JURNAL PENDIDIKAN INDONESIA

p-ISSN : 2745-7141 e-ISSN : 2746-1920

Jurnal Pendidikan Indonesia
Vol. 5 No. 9, September 2024

Klasifikasi *Shellcode* dengan *Machine Learning* Berbasis Klasifikasi Biner

Jaka Naufal Semendawai[1*], Deris Stiawan[2], Iwan Pahendra[3]
Teknik Informatika, Fakultas Ilmu Komputer, Universitas Sriwijaya, Indonesia
jaka.semendawai@gmail.com[*1], deris@unsri.ac.id[2], iwanpahendra@unsri.ac.id[3]

INFO ARTIKEL

Kata Kunci: Keamanan Siber, Klasifikasi Biner, *Machine Learning, Supervised Machine Learning, Hyperparameter Tuning*.

ABSTRAK

Internet dapat menghubungkan satu orang dengan orang lain dengan menggunakan perangkat masing-masing. Internet sendiri memiliki dampak positif dan negatif. Salah satu contoh dampak negatif dari internet adalah adanya malware yang dapat mengganggu atau bahkan merusak perangkat atau penggunanya; itulah mengapa keamanan siber diperlukan. Banyak cara yang dapat dilakukan untuk mencegah atau mendeteksi malware. Salah satunya adalah dengan menggunakan teknik *machine learning*. Dataset pelatihan dan pengujian untuk eksperimen ini berasal dari dataset UNSW_NB15. *K-Nearest Neighbour (KNN), Decision Tree,* dan *Naïve Bayes* diimplementasikan untuk mengklasifikasikan apakah sebuah record pada data testing merupakan serangan *Shellcode* atau non-*Shellcode*. *Classifier KNN, Decision Tree*, dan *Naïve Bayes* mencapai tingkat akurasi masing-masing sebesar 96.82%, 97.08%, dan 63.43%. Hasil dari penelitian ini diharapkan dapat memberikan wawasan mengenai penggunaan machine learning dalam mendeteksi atau mengklasifikasikan malwares atau jenis serangan siber lainnya

Keywords:
Binary Classification, Cyber Security, Machine Learning, Supervised Machine Learning, Hyperparameter Tuning

ABSTRACT

Internet can link one person to another using their respective devices. The internet itself has both positive and negative impacts. One example of the internet's negative impact is a malware that can disrupt or even kill a device or its users; that is why cyber security is required. Many methods can be used to prevent or detect malwares. One of the efforts is to use machine learning techniques. Training and testing dataset for the experiments is derived from the UNSW_NB15 dataset. K-Nearest Neighbour (KNN), Decision Tree, and Naïve Bayes classifiers are implemented to classify whether a record in the testing data is Shellcode or non-Shellcode attack. The KNN, Decision Tree and Naïve Bayes classifiers achieve accuracy level of 96.82%, 97.08%, and 63.43%, respectively. The results of this research are expected to provide insight into the use of machine learning in detecting or classifying malwares or other types of cyber attacks.

PENDAHULUAN

Kesadaran akan pentingnya *cyber security* di Indonesia masih tergolong sangat rendah. Hal tersebut dibuktikan oleh data yang diterbitkan oleh *International Communication Union* (ITU) di mana tingkat *cyber security* di Indonesia ada di ranking 70. Hal tersebut menyatakan bahwa Indonesia sangat rentan untuk dikirim serangan siber dari *hacker* di negara lain. Selain itu, menurut data dari *treat exposure rate* (TER), Indonesia memiliki angka kerentanan serangan *malware* sebesar 23,54% (Ashari, 2020).

(Patterson et al., 2023) mengemukakan bahwa organisasi apapun sudah harus memikirkan tentang esensi dari *cyber security*. Hal tersebut dikarenakan semakin tingginya kasus serangan *cyber* yang harus dapat dilawan dengan adanya pengetahuan tentang *cyber security*, karena sudah menyangkut masalah privasi data dan ketahanan infrastruktur. Hal tersebut dapat dilihat dari riset yang dilakukan oleh (Singelton et al., 2021) di mana pada tahun 2021 terdapat serangan yang menggunakan *ransomware* terhadap 10 jenis perusahaan berbeda, dengan nilai rata-rata yaitu 17,4% dari keseluruhan jenis serangan yang terjadi pada perusahaan-perusahaan tersebut.

Salah satu *malware* yang sedang menjadi tren untuk digunakan sebagai alat serangan adalah *shellcode*. Penggunaan *shellcode* pada serangan *cyber* sudah menjadi tren di kalangan *hacker*. *Shellcode* dapat melakukan aktivitas ilegal seperti serangan DoS, pencurian data, hingga perusakan sistem secara otomatis pada komputer tujuan (Yang et al., 2022). *Shellcode* merupakan sebuah kode yang dirancang agar dapat menjalankan tugasnya secara otomatis. *Shellcode* dapat memberikan izin kepada *attacker* untuk mengeksploitasi komputer tujuan secara menyeluruh. *Shellcode* umumnya diproduksi menggunakan bahasa *assembly*.

