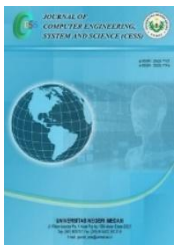
CESS (Journal of Computer Engineering, System and Science)
Vol. 10, No. 2, Juli 2025, pp.656-665

CESS
(Journal of Computer Engineering, System and Science)
Available online: https://jurnal.unimed.ac.id/2012/index.php/cess
ISSN: 2502-714x (Print) | ISSN: 2502-7131 (Online)



Sistem Deteksi *Malware* Menggunakan *Information Gain* dan *Decision Tree*

Malware Detection System Using Information Gain and Decision Tree

Rangga Aditya Pratama[1], Danang Triantoro Murdiansyah[2*]

[1,2]Prodi Informatika, Fakultas Informatika, Telkom University, Indonesia
Jalan Telekomunikasi, Bandung
Email: [1]ranggaaditya@students.telkomuniversity.ac.id, [2]danangtri@telkomuniversity.ac.id

**Corresponding Author*

A B S T R A K

Malicious Software, atau yang dikenal dengan *malware*, merupakan perangkat lunak berbahaya yang dapat menyebabkan hal-hal yang tidak diinginkan, seperti kehilangan data, pencurian informasi, penyebaran data pribadi, dan penyalahgunaan informasi penting. Pada penelitian ini, metode yang digunakan untuk deteksi *malware* adalah metode *Decision Tree* untuk klasifikasi dan metode *Information Gain* untuk seleksi fitur. Metode *Decision Tree* mempermudah dalam melacak dan memahami keputusan dengan struktur pohonnya, sementara *Information Gain* membantu dalam memilih fitur yang paling relevan dan informatif. Penelitian ini bertujuan untuk meningkatkan akurasi dan efisiensi dalam mendeteksi malware *Portable Executable (PE)* yang menargetkan file eksekusi pada sistem operasi Windows. Dataset yang digunakan adalah dataset *SOMLAP* (*Swarm Optimization and Machine Learning Applied to PE Malware Detection*), yang mengandung 51409 sampel *file excutable* Windows yang diekstrasi, yang terdiri dari *file benign* (*non malware*) dan *file malware*. Hasil penelitian menunjukkan bahwa pada proporsi data 90:10, metode pemilihan 20 fitur dengan *Information Gain* berhasil meningkatkan efisiensi dan efektivitas dalam mendeteksi *malware PE* dengan rata-rata akurasi 99,3% dan rata-rata waktu pemrosesan yang diperlukan sebesar 32 detik dibandingkan dengan metode terbaik pada penelitian sebelumnya, yaitu *Ant Colony Optimization* dan *Decision Tree* yang memiliki rata-rata akurasi 98,8% dan rata-rata waktu pemrosesan sebesar 43 detik.

Kata Kunci: *malware; portable executable; decision tree; information gain; klasifikasi.*

A B S T R A C T

Malicious Software, commonly known as malware, is harmful software that can cause undesirable effects, such as data loss, information theft, dissemination of personal data, and misuse of important information. In this study, the methods used for malware detection are the Decision Tree method for classification and the Information Gain method for feature

selection. The Decision Tree method facilitates tracking and understanding decisions with its tree structure, while Information Gain helps in selecting the most relevant and informative features. This study aims to improve accuracy and efficiency in detecting Portable Executable (PE) malware targeting executable files on the Windows operating system. The dataset used is SOMLAP (Swarm Optimization and Machine Learning Applied to PE Malware Detection) dataset, which contains 51409 extracted Windows executable file samples, consisting of benign (non-malware) files and malware files. The results of the research show that with a data proportion of 90:10, the method of selecting 20 features with Information Gain successfully increased the efficiency and effectiveness in detecting PE malware with an average accuracy of 99.3% and an average processing time of 32 seconds. This result is compared to the best method in previous research, which was Ant Colony Optimization and Decision Tree, with an average accuracy of 98.8% and an average processing time of 43 seconds.

Keywords: *malware; portable executable; decision tree; information gain; classification*

1. PENDAHULUAN

Keamanan merupakan hal yang cukup penting untuk memastikan keadaan bebas dari bahaya. Pencegahan dan penanganan berbagai ancaman bahaya, seperti *malware (malicious software)*, sangatlah diperlukan dalam menjaga keamanan sistem jaringan komputer. *Malware* ini merupakan perangkat lunak atau *software* yang diciptakan untuk menyusup atau merusak sistem komputer [1]. *Malware* memang dirancang untuk mencuri data, mengganggu kinerja sistem, dan merusak sistem jaringan komputer itu sendiri. Ancaman *malware* ini terus meningkat, baik dalam kualitas maupun kuantitas, seiring kemajuan teknologi [2].

