OPEN ACCESS
J. OF ICT, VOL. 4, NO. 2, PP.010-020, DES 2022

ISSN : 2686-1089 (ONLINE)
DOI: 10.52661

Journal of Informatics and Communications Technology (JICT)

Model Deteksi Botnet Menggunakan Algoritma Decision Tree Dengan Untuk Mengidentifikasi Serangan Click Fraud

Rafli Firdaus [#1], Asep Id Hadiana [*2], Fatan Kasyidi [#3]

1,2,3 *Program Studi Informatika, Universitas Jenderal Achmad Yani*
Jl. Terusan Jend. Sudirman, Cibeber, Kec. Cimahi Sel., Kota Cimahi, Jawa Barat 40531

[1] raflifirdaus18@if.unjani.ac.id / firdaus.rafli24@gmail.com
[2] asep.hadiana@lecture.unjani.ac.id
[3] fatan.kasyidi@lecture.unjani.ac.id

Abstrak

Malicious Software (*Malware*) merupakan program yang dibuat khusus untuk merugikan orang lain. Salah satunya *Botnet*, di mana *Botnet* dapat menginfeksi perangkat komputer serta membuat komputer tersebut sebagai suatu alat yang nantinya akan dikendalikan secara paksa oleh pemilik dari program *Malware* tersebut. *Botnet* sendiri dapat melakukan serangan *Click Fraud* untuk melakukan *Fake Clicks* terhadap iklan yang bersifat *Pay Per Click*. *Botnet* dengan serangan *Click Fraud* memiliki pola tingkah laku yang dapat diklasifikasikan dengan menggunakan *Dataset CTU*-13. Sehingga *Flow Traffic* dari *Botnet* yang melakukan serangan *Click Fraud* akan dapat terdeteksi dengan menggunakan algoritma *CART* dengan menggunakan teknik *SMOTE* untuk melakukan *Oversampling* dan teknik *Random Undersampling* untuk menangati ketidakseimbangan sebaran data untuk setiap kelasnya. Dengan menggunakan rasio *Undersampling* yaitu 50% dan terdapat 2 skenario untuk penggunaan teknik *SMOTE*, yaitu sebelum dan setelah data dibagi menjadi data latih dan data uji. Berdasarkan dari hasil penelitian yang telah dilakukan dapat disimpulkan bahwa dengan penggunaan teknik *SMOTE* dan *Random Undersampling* dalam kasus untuk pendeteksian *Botnet* yang melakukan serangan *Click Fraud* sebelum membagi dataset menjadi data latih dan data uji dapat meningkatkan akurasi ataupun kinerja dari model tersebut dengan mencapai tingkat akurasi sebesar **99.97**%. Dan Nilai *F-Score* dari model yang menggunakan *SMOTE* dan *Random Undersampling* adalah **99.96**%.

Kata Kunci: *Botnet, CART, Click Fraud, Malware, Random Undersampling, SMOTE*

I. Pendahuluan

Malicious Software (*malware*) merupakan segala jenis perangkat lunak yang digunakan secara bertentangan antara pemilik *Host* dari suatu komputer ataupun *Virtual Host*[1], dengan memiliki berbagai tujuan seperti menyebabkan kerusakan terhadap suatu komputer, *Server*, jaringan komputer atau menyebabkan hilangnya informasi yang bersifat pribadi[2]. Hal-hal berikut menjadi dasar bagi pelaku kejahatan internet untuk meningkatkan keefisienan teknologi *Malware* agar mempermudah peretasan guna mendapatkan keuntungan bagi pelaku kejahatan itu sendiri. Jenis *Malware* dapat diklasifikasikan berdasarkan karakteristik-karakteristik jalannya suatu *program*. *Malware* juga diklasifikasikan tergantung dari jumlah informasi aktual atau pesan yang dikirim. Beberapa jenis *malware* yang sering muncul yaitu, *Viruses*, *Botnet*, *Spyware*, dan *Trojans*[3].

Salah satu jenis *Malware* yang paling berbahaya dalam dunia *cybersecurity*[4] pada saat ini adalah *Botnet*[5]. *Botnet* dapat didefinisikan sebagai kumpulan-kumpulan dari komputer yang telah terserang oleh *Malware* di

mana komputer tersebut dikendalikan oleh *Botmaster*[6] (orang yang memegang kendali dari *Botnet*), melalui suatu entitas *Command and Control Server*[7]. *Command and Control Server* (C&C) merupakan pusat kendali dari perintah-perintah *Botnet* yang muncul untuk melakukan tugas dan serangan yang terkoordinasi[8]. Beberapa contoh serangan yang dapat dilakukan oleh *Botnet* di antaranya, mencuri data pribadi, melakukan serangan *Distributed Denial of Service* (DDoS), mengirim *Spam Email, Phising* dan *Click Fraud*[9].

Sebelumya terdapat penelitian terdahulu yang terkait mengenai pendedeteksian *Botnet* dengan menggunakan *Machine Learning*. Penelitian ini menggunakan algoritma *k-Nearest Neighbour* (KNN)[10], tentang deteksi *Botnet* dengan menggunakan *Dataset Czech Technical University* (*CTU*-13), di mana penelitian tersebut dapat melakukan pengelompokkan *Flow* normal dan *Flow Botnet* dengan menggunakan algoritma *k-Nearest Neighbour* dengan tingkat akurasi sebesar 93,9%. Akan tetapi metode *k-Nearest Neighbour* ini tidak cocok dalam masalah pendeteksian *Botnet*, dikarenakan biaya komputasi yang tinggi sehingga menyebabkan lamanya waktu yang digunakan ketika melakukan proses klasifikasi[11]. Terdapat penelitian terkait sebelumnya yang menyebutkan bahwa pemilihan fitur-fitur untuk deteksi *botnet* yang paling optimal adalah *Source Ip*, *Destination Ip*, *Start Time*, *Duration*, *Ip Protocol*, *Protocol State*, *Total Number of Packets* dan *total Bytes Exchanged*[12].

