Building of Informatics, Technology and Science (BITS)
Volume 7, No 4, March 2026 Page: 2507−2517
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i4.9443

Optimasi Deteksi Malware Android pada Dataset Drebin Menggunakan Ensemble Learning

Haidar Nafiis Usmany*, Wildanil Ghozi

Fakultas Ilmu Komputer, Program Studi Teknik Informatika, Universitas Dian Nuswantoro, Semarang, Indonesia
Email: [1,*] 111202214146@mhs.dinus.ac.id [2,] wildanil.ghozi@dsn.dinus.ac.id
Email Penulis Korespondensi: 111202214146@mhs.dinus.ac.id
Submitted: **24/02/2026**; Accepted: **19/03/2026**; Published**: 19/03/2026**

Abstrak−Peningkatan jumlah dan kompleksitas malware Android menuntut sistem deteksi yang akurat, efisien, serta mampu menangani data berdimensi tinggi. Pendekatan berbasis machine learning menjadi salah satu solusi yang banyak digunakan dalam penelitian keamanan siber. Namun demikian, kinerja model klasifikasi sering dipengaruhi oleh redundansi fitur dan konfigurasi hyperparameter yang kurang optimal. Penelitian ini bertujuan untuk mengevaluasi efektivitas kombinasi seleksi fitur berbasis Random Forest dengan algoritma klasifikasi boosting modern dalam mendeteksi malware Android. Dataset yang digunakan adalah Drebin 215, yang dipilih karena merupakan salah satu dataset benchmark yang paling banyak digunakan dalam penelitian deteksi malware Android berbasis analisis statis, sehingga memungkinkan perbandingan hasil penelitian secara lebih objektif. Proses seleksi fitur dilakukan menggunakan metode *feature importance* dari Random Forest untuk mereduksi dimensi data sebelum tahap klasifikasi. Model yang digunakan meliputi XGBoost, Light Gradient Boosting Machine (LightGBM), dan CatBoost. Pengujian dilakukan pada dua skenario, yaitu tanpa optimasi hyperparameter (non-tuning) dan dengan optimasi menggunakan metode Grid Search. Evaluasi performa dilakukan menggunakan metrik accuracy, precision, recall, F1-score, dan ROC-AUC serta dianalisis dari sisi waktu komputasi. Hasil eksperimen menunjukkan bahwa seluruh model mampu mencapai performa klasifikasi yang sangat baik pada dataset benchmark Drebin, dengan nilai accuracy di atas 0.98. LightGBM menunjukkan performa terbaik, dengan accuracy sebesar 0.9900 dan F1-score sebesar 0.9865, yang diduga dipengaruhi oleh efisiensi mekanisme *histogram-based learning* dan strategi *leaf-wise tree growth* yang memungkinkan pembelajaran lebih cepat dan efektif pada data berdimensi tinggi. Meskipun demikian, performa tinggi yang diperoleh pada dataset benchmark ini masih memerlukan evaluasi lebih lanjut pada dataset yang lebih beragam atau lingkungan yang lebih dinamis untuk memastikan kemampuan generalisasi model dalam skenario dunia nyata. Hasil penelitian ini menunjukkan bahwa kombinasi seleksi fitur berbasis Random Forest dan algoritma boosting dapat menjadi pendekatan yang efektif dalam meningkatkan efisiensi dan performa sistem deteksi malware Android.

Kata Kunci: Android Malware Detection; Random Forest Feature Selection; Ensemble Learning; Drebin Dataset; Hyperparameter Tuning

Abstract−The increasing number and complexity of Android malware require detection systems that are accurate, efficient, and capable of handling high-dimensional data. Machine learning–based approaches have become one of the widely adopted solutions in cybersecurity research. However, the performance of classification models is often affected by feature redundancy and suboptimal hyperparameter configurations. This study aims to evaluate the effectiveness of combining Random Forest–based feature selection with modern boosting classification algorithms for Android malware detection. The dataset used in this study is the Drebin 215 dataset, which was selected because it is one of the most widely used benchmark datasets for Android malware detection based on static analysis, enabling more objective comparison with previous studies. Feature selection was performed using the Random Forest feature importance method to reduce data dimensionality prior to the classification stage. The classification models employed include XGBoost, Light Gradient Boosting Machine (LightGBM), and CatBoost. The experiments were conducted under two scenarios: without hyperparameter optimization (non-tuning) and with hyperparameter optimization using the Grid Search method. Model performance was evaluated using accuracy, precision, recall, F1-score, and ROC-AUC metrics, as well as computational time analysis. The experimental results show that all models achieved very strong classification performance on the Drebin benchmark dataset, with accuracy values exceeding 0.98. Among the evaluated models, LightGBM achieved the best performance, with an accuracy of 0.9900 and an F1-score of 0.9865. This performance advantage is likely influenced by the efficiency of its histogram-based learning mechanism and leaf-wise tree growth strategy, which enables faster and more effective learning on high-dimensional data. Nevertheless, the high performance observed on this benchmark dataset still requires further evaluation on more diverse datasets or dynamic environments to ensure the generalization capability of the model in real-world scenarios. The findings of this study indicate that the combination of Random Forest–based feature selection and boosting algorithms can serve as an effective approach for improving the efficiency and performance of Android malware detection systems.

Keywords: Android Malware Detection; Random Forest Feature Selection; Ensemble Learning; Drebin Dataset; Hyperparameter Tuning

1. PENDAHULUAN

Perkembangan pesat teknologi perangkat bergerak, khususnya sistem operasi Android, telah memberikan kemudahan akses informasi dan layanan digital. Android saat ini menjadi sistem operasi mobile yang paling dominan secara global sehingga banyak digunakan dalam berbagai aktivitas seperti komunikasi, transaksi keuangan, hingga penyimpanan data pribadi. Namun, dominasi tersebut juga menjadikan Android sebagai target utama serangan siber, terutama dalam bentuk malware yang dapat mencuri data sensitif, melakukan penipuan finansial, hingga mengambil alih kendali perangkat tanpa izin. Seiring dengan meningkatnya jumlah aplikasi Android yang beredar, jumlah dan kompleksitas varian malware juga terus berkembang sehingga menimbulkan tantangan serius dalam menjaga keamanan ekosistem perangkat bergerak[1].

Building of Informatics, Technology and Science (BITS)
Volume 7, No 4, March 2026 Page: 2507−2517
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i4.9443

Metode deteksi malware Android secara tradisional umumnya menggunakan pendekatan berbasis tanda tangan atau signature-based detection. Pendekatan ini bekerja dengan mencocokkan pola kode aplikasi dengan basis data signature malware yang telah diketahui sebelumnya. Meskipun metode ini relatif cepat dan mudah diimplementasikan, pendekatan tersebut memiliki keterbatasan dalam mendeteksi varian malware baru atau malware yang menggunakan teknik penyamaran tertentu. Teknik seperti code obfuscation, code encryption, packing, dan control flow modification dapat mengubah struktur kode aplikasi tanpa mengubah fungsinya, sehingga malware yang dimodifikasi sering kali tidak dapat dikenali oleh sistem deteksi berbasis signature. Selain itu, metode ini memerlukan pembaruan basis data secara berkala agar mampu mengenali malware terbaru, sehingga kurang efektif dalam menghadapi perkembangan malware yang sangat cepat [2].

Dalam beberapa tahun terakhir, pendekatan berbasis machine learning (ML) semakin banyak digunakan dalam penelitian deteksi malware Android. Metode machine learning mampu mempelajari pola kompleks dari fitur aplikasi, seperti permission, API call, system command, dan komponen aplikasi lainnya, sehingga dapat membedakan aplikasi berbahaya dan aplikasi benign secara lebih efektif. Berbagai algoritma klasifikasi seperti Random Forest, Support Vector Machine, dan Decision Tree telah banyak diterapkan dalam penelitian sebelumnya dan menunjukkan performa yang cukup baik dalam mendeteksi malware Android [3],[4]. Meskipun demikian, penerapan machine learning dalam deteksi malware masih menghadapi beberapa tantangan utama, terutama terkait dengan tingginya dimensi fitur, adanya redundansi informasi, serta potensi overfitting yang dapat menurunkan kemampuan generalisasi model terhadap data baru.

