Building of Informatics, Technology and Science (BITS)
Volume 7, No 4, March 2026 Page: 2193−2204
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i4.8865

Deteksi Malware Android Berbasis Ensemble Soft Voting LightGBM, Logistic Regression dan CatBoost

Ardian Danendra, Elkaf Rahmawan Pramudya*

Fakultas Ilmu Komputer, Program Studi Teknik Informatika, Universitas Dian Nuswantoro, Semarang, Indonesia
Email: [1]111202214493@mhs.dinus.ac.id, [2,*]elkaf.rahmawan@dsn.dinus.ac.id
Email Penulis Korespondensi: elkaf.rahmawan@dsn.dinus.ac.id
Submitted: **06/12/2025**; Accepted: **05/03/2026**; Published**: 05/03/2026**

Abstrak−Sistem operasi Android menghadapi tantangan serius dengan evolusi malware yang semakin kompleks dan beragam. Penelitian ini mengusulkan sistem deteksi malware Android berbasis ensemble soft voting yang mengintegrasikan tiga algoritma (LightGBM, Logistic Regression, dan CatBoost) untuk meningkatkan akurasi deteksi sekaligus menjaga efisiensi komputasi. Dataset yang digunakan adalah CCCS-CIC-AndMal-2020 yang sangat imbalanced dengan lebih dari 400.000 sampel aplikasi Android. Model yang diusulkan memanfaatkan fitur hybrid yang menggabungkan fitur statis (seperti permissions, intents, dan API calls dari AndroidManifest) dengan fitur dinamis (seperti aktivitas memori, panggilan API saat runtime, logcat, dan trafik jaringan di lingkungan emulasi). Kombinasi kedua jenis fitur ini dipilih untuk menyeimbangkan biaya ekstraksi yang relatif rendah pada fitur statis dengan ketahanan yang lebih baik terhadap teknik obfuscation pada fitur dinamis. Metodologi meliputi preprocessing multi-tahap (IQR capping 40×, StandardScaler, RFE 150 fitur, SMOTE 30%) untuk meningkatkan kualitas data dan mengurangi dimensi sebesar 56% tanpa kehilangan informasi penting. Model ensemble dilatih dengan bobot berbasis F1-Macro (33.46% LightGBM, 30.99% Logistic Regression, 35.55% CatBoost) yang mendekati proporsi 1:1:1. Hasil evaluasi pada testing set menunjukkan performa sangat tinggi: Accuracy 95.58%, Balanced Accuracy 92.21%, F1-Macro 0.9208, True Positive Rate 100%, dan False Alarm Rate 0.00%. Kombinasi metrik tersebut mengindikasikan bahwa model mampu mendeteksi seluruh sampel malware tanpa false positive pada aplikasi benign, sehingga layak untuk deployment di production environment. Penelitian ini berkontribusi dengan demonstrating efektivitas ensemble soft voting yang efisien (hanya 3 model) untuk deteksi malware Android dengan metrik evaluasi multi-dimensi yang representatif untuk data imbalanced.

Kata Kunci: Deteksi Malware; Ensemble Learning; Soft Voting; SMOTE; Android

Abstract−The Android operating system faces serious challenges with increasingly complex and diverse malware evolution. This research proposes an Android malware detection system based on soft voting ensemble that integrates three algorithms (LightGBM, Logistic Regression, and CatBoost) to improve detection accuracy while maintaining computational efficiency. The dataset used is CCCS-CIC-AndMal-2020, which is highly imbalanced with over 400,000 Android application samples. The proposed model leverages hybrid features that combine static information (permissions, intents, API calls from the AndroidManifest) with dynamic behavior (memory activities, runtime API calls, logcat, and network traffic in an emulated environment), balancing low extraction cost with improved robustness against obfuscation. The methodology includes multi-stage preprocessing (IQR capping 40×, StandardScaler, RFE 150 features, SMOTE 30%) to improve data quality and reduce dimensionality by 56% without losing important information. The ensemble model is trained with F1-Macro-based weights (33.46% LightGBM, 30.99% Logistic Regression, 35.55% CatBoost) approximating 1:1:1 proportion. Evaluation results on the testing set demonstrate very high performance: Accuracy 95.58%, Balanced Accuracy 92.21%, F1-Macro 0.9208, True Positive Rate 100%, and False Alarm Rate 0.00%. The combination of these metrics indicates that the model can detect all malware samples without false positives on benign applications, making it suitable for production deployment. This research contributes by demonstrating the effectiveness of an efficient soft voting ensemble (only 3 models) for Android malware detection with multi-dimensional evaluation metrics representative of imbalanced data.

