KLASIFIKASI SERANGAN DDOS MENGGUNAKAN ALGORITMA *SUPPORT VECTOR MACHINE* DAN *CORRELATION-BASED*

I Nyoman Arista Wisnawa[a1], I Made Widiartha[a2], I Made Widhi Wirawan[a3], Made Agung Raharja[a4]

[a]Program Studi Informatika, Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Udayana, Indonesia
[1]wisnawa.2208561082@student.unud.ac.id
[2]madewidiartha@unud.ac.id
[3]made_widhi@unud.ac.id
[4]made.agung@unud.ac.id

Abstract

Distributed Denial-of-Service (DDoS) that attacks targeting server resources are increasing every year, making early detection crucial. While network traffic analysis can aids detection, high dimensionality data and noise make manual efforts more difficult. This research implements a Support Vector Machine (SVM) with Correlation-Based Feature Selection (CFS) to reduce dimensionality, comparing it against a full-feature SVM model. Both use Radial Basis Function (RBF) kernel with hyperparameter C = 97 and gamma = 0.74. The CFS-SVM model will implemented in real-time. Performance is evaluated using confusion matrix (recall, precision, F1-Score, and accuracy) and computational time needed. Results show the CFS SVM model achieves 99.55% recall, 97.28% precision, 98.40% F1-Score, and 98.40% accuracy, needing 310.73s for training and 0.21s for testing. In Contrast, the full-feature model yields 98.89% for recall, 99.95% precision, 99.42% F1-Score, and 99.42% accuracy, but taking 438.96s for training proses and 0.29s for testing. Although the full-feature model exhibits superior metrics, applying CFS significantly reduces computational time and the computational load of feature extraction. Therefore, the CFS-SVM approach has proven to be more suitable for real-time detection systems.

Keywords: *Classification, Distributed Denial-of-Service, Correlation-Based Feature Selection, SVM, real-time detection, network security*

1. Pendahuluan

Distributed Denial-of-Service atau DDoS merupakan salah satu ancaman serius untuk infrastruktur jaringan saat ini. Berbeda dengan pendahulunya yaitu Denial of Service (DoS), serangan DDoS ini menargetkan suatu *website* atau server, memanfaatkan banyak perangkat *bot* untuk menghabiskan *resource* server atau *website* agar layanan *website* atau server agar menjadi terganggu atau *down* [1][2]. Dari laporan yang di publikasi oleh beberapa penyedia layanan global, terlihat pola kenaikan volume serangan yang signifikan yang bahkan mencapai lebih dari jutaan insiden pada awal tahun 2025 [3][4]. Seperti yang dicatat oleh Gcore pada Q1 dan Q2 tahun 2025, terdapat peningkatan serangan yang mencapai 1.173 juta serangan [5]. Peningkatan serangan ini memerlukan waktu respons yang tinggi, sehingga menuntut adanya mekanisme deteksi dini yang tidak hanya akurat tetapi juga mampu memproses dan memitigasi lalu lintas data secara *real-time* untuk meminimalisir dampak serangan.

Namun karakteristik data *traffic* jaringan menjadi kendala untuk sistem deteksi tersebut. Hal ini karena data tersebut umumnya memiliki dimensi yang tinggi dan berisi atribut yang redundan. Berbagai metode *machine learning* dan bahkan *deep learning* telah diuji untuk melakukan deteksi pola serangan DDoS. Sayangnya, walaupun banyak yang mencapai hasil akurasi yang bagus, model-model yang digunakan banyak yang memiliki komputasi yang berat. Dalam implementasi *real-time*, model tidak cukup hanya dengan memiliki akurasi tinggi tetapi harus memiliki beban komputasi yang juga ringan.

Support Vector Machine (SVM) merupakan salah satu metode klasifikasi *machine learning* yang ringan dan efektif untuk klasifikasi data dengan dimensi tinggi dengan kompleksitas nonlinear [6].

p-ISSN: 2301-5373
e-ISSN: 2654-5101

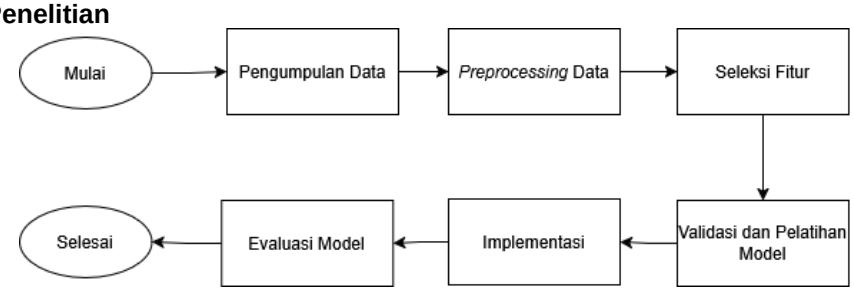
Walaupun demikian, performa SVM masih rentan terhadap data dengan dimensi yang terlalu tinggi, pelatihan model dengan terlalu banyak fitur akan memperlambat penentuan hyperplane serta akan berpengaruh pada waktu prediksi model. Oleh karena itu, dibutuhkan seleksi fitur seperti *Correlation-Based Feature Selection* (CFS) untuk memilih fitur-fitur yang paling relevan untuk membantu mengurangi beban komputasi [2]. Dengan pemilihan fitur yang sesuai dan parameter yang diatur dengan baik akan sangat mempengaruhi performa klasifikasi dari metode SVM [7]. CFS bekerja dengan mengurangi data redundan dengan membandingkan keterkaitan antar fitur dan keterkaitan fitur-fitur tersebut dengan label. CFS akan membuang salah satu dari dua fitur yang memiliki keterkaitan tinggi antar fitur tetapi juga memiliki keterkaitan tinggi dengan label, juga akan menghilangkan fitur yang memiliki kaitan yang rendah dengan label.