Malware tidak hanya mengancam desktop dan laptop, tetapi juga perangkat lain seperti smartphone, tablet, dan *Internet of Things* (IoT). Dengan banyaknya variasi perangkat komputer, dapat memudahkan orang yang tidak bertanggung jawab menyebarkan *malware* tanpa diketahui oleh kita. Penyebaran *malware* dapat terjadi melalui file-file atau email asing, link berbahaya, dan media sosial. Contohnya, email berisi link *malware* yang diakses penerima dapat mengaktifkan malware secara otomatis, mencuri data, atau merusak sistem komputer. Sosial media juga sering menjadi sarana penyebaran *malware* melalui postingan yang mengandung link-link berbahaya.

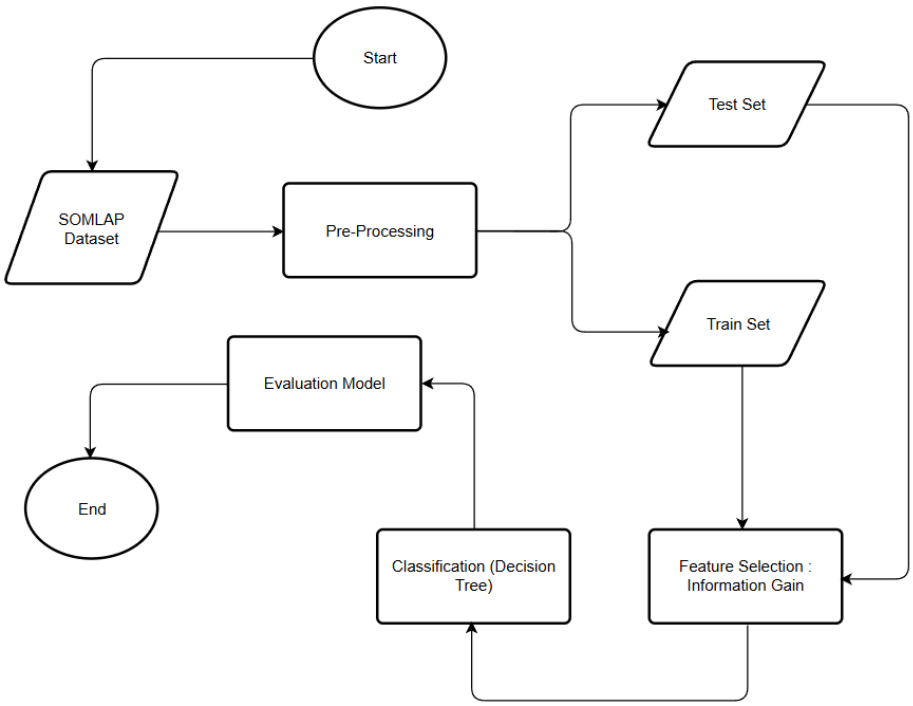
Seiring meningkatnya ancaman *malware*, sistem keamanan jaringan komputer juga mengalami peningkatan. Sistem pengujian terhadap jaringan yang dimiliki sangatlah diperlukan [3]. Dengan kemampuan untuk mengidentifikasi dan menghapus *malware*, deteksi *malware* menjadi kunci penting dalam mencegah dampak negatif pada kehidupan sehari-hari. Penelitian dalam bidang deteksi *malware* semakin banyak dilakukan, salah satunya adalah penelitian menggunakan berbagai metode Machine Learning, berbagai metode seleksi fitur, dan metode ekstraksi atribut ke dalam dataset yang mereka buat yang bertujuan untuk mendeteksi *Malware PE (Portable Executable)* [4]. Penelitian tersebut berjudul "*Swarm Optimization and Machine Learning Applied to PE (Portable Executable) Malware Detection towards Cyber Threat Intelligence*". Pada penelitian tersebut digunakan *Ant Colony Optimization* dan *Decision Tree*. Metode dalam penelitian tersebut menunjukkan akurasi yang baik dalam lingkungan pengujian yang kami pakai, yaitu 98.8%, namun waktu prosesnya masih cukup lama untuk kasus ini, yaitu 43 detik.

Penelitian ini utamanya bertujuan untuk mendapatkan waktu proses deteksi *malware* yang lebih baik dibandingkan penelitian sebelumnya, dengan akurasi yang sama baik atau lebih baik. Penelitian ini menggunakan metode *Decision Tree* sebagai model klasifikasi untuk mendeteksi *malware* dan metode *Information Gain* untuk seleksi fitur. Hasil akan dibandingkan kinerja metode *Information Gain-Decision Tree* dengan *Ant Colony Optimization-Decision Tree*.