Algoritma *CART* merupakan algoritma pembelajaran populer dalam *Supervised Machine Learning*[13]. Algoritma *CART* merupakan algoritma *Decision Tree Learning* yang dapat digunakan untuk melakukan klasifikasi ataupun regresi. Untuk penggunaan algoritma *CART* sebagai klasifikasi ini akan mencoba untuk memprediksi suatu kelas[14] dengan tingkat ketelitian yang tinggi dan cocok untuk pendeteksian *Botnet*. *Dataset* yang digunakan pada penelitian ini adalah *CTU-13 Dataset* skenario 9. CTU-13 *Dataset* merupakan *Dataset* hasil penangkapan data lalu lintas botnet di CTU (*Czech Technical University*), *Czech Republic*, pada tahun 2011. Lalu lintas jaringan yang ditangkap dalam dalam *Dataset* ini adalah lalu lintas jaringan botnet pada jaringan sebenarnya yang tercampur dengan lalu lintas jaringan normal dan *background*. *Dataset CTU*-13 ini merupakan *dataset* yang memiliki label, di mana label adalah lalu lintas jaringan *botnet*, *normal*, dan *background*. Akan tetapi pada penelitian [15] menyebutkan bahwa pada *Dataset CTU*-13 terjadi ketidakseimbangan data antar kelas, yang mana dapat menimbulkan *Overfitting* ketika suatu model dilatih dengan menggunakan *Dataset* yang memiliki sebaran kelas data yang tidak seimbang.

Oleh karena itu, algoritma *Decision Tree CART* ini dipilih dalam penelitian ini dengan menggunakan teknik *SMOTE* dan *Random Undersampling* sebagai metode untuk mencegah terjadinya *Overfitting* dari model *Decision Tree*. Penelitian ini bertujuan untuk membuat suatu model deteksi *Botnet* jenis serangan *Click Fraud* dengan menggunakan algoritma *CART* dengan *SMOTE* dan *Random Undersampling* sebagai metode untuk mengatasi *Overfitting*. Penelitian ini menggunakan *dataset* lalu lintas *Botnet CTU*-13 yang tersedia secara publik.

II. Tinjauan Pustaka

Terdapat penelitian terdahulu terkait pendedeteksian *Botnet* dengan menggunakan *Machine Learning.* Dalam penelitian ini menggunakan algoritma k-*Nearest Neighbors* (KNN), dan algoritma *Decision Tree* digunakan untuk mendeteksi serangan *botnet* dengan menggunakan metode *Feature Ranking* untuk meningkatkan efisiensi waktu dan mendapatkan tingkat akurasi yang lebih tinggi[16]. *Feature Ranking Method* yang digunakan adalah *Gini Importance* dan *Information Gain. Dataset* yang digunakan dalam penelitian ini yaitu *CTU-13 Dataset* yang dibagi menjadi dua, yaitu *Dataset Quasi-Balanced-CTU13* dan *Extended QB-CTU13* di mana pada *Dataset Extended QB-CTU13* menambahkan 3 kelas *Botnet* lain. Fitur yang terdapat pada lalu lintas jaringan antara lain, *sPort*, *dPort*, *mLen*, *vLen*, *mTime*, *vTime*, *mResp*, *vResp*, *nBytes*, *nSYN*, dan *nPackets*. Hal pertama yang dilakukan peneliti adalah membagi *CTU*-13 *Dataset* menjadi 2, setelah itu peneliti akan melakukan *Feature Ranking* terhadap 11 fitur pada lalu lintas jaringan untuk deteksi *Botnet* dan membaginya menjadi 3 subset fitur dengan menggunakan *Gini Importance* dan *Gain Information*. Tiga subset fitur tersebut adalah:

- *dPort, nPackets, nBytes, vLen, mLen*
- *dPort, nPackets, nBytes, vLen, mLen, mTime*
- *dPort, nPackets, nBytes, vLen, mLen, mTime, vTime*

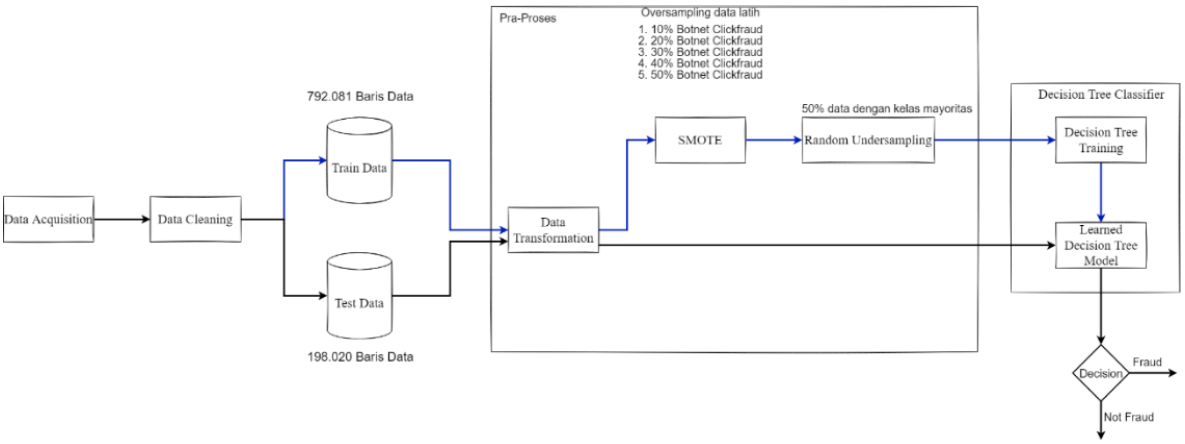
Hasil dari penelitian tersebut merupakan perbandingan antara model deteksi *Botnet Decision Tree* dengan menggunakan *Dataset Extended QB-CTU13* dan penggunaan fitur yang tingkat relevansinya telah di golongkan dengan berdasarkan *Gini Importance* dan *Information Gain* dengan jumlah fitur 5 dan model deteksi *Botnet Decision Tree* dengan menggunakan keseluruhan fitur yang berjumlah 11. Tingkat akurasi model deteksi *Botnet* dengan menggunakan 5 fitur adalah 99% sedangkan untuk tingkat akurasi model deteksi *Botnet* dengan menggunakan 11 fitur adalah 94%. Akan tetapi pada penelitian tersebut, dataset yang digunakan tidak seimbang dikarenakan jumlah data dengan label flow botnet lebih sedikit dibandingkan dengan data dengan label flow *background* dan normal. Oleh karena itu penelitian yang diajukan ini akan membuat suatu model pendeteksi *Botnet* yang melakukan serangan *Click Fraud* dengan menggunakan algoritma *CART* dan metode *Sampling* untuk mengatasi ketidakseimbangan dataset.