Sejumlah penelitian telah membuktikan efektivitas seleksi fitur dan *machine learning*, seperti pada penelitian yang berjudul "Klasifikasi Serangan *Distributed Denial of Service* (DDoS) Menggunakan *Random Forest* dengan CFS" membahas tentang penerapan algoritma *random forest* dengan seleksi fitur *correlation-based* yang mengalami kenaikan nilai akurasi rata-rata sebesar 99,784% [8]. Selain seleksi fitur, pemilihan kernel pada SVM juga dapat mempengaruhi kinerja dari model SVM. Pada penelitian yang berjudul "*Software Defined Network : The Comparison of* SVM *kernel on* DDoS *Detection*", membahas tentang perbandingan kinerja model SVM terhadap penggunaan jenis kernel pada SVM. Dari penelitian tersebut, kernel SVM yang memiliki akurasi tertinggi adalah kernel *Radial Basis Function* atau RBF. Nilai SVM dengan kernel RBF ini mendapatkan nilai akurasi 99,26% [6]. Meskipun akurasi pada penelitian-penelitian tersebut sangat tinggi, tetapi sebagian besar hanya menjadikan akurasi sebagai tolak ukur utama keberhasilan. Pada konteks deteksi DDoS *real-time*, memastikan model minim menghasilkan *false negative* serta memiliki waktu komputasi yang cepat juga sangat penting. Penggunaan metrik yang lebih sensitive seperti *recall* dan *F1-Score* lebih relevan. Maka dari itu, dalam penelitian ini dilakukan evaluasi terhadap nilai recall, F1-Score, dan beban komputasi model.

Penggunaan algoritma SVM dengan seleksi fitur *Correlation-Based* untuk melakukan klasifikasi serangan DDoS serta penyeimbangan antara akurasi deteksi dan beban komputasi ini diharapkan dapat memberikan *insight* dan strategi baru bagi para ahli atau profesional dalam mendeteksi serangan DDoS yang efisien pada scenario *real-time*.

2. Metode Penelitian

2.1. Desain Penelitian



Gambar 1. Alur pelaksanaan penelitian

Pada metodologi penelitian ini akan menjelaskan langkah-langkah yang dilakukan dalam melakukan klasifikasi serangan DDoS menggunakan algoritma SVM dan seleksi fitur *Correlation-based*. Langkah-langkah tersebut dapat dilihat pada gambar 1 yang meliputi pengumpulan data, *preprocessing* data, seleksi fitur, validasi dan pelatihan model, implementasi, dan evaluasi model.

2.2. Pengumpulan Data

Penelitian ini menggunakan data sekunder, yakni data yang diperoleh dari sumber-sumber lain, seperti repositori online, basis data publik, atau laporan penelitian sebelumnya. Data sekunder penelitian tentang serangan DDoS yang digunakan peneliti diperoleh dari situs Kaggle yaitu CIC-DDoS2019, dikeluarkan atau dibuat oleh Canadian Institute of Cybersecurity. Dataset CIC-DDoS2019 memuat total 50.063.112 catatan *traffic*, yang jika diuraikan yaitu 50.006.249 serangan DDoS dan 56.863 *traffic* normal. Dataset berisi 17 jenis serangan DDoS dengan 87 fitur *traffic* jaringan yang telah di ekstraksi.

Tabel 1. Sampel dataset

Flow ID	Source IP	Source Port	Destination IP	...	Label
172.16.0.5-192.168.50.4-35468-49856-17	172.16.0.5	35468	192.168.50.4	...	UDP
172.16.0.5-192.168.50.4-44167-44225-17	172.16.0.5	44167	192.168.50.4	...	UDP
172.16.0.5-192.168.50.4-36215-28771-17	172.16.0.5	36215	192.168.50.4	...	UDP
192.168.10.50-192.168.50.8-21-60319-6	192.168.50.8	60319	192.168.10.50	...	BENIGN
192.168.50.253-224.0.0.5-0-0-0	192.168.50.253	0	224.0.0.5	...	BENIGN

2.3. Preprocessing Data

Preprocessing data merupakan proses untuk menghilangkan berbagai permasalahan pada data yang akan digunakan sehingga tidak mengalami kendala saat pemrosesan data. Ada beberapa proses yang dilakukan yaitu *data cleaning*, penghapusan kolom metadata, data sampling, serta *data transformation and columns header normalization*. *Data cleaning* memastikan data yang digunakan tidak memiliki kesalahan sehingga siap untuk digunakan untuk proses analisis [9]. Pada penelitian ini, dataset akan dibersihkan dari data-data yang hilang atau kosong. Selain data yang hilang, peneliti juga akan mengeliminasi data yang bernilai *null* serta data yang tidak diperlukan dalam proses klasifikasi nantinya.

Dataset terutama *traffic* jaringan biasanya berisi kolom atau data berupa metadata yang tidak penting atau bisa dikatakan sebagai sampah untuk model *machine learning*. Proses ini dilakukan dengan menganalisa fitur-fitur atau kolom pada dataset. Tujuan dari penghapusan kolom-kolom metadata ini adalah untuk menghindari bias terutama pada saat proses seleksi fitur, selain itu juga menghindari beban komputasi yang tidak diperlukan jika fitur-fitur metadata ini digunakan untuk proses pelatihan ataupun proses evaluasi model.

Kemudian, dilakukan sampling data dengan menggunakan *stratified* dan *random sampling*. *Stratified sampling* mengelompokkan data menjadi kelompok-kelompok yang ditentukan berdasarkan kesamaan atau homogenitas data [10]. kemudian dilakukan *random sampling* untuk dijadikan sebuah sampel dari setiap kelompok yang ditentukan tersebut. Sampel data yang diperoleh untuk setiap kelompok kemudian digabungkan menjadi satu sampel data. Kemudian, dilakukan *data transformation* untuk mengubah format data agar sesuai dengan format data yang diperlukan serta normalisasi nama kolom seperti menghilangkan spasi yang ada pada awal nama kolom dan mengganti simbol-simbol seperti “/” atau spasi pada nama kolom yang berupa 2 kata atau lebih dengan “_”.

2.4. Correlation-Based Feature Selection (CFS)

Correlation-based feature selection atau yang disingkat dengan CFS merupakan salah satu metode untuk seleksi fitur pada sebuah dataset dengan melakukan suatu perhitungan serta perbandingan tingkat korelasi antara fitur dengan kelas serta dengan fitur lainnya. Nantinya dari hasil perbandingan tersebut, fitur akan dipilih berdasarkan yang memiliki nilai korelasi tinggi dengan kelasnya tetapi memiliki korelasi yang rendah dengan fitur lainnya [11].