2. METODE PENELITIAN

2.1. Desain Sistem

Pada sistem yang dirancang, tahapan yang dilewati yaitu menyiapkan *SOMLAP* Dataset, *pre-processing*, pembagian data latih dan uji, proses seleksi fitur dengan *Information Gain*, proses klasifikasi dengan *Decision Tree*, kemudian evaluasi model. Desain sistem yang dirancang digambarkan pada Gambar 1 berikut.



Gambar 1. Flowchart Desain Sistem

2.2. SOMLAP Dataset

Dataset yang digunakan pada penelitian ini yaitu *SOMLAP* (*Swarm Optimization and Machine Learning Applied to PE Malware Detection*) [4]. Dataset ini merupakan dataset yang digunakan pada penelitian yang berjudul "*Swarm Optimization and Machine Learning Applied to PE Malware Detection towards Cyber Threat Intelligence*". Dataset ini mengandung 51409 sampel *file excutable* Windows yang diekstrasi, yang terdiri dari *file benign* (*non malware*) dan *file malware*. Fitur-fitur didalamnya didapatkan sebagian besar dari dataset *ClaMP* dan dari fitur-fitur baru yang ditambahkan untuk satu dataset yang didasarkan pada header file *PE (Portable Executable)* [5][6]. Dari jumlah tersebut, 19809 (38.54%) merupakan file *malware* yang dikumpulkan oleh Virus Share, dan 31600 (61.46%) adalah file benign dan DLL yang dikumpulkan dari OS Windows 10. Modul "*pefile*" digunakan untuk ekstraksi fitur dari file

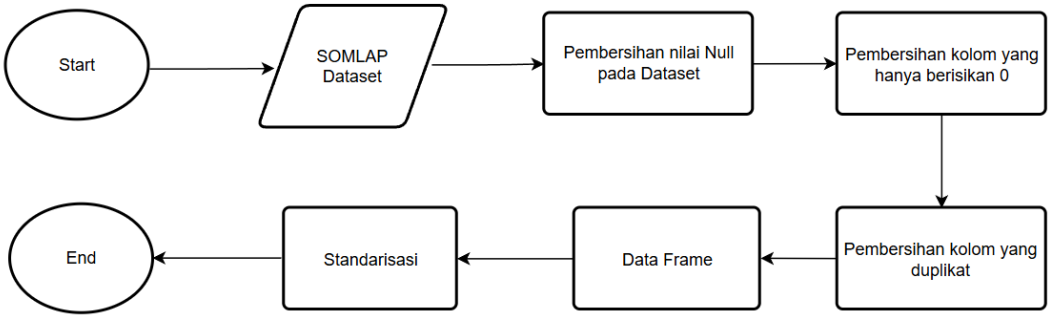
eksekusi dan DLL. Total terdapat 108 atribut dalam dataset SOMLAP. Untuk gambaran dataset *SOMLAP* yang disebutkan yaitu seperti pada Gambar 2 berikut ini.

blp	Fp	Rn	Prhdr	Minpar	Maxpar	Ivalss	Ivalsp	dosclksum	lip	Ics	Rta	Ovn	Idoem	Infoem	exehdradr	mach	nsec	tds	ptrst	stcnt	ohs	char	si
144	3	0	4	0	65535	0	184	0	0	0	64	0	0	0	240	34404	6	1	0	0	240	34	
144	3	0	4	0	65535	0	184	0	0	0	64	0	0	0	248	34404	8	1	0	0	240	34	
144	3	0	4	0	65535	0	184	0	0	0	64	0	0	0	232	34404	7	1	0	0	240	34	
144	3	0	4	0	65535	0	184	0	0	0	64	0	0	0	232	34404	6	1	0	0	240	34	
144	3	0	4	0	65535	0	184	0	0	0	64	0	0	0	248	34404	6	0	0	0	240	34	
144	3	0	4	0	65535	0	184	0	0	0	64	0	0	0	264	332	5	0	0	0	224	258	
144	3	0	4	0	65535	0	184	0	0	0	64	0	0	0	264	34404	6	0	0	0	240	8226	
144	3	0	4	0	65535	0	184	0	0	0	64	0	0	0	264	332	5	0	0	0	224	258	
144	3	0	4	0	65535	0	184	0	0	0	64	0	0	0	232	34404	7	1	0	0	240	34	
144	3	0	4	0	65535	0	184	0	0	0	64	0	0	0	200	34404	4	0	0	0	240	8226	
144	3	0	4	0	65535	0	184	0	0	0	64	0	0	0	200	34404	4	0	0	0	240	34	
144	3	0	4	0	65535	0	184	0	0	0	64	0	0	0	248	34404	7	1	0	0	240	34	
144	3	0	4	0	65535	0	184	0	0	0	64	0	0	0	232	332	5	0	0	0	224	8450	
144	3	0	4	0	65535	0	184	0	0	0	64	0	0	0	224	332	5	0	0	0	224	258	
144	3	0	4	0	65535	0	184	0	0	0	64	0	0	0	240	34404	6	1	0	0	240	34	
144	3	0	4	0	65535	0	184	0	0	0	64	0	0	0	168	34404	2	1	0	0	240	8226	
144	3	0	4	0	65535	0	184	0	0	0	64	0	0	0	184	332	2	1	0	0	224	8450	
144	3	0	4	0	65535	0	184	0	0	0	64	0	0	0	128	332	3	0	0	0	224	8462	
144	3	0	4	0	65535	0	184	0	0	0	64	0	0	0	128	332	3	0	0	0	224	8462	
144	3	0	4	0	65535	0	184	0	0	0	64	0	0	0	240	332	4	0	0	0	224	8462	
144	3	0	4	0	65535	0	184	0	0	0	64	0	0	0	232	332	4	0	0	0	224	8462	
144	3	0	4	0	65535	0	184	0	0	0	64	0	0	0	240	332	4	0	0	0	224	8462	
144	3	0	4	0	65535	0	184	0	0	0	64	0	0	0	232	332	4	0	0	0	224	8462	
144	3	0	4	0	65535	0	184	0	0	0	64	0	0	0	248	332	4	0	0	0	224	8462	
144	3	0	4	0	65535	0	184	0	0	0	64	0	0	0	248	332	4	0	0	0	224	8462	
144	3	0	4	0	65535	0	184	0	0	0	64	0	0	0	240	332	4	0	0	0	224	8462	