Salah satu pendekatan yang banyak digunakan untuk mengatasi permasalahan tersebut adalah seleksi fitur. Seleksi fitur bertujuan untuk mengurangi dimensi data dengan mempertahankan fitur-fitur yang paling relevan terhadap proses klasifikasi. Dengan mengurangi fitur yang tidak penting atau redundan, seleksi fitur dapat meningkatkan efisiensi komputasi serta membantu model dalam mempelajari pola data secara lebih efektif. Beberapa penelitian menunjukkan bahwa integrasi metode seleksi fitur mampu meningkatkan stabilitas dan performa model deteksi malware Android [1], [2], [5]. Salah satu metode seleksi fitur yang cukup efektif adalah feature importance berbasis Random Forest. Metode ini mengevaluasi kontribusi setiap fitur berdasarkan penurunan nilai impuritas pada pohon keputusan yang dibangun selama proses pelatihan. Keunggulan pendekatan ini adalah kemampuannya dalam menangani data berdimensi tinggi serta mempertimbangkan interaksi antar fitur dalam proses pemilihan fitur yang relevan.

Selain metode seleksi fitur, pemilihan algoritma klasifikasi juga menjadi faktor penting dalam meningkatkan performa deteksi malware. Dalam beberapa tahun terakhir, algoritma *tree-based boosting* modern seperti Extreme Gradient Boosting (XGBoost), Light Gradient Boosting Machine (LightGBM), dan CatBoost semakin banyak digunakan karena kemampuannya dalam menangani data kompleks serta meningkatkan akurasi prediksi. XGBoost dikenal memiliki mekanisme regularisasi yang kuat untuk menghindari *overfitting*, LightGBM menawarkan efisiensi pelatihan melalui pendekatan *histogram-based learning*, sedangkan CatBoost dirancang untuk meningkatkan stabilitas model serta mengurangi bias prediksi pada berbagai jenis data [6], [7], [8].

Selain pemilihan algoritma, aspek penting lain dalam machine learning adalah optimasi hyperparameter. Hyperparameter tuning berperan dalam menentukan konfigurasi terbaik suatu model agar mampu memaksimalkan performa klasifikasi. Beberapa penelitian menunjukkan bahwa model yang dioptimasi melalui hyperparameter tuning cenderung menghasilkan kinerja yang lebih baik dibandingkan model dengan parameter default [2], [3]. Meskipun demikian, proses optimasi ini juga meningkatkan waktu komputasi dan kompleksitas pelatihan sehingga perlu dianalisis keseimbangan antara peningkatan performa dan efisiensi waktu pelatihan model.

Drebin 215 Dataset hingga saat ini masih menjadi salah satu benchmark utama dalam penelitian deteksi malware Android karena menyediakan fitur statis yang komprehensif dan telah digunakan secara luas oleh komunitas [1], [2], [3], [5]. Meskipun demikian, berdasarkan tinjauan terhadap beberapa penelitian terkait, masih terdapat beberapa keterbatasan dalam penelitian sebelumnya. Pertama, sebagian penelitian hanya menggunakan satu algoritma klasifikasi tanpa melakukan perbandingan komprehensif antara beberapa algoritma boosting modern. Kedua, masih terbatas penelitian yang mengkaji pengaruh seleksi fitur berbasis Random Forest terhadap kinerja algoritma boosting pada dataset malware Android. Ketiga, analisis mengenai dampak hyperparameter tuning terhadap performa model dan waktu komputasi juga masih relatif terbatas.

Meskipun berbagai penelitian telah dilakukan dalam bidang deteksi malware Android berbasis machine learning, masih terdapat beberapa keterbatasan dalam penelitian sebelumnya. Berdasarkan penelitian Bintoro pada tahun 2024, mengusulkan metode deteksi malware Android menggunakan Neural Networks yang dikombinasikan dengan teknik feature selection [1]. Hasil penelitian menunjukkan peningkatan performa klasifikasi, namun pendekatan tersebut berfokus pada model deep learning sehingga belum mengevaluasi kinerja algoritma boosting modern yang dikenal memiliki efisiensi komputasi dan akurasi yang tinggi pada data berdimensi besar. Selanjutnya, pada penelitian Latifah pada tahun 2024, dilakukan analisis komparatif beberapa metode feature selection dengan algoritma XGBoost pada dataset Drebin [2]. Meskipun demikian, penelitian tersebut hanya menggunakan satu algoritma klasifikasi sehingga belum melakukan komparasi dengan algoritma boosting populer lainnya seperti LightGBM dan CatBoost dalam satu kerangka eksperimen yang sama. Berdasarkan penelitian yang dilakukan oleh Zaidi pada tahun 2025, mengombinasikan XGBoost dengan Convolutional Neural Networks untuk meningkatkan performa deteksi malware Android [3], tetapi pendekatan tersebut tidak mengevaluasi perbandingan performa beberapa algoritma boosting secara langsung. Selain itu, penelitian yang dilakukan Zhou pada tahun 2024

Building of Informatics, Technology and Science (BITS)
Volume 7, No 4, March 2026 Page: 2507−2517
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i4.9443

menggunakan LightGBM dengan pendekatan optimasi algoritma untuk deteksi malware Android [8], namun penelitian tersebut hanya berfokus pada satu model klasifikasi sehingga belum menganalisis perbandingan performa dengan algoritma boosting lain seperti XGBoost dan CatBoost.

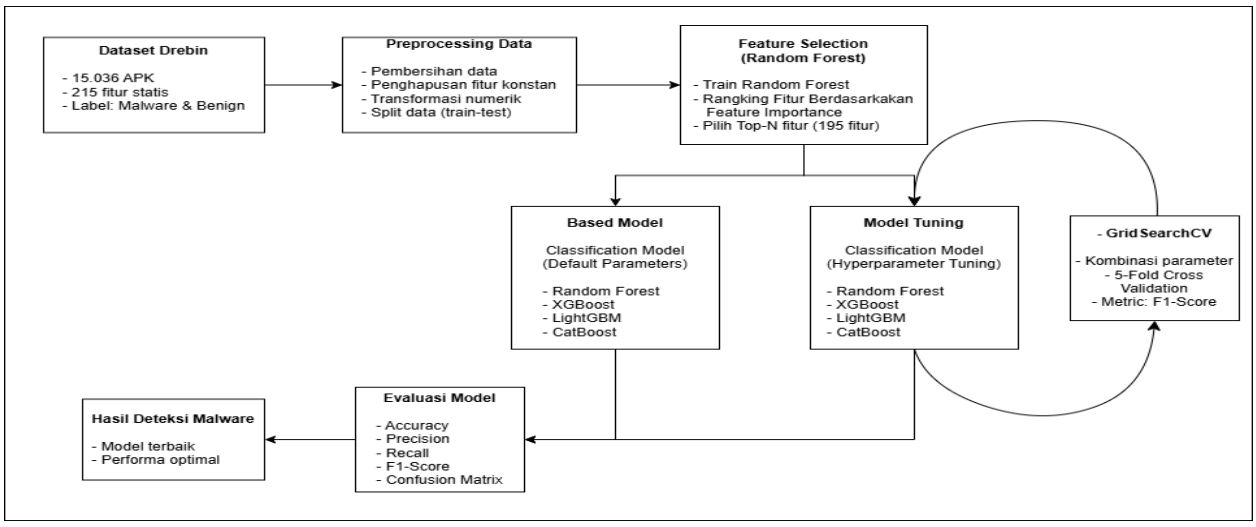
Berdasarkan celah penelitian tersebut, penelitian ini mengusulkan pendekatan deteksi malware Android berbasis machine learning dengan mengombinasikan seleksi fitur Random Forest dan beberapa algoritma klasifikasi berbasis boosting. Algoritma yang digunakan dalam penelitian ini meliputi Random Forest, XGBoost, LightGBM, dan CatBoost. Selain itu, penelitian ini juga menganalisis pengaruh hyperparameter tuning terhadap performa model dengan membandingkan hasil klasifikasi menggunakan parameter default dan parameter yang telah dioptimasi. Melalui pendekatan ini, penelitian ini memberikan kontribusi berupa evaluasi komparatif beberapa algoritma boosting modern (XGBoost, LightGBM, dan CatBoost) dalam deteksi malware Android, penerapan seleksi fitur berbasis Random Forest untuk mengurangi dimensi fitur dan meningkatkan efektivitas klasifikasi, serta analisis pengaruh hyperparameter tuning terhadap performa model dan efisiensi waktu komputasi sehingga dapat memberikan pemahaman yang lebih komprehensif mengenai keseimbangan antara akurasi deteksi dan efisiensi proses pelatihan model.