Keywords: Malware Detection; Ensemble Learning; Soft Voting; SMOTE; Android

1. PENDAHULUAN

Android sebagai sistem operasi smartphone paling dominan di dunia, dengan pangsa pasar tertinggi di dunia dengan *global marketshare* mencapai 71,8% pada akhir 2023, menghadapi tantangan besar berupa evolusi serangan malware yang semakin kompleks dan intensif. Karakteristik *open-source*, biaya rendah, dan kemudahan akses turut mempercepat perkembangan Android, tetapi juga meningkatkan risiko bagi penggunanya [1], [2]. Malware yang menyasar Android berkembang dalam jenis dan intensitas, mencakup *logic bombs*, *repackaging attacks*, banking trojans, botnet, ransomware, hingga pencurian kredensial melalui aplikasi atau dokumen palsu yang disebarkan via media sosial dan aplikasi pesan instan[3], [4]. Teknik baru seperti *obfuscation*, *polymorphic malware* dan *runtime injection* semakin sering digunakan untuk menghindari pendeteksian tradisional dan mempersulit proses mitigasi[3], [5].

Metode deteksi malware Android secara umum dapat dikategorikan menjadi tiga pendekatan, yaitu *signature based*, *anomaly based*, dan *behavior based* detection [6], [7]. Pendekatan *signature based* terbukti kurang mampu menghadapi varian malware baru yang memanfaatkan *obfuscation* dan *mutation* pada kode[5], [6], [7]. *Anomaly based detection* memberikan fleksibilitas mendeteksi pola serangan baru tapi sering menghasilkan false positive akibat dinamika data normal[6], [7]. Sedangkan *behavior based* membutuhkan analisis interaksi aplikasi yang rumit dan tetap rawan terhadap teknik penghindaran serta *zeroday attack[6], [7]*. Untuk mengatasi keterbatasan ini, riset terkini mengadopsi *machine learning* dan *ensemble learning* pada deteksi malware Android modern. *Ensemble learning* adalah teknik penggabungan dari beberapa algoritma prediksi menjadi satu untuk menghasilkan deteksi yang lebih

Building of Informatics, Technology and Science (BITS)
Volume 7, No 4, March 2026 Page: 2193−2204
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i4.8865

akurat dan stabil daripada model tunggal. Namun, keberhasilan pendekatan ini juga sangat dipengaruhi oleh kualitas dan karakteristik dataset yang digunakan dalam evaluasi[5], [8].

Sebagai *benchmark*, dataset CCCS-CIC-AndMal-2020 telah digunakan secara luas, berisi lebih dari 400.000 sampel aplikasi Android, terdiri atas 200.000 *benign* dan 200.000 malware dari puluhan kategori dan ratusan *family[5], [8]*. Dataset ini sangat *imbalanced* seperti jumlah *benign* jauh lebih besar sedangkan jumlah dari beberapa varian malware lebih sedikit dari pada yang lain seperti *Banker*, *File Injector, Scareware[5], [8]*. Oleh sebab itu, pada berbagai penelitian teknik *Synthetic Minority Over-sampling Technique* (SMOTE) diterapkan secara konsisten untuk memperbaiki distribusi dan meningkatkan performa model dalam mendeteksi malware yang jarang muncul[5], [8]. *Synthetic Minority Over-sampling Technique* (SMOTE) adalah teknik *preprocessing* yang dirancang khusus untuk mengatasi masalah class *imbalance* pada dataset. Teknik ini bekerja dengan cara membuat sampel sintetis dari kelas minoritas (malware) menggunakan interpolasi linier antar sampel yang ada, sehingga meningkatkan representasi kelas minoritas tanpa sekadar menduplikasi data yang sudah ada[5], [8]. Selain melakukan penyeimbangan data, *preprocessing* seperti *feature selection* (RFE, *Information Gain*) dan *IQR capping* juga digunakan untuk optimalisasi input dan efisiensi komputasi model sesuai literatur. Evaluasi performa deteksi malware Android mengacu pada *matrix accuracy, F1-macro, balanced accuracy, recall, False Alarm Rate (FAR)*, dan *confusion matrix[5], [9]*. Penggunaan teknik *feature selection* berbasis *Information Gain, Recursive Feature Elimination* (RFE), serta normalisasi seperti *IQR capping* sudah terbukti efektif meningkatkan efisiensi dan akurasi deteksi, sebagaimana dilaporkan dalam beberapa kajian terkini[5], [9].