Gambar 2. Gambaran Dataset SOMLAP

2.3. Preprocessing

Pada tahap *preprocessing* ini akan dilakukan proses persiapan data sebelum digunakan pada model *machine learning*. Tahap *preprocessing* akan digambarkan pada Gambar 3 berikut.



Gambar 3. Flowchart Alur Tahap *Preprocessing*

Pada *preprocessing* dilakukan pengecekan apakah terdapat data yang berisikan *null*, baris/kolom nol, atau merupakan duplikat dari baris/kolom lain. Jika terdapat data seperti itu, maka data tersebut akan dihapus. Selain itu, pada *preprocessing* dilakukan juga standarisasi / normalisasi untuk menyamakan skala fitur-fitur dalam dataset. Normalisasi dilakukan dengan menggunakan StandardScaler dari library Scikit-Learn [7].

2.4. Pembagian Data Latih dan Data Uji

Pada tahap pembagian data, data yang telah melalui proses *preprocessing* akan dibagi menjadi data latih (*train set*) dan data uji (*test set*). Data latih digunakan untuk melatih model deteksi *malware* [8] agar model dapat mengklasifikasikan *benign* (*non malware*) dan *malware*. Sedangkan untuk data uji, data akan digunakan untuk pengujian suatu model deteksi *malware* agar model dapat memprediksi apakah suatu sampel data merupakan *benign* (*non malware*) atau *malware*. Untuk pembagian data, terdapat 3 rasio pembagian data (data latih: data uji), yaitu 70:30, 80:20, dan 90:10.

2.5. *Feature Selection*

Setelah tahap *preprocessing* dan pembagian data, tahap selanjutnya data akan dilakukan feature selection untuk memilih fitur-fitur mana yang akan digunakan oleh model klasifikasi. Dalam penelitian ini, *feature selection* yang digunakan yaitu *Information Gain* [9][10][11], dan juga *Ant Colony Optimization* sebagai pembanding nya, adapun metode klasifikasi yang digunakan adalah metode Pohon Keputusan (*Decision Tree*). Metode *Information Gain* dalam pemilihan fitur-fitur yang paling informatif dan relevan akan melakukan perhitungan untuk mengukur seberapa banyak informasi mengenai kelas target (*malware* atau *benign*) yang didapatkan dari setiap fitur-fitur relevan tersebut. Kemudian, untuk model klasifikasi yang digunakan dalam penelitian ini yaitu *Decision Tree*, dimana fitur-fitur relevan yang telah dipilih akan digunakan oleh *Decision Tree* [12]. Model Klasifikasi dengan *Decision Tree* akan membuat suatu keputusan dengan berulang kali berdasarkan fitur-fitur yang dipilih dan melakukan pembagian dataset menjadi *subset* yang lebih kecil agar data-data yang ada pada *subset* menjadi kelas yang sama.

2.6. Information Gain

Dengan *Information Gain* [9][10] dalam pemilihan fitur-fitur yang paling informatif dan relevan, akan melakukan perhitungan untuk mengukur seberapa banyak informasi mengenai kelas target (malware atau benign) yang didapatkan dari setiap fitur-fitur relevan tersebut. Semakin tinggi nilai *Information Gain*, semakin besar kontribusi fitur tersebut dalam pemisahan data. *Information Gain* dihitung berdasarkan konsep entropi dari teori informasi. Entropi merupakan mengukur suatu ketidakpastian dalam himpunan data.

