Hyperparameter Tuning menggunakan *GridsearchCV* pada Random Forest untuk Deteksi *Malware*

Iik Muhamad Malik Matin
Teknik Informatika dan Komputer, Politeknik Negeri Jakarta
Kota Depok, Jawa Barat
iik.muhamad.malik.matin@tik.pnj.ac.id

Diterima: 16 Maret 2023. Disetujui: 27 Mei 2023. Dipublikasikan: 31 Mei 2023

Abstract - Random forest is a popular machine learning algorithm used for classification. In detecting malware, Random forest can help identify malware with good accuracy. However, in use, the random forest is only given one value for each hyperparameter so that the performance measurement is not optimal. for that required hyperparameter tuning process. GridsearchCV is a hyperparameter tuning method that allows the user to scan a number of selected hyperparameters. In this paper, experiments are carried out using GridsearchCV to perform hyperparameter tuning on random forests to detect malware. The hyperparameter tuning method using GridsearchCV produces the best parameters, namely entropy=criterion, max_depth with a value of 128, max_features=log2, max_samples_split=2, and n_estimators=400. The best parameter produces the highest performance increase in recall of 0.37% and each in accuracy and F1-score of 0.19%.

Keywords: Hyperparameter, GridsearchCV, Random Forest, Malware

Abstrak*—Random forest merupakan algoritma *machine learning* yang populer digunakan untuk klasifikasi. Dalam mendeteksi malware, Random forest dapat membantu mengidentifikasi malware dengan akurasi yang baik. Namun dalam penggunaannya, random forest hanya diberikan satu nilai setiap *hyperparameter*-nya sehingga pengukuran performa tidak maksimal. Untuk itu diperlukan proses *hyperparameter tuning*. GridsearchCV adalah metode *hyperparameter tuning* yang memungkinkan pengguna untuk melakukan pemindaian pada sejumlah *hyperparameter* yang dipilih. Dalam paper ini, dilakukan eksperimen dengan menggunakan *gridsearchCV* untuk melakukan *hyperparameter tuning* pada random forest untuk melakukan deteksi malware. Metode *hyperparameter tuning* menggunakan gridsearchCV menghasilkan parameter terbaik yaitu *entropy=criterion*, *max_depth* dengan nilai 128, *max_features=log2*, *max_samples_split=2*, dan *n_estimators=400*. Parameter terbaik menghasilkan peningkatan performa paling tinggi pada *recall* sebesar 0,37% dan masing-masing pada akurasi dan *F1-score* sebesar 0,19%.

Kata Kunci: Hyperparameter, GridsearchCV, Random Forest, Malware

I. PENDAHULUAN

Malware merupakan suatu istilah yang merujuk pada suatu perangkat lunak yang dapat merusak data, informasi, atau sistem komputer. *Malware* dapat dimanfaatkan untuk kejahatan dengan memasuki sistem tanpa otorisasi administrator komputer atau pengguna sistem tersebut seperti menyerang jaringan, merusak infrastruktur vital, membahayakan komputer dan perangkat pintar serta mencuri data sensitif [1][2].

Laporan Av-Test *Institute* [3] menunjukan pertumbuhan *malware* yang signifikan. Hal ini dapat dilihat dari tahun 2013 pertumbuhan *malware* sebanyak 37.511.458 hingga puncaknya pada tahun 2021 mencapai 150.931.321 *malware* dimana 95% diantaranya merupakan *malware* berbasis windows.

Deteksi malware saat ini dapat menggunakan *machine learning*. Deteksi *malware* menggunakan *machine learning* sangat andal dan mendukung metode statis maupun dinamis. Deteksi serangan dapat dilakukan secara efisien dengan deteksi berbasis ML [4]. *Machine learning* bekerja dengan belajar dari data yang tersedia sebagai pengetahuan. Pengetahuan yang didapatkan digunakan sebagai pertimbangan dalam melakukan deteksi.

Dalam mendeteksi *malware*, *machine learning* dikenal dengan metode *supervised* dan *unsupervised* [5], [6]. Konsep metode *supervised* adalah dengan memberikan input target dan mencocokan pola *malware* pada kelas yang ditentukan Proses pelatihan dilakukan terus-menerus sehingga model dapat memprediksi sampel secara benar [7]. Beberapa algoritma yang umum digunakan pada *supervised machine learning* diantaranya SVM [7]–[11]. *Decision Tree* [4], [11]–[14], k-NN [11], [15]. Naïve Bayes [16]–[18]. dan Random Forest [10], [13].

Random forest adalah salah satu algoritma klasifikasi berbasis *assemble learning* yang mudah menyesuaikan dengan dataset tertentu dengan menghitung biaya kesalahan klasifikasi tahap training. Random forest menerapkan nilai bobot ketika kelas minoritas salah diklasifikasikan atau dengan menyeimbangkan distribusi kelas untuk representasi yang sama di setiap *tree* [19]. Kekuatan random forest adalah dapat bekerja dengan baik pada kumpulan data yang besar dan ekspansif, memiliki metode yang kuat untuk memperkirakan data yang hilang dan mempertahankan presisi tanpa adanya proporsi data yang besar, memiliki teknik untuk menyeimbangkan kesalahan dalam kumpulan data yang tidak seimbang untuk populasi kelas [20].