2. METODOLOGI PENELITIAN

2.1 Tahapan Penelitian

Tahapan penelitian ini bertujuan untuk mengevaluasi pengaruh optimasi hyperparameter terhadap kinerja model deteksi malware Android dibandingkan dengan konfigurasi standar (non-tuning). Penelitian menggunakan dataset Drebin yang telah banyak digunakan sebagai benchmark dalam deteksi malware Android. Dataset tersebut terlebih dahulu melalui tahap pra-pemrosesan untuk memastikan data siap digunakan dalam proses pemodelan. Selanjutnya, dilakukan seleksi fitur menggunakan metode feature importance berbasis Random Forest untuk mengurangi dimensi data dan menghilangkan fitur redundan sehingga dapat meningkatkan efisiensi komputasi dan stabilitas model. Subset fitur hasil seleksi kemudian digunakan secara konsisten pada seluruh eksperimen agar perbandingan antar model tetap adil.

Pada tahap pemodelan, empat algoritma klasifikasi digunakan, yaitu Random Forest, XGBoost, LightGBM, dan CatBoost. Setiap algoritma dievaluasi dalam dua skenario, yaitu menggunakan parameter bawaan (non-tuning) dan dengan optimasi hyperparameter (tuning). Hasil dari kedua skenario kemudian dibandingkan untuk menganalisis perbedaan kinerja model dari sisi akurasi klasifikasi dan waktu komputasi menggunakan metrik evaluasi seperti akurasi, presisi, recall, dan F1-score. Alur keseluruhan tahapan penelitian ditunjukkan pada Gambar 1 yang menggambarkan proses perbandingan model berbasis parameter standar dan model hasil tuning setelah tahap seleksi fitur.



Gambar 1. Alur tahapan penelitian

Gambar 1 menunjukkan alur tahapan penelitian yang dilakukan dalam proses deteksi malware Android menggunakan pendekatan machine learning. Penelitian diawali dengan penggunaan dataset Drebin 215 yang terdiri dari 15.036 file APK dengan 215 fitur statis serta label klasifikasi malware dan benign. Dataset tersebut kemudian melalui tahap preprocessing data yang meliputi pembersihan data, penghapusan fitur konstan, transformasi data ke bentuk numerik, serta pembagian data menjadi data latih dan data uji. Selanjutnya dilakukan proses seleksi fitur menggunakan Random Forest dengan memanfaatkan nilai *feature importance*. Pada tahap ini, model Random Forest dilatih untuk menghitung kontribusi setiap fitur terhadap proses klasifikasi, kemudian fitur-fitur tersebut diranking

Building of Informatics, Technology and Science (BITS)
Volume 7, No 4, March 2026 Page: 2507−2517
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i4.9443

berdasarkan tingkat kepentingannya. Berdasarkan proses ranking tersebut dipilih 195 fitur teratas yang dianggap paling relevan untuk digunakan pada tahap pemodelan.

Tahap berikutnya adalah pembangunan model klasifikasi yang dilakukan melalui dua skenario eksperimen, yaitu model dasar (based model) dan model dengan optimasi hyperparameter (model tuning). Pada skenario based model, algoritma klasifikasi yang digunakan meliputi Random Forest, XGBoost, LightGBM, dan CatBoost dengan menggunakan parameter default. Sementara itu, pada skenario model tuning dilakukan optimasi parameter menggunakan metode GridSearchCV dengan kombinasi parameter tertentu dan validasi silang sebanyak 5-fold cross validation dengan metrik evaluasi F1-score. Seluruh model yang dihasilkan kemudian dievaluasi menggunakan beberapa metrik evaluasi klasifikasi, yaitu accuracy, precision, recall, F1-score, serta analisis confusion matrix. Tahap akhir penelitian adalah melakukan analisis terhadap hasil evaluasi untuk menentukan model dengan performa terbaik dalam mendeteksi malware Android berdasarkan kinerja klasifikasi dan efisiensi komputasi.

2.2 Dataset dan Pra-pemrosesan Data

Dataset Drebin 215 digunakan sebagai sumber data utama dalam penelitian ini dan diunduh melalui platform Kaggle pada tautan https://www.kaggle.com/datasets/shashwatwork/android-malware-dataset-for-machine-learning/data?select=drebin-215-dataset-5560malware-9476-benign.csv. Dataset tersebut diunggah oleh pengguna perorangan, namun berasal dari repositori penelitian yang dipublikasikan oleh Suleiman Yerima pada platform figshare dengan DOI: https://doi.org/10.6084/m9.figshare.5854653.v1. Dataset ini selanjutnya digunakan dalam penelitian berjudul DroidFusion: A Novel Multilevel Classifier Fusion Approach for Android Malware Detection yang dipublikasikan pada IEEE Transactions on Cybernetics [9]. Dataset terdiri dari 15.036 entri aplikasi Android, dengan 5.560 sampel malware yang sebagian berasal dari proyek Drebin dan 9.476 sampel benign. Proyek Drebin sendiri diperkenalkan oleh Daniel Arp dkk. dalam studi Drebin: Effective and Explainable Detection of Android Malware in Your Pocket yang dipresentasikan pada Network and Distributed System Security Symposium (NDSS) tahun 2014 [10]. Dataset ini memiliki 215 fitur statis hasil ekstraksi melalui analisis kode statis aplikasi Android, tidak termasuk kolom label yang mengklasifikasikan setiap entri sebagai malware (‘S’) atau benign (‘B’).

Dalam proses pemodelan, dataset kemudian melalui tahap pra-pemrosesan data yang meliputi pembersihan data, penghapusan fitur konstan, serta transformasi data ke dalam format numerik yang dapat diproses oleh algoritma machine learning. Selanjutnya, dataset dibagi menjadi data latih dan data uji menggunakan rasio 80:20, di mana 80% data digunakan sebagai data pelatihan (training set) dan 20% data digunakan sebagai data pengujian (testing set) untuk mengevaluasi kinerja model secara objektif.

Ringkasan komposisi dan karakteristik dataset yang digunakan dalam penelitian ini disajikan pada Tabel 1.

Tabel 1. Detail Dataset

Nama Dataset	Banyaknya File	Total Sample	Jumlah kelas	Banyaknya Fitur	Missing Value
Drebin 215 Dataset	1	15,036	2	215	0

Berdasarkan Tabel 1, dataset Drebin 215 terdiri dari 15.036 sampel aplikasi Android yang terbagi ke dalam dua kelas, yaitu malware dan benign, serta memiliki 215 fitur statis tanpa nilai missing value. Sebelum proses pemodelan, dataset melalui tahap pra-pemrosesan data yang meliputi pembersihan data, penghapusan fitur konstan, serta transformasi atribut ke dalam format numerik. Selanjutnya dataset dibagi menjadi data latih dan data uji dengan rasio 80:20, di mana 80% data digunakan untuk pelatihan model dan 20% data digunakan untuk pengujian guna mengevaluasi kinerja model secara objektif. Tahapan ini bertujuan untuk memastikan kualitas data serta meminimalkan bias pada proses pelatihan model machine learning.