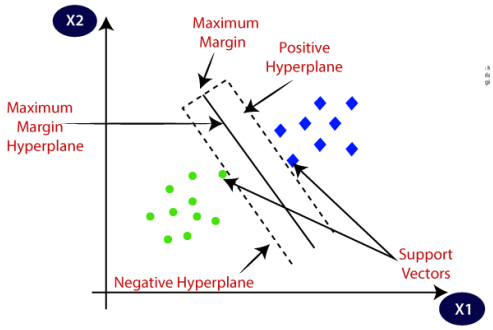
Pemilihan fitur yang memiliki korelasi yang rendah dengan fitur yang lainnya tersebut bertujuan untuk mengurangi redundansi. Fitur redundan tersebut cenderung membawa informasi yang mirip satu sama lain sehingga tidak memberikan manfaat yang signifikan. Dengan berkurangnya fitur yang digunakan, hal ini akan membuat model lebih sederhana secara kompleksitas dan meningkatkan efisiensi model. CFS terbukti menghasilkan banyak hasil yang signifikan untuk membantu dalam meningkatkan performa dari proses analisis pengambilan keputusan [12].

$$cr_{zc} = \frac{\mathrm{f}\overline{cr_{zi}}}{\sqrt{f+f(f-1)\overline{cr_{ii}}}} \quad (1)$$

p-ISSN: 2301-5373
e-ISSN: 2654-5101

2.5. Support Vector Machine (SVM)

Support Vector Machine atau SVM merupakan sebuah metode *supervised learning* yang digunakan untuk klasifikasi dan regresi [13]. Cara kerja dari metode SVM sendiri adalah dengan mencari sebuah *hyperplane* sebaik mungkin untuk memisahkan dua kelas dengan mencari jarak paling jauh antar kelas [14]. Pada SVM, terdapat beberapa hal mendasar yang harus diketahui, yaitu *hyperplane*, margin, *support vector*, dan *kernel trick*.



Gambar 2. Ilustrasi klasifikasi dengan SVM [15]

a. *Hyperplane* merupakan bidang pemisah antara kelas titik data [16]. Dalam suatu kasus akan terdapat beberapa kemungkinan *hyperplane* yang dapat dipilih. *Hyperplane* ini ditujukan untuk mencari margin maksimum antar kelas. *Hyperplane* secara umum dapat dirumuskan sebagai persamaan berikut:

$$w.x_i + b = 0 \quad (2)$$

b. Margin merupakan jarak yang memisahkan antar titik data terdekat dari *hyperplane*. Jarak margin yang dicari nantinya merupakan jarak margin maksimal. Jarak margin maksimal akan membantu dalam mengklasifikasikan titik data uji dengan baik. Secara umum margin dapat dicari dengan persamaan berikut:

$$\frac{2}{||w||} \quad (3)$$

c. *Support vector* merupakan titik data terluar yang paling dekat dengan *hyperplane*. *Support vector* ini digunakan dalam SVM untuk membantu menemukan *hyperplane* yang paling optimal

d. *Kernel trick* merupakan teknik untuk mencari *hyperplane* pada data yang memiliki karakteristik nonlinear. Pada penelitian ini akan menggunakan kernel RBF untuk klasifikasi serangan. Penggunaan kernel ini dikarenakan pada penelitian yang dilakukan oleh (Perwira & Prapcoyo, 2021), kernel RBF menghasilkan nilai akurasi yang paling tinggi dibandingkan dengan kernel yang lain. Berikut merupakan persamaan kernel RBF:

$$K(x_1, x_2) = \exp(-\gamma||x_1 - x_2||^2) \quad (4)$$

2.6. K-Fold Cross Validation

K-fold Cross Validation (CV) merupakan metode evaluasi yang biasanya digunakan untuk mengetahui jika terdapat masalah seperti bias atau overfitting pada model [17]. CV bekerja dengan membagi dataset menjadi beberapa bagian (*K-fold*), satu *fold* akan menjadi data tes dan data lainnya akan menjadi data latih, seluruh data pada dataset akan pernah setidaknya sekali menjadi data tes, kemudian dicari rata-rata hasilnya.

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Percobaan 1	Data Test	Data Train	Data Train	Data Train	Data Train
Percobaan 2	Data Train	Data Test	Data Train	Data Train	Data Train
Percobaan 3	Data Train	Data Train	Data Test	Data Train	Data Train
Percobaan 4	Data Train	Data Train	Data Train	Data Test	Data Train
Percobaan 5	Data Train	Data Train	Data Train	Data Train	Data Test

Gambar 3. Ilustrasi *Cross Validation*

2.7. Confusion Matrix

Confusion matrix merupakan salah satu metode yang sering digunakan untuk mengevaluasi kinerja dari suatu model. Untuk melakukan evaluasi model, akan membandingkan hasil klasifikasi real dengan hasil prediksi dari model [18]. Dari *confusion matrix* juga peneliti dapat mengetahui akurasi, presisi, nilai *recall*, dan *F1-Score* dari sistem klasifikasi

Tabel 2. *Confusion matrix*

		Predicted	
		True	*False*
Actual	*True*	*True Positives* (TP)	*False Negatives* (FN)
	False	*False Positives* (FP)	*True Negatives* (TN)

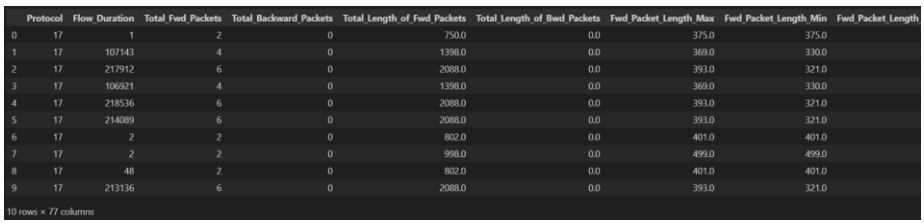
2.8. Black Box Testing

Black box testing digunakan untuk memprediksi masalah kesalahan fungsi dan kesalahan pada antarmuka [19]. Metode pengujian ini tidak membutuhkan pengetahuan yang mendalam tentang programing ataupun internal dari program [20]. Pengujian dengan *black box testing* berfokus pada *input* dan *output* dari program untuk memastikan program berjalan sesuai dengan yang seharusnya.