Beberapa penelitian telah membuktikan kemampuan random forest dalam mendeteksi *malware* lebih baik dari algoritma lain seperti yang dilakukan oleh [21]–[23]. Namun, penggunaan random forest kurang optimal jika desain *machine learning* dirancang tanpa dilakukan optimasi. Selain itu, penggunaan random forest secara *default* tidak dapat menghasilkan performa yang optimal. Hal ini karena setiap parameter yang digunakan hanya memiliki satu nilai konstanta. Padahal, parameter yang digunakan belum tentu merupakan parameter yang optimal.

Beberapa penelitian berusaha meningkatkan performa dengan optimasi dilakukan oleh Heba Al-Harahsheh, dkk [24] yang menggunakan teknik *feature selection* dalam optimasi. 5 fitur seleksi digunakan yaitu *Chi-Square*, *Genetic programing Mean* (GPM), *Genetic programing Mean Plus* (GPMP), *Filter-based Filter*, *Wrapper-based*. Fitur seleksi untuk optimasi juga dilakukan oleh Ravi Kiran Varma dkk [25]. Selain itu, pada penelitian ini juga melakukan optimasi performa secara *heuristic* menggunakan *Cuckoo Search Optimization* dan Roughset. Penelitian [26] menggunakan *Ant Colony Optimization* dan *Rought Set* untuk mereduksi fitur yag tidak berguna untuk meningkatkan performa model *machine learning*. Dari beberapa penelitian tersebut belum melakukan optimasi pada *hyperparameter*. *Hyperparameter tuning* sangat penting dilakukan karena performa random forest sangat tergantung dari *hyperparameter* yang ditetapkan.

Dalam paper ini, dilakukan eksperimen dengan menggunakan *GridsearchCV* untuk melakukan *hyperparameter tuning* pada random forest untuk mendeteksi *malware*. *Hyperparameter tuning* bekerja dengan menemukan *hyperparameter set* yang paling optimal. Caranya, *hyperparameter* diuji secara *trial and error* sampai menemukan nilai optimal [27]. Makalah ini menggunakan metode *Gridsearch* yang disediakan oleh Sckit-learn [28] untuk menemukan *hyperparameter* terbaik. *GridsearchCV* adalah metode *hyperparameter tuning* yang memungkinkan untuk melakukan pemindaian pada sejumlah *hyperparameter* yang dipilih. Dalam *GridsearchCV*, sejumlah kombinasi *hyperparameter* akan diterapkan pada model dan performa dari setiap kombinasi akan dievaluasi menggunakan *cross-validation*. Kombinasi *hyperparameter* dengan performa terbaik akan dipilih sebagai *hyperparameter* terbaik untuk model.

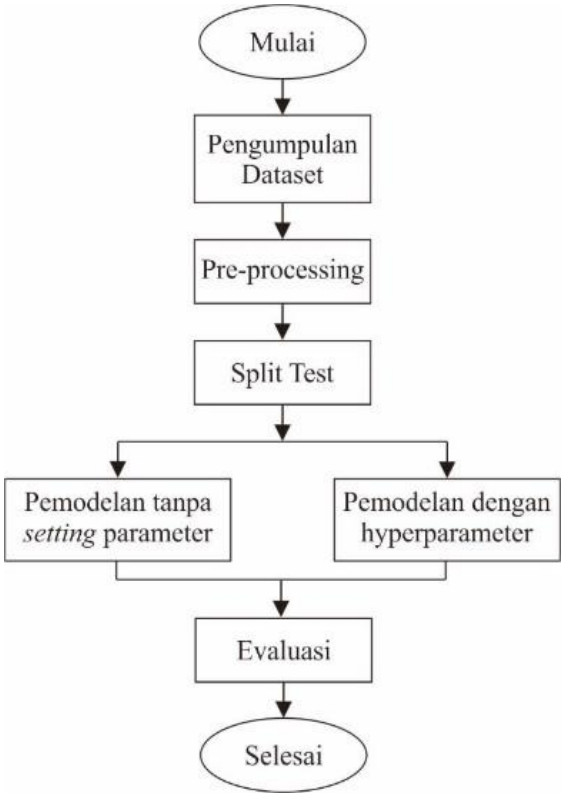
Kontribusi dari penelitian ini sebagai berikut:

- Membangun model klasifikasi malware menggunakan algoritma random forest
- Mencari arsitektur model terbaik untuk klasifikasi malware berdasarkan *hyperparameter* random forest
- Membandingkan akurasi model pembelajaran random forest dengan parameter default dengan *hyperparameter tuning*

Hasil eksperimen ini akan dianalisis untuk menentukan apakah *hyperparameter tuning* meningkatkan performa model dalam mengidentifikasi *malware*. Penelitian lanjutan juga akan membahas bagaimana hasil eksperimen ini dapat digunakan untuk memperbaiki deteksi *malware* pada masa depan.