2.3 Seleksi Fitur Menggunakan Random Forest

Seleksi fitur pada penelitian ini dilakukan menggunakan algoritma Random Forest karena kemampuannya dalam mengukur kontribusi setiap fitur terhadap proses klasifikasi melalui mekanisme feature importance berbasis penurunan Gini Impurity. Random Forest dipilih karena mampu menangani data berdimensi tinggi, tahan terhadap noise, serta menghasilkan estimasi kepentingan fitur yang stabil. Gini Impurity pada suatu node dirumuskan pada persamaan 1.

$$Gini = 1 - \sum_{i=1}^{c} {p_i}^2 \quad (1)$$

di mana p_i menyatakan probabilitas kelas ke-i dan C adalah jumlah kelas. Nilai feature importance untuk fitur ke-j dihitung sebagai rata-rata penurunan Gini Impurity pada seluruh pohon keputusan dalam Random Forest 2 :

$$FI_j = \frac{1}{T}\sum_{t=1}^{T} \Delta Gini_{j,t} \quad (2)$$

dengan T sebagai jumlah pohon dan $\Delta Gini_{j,t}$ sebagai penurunan Gini akibat fitur ke-j pada pohon ke-t. Berdasarkan nilai *feature importance* yang diperoleh, seluruh fitur kemudian diranking dari nilai kepentingan tertinggi hingga terendah. Pada penelitian ini dipilih 195 fitur dengan nilai importance tertinggi sebagai subset fitur optimal yang digunakan pada tahap pemodelan klasifikasi. Pemilihan jumlah fitur ini dilakukan dengan mempertimbangkan

Building of Informatics, Technology and Science (BITS)
Volume 7, No 4, March 2026 Page: 2507−2517
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i4.9443

keseimbangan antara reduksi dimensi data dan kinerja model klasifikasi, sehingga fitur yang kurang relevan dapat dieliminasi tanpa menurunkan performa model secara signifikan.

2.3.1 Implementasi dan Alasan Pemilihan Parameter

Model Random Forest dilatih menggunakan parameter n_estimators = 100, random_state = 42, dan n_jobs = −1. Parameter tersebut digunakan untuk menjaga stabilitas model, memastikan reproduksibilitas eksperimen, serta meningkatkan efisiensi proses pelatihan dengan memanfaatkan seluruh core prosesor yang tersedia.
Seleksi fitur dilakukan dengan mengurutkan seluruh fitur berdasarkan nilai feature importance yang dihasilkan oleh Random Forest. Fitur kemudian dipilih secara bertahap mulai dari jumlah kecil hingga seluruh fitur digunakan, dan setiap subset fitur dievaluasi untuk melihat pengaruh jumlah fitur terhadap performa klasifikasi

Evaluasi performa seleksi fitur dilakukan menggunakan metrik accuracy dan F1-score untuk menilai kemampuan model dalam membedakan aplikasi malware dan aplikasi benign. Pendekatan ini digunakan untuk memperoleh subset fitur yang optimal sebelum diterapkan pada tahap klasifikasi menggunakan algoritma Random Forest, XGBoost, LightGBM, dan CatBoost *[1], [11]*

2.4 Metode Klasifikasi

2.4.1. Extreme Gradient Boosting (XGBoost)

Extreme Gradient Boosting (XGBoost) merupakan algoritma gradient boosting berbasis pohon keputusan yang membangun model secara bertahap dengan menggabungkan sejumlah fungsi pohon (weak learners). Prediksi akhir XGBoost dinyatakan sebagai penjumlahan keluaran seluruh pohon keputusan, sebagaimana ditunjukkan pada Persamaan (3):

$$\hat{y}i = \sum_{K=1}^{K} f_k\,(x_i), f_k \in \mathcal{r} \tag{3}$$

di mana $\hat{y}_i$merupakan nilai prediksi untuk data ke-i, $f_k(x_i)$ adalah fungsi pohon keputusan ke-k, dan K menyatakan jumlah pohon yang digunakan dalam model. Proses pembelajaran XGBoost dilakukan dengan meminimalkan fungsi objektif yang terdiri dari fungsi kerugian $l(y_i, \hat{y}_i)$ dan fungsi regularisasi $\Omega(f_k)$, sehingga model tidak hanya berfokus pada akurasi tetapi juga mengontrol kompleksitas untuk menghindari overfitting. Mekanisme regularisasi ini menjadi salah satu keunggulan utama XGBoost dalam menangani data berdimensi tinggi seperti fitur statis malware Android [3], [12].

2.4.2. Light Gradient Boosting Machine (LightGBM)

Light Gradient Boosting Machine (LightGBM) merupakan algoritma gradient boosting yang dirancang untuk meningkatkan kecepatan pelatihan dan efisiensi memori melalui pendekatan histogram-based learning serta strategi leaf-wise tree growth. Berbeda dengan metode level-wise, LightGBM memilih node dengan peningkatan loss terbesar untuk dikembangkan terlebih dahulu. Fungsi objektif LightGBM pada iterasi ke-t dirumuskan pada Persamaan (4):

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(\,y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) \tag{4}$$

di mana $f_t(x_i)$ merupakan fungsi pohon keputusan pada iterasi ke-t, $l(\cdot)$ adalah fungsi kerugian, dan $\Omega(f_t)$ merupakan fungsi regularisasi untuk mengontrol kompleksitas model. Pendekatan ini memungkinkan LightGBM mencapai performa tinggi dengan waktu komputasi yang relatif lebih singkat, sehingga sangat sesuai untuk dataset berskala besar dan berdimensi tinggi seperti dataset Drebin [13], [14]

2.4.3. CatBoost

CatBoost adalah algoritma gradient boosting berbasis pohon keputusan yang mengadopsi teknik ordered boosting untuk mengurangi bias prediksi dan meningkatkan stabilitas model selama proses pelatihan. Model CatBoost membangun prediksi secara aditif, sebagaimana ditunjukkan pada Persamaan (5) :

$$F(x) = F_0(x) + \sum_{m=1}^{M} \gamma_m\, h_m(x) \tag{5}$$

di mana $F_0(x)$ merupakan prediksi awal, $h_m(x)$ adalah fungsi pohon keputusan ke-m, γ_m menyatakan bobot pembelajaran, dan M adalah jumlah iterasi boosting. Teknik ordered boosting memastikan bahwa pada setiap iterasi, model tidak menggunakan informasi dari data yang sama untuk memperbarui prediksi, sehingga mampu mengurangi bias dan overfitting. Karakteristik ini membuat CatBoost menghasilkan performa yang stabil meskipun dengan penyesuaian hyperparameter yang minimal [15].

2.5 Evaluasi Kinerja Model

Evaluasi kinerja model dilakukan menggunakan metrik klasifikasi standar untuk tugas deteksi biner. Metrik-metrik ini diambil dari elemen *confusion matrix* yang mencakup *True Positive (TP)*, *True Negative (TN)*, *False Positive (FP)*,

Building of Informatics, Technology and Science (BITS)
Volume 7, No 4, March 2026 Page: 2507−2517
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i4.9443

dan *False Negative (FN)*. Akurasi menggambarkan proporsi prediksi yang benar terhadap total prediksi, sehingga memberikan gambaran kinerja keseluruhan model [16]:

$$Accuracy = \frac{Tp+TN}{Tp+TN+FP+FN} \quad (6)$$

Presisi mengukur ketepatan prediksi positif, yaitu seberapa banyak prediksi positif benar dibandingkan semua prediksi positif. Presisi penting untuk mengukur *false alarm* yang dihasilkan model [17]:

$$Precision = \frac{Tp}{Tp+FP} \quad (7)$$

Recall (*sensitivity* atau *true positive rate*) menunjukkan kemampuan model untuk menangkap kasus positif sebenarnya [18]:

$$Recall = \frac{Tp}{Tp+FN} \quad (8)$$

F1-score merupakan *harmonic mean* antara presisi dan recall, memberikan ukuran keseimbangan antara keduanya terutama ketika distribusi kelas tidak seimbang [19]:

$$F1-score = 2 \times \frac{Precision \times Recall}{Precision+Recall} \quad (9)$$

Penggunaan metrik-metrik ini sudah menjadi standar dalam penelitian klasifikasi biner berbasis *machine learning*, termasuk deteksi malware, karena memberikan gambaran performa model dari berbagai aspek yang berbeda [20]. Metodologi penelitian ini mengintegrasikan seleksi fitur berbasis Random Forest dengan algoritma *tree-based boosting* modern dan pendekatan *ensemble learning*. Rangkaian tahapan yang sistematis diharapkan mampu menghasilkan sistem deteksi malware Android yang akurat, efisien, dan dapat diimplementasikan secara praktis.