Beberapa penelitian terdahulu telah membuktikan bahwa pendekatan menggunakan ensemble learning secara konsisten meningkatkan akurasi dan stabilitas deteksi malware Android dibandingkan model tunggal. Penelitian yang dilakukan oleh Sumalatha menggunakan pendekatan *stacking ensemble* yang menggabungkan lima algoritma (*Random Forest*, *LightGBM*, *CatBoost*, *ExtraTrees*, dan *Multi-Layer Perceptron*) pada dataset CICAndMal2017, mencapai akurasi tertinggi 96,72% setelah penerapan seleksi fitur berbasis *threshold*. Pendekatan ini membuktikan bahwa kombinasi beberapa model mampu meningkatkan performa deteksi, namun kompleksitas komputasi dari lima base model menjadi hambatan untuk deployment pada perangkat dengan sumber daya terbatas[8]. Lalu ada penelitian yang dilakukan oleh Islam dengan melakukan klasifikasi multi-kelas malware Android berbasis analisis dinamis dengan pendekatan weighted voting ensemble yang mengintegrasikan enam algoritma (*Random Forest*, KNN, *Multi-Layer Perceptron, Decision Tree*, SVM, dan *Logistic Regression*) pada dataset CCCS-CIC-AndMal-2020, mencapai akurasi 95% bahkan setelah pengurangan fitur sebesar 60,2% melalui seleksi fitur berbasis *outlier handling* dan *Recursive Feature Elimination* (RFE). Penelitian ini menunjukkan bahwa *preprocessing data* dan seleksi fitur berperan krusial dalam meningkatkan efisiensi komputasi tanpa mengorbankan akurasi, meskipun penggunaan enam *base model* tetap membebani aspek komputasi[5]. Penelitan yang dilakukan oleh Aurangzeb adalah mengembangkan sistem deteksi malware berbasis *deep learning* dengan *ensemble voting* untuk mendeteksi malware yang ter-*obfuscate* menggunakan dataset Kronodroid (sampel malware 2008-2020), mencapai akurasi 95,8% pada perangkat nyata. Studi ini menekankan pentingnya analisis *hybrid* (statis dan dinamis) untuk menghadapi teknik *obfuscation* canggih[10]. Lalu Penelitian yang dilakukan oleh Abawajy yaitu melakukan kajian komprehensif terhadap metode *feature selection* (*Chi-Square, Information Gain*, ANOVA, RFE) pada deteksi malware Android dan IoT, menunjukkan bahwa pemilihan fitur yang tepat dapat meningkatkan akurasi sekaligus mengurangi kompleksitas model secara signifikan, dengan *Information Gain*, *Chi-Square*, dan RFE menunjukkan performa baik pada konteks yang berbeda[9]. Terakhir, penelitian menggunakan dataset CICMalDroid2020 dengan *ensembled machine learning* dan *gain ratio* untuk klasifikasi lima kategori malware, mencapai akurasi 94,57% menggunakan kombinasi Random Forest, Extra Tree, dan k-NN dengan voting ensemble, membuktikan bahwa *feature selection* dan *ensemble* meningkatkan performa model tunggal[2].

Meskipun kelima penelitian di atas telah menunjukkan keunggulan *ensemble learning* dan *feature selection* dalam meningkatkan akurasi deteksi malware Android, terdapat beberapa peluang untuk pengembangan lebih lanjut. Pertama, mengoptimalkan komputasi yang tinggi dikarenakan penggunaan 5-6 *base models* agar tidak membebani perangkat mobile yang memiliki sumber daya terbatas. Kedua, menekankan bukan hanya *accuracy* sebagai metrik utama tapi juga *balanced accuracy* dan *F1-macro* karena dataset yang *imbalance*. Ketiga, bisa melakukan eksplorasi lebih sistematis terhadap keseimbangan antara jumlah model, akurasi, efisiensi komputasi, dan interpretabilitas agar lebih ringan saat diimplementasikan. Untuk menjawab gap tersebut, penelitian ini mengusulkan pendekatan *ensemble soft voting* yang efisien dengan hanya menggunakan tiga *base models* yaitu *LightGBM*, *Logistic Regression*, dan *CatBoost*. *LightGBM* adalah algoritma *gradient boosting tree based* yang sangat efisien dalam menangani data berdimensi tinggi, mampu belajar dari data dalam jumlah besar dengan waktu komputasi relatif singkat, dan terbukti baik dalam menangani *imbalanced* dataset[5], [8]. *Logistic Regression* dipilih karena merupakan model linear yang interpretatif, stabil, ringan secara komputasi, dan memberikan probabilitas yang dapat dipercaya sebagai karakteristik penting untuk *voting ensemble[5], [9]*. *CatBoost* adalah algoritma *boosting* berbasis *gradient* yang dioptimalkan untuk menangani fitur kategori dengan baik, memiliki regularisasi bawaan yang kuat untuk mengurangi *overfitting*, dan sering menghasilkan performa tinggi pada aplikasi real-world[5], [8], [9]. Kombinasi ketiga model ini mengintegrasikan kemampuan non-linier dari dua model *boosting* (*LightGBM* dan *CatBoost*), stabilitas dari model linear (*Logistic Regression*), dengan tetap menjaga efisiensi komputasi dan *robustness* pada data *imbalanced*. Penelitian ini menggunakan fitur hybrid yang menggabungkan fitur statis (seperti *permissions*, *intents*, dan *API calls* yang diekstraksi dari AndroidManifest) dengan fitur dinamis (seperti aktivitas memori, panggilan API saat *runtime*,