II. METODE PENELITIAN

Penelitian ini dilakukan untuk mengkaji performansi dari model dengan mengubah nilai *hyperparameter* agar dapat mengklasifikasikan malware lebih optimal. Alur pengembangan model dilakukan dengan tahapan-tahapan yang ditunjukan pada gambar 1.



Gambar 1. Alur Pengembangan Model

Alur pengembangan model yang ditunjukkan oleh gambar 1 terdiri dari 4 tahapan yaitu pengumpulan dataset, *pre-processing*, *split test*, pemodelan yang terdiri dari model *default* dan model *hyperparameter*, dan evaluasi model.

A. *Dataset*

Pada penelitian ini, dataset yang digunakan adalah *Classification of Malware with PE Header* (ClamP) [29]. Dataset ClamP merupakan dataset yang terdiri dari 77 fitur dalam dataset yang diekstraksi dari header PE dari *file executable*. *Header* PE memiliki 4 bagian penting yang terdiri dari [30]:

a. *DOS Header Feature*

DOS Header atau MS-DOS Header menunjukan bahwa *file* merupakan jenis PE. DOS *header* terdiri dari panjang 64-bit. Fitur *e_magic* pada DOS *header* dengan nilai heksadesimal 4D5A yang berarti 'MZ' diawal menunjukan bahwa *file* dapat dieksekusi MS-DOS [31]. Tabel 1 menunjukan fitur pada DOS *header* dan penjelasannya.

b. *PE Header Feature*

Fitur PE *Header* terdapat pada depan *file* objek atau setelah *signature image file*. Tabel II menunjukan fitur-fitur yang *header file* PE. Utama terdapat pada *Timedatetestamp*. Fitur *Timedatetestamp* diubah saat membuat *malware* pertama kali.

c. *Optional Header Feature*

Header ini menjelaskan struktur dari *file* header PE. *Header* ini berfungsi untuk memuat informasi pada *file image* [32]. *Optional header* terdiri dari versi ukuran yang ditunjukan pada Tabel III dan versi atribut yang ditunjukan pada Tabel IV sedangkan Tabel V menunjukan fitur *optional header* yang menyertakan lokasi.

d. *Section Header*

Section Header atau *section table* adalah array yang bertipe *IMAGE_SECTION_HEADER* yang menjelaskan setiap bagian-bagian PE. Setiap tabel berisi informasi tentang bagian dalam *file* PE seperti atribut dan *offset* virtualnya, seperti ditampilkan pada Tabel VI.

TABEL I. DOS *HEADER*

Fitur	Deskripsi
e-magic	Nilai *Magic*
e-cblp	Nilai Bit pada akhi halaman *file*
e_cparhdr	Jumlah halaman pada *file*
e_maxalloc	Nilai maksimum paragraf yang dibutuhkan
e_sp	Nilai inisial sp
e_lfanew	Alamat *file header* exe baru
e_sum	Nilai *checksum*
e_minalloc	Nilai minimum alokasi paragraf yang dibutuhkan

TABEL II. PE HEADER

Fitur	Deskripsi
Timestampdate	Tanggal dan waktu *file* dibuat
NumberofSection	Ukuran bagian tabel windows
Symbolatribute	Menentukan lokasi dan ukuran COFF *Header*
DLL Characteristic	Terdiri dari kombinasi 16 fitur berbeda

TABEL III. *OPTIONAL HEADER* VERSI UKURAN

Fitur	Deskripsi
MajorLinkVersion	Nilai versi mayor pada *linker*
MinorLinkVersion	Nilai *linkers minor version*
MajorOperatingsystemVersion	Nilai versi major dari OS
MinorrOperatingsystemVersion	Nilai *version pada OS minor*
Majorsubsystem	Nilai versi pada *subsystem* Major
Minorsubsystem	Jumlah versi *Subsystem Minor*

TABEL IV. *OPTIONAL HEADER* VERSI ATRIBUT

Fitur	Deskripsi
SizeofCode	Ukuran *size section*
SizeofIntializedData	Ukuran data yang diinisiasi
SizeofuninitializedDat a	Ukuran data yang tidak diinisiasi
SizeofImage	Ukuran *file image*
Sizeofheader	Ukuran semua bagian *header*
SizeofStackreserve	Jumlah *byte* yang disimpan dalam *stack*
SizeofStackcommit	Jumlah *byte* yang diperlukan untuk melakukan *stack*
SizeofHeapreserve	Jumlah data yang disimpan dalam *heap*
SizeofHeapcommit	Jumlah byte yang diperlukan untuk melakukan *heap*
SizeofOptionalHeader	menunjukan ukuran *header* yang tidak tetap