2.6 Grid Search

Pada tahap pengujian lanjutan, penelitian ini menerapkan Grid Search sebagai metode optimasi hyperparameter pada seluruh model klasifikasi, yaitu Random Forest, XGBoost, LightGBM, dan CatBoost. Grid Search bekerja dengan mengevaluasi seluruh kombinasi hyperparameter yang telah ditentukan sebelumnya dalam suatu ruang pencarian (*search space*) dan memilih konfigurasi terbaik berdasarkan metrik evaluasi tertentu. Dalam penelitian ini, metrik yang digunakan sebagai kriteria pemilihan model terbaik adalah F1-score, karena metrik ini mampu merepresentasikan keseimbangan antara *precision* dan *recall*, yang sangat penting pada permasalahan deteksi malware Android yang memiliki risiko tinggi terhadap kesalahan klasifikasi, khususnya *false negative*. Secara matematis, proses Grid Search dapat diformulasikan sebagai pencarian parameter optimal θ_{opt} yang memaksimalkan fungsi evaluasi model, sebagaimana ditunjukkan pada Persamaan (10):

$$\theta_{opt} = \underset{\theta \in \Omega}{argmax} E(\theta) \quad (10)$$

di mana θ_{opt} adalah kombinasi hyperparameter optimal, Ω merupakan ruang pencarian hyperparameter, dan $E(\theta)$ adalah fungsi evaluasi model. Dalam penelitian ini, fungsi evaluasi yang digunakan adalah F1-score, karena metrik tersebut mampu merepresentasikan keseimbangan antara *precision* dan *recall* pada permasalahan deteksi malware yang memiliki risiko tinggi terhadap kesalahan klasifikasi. Setiap kombinasi hyperparameter dievaluasi menggunakan teknik cross-validation, sehingga nilai $E(\theta)$yang diperoleh mencerminkan performa rata-rata model pada beberapa subset data latih. Pendekatan ini bertujuan untuk meningkatkan kemampuan generalisasi model serta mengurangi risiko overfitting akibat pemilihan parameter yang terlalu spesifik terhadap satu pembagian data. Hyperparameter yang dioptimasi mencakup jumlah estimator dan kedalaman pohon pada Random Forest dan XGBoost, laju pembelajaran serta jumlah estimator pada LightGBM, serta jumlah iterasi dan kedalaman pohon pada CatBoost.

Tabel 2. Ruang Pencarian Hyperparameter Grid Search

Model	Hyperparameter yang Dioptimasi	Search Space	Nilai
XGBoost	n_estimators	200, 400	400
	Learning_rate	0.03, 0.05, 0.1	0.1
	max_depth	4,6, 8	8
	Subsample	0.8, 1.0	0.8
	colsample_bytree	0.8, 1.0	0.8
LightGBM	n_estimators	200, 400, 600	400
	learning_rate	0.01, 0.05, 0.1	0.1
	max_depth	-1, 10, 20	-1
	num_leaves	31, 63, 127	31
	iterations	100, 200	200
CatBoost	depth	6, 8	8
	learning_rate	0.05, 0.1	0.1

Building of Informatics, Technology and Science (BITS)
Volume 7, No 4, March 2026 Page: 2507−2517
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i4.9443

Berdasarkan hasil proses Grid Search dengan 5-fold cross validation, diperoleh kombinasi hyperparameter terbaik untuk masing-masing model sebagaimana disajikan pada Tabel 2.5. Pada model XGBoost menghasilkan kombinasi terbaik pada n_estimators = 400, learning_rate = 0.1, max_depth = 8, subsample = 0.8, dan colsample_bytree = 0.8. Model LightGBM, konfigurasi optimal diperoleh pada parameter n_estimators = 400, learning_rate = 0.1, max_depth = −1, dan num_leaves = 31. Sementara itu, model CatBoost menunjukkan konfigurasi optimal pada iterations = 200, depth = 8, dan learning_rate = 0.1. Hasil tersebut menunjukkan bahwa eksplorasi ruang pencarian hyperparameter yang disajikan pada Tabel 2 mampu mengidentifikasi konfigurasi parameter yang paling sesuai untuk masing-masing algoritma.

3. HASIL DAN PEMBAHASAN

Bab ini menyajikan hasil penelitian dan pembahasan terkait penerapan metode deteksi malware Android berbasis machine learning dengan integrasi seleksi fitur menggunakan Random Forest dan algoritma klasifikasi tree-based modern. Pembahasan dimulai dari analisis karakteristik data hasil pra-pemrosesan, dilanjutkan dengan hasil seleksi fitur, evaluasi performa model klasifikasi pada skenario non-tuning dan tuning hyperparameter, serta perbandingan hasil dengan penelitian sebelumnya. Penyajian hasil didukung dengan tabel dan ilustrasi untuk memperjelas analisis yang dilakukan.

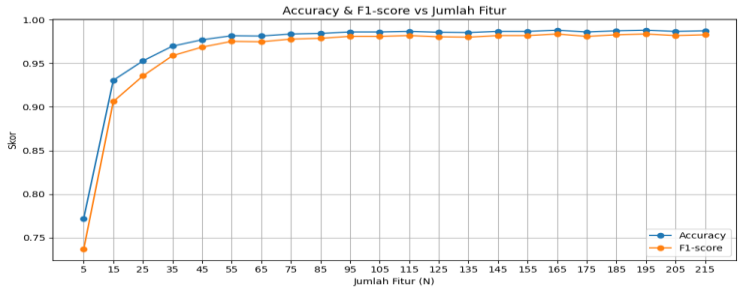
3.1 Hasil Pra-pemrosesan dan Seleksi Fitur

3.1.1 Hasil Pra-pemrosesan Data

Tahap awal eksperimen dilakukan melalui pra-pemrosesan dataset Drebin 215 untuk memastikan kualitas data sebelum proses pemodelan. Dataset memiliki dimensi fitur yang tinggi sehingga berpotensi menimbulkan redundansi dan meningkatkan kompleksitas komputasi. Oleh karena itu, dilakukan beberapa tahapan pra-pemrosesan agar data sesuai dengan kebutuhan algoritma machine learning. Hasil pra-pemrosesan menunjukkan bahwa dataset tidak memiliki missing value, sehingga tidak diperlukan imputasi data. Fitur dengan nilai konstan dihapus karena tidak berkontribusi pada proses klasifikasi, kemudian dilakukan normalisasi untuk menyelaraskan skala fitur numerik. Dataset selanjutnya dibagi menjadi data latih dan data uji menggunakan stratified sampling untuk menjaga proporsi kelas malware dan benign tetap seimbang. Ringkasan hasil pra-pemrosesan ditunjukkan pada Tabel 3.

3.1.2 Hasil Seleksi Fitur Menggunakan Random Forest

Seleksi fitur pada penelitian ini dilakukan menggunakan algoritma Random Forest dengan pendekatan feature importance yang dihitung berdasarkan penurunan nilai Gini Impurity. Pendekatan ini dipilih karena Random Forest mampu mengukur tingkat kontribusi setiap fitur terhadap proses pengambilan keputusan pada model klasifikasi secara efektif. Metode ini digunakan untuk mengidentifikasi fitur-fitur yang paling berkontribusi dalam membedakan aplikasi malware dan benign pada dataset Android. Proses seleksi fitur dimulai dengan melatih model Random Forest menggunakan seluruh fitur hasil tahap pra-pemrosesan data. Setelah proses pelatihan selesai, model akan menghitung nilai feature importance untuk setiap fitur berdasarkan seberapa besar fitur tersebut berperan dalam meningkatkan kualitas pemisahan kelas pada pohon keputusan yang terbentuk. Nilai tersebut kemudian diurutkan dari yang paling tinggi hingga yang paling rendah untuk menentukan tingkat kepentingan masing-masing fitur. Selanjutnya, sejumlah fitur terbaik (top-N features) dipilih sebagai masukan pada tahap klasifikasi menggunakan algoritma machine learning yang digunakan dalam penelitian ini. Proses ini bertujuan untuk mengurangi dimensi data, menghilangkan fitur yang kurang relevan, serta meningkatkan efisiensi proses pelatihan model. Hasil analisis menunjukkan bahwa hanya sebagian kecil fitur yang memiliki kontribusi dominan terhadap proses klasifikasi, sedangkan sebagian besar fitur lainnya memberikan pengaruh yang relatif kecil terhadap performa model. Distribusi nilai feature importance tersebut ditunjukkan pada Gambar 2.