3. Hasil dan Pembahasan

3.1. Preprocessing

Dari hasil *preprocessing*, tahap-tahap yang sebelumnya dijelaskan yaitu *data cleaning*, penghapusan kolom metadata, data sampling, serta *data transformation and columns header normalization* dilakukan agar data bersih dari nilai hilang dan memiliki format data dan penamaan kolom yang sama sehingga tidak terjadi *error* saat digunakan dalam proses pelatihan model. Berikut pada gambar 4 merupakan data hasil proses *preprocessing* yang dilakukan.



	Protocol	Flow_Duration	Total_Fwd_Packets	Total_Backward_Packets	Total_Length_of_Fwd_Packets	Total_Length_of_Bwd_Packets	Fwd_Packet_Length_Max	Fwd_Packet_Length_Min	Fwd_Packet_Length
0	17	1	2	0	750.0	0.0	375.0	375.0	
1	17	107143	4	0	1398.0	0.0	369.0	330.0	
2	17	217912	6	0	2088.0	0.0	393.0	321.0	
3	17	106921	4	0	1398.0	0.0	369.0	330.0	
4	17	218536	6	0	2088.0	0.0	393.0	321.0	
5	17	214089	6	0	2088.0	0.0	393.0	321.0	
6	17	2	2	0	802.0	0.0	401.0	401.0	
7	17	2	2	0	998.0	0.0	499.0	499.0	
8	17	48	2	0	802.0	0.0	401.0	401.0	
9	17	213136	6	0	2088.0	0.0	393.0	321.0	

10 rows × 77 columns

Gambar 4. Sampel dataset hasil *preprocessing batch* 1

3.2. Seleksi Fitur

Setelah *preprocessing*, dilakukan seleksi fitur *correlation-based* (CFS) terhadap dataset final setelah semua dataset hasil *preprocessing* digabungkan. Hasil seleksi fitur mendapati terpilihnya 8 fitur dengan nilai CFS mencapai 0.7669, tetapi karena terdapat satu fitur yang redundan dengan fitur lain, salah satu fitur tersebut dihapus sehingga hanya tersisa 7 fitur. Fitur-fitur yang terpilih yaitu

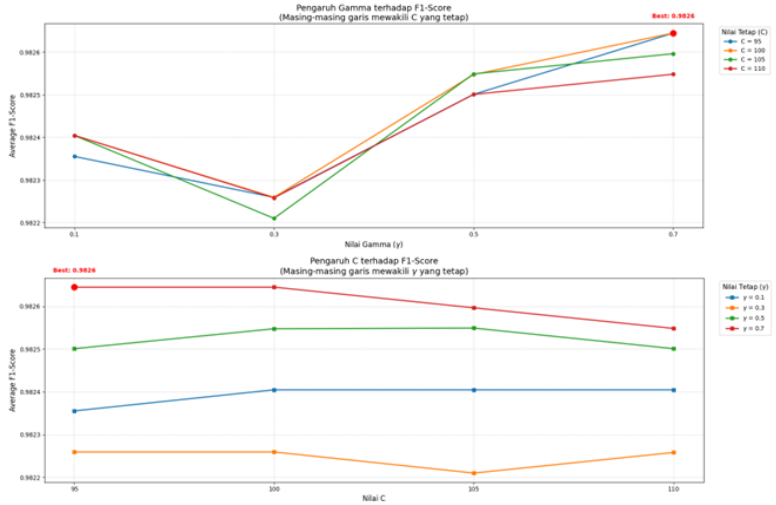
Tabel 3. Fitur yang terpilih

No.	Fitur yang terpilih
1.	*Min_Packet_Length*
2.	*URG_Flag_Count*
3.	*Fwd_Packets_s*
4.	*Bwd_Packet_Length_Min*
5.	*Packet_Length_Std*
6.	*Fwd_PSH_Flags*
7.	*Init_Win_bytes_backward.*

p-ISSN: 2301-5373
e-ISSN: 2654-5101

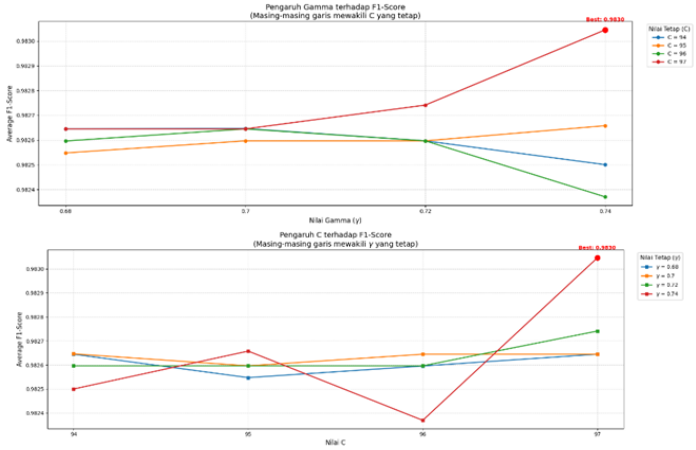
3.3. Cross Validatoin (CV) dan Tunning Hyperparameter

Selanjutnya melakukan validasi model sekaligus *tunning hyperparameter*. Peneliti menggunakan metode *K-fold Cross Validation* dengan nilai K adalah 5. Validasi dilakukan agar meminimalkan bias model terhadap data latih serta memastikan performa model. Pada proses ini dilakukan juga *tunning* terhadap *hyperparameter* untuk menentukan kombinasi *hyperparameter* yang optimal. Pada gambar berikut merupakan hasil validasi dan *tunning hyperparameter* pertama:



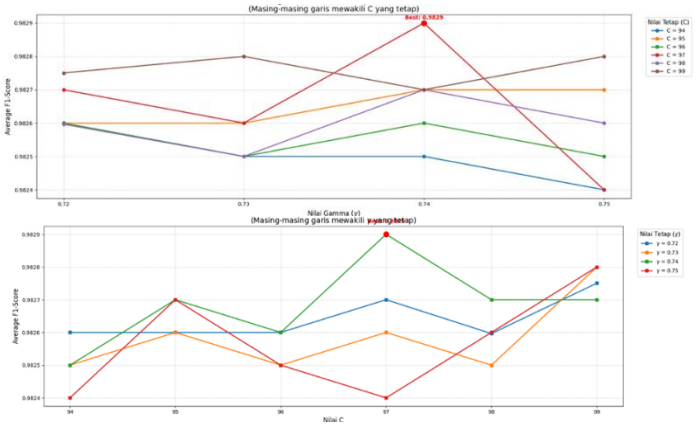
Gambar 5. Grafik hasil *tunning hyperparameter* pertama

Setelah mendapatkan hasil tersebut, saya melakukan *tunning* kedua di sekitar kombinasi *hyperparameter* C = 95 dan gamma = 0.7. Saya menggunakan nilai C yaitu 94, 95, 96, dan 97 serta nilai gamma yaitu 0.68, 0.7, 0.72, dan 0.74. Didapatkan bahwa hasilnya nilai optimal C naik ke 97 dan nilai optimal gamma juga naik ke 0.74. Gambar 6 berikut merupakan hasil *tunning* kedua.