Tabel I
Tabel Penelitian Terdahulu

No.	Judul Penelitian	Penulis	Tahun	Metode	Akurasi
1.	*Click-Fraud Detection Using XG Boost and Convolutional Neural Networks*[17]	Mohapatra, Tripti Pragyan dan Manikandan, K	2020	*XGBoost*	99.8%
				Convolutional Neural Networks	99.47%
2.	*Towards a Machine Learning Approach for Detecting Click Fraud in Mobile Advertizing*[18]	Mouawi, Riwa Awad, Mariette Chehab, Ali El Hajj, Imad H. dan Kayssi, Ayman	2019	*k-Nearest Neighbour*	98%
				Support Vector Machine	96%
				Artificial Neural Network	93%
3.	*A Click Fraud Detection Scheme based on Cost sensitive BPNN and ABC in Mobile Advertising*[19]	Zhang, Xin Liu, Xuejun dan Guo, Han	2018	*Cost-sensitive Back Propagation Neural Network - Artificial Bee Colony algorithm*	99.14%

III. Metodologi Penelitian

Perancangan Model pendeteksi *Botnet* ini menggunakan metode algoritma *Decision Tree* dan juga teknik *SMOTE* dan *Random Undersampling*. Proses perancangan model ini diawali dengan pengumpulan data dengan menggunakan *dataset* CTU-13, dataset yang dikumpulkan dipisahkan menjadi data pelatihan (training data) dan data pengujian (testing data). Tahap selanjutnya yaitu Praproses, pada tahapan ini terdapat tiga proses yaitu *Data Transformation*, *SMOTE*, dan *Random Undersampling*, untuk *Data Transformation* ini akan menubah fitur-fitur yang bertipe data kategorik menjadi numerik agar dapat dibaca oleh model pengklasifikasian. Tahapan pra-proses selanjutnya adalah *SMOTE* di mana selanjutnya data latih akan melalui proses *Oversampling* dengan membuat data sintesis atau buatan sesuai dengan skenario yaitu 10%, 20%, 30%, 40%, dan 50% dari data dengan kelas minoritas. Setelah itu proses dilanjutkan dengan *Undersampling* di mana data dengan kelas mayoritas akan dilakukan penghapusan secara acak, tahapan *SMOTE* dan *Undersampling* ini membantu untuk mengatasi *Dataset* yang memiliki permasahalan *Imbalanced Data* dan mencegah terjadinya *Overfitting* dalam suatu model *Machine Learning*. Lalu langkah terakhir adalah memasukkan kumpulan data yang telah dilakukan praproses terlebih dahulu dan akan dimasukkan ke dalam sistem klasifikasi *Botnet* serangan *Click Fraud* dengan menggunakan *Algoritma Decision Tree*. Sistem klasifikasi *Botnet* serangan *Click Fraud* memiliki dua modul – satu untuk melatih sistem dengan menggunakan data latih dan yang lainnya untuk mengevaluasi sistem dengan menggunakan data uji. Seperti yang terlihat pada gambar 3.



Gambar 1 Metodologi Penelitian

1. Data Acquisition

Dataset yang digunakan pada penelitian ini adalah *CTU-13 Dataset* skenario 9. CTU-13 *Dataset* merupakan *Dataset* hasil penangkapan data lalu lintas botnet di CTU (*Czech Technical University*), *Czech Republic*, pada tahun 2011. Lalu lintas jaringan yang ditangkap dalam dalam *Dataset* ini adalah lalu lintas jaringan botnet pada jaringan sebenarnya yang tercampur dengan lalu lintas jaringan normal dan *background*. *Dataset CTU*-13 ini merupakan *dataset* yang memiliki label, di mana label adalah lalu lintas jaringan *botnet*, *normal*, dan *background*. *Total* jumlah baris data yang terdapat yaitu sebanyak 2.087.509 baris data, akan tetapi dikarenakan terdapat keterbatasan kemampuan perangkat lunak untuk membaca *file* total data yang dapat diperoleh sebanyak 1.054.119 baris data.

2. Data Cleaning

Setelah data diperoleh langkah selanjutnya adalah melakukan *Data Cleaning*, pada tahapan ini akan melakukan penghapusan terhadap baris data yang memiliki *Missing Value*. Ketika suatu *Dataset* memiliki *Missing Values* akan memiliki dampak bagi model *Machine Learning* ketika akan membaca data dengan *Missing Values* yang akan mengakibatkan data menjadi rusak dan mengakibatkan tingkat akurasi dari *Model Machine Learning* akan menjadi turun, maka dengan melakukan *Data Cleaning* ini dapat membantu untuk mencegah dampak buruk tersebut bagi model pendeteksian *Botnet* yang melakukan serangan *Click Fraud*.