TABEL V. FITUR *SECTION HEADER*

Fitur	Deskripsi
Raw size	ukuran yang dapat disimpan kedalam disk
Virtual size	Bagian yang dapat disimpan kedalam memori
Virtual address	alamat memori virtual
Physical address	alamat memori fisik
Entropy	nilai yang dihitung dari *header* eksternal

TABEL VI. FITUR SECTION HEADER

Fitur	Deskripsi
Section Alignment	Bagian baris yang dimuat kedalam memori
File Alignment	Bagian baris raw data pada image *file*
Baseofcode	Pointer di awal kode
BaseofData	Pointer di awal bagian data
Image Base	Alamat *byte* pertama saat gambar dimuat dalam memori

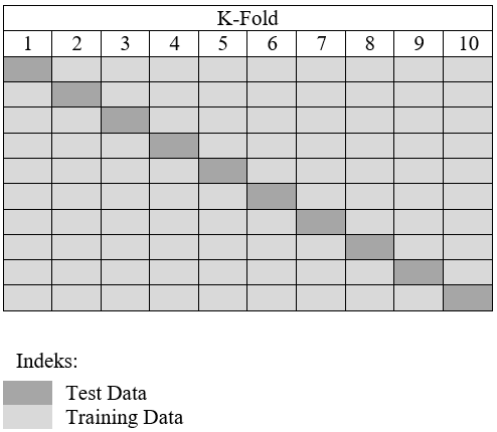
B. *Preprocessing*

Preprocessing adalah tahapan memodifikasi data agar data dapat lebih representatif. Pada penelitian ini dilakukan *preprocessing* berupa:

- *Data cleaning*, yaitu menghilangkan data yang tidak konsisten, dan data yang tidak ada nilainya.
- Data *reduction*, yaitu menghilangkan data yang memiliki nilai yang sama atau menduplikasi.
- *Data transformation*, yaitu mengubah bentuk yang lebih sesuai.

C. *Split test*

Split test adalah tahapan untuk membagi data ke dalam beberapa bagian. Pada penelitian ini *split test* dibagi untuk *test data*, *training data*. Pada penelitian ini rasio *training data* dan *testing data* sebesar 80:20. Selain itu, pada *training data* digunakan *cross validation* untuk meningkatkan validitas model. *Cross validation* yang digunakan adalah *kfold validation* dengan jumlah 10 kali validasi. Gambar 2 menunjukan ilustrasi 10 k-*fold validation*.



Gambar 2. *Cross Validation* dengan K=10

D. Pemodelan

Pada penelitian ini terdiri dari 2 model prediksi yaitu model random forest tanpa *hyperparameter tuning* yang artinya semua parameter yang digunakan mengikuti *setting* secara *default*. Sedangkan model yang kedua menggunakan random forest dengan mengubah nilai *hyperparameter* yang paling penting pada random forest yang terdiri dari:

a. *Criterion*
   Criterion adalah fungsi untuk mengukur kualitas *split*. Input *criterion* yaitu *gini*, *entropy*, dan *log_loss.*
b. *Max_Depth*
   Max_Depth digunakan untuk menentukan jumlah kedalaman yang diizinkan dalam sebuah *tree*. Nilai *max_depth* diinputkan dengan nilai *integer.*
c. *N_Estimator*
   n_estimator adalah *hyperparameter* random forest yang berfungsi untuk menentukan jumlah *tree* dalam sebuah pohon keputusan. *N_estimator* menunjukan pengaruh pada akurasi model random forest [33]. *Max_depth* diberikan nilai *integer*.
d. *Max_Features*

Max_features adalah *hyperparameter* untuk menentukan jumlah fitur yang dipertimbangkan saat mencari *split* terbaik. *Max_features* diinputkan dengan nilai *sqrt*, *log2* dan *none*.

e. *Min_samples_split*
Min_samples_split digunakan untuk menentukan nilai minimum sampel yang diperlukan untuk membagi simpul internal. *Min_samples_split* diinputkan dengan nilai *integer* atau *float*.

Pada penelitian ini ditentukan nilai *hyperparameter tuning* yang ditunjukan pada Tabel VII.

TABEL VII. *HYPERPARAMETER SET*

Hyperparameter	**Nilai**
Criterion	*Gini*, *Entropy*, *Log_Loss*
Max_Depth	32, 64, 112, 128
N_Estimator	7, 100, 125, 150
Max_Feature	Sqrt Log2
Min_Samples_Split	2,4,6

E. *Evaluasi*

Dataset ClamP memiliki 2 jenis klasifikasi yaitu jinak dan *malware* sehingga dapat terjadi empat kemungkinan keluaran hasil klasifikasi yang ditunjukan pada Gambar 3.