Gambar 6. Grafik hasil *tunning hyperparameter* kedua

Terlihat bahwa nilai *hyperparameter* kembali mengalami kenaikan, peneliti melakukan *tunning* ketiga di antara nilai tersebut untuk memastikan apakah terdapat kombinasi *hyperparameter* yang lebih optimal. Nilai C yang digunakan untuk tunning ketiga ini yaitu 94, 95, 96, 97, 98, dan 99. Sedangkan untuk nilai gamma yaitu 0.72, 0.73, 0.74, 0.75. Berikut merupakan hasil *tunning hyperparameter* ketiga.



Gambar 7. Grafik hasil *tunning hyperparameter* ketiga

Dari hasil *tunning* ketiga tersebut, diketahui bahwa terjadi penurunan performa model saat menggunakan nilai *hyperparameter* lain, sehingga dapat disimpulkan untuk kasus ini, nilai optimal dari C adalah 97 dan nilai optimal gamma adalah 0.74. Dari hasil tersebut, peneliti menetapkan kombinasi C=97 dan gamma=0.74 untuk diimplementasikan pada pelatihan final.

3.4. Pelatihan Model SVM

Untuk pelatihan model, peneliti membagi dataset menjadi X dan y, dimana variabel X berisi fitur-fitur yang digunakan dan variabel y berisi fitur label dari setiap data. Kemudian dilakukan pemisahan (*splitting*) menjadi data latih dan data uji dengan 80% untuk data latih dan 20% untuk data uji. Kemudian dilakukan normalisasi data sebelum dilakukan proses pelatihan model. Peneliti melatih 2 jenis model, yaitu model SVM-CFS dan model *full feature*. Dari hasil pelatihan, didapatkan jumlah *support vector* yaitu 2598, *recall* sebesar 0.9955, *precision* sebesar 0.9728, nilai F1 sebesar 0.9840, serta *accuracy* sebesar 98.40%. Waktu yang dibutuhkan model untuk proses latihan adalah sebesar 310.73 detik, sedangkan untuk melakukan prediksi data uji menghabiskan waktu 0.21detik.

```
Recall    : 0.9955
Precision : 0.9728
F1_Score : 0.9840
Accuracy  : 98.40%
Selected feature training time : 310.7302s
Selected feature testing time  : 0.2121s
```

Gambar 7. Hasil metrik dan pengujian waktu model SVM-CFS

Untuk pelatihan model SVM *full feature*, alur dari pelaksanaan pelatihan juga sama, perbedaannya hanya pada dataset yang digunakan yaitu menggunakan dataset yang masih memiliki seluruh fitur sebelum melewati proses seleksi fitur. Model yang dilatih dengan dataset *full feature* mendapatkan *support vector* sebanyak 3605, *recall* sebesar 0.9889, *precision* sebesar 0.9995, nilai F1 sebesar 0.9942, dan nilai *accuracy* sebesar 99.42%. Waktu latih yang dibutuhkan untuk model naik signifikan yaitu menjadi sebesar 438.96 detik, sedangkan waktu uji membutuhkan waktu 0.29 detik.

```
Recall    : 0.9889
Precision : 0.9995
F1_Score : 0.9942
Accuracy  : 99.42%
Full feature training time : 438.9648s
Full feature testing time  : 0.2928s
```

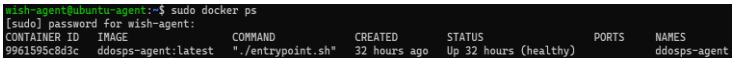
Gambar 8. Hasil metrik dan pengujian waktu model dengan *full feature*

3.5. Implementasi

Tahap selanjutnya adalah implementasi sistem. Pada tahap ini, peneliti akan mengintegrasikan atau menerapkan model ke dalam sistem berbasis web. Model terbaik yang telah didapatkan sebelumnya disimpan dalam format npz untuk memudahkan integrasi ke sistem. Pada penelitian ini, untuk sistem *agent* tidak memiliki antarmuka. *Agent* dijalankan pada server yang merupakan *virtual machine* (VM) yang diisi dengan *ubuntu server*. VM di atur agar menyerupai perangkat fisik terpisah pada jaringan

p-ISSN: 2301-5373
e-ISSN: 2654-5101

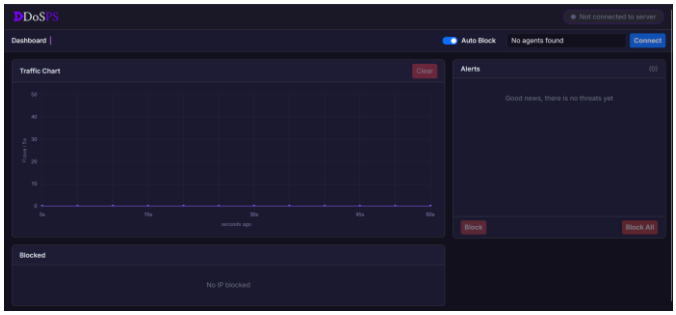
yang memiliki IP sendiri yang berbeda dengan IP *Host* untuk menyimulasikan dua perangkat fisik yang berada di jaringan yang sama. *Agent* di *containerized* dan dijalankan otomatis di latar belakang sebagai proses *daemon* dengan memanfaatkan *docker*.