3. Pembagian Dataset

Setelah data dilakukan proses penghapusan, selanjutnya *Dataset* akan dilakukan pembagian ke dalam data latih dan data uji. Untuk rasio perbandingan pembagian data uji dan data latih yang dipilih adalah 80:20, di mana 80% akan dijadikan data latih dan 20%, dikarenakan data yang telah diperoleh lebih besar dari 100.000 baris data dan kurang dari 1.000.000 baris data.

Total jumlah baris data untuk data latih adalah 796.978 dan untuk data uji sebanyak 199.246. Setelah dilakukan pembagian *dataset* menjadi data latih dan data uji terdapat permasalahan yang berpotensi dapat menyebabkan model dari *Decision Tree* yang dibangun terjadi *Overfitting* karena terjadinya *Imbalance Data* di mana perbandingan jumlah data masing-masing label tidak seimbang.

4. Data Transformation

Terdapat beberapa fitur di dalam *dataset CTU*-13 yang telah diperoleh bertipe data kategori, dikarenakan untuk implementasi algoritma *CART* dengan menggunakan *Library Scikit-learn* belum bisa dapat menerima masukkan data dengan tipe data kategori, maka tipe data dari fitur *Proto*, *SrcAddr*, *Dir*, *DstAddr*, *Dport*, dan *State* perlu dikonversi menjadi numerik dikarenakan tipe data dari fitur-fitur tersebut adalah kategori, seperti yang terlihat pada sampel *dataset* pada gambar 2.

StartTime	Dur	Proto	SrcAddr	Sport	Dir	DstAddr	Dport	State	sTos	dTos	TotPkts	TotBytes	SrcBytes	Click Fraud
8/17/2011 12:01	16.637154	tcp	147.32.84.59	49181	->	195.113.232.97	80	FSPA_FSPA	0	0	6	412	272	No
8/17/2011 12:01	16.637119	tcp	147.32.84.59	49182	->	195.113.232.97	80	SRPA_SPA	0	0	6	412	272	No
8/17/2011 14:32	18.614891	tcp	147.32.84.59	4323	->	195.113.232.82	80	SRPA_SPA	0	0	6	416	276	No
8/17/2011 15:01	30.001623	udp	147.32.84.192	1667	<->	147.32.80.9	53	CON	0	0	6	444	370	No
8/17/2011 17:03	15.854728	tcp	147.32.84.204	2407	->	195.113.232.97	80	SRPA_FSPA	0	0	10	1796	764	Yes
8/17/2011 15:30	4.677235	tcp	147.32.84.192	4709	->	195.113.232.82	80	FSPA_FSPA	0	0	13	5781	785	Yes
8/17/2011 14:54	19.471376	tcp	147.32.84.192	1465	->	195.113.232.82	80	SRPA_SPA	0	0	16	9292	884	No
8/17/2011 14:49	2.846505	tcp	147.32.84.192	1060	->	195.113.232.82	80	FSPA_FSPA	0	0	22	14403	954	Yes
8/17/2011 15:12	10.952472	tcp	147.32.84.204	2174	->	195.113.232.97	80	SRPA_SPA	0	0	113	96217	3774	Yes
8/17/2011 15:40	25.762432	tcp	147.32.84.192	1629	->	195.113.232.97	80	SRPA_SPA	0	0	142	123984	4782	Yes
8/17/2011 14:56	8.179756	tcp	147.32.84.204	1059	->	195.113.232.82	82	FSPA_FSPA	0	0	149	137376	3661	No
8/17/2011 15:15	23.991753	tcp	147.32.84.192	3120	->	195.113.232.82	80	SRPA_SPA	0	0	205	190165	5397	Yes
8/17/2011 15:52	23.711479	tcp	147.32.84.204	2029	->	195.113.232.97	80	SRPA_SPA	0	0	206	179728	6854	Yes

Gambar 2 Sampel Dataset

Untuk transformasi data dari fitur *SrcAddr* dan *DstAddr* akan dilakukan konversi *Ip Address* ke dalam desimal. Proses konversi fitur *Proto* akan menggunakan pemetaan kamus yang digunakan pada penelitian[20] dan untuk Proses transformasi tipe data dari fitur *State*s akan menggunakan kamus pemetaan pada penelitian[21]. Seperti yang terlihat pada tabel II dan tabel III.

Tabel II
Tabel kamus pemetaan fitur protokol

Protokol	**Nilai**	**Protokol**	**Nilai**
TCP	0	IGMP	8
UDP	1	IPv6 - ICMP	9
RTP	2	IPv6	10
PIM	3	UDT	11
ICMP	4	ESP	12
ARP	5	UNAS	13
IPX/SPX	6	RARP	14
RCTP	7		

Tabel III
Tabel kamus pemetaan beberapa fitur *state*

No	**Status**	**Kode**	**No**	**Status**	**Kode**	**No**	**Status**	**Kode**
1	‘ ’	1	57	RPAC_PA	1	7	PAC_PA	5
2	FSR_SA	30	58	FRPA_R	1	8	SRPA_SPA	9646
3	_FSA	296	59	SPA_SPA	2989	9	SRPA_FSRA	13
4	FSRPA_FSA	77	60	PA_RA	3	10	FPA_FRPA	49
5	SPA_SA	31	61	SPA_SRPA	4185	11	SRA_SPA	10
6	FSA_SRA	1181	62	RA_FA	8	12	SA_SRA	838

5. SMOTE Oversampling

Teknik *Oversampling* yang digunakan dalam penelitian ini adalah *Synthetic Minority Oversampling* (SMOTE), pada tahapan *Oversampling* ini akan membuat data kelas minor buatan atau sintesis dengan skenario rasio perbandingan terhadap kelas mayoritas sebesar 10%, 20%, 30%, 40%, dan 50% dengan 2 skenario yaitu, penggunaan SMOTE sebelum dan sesudah dataset dibagi menjadi data latih dan data uji.