Berdasarkan algoritma tersebut, langkah pertama akan dilakukan perhitungan *Entropi* dataset awal (S) secara menyeluruh. Kemudian hitung entropi dataset awal untuk setiap fitur dalam dataset. Dari setiap fitur tersebut yang memiliki nilai unik, akan diberi subset data berdasarkan nilai tersebut. Sehingga dari masing-masing subset, akan dihitung *entropi* berbobot dan dikurangi dari *entropi* dataset awal untuk mendapatkan nilai *Information Gain*. Setelah itu, hasil dari seluruh perhitungan *Information Gain* terhadap seluruh fitur akan diurutkan dari tertinggi ke terendah. Dan berikut adalah hasil pengukuran 20 fitur dengan *Information Gain* tertinggi berdasarkan algoritma tersebut.

Tabel 1. Daftar 20 Fitur Tertinggi

```
Top 20 fitur dengan Information Gain
tertinggi:
char              0.498620
majssver          0.439416
majlv             0.436572
dllch             0.414346
optcksum          0.410711
majosver          0.389053
nsec              0.373715
minlv             0.356264
cbase             0.320753
majiver           0.314353
Text_byteaddr     0.291916
exehdradr         0.285196
Rsrc_mscfaddr     0.280689
Rsrc_secsize      0.279948
Rsrc_entro        0.267716
Text_entro        0.261836
dbase             0.247983
adrentpt          0.237983
```

2.7. Decision Tree

Pohon Keputusan atau disebut juga *Decision Tree* [13] merupakan algoritma pembelajaran terarah yang digunakan untuk klasifikasi dan regresi. Algoritma ini bekerja dengan membagi data menjadi *subset* berdasarkan nilai dari fitur input. Proses ini bersifat rekursif dan berlanjut hingga algoritma menentukan bahwa pembagian lebih lanjut tidak diperlukan atau tidak memungkinkan.

Langkah awal yang dilakukan yaitu mengecek apakah terdapat kelas yang sama pada *instance* dan akan mengembalikan *node* daun dengan kelas tersebut jika terdapat kelas yang sama. Lalu, jika tidak ada fitur yang dapat dibagi, maka *node* daun dengan kelas paling umum pada dataset akan dikembalikan, dan fitur terbaik akan ditentukan. Selain itu, fitur terbaik tersebut akan digunakan untuk membuat *node* keputusan yang dapat menentukan subset dataset untuk setiap nilai unik dari fitur yang dipilih. Sehingga pohon keputusan dapat dibangun secara rekursif berdasarkan *node* keputusan yang diperoleh. Hasil akhir dari algoritma *Decision Tree* yaitu model berbentuk pohon yang terdiri dari keputusan-keputusan.

2.8. Metrik Evaluasi

Pada tahap ini, kinerja dari model yang telah dilatih akan dievaluasi menggunakan beberapa metrik evaluasi. Metrik evaluasi yang digunakan adalah *accuracy*, *precision*, *recall*, dan *F1-Score*. Berikut adalah formula dari *accuracy*, *precision*, *recall*, dan *F1-Score*, dengan TP (*True Positives*) merupakan banyak *malware* yang terdeteksi sebagai *malware*, TN (*True Negatives*) merupakan banyak bukan *malware* yang terdeteksi sebagai bukan *malware*, FP (*False Positives*) merupakan banyak bukan *malware* yang salah terdeteksi sebagai malware, dan FN (*False Negatives*) merupakan banyak *malware* yang salah terdeteksi sebagai sampel bersih.

- *Accuracy* digunakan untuk mengukur seberapa baik model dalam memprediksi target, baik sebagai *malware* ataupun bukan *malware*, dengan benar,

$$accuracy = \frac{TP+TN}{TP+TN+FP+FN}. \quad (1)$$

- *Precision* digunakan untuk mengukur seberapa baik model dalam memprediksi target yang positif (*malware*) sebagai *malware* dengan benar,

$$Precision = \frac{TP}{TP+FP}. \quad (2)$$

- *Recall* digunakan untuk mengukur seberapa baik model dalam menemukan target positif (*malware*) dari semua *malware* yang ada,

$$Recall = \frac{TP}{TP+FN}. \quad (3)$$

- *F1-Score* adalah formula yang digunakan untuk mengombinasikan *precision* dan *recall*, memperhitungkan gambaran keseimbangan dari *precision* dan *recall*,