		Predicted label	
		Benign	Malware (Intrusion)
True label	Benign	TN	FP
	Malware (Intrusion)	FN	TP

Gambar 3. *Confusion Matrix*

Terdapat empat kemungkinan yang ditunjukan pada Gambar 3 dapat dijelaskan sebagai berikut:

1. *True Negative* (*TN*)
   Hasil pengukuran *true negative* menunjukan jumlah jinak yang dapat diidentifikasi dengan benar.
2. *False Negative* (*FN*)
   Hasil pengukuran *false negative* menunjukan hasil klasifikasi merupakan jinak, padahal yang diidentifikasi adalah *malware*.
3. *True Positive* (*TP*)
   Hasil pengukuran *true positive* menunjukan jumlah *malware* yang teridentifikasi dengan benar.
4. *False Positive* (FP)
   Hasil pengukuran *false positive* menunjukan *malware* padahal yang diidentifikasi adalah jinak.

Keempat kemungkinan keluaran hasil klasifikasi digunakan untuk mengukur performa model yang terdiri dari akurasi, presisi, TPR (*Recall*) dan F1-s*core* dengan formula berikut ini:

$$\text{Akurasi} = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

$$\text{presisi} = \frac{TP}{TP+FP} \quad (2)$$

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

$$F - Score-= 2 \times \frac{Precision \times TPR}{Precision+TPR} \quad (4)$$

Akurasi (1) adalah rasio yang diidentifikasi benar. Presisi (2) adalah relevansi dalam mengidentifikasi dari hasil klarifikasi yang diberikan keluaran *malware*. *True Positive Rate* (TPR) atau *recall* (3) adalah rasio dari tingkat keberhasilan dalam melakukan identifikasi *malware*. *F-score* (4) merupakan tingkat akurasi yang dihitung berdasarkan ketepatan dan penarikan kembali suatu tes.

III. HASIL DAN PEMBAHASAN

Hyperparameter tuning berguna untuk meningkatkan performa model *machine learning* dengan cara menentukan nilai-nilai *hyperparameter*. *GridsearchCV* digunakan untuk menguji model secara *bruteforce* menggunakan nilai-nilai *hyperparameter* yang telah ditentukan. Pada penelitian ini telah dilakukan *hyperparameter tuning* menggunakan *gridsearchCV* pada algoritma random forest untuk mendeteksi *malware* untuk mendapatkan hasil yang optimal.

Pada penelitian ini diusulkan optimasi performa random forest dengan mengkonfigurasi *hyperparameter* dengan metode *gridsearchCV*. Untuk menentukan performa mana yang lebih baik, dibangun model dengan 2 kondisi yaitu pada saat *hyperparameter* random forest dalam kondisi *default* dan kondisi saat *hyperparameter* random forest telah diset. Performa yang dihasilkan dari kedua kondisi

model dapat diketahu dengan membandingkan kedua model tersebut.

Pada penelitian ini dataset yang digunakan berasal dari ClaMP malware. *Hyperparameter tuning* yang digunakan terdiri dari *gini, entropy, log_loss* untuk *criterion*; 32, 64, 112 untuk *max_depth*, 25, 50, 100, 200, 400, dan 800 untuk *n_estimator*; *sqrt* dan *log2* untuk *max_feature*, dan 2, 4, 8, 16, dan 32 untuk *min_samples_split*. Untuk mengetahui nilai optimasi yang didapatkan, dilakukan perbandingan dengan model dengan *hyperparameter* ditentukan secara *default*.

Tabel VIII menunjukan perbedaan *hyperparameter* random forest saat *default* dan *hyperparameter tuning* pada random forest.

TABEL VIII. PERBANDINGAN *HYPERPARAMETER*

Hyperparameter	***Hyperparameter tuning***	***Default***
Criterion	*Gini, Entropy, Log_Loss*	*Gini*
Max_Depth	32, 64, 112, 128	-
N_Estimator	7, 100, 125, 150	100
Max_Feature	Sqrt Log2	Sqrt
Min_Samples_Split	2,4,6	2

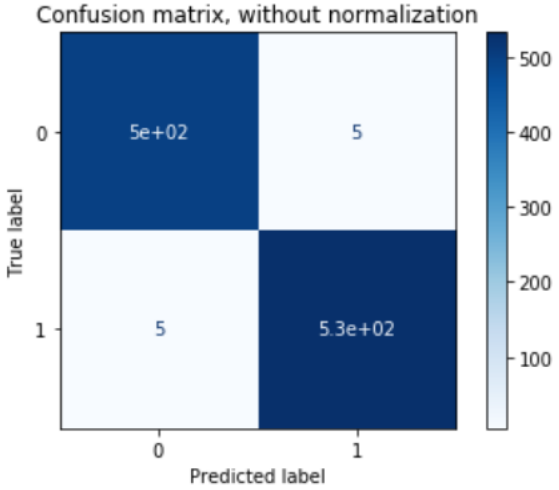
Dataset yang digunakan adalah *Classification of Malware with PE Header* (ClamP). Hanya data yang memiliki nilai yang lengkap yang diproses. Artinya, jika terjadi data kosong maka data tersebut dihapus. Dataset yang telah diekplorasi data kemudian dibagi menjadi 4 bagian yaitu:

- *X_train*: Data kelas untuk pelatihan
- *X_test*: Data kelas untuk pengujian
- *y_train*: Data fitur untuk pelatihan
- *y_test*: Data fitur untuk pengujian

untuk data *training* ditetapkan sebanyak 80% dan data testing sebanyak 20%. *Cross validation* sebanyak 10 kali.