6. Random Undersampling

Untuk teknik *Undersampling* yang akan dilakukan adalah menggunakan adalah menggunakan *Random Undersampling* yang telah disediakan oleh *Library Scikit-learn*, dengan *Sampling Strategy* 50%. Contoh ketika terdapat dataset dengan jumlah kelas mayoritas sebesar 1,000 data dan data dengan kelas minoritas sebanyak 200 data. Tabel IV merupakan contoh perhitungan untuk menentukan berapa banyak jumlah data dengan kelas mayoritas setelah dilakukan *Undersampling.*

$$\frac{\boldsymbol{Jumlah\ Kelas\ Minoritas}}{\boldsymbol{x}} = \boldsymbol{Sampling\ Strategy} \quad (1)$$

Tabel IV
Tabel contoh perhitungan jumlah data kelas mayoritas setelah dilakukan undersampling

$$\frac{200}{x} = 0.5$$
$$x \,.\, \frac{200}{x} = 0.5 \,.\, x$$
$$200 = 0.5x$$
$$\frac{200}{0.5} = \frac{0.5x}{0.5}$$
$$400 = x$$

7. Klasifikasi Decision Tree

Proses klasifikasi akan menggunakan algoritma *CART*, Berikut merupakan tahapan-tahapan untuk melakukan pelatihan model klasifikasi *Decision Tree* dengan algoritma *CART* menggunakan dataset sampel pada gambar 2.

1) Tentukan *Root Node* dari *Decision Tree*

Untuk menentukan Root Node dapat dilakukan dengan melakukan perhitungan nilai Gini Impurity untuk seluruh fitur dan Gini Split untuk masing-masing fitur dengan menggunakan persamaan (2).

$$I_{Gini} = 1 - \sum_{i=1}^{j} P_i^2 \quad (2)$$

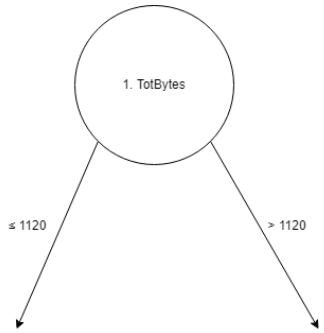
Tabel V
Tabel perhitungan *gini impurity* untuk fitur *duration*

$$\boldsymbol{I_{Gini}(Duration \leq 6.4284955) = 1 - (\frac{2}{2})^2 - (\frac{0}{2})^2 = 0}$$
$$\boldsymbol{I_{Gini}(Duration > 6.4284955) = 1 - (\frac{5}{11})^2 - (\frac{6}{11})^2 = 0.4958}$$
$$\boldsymbol{Gini(6.4284955 \leq Duration > 6.4284955)}$$
$$\boldsymbol{= \left(\frac{2}{13}\right) \times 0 + \left(\frac{11}{13} \times 0.4958\right) = 0.4113}$$
$$\boldsymbol{I_{Gini}(Duration \leq 21.5914275) = 1 - (\frac{4}{9})^2 - (\frac{5}{9})^2 = 0.4938}$$
$$\boldsymbol{I_{Gini}(Duration > 21.5914275) = 1 - (\frac{3}{4})^2 - (\frac{1}{4})^2 = 0.3750}$$
$$\boldsymbol{Gini(21.591427 \leq Duration > 21.591427)}$$
$$\boldsymbol{= \left(\frac{9}{13}\right) \times 0.4938 + \left(\frac{4}{13} \times 0.3750\right) = 0.4113}$$

Tabel VI
Tabel perhitungan *gini impurity* untuk fitur *TotBytes*

$$I_{Gini}(\boldsymbol{TotBytes} \leq \mathbf{1120}) = 1 - (\frac{0}{4})^2 - (\frac{4}{4})^2 = \mathbf{0}$$
$$I_{Gini}(\boldsymbol{TotBytes} > \mathbf{1120}) = 1 - (\frac{7}{9})^2 - (\frac{2}{9})^2 = \mathbf{0.3456}$$
$$\boldsymbol{Gini}(\mathbf{1120} \leq \boldsymbol{TotBytes} > \mathbf{1120}) = \left(\frac{4}{13}\right) \times \mathbf{0} + \left(\frac{9}{13} \times \mathbf{0.3456}\right) = \mathbf{0.2392}$$
$$I_{Gini}(\boldsymbol{TotBytes} \leq \mathbf{11847.5}) = 1 - (\frac{2}{7})^2 - (\frac{5}{7})^2 = \mathbf{0.4081}$$
$$I_{Gini}(\boldsymbol{TotBytes} > \mathbf{11847.5}) = 1 - (\frac{5}{6})^2 - (\frac{1}{6})^2 = \mathbf{0.2777}$$
$$\boldsymbol{Gini}(\mathbf{11847.5} \leq \boldsymbol{TotBytes} > \mathbf{11847.5}) = \left(\frac{7}{13}\right) \times \mathbf{0.4081} + \left(\frac{6}{13} \times \mathbf{0.2777}\right) = \mathbf{0.3479}$$

Tabel V dan Tabel VI merupakan hasil perhitungan *Gini Impurity* untuk fitur *Duration* dan *TotBytes*. Terlihat fitur *TotBytes* dengan nilai 1120≤*TotBytes*>1120 memiliki nilai *Gini Impurity* yang lebih kecil dibandingkan dengan nilai *Gini Impurity* dari fitur-fitur lain, sehingga menjadikannya fitur yang memiliki nilai *Gini Impurity* yang optimal untuk dijadikan sebagai *Root Node*.