Gambar 9. Agent berjalan di latar belakang

Agent berfungsi untuk menangkap *traffic* jaringan, paket-paket yang masuk tidak dikirim secara langsung ke admin melainkan dibuat menjadi *traffic flow* lalu melakukan ekstraksi fitur sesuai dengan fitur-fitur yang digunakan model dan baru dikirim ke admin. Peneliti menggunakan *library python Scapy* pada *agent* untuk menangkap dan mengekstrak fitur-fitur *traffic*. Selain itu, *agent* juga berfungsi menjalankan tindakan mitigasi dengan membuat aturan pada *iptables* sesuai perintah yang dikirim oleh *agent*.

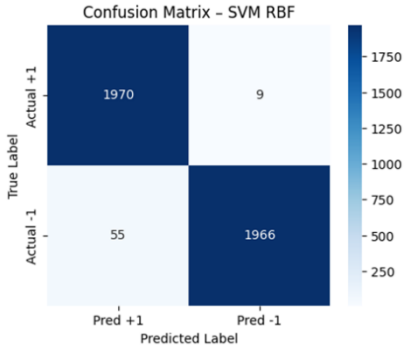
Sedangkan, untuk sistem admin dikembangkan dengan menggunakan html, css, dan javascript untuk mengembangkan antarmuka *website*. Untuk backend, peneliti menggunakan framework flask dan flask socket.io untuk melakukan komunikasi dengan *agent*.



Gambar 10. Tampilan awal admin

3.6. Evaluasi

Setelah proses implementasi sistem, selanjutnya peneliti melakukan evaluasi terhadap model klasifikasi dan sistem berbasis web. Model yang digunakan merupakan model dengan seleksi fitur dan *full feature* dengan *hyperparameter* C bernilai 97 dan gamma bernilai 0.74. Adapun hasil evaluasi model dengan seleksi fitur dapat dilihat pada *confusion matrix* berikut.



Gambar 11. *Confusion Matrix* model SVM-CFS

Dari *confusion matrix* tersebut, diketahui model dapat mengklasifikasikan 1970 *traffic* serangan dan 1966 *traffic* normal dengan benar. Dari hasil tersebut, peneliti mendapatkan nilai *recall*, *precision*, nilai f1, dan *accuracy* yang didapatkan.

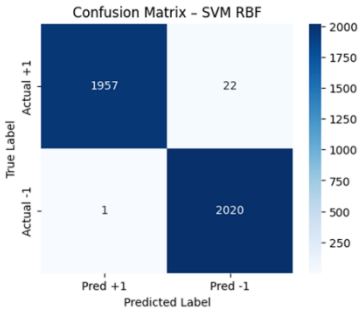
Tabel 4. Hasil perhitungan metrik evaluasi model SVM-CFS

Recall	Precision	F1-Score	Accuracy
99.55%	97.28%	98.40%	98.40%

Secara keseluruhan, model memiliki nilai metrik evaluasi yang cukup bagus dengan rata-rata di atas 95%. Tetapi jika dilihat dari *confusion matrix*, model sedikit ragu-ragu dalam mendeteksi label normal

sehingga memiliki *false positif* yang cukup tinggi. Untuk waktu latih sendiri model ini memakan waktu 310.73 detik, sedangkan pengujian memakan waktu 0.21 detik.

Untuk model dengan *full feature*, selain untuk nilai *recall*, secara keseluruhan model ini mendapatkan nilai yang lebih unggul. Dilihat dari *confusion matrix*, model ini mengklasifikasikan dengan benar 1957 data serangan dan 2020 data normal, nilai *precision*, f1, dan *accuracy* juga meningkat. Walaupun begitu, waktu yang dibutuhkan untuk pelatihan dan pengujian meningkat, masing-masing menjadi 438.96 detik dan 0.29 detik.



Gambar 12. *Confusion matrix* model dengan *full feature*

Tabel 5. Hasil perhitungan metrik evaluasi model *full feature*

Recall	Precision	F1-Score	Accuracy
98.88%	99.95%	99.42%	99.42%

Evaluasi menunjukkan adanya *trade off* antara model SVM-CFS dan *full feature*. Peningkatan beberapa metrik model *full feature* dapat diimbangi oleh metrik *recall* serta beban komputasi model SVM-CFS yang lebih rendah. Dalam konteks deteksi DDoS, *false positive* lebih dapat ditoleransi dibandingkan *false negative*, karena kebocoran serangan sedikit saja dapat berakibat fatal pada aspek *availability* layanan. Oleh karena itu, nilai *recall* menjadi acuan utama. Model SVM CFS terbukti menghasilkan *recall* lebih tinggi sehingga dapat meminimalisir adanya *false negative*.

Model SVM CFS juga berhasil mengurangi kompleksitas dimensi, dilihat dari pengurangan jumlah *support vectors* secara signifikan dari 3605 menjadi 2598. Penurunan ini berdampak pada pengurangan beban komputasi, hal ini terlihat dari waktu pengujian yang hanya memakan waktu 0,21 detik. Pada implementasi *real-time*, kecepatan respons sama pentingnya dengan akurasi karena serangan DDoS yang mampu melumpuhkan layanan atau server dalam hitungan detik. Melalui pengurangan dimensi ini juga, sistem hanya perlu mengekstrak sedikit fitur statistik sehingga mempercepat deteksi. Selain itu, model SVM CFS menunjukkan penurunan waktu pelatihan yang signifikan, hal ini meningkatkan skalabilitas dan adaptabilitas sistem. Mengingat ruang lingkup keamanan jaringan menuntut pembaruan secara berkala, waktu pelatihan yang cepat ini memberikan keuntungan saat diperlukannya melakukan pelatihan ulang model dengan menggunakan dataset *traffic* terbaru agar sistem selalu relevan dalam mengatasi perubahan pola atau pola baru dari serangan DDoS.

Setelah evaluasi model, selanjutnya melakukan evaluasi terhadap sistem admin menggunakan metode *black box testing* untuk memastikan bahwa fungsi-fungsi pada sistem admin dapat berjalan sebagaimana mestinya. Pengujian akan dilakukan sesuai dengan skenario pengujian yang telah dirancang sebelumnya. Dari hasil pengujian yang dilakukan, secara keseluruhan semua fungsi berhasil memberikan *output* yang diharapkan, sehingga sistem admin lolos pengujian *Black box*. Tabel 6 dibawah merupakan hasil pengujian yang dilakukan.