Struktur dasar dari *shellcode* yaitu sebagai berikut. Yang pertama adalah *No Operation Instructions* (NOP *Sled*). NOP *sled* digunakan untuk memastikan jika eksekusi yang dilakukan tidak gagal. Kemudian, *Bootstrap Code* yang digunakan untuk menyiapkan lingkungan eksekusi. *Bootstrap code* biasanya terdiri atas kode-kode nilai *register* tertentu yang akan digunakan oleh *payload*. Selanjutnya adalah *payload*. *Payload* digunakan untuk melakukan tugas utama dari *hacker* tergantung dengan kode yang dituliskan di dalamnya. Yang terakhir adalah *clean-up code*. *Clean-up code* digunakan untuk menghilangkan jejak dari *hacker* setelah melakukan serangan ke sistem tujuan. Hal tersebut yang dapat membuat *shellcode* yang baik dan dapat memberikan performa dari *shellcode* dalam mengeksekusi dengan bersih. Di dalam *shellcode* terdapat beberapa fungsi yang disebut sebagai *root shell*. Hal tersebut merupakan yang paling banyak digunakan sebelum terdapat beberapa perkembangan lebih lanjut (Anley et al., 2007).

Shellcode dikembangkan dengan banyak sekali alat yang memiliki fungsinya masing-masing. Fungsi dari alat yang digunakan untuk mengembangkan *shellcode* terdiri atas alat untuk menulis *code*, *compile* kode, *convert*, *test*, dan *debug shellcode*. Beberapa alat tersebut dapat memudahkan dalam pembentukan ataupun pengembangan dari *shellcode*. Beberapa alat yang digunakan untuk membentuk *shellcode* yaitu *NASM, GDB, ObjDump, Ktrace, Strace, Readelf*. *NASM* merupakan alat yang terdiri dari *assembler* yang disebut *NASM* dan *disassembler* yang disebut *NDISAM* (Foster et al., 2005). *Shellcode* dapat melakukan pengunduhan dan pengeksekusian terhadap *malware* secara otomatis ketika *hacker* berhasil masuk ke dalam sistem tujuannya. Namun, pendeteksian terhadap *shellcode* tersebut belum banyak dikembangkan.

Penelitian dari (Akabane et al., 2019) yang mengembangkan metode pencegahan terhadap adanya aktivitas *shellcode* dengan hasil riset mereka yaitu *EAF Guard Driver* cukup memberikan hasil yang mengesankan bagi pencegahan *shellcode*. *Driver* tersebut dapat mencegah aktivitas *shellcode* dengan baik tanpa adanya *false alarm rate*. Dan juga hasil pengujian *benchmark* dari *EAF Guard Driver* mengalami peningkatan hingga 0,02%.

Selain itu, terdapat penelitian yang dilakukan oleh (Kanemoto et al., 2019) di mana mereka menggunakan metode *shellcode emulation* yang berlandaskan kepada nilai akurasi dan performa. Mereka bertujuan untuk mendapatkan model yang dapat mengidentifikasi pemberitahuan yang penting yang dapat memberikan informasi mengenai adanya gangguan keamanan pada sistem secara otomatis. Hasil dari penelitian ini yaitu tingkat akurasi dan performa yang didapatkan yaitu kurang lebih 60% *remote shellcode* terdeteksi.

Penelitian ini dilakukan dengan menggunakan *machine learning* berbasis *supervised machine learning*. Riset serupa juga dilakukan oleh (Moon et al., 2022) yang menggunakan *supervised machine learning* untuk melakukan pendeteksian terhadap *malware*. Pada riset tersebut, mereka menggunakan *feature hashing* karena dapat menghemat hingga 70% memori yang digunakan, namun meningkatkan akurasi dalam pendeteksian *malware*. Penelitian yang dilakukan oleh (Rajesh Bingu, 2023) melakukan klasifikasi berbasis *binary classification* dan *multiclass classification* dengan bantuan beberapa model *machine learning*. Hasil yang didapatkan dari penelitian tersebut yaitu pada klasifikasi berbasis *binary* menghasilkan nilai akurasi dari 99,17% hingga 99,65%.

Pada penelitian ini, peneliti menggunakan beberapa jenis *machine learning*, yaitu *K-Nearest Neighbors, Decision Tree*, *Naive Bayes*. Penelitian yang dilakukan oleh (Gouda et al., 2024) dengan menggunakan model *decision tree* dan *k-nearest neighbors* untuk melakukan pendeteksian terhadap *malware*, dan menggunakan *data set UQ-NIDS-V2*. Di dalam *data set* tersebut terdapat serangan *shellcode*. Hasil dari penelitian tersebut menunjukkan dengan menggunakan model *decision tree* tidak mampu mendeteksi serangan *shellcode*. Hal tersebut dapat dilihat dari tingkat presisi, sensitivitas, dan *F1 Score* yaitu 0%. Namun, untuk nilai akurasi dalam pendeteksian semua jenis *malware* yaitu 98,78%. Kemudian, untuk model *K-nearest neighbors*, dalam pendeteksian *shellcode* juga tidak baik. Hal tersebut juga dapat dilihat dari tingkat presisi, sensitivitas, dan *F1 Score* yaitu 0%. Dan, untuk nilai akurasi dalam pendeteksian jenis *malware* yaitu 98,16%. Selain itu, penelitian yang dilakukan oleh (Samantaray et al., 2024) yang menggunakan berbagai jenis *machine learning* seperti SVM, KNN, LR, DT, NB, dan RF. Kemudian, model tersebut menggunakan algoritma *MaxAbsScaler*. Hasil yang didapatkan dari penelitian ini yaitu nilai akurasi yang didapatkan dalam melakukan klasifikasi berbasis *multiclass classification* mengalami peningkatan dari 60% menjadi 94% dengan bantuan teknik *MaxAbsScaler*.