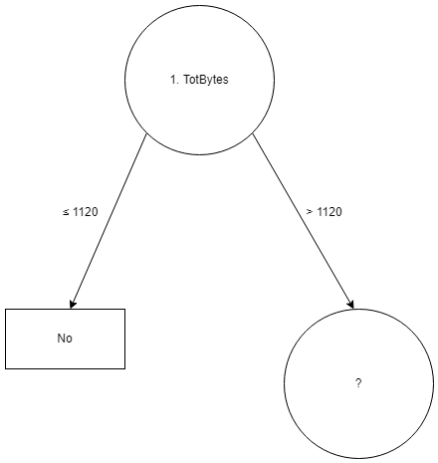
Gambar 3 Root Node

2) Tentukan cabang dari setiap nilai fitur atau atribut tertentu

Seperti yang terlihat pada Tabel VI fitur TotBytes memiliki dua nilai yaitu ≤ 1120 dan > 1120. Setelah itu untuk menentukan cabang dapat melihat dari nilai Gini Impurity untuk setiap nilai dari fitur tersebut, pada Tabel VII terdapat nilai pada fitur TotBytes yaitu ≤ 1120 yang memiliki nilai Gini Impurity 0, yang mengartikan bahwa Node dengan nilai ≤ 1120 adalah pure, sehingga tidak dapat dilakukan proses split lagi dan menjadikannya sebagai Leaf Node. Pada Node tersebut memiliki jumlah data dengan label Tidak sebanyak 4 dan data dengan label Ya sebanyak 0, sehingga pada cabang dengan nilai atribut ≤ 1120 dapat disimpulkan tidak melakukan Click Fraud sedangkan untuk nilai > 1120 akan dilakukan pencarian cabang pada tahap selanjutnya, sehingga Node tersebut disebut sebagai Internal Node. Sehingga proses pencarian cabang dapat dilihat pada gambar 4.

Tabel VII
Tabel cabang berdasarkan nilai atribut *TotBytes*

Fitur	Nilai	Count	Jumlah Ya	Jumlah Tidak	Gini Impurity	Gini Split
TotBytes	≤ 1120	4	0	4	0	0.2392
	> 1120	9	7	2	0.3456	



Gambar 4 Pencarian cabang berdasarkan atribut TotBytes > 1120

3) Lakukan perhitungan Gini Impurity untuk semua fitur berdasarkan nilai pada fitur yang menjadi cabang.
4) Ulangi langkah ke-3 dan 4 sampai tidak ada lagi Node yang dapat di-split atau semua konstruksi pohon sudah mencapai seluruh Leaf Node.

8. Evaluasi Model

Teknik pengujian untuk model Decision Tree dengan algoritma CART ini adalah menggunakan Confusion Matrix. Confusion Matrix merupakan suatu tabel yang digunakan untuk dapat mengevaluasi performa dari suatu algoritma klasifikasi. Confusion Matrix memvisualisasikan dan meringkas performa dari suatu algoritma klasifikasi. Selain itu Confusion Matrix juga dapat diartikan sebagai suatu tabel ringkasan dari jumlah prediksi tepat atau tidaknya yang dihasilkan oleh suatu model klasifikasi. Nilai True-Positive dan True-Negative akan memberikan informasi mengenai model Classifier ketika melakukan klasifikasi data dengan hasil yang benar, sedangkan untuk nilai False-Positive dan False-Negative akan memberikan informasi mengenai model Classifier ketika melakukan klasifikasi data dengan hasil yang tidak sesuai. Pada penelitian ini Flow Traffic normal, Background, dan Flow Botnet yang tidak melakukan Click Fraud akan ditandai dengan nilai negatif, sedangkan Flow Botnet yang melakukan Click Fraud akan ditandai dengan nilai positif. Tabel VIII merupakan gambaran tabel Confusion Matrix yang akan digunakan pada penelitian ini.

Tabel VIII
Tabel Confusion Matrix

Prediksi / Aktual	Click Fraud	Not Click Fraud
Click Fraud	TP	FN
Not Click Fraud	FP	TN

IV. Hasil Dan Diskusi

Hasil dari eksperimen dan pengujian untuk model pendeteksian *Botnet* yang melakukan serangan *Click Fraud* dengan mengimplementasikan *SMOTE* dan *Random Undersampling* sebelum dataset dibagi menjadi data latih dan data uji memiliki performa yang sangat baik di mana pada setiap perubahan rasio *Sampling* pada *SMOTE* nilai akurasi akan bertambah sebesar 0.01, seperti yang terlihat pada tabel IX. Diikuti dengan nilai presisi dan *recall* dimana setiap perubahan rasio sampling pada *SMOTE* nilai presisi dan *recall* akan meningkat. Sehingga dapat disimpulkan bahwa penggunaan *SMOTE* dan *Random Undersampling* sebelum dilakukan pembagian dataset menjadi data latih dan data uji dapat meningkatkan nilai akurasi sebesar 0.01 pada setiap rasio *Sampling* yang digunakan.