Gambar 2. Distribusi Feature Importance Random Forest

Berdasarkan Gambar 2, terlihat bahwa peningkatan jumlah fitur yang digunakan pada proses klasifikasi menghasilkan peningkatan nilai *accuracy* dan *F1-score* secara signifikan pada jumlah fitur yang kecil. Namun setelah jumlah fitur mencapai kisaran tertentu, peningkatan performa mulai cenderung stabil dan tidak mengalami perubahan yang signifikan. Hal ini menunjukkan bahwa sebagian besar informasi penting untuk membedakan aplikasi malware

Building of Informatics, Technology and Science (BITS)
Volume 7, No 4, March 2026 Page: 2507−2517
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i4.9443

dan benign telah terwakili oleh sejumlah fitur utama. Berdasarkan hasil pengujian, performa model mencapai kondisi optimal pada 195 fitur, dengan nilai accuracy sebesar 0.9880 dan F1-score sebesar 0.9837. Setelah titik tersebut, penambahan jumlah fitur hingga seluruh fitur yang tersedia tidak memberikan peningkatan performa yang berarti. Oleh karena itu, 195 fitur dipilih sebagai subset fitur terbaik karena mampu memberikan keseimbangan antara kompleksitas model dan kinerja klasifikasi.

Tabel 3. Hasil Pra-pemrosesan Dataset Drebin

Komponen	Jumlah Data	Jumlah Fitur awal	Fitur yang terpilih
Data Latih	12028	215	195
Data Uji	3008	215	195

Hasil proses pra-pemrosesan dataset Drebin disajikan pada Tabel 3. Dataset dibagi menjadi data latih sebanyak 12.028 sampel dan data uji sebanyak 3.008 sampel dengan total fitur awal sebanyak 215 fitur. Setelah dilakukan proses seleksi fitur menggunakan metode Random Forest, diperoleh subset fitur optimal sebanyak 195 fitur yang digunakan pada tahap pemodelan. Pemilihan jumlah fitur tersebut didasarkan pada evaluasi performa model terhadap variasi jumlah fitur, di mana pada 195 fitur diperoleh nilai accuracy sebesar 0.9880 dan F1-score sebesar 0.9837. Selain itu, penambahan jumlah fitur setelah titik tersebut tidak memberikan peningkatan performa yang signifikan, sehingga 195 fitur dipilih sebagai subset fitur optimal karena mampu memberikan keseimbangan antara kompleksitas model dan performa klasifikasi.

3.2 Implementasi dan Pengujian Model

3.2.1 Pengujian Based Model

Tahap pengujian awal dilakukan menggunakan konfigurasi dasar *based model* pada algoritma klasifikasi XGBoost, LightGBM, dan CatBoost. Tujuan dari pengujian ini adalah untuk memperoleh gambaran performa awal masing-masing model sebelum dilakukan optimasi hyperparameter. Seluruh model dilatih menggunakan fitur hasil seleksi berbasis Random Forest dan dievaluasi menggunakan data uji. Metrik evaluasi yang digunakan meliputi *accuracy*, *precision*, *recall*, *F1-score*, dan *ROC-AUC*. Hasil evaluasi model Based disajikan pada Tabel 4.

Tabel 4. Hasil Evaluasi Model Based

Model	Default parameter	Accuracy	Precision	Recall	F1-Score	Roc_Auc
XGBoost	n_estimators=100, 200, eval_metric="logloss", random_state=42	0.9874	0.9847	0.9811	0.9829	0.9987
LightGBM	n_estimators : 100, 150, 200 random_state : 42	0.9894	0.9874	0.9838	0.9856	0.9990
CatBoost	iterations= 300, verbose=0, random_state=42	0.987	0.9873	0.9775	0.9824	0.9989

Berdasarkan hasil pada Tabel 4, seluruh model non-tuning menunjukkan performa yang sangat tinggi dengan nilai *accuracy* di atas 0.98 dan *ROC-AUC* mendekati 1. XGBoost mencatat performa terbaik dengan *accuracy* sebesar 0.9874 dan *F1-score* sebesar 0.9829, diikuti oleh LightGBM dengan *accuracy* 0.9894 dan *F1-score* 0.9856. CatBoost menunjukkan performa yang sedikit lebih rendah dibandingkan dua model lainnya, khususnya pada metrik *recall*, namun tetap menghasilkan kinerja klasifikasi yang sangat baik. Nilai *ROC-AUC* yang tinggi pada seluruh model mengindikasikan kemampuan pemisahan kelas malware dan benign yang sangat kuat meskipun tanpa optimasi hyperparameter.

3.2.2 Pengujian Model dengan Tuning Hyperparameter

Tahap selanjutnya adalah pengujian model dengan optimasi hyperparameter menggunakan metode *Grid Search*. Proses tuning dilakukan untuk menyesuaikan parameter-parameter penting pada setiap algoritma guna memperoleh konfigurasi terbaik, dengan validasi silang untuk menghindari *overfitting*. Hasil evaluasi model setelah tuning hyperparameter dan Hasil pengujian model dengan tuning hyperparameter disajikan pada Tabel 5.

Tabel 5. Hasil Evaluasi Model dengan Tuning Hyperparameter

Model	Best Parameter	Accuracy	Precision	Recall	F1-Score	Roc_Auc
XGBoost	colsample_bytree: 0.8, learning_rate: 0.1, max_depth: 8, n_estimators: 400, subsample: 0.8	0.9867	0.9838	0.9802	0.9820	0.9989
LightGBM	learning_rate: 0.1, max_depth: -1, n_estimators: 400, num_leaves: 31	0.9900	0.9865	0.9865	0.9865	0.9990
CatBoost	depth: 8, iterations: 200, learning_rate: 0.1	0.9880	0.9900	0.9775	0.9837	0.9988

Building of Informatics, Technology and Science (BITS)
Volume 7, No 4, March 2026 Page: 2507−2517
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i4.9443

Hasil pada Tabel 5 menunjukkan bahwa hyperparameter tuning tidak selalu meningkatkan performa seluruh model. LightGBM menjadi model terbaik pada skenario tuning dengan accuracy 0.9900, F1-score 0.9865, dan ROC-AUC 0.9990, yang sedikit lebih baik dibandingkan skenario based. Sebaliknya, XGBoost mengalami sedikit penurunan performa dengan accuracy 0.9867 dan F1-score 0.9820, lebih rendah dibandingkan konfigurasi default, yang menunjukkan bahwa peningkatan kompleksitas model tidak selalu meningkatkan kemampuan generalisasi. Sementara itu, CatBoost menunjukkan performa yang relatif stabil dengan accuracy 0.9880 dan F1-score 0.9837, dengan perubahan yang bersifat marginal.

Secara keseluruhan, hasil ini menunjukkan bahwa dampak hyperparameter tuning terhadap peningkatan performa relatif terbatas, terutama ketika model telah menggunakan fitur hasil seleksi yang optimal. Dalam kondisi tersebut, konfigurasi default pada algoritma boosting sudah mampu menghasilkan performa yang sangat tinggi, sehingga peningkatan kompleksitas model perlu dipertimbangkan terhadap efisiensi komputasi dan stabilitas generalisasi.

3.2.3 Perbandingan Waktu Komputasi

Evaluasi waktu pelatihan dilakukan untuk membandingkan efisiensi komputasi pada skenario based model, tanpa seleksi fitur, dan dengan tuning hyperparameter. Ringkasan perbandingan waktu dan performa disajikan pada Tabel 6.