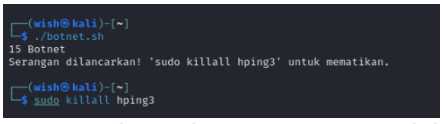
Tabel 6. Hasil pengujian *black box* sistem admin

No.	Fitur	Skenario	Input	Output yang Diharapkan	Hasil
1	Deteksi agent	Pengguna membuka laman web	-	Sistem menampilkan agent yang	Berhasil

p-ISSN: 2301-5373
e-ISSN: 2654-5101

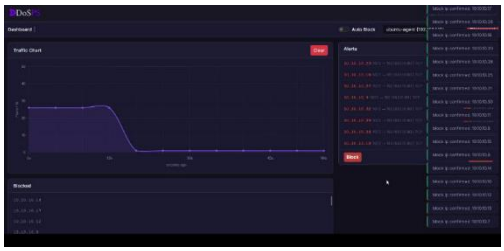
		admin		tersedia	
2	*Connect* agent	Pengguna menekan tombol *connect* setelah memilih agent	Perintah koneksi ke agent dengan menekan tombol *connect*	Admin terkoneksi dengan agent dan indikator koneksi berubah	Berhasil
3	*Auto block*	Pengguna mengaktifkan atau mematikan fitur *auto block*	Menekan tombol *auto block*	Pesan *Pop up* status *auto block*	Berhasil
4	Clear traffic	Pengguna menekan tombol *clear* pada *traffic chart*	Menekan tombol *clear*	Menghapus *traffic* yang telah dikirim agent sebelumnya	berhasil
5	*Block* beberapa IP terDDoS	Pengguna memilih IP, lalu menekan tombol *block*	List IP penyerang	*Pop up* blokir berhasil untuk IP yang dipilih	Berhasil
6	*Block all*	Pengguna menekan tombol *block all*	Menekan tombol *block all*	*Pop up* blokir berhasil untuk semua IP	berhasil

Selanjutnya adalah demo penyerangan dan pemblokiran IP penyerang. Penyerang menjalankan serangannya dengan skrip untuk menjalankan perintah hping3 dengan 15 IP berbeda, mensimulasikan 15 *botnet* untuk melakukan penyerangan DDoS.



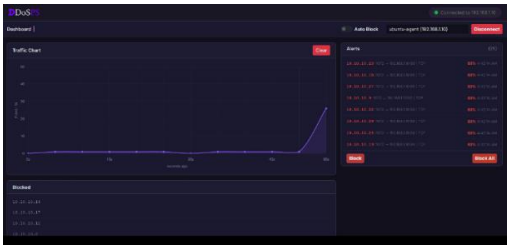
Gambar 13. Penyerang melancarkan serangan untuk beberapa saat

Pada *dashboard* admin, serangan diklasifikasikan lalu dengan fitur *block all* secara otomatis mengirimkan perintah mitigasi untuk memblokir semua IP penyerang yang terdeteksi ke *agent*.

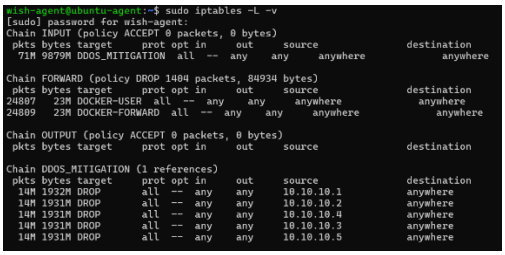


Gambar 14. Tampilan *dashboard* setelah serangan

Terlihat adanya lonjakan *traffic*, admin berhasil melakukan klasifikasi terhadap *flow* yang terkirim dan langsung mengirim perintah mitigasi. Selanjutnya, peneliti melakukan penyerangan kembali untuk mengetahui apakah ada perubahan pada *dashboard* admin serta pengecekan dengan perintah sudo iptables -L -v.



Gambar 15. Admin setelah serangan kedua



```
wish-agent@ubuntu-agent:~$ sudo iptables -L -v
[sudo] password for wish-agent:
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
  71M 9879M DDOS_MITIGATION  all  --  any    any     anywhere             anywhere

Chain FORWARD (policy DROP 1404 packets, 84934 bytes)
 pkts bytes target     prot opt in     out     source               destination
24807   23M DOCKER-USER  all  --  any    any     anywhere             anywhere
24809   23M DOCKER-FORWARD  all  --  any    any     anywhere             anywhere

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain DDOS_MITIGATION (1 references)
 pkts bytes target     prot opt in     out     source               destination
  14M 1932M DROP       all  --  any    any     10.10.10.1           anywhere
  14M 1931M DROP       all  --  any    any     10.10.10.2           anywhere
  14M 1931M DROP       all  --  any    any     10.10.10.4           anywhere
  14M 1931M DROP       all  --  any    any     10.10.10.3           anywhere
  14M 1931M DROP       all  --  any    any     10.10.10.5           anywhere
```

Gambar 16. Tampilan perintah sudo iptables -L -v

Terlihat *dashboard* tidak lagi menangkap *traffic* serangan, dilihat dari hasil perintah *iptables* yang dilakukan juga memperlihatkan beberapa IP penyerang dengan kolom target berisi opsi DROP yang artinya *network packets* dari IP tersebut akan di buang. Ini berarti aturan *iptables* berhasil dibuat dan aktif melakukan *drop network packets*. Dari gambar tersebut diketahui bahwa sudah ada 14 juta paket yang di buang dengan ukuran total 1932Mb untuk masing-masing IP penyerang.

4. Kesimpulan

Berdasarkan proses *tuning hyperparameter*, diperoleh kombinasi nilai optimal yaitu C = 97 dan gamma = 0.74 yang pada proses tunning menghasilkan *F1-Score* sebesar 0.9829, *recall* 0.9940, *precision* 0.9720, dan *accuracy* 0.9827. Lalu, dari hasil evaluasi model terlihat bahwa model SVM dengan seleksi fitur memiliki trade-off yang menguntungkan dibandingkan dengan model SVM dengan full feature. Walaupun model full feature mengalami kenaikan pada precision sebesar 99,95%, F1-score sebesar 99,42%, serta accuracy sebesar 99,42%, model dengan seleksi fitur mendapatkan nilai recall yang lebih tinggi yaitu sebesar sebesar 99,55%. Selain itu, model dengan seleksi fitur juga mengurangi beban komputasi secara signifikan. Waktu pelatihan SVM CFS berkurang menjadi 310,73 detik dan waktu pengujian menjadi 0,21 detik. Hal ini membuktikan bahwa model dengan seleksi fitur lebih ideal untuk implementasi secara real-time.