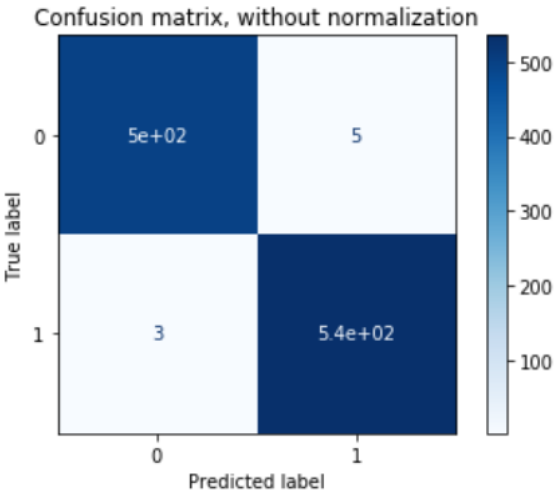
Pada model pertama digunakan *GridsearchCV* dengan memberikan nilai pada parameter *scoring* terdiri dari *accuray, precision, recall, f1* kemudian menetapkan nilai *hyperparameter* yang telah ditentukan pada tabel 7. Pada model kedua digunakan random forest secara *default* sehingga tidak perlu konfigurasi *hyperparameter tuning* lagi.

Hasil pada model random forest *default* menunjukan 499 *malware* terdeteksi dengan benar atau true negatif. 5 *malware* teridentifikasi jinak atau false positive, 5 jinak diidentifikasi *malware* atau false negatif, dan 533 jinak diidentifikasi secara tepat atau false positif. Hasil tersebut dapat digambarkan dalam *confusion matrix* pada gambar 4.



Gambar 4. *Confusion Matrix* pada *Random Forest Default*

Hasil pada model random forest dengan *hyperparameter tuning* didapat model optimal dengan masing-masing nilai *entropy* pada *criterion*, 128 pada *max_depth*, *log2* pada *max_features*, 2 pada *max_samples_slit*, dan 400 pada *n_estimators* dengan sebanyak 499 *malware* terdeteksi dengan benar atau true negatif. 5 *malware* teridentifikasi jinak atau *false positive*, 3 jinak diidentifikasi *malware* atau *false negative*, dan 535 jinak diidentifikasi secara tepat atau *false positive*. Jumlah sampel yang terdeteksi ditunjukan pada gambar 5.



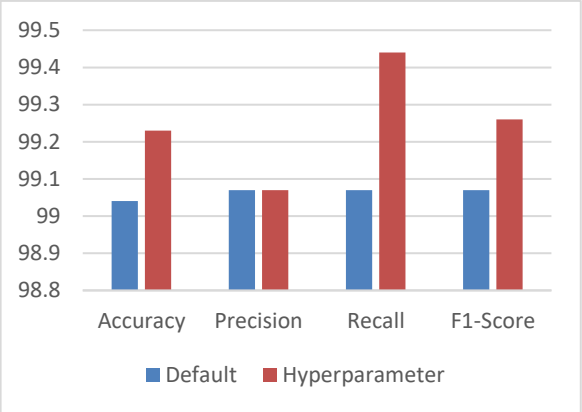
Gambar 5. *Confusion Matrix* pada *Random Forest* dengan *Hyperparameter Tuning*.

Dari hasil *confusion matrix* yang didapat pada gambar 4 dan gambar 5 maka dapat dihitung akurasi, presisi, TPR, dan f1 dan perbandingan antara random forest *default* dengan random forest dengan *hyperparameter tuning* yang ditunjukan pada Tabel IX.

TABEL IX. HASIL PERFORMA RANDOM FOREST

Metric	***Default***	***Hyperparameter tuning***
Accuracy	99,04	99,23
Precision	99,07	99,07
Recall	99,07	99,44
F1-Score	99,07	99,26

Dari Tabel IX dapat dilihat bahwa terdapat peningkatan tertinggi yaitu pada *recall* sebesar 0,37% dan masing-masing pada akurasi dan F1-*score* sebanyak 0,19% sedangkan pada *precision* tidak mengalami peningkatan karena hasil prediksi *malware* terhadap keseluruhan antara random forest *default* dengan *hyperparameter tuning* memiliki nilai yang sama. Gambar 6 menggambarkan selisih performa antara mode *default* dan *hyperparameter*.