Tabel IX
Tabel hasil pengujian terhadap model yang mengimplementasikan SMOTE sebelum dilakukan pembagian data latih dan data uji

No	Skenario	Presisi (%)	Recall (%)	Akurasi (%)
1	SMOTE (10%) Random Undersampling (50%)	99.92	99.93	99.94
2	SMOTE (20%) Random Undersampling (50%)	99.94	99.95	99.95
3	SMOTE (30%) Random Undersampling (50%)	99.95	99.95	99.96
4	SMOTE (40%) Random Undersampling (50%)	99.95	99.96	99.96
5	SMOTE (50%) Random Undersampling (50%)	99.96	99.96	99.97

Evaluasi model pendeteksian Botnet yang melakukan serangan *Click Fraud* yang terakhir, yaitu mengukur kinerja klasifikasi dari algoritma CART. Pengukuran kinerja untuk klasifikasi dapat menggunakan F-*measure*. Nilai F-*measure* dapat diperoleh dari nilai rata-rata antara nilai presisi dan recall, oleh karena itu pengukuran kinerja sistem dengan menggunakan F-measure dapat memberikan penilaian kinerja atau performa yang lebih seimbang. Sepeti yang terlihat pada Tabel X, yang merupakan simpulan hasil evaluasi untuk nilai *F-measure* yang telah dilakukan pada tahap pengujian.

Tabel X
Tabel nilai F-Measure model yang mengimplementasikan SMOTE sebelum dilakukan pembagian data latih dan data uji

No	Skenario	Presisi (%)	Recall (%)	F-Measure (%)
1	SMOTE (10%) Random Undersampling (50%)	99.92	99.93	99.93
2	SMOTE (20%) Random Undersampling (50%)	99.94	99.95	99.94
3	SMOTE (30%) Random Undersampling (50%)	99.95	99.95	99.95
4	SMOTE (40%) Random Undersampling (50%)	99.95	99.96	99.95
5	SMOTE (50%) Random Undersampling (50%)	99.96	99.96	99.96

Dari hasil evaluasi *F-measure* pada tabel X, dapat disimpulkan bahwa metode model deteksi botnet yang melakukan serangan *Click Fraud* dengan menggunakan metoda *SMOTE* dan *Undersampling* ketika dataset belum dilakukan pembagian ke dalam data latih dan data uji memiliki hasil *F1-Score* maksimal sebesar 99.96%. Hal ini dapat dikatakan bahwa dengan penggunaan algoritma *CART* dan menggunakan metoda *SMOTE* dan *Undersampling* ketika dataset belum dilakukan pembagian ke dalam data latih dan data uji saja tanpa memiliki kinerja yang lebih baik dalam melakukan klasifikasi data *Flow Botnet* yang melakukan dan tidak melakukan serangan *Click Fraud*.

Dengan nilai akurasi sebesar 99%, setelah dilakukan analisa terhadap penyebab *Overfitting* dalam *Decision Tree* di mana terdapat dua hal yang dapat menyebabkan model menjadi *Overfitting*, yaitu pertama terdapat data *Noise* dan kurangnya jumlah data latih yang digunakan untuk melakukan pelatihan model. Pada model pendeteksian *Botnet* yang melakukan serangan *Click Fraud* ini kedua faktor tersebut tidak ditemukan, dikarenakan data sebelumnya sudah dilakukan proses *Cleaning* dan juga untuk jumlah data latih telah merepresentasikan jumlah yang optimal. Sehingga dapat disimpulkan bahwa model pendeteksian *Botnet* yang melakukan serangan *Click Fraud* ini dengan menggunakan algoritma *CART* dan metoda *SMOTE* dan *Random Undersampling* memiliki performa yang *Overpower* dalam perihal pendeteksian *Botnet* dengan menggunakan dataset *CTU*-13

V. Kesimpulan

Pada penelitian kali ini telah dilakukan pembuatan model deteksi *Botnet* yang melakukan serangan *Click Fraud* dengan menggunakan algoritma *CART* dan untuk pelatihan dan pengujian model dengan menggunakan dataset *Botnet* CTU-13 skenario 3. Hasil evaluasi model pendeteksian *Botnet* yang melakukan serangan *Click Fraud* dengan menggunakan metoda *SMOTE* dan *Random Undersampling* sebelum dilakukan pembagian dataset menjadi data latih dan data uji memiliki nilai akurasi maksimal yang tinggi yaitu 99.97%, dengan rasio *SMOTE* 50% dan *Random Undersampling* 50% dengan nilai *F1-Score F1-Score* 99.96%.

Dengan nilai akurasi maksimum sebesar 99.96%. Setelah dilakukan analisa terhadap penyebab *Overfitting* dalam *Decision Tree* di mana terdapat dua hal yang dapat menyebabkan model menjadi *Overfitting*, yaitu pertama terdapat data *Noise* dan kurangnya jumlah data latih yang digunakan untuk melakukan pelatihan model. Pada model pendeteksian *Botnet* yang melakukan serangan *Click Fraud* ini kedua faktor tersebut tidak ditemukan, dikarenakan data sebelumnya sudah dilakukan proses *Cleaning* dan juga untuk jumlah data latih telah merepresentasikan jumlah yang optimal. Sehingga, dapat disimpulkan bahwa penggunaan metoda *SMOTE* dan *Random Undersampling* sebelum dataset dibagi menjadi data latih dan data uji memiliki performa yang *Overpower* dalam perihal pendeteksian *Botnet* yang melakukan serangan *Click Fraud*.

Referensi

[1] A. Shafee, "Botnets and their detection techniques," *2020 Int. Symp. Networks, Comput. Commun. ISNCC 2020*, 2020, doi: 10.1109/ISNCC49221.2020.9297307.

[2] A. P. Namanya, A. Cullen, I. U. Awan, and J. P. Disso, "The World of Malware: An Overview," *Proc. - 2018 IEEE 6th Int. Conf. Futur. Internet Things Cloud, FiCloud 2018*, pp. 420–427, 2018, doi: 10.1109/FiCloud.2018.00067.

[3] C. Rohith and G. Kaur, "A Comprehensive Study on Malware Detection and Prevention Techniques used by Anti-Virus," *Proc. 2021 2nd Int. Conf. Intell. Eng. Manag. ICIEM 2021*, pp. 429–434, 2021, doi: 10.1109/ICIEM51511.2021.9445322.