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}. \quad (4)$$

3. HASIL DAN PEMBAHASAN

Sebagaimana telah disebutkan pada subbab 2.1, terdapat 3 skenario pembagian data, yaitu 70:30, 80:20, dan 90:10. Untuk skenario 1, dilakukan pengujian dengan pembagian data 70% untuk data latih dan 30% untuk data uji. Data latih yang diambil sebanyak 35985 baris dan 108 kolom, dan data uji yang diambil sebanyak 15423 baris dan 108 kolom, dari total keseluruhan yaitu 51409 baris dan 108 kolom. Untuk skenario 2, dilakukan pengujian dengan pembagian data 80% untuk data latih dan 20% untuk data uji. Data latih yang diambil sebanyak 41126 baris dan 108 kolom, dan data uji yang diambil sebanyak 10282 baris dan 108 kolom, dari total keseluruhan yaitu 51409 baris dan 108 kolom. Untuk skenario 3, akan dilakukan pengujian dengan pembagian data 90% untuk data latih dan 10% untuk data uji. Data latih yang diambil sebanyak 46267 baris dan 108 kolom, dan data uji yang diambil sebanyak 5141 baris dan 108 kolom, dari total keseluruhan yaitu 51409 baris dan 108 kolom.

Pada masing-masing skenario pembagian data, dijalankan 3 kombinasi algoritma yang berbeda, yaitu IG-DT (*Information Gain - Decision Tree*), ACO-DT (*Ant Colony Optimization - Decision Tree*), dan DT (*Decision Tree*) tanpa seleksi fitur. Selain itu, pada pengujian ini, IG-DT akan ditargetkan untuk mencari fitur sebanyak 20 fitur dengan cara mengambil 20 fitur dengan nilai Gain tertinggi dari keseluruhan fitur. Sedangkan untuk ACO-DT akan ditargetkan untuk mencari fitur sebanyak 20 fitur dengan cara mengatur n_ants (jumlah semut) sebanyak 10, n_best (jumlah semut terbaik yang digunakan untuk memperbarui tingkat pheromone) sebanyak 5, n_iterations (jumlah total iterasi yang akan dijalankan) sebanyak 15, alpha dan beta sebanyak 1, decay (pengurangan tingkat pheromone secara bertahap setiap iterasi) sebesar 0.95, dan target_features (penargetan jumlah fitur) sebanyak 20. Penargetan jumlah seleksi fitur sebanyak 20 bertujuan agar melakukan perbandingan klasifikasi IG-DT dengan ACO-DT seimbang dan adil, meskipun pada penelitian paper acuan yang berjudul "*Swarm Optimization and Machine Learning Applied to PE Malware Detection towards Cyber Threat Intelligence*" terdapat pengambilan jumlah fitur yang acak antar klasifikasi. Setelah skenario tersebut dijalankan, berikutnya akan dilakukan pengujian 10 kali running dari masing-masing ketiga skenario tersebut yang bertujuan untuk menghitung rata-rata dari masing-masing skenario. Hasil pengujian *accuracy*, precision, recall, dan F1-score dapat dilihat pada Tabel 2, sedangkan hasil pengujian waktu dapat dilihat pada Tabel 3.

Tabel 2. Tabel Hasil Pengujian

Pengujian		Rata-Rata Nilai		
		IG-DT	ACO-DT	DT
Proporsi data 70:30	*Accuracy*	99.199%	98.671%	99.131%
	Precision	99.141%	98.571%	99.078%
	Recall	98.753%	97.935%	98.639%
	F1-Score	98.947%	98.252%	98.858%
Proporsi data 80:20	*Accuracy*	99.246%	98.787%	99.144%
	Precision	99.205%	98.721%	99.205%
	Recall	98.818%	98.095%	98.548%
	F1-Score	99.011%	98.407%	98.876%
Proporsi data 90:10	*Accuracy*	99.294%	98.761%	99.300%
	Precision	99.299%	98.688%	99.390%
	Recall	98.862%	98.084%	98.787%
	F1-Score	99.080%	98.385%	99.087%

Tabel 3. Tabel Hasil Pengujian Waktu

Pengujian		Rata-Rata Nilai		
		IG-DT	ACO-DT	DT
Proporsi data 70:30	Waktu seleksi fitur	24.232 Detik	33.791 Detik	-
	Waktu *training*	0.398 Detik	0.230 Detik	0.885 Detik
	Waktu *testing*	0.005 Detik	0.004 Detik	0.007 Detik
	Waktu Keseluruhan	24.650 Detik	34.037 Detik	0.892 Detik
Proporsi data 80:20	Waktu seleksi fitur	27.793 Detik	37.046 Detik	-
	Waktu *training*	0.425 Detik	0.257 Detik	1.061 Detik
	Waktu *testing*	0.004 Detik	0.004 Detik	0.006 Detik
	Waktu Keseluruhan	28.236 Detik	37.318 Detik	1.067 Detik
Proporsi data 90:10	Waktu seleksi fitur	31.698 Detik	42.782 Detik	-
	Waktu *training*	0.524 Detik	0.283 Detik	1.209 Detik
	Waktu *testing*	0.004 Detik	0.003 Detik	0.004 Detik
	Waktu Keseluruhan	32.237 Detik	43.079 Detik	1.213 Detik