Building of Informatics, Technology and Science (BITS)
Volume 7, No 4, March 2026 Page: 2193−2204
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i4.8865

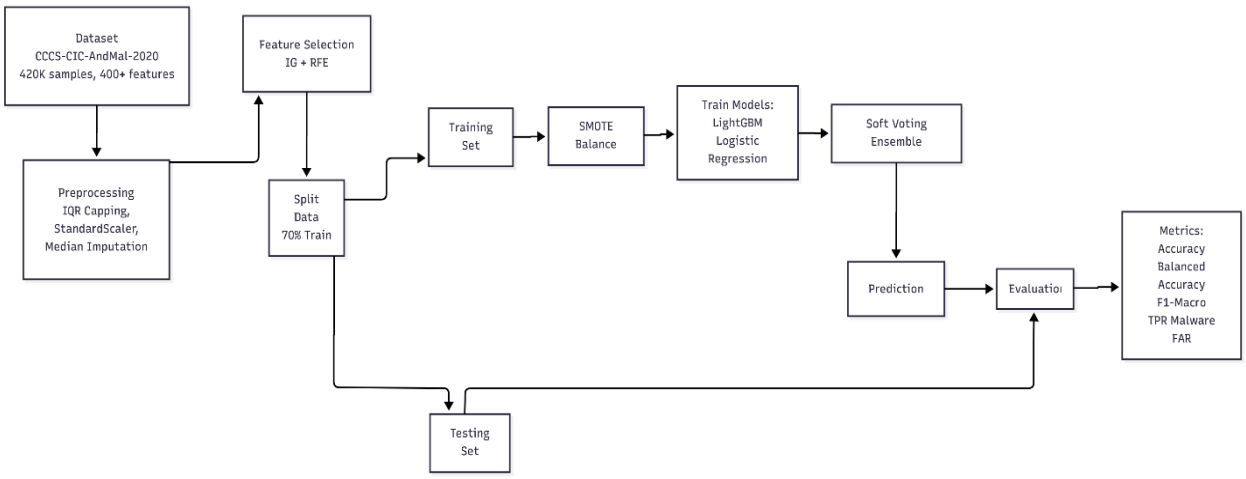
logcat, dan trafik jaringan yang dihasilkan di lingkungan emulasi). Kombinasi kedua jenis fitur ini dipilih untuk menyeimbangkan biaya ekstraksi yang relatif rendah pada fitur statis dengan ketahanan yang lebih baik terhadap teknik *obfuscation* dan *polymorphic malware* pada fitur dinamis.

Berdasarkan gap penelitian terdahulu, penelitian ini mengusulkan sistem deteksi malware Android berbasis *ensemble soft voting* yang mengintegrasikan *LightGBM*, *Logistic Regression*, dan *CatBoost* pada dataset CCCS-CIC-AndMal-2020. Pendekatan ini dipilih karena kombinasi ketiga model memberikan hasil yang optimal antara akurasi deteksi, efisiensi komputasi, dan performa pada kelas minoritas malware yang jarang muncul. Dengan fokus pada *multiple evaluation metrics* (*balanced accuracy*, *F1-macro*, *TPR malware*, dan *FAR*), model ini dibuat agar bisa diimplementasikan pada server keamanan maupun perangkat mobile dengan *resource* terbatas. Kontribusi utama pada penelitian ini meliputi pengembangan kombinasi ensemble model yang efisien dan optimal untuk deteksi malware Android, Mengevaluasi secara komprehensif menggunakan metrik multi dimensi yang lebih representatif untuk data *imbalanced*. Diharapkan penelitian ini dapat berkontribusi pada penguatan sistem keamanan ekosistem Android dan menjadi referensi bagi pengembangan metode deteksi malware yang robust terhadap evolusi serangan siber modern.