[4] G. Y. Pangestu, A. I. Hadiana, and P. N. Sabrina, "Kriptografi Untuk Enkripsi Ganda Pada Gambar Menggunakan Algoritma AES (Advanced Encryption Standard) Dan RC5 (Rivest Code 5)," vol. 1, pp. 25–32, 2022.

[5] A. B. De Neira, A. M. Araujo, and M. Nogueira, "Early botnet detection for the internet and the internet of things by autonomous machine learning," *Proc. - 2020 16th Int. Conf. Mobility, Sens. Networking, MSN 2020*, pp. 516–523, 2020, doi: 10.1109/MSN50589.2020.00087.

[6] S. Gaonkar, N. F. Dessai, J. Costa, A. Borkar, S. Aswale, and P. Shetgaonkar, "A Survey on Botnet Detection Techniques," *Int. Conf. Emerg. Trends Inf. Technol. Eng. ic-ETITE 2020*, pp. 1–6, 2020, doi: 10.1109/ic-ETITE47903.2020.Id-70.

[7] A. Azab, M. Alazab, and M. Aiash, "Machine learning based botnet identification traffic," *Proc. - 15th IEEE Int. Conf. Trust. Secur. Priv. Comput. Commun. 10th IEEE Int. Conf. Big Data Sci. Eng. 14th IEEE Int. Symp. Parallel Distrib. Proce*, pp. 1788–1794, 2016, doi: 10.1109/TrustCom.2016.0275.

[8] P. Barthakur, M. Dahal, and M. Kanti Ghose, "An Efficient Machine Learning Based Classification Scheme for Detecting Distributed Command & Control Traffic of P2P Botnets," *Int. J. Mod. Educ. Comput. Sci.*, vol. 5, no. 10, pp. 9–18, 2013, doi: 10.5815/ijmecs.2013.10.02.

[9] N. Kaur and M. Singh, "Botnet and botnet detection techniques in cyber realm," *Proc. Int. Conf. Inven. Comput. Technol. ICICT 2016*, vol. 2016, 2016, doi: 10.1109/INVENTIVE.2016.7830080.

[10] V. U. Putri, E. B. Cahyono, and Y. Azhar, "Deteksi Botnet Pada Passive DNS Dengan Menggunakan Metode K Nearest Neighbor," *J. Repos.*, vol. 2, no. 12, p. 1631, 2020, doi: 10.22219/repositor.v2i12.450.

[11] M. Alshamkhany, W. Alshamkhany, M. Mansour, M. Khan, S. Dhou, and F. Aloul, "Botnet Attack Detection using Machine Learning," *Proc. 2020 14th Int. Conf. Innov. Inf. Technol. IIT 2020*, no. February, pp. 203–208, 2020, doi: 10.1109/IIT50501.2020.9299061.

[12] Z. M. Algelal, E. A. G. Aldhaher, D. N. Abdul-Wadood, and R. H. A. Al-Sagheer, "Botnet detection using ensemble classifiers of network flow," *Int. J. Electr. Comput. Eng.*, vol. 10, no. 3, pp. 2543–2550, 2020, doi: 10.11591/ijece.v10i3.pp2543-2550.

[13] N. N. Qomariyah, E. Heriyanni, A. N. Fajar, and D. Kazakov, "Comparative Analysis of Decision Tree Algorithm for Learning Ordinal Data Expressed as Pairwise Comparisons," *2020 8th Int. Conf. Inf. Commun. Technol. ICoICT 2020*, pp. 1–4, 2020, doi: 10.1109/ICoICT49345.2020.9166341.

[14] B. Charbuty and A. Abdulazeez, "Classification Based on Decision Tree Algorithm for Machine Learning," *J. Appl. Sci. Technol. Trends*, vol. 2, no. 01, pp. 20–28, 2021, doi: 10.38094/jastt20165.

[15] P. Gahelot and N. Dayal, "Flow Based Botnet Traffic Detection Using Machine Learning," *Lect. Notes Electr. Eng.*, vol. 605, pp. 418–426, 2020, doi: 10.1007/978-3-030-30577-2_36.

[16] J. Velasco-Mata, V. Gonzalez-Castro, E. F. Fernandez, and E. Alegre, "Efficient Detection of Botnet Traffic by Features Selection and Decision Trees," *IEEE Access*, vol. 9, pp. 120567–120579, 2021, doi: 10.1109/ACCESS.2021.3108222.

[17] T. P. Mohapatra and K. Manikandan, "Click-Fraud Detection Using XG Boost and Convolutional Neural Networks," vol. 29, no. 12, pp. 2410–2421, 2020.

[18] R. Mouawi, M. Awad, A. Chehab, I. H. El Hajj, and A. Kayssi, "Towards a Machine Learning Approach for Detecting Click Fraud in Mobile Advertizing," *Proc. 2018 13th Int. Conf. Innov. Inf. Technol. IIT 2018*, pp. 88–92, 2019, doi: 10.1109/INNOVATIONS.2018.8605973.

[19] X. Zhang, X. Liu, and H. Guo, "A click fraud detection scheme based on cost sensitive BPNN and ABC in mobile advertising," *2018 IEEE 4th Int. Conf. Comput. Commun. ICCC 2018*, pp. 1360–1365, 2018, doi: 10.1109/CompComm.2018.8780941.

[20] D. Sianturi, "UNIVERSITAS SUMATERA UTARA Poliklinik UNIVERSITAS SUMATERA UTARA," *J. Pembang. Wil. Kota*, vol. 1, no. 3, pp. 82–91, 2021.

[21] D. N. Fuadin, D. Pembimbing, P. Magister, D. T. Elektro, and F. T. Elektro, "Deteksi Botnet Menggunakan Naïve Bayes," *Thesis Fuadin, Didin Nizarul*, 2017.