Pada penelitian ini, kami menggunakan machine learning, yaitu K-nearest neighbors, decision tree, dan naïve Bayes. Hal ini dikarenakan ketiga model tersebut dapat digunakan untuk melakukan klasifikasi berbasis biner. Kemudian, dari hasil pengujian dengan menggunakan ketiga model tersebut, kami akan mengambil data seperti akurasi, F1-Score, precision, dan recall, yang kemudian kami akan menganalisis data tersebut. Pada model KNN, kita menggunakan berbagai macam pengujian, yaitu scaling data dan tuning hyperparameter pada data yang telah diperbaiki. Selain itu, scaling data atau hyperparameter tuning hanya

digunakan pada data, dan scaling data atau hyperparameter tuning tidak digunakan sama sekali. Kami juga melakukan hal ini untuk model pohon keputusan. Dan kami juga melakukan model naïve bayes tetapi memvariasikan perbandingan data training dan testing.

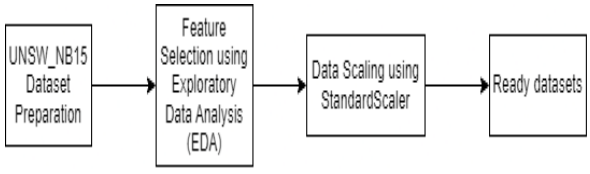
METODE PENELITIAN

Penelitian ini meneliti tentang serangan Shellcode menggunakan klasifikasi biner untuk melihat performa dari *classifier KNN, Decision Tree dan Naïve Bayes* dalam mendeteksi serangan *shellcode*. Data yang digunakan pada penelitian ini diambil dari dataset UNSW_NB15, dimana sudah banyak penelitian yang dilakukan pada dataset tersebut dengan jenis serangan *shellcode*. Rasional dibalik penggunaan dataset UNSW-NB15 adalah karena dataset ini memiliki data terkini, logical, dan fitur dari setiap serangan yang dapat memberikan akses untuk menganalisa setiap teknik serangan. (Moustafa et al., 2018). Langkah-langkah yang akan dilakukan dalam penelitian ini adalah sebagai berikut.

1. Membuat dataset baru dengan memfilter jenis serangan Shellcode dan non-Shellcode dari dataset UNSW_NB15.
2. Memberikan label pada data dengan menggunakan label 0 dan 1, di mana label 0 untuk serangan non-*Shellcode* dan label 1 untuk jenis serangan *Shellcode*.
3. Membagi data untuk dataset pelatihan dan dataset pengujian.
4. Menguji data dengan menggunakan model *KNN, Decision Tree,* dan *Naïve Bayes*.
5. Membandingkan kinerja pengklasifikasi *KNN, Decision Tree,* dan *Naïve Bayes* dalam hal akurasi deteksi serangan non-*Shellcode* vs. serangan *Shellcode.*
6. Menganalisis hasil yang telah didapatkan dari percobaan.

Prapemrosesan Data

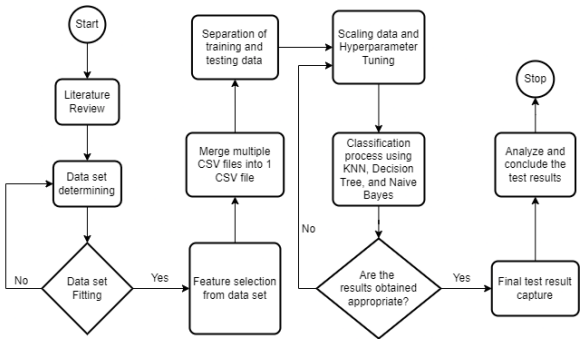
Sebelum kami menguji data, kami terlebih dahulu melakukan prapemrosesan data. Setelah kami mendapatkan file CSV dari kumpulan data, UNSW_NB15, kami melakukan seleksi fitur menggunakan metode Exploratory Data Analysis (EDA). Kemudian, setelah kami mendapatkan fitur terbaik dari metode EDA, kami membuat file CSV baru yang berisi data dengan fitur yang dipilih dari metode EDA. Setelah itu, kami menskalakan data untuk beberapa pengujian menggunakan StandardScaler. Hasil dari langkah-langkah sebelumnya menghasilkan kumpulan data yang siap untuk diuji. Grafik dari data prapemrosesan akan ditunjukkan pada Gambar 1 di bawah ini:



Gambar 1. Proses Prapemrosesan Data.

Alur Penelitian

Pada langkah awal penelitian, penulis melakukan studi literatur untuk mencari beberapa penelitian terdahulu yang mendukung penelitian ini. Kemudian, beberapa dataset yang tersedia untuk umum diselidiki untuk menentukan dataset yang paling sesuai dengan kebutuhan eksperimen. Setelah penyelidikan yang cermat, dataset UNSW_NB15 dipilih. Beberapa atribut penting dari catatan lalu lintas dalam kumpulan data kemudian dianggap sebagai fitur. Kami menggunakan metode *Exploratory Data Analysis* untuk memilih fitur terbaik. Beberapa file csv dari dataset yang telah diproses digabungkan menjadi satu file csv. Untuk beberapa pengujian, kami menskalakan data menggunakan metode StandardScaler. Selain itu, kami juga menggunakan tuning hyperparameter untuk setiap machine learning untuk mendapatkan hasil yang baik dari pengujian. Pemisahan data untuk tujuan pelatihan dan pengujian dilakukan dengan menggunakan library machine learning yang tersedia dalam bahasa pemrograman Python. Selanjutnya, percobaan menggunakan *classifier KNN, Decision Tree,* dan *Naïve Bayes* dilakukan dan hasilnya dianalisis lebih lanjut. Alur kerja dari metode yang diusulkan diilustrasikan pada Gambar 2.