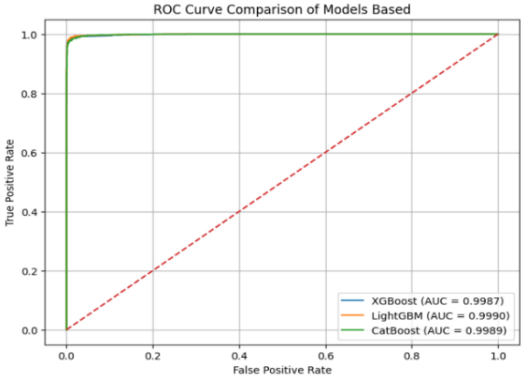
Tabel 6. Hasil Perbandingan Waktu Based dan Tuning

Model	Waktu based model (detik)	Best F1 based model	Waktu tanpa seleksi fitur	Best F1 tanpa seleksi fitur	Waktu Tuning model	Best F1 tuning model	Keterangan
XGBoost	1.08 detik	0.9829	1.83 detik	0.9842	583.14 detik	0.9842	Tuning meningkatkan jumlah estimator dan kedalaman pohon
LightGBM	1.06 detik	0.9856	8.97 detik	0.9805	604.89 detik	0.9859	Proses tuning menambah jumlah estimator dan kompleksitas leaf
CatBoost	3.73 detik	0.9824	4.24 detik	0.9846	202.25 detik	0.9822	Iterasi dan depth ditingkatkan saat tuning
Total	5.87 detik		15.04 detik		1390.28 detik		Akumulasi waktu yang dibutuhkan seluruh model

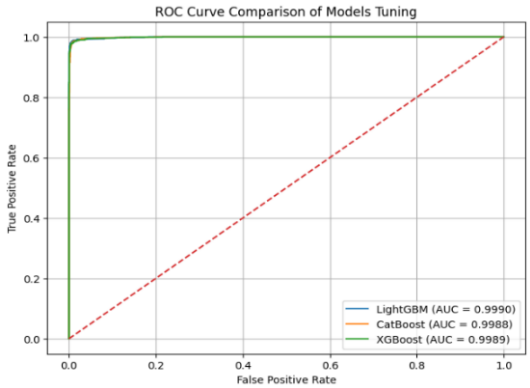
Tabel 6 menunjukkan bahwa seluruh model mengalami peningkatan waktu pelatihan yang sangat signifikan pada skenario tuning hyperparameter dibandingkan *based model*. Peningkatan waktu paling besar terjadi pada XGBoost, yang disebabkan oleh eksplorasi ruang parameter yang luas dan proses pelatihan berulang selama *Grid Search*. Temuan ini menegaskan adanya *trade-off* antara peningkatan performa dan efisiensi komputasi.

3.3 Pembahasan

Berdasarkan hasil evaluasi performa yang ditunjukkan pada Tabel 4 dan Tabel 5, seluruh model klasifikasi yang digunakan dalam penelitian ini menunjukkan kinerja yang sangat tinggi, dengan nilai accuracy, precision, recall, F1-score, dan ROC-AUC yang mendekati satu. Hal ini mengindikasikan bahwa kombinasi seleksi fitur berbasis Random Forest dengan algoritma klasifikasi tree-based modern mampu merepresentasikan karakteristik malware Android secara efektif, meskipun dataset yang digunakan memiliki dimensi fitur yang tinggi. Nilai ROC-AUC di atas 0.9990 pada seluruh model memperlihatkan kemampuan diskriminatif yang sangat kuat dalam memisahkan kelas malware dan benign, sehingga risiko kesalahan klasifikasi dapat ditekan secara signifikan.



Gambar 3.A. Kurva ROC perbandingan model pada skenario based

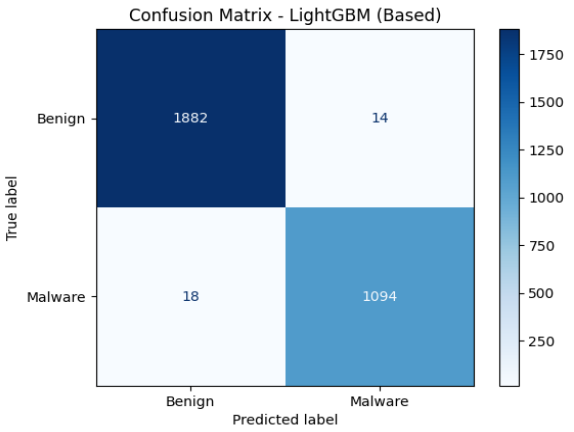


Gambar 3.B. Kurva ROC perbandingan model setelah tuning hyperparameter

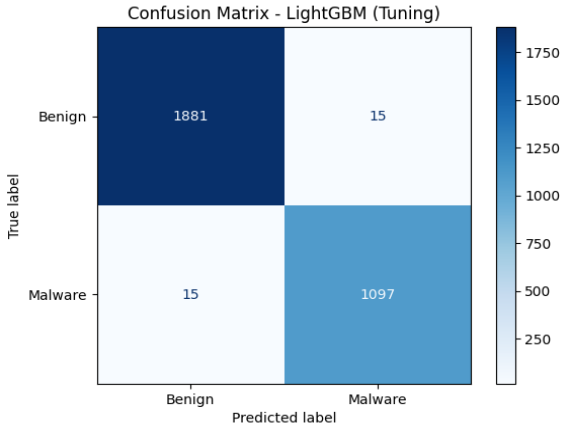
Building of Informatics, Technology and Science (BITS)
Volume 7, No 4, March 2026 Page: 2507−2517
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i4.9443

Kemampuan diskriminatif model ditunjukkan melalui kurva ROC pada Gambar 3.A. dan Gambar 3.B. Seluruh kurva mendekati sudut kiri atas grafik, yang menandakan nilai True Positive Rate (TPR) tinggi dan False Positive Rate (FPR) rendah. Nilai ROC-AUC berada pada rentang 0,9987–0,9990 untuk skenario based dan 0,9988–0,9990 untuk skenario tuning, menunjukkan bahwa ketiga model memiliki kemampuan pemisahan kelas yang hampir sempurna. Perbedaan antar model sangat kecil dan kurva ROC terlihat hampir identik.

Pada skenario tanpa tuning, LightGBM menjadi model terbaik dengan accuracy 0,9894 dan F1-score 0,9856, diikuti oleh XGBoost dan CatBoost. Setelah dilakukan tuning, LightGBM kembali menunjukkan performa tertinggi dengan accuracy 0,9900 dan F1-score 0,9865. Sementara itu, CatBoost dan XGBoost menunjukkan perubahan performa yang relatif kecil setelah tuning. Secara keseluruhan, seluruh algoritma boosting menghasilkan performa yang sangat tinggi dalam mendeteksi malware pada Drebin 215 dataset. Namun, LightGBM terbukti paling konsisten dan stabil pada kedua skenario, sehingga dapat dianggap sebagai model terbaik dalam penelitian ini.



Gambar 4.A. Confusion matrix model LightGBM pada skenario Based



Gambar 4.B. Confusion matrix model XGBoost setelah tuning

Distribusi kesalahan klasifikasi dapat diamati melalui confusion matrix pada Gambar 4.A (LightGBM based) dan Gambar 4.B (LightGBM tuning). Pada skenario based, dari 1.896 sampel benign, sebanyak 1.882 berhasil diklasifikasikan dengan benar (True Negative) dan 14 salah diklasifikasikan sebagai malware (False Positive). Pada kelas malware, model berhasil mendeteksi 1.094 sampel secara benar (True Positive) dengan 18 sampel tidak terdeteksi (False Negative). Setelah dilakukan hyperparameter tuning, terjadi sedikit perubahan pada distribusi kesalahan. Pada kelas benign, 1.881 sampel terklasifikasi benar dengan 15 False Positive, sedangkan pada kelas malware jumlah True Positive meningkat menjadi 1.097 dan False Negative menurun menjadi 15, yang menunjukkan adanya peningkatan sensitivitas model terhadap malware. Dalam konteks keamanan siber, kesalahan False Negative lebih kritis dibandingkan False Positive karena malware yang tidak terdeteksi dapat menyebabkan kerugian yang lebih besar. Pada kedua skenario, proporsi FN berada di bawah 2% dari total data malware, yang menunjukkan kemampuan deteksi yang sangat tinggi. Secara keseluruhan, perbedaan antara skenario based dan tuning bersifat marginal. Meskipun tuning sedikit meningkatkan kemampuan deteksi malware, peningkatan performa tersebut relatif kecil dibandingkan dengan kenaikan waktu pelatihan. Oleh karena itu, konfigurasi default yang dikombinasikan dengan seleksi fitur yang optimal sudah mampu memberikan performa deteksi malware yang sangat kompetitif dengan efisiensi komputasi yang lebih baik.