Referensi

[1] H. Ramli and M. Y. Alifsyah, “Analisis Keamanan Komputer Terhadap Serangan Distributed Denial of Service (DDOS),” *Journal of Renewable Energy and Smart Device*, vol. 1, no. 1, pp. 25–30, 2023.

[2] S. Soim *et al.*, “Optimizing Performance Random Forest Algorithm Using Correlation-Based Feature Selection (CFS) Method to Improve Distributed Denial of Service (DDoS) Attack Detection Accuracy,” *Indonesian Journal of Artificial Intelligence and Data Mining (IJAIDM)*, vol. 7, no. 2, pp. 220–228, 2024, doi: 10.24014/ijaidm.v7i2.24783.

[3] Microsoft, “2022 in review: DDoS attack trends and insights,” www.microsoft.com. Accessed: Apr. 20, 2024. [Online]. Available: https://www.microsoft.com/en-us/security/blog/2023/02/21/2022-in-review-ddos-attack-trends-and-insights/

[4] G. Smith, “Top +35 DDoS Statistics (2024),” stationx.net. Accessed: Apr. 20, 2024. [Online]. Available: https://www.stationx.net/ddos-statistics/

[5] gcore.com, “Gcore Radar report reveals 41% surge in DDoS attack volumes,” gcore.com. Accessed: Apr. 26, 2004. [Online]. Available: https://gcore.com/press-releases/gcore-radar-ddos-attack-trends-q1-q2-2025

[6] R. I. Perwira and H. Prapcoyo, “Software Defined Network : The Comparison of SVM kernel on DDoS Detection,” *RSF Conference Series: Engineering and Technology*, vol. 1, no. 1, pp. 281–290, Dec. 2021, doi: 10.31098/cset.v1i1.413.

[7] M. Azhari, Z. Situmorang, and R. Rosnelly, “Perbandingan Akurasi, Recall, dan Presisi Klasifikasi pada Algoritma C4.5, Random Forest, SVM dan Naive Bayes,” *JURNAL MEDIA INFORMATIKA BUDIDARMA*, vol. 5, no. 2, p. 640, Apr. 2021, doi: 10.30865/mib.v5i2.2937.

[8] I. M. W. Bhaskaraa *et al.*, “Klasifikasi Serangan Distributed Denial of Service (DDoS) Menggunakan Random Forest dengan CFS,” *Jurnal Elektronik Ilmu Komputer Udayana p-ISSN*, vol. 2301, p. 5373, 2022.

[9] G. Y. Lee, L. Alzamil, B. Doskenov, and A. Termehchy, “A Survey on Data Cleaning Methods for Improved Machine Learning Model Performance,” Sep. 2021, [Online]. Available: http://arxiv.org/abs/2109.07127

[10] R. Iliyasu and I. Etikan, “Comparison of quota sampling and stratified random sampling,” *Biom. Biostat. Int. J.*, vol. 10, no. 1, pp. 24–27, Feb. 2021, doi: 10.15406/bbij.2021.10.00326.

[11] S. Sahazada, “Correlation-based Feature Selection in a Data Science Project,” Medium.com. Accessed: Apr. 26, 2024. [Online]. Available: https://medium.com/@sariq16/correlation-based-feature-selection-in-a-data-science-project-3ca08d2af5c6

[12] M. Mohamad, A. Selamat, O. Krejcar, R. G. Crespo, E. Herrera-Viedma, and H. Fujita, “Enhancing big data feature selection using a hybrid correlation-based feature selection,” *Electronics (Switzerland)*, vol. 10, no. 23, Dec. 2021, doi: 10.3390/electronics10232984.

[13] O. Inan and M. S. Uzer, “A Method of Classification Performance Improvement Via a Strategy of Clustering-Based Data Elimination Integrated with k-Fold Cross-Validation,” *Arab. J. Sci. Eng.*, vol. 46, no. 2, pp. 1199–1212, Feb. 2021, doi: 10.1007/s13369-020-04972-y.

[14] B. W. Kurniadi, H. Prasetyo, G. L. Ahmad, B. Aditya Wibisono, and D. Sandya Prasvita, *Analisis Perbandingan Algoritma SVM dan CNN untuk Klasifikasi Buah*. 2021.

[15] A. Verma, “Support Vector Machines (SVM) Supervised Machine Learning ,” dev.to. Accessed: Apr. 26, 2024. [Online]. Available: https://dev.to/anurag629/support-vector-machines-svm-supervised-machine-learning-3lfo

[16] K. Putri *et al.*, “Implementasi Algoritma Support Vector Machine dalam Klasifikasi Deteksi Depresi dari Postingan pada Media Sosial,” *Jurnal Nasional Teknologi Informasi dan Aplikasinya*, vol. 2, no. 1, 2023.

[17] I. K. Nti, O. Nyarko-Boateng, and J. Aning, “Performance of Machine Learning Algorithms with Different K Values in K-fold CrossValidation,” *International Journal of Information Technology and Computer Science*, vol. 13, no. 6, pp. 61–71, Dec. 2021, doi: 10.5815/ijitcs.2021.06.05.

[18] Atikah Dhani Ayuningtyas, “DETEKSI SERANGAN DISTRIBUTED DENIAL OF SERVICE (DDOS),” UIN Syarif Hidayatullah Jakarta, 2023.

[19] S. D. S. Saian, N. L. Kakihary, and T. Wahyono, “Pengujian Content Management System (Cms) Sekolahku Menggunakan Metode Black Box Testing Dengan Teknik Boundary Value Analysis,” *It-Explore: Jurnal Penerapan Teknologi Informasi dan Komunikasi*, vol. 1, no. 2, pp. 100–113, 2022.

[20] N. Latifah, D. L. Fithri, F. Nugraha, Y. Irawan, and Z. H. Ulya, “Equivalence Partitions Method on Black Box Testing of Motor Service Information Systems at CM Jaya Motor Kudus Web-Based with SMS Gateway,” 2022. [Online]. Available: https://journal-computing.org/index.php/journal-ita/index