Gambar 2. Alur Penelitian

Pemilihan Fitur dari Dataset UNSW_NB15

Dataset UNSW_NB15 memiliki 49 fitur. Fitur-fitur tersebut memiliki fungsinya masing-masing. Fitur ini didapatkan dari hasil penelitian sebelumnya, dimana peneliti sebelumnya mengambil data dengan menggunakan aplikasi Tcpdump untuk melihat trafik yang terjadi pada saat eksperimen dijalankan. Pada penelitian ini, penulis hanya mengambil beberapa fitur yang sesuai untuk digunakan pada eksperimen agar mendapatkan hasil yang optimal dan memenuhi tujuan dari penelitian ini. Kami menggunakan Exploratory Data Analysis (EDA) untuk mendapatkan fitur terbaik untuk penelitian kami. Tabel 1 memberikan deskripsi dari fitur-fitur yang dipilih.

Tabel 1. Deskripsi Fitur Dataset UNSW_NB15

Number	Feature	Type	Description
1.	dur	Float	Record total duration
2.	sbytes	Integer	Source to destination transaction bytes
3.	dbytes	Integer	Destination to source transaction bytes
4.	Sload	Float	Source bits per second
5.	smeanz	Integer	Mean of the flow packet size transmitted by the source

6.	dmeanz	Integer	Mean of the flow packet size transmitted by the destination
7.	Stime	Timestamp	Record start time
8.	Sintpkt	Float	TCP connection setup time, the time between the SYN and the SYN_ACK packets.
9.	label	Binary	0 for non-shellcode and 1 for shellcode records

KNN *Classifier*

Metode KNN melakukan klasifikasi berdasarkan pembelajaran dengan analogi. Algoritma KNN dapat menemukan pola untuk nilai terdekat ke-k. Nilai k adalah k tetangga dari nilai sampel yang tidak diketahui. Tingkat "kedekatan" dijelaskan dalam istilah jarak Eudian. Jarak Eudian adalah jarak antara $X = (x_1, x_2, \dots, x_n)$ and $Y = (y_1, y_2, \dots, y_n)$, yang akan dituliskan dalam (3) (Han & Kamber, 1998).

$$d(X,Y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \qquad (1)$$

Penggunaan *K-Nearest neighbors* untuk klasifikasi data diawali dengan tahap penggabungan *file* mentah dari *dataset* yang digunakan. Kemudian, dari data mentah tersebut akan dilakukan seleksi fitur yang cocok untuk digunakan pada proses klasifikasi dengan menggunakan metode *Exploratory Data Analysis*. Kemudian, peneliti menggunakan teknik *oversampling* untuk mendapatkan *dataset* yang seimbang. Selanjutnya, akan dilakukan pemisahan data *training* dan *testing*. Pada riset ini, penulis menggunakan variasi rasio yaitu 90:10; 80:20; 70:30; 60:40; dan 50:50 untuk data *training* dan *testing*. Untuk klasifikasi menggunakan KNN, peneliti menggunakan metode *hyperparameter tuning* untuk mendapatkan performa yang optimal. Kemudian, visualisasi data dari model KNN akan ditampilkan dalam bentuk grafik nilai akurasi dan *F1-Score* dari keseluruhan pengujian.. Setelah mendapatkan hasil klasifikasi berupa *F1-Score* dan tingkat akurasi, penulis akan melakukan analisa dan penarikan kesimpulan dari hasil tersebut.

Decision Tree *Classifier*

Decision Tree Classifier adalah sebuah proses klasifikasi data dengan mengubah bentuk data (tabel) menjadi model pohon, mengubah model pohon menjadi aturan, dan menyederhanakan aturan (pemangkasan). Sebuah properti statistik yang disebut information gain digunakan untuk menentukan atribut terbaik. Information gain mengukur seberapa handal sebuah atribut dalam memisahkan sampel pelatihan sesuai dengan klasifikasi targetnya. Untuk menentukan information gain yang tepat, kita mulai dengan memilih ukuran yang disebut entropi dalam teori informasi, yang mencirikan kemurnian/ketidakmurnian dari kumpulan sampel secara acak (Sarimuddin et al., 2020).

Penggunaan *decision tree* untuk klasifikasi data diawali dengan tahap penggabungan *file* mentah dari *dataset* yang digunakan. Kemudian, dari data mentah tersebut akan dilakukan seleksi fitur yang cocok untuk digunakan pada proses klasifikasi dengan menggunakan metode

Exploratory Data Analysis. Kemudian, peneliti menggunakan teknik *oversampling* untuk mendapatkan *dataset* yang seimbang. Selanjutnya, akan dilakukan pemisahan data *training* dan *testing*. Pada riset ini, penulis menggunakan variasi rasio yaitu 90:10; 80:20; 70:30; 60:40; dan 50:50 untuk data *training* dan *testing*. Pada model *decision tree*, peneliti menggunakan metode *hyperparameter tuning* untuk mendapatkan performa yang lebih optimal. Kemudian, penulis menggunakan model *graphviz* untuk menampilkan pohon keputusan dari *dataset*. Setelah dilakukan visualisasi data ke dalam bentuk pohon keputusan, dan juga visualisasi data dari model DT akan ditampilkan dalam bentuk grafik nilai akurasi dan *F1-Score* dari keseluruhan pengujian. Setelah mendapatkan hasil klasifikasi berupa *F1-Score* dan tingkat akurasi, penulis akan melakukan analisa dan penarikan kesimpulan dari hasil tersebut.

Naïve Bayes *Classifier*

Ada dua tahapan yang diperlukan apabila kita menggunakan *classifier Naïve Bayes* untuk mengklasifikasi data. Pertama, membuat *bag of words*, kemudian dilanjutkan dengan melakukan training. Hal ini dilakukan untuk mendapatkan model klasifikasi dalam bentuk probabilitas. Selanjutnya, guna menjaga hasil testing agar tidak rusak, diperlukan *Laplace Smoothing,* sehingga peluang 0 pada *training* dapat dihindari. *Laplace smoothing* menambahkan angka 1 dibagi dengan jumlah semua fitur ditambahkan ke semua fitur sehingga tidak ada yang bernilai 0. Persamaan Naïve Bayes yang digunakan untuk menentukan kelas akan dituliskan dalam (4) (Fitriyyah et al., 2019).

$$V_{MAP} = argmax_{vj} evP(v_j) \prod_{i=1}^{n} P(a_j|v_j) \quad (2)$$

Penggunaan *naive bayes* untuk klasifikasi data diawali dengan tahap penggabungan *file* mentah dari *dataset* yang digunakan. Kemudian, dari data mentah tersebut akan dilakukan seleksi fitur yang cocok untuk digunakan pada proses klasifikasi dengan menggunakan metode *Exploratory Data Analysis*. Kemudian, peneliti menggunakan teknik *oversampling* untuk mendapatkan *dataset* yang seimbang. Selanjutnya, akan dilakukan pemisahan data *training* dan *testing*. Pada riset ini, penulis menggunakan variasi rasio yaitu 90:10; 80:20; 70:30; 60:40; dan 50:50 untuk data *training* dan *testing*. Pada klasifikasi menggunakan *Naive Bayes*, peneliti menggunakan *Gaussian Naive Bayes*. Setelah dilakukan proses klasifikasi menggunakan *naive bayes*, hasil klasifikasi *F1-Score*, dan tingkat akurasi, penulis akan melakukan analisa dan penarikan kesimpulan dari hasil tersebut. Visualisasi data dari hasil pengujian ini akan ditampilkan ke dalam grafik nilai akurasi dan *F1-Score* dari masing-masing pengujian.

Confussion Matrix

Confusion Matrix adalah matriks yang berisi nilai aktual dan prediksi klasifikasi, yang digunakan untuk mengevaluasi klasifikasi dan memprediksi objek yang benar atau salah (Abdillah, 2015). Tabel *confusion matrix* akan ditunjukkan pada tabel 2.

Tabel 2. Confusion Matrix

Classification	Predicted Class	
	Shellcode	Non-shellcode

Shellcode	True Positive (TP)		False Negative (FN)	
Non-shellcode	False Positive (FP)		True Negative (TN)	

Berdasarkan tabel confusion matrix di atas, perhitungan nilai akurasi dapat dilakukan dengan menggunakan rumus (3) sebagai berikut:

$$Accuracy = \left(\frac{TP+TN}{TP+FP+TN+FN}\right) * 100\% \qquad (3)$$

HASIL DAN PEMBAHASAN

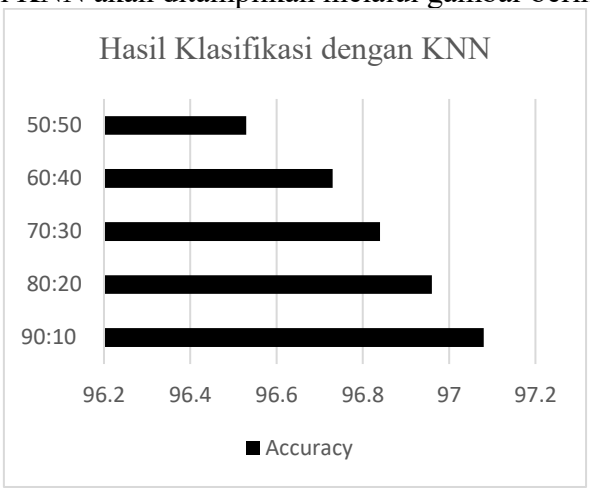
Penelitian ini dilakukan pada komputer dengan spesifikasi sebagai berikut: RAM 8 GB, Prosesor Intel Core I5-8520U, berjalan pada sistem operasi Windows 10. Pengklasifikasi diimplementasikan pada platform *Google Colaboratory*. Hasil penelitian akan disajikan pada bagian berikut ini.

Hasil Penelitian

Penelitian ini menggunakan tiga klasifikasi biner, yakni: KNN, Decision Tree and Naïve Baiyes (NB).

Klasifikasi dengan K-Nearest Neigbors

Hasil Klasifikasi dengan KNN akan ditampilkan melalui gambar berikut ini:

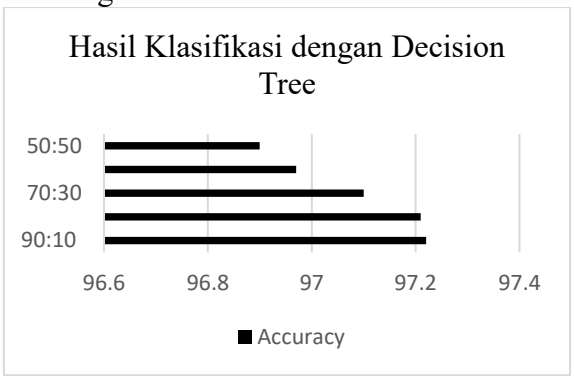


Gambar 2. Tingkat Akurasi Hasil Klasifikasi dengan KNN.

Berdasarkan gambar di atas, model *K-Nearest Neighbors* menghasilkan performa yang cukup baik dalam melakukan klasifikasi terhadap data set ini. Hal tersebut dapat dilihat dari beberapa hasil pengujian dengan menggunakan metode KNN. Dari hasil pengujian yang menggunakan rasio training dan testing data sebesar 90:10. Pengujian ini menggunakan metode StandardScaler dan Hyperparameter Tuning, di mana menghasilkan nilai akurasi sebesar 97,08% dan F1-Score sebesar 97,07%. Nilai tersebut merupakan yang terbaik yang dihasilkan oleh pengujian dengan menggunakan metode ¬K-Nearest Neighbors. Kemudian, parameter yang memberikan performa terbaik didapatkan dari penggunaan hyperparameter tuning, di mana parameter tersebut adalah metric = euclidean, n_neighbors = 4, dan weight = distance. Sehingga, performa maksimal dari KNN didapatkan ketika menggunakan nilai K = 4.

Klasifikasi dengan Decision Tree

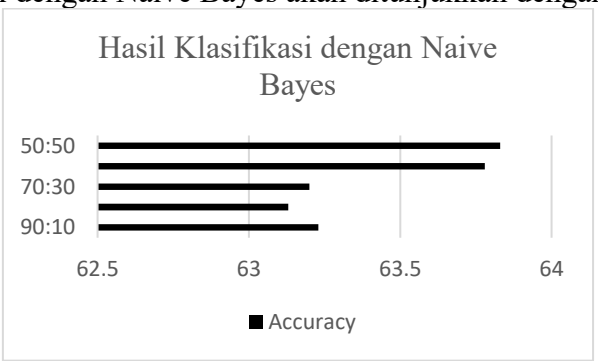
Kemudian, model *decision tree* merupakan metode yang menghasilkan performa terbaik dalam melakukan klasifikasi serangan *shellcode*. Pengujian yang menghasilkan performa terbaik yaitu saat melakukan *hyperparameter tuning*. Pengujian ini menggunakan perbandingan antara *training* dan *testing* data sebesar 90:10, di mana nilai akurasi yang dihasilkan yaitu 97,22%. Kemudian, nilai *F1-Score* yang dihasilkan oleh pengujian ini sebesar 97,22% untuk klasifikasi serangan *shellcode*. Hal tersebut membuktikan jika model ini dapat melakukan klasifikasi serangan *shellcode*. Lalu, dengan melakukan *hyperparameter tuning* pada model *decision tree*, parameter terbaik yang dihasilkan adalah *criterion*: *entropy* dan *random state* = 42 dengan *F1-Score* sebesar 97,22%. Hasil dari klasifikasi dengan *decision tree* akan ditampilkan ke dalam gambar berikut ini.



Gambar 3. Tingkat Akurasi Hasil Klasifikasi dengan *Decision Tree.*.

Klasifikasi dengan Naïve Bayes

Hasil Klasifikasi dengan Naïve Bayes akan ditunjukkan dengan gambar berikut ini.



Gambar 4. Tingkat Akurasi Hasil Klasifikasi dengan Naïve Bayes.

Lalu, model terakhir yang digunakan pada penelitian ini adalah Naïve Bayes. Model ini menghasilkan performa yang tidak sebaik model KNN dan decision tree. Hal tersebut dapat dilihat dari pengujian yang menghasilkan performa terbaik yaitu saat menggunakan perbandingan antara training dan testing data sebesar 50:50. Selain itu, pada pengujian ini juga melakukan tahap Hyperparameter Tuning, di mana menghasilkan nilai akurasi yang dihasilkan sebesar 63,83%. Selain itu, F1-Score yang dihasilkan dari pengujian ini sebesar 51,93%. Hal tersebut membuktikan jika model ini mampu melakukan klasifikasi serangan shellcode dan serangan selain shellcode. Namun, model ini tidak terlalu handal dalam melakukan klasifikasi terhadap data dari serangan shellcode. Kemudian, dengan melakukan hyperparameter tuning pada Gaussian Naïve Bayes menghasilkan performa terbaik dari model ini, di mana parameter tersebut yaitu var_smoothing = 101 dengan F1-Score sebesar 0,613.

Pembahasan

Tabel 3. Akurasi dan *value* F1-Score dari seluruh *Machine Learning*.

Method	Variation	Accuracy (%)	F1-Score (%)
K-Nearest Neighbors	90:10	97	97
	80:20	97	97
	70:30	97	97
	60:40	96	97
	50:50	96	98
Decision Tree	**90:10**	**98**	**97**
	80:20	98	97
	70:30	97	97
	60:40	97	97
	50:50	97	97
Naïve Bayes	90:10	77	37
	80:20	78	37
	70:30	77	37
	60:40	77	39
	50:50	77	39

Dari Tabel 3 di atas, dapat dilihat bahwa hasil dari klasifikasi dengan *K-Nearest Neighbor* dan *Decision Tree* memiliki kinerja terbaik. Meskipun demikian, performa keseluruhan dari *classifier* KNN dalam klasifikasi serangan *Shellcode* tidak cukup baik. Hal ini disebabkan oleh karakteristik dari pengklasifikasi KNN yang tahan *(robust)* terhadap perubahan data yang sangat ekstrim *(outlier)* dimana data yang digunakan untuk pengujian memiliki perubahan data yang signifikan. Namun, klasifikasi ini tidak optimal untuk data yang cukup besar meskipun *classifier* ini tahan terhadap perubahan data yang besar. Hal ini dikarenakan jika perubahan data terlalu besar, maka perubahan ini akan menjadi sebuah kelemahan bagi *classifier* ini. Jadi, ada batasan perubahan data pada *classifier* KNN. Sedangkan untuk *classifier Decision Tree* juga memberikan performa yang relatif baik dalam klasifikasi *Shellcode*. Pencapaian yang baik tersebut juga dipengaruhi oleh kelebihan model *Decision Tree* yang tidak sensitif terhadap perubahan data yang signifikan *(outlier).* Selain itu, model *Decision Tree* juga memiliki nilai akurasi yang cukup baik dalam mengklasifikasikan dan memprediksi data. *CLassifier* Naïve Baiyes juga memberikan performa yang baik dalam mengklasifikasikan dan memprediksi pada saat percobaan pengujian. Namun, *classifier* ini tidak cukup baik dalam klasifikasi serangan

Shellcode, bahkan tidak direkomendasikan. Hal ini dikarenakan karakteristik *classifier* yang memiliki performa yang terbatas untuk data yang kompleks. Selain itu, model ini juga cukup sensitif terhadap fitur yang digunakan. Secara keseluruhan, visualisasi dari data klasifikasi yang digunakan dalam pengujian menunjukkan bahwa ketiga *classifier* berbasis *machine learning* memiliki kinerja yang relatif sangat baik dalam mendeteksi data positif dibandingkan dengan data negatif

Pembahasan

Berdasarkan hasil analisis diketahui bahwa jumlah jurnal yang membahas tentang pengaruh CAR, LDR, NPL, dan BOPO terhadap ROA bank yang terdaftar di BEI tahun 2014 -2024 adalah sebanyak 120 Jurnal yang dapat diakses. Permasalahan utama yang menjadi pembahasan pada jurnal – jurnal tersebut dapat di bagi menjadi sebagai berikut:

1. Pengaruh CAR (Capital Adequacy Ratio) terhadap ROA (Return on Assets):
   a) Permasalahan: Hasil penelitian menunjukkan bahwa pengaruh CAR terhadap ROA tidak konsisten. Beberapa penelitian menemukan CAR berpengaruh positif signifikan terhadap ROA, namun sebagian lainnya tidak menemukan pengaruh signifikan. Ini menciptakan ketidakpastian mengenai hubungan langsung antara permodalan bank dan profitabilitasnya.
   b) Rekomendasi: Bank harus lebih cermat dalam menjaga rasio kecukupan modal dengan mempertimbangkan faktor risiko yang dihadapi, bukan hanya berfokus pada penambahan modal. Penelitian tambahan diperlukan untuk memahami kapan dan bagaimana CAR dapat memberikan dampak positif terhadap profitabilitas bank.

2. Pengaruh NPL (Non-Performing Loan) terhadap ROA:
   a) Permasalahan: NPL umumnya ditemukan memiliki pengaruh negatif signifikan terhadap ROA. Tingginya NPL menunjukkan kualitas kredit yang buruk, yang berdampak langsung pada profitabilitas bank.
   b) Rekomendasi: Manajemen risiko kredit harus diperkuat dengan strategi mitigasi yang lebih baik untuk menurunkan NPL. Bank perlu lebih selektif dalam menyalurkan kredit dan melakukan pengawasan ketat terhadap debitur yang berpotensi menimbulkan kredit macet.

3. Pengaruh LDR (Loan to Deposit Ratio) terhadap ROA:
   a) Permasalahan: Hasil penelitian tentang pengaruh LDR terhadap ROA juga bervariasi. Beberapa studi menemukan pengaruh positif, namun banyak yang tidak signifikan. Hal ini menunjukkan bahwa likuiditas tidak selalu berkorelasi dengan profitabilitas.
   b) Rekomendasi: Bank harus menjaga keseimbangan antara penyaluran kredit dan dana yang dihimpun untuk memastikan likuiditas yang cukup tanpa mengorbankan profitabilitas. Perlu adanya kebijakan likuiditas yang lebih fleksibel namun terukur dalam menghadapi dinamika ekonomi yang fluktuatif.

4. Pengaruh BOPO (Beban Operasional terhadap Pendapatan Operasional) terhadap ROA:
   a) Permasalahan: BOPO ditemukan secara konsisten memiliki pengaruh negatif signifikan terhadap ROA. Tingginya biaya operasional menggerus profitabilitas bank, yang berarti efisiensi operasional merupakan tantangan utama.
   b) Rekomendasi: Bank perlu mengadopsi strategi efisiensi yang lebih baik dalam pengelolaan biaya operasional. Teknologi digital dan automasi proses bisa menjadi solusi untuk mengurangi biaya dan meningkatkan produktivitas operasional.

5. Pengaruh CAR, NPL, LDR, dan BOPO terhadap ROA:
   a) Permasalahan: Hasil penelitian menunjukkan bahwa keempat variabel ini seringkali berpengaruh signifikan secara simultan terhadap ROA, meskipun pengaruh individual masing-masing variabel kadang tidak signifikan.
   b) Rekomendasi: Pendekatan komprehensif dalam mengelola rasio keuangan sangat penting. Bank harus melakukan analisis terpadu untuk menyeimbangkan rasio-rasio ini, sehingga dapat meningkatkan profitabilitas secara keseluruhan.

KESIMPULAN

Penelitian ini menganalisis klasifikasi *Shellcode* dengan *machine learning* berbasis klasifikasi biner, yaitu: KNN, Decision Tree, dan Naïve Bayes. Secara keseluruhan, ketiga *classifier* memiliki kinerja yang baik dalam mengklasifikasi serangan *Shellcode*. Pengklasifikasi *Decision Tree* mencapai tingkat akurasi terbaik sebesar 97,21%. Namun, tingkat akurasi dari metode KNN dan Naive Bayes juga menunjukkan hasil yang tidak mengecewakan. Hal ini mengindikasikan bahwa klasifikasi berbasis klasifikasi biner dapat mengklasifikasikan serangan *Shellcode* yang diambil dari dataset UNSW_NB15. Hasil dari penelitian ini diharapkan dapat memberikan wawasan tentang penggunaan *machine learning* dalam mendeteksi atau mengklasifikasikan *malwares* atau jenis serangan siber lainnya. Penelitian selanjutnya dapat mengeksplorasi metode *preprocessing* dan/atau klasifikasi dengan klasifikasi multikelas untuk mendeteksi dan mengklasifikasikan jenis serangan siber, khususnya *Shellcode Attack*. Selain itu, jenis serangan dapat lebih bervariasi dan tidak terbatas pada dataset yang digunakan dalam penelitian ini.

DAFTAR PUSTAKA

Abdillah, S. (2015). Penerapan Algoritma Decision Tree C4.5 Untuk Diagnosa Penyakit Stroke Dengan Klasifikasi Data Mining Pada Rumah Sakit Santra Maria Pemalang. *Jurnal Teknik Informatika*, 1–12.

Akabane, S., Miwa, T., & Okamoto, T. (2019). An EAF guard driver to prevent shellcode from removing guard pages. *Procedia Computer Science*, *159*, 2432–2439. https://doi.org/10.1016/j.procs.2019.09.418

Anley, C., Heasman, J., Linder, F., & Richarte, G. (2007). *The Shellcoder's Handbook: Discovering and Exploiting Security Holes, 2nd Edition*.

Ashari, M. (2020). *Keamanan Informasi: Sudah Saatnya Kita Peduli*. DJKN Kemenkeu.

Fitriyyah, S. N. J., Safriadi, N., & Pratama, E. E. (2019). Analisis Sentimen Calon Presiden Indonesia 2019 dari Media Sosial Twitter Menggunakan Metode Naive Bayes. *Jurnal Edukasi Dan Penelitian Informatika (JEPIN)*, *5*(3), 279. https://doi.org/10.26418/jp.v5i3.34368

Foster, J. C., Osipov, V., Bhalla, N., Heinen, N., & Aitel, D. (2005). Buffer overflow attacks: Detect, exploit, prevent. In *Buffer Overflow Attacks: Detect, Exploit, Prevent*. https://doi.org/10.1016/B978-1-932266-67-2.X5031-2

Gouda, H. A., Ahmed, M. A., & Roushdy, M. I. (2024). Optimizing anomaly-based attack detection using classification machine learning. *Neural Computing and Applications*, *36*(6), 3239–3257. https://doi.org/10.1007/s00521-023-09309-y

Han, J., & Kamber, M. (1998). Data Mining: Concepts and Techniques. In *Morgan Kaufmann Publisher*. https://doi.org/10.3726/978-3-653-01927-8/2

Kanemoto, Y., Aoki, K., Iwamura, M., Miyoshi, J., Kotani, D., Takakura, H., & Okabe, Y. (2019). Detecting successful attacks from IDS alerts based on emulation of remote

shellcodes. *Proceedings - International Computer Software and Applications Conference*, *2*, 471–476. https://doi.org/10.1109/COMPSAC.2019.10251

Moon, D., Lee, J. K., & Yoon, M. K. (2022). Compact feature hashing for machine learning based malware detection. *ICT Express*, *8*(1), 124–129. https://doi.org/10.1016/j.icte.2021.08.005

Moustafa, N., Adi, E., Turnbull, B., & Hu, J. (2018). A New Threat Intelligence Scheme for Safeguarding Industry 4.0 Systems. *IEEE Access*, *6*, 32910–32924. https://doi.org/10.1109/ACCESS.2018.2844794

Patterson, C. M., Nurse, J. R. C., & Franqueira, V. N. L. (2023). Learning from cyber security incidents: A systematic review and future research agenda. *Computers & Security*, *132*, 103309.

Rajesh Bingu, E. al. (2023). Performance Comparison Analysis of Classification Methodologies for Effective Detection of Intrusions. *International Journal on Recent and Innovation Trends in Computing and Communication*, *11*(9), 2860–2879. https://doi.org/10.17762/ijritcc.v11i9.9375

Samantaray, M., Barik, R. C., & Biswal, A. K. (2024). A comparative assessment of machine learning algorithms in the IoT-based network intrusion detection systems. *Decision Analytics Journal*, *11*(May), 100478. https://doi.org/10.1016/j.dajour.2024.100478

Sarimuddin, S., Sari, J. Y., Mail, M., Masalu, M. A., Aristika, R. S., & Nurfagra, N. (2020). Klasifikasi Data Aging Tunggakan Nasabah Menggunakan Metode Decision Tree Pada ULaMM Unit Kolaka. *INFORMAL: Informatics Journal*, *5*(1), 26. https://doi.org/10.19184/isj.v5i1.16964

Singelton, C., Wikoff, A., & McMillen, D. (2021). IBM: 2021 X-Force Threat Intelligence Index. *Network Security*, 36. https://doi.org/10.1016/s1353-4858(21)00026-x

Yang, G., Chen, X., Zhou, Y., & Yu, C. (2022). DualSC: Automatic Generation and Summarization of Shellcode via Transformer and Dual Learning. *Proceedings - 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER 2022*, 361–372. https://doi.org/10.1109/SANER53432.2022.00052