Gambar 6. Perbandingan performa random forest pada mode *default* dan *hyperparameter*

IV. KESIMPULAN DAN SARAN

Model klasifikasi *malware* menggunakan algoritma random forest telah dibangun. Model yang optimal didapatkan dengan *hyperparameter tuning* masing-masing parameter *criterion=entropy*, max_depth=128, max_features=log2, max_samples_split=2, dan n_estimators=400. *Hyperparameter* tersebut telah meningkatkan performa model terbaik dengan 99,23 % pada akurasi, 99,7% pada presisi, 99,44% pada TPR dan 99,26% pada F1-Score. Peningkatan paling tinggi pada recall sebesar 0,37% dan masing-masing pada akurasi dan F1-score sebesar 0,19%.

Peningkatan model para random forest menggunakan parameter yang masih terbatas, sehingga masih bisa dikembangkan lebih jauh lagi. Kesempatan penelitian ke depan dapat dikembangkan pada nilai-nilai *hyperparameter tuning* yang lebih luas, algoritma yang lebih variatif dan fitur seleksi untuk mendapatkan performa model yang lebih baik lagi.

REFERENSI

[1] H. Rathore, S. Agarwal, S. K. Sahay, and M. Sewak, "Malware detection using machine learning and deep learning," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11297 LNCS, pp. 402–411, 2018, doi: 10.1007/978-3-030-04780-1_28.

[2] Y. Kamalrul Bin Mohamed Yunus and S. Bin Ngah, "Review of Hybrid Analysis Technique for Malware Detection," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 769, no. 1, 2020, doi: 10.1088/1757-899X/769/1/012075.

[3] Av-Test Institute, "Malware Statistic & Trend Report." [Online]. Available: https://portal.av-atlas.org/malware/statistics. [Accessed: 24-Feb-2023].

[4] S. J. Kattamuri, R. K. V. Penmatsa, S. Chakravarty, and V. S. P. Madabathula, "Swarm Optimization and Machine Learning Applied to PE Malware Detection towards Cyber Threat Intelligence," *Electronics*, vol. 12, no. 2, p. 342, 2023, doi: 10.3390/electronics12020342.

[5] M. A. Jerlin and K. Marimuthu, "A New Malware Detection System Using Machine Learning Techniques for API Call Sequences," *J. Appl. Secur. Res.*, vol. 13, no. 1, pp. 45–62, 2018, doi: 10.1080/19361610.2018.1387734.

[6] A. Abdallah *et al.*, "An Optimal Framework for SDN Based on Deep Neural Network," *Comput. Mater. Contin.*, vol. 73, no. 1, pp. 1125–1140, 2022, doi: 10.32604/cmc.2022.025810.

[7] S. R. T. Mat, M. F. A. Razak, M. N. M. Kahar, J. M. Arif, and A. Firdaus, "A Bayesian probability model for Android malware detection," *ICT Express*, vol. 8, no. 3, pp. 424–431, 2022, doi: 10.1016/j.icte.2021.09.003.

[8] E. S. Lamdompak Sistem Komputer and F. Ilmu Komputer, "Klasifikasi Malware Trojan Ransomware Dengan Algoritma Support Vector Machine (SVM)," vol. 2, no. 1, pp. 122–127, 2016.

[9] R. Chaganti, V. Ravi, and T. D. Pham, "A multi-view feature fusion approach for effective malware classification using Deep Learning," *J. Inf. Secur. Appl.*, vol. 72, 2023, doi: 10.1016/j.jisa.2022.103402.

[10] A. Y. Daeef, A. Al-Naji, A. K. Nahar, and J. Chahl, "Features Engineering to Differentiate between Malware and Legitimate Software," *Appl. Sci.*, vol. 13, no. 3, 2023, doi: 10.3390/app13031972.

[11] D. Yuxin and Z. Siyi, "Malware detection based on deep learning algorithm," *Neural Comput. Appl.*, vol. 31, no. 2, pp. 461–472, 2019, doi: 10.1007/s00521-017-3077-6.

[12] M. H. L. Louk and B. A. Tama, "*Tree*-Based Classifier Ensembles for PE Malware Analysis: A Performance Revisit," *Algorithms*, vol. 15, no. 9, pp. 1–15, 2022, doi: 10.3390/a15090332.

[13] N. A. Azeez, O. E. Odufuwa, S. Misra, J. Oluranti, and R. Damaševičius, "Windows PE malware detection using ensemble learning," *Informatics*, vol. 8, no. 1, 2021, doi: 10.3390/informatics8010010.

[14] H. S. Anderson and P. Roth, "EMBER: An Open Dataset

for Training Static PE Malware Machine Learning Models," 2018.

[15] S. Choi, "Combined kNN classification and hierarchical similarity hash for fast malware detection," *Appl. Sci.*, vol. 10, no. 15, pp. 1–16, 2020, doi: 10.3390/app10155173.