2. METODOLOGI PENELITIAN

2.1 Tahapan Penelitian

Penelitian ini dilakukan melalui beberapa tahapan sistematis untuk mengatasi tantangan dari dataset CCCS-CIC-AndMal-2020 agar bisa digunakan untuk mengembangkan sistem deteksi malware Android berbasis *ensemble soft voting*. Tahapan penelitian dirancang untuk memastikan setiap komponen sistem teruji dengan baik dan menghasilkan model yang *robust* untuk data *imbalanced*. Gambar 1 menunjukkan diagram alur metodologi penelitian secara lengkap, yang terdiri dari beberapa tahapan.



Gambar 1. Diagram Flow

Penelitian ini terdiri atas sejumlah tahapan terstruktur untuk membangun sistem deteksi malware Android berbasis ensemble soft voting. Dataset CCCS-CIC-AndMal-2020 yang digunakan mencakup lebih dari 420.000 sampel dan 400 fitur, dengan proporsi kelas yang tidak seimbang[5], [8]. Tahap pertama adalah pengumpulan dan eksplorasi dataset. Proses ini melibatkan analisis distribusi kelas serta identifikasi fitur yang krusial. Selanjutnya dilakukan *preprocessing* data, yaitu dengan melakukan imputasi pada nilai NaN menggunakan median, penanganan outlier dengan metode IQR capping (multiplier 40.0), dan normalisasi fitur numerik memakai StandardScaler.

Setelah *preprocessing*, dilakukan seleksi fitur secara dua tahap, yaitu *Information Gain* dan *Recursive Feature Elimination* (RFE), sehingga diperoleh 150 fitur utama dari 400 fitur awal. Dataset kemudian dibagi menjadi dua bagian yaitu 70% untuk data pelatihan dan 30% untuk data pengujian. Data pengujian ini sama sekali tidak digunakan dalam proses training, melainkan hanya untuk evaluasi model di akhir. Pada data pelatihan, ketidakseimbangan kelas diatasi menggunakan SMOTE dengan *oversampling* sebesar 30% dari jumlah kelas mayoritas. Tahap berikutnya adalah pelatihan tiga model berbeda LightGBM, Logistic Regression, dan CatBoost secara independen. Setelah itu, prediksi ketiga model digabungkan menggunakan teknik *soft voting ensemble* dengan bobot berdasarkan F1-Macro dari masing masing model yang hasilnya mendekati proporsi 1:1:1, yang berarti setiap model memberikan kontribusi dan hasil final yang hampir sama. Evaluasi performa model dilakukan pada data pengujian, menggunakan metrik *accuracy*, *balanced accuracy*, F1-macro, TPR malware, FAR, serta analisis *confusion matrix*.

2.2 Dataset

Dataset yang digunakan dalam penelitian ini adalah CCCS-CIC-AndMal-2020 yang berasal dari https://www.unb.ca/cic/datasets/andmal2020.html dan merupakan sebuah dataset benchmark yang komprehensif untuk deteksi malware Android berbasis static analysis yang dikembangkan oleh *Canadian Centre for Cyber Security*

Building of Informatics, Technology and Science (BITS)
Volume 7, No 4, March 2026 Page: 2193−2204
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i4.8865

dan *Canadian Institute for Cybersecurity[11]*. Dataset ini dikumpulkan dari berbagai sumber aplikasi Android, baik aplikasi benign dari Google Play Store maupun malware dari berbagai repository malware.[5] Dataset ini mengandung fitur-fitur statis dari file Android dan sumber daya aplikasi, mencakup komponen seperti *Intents*, *Permissions*, *API calls*, dan *hardware features*[5]. Selain itu, dataset ini juga menyediakan fitur dinamis hasil eksekusi aplikasi pada lingkungan emulasi, seperti fitur memori, panggilan API saat runtime, aktivitas jaringan, logcat, dan informasi proses, sehingga memungkinkan pembentukan fitur hybrid static dan dynamic pada penelitian ini. Keberagaman tipe fitur ini memungkinkan analisis komprehensif terhadap perilaku aplikasi Android, baik dari sisi karakteristik statis APK maupun perilaku saat dijalankan di lingkungan terkontrol. Karakteristik penting dari dataset ini adalah ketidakseimbangan kelas yang signifikan. