\%$$

Maka rata-rata recall dari semua kelas dapat dihitung menjadi sebagai berikut :

$$\bar{X}_{recall} = \frac{96{,}82 + 98{,}19}{2} = 97{,}50\%$$

Sehingga selanjutnya dapat dihitung $F_{measure}$ dari hasil klasifikasi pada skenario 10 sebagai berikut :

$$F_{measure} = 2 * \left(\frac{69{,}56 * 97{,}50}{69{,}56 + 97{,}50}\right) = 81{,}19\%$$

10. Pengujian dengan Skenario 11

Tabel 4.25 adalah merupakan hasil dari *cross-validation* terhadap 10 *fold* subset pada skenario 11 dengan SMOTE 100%.

Tabel 4.25 Confusion matrix dari cross-validation 10 fold Skenario 11

Confusion matrix		***Kelas Prediksi***	
		a	***b***
Kelas Aktual	**a = Normal**	501578	16309
	b = Botnet	210	11134

Berdasarkan *confusion matrix* pada Tabel 4.25 dapat dihitung tingkat akurasi, presisi dan *recall*/*sensitivity* untuk hasil *cross-validation 10 folds*, sebagai berikut :

$$Akurasi\ = \frac{(501578) + (11134)}{529231} x100\% = 96{,}88\%$$

$$Presisi(kelas\ a) = \frac{501578}{501578 + (210)} x100\% = 99{,}95\%$$

$$Presisi(kelas\ b) = \frac{11134}{11134 + (16309)} x100\% = 40{,}57\%$$

Maka rata-rata presisi dari dataset dari semua kelas adalah :

$$\bar{X}_{presisi} = \frac{99{,}95 + 40{,}57}{2} = 70{,}26\%$$

Tingkat *recall* selanjutnya juga dihitung, untuk menemukan nilai F*measure*, sebagai berikut :

$$Recall\ (kelas\ a) = \frac{501578}{501578 + 16309} x100\% = 96{,}85\%$$

$$Recall\ (kelas\ b) = \frac{11134}{11134 + 210} x100\% = 98{,}14\%$$

Maka rata-rata *recall* dari semua kelas dapat dihitung menjadi sebagai berikut :

$$\bar{X}_{recall} = \frac{96{,}85 + 98{,}14}{2} = 97{,}49\%$$

Sehingga selanjutnya dapat dihitung F_{measure} dari hasil klasifikasi pada skenario 11 sebagai berikut :

$$F_{measure} = 2 * \left(\frac{70{,}26 * 97{,}49}{70{,}26 + 97{,}49}\right) = 81{,}67\%$$

4.3.3 Pengujian Model NB dengan SMOTE dan Metode BFS

Pada tahap eksperimen ketiga ini menggunakan Naïve Bayes *classifier* dengan oversampling data berurutan dengan besaran **10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%** dan **100%** dari data minor dan seleksi fitur menggunakan metode BFS

1. Pengujian dengan Skenario 12

Tabel 4.26 adalah merupakan hasil dari *cross-validation* terhadap 10 *fold* subset pada skenario 12 dengan SMOTE 10% dan seleksi fitur metode BFS.

Tabel 4.26 Confusion matrix dari cross-validation 10 folds Skenario 12

Confusion matrix		***Kelas Prediksi***	
		a	***b***
Kelas Aktual	**a = Normal**	511178	6709
	b = Botnet	432	5807

Berdasarkan *confusion matrix* pada Tabel 4.26. dapat dihitung tingkat akurasi, presisi dan *recall*/*sensitivity* untuk hasil *cross-validation 10 folds*, sebagai berikut :

$$Akurasi = \frac{(511178) + (5807)}{524126} x100\% = 98{,}64\%$$

$$presisi(kelas\ a) = \frac{511178}{511178 + (432)} x100\% = 99{,}92\%$$

$$presisi(kelas\ b) = \frac{5807}{5807 + (6709)} x100\% = 46{,}40\%$$

Maka rata-rata presisi dari dataset dari semua kelas adalah :

$$\bar{X}_{presisi} = \frac{99{,}92 + 46{,}40}{2} = 73{,}16\%$$

Tingkat *recall* selanjutnya juga dihitung, untuk menemukan nilai F*measure*, sebagai berikut :

$$Recall\ (kelas\ a) = \frac{511178}{511178 + 6709} x100\% = 98{,}70\%$$

$$Recall\ (kelas\ b) = \frac{5807}{5807 + 432} x100\% = 93{,}07\%$$

Maka rata-rata *recall* dari semua kelas dapat dihitung menjadi sebagai berikut :

$$\bar{X}_{recall} = \frac{98{,}07 + 93{,}07}{2} = 95.89\%$$

Sehingga selanjutnya dapat dihitung *Fmeasure* dari hasil klasifikasi pada skenario 12 sebagai berikut :

$$F_{measure} = 2 * \left(\frac{73{,}16 * 95{,}89}{73{,}16 + 95{,}89}\right) = 82{,}995\%$$

2. Pengujian dengan Skenario 13

Dataset diklasifikasi menggunakan Naïve Bayes *classifier* dengan oversampling data sebesar **20%** dari data minor dan seleksi fitur metode BFS. evaluasi secara *cross-validation* terhadap 10 *fold* subset

Tabel 4.27 Confusion matrix dari cross-validation 10 folds Skenario 13

Confusion matrix		***Kelas Prediksi***	
		a	***b***
Kelas Aktual	**a = Normal**	510357	7530
	b = Botnet	470	6336

Berdasarkan *confusion matrix* pada Tabel 4.27. dapat dihitung tingkat akurasi, presisi dan *recall*/*sensitivity* untuk hasil *cross-validation 10 folds*, sebagai berikut :

$$Akurasi = \frac{(510357) + (6336)}{524693} x100\% = 98{,}48\%$$

$$presisi(kelas\ a) = \frac{510357}{510357 + (470)} x100\% = 99{,}91\%$$

$$presisi(kelas\ b) = \frac{6336}{6336 + (7530)} x100\% = 45.70\%$$

Maka rata-rata presisi dari dataset dari semua kelas adalah :

$$\bar{X}_{presisi} = \frac{99{,}91 + 45.70}{2} = 72{,}81\%$$

Tingkat *recall* selanjutnya juga dihitung, untuk menemukan nilai $F_{measure}$, sebagai berikut :

$$Recall\ (kelas\ a) = \frac{510357}{510357 + 7530} x100\% = 98{,}55\%$$

$$Recall\ (kelas\ b) = \frac{6336}{6336 + 470} x100\% = 93{,}1\%$$

Maka rata-rata recall dari semua kelas dapat dihitung menjadi sebagai berikut :

$$\bar{X}_{recall} = \frac{98{,}55 + 93{,}1}{2} = 95{,}83\%$$

Sehingga selanjutnya dapat dihitung $F_{measure}$ dari hasil klasifikasi pada skenario 13 sebagai berikut :

$$F_{measure} = 2 * \left(\frac{72{,}81 * 49{,}75}{72{,}81 + 49{,}75}\right) = 81{,}58\%$$

3. Pengujian dengan Skenario 14

Dataset diklasifikasi menggunakan Naïve Bayes *classifier* dengan oversampling data sebesar **30%** dari data minor dan seleksi fitur metode BFS. evaluasi secara *cross-validation* terhadap 10 *fold* subset.

Tabel 4.28 Confusion matrix dari cross-validation 10 folds Skenario 14

Confusion matrix		***Kelas Prediksi***	
		a	***b***
Kelas Aktual	**a = Normal**	512871	5016
	b = Botnet	489	6884

Berdasarkan *confusion matrix* pada Tabel 4.28. dapat dihitung tingkat akurasi, presisi dan *recall*/*sensitivity* untuk hasil *cross-validation 10 folds*, sebagai berikut :

$$Akurasi = \frac{(512871) + (6884)}{525260} x100\% = 98{,}95\%$$

$$presisi(kelas\ a) = \frac{512871}{512871 + (489)} x100\% = 99{,}91\%$$

$$presisi(kelas\ b) = \frac{6884}{6884 + (5016)} x100\% = 57.85\%$$

Maka rata-rata presisi dari dataset dari semua kelas adalah :

$$\bar{X}_{presisi} = \frac{99{,}91 + 57{,}58}{2} = 78{,}745\%$$

Tingkat *recall* selanjutnya juga dihitung, untuk menemukan nilai F*measure*, sebagai berikut :

$$Recall\ (kelas\ a) = \frac{512871}{512871 + 5016} x100\% = 98{,}55\%$$

$$Recall\ (kelas\ b) = \frac{6884}{6884 + 489} x100\% = 93{,}37\%$$

Maka rata-rata *recall* dari semua kelas dapat dihitung menjadi sebagai berikut :

$$\bar{X}_{recall} = \frac{98{,}55 + 93{,}37}{2} = 95{,}96\%$$

Sehingga selanjutnya dapat dihitung *F measure* dari hasil klasifikasi pada skenario 14 sebagai berikut :

$$F_{measure} = 2 * \left(\frac{98{,}96 * 78{,}745}{98{,}96 + 78{,}745}\right) = 87{,}71\%$$

4. Pengujian dengan Skenario 15

Dataset diklasifikasi menggunakan Naïve Bayes *classifier* dengan oversampling data sebesar **40%** dari data minor dan seleksi fitur metode BFS. evaluasi secara *cross-validation* terhadap 10 *fold* subset

Tabel 4.29 Confusion matrix dari cross-validation 10 folds Skenario 15

Confusion matrix		***Kelas Prediksi***	
		a	***b***
Kelas Aktual	**a = Normal**	514181	3706
	b = Botnet	554	7386

Berdasarkan *confusion matrix* pada Tabel 4.29. dapat dihitung tingkat akurasi, presisi dan *recall/sensitivity* untuk hasil *cross-validation 10 folds*, sebagai berikut :

$$Akurasi = \frac{(514181) + (7386)}{525827} x100\% = 99{,}19\%$$

$$presisi(kelas\ a) = \frac{514181}{514181 + (554)} x100\% = 99{,}90\%$$

$$presisi(kelas\ b) = \frac{7386}{7386 + (3706)} x100\% = 66{,}59\%$$

Maka rata-rata presisi dari dataset dari semua kelas adalah :

$$\bar{X}_{presisi} = \frac{99{,}90 + 66{,}59}{2} = 83{,}245\%$$

Tingkat *recall* selanjutnya juga dihitung, untuk menemukan nilai F-measure, sebagai berikut :

$$Recall\ (kelas\ a) = \frac{514181}{514181 + 3706} x100\% = 99{,}29\%$$

$$Recall\ (kelas\ b) = \frac{7386}{7386 + 554} x100\% = 93{,}03\%$$

Maka rata-rata *recall* dari semua kelas dapat dihitung menjadi sebagai berikut :

$$\bar{X}_{recall} = \frac{99{,}29 + 93{,}03}{2} = 96{,}16\%$$

Sehingga selanjutnya dapat dihitung $F_{measure}$ dari hasil klasifikasi pada skenario 15 sebagai berikut :

$$F_{measure} = 2 * \left(\frac{83{,}245 * 96{,}16}{83{,}245 + 96{,}16}\right) = 89{,}23\%$$

5. Pengujian dengan Skenario 16

Dataset diklasifikasi menggunakan Naïve Bayes *classifier* dengan oversampling data sebesar **50%** dari data minor dan seleksi fitur metode BFS. evaluasi secara *cross-validation* terhadap 10 *fold* subset.

Tabel 4.30 Confusion matrix dari cross-validation 10 folds Skenario 16

Confusion matrix		***Kelas Prediksi***	
		a	***b***
Kelas Aktual	**a = Normal**	514090	3797
	b = Botnet	599	7909

Berdasarkan *confusion matrix* pada Tabel 4.30. dapat dihitung tingkat akurasi, presisi dan *recall*/*sensitivity* untuk hasil *cross-validation 10 folds*, sebagai berikut :

$$Akurasi = \frac{(514090) + (7909)}{526395} x100\% = 99{,}16\%$$

$$presisi(kelas\ a) = \frac{514090}{514090 + (599)} x100\% = 99{,}89\%$$

$$presisi(kelas\ b) = \frac{7909}{7909 + (3797)} x100\% = 67{,}57\%$$

Maka rata-rata presisi dari dataset dari semua kelas adalah :

$$\bar{X}_{presisi} = \frac{99{,}89 + 67{,}57}{2} = 83{,}73\%$$

Tingkat *recall* selanjutnya juga dihitung, untuk menemukan nilai F*measure*, sebagai berikut :

$$Recall\ (kelas\ a) = \frac{514090}{514090 + 3797} x100\% = 99{,}27\%$$

$$Recall\ (kelas\ b) = \frac{7909}{7909 + 599} x100\% = 92{,}96\%$$

Maka rata-rata *recall* dari semua kelas dapat dihitung menjadi sebagai berikut :

$$\bar{X}_{recall} = \frac{99{,}27 + 92{,}96}{2} = 96{,}115\%$$

Sehingga selanjutnya dapat dihitung $F_{measure}$ dari hasil klasifikasi pada skenario 16 sebagai berikut :

$$F_{measure} = 2 * \left(\frac{96{,}115 * 83{,}73}{96{,}115 + 83{,}73}\right) = 89{,}49\%$$

6. Pengujian dengan Skenario 17

Dataset diklasifikasi menggunakan Naïve Bayes *classifier* dengan oversampling data sebesar **60%** dari data minor dan seleksi fitur metode BFS. evaluasi secara *cross-validation* terhadap 10 *fold* subset

Tabel 4.31 Confusion matrix dari cross-validation 10 folds Skenario 17

Confusion matrix		***Kelas Prediksi***	
		a	***b***
Kelas Aktual	**a = Normal**	514017	3870
	b = Botnet	644	8431

Berdasarkan *confusion matrix* pada Tabel 4.31. dapat dihitung tingkat akurasi, presisi dan *recall*/*sensitivity* untuk hasil *cross-validation 10 folds*, sebagai berikut :

$$Akurasi = \frac{(514017) + (8431)}{526962} x100\% = 99{,}14\%$$

$$presisi(kelas\ a) = \frac{514017}{514017 + (644)} x100\% = 99{,}88\%$$

$$presisi(kelas\ b) = \frac{8431}{8431 + (3870)} x100\% = 68{,}54\%$$

Maka rata-rata presisi dari dataset dari semua kelas adalah :

$$\bar{X}_{presisi} = \frac{99{,}88 + 68{,}54}{2} = 84{,}21\%$$

Tingkat *recall* selanjutnya juga dihitung, untuk menemukan nilai $F_{measure}$, sebagai berikut :

$$Recall\ (kelas\ a) = \frac{514017}{514017 + 3870} x100\% = 99{,}26\%$$

$$Recall\ (kelas\ b) = \frac{8431}{8431 + 644} x100\% = 92{,}91\%$$

Maka rata-rata recall dari semua kelas dapat dihitung menjadi sebagai berikut :

$$\bar{X}_{recall} = \frac{99{,}26 + 92{,}91}{2} = 96{,}085\%$$

Sehingga selanjutnya dapat dihitung $F_{measure}$ dari hasil klasifikasi pada skenario 17 sebagai berikut :

$$F_{measure} = 2 * \left(\frac{84{,}21 * 96{,}085}{84{,}21 + 96{,}085}\right) = 89{,}76\%$$

7. Pengujian dengan Skenario 18

Dataset diklasifikasi menggunakan Naïve Bayes *classifier* dengan oversampling data sebesar **70%** dari data minor dan seleksi fitur metode BFS. evaluasi secara *cross-validation* terhadap 10 *fold* subset

Tabel 4.32 Confusion matrix dari cross-validation 10 folds Skenario 18

Confusion matrix		***Kelas Prediksi***	
		a	***b***
Kelas Aktual	**a = Normal**	513906	3981
	b = Botnet	676	8966

Berdasarkan *confusion matrix* pada Tabel 4.32. dapat dihitung tingkat akurasi, presisi dan *recall/sensitivity* untuk hasil *cross-validation 10 folds*, sebagai berikut :

$$Akurasi = \frac{(513906) + (8966)}{527529} x100\% = 99{,}12\%$$

$$presisi(kelas\ a) = \frac{513906}{513906 + (676)} x100\% = 99{,}87\%$$

$$presisi(kelas\ b) = \frac{8966}{8966 + (3981)} x100\% = 69{,}26\%$$

Maka rata-rata presisi dari dataset dari semua kelas adalah :

$$\bar{X}_{presisi} = \frac{99{,}87 + 69{,}26}{2} = 84{,}565\%$$

Tingkat *recall* selanjutnya juga dihitung, untuk menemukan nilai $F_{measure}$, sebagai berikut :

$$Recall\ (kelas\ a) = \frac{513906}{513906 + 3981} x100\% = 99{,}24\%$$

$$Recall\ (kelas\ b) = \frac{8966}{8966 + 676} x100\% = 92{,}97\%$$

Maka rata-rata *recall* dari semua kelas dapat dihitung menjadi sebagai berikut :

$$\bar{X}_{recall} = \frac{99{,}24 + 92{,}97}{2} = 96{,}12\%$$

Sehingga selanjutnya dapat dihitung F_{measure} dari hasil klasifikasi pada skenario 18 sebagai berikut :

$$F_{measure} = \ 2 * \left(\frac{84{,}565 * 96{,}12}{84{,}565 + 96{,}12}\right) = 89{,}98\%$$

8. Pengujian dengan Skenario 19

Dataset diklasifikasi menggunakan Naïve Bayes *classifier* dengan oversampling data sebesar **80%** dari data minor dan seleksi fitur metode BFS. evaluasi secara *cross-validation* terhadap 10 *fold* subset

Berdasarkan *confusion matrix* pada Tabel 4.33. dapat dihitung tingkat akurasi, presisi dan *recall*/*sensitivity* untuk hasil *cross-validation 10 folds*, sebagai berikut :

$$Akurasi\ = \frac{(513744) + (9487)}{528096} x100\% = \ 99{,}08\%$$

$$presisi(kelas\ a) = \frac{513744}{513744 + (722)} x100\% = 99{,}86\%$$

$$presisi(kelas\ b) = \frac{9487}{9487 + (4143)} x100\% = 69{,}61\%$$

Tabel 4.33 Confusion matrix dari cross-validation 10 folds Skenario 19

Confusion matrix		***Kelas Prediksi***	
		a	***b***
Kelas Aktual	**a = Normal**	513744	4143
	b = Botnet	722	9487

Maka rata-rata presisi dari dataset dari semua kelas adalah :

$$\bar{X}_{presisi} = \frac{99{,}86 + 69{,}61}{2} = 84{,}74\%$$

Tingkat *recall* selanjutnya juga dihitung, untuk menemukan nilai F*measure*, sebagai berikut :

$$Recall\ (kelas\ a) = \frac{513744}{513744 + 4143} x100\% = 99{,}21\%$$

$$Recall\ (kelas\ b) = \frac{9487}{9487 + 722} x100\% = 92{,}93\%$$

Maka rata-rata recall dari semua kelas dapat dihitung menjadi sebagai berikut :

$$\bar{X}_{recall} = \frac{99{,}21 + 92{,}93}{2} = 96{,}07\%$$

Sehingga selanjutnya dapat dihitung *Fmeasure* dari hasil klasifikasi pada skenario 19 sebagai berikut :

$$F_{measure} = 2 * \left(\frac{84{,}74 * 96{,}07}{84{,}74 + 96{,}07}\right) = 90{,}05\%$$

9. Pengujian dengan Skenario 20

Dataset diklasifikasi menggunakan Naïve Bayes *classifier* dengan oversampling data sebesar **90%** dari data minor dan seleksi fitur metode BFS. evaluasi secara *cross-validation* terhadap 10 *fold* subset

Tabel 4.34 Confusion matrix dari cross-validation 10 folds Skenario 20

Confusion matrix		***Kelas Prediksi***	
		a	***b***
Kelas Aktual	**a = Normal**	513602	4285
	b = Botnet	759	10017

Berdasarkan *confusion matrix* pada Tabel 4.34. dapat dihitung tingkat akurasi, presisi dan *recall*/*sensitivity* untuk hasil *cross-validation 10 folds*, sebagai berikut :

$$Akurasi\ = \frac{(513602) + (10017)}{528663} x100\% = 99{,}05\%$$

$$presisi(kelas\ a) = \frac{513602}{513602 + (759)} x100\% = 99{,}86\%$$

$$presisi(kelas\ b) = \frac{10017}{10017 + (4285)} x100\% = 70{,}04\%$$

Maka rata-rata presisi dari dataset dari semua kelas adalah :

$$\bar{X}_{presisi} = \frac{99{,}86 + 70{,}04}{2} = 84{,}95\%$$

Tingkat *recall* selanjutnya juga dihitung, untuk menemukan nilai F*measure*, sebagai berikut :

$$Recall\ (kelas\ a) = \frac{513602}{513602 + 4285} x100\% = 99{,}18\%$$

$$Recall\ (kelas\ b) = \frac{10017}{10017 + 759} x100\% = 92{,}96\%$$

Maka rata-rata *recall* dari semua kelas dapat dihitung menjadi sebagai berikut :

$$\bar{X}_{recall} = \frac{99{,}18 + 92{,}96}{2} = 96{,}07\%$$

Sehingga selanjutnya dapat dihitung *F*$_{measure}$ dari hasil klasifikasi pada skenario 20 sebagai berikut :

$$F_{measure} = 2 * \left(\frac{84{,}95 * 96{,}07}{84{,}95 + 96{,}07}\right) = 90{,}17\%$$

10. Pengujian dengan Skenario 21

Tabel 4.35 adalah merupakan hasil dari *cross-validation* terhadap 10 *fold* subset pada skenario 2 dengan SMOTE 10% dan seleksi fitur metode BFS.

Tabel 4.35 Confusion matrix dari cross-validation 10 folds Skenario 21

Confusion matrix		***Kelas Prediksi***	
		a	***b***
Kelas Aktual	**a = Normal**	513429	4458
	b = Botnet	812	10532

Berdasarkan *confusion matrix* pada Tabel 4.35 dapat dihitung tingkat akurasi, presisi dan *recall*/*sensitivity* untuk hasil *cross-validation 10 folds*, sebagai berikut :

$$Akurasi\ = \frac{(513429) + (10532)}{529231} x100\% = 99{,}00\%$$

$$presisi(kelas\ a) = \frac{513429}{513429 + (812)} x100\% = 99{,}85\%$$

$$presisi(kelas\ b) = \frac{10532}{10532 + (4458)} x100\% = 70{,}27\%$$

Maka rata-rata presisi dari dataset dari semua kelas adalah :

$$\bar{X}_{presisi} = \frac{99{,}85 + 70{,}27}{2} = 85{,}06\%$$

Tingkat *recall* selanjutnya juga dihitung, untuk menemukan nilai F*measure*, sebagai berikut :

$$Recall\ (kelas\ a) = \frac{513429}{513429 + 4458} x100\% = 99{,}14\%$$

$$Recall\ (kelas\ b) = \frac{10532}{10532 + 812} x100\% = 92{,}85\%$$

Maka rata-rata *recall* dari semua kelas dapat dihitung menjadi sebagai berikut :

$$\bar{X}_{recall} = \frac{99{,}14 + 92{,}85}{2} = 95{,}99\%$$

Sehingga selanjutnya dapat dihitung *F measure* dari hasil klasifikasi pada skenario 11 sebagai berikut :

$$F_{measure} = 2 * \left(\frac{85{,}06 * 95{,}99}{85{,}06 + 95{,}99}\right) = 90{,}20\%$$

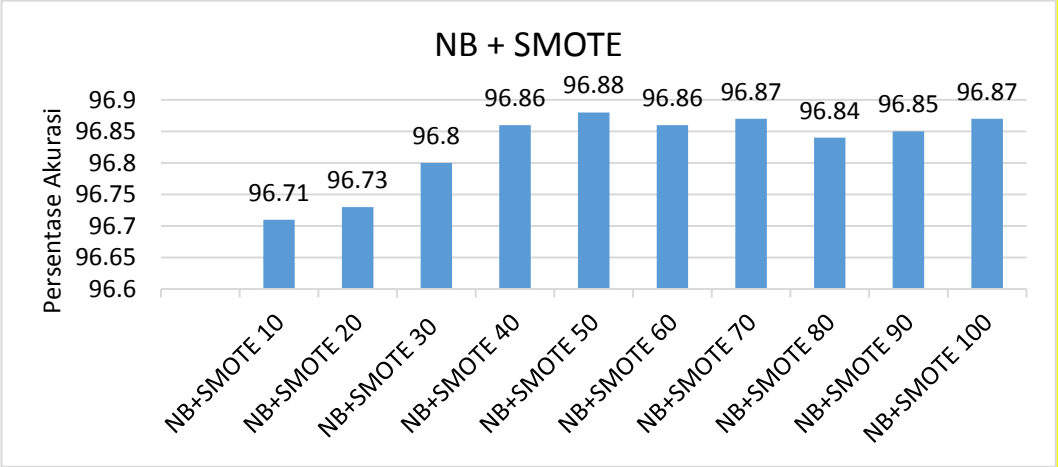
4.4 Analisa Hasil

Berdasarkan dari pengujian model dan dataset pada tahap 4.3 dapat diperoleh hasil dari masing-masing skenario. Tabel 4.36 disajikan perbandingan hasil tingkat akurasi, presisi dan *recall*/*sensitivity*. Perbandingan meliputi penggunaan model klasifikasi dengan Naïve Bayes, Klasifikasi Naïve Bayes dengan SMOTE dan Klasifikasi Naïve Bayes dengan SMOTE dan seleksi fitur menggunakan metode BFS.

Dari hasil pengujian model pertama deteksi Botnet yang hanya menggunakan Naïve Bayes *classifier* saja dengan *cross-validation* terhadap 10 *fold* subset (pengujian skenario 1), diketahui hasil akurasi adalah **96,68%** dan nilai F-measure **76,01%**.

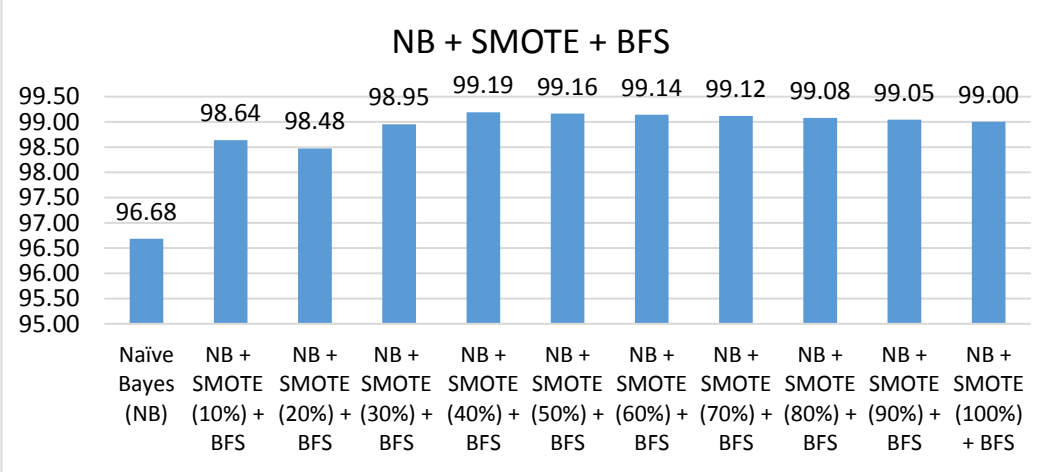
Kemudian pengujian berikutnya dengan model kedua deteksi Botnet yaitu Klasifikasi Naïve Bayes dengan SMOTE tanpa seleksi fitur, Gambar 4.5 disajikan grafik persentase akurasi pengujian dari skenario 2 sampai dengan skenario 11. Penggunaan teknik SMOTE untuk sistem deteksi Botnet dapat meningkatkan

akurasi hingga **0,2%** pada besaran SMOTE 50%, dengan pengujian penambahan persentase kelas buatan tidak dapat meningkatkan akurasi secara signifikan. Seperti pada gambar 4.5 dapat dilihat, dimana skenario SMOTE 60% sampai dengan 100% akurasi naik turun antara 96,86% - 96,88%.



Gambar 4.5 Akurasi Pengujian NB+SMOTE

Kemudian pengujian berikutnya dengan model ketiga deteksi botnet yaitu Klasifikasi Naïve Bayes dengan SMOTE dan seleksi fitur dengan Metode BFS. Dapat dilihat pada Gambar 4.6, dimana nilai akurasi menggunakan Naïve Bayes dengan SMOTE dan metode BFS semakin meningkat, disisi lain bahwa asumsi semakin banyak data sintetis (buatan) maka akurasi semakin baik adalah tidak tepat, dari hasil pengujian dapat dilihat pada gambar 4.6 bahwa akurasi terbaik pada *NB+SMOTE 40%+BFS* (pengujian dengan skenario 15) yaitu **99,19%** dan hasil skenario berikutnya tren akurasi manjadi turun,.



Gambar 4.6 Akurasi Pengujian NB + SMOTE + BFS

Dari hasil pengujian yang disajikan pada Tabel 4.36 dapat diketahui metode usulan tersebut dengan studi dataset CTU-13 skenario 1, menunjukkan nilai akurasi terbaik adalah pada skenario 15 pengujian dengan data SMOTE sebesar 40% dan Metode BFS.

Tabel 4.36 Perbandingan akurasi masing-masing skenario

No	Skenario	NB+SMOTE			NB+SMOTE+BFS		
		Presisi (%)	**Recall (%)**	**Akurasi (%)**	**Presisi (%)**	**Recall (%)**	**Akurasi (%)**
1	Naïve Bayes (NB)	62.22	97.64	**96.685**	62.22	97.64	**97.99**
2	SMOTE (10%)	63.19	97.59	96.713	73.16	95.89	98.64
3	SMOTE (20%)	64.09	97.52	96.732	72.81	95.83	98.48
4	SMOTE (30%)	65.14	97.51	96.805	78.75	95.96	98.95
5	SMOTE (40%)	66.11	97.46	96.862	83.25	96.16	**99.19**
6	SMOTE (50%)	66.94	97.51	**96.881**	83.73	96.12	99.16
7	SMOTE (60%)	67.61	97.51	96.864	84.21	96.09	99.14
8	SMOTE (70%)	68.32	97.51	96.87	84.57	96.12	99.12
9	SMOTE (80%)	68.91	97.51	96.85	84.74	96.07	99.08
10	SMOTE (90%)	69.56	97.51	96.854	84.95	96.07	99.05
11	SMOTE (100%)	70.26	97.51	96.879	85.06	95.99	99.00

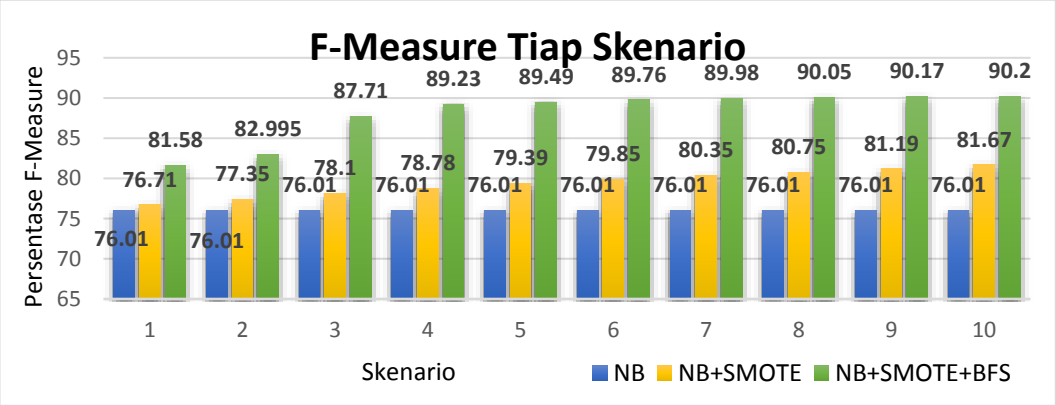
Selanjutnya untuk mengukur kinerja klasifikasi menggunakan Metode Naïve Bayes *Classifier* dapat menggunakan F-*Measure* atau dapat juga disebut F1-*Score*. F-*measure* didapatkan dari nilai rata-rata harmonis dari presisi dan *recall*, oleh karena itu pengukuran kinerja sistem menggunkan *F-measure* dapat memberikan penilaian kinerja yang lebih seimbang. Berikut Tabel 4.37 yang merupakan simpulan hasil perhitungan yang telah dilakukan pada pengujian di tahap 4.3.

Dari hasil simpulan F1-*Score* pada Tabel 4.37 dan Gambar 4.6 di atas dapat disimpulkan bahwa metode model deteksi botnet menggunakan Naïve Bayes *Classifier* dengan SMOTE dan Metode BFS memiliki kinerja yang lebih baik dalam melakukan klasifikasi data *flow* normal dengan Botnet. Memiliki tren naik secara kontinyu berdasarkan input persentase SMOTE sesuai skenario.

Tabel 4.37 Hasil perhitungan F-Measure

No	Skenario	NB+SMOTE			NB+SMOTE+BFS		
		Presisi (%)	Recall (%)	F-Measure (%)	Presisi (%)	Recall (%)	F-Measure (%)
1	Naïve Bayes (NB)	62.22	97.64	**76.01**	62.22	97.64	**76.01**
2	SMOTE (10%)	63.19	97.59	76,71	73.16	95.89	81.58
3	SMOTE (20%)	64.09	97.52	77,35	72.81	95.83	82.995
4	SMOTE (30%)	65.14	97.51	78.1	78.75	95.96	87.71
5	SMOTE (40%)	66.11	97.46	78.78	83.25	96.16	89.23
6	SMOTE (50%)	66.94	97.51	79.39	83.73	96.12	89.49
7	SMOTE (60%)	67.61	97.51	79.85	84.21	96.09	89.76
8	SMOTE (70%)	68.32	97.51	80,35	84.57	96.12	89.98
9	SMOTE (80%)	68.91	97.51	80,75	84.74	96.07	90.05
10	SMOTE (90%)	69.56	97.51	81,19	84.95	96.07	90.17
11	SMOTE (100%)	70.26	97.51	**81,67**	85.06	95.99	**90.2**

Dari 20 skenario pengujian dengan SMOTE, F1-*Score* terbaik pada skenario SMOTE dan BFS dengan besaran *oversampling data* 100% dengan hasil F1-*Score* **90,2%** dibandingkan dengan metode Naïve Bayes dengan SMOTE 100% tanpa seleksi fitur yaitu sebesar **81,67%** atau hanya Naïve Bayes *classifier* saja yaitu **76,01**% . Hal ini dapat dikatakan bahwa metode usulan dapat dengan tepat menempatkan obyek sesuai dengan kelasnya berdasarkan nilai atribut yang dimilikinya.



Gambar 4.7 Grafik Perbandingan F-Measure tiap Skenario dengan Naive Bayes Classifier

Halaman ini sengaja dikosongkan

BAB 5
KESIMPULAN

5.1 Kesimpulan

Pada penelitian kali ini telah dilakukan pengujian dengan dataset Botnet CTU-13 data skenario 1, sebagai data uji model deteksi Botnet menggunakan Naïve Bayes dengan SMOTE dan metode BFS. Dengan data pengujian total 523.559 *flow,* yang berisi 517.887 *flow* trafik normal dan 5.672 *flow* trafik Botnet, dengan 21 skenario pengujian, maka didapatkan kesimpulan sebagai berikut :

Pertama, 10 skenario pengujian (pengujian dengan skenario 12 sampai dengan 21) dengan model deteksi Botnet menggunakan Naïve Bayes *classifier* dengan SMOTE dan metode BFS menunjukkan akurasi lebih baik yaitu dengan kenaikan masing-masing 1.95%, 1.79%, 2.27%, 2.51%, 2.48%, 2.46%, 2.43%, 2.39%, 2.36%, 2.32% dimana nilai akurasi terbaik pada skenario 5 pengujian dengan data SMOTE sebesar 40%, perbandingan akurasi antara Naïve Bayes dan metode usulan pada skenario 5 adalah 2.51% lebih baik akurasinya atau Naïve Bayes dengan SMOTE tanpa seleksi fitur akurasi metode usulan 2,3% lebih baik.

Kedua, Penggunaan teknik SMOTE dan metode BFS untuk deteksi Botnet dengan membangkitkan *oversampling data* minor dengan persentase tinggi dalam hal pengujian yang telah dilakukan 100% data minor tidak serta merta akan meningkatkan akurasi lebih baik. Diketahui dari hasil pengujian hasil akurasi terbaik pada skenario 5 dengan 40% *oversampling data*, tren akurasi menurun kembali dengan 50% *oversampling data* dan seterusnya. Sehinga Metode usulan mampu mendetekesi Botnet dengan baik yaitu akurasi deteksi mencapai 99.19% pada skenario SMOTE 40%

Ketiga, F1-Score terbaik pada skenario SMOTE ke 10 (pengujian dengan skenario 21) dengan besaran oversampling data 100% dan seleksi fitur metode BFS dengan hasil F1-Score **90,2** dibandingkan dengan metode Naïve Bayes dengan SMOTE 100% tanpa seleksi fitur yaitu sebesar **81,67%** atau hanya Naïve Bayes *classifier* saja yaitu **76,01**% . Hal ini dapat dikatakan bahwa metode usulan dapat

dengan tepat menempatkan obyek sesuai dengan kelasnya berdasarkan nilai atribut yang dimilikinya.

5.2 Saran

Dalam model yang telah dilakukan pada penelitian ini hanya menggunakan skenario 1, maka untuk penelitian berikutnya dapat menggunakan keseluruhan dataset CTU-13 dan mengembangkan model menjadi adaptif, sehingga sistem deteksi akan secara otomatis mengenali trafik Botnet dengan karakteristik yang memiliki kemiripan jika terdapat jenis Botnet baru. Sekaligus diintegrasi kedalam *intrusion prevention system*.

DAFTAR PUSTAKA

[1] Vania, J., Meniya, . A. & Jethva, H. B., 2013. A Review on Botnet and Detection Technique. *International Journal of Computer Trends and Technology,* 4(1).

[2] Amaliyah, B. e. a., 2011. *Klasifikasi Voted Preceptron untuk Identifikasi Melanoma.* Yogyakarta, s.n.

[3] Anon., 2014. *Calculated Fields in Argus.* [Online] Available at: http://nsmwiki.org/Argus

[4] Azab, A., Alazab, M. & Aiash, M., 2016. *Machine learning based Botnet Identification Traffic.* s.l., s.n.

[5] B., 2013. Penerapan Algoritma Naive Bayes Untuk Mengklasifikasi Data Nasabah Asuransi. *TECHSI : Jurnal Penelitian Teknik Informatika.*

[6] Barro, R. . A., Sulvianti, I. D. & Afendi, F. M., 2013. Penerapan Synthetic Minority Oversampling Technique (Smote) Terhadap Data Tidak Seimbang Pada Pembuatan Model Komposisi Jamu. Volume 1(1), pp. e9(1-6).

[7] Bijalwan, A., Chand, N., Pilli, . E. S. & Khrisna, C. R., 2016. *Botnet analysis using ensemble classifier.* s.l., s.n., p. 502–504.

[8] Chawla, N. W. & Bowyer, K. W., 2002. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research ,* Volume 16, p. 321–357.

[9] Chowdhury, S. et al., 2017. Botnet Detection using graph-based feature clustering. *Journal of Big Data.*

[10] Garcia, S., Grill, M., Stiborek, H. & Zunino, A., 2014. An empirical comparison of botnet detection.

[11] He, H., M., I. & Garcia, E. . A., 2009. Learning from Imbalanced Data. *IEEE Transactions On Knowledge And Data Engineering,* September, Volume 21, p. 9.

[12] Karthikeyan, T. & Thangaraju, P., 2015. Best First and Greedy Search Based CFS- Naïve Bayes Classification Algorithms for Hepatitis Diagnosis. *Biosciences Biotechnology Research Asia,* Volume 12(1), pp. 983-990.

[13] Le, D. C., Zincir, A. N. & I., M., 2016. *Data Analytics on Network Traffic Flows for Botnet Behaviour Detection.* s.l., s.n.

[14] Pattekari, S. A. & Parveen, A., 2012. Prediction System for Heart Disease Using Naive Bayes. *International Journal of Advanced Computer and Mathematical Sciences,* pp. 2230-9624, Vol. 3.

[15] Prasetyo, E., 2012. *Data Mining - Konsep dan Aplikasi Menggunakan Matlab.* Yogyakarta: Andi Yogyakarta.

[16] R., K. & A.V, R., 2014. Flow Based Analysis To Identify Botnet Infected Systems. *Journal of Theoretical and Applied Information Technology,* Volume 67 NO.2.

[17] Saad, S. et al., 2011. *Detecting P2P Botnets through Network Behavior Analysis and Machine Learning.* s.l., IEEE.

[18] Sebastian, G., 2017. *Malware capture facility project.* [Online] Available at: https://stratosphereips.org

[19] Sharma, P., Doshi, D. & Prajapati, M. M., 2016. *Cybercrime: Internal security threat.* s.l., s.n.

[20] Supriyanti, W., K. & Amborowati, A., 2016. Perbandingan Kinerja Algoritma C4.5 Dan Naive Bayes Untuk Ketepatan Pemilihan Konsentrasi Mahasiswa. *Jurnal INFORMA Politeknik Indonusa Surakarta,* pp. 2442-7942.

[21] Tsai, M.-F. & Yu, S.-S., 2016. Distance Metric Based Oversampling Method for Bioinformatics and Performance Evaluation. *Journal Medical System,* Volume 40, p. 159.

[22] Xie, Z., Jiang, L., Ye, T. & Li, X.-L., 2015. A Synthetic Minority Oversampling Method Based on Local Densities in Low-Dimensional Space for Imbalanced Learning. *DASSFA,* pp. 3-18.

[23] Xu, Y. et al., 2017. Fuzzy–synthetic minority oversampling technique: Oversampling based on fuzzy set theory for Android malware detection in imbalanced datasets. *International Jurnal of Distributed Sensor Network,* Volume 13.

[24] Zao, D. et al., 2013. Botnet detection based on traffic behavior analysis and flow intervals. *Computer and Security - ScienceDirect.*

BIOGRAFI PENULIS

Lab HCCV – Teknik Elektro ITS (10-7-2017)

Penulis lahir di Blitar pada tanggal 19 Desember 1984, yang merupakan putra bungsu dari tiga bersaudara. Dari kecil hingga menamatkan SMA berada di Blitar. Penulis pernah menempuh pendidikan dasar di MI Ringinanom, lalu pendidikan menengah di MTs Negeri 1 Blitar, dan pendidikan menengah atas di SMK N 1 Blitar. Kemudian melanjutkan ke jenjang perguruan tinggi D3 di PENS-ITS Surabaya (sekarang PENS) lulus 2007, Orang tua menuntut agar sekolah lagi untuk mengejar S1 maka penulis menempuh lintas jalur di ITS pada jurusan Teknik Informatika tahun 2007-2009.

Pengalaman kerja antara lain sebagai staf dukungan teknis *Network Operation Center* (NOC) pada perusahaan swasta PT Pasifik Satelit Nusantara di Surabaya dari 2008-2010. Dan pada tahun 2010-sekarang, Penulis bekerja menjadi Aparatur Sipil Negara sebagai Pengelola jaringan TIK pada Pusat Sistem Informasi dan Teknologi Keuangan (Pusintek) di lingkup Kementerian Keuangan. Pada tahun 2015, Penulis mengikuti program beasiswa pendidikan S2 dari Kementerian Komunikasi dan Informasi (Kominfo) pada Bidang keahlian Telematika – CIO (*Chief Information Officer*) di Teknik Elektro ITS Surabaya. Tema penelitian yang diambil adalah bidang Keamanan Informasi.

Penulis dapat dihubungi melalui e-mail: **fuadin19@gmail.com**.