4. KESIMPULAN

Penelitian ini mengevaluasi efektivitas kombinasi seleksi fitur berbasis Random Forest dengan algoritma klasifikasi tree-based modern untuk mendeteksi malware Android menggunakan dataset Drebin 215. Hasil penelitian menunjukkan bahwa metode seleksi fitur Random Forest mampu mereduksi dimensi data dari 215 fitur menjadi 195 fitur tanpa menghilangkan informasi penting yang berkontribusi terhadap proses klasifikasi. Model yang dihasilkan menunjukkan performa yang sangat baik pada seluruh algoritma yang diuji, yaitu XGBoost, LightGBM, dan CatBoost. Berdasarkan hasil evaluasi, LightGBM memberikan performa terbaik, dengan nilai accuracy sebesar 0.9900, F1-score sebesar 0.9865, dan ROC-AUC sebesar 0.9990 setelah proses tuning hyperparameter. Namun demikian, analisis perbandingan antara skenario tanpa tuning dan dengan tuning menunjukkan bahwa peningkatan performa yang diperoleh relatif kecil dibandingkan dengan peningkatan waktu komputasi yang diperlukan. Temuan ini menunjukkan bahwa kualitas proses seleksi fitur memiliki peran yang sangat penting dalam meningkatkan performa model klasifikasi, bahkan ketika menggunakan konfigurasi parameter default. Oleh karena itu, kombinasi seleksi fitur yang optimal dan algoritma boosting berbasis pohon dapat menjadi pendekatan yang efektif dan efisien dalam sistem deteksi malware Android berbasis fitur statis. Meskipun demikian, penelitian ini masih memiliki keterbatasan karena hanya menggunakan fitur statis dan satu dataset benchmark, sehingga penelitian selanjutnya

Building of Informatics, Technology and Science (BITS)
Volume 7, No 4, March 2026 Page: 2507−2517
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i4.9443

disarankan untuk mengintegrasikan fitur dinamis, menggunakan dataset yang lebih beragam, serta mengeksplorasi teknik optimasi yang lebih adaptif untuk meningkatkan robustnes dan kemampuan generalisasi sistem deteksi malware.

REFERENCES

[1] J. Bintoro, F. A. Rafrastara, I. A. Latifah, W. Ghozi, and W. Yassin, "Optimizing Android Malware Detection Using Neural Networks And Feature Selection Method," *Jurnal Teknik Informatika (Jutif)*, vol. 5, no. 6, pp. 1663–1672, Dec. 2024, doi: 10.52436/1.jutif.2024.5.6.3898.

[2] I. A. Latifah, F. A. Rafrastara, J. Bintoro, W. Ghozi, and W. M. Osman, "Comparative Analysis of Feature Selection Methods with XGBoost for Malware Detection on the Drebin Dataset," *Jurnal Sisfokom (Sistem Informasi dan Komputer)*, vol. 13, no. 3, pp. 403–409, Nov. 2024, doi: 10.32736/sisfokom.v13i3.2294.

[3] A. R. Zaidi, T. Abbas, A. Daud, O. Alghushairy, H. Dawood, and N. Sarwar, "Enhancing Android Malware Detection with XGBoost and Convolutional Neural Networks," *Computers, Materials and Continua*, vol. 84, no. 2, pp. 3281–3304, 2025, doi: 10.32604/cmc.2025.063646.

[4] M. A. Haq and M. Khuthaylah, "Leveraging Machine Learning for Android Malware Analysis: Insights from Static and Dynamic Techniques," *Engineering, Technology and Applied Science Research*, vol. 14, no. 4, pp. 15027–15032, Aug. 2024, doi: 10.48084/etasr.7632.

[5] M. U. Rashid *et al.*, "Hybrid Android Malware Detection and Classification Using Deep Neural Networks," *International Journal of Computational Intelligence Systems*, vol. 18, no. 1, Dec. 2025, doi: 10.1007/s44196-025-00783-x.

[6] M. Aamir *et al.*, "AMDDLmodel: Android smartphones malware detection using deep learning model," *PLoS One*, vol. 19, no. 1 January, Jan. 2024, doi: 10.1371/journal.pone.0296722.

[7] A. Alhussen, "Advanced Android Malware Detection through Deep Learning Optimization," *Engineering, Technology and Applied Science Research*, vol. 14, no. 3, pp. 14552–14557, Jun. 2024, doi: 10.48084/etasr.7443.

[8] S. Zhou, H. Li, X. Fu, D. Han, and X. He, "Novel Multi-Classification Dynamic Detection Model for Android Malware Based on Improved Zebra Optimization Algorithm and LightGBM," *Sensors*, vol. 24, no. 18, Sep. 2024, doi: 10.3390/s24185975.

[9] R. S. Arslan, "JDroid: Android malware detection using hybrid opcode feature vector," *PeerJ Comput. Sci.*, vol. 11, 2025, doi: 10.7717/peerj-cs.3051.

[10] N. H. Saeed, A. A. Hamza, M. A. Sobh, and A. M. Bahaa-Eldin, "Efficient feature ranked hybrid framework for android Iot malware detection," *Sci. Rep.*, vol. 16, no. 1, Dec. 2026, doi: 10.1038/s41598-026-35238-6.

[11] R. Islam, M. I. Sayed, S. Saha, M. J. Hossain, and M. A. Masud, "Android malware classification using optimum feature selection and ensemble machine learning," *Internet of Things and Cyber-Physical Systems*, vol. 3, pp. 100–111, Jan. 2023, doi: 10.1016/j.iotcps.2023.03.001.

[12] S. Nazarinezhad, N. Khosrojerdi, and A. R. Shafieesabet, "Android Malware Detection by XGBoost Algorithm," *Journal of Artificial Intelligence, Applications, and Innovations*, vol. 1, no. 3, pp. 31–37, 2024, doi: 10.61838/jaiai.1.3.4.

[13] A. Guerra-Manzanares, "Machine Learning for Android Malware Detection: Mission Accomplished? A Comprehensive Review of Open Challenges and Future Perspectives," *Comput. Secur.*, vol. 138, Mar. 2024, doi: 10.1016/j.cose.2023.103654.

[14] J. M. Arif, M. F. A. Razak, S. Awang, S. R. T. Mat, N. S. N. Ismail, and A. Firdaus, "A static analysis approach for Android permission-based malware detection systems," *PLoS One*, vol. 16, no. 9 September, Sep. 2021, doi: 10.1371/journal.pone.0257968.

[15] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "CatBoost: unbiased boosting with categorical features," 2018. [Online]. Available: https://github.com/catboost/catboost

[16] A. Farhan AlShammari, "Implementation of Model Evaluation using Confusion Matrix in Python," *Int. J. Comput. Appl.*, vol. 186, no. 50, pp. 975–8887, 2024.

[17] O. Rainio, J. Teuho, and R. Klén, "Evaluation metrics and statistical tests for machine learning," *Sci. Rep.*, vol. 14, no. 1, Dec. 2024, doi: 10.1038/s41598-024-56706-x.

[18] S. Sathyanarayanan, "Confusion Matrix-Based Performance Evaluation Metrics," *African Journal of Biomedical Research*, pp. 4023–4031, Nov. 2024, doi: 10.53555/ajbr.v27i4s.4345.

[19] F. Mukhlif, I. Hashem, and N. Ithnin, "Performance Metrics of Different Machine Learning Models for Windows Malware Detection," *Journal of Advanced Industrial Technology and Application*, vol. 6, no. 2, Dec. 2025, doi: 10.30880/jaita.2025.06.02.004.

[20] K. M. Sujon, R. Hassan, K. Choi, and M. A. Samad, "Accuracy, precision, recall, f1-score, or MCC? empirical evidence from advanced statistics, ML, and XAI for evaluating business predictive models," *J. Big Data*, vol. 12, no. 1, Dec. 2025, doi: 10.1186/s40537-025-01313-4.