3.1. Analisis Hasil Pengujian

Pada Tabel 2 dapat dilihat bahwa algoritma yang memiliki *accuracy* dan *F1-score* tertinggi adalah DT, dengan nilai *accuracy* 99.3% dan *F1-score* 99.87%. Hasil tersebut terjadi pada proporsi data 90:10. Hasil tersebut dapat dikatakan terjadi anomali, karena pada umumnya algoritma yang dikombinasikan dengan seleksi fitur yang akan mencapai hasil lebih tinggi. Analisis kami terhadap mengapa anomali tersebut terjadi adalah diantaranya karena terdapat fitur yang tampak lemah secara individu dan tereliminasi saat seleksi fitur, namun sebenarnya fitur tersebut menjadi penting jika dikombinasikan dengan fitur lain [14]. Kemudian secara umum DT dapat manangani sendiri fitur yang tidak relevan tanpa bantuan metode seleksi fitur lain, dan menghilangkan fitur yang tidak relevan pada tahap sebelum diproses oleh DT terkadang dapat menurunkan performa DT [15][16].

Pada Tabel 2 juga dapat dilihat bahwa rata-rata hasil pengujian IG-DT lebih baik daripada ACO-DT, dan pada Tabel 3 dapat dilihat rata-rata waktu seleksi fitur IG-DT lebih cepat daripada ACO-DT. Hal ini dikarenakan ACO melibatkan banyak iterasi dan evaluasi solusi untuk menemukan kombinasi fitur optimal, sedangkan IG hanya memerlukan penghitungan entropi secara langsung, sehingga ACO [17] memiliki kecepatan yang lebih lambat dibandingkan IG [18]. Selain itu, metode IG secara efektif memilih fitur-fitur yang paling informatif berdasarkan keterkaitan nya dengan target, sedangkan ACO dapat berisiko mengalami overfitting terutama pada dataset besar, meskipun memiliki kemampuan eksplorasi yang baik, sehingga IG cenderung menghasilkan *accuracy* yang lebih baik daripada ACO untuk kasus pada penelitian ini.

4. KESIMPULAN

Decision Tree dan *Information Gain* (IG-DT) lebih efektif dalam deteksi *malware*, yang ditunjukkan dari *accuracy*, *F1-score*, dan waktu proses yang lebih unggul dibandingkan dengan *Decision Tree* dan *Ant Colony Optimization* (ACO-DT). Meskipun penggunaan ACO sebagai metode seleksi fitur cukup efisien dalam mendeteksi *malware*, namun penggunaan IG sebagai metode seleksi fitur menunjukkan bahwa fitur-fitur dengan nilai IG tertinggi memberikan kontribusi signifikan terhadap kinerja model, serta memberikan hasil *accuracy, F1-score,* dan

efisiensi waktu proses yang lebih baik. Hal ini terbukti dari hasil pengujian yang ditunjukkan dari pembagian data 90:10, metode seleksi fitur IG, pada IG-DT, berhasil meningkatkan efisiensi dan efektivitas dalam mendeteksi *malware* PE dengan rata-rata *accuracy* 99.3%, rata-rata *F1-score* 99.08%, dan rata-rata waktu proses yang diperlukan sebesar 32 detik, dibandingkan dengan metode seleksi fitur terbaik yang digunakan pada penelitian sebelumnya, yaitu ACO, pada ACO-DT, yang memiliki rata-rata *accuracy* 98,78%, rata-rata *F1-score* 98.407% dan rata-rata waktu proses sebesar 34.037 detik. Untuk penelitian selanjutnya pada kasus di penelitian ini adalah dapat dicoba dengan menggunakan metode klasifikasi lain, seperti *ensemble deep learning*. Metode tersebut memiliki potensi menghasilkan hasil yang lebih baik dalam mengklasifikasikan. Kemudian untuk seleksi fitur dapat dicoba diantaranya seleksi fitur dari kategori *filter methods*, untuk mendapatkan waktu proses yang cepat.

DAFTAR PUSTAKA

[1] Y. Ilhamdi and Y. N. Kunang, "Analisis Malware Pada Sistem Operasi Windows Menggunakan Teknik Forensik," in *Bina Darma Conference on Computer Science (BDCCS)*, 2021, pp. 256–264.

[2] N. A. Nurfauzi, "Deteksi serangan malware pada cloud server menggunakan metode anomaly based," Universitas Islam Negeri Maulana Malik Ibrahim, 2022.