[16] E. M. Alkhateeb and M. Stamp, "A Dynamic Heuristic Method for Detecting Packed Malware Using Naive Bayes," *2019 Int. Conf. Electr. Comput. Technol. Appl. ICECTA 2019*, 2019, doi: 10.1109/ICECTA48151.2019.8959765.

[17] L. Sayfullina *et al.*, "Efficient detection of zero-day android malware using normalized bernoulli naive bayes," *Proc. - 14th IEEE Int. Conf. Trust. Secur. Priv. Comput. Commun. Trust. 2015*, vol. 1, pp. 198–205, 2015, doi: 10.1109/Trustcom.2015.375.

[18] P. D. Utami and R. Sari, "Filtering Hoax Menggunakan Naive Bayes Classifier," *Multinetics*, vol. 4, no. 1, p. 57, 2018, doi: 10.32722/vol4.no1.2018.pp57-61.

[19] L. Breiman, "Random Forests," *Mach. Learn.*, vol. 45, pp. 5–32, 2001, doi: 10.1109/ICCECE51280.2021.9342376.

[20] P. Agrawal and P. Trivedi, "Android Malware Detection Using Machine Learning Classifiers," in *In Data Management, Analytics and Innovation: Proceedings of ICDMAI 2020*, 2021, vol. 1, doi: 10.1007/978-981-19-3035-5_15.

[21] A. Hussain, M. Asif, M. Bin Ahmad, T. Mahmood, and M. A. Raza, "Malware Detection Using Machine Learning Algorithms for Windows Platform," *Lect. Notes Networks Syst.*, vol. 350, pp. 619–632, 2022, doi: 10.1007/978-981-16-7618-5_53.

[22] C. D. Morales-Molina, D. Santamaria-Guerrero, G. Sanchez-Perez, H. Perez-Meana, and A. Hernandez-Suarez, "Methodology for malware classification using a random forest classifier," *2018 IEEE Int. Autumn Meet. Power, Electron. Comput. ROPEC 2018*, no. Ropec, pp. 1–6, 2019, doi: 10.1109/ROPEC.2018.8661441.

[23] S. Naz and D. K. Singh, "Review of Machine Learning Methods for Windows Malware Detection," *2019 10th Int. Conf. Comput. Commun. Netw. Technol. ICCCNT 2019*, pp. 6–11, 2019, doi: 10.1109/ICCCNT45670.2019.8944796.

[24] H. Al-Harahsheh, M. Al-Shraideh, and S. Al-Sharaeh, "Performance of Malware Detection Classifier Using Genetic Programming in Feature Selection," *Inform.*, vol. 45, no. 4, pp. 517–529, 2021, doi: 10.31449/INF.V45I4.3819.

[25] R. K. P. Varma, P. Raju, K. V. S. Raju, and A. Kalidindi, "Feature selection and performance improvement of malware detection system using cuckoo search optimization and rough sets," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 5, pp. 708–714, 2022, doi: 10.14569/IJACSA.2020.0110587.

[26] R. K. V. Penmatsa, A. Kalidindi, and S. K. R. Mallidi, "Feature Reduction and Optimization of Malware Detection System Using Ant Colony Optimization and Rough Sets," *Int. J. Inf. Secur. Priv.*, vol. 14, no. 3, pp. 95–114, 2020, doi: 10.4018/ijisp.2020070106.

[27] B. H. Shekar and G. Dagnew, "Grid search-based *hyperparameter tuning* and classification of microarray cancer data," *2019 2nd Int. Conf. Adv. Comput. Commun. Paradig. ICACCP 2019*, pp. 1–8, 2019, doi: 10.1109/ICACCP.2019.8882943.

[28] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *J. Mach. Learn. Res.*, vol. 127, no. 9, pp. 2825–2830, 2011, doi: 10.1289/EHP4713.

[29] A. Kumar, "ClaMP (Classification of Malware with PE headers)." [Online]. Available: https://github.com/urwithajit9/clamp. [Accessed: 21-Jul-2020].

[30] Y. Liao, "PE-Header-Based Malware Study and Detection," *Univ. Georg. Georg. US*, p. 4, 2018.

[31] F. Zatloukal and J. Znoj, "Malware Detection Based on Multiple PE Headers Identification and Optimization for Specific Types of Files," *J. Adv. Eng. Comput.*, vol. 1, no. 2, p. 153, 2017, doi: 10.25073/jaec.201712.64.

[32] D. Devi and S. Nandi, "PE File Features in Detection of Packed Executables," *Int. J. Comput. RTheory Eng.*, vol. 4, pp. 476–478, 2012.

[33] Q. Zhou, W. Lan, Y. Zhou, and G. Mo, "Effectiveness Evaluation of Anti-bird Devices based on Random Forest Algorithm," *2020 7th Int. Conf. Information, Cybern. Comput. Soc. Syst. ICCSS 2020*, pp. 743–748, 2020, doi: 10.1109/ICCSS52145.2020.9336891.