[3] Y. W, Y. B. Fitriana, S. Esabela, and F. Hamdani, "Deteksi Serangan Malware Pada Web Aplikasi Menggunakan Metode Malware Analis Dinamis dan Statis," *Digital Transformation Technology*, vol. 4, no. 1, pp. 461–470, Jul. 2024, doi: 10.47709/DIGITECH.V4I1.4270.

[4] D. Perakovi *et al.*, "Swarm Optimization and Machine Learning Applied to PE Malware Detection towards Cyber Threat Intelligence," *Electronics 2023, Vol. 12, Page 342*, vol. 12, no. 2, p. 342, Jan. 2023, doi: 10.3390/ELECTRONICS12020342.

[5] A. Kumar, K. S. Kuppusamy, and G. Aghila, "A learning model to detect maliciousness of portable executable using integrated feature set," *Journal of King Saud University - Computer and Information Sciences*, vol. 31, no. 2, pp. 252–265, Apr. 2019, doi: 10.1016/J.JKSUCI.2017.01.003.

[6] D. W. Asry, E. Siswanto, D. Kurniawan, and H. I. Huda, "Deteksi Malware Statis Menggunakan Deep Neural Networks Pada Portable Executable," *Teknik: Jurnal Ilmu Teknik dan Informatika*, vol. 3, no. 1, pp. 19–34, May 2023, doi: 10.51903/TEKNIK.V3I1.325.

[7] R. Latifah, E. S. Wulandari, and P. E. Kreshna, "Model Decision Tree Untuk Prediksi Jadwal Kerja Menggunakan Scikit-Learn," *Prosiding Semnastek*, 2019.

[8] M. Hazri, "Analisis Malware PlasmaRAT dengan Metode Reverse Engineering," *Jurnal Rekayasa Teknologi Informasi (JURTI)*, vol. 4, no. 2, pp. 192–199, Nov. 2020, doi: 10.30872/JURTI.V4I2.4131.

[9] R. Kesuma Dinata, H. Novriando, N. Hasdyna, S. Retno, J. Hadari Nawawi, and K. Barat, "Reduksi Atribut Menggunakan Information Gain untuk Optimasi Cluster Algoritma K-Means," *JEPIN (Jurnal Edukasi dan Penelitian Informatika)*, vol. 6, no. 1, pp. 48–53, Apr. 2020, doi: 10.26418/JP.V6I1.37606.

[10] M. R. Hasibuan and M. Marji, "Pemilihan Fitur dengan Information Gain untuk Klasifikasi Penyakit Gagal Ginjal menggunakan Metode Modified K-Nearest Neighbor (MKNN)," *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, vol. 3, no. 11,

pp. 10435–10443, 2019, Accessed: Jul. 21, 2025. [Online]. Available: https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/6691

[11] A. Bijaksana, P. Negara, H. Muhardi, and I. M. Putri, "Analisis Sentimen Maskapai Penerbangan Menggunakan Metode Naive Bayes dan Seleksi Fitur Information Gain," *Jurnal Teknologi Informasi dan Ilmu Komputer*, vol. 7, no. 3, pp. 599–606, May 2020, doi: 10.25126/JTIIK.2020711947.

[12] K. Halim, Dyah, E. Herwindiati, and T. Sutrisno, "Penerapan Metode Decision Tree untuk Prakiraan Cuaca Kota Bekasi," *Jurnal Ilmu Komputer dan Sistem Informasi*, vol. 11, no. 2, Aug. 2023, doi: 10.24912/JIKSI.V11I2.26026.

[13] H. Blockeel, L. Devos, B. Frénay, G. Nanfack, and S. Nijssen, "Decision trees: from efficient prediction to responsible AI," *Front Artif Intell*, vol. 6, p. 1124553, Jul. 2023, doi: 10.3389/FRAI.2023.1124553/XML.

[14] M. Kuhn and K. Johnson, "Applied predictive modeling," *Applied Predictive Modeling*, pp. 1–600, Jan. 2013, doi: 10.1007/978-1-4614-6849-3/COVER.

[15] T. Hastie, R. Tibshirani, and J. Friedman, "The Elements of Statistical Learning," 2009, doi: 10.1007/978-0-387-84858-7.

[16] I. Guyon and A. Elisseeff, "An Introduction to Variable and Feature Selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003, doi: 10.5555/944919.

[17] L. Lhotská, M. Macaš, and M. Burša, "PSO and ACO in Optimization Problems," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 4224 LNCS, pp. 1390–1398, 2006, doi: 10.1007/11875581_165.

[18] E. Odhiambo Omuya, G. Onyango Okeyo, and M. Waema Kimwele, "Feature Selection for Classification using Principal Component Analysis and Information Gain," *Expert Syst Appl*, vol. 174, p. 114765, Jul. 2021, doi: 10.1016/J.ESWA.2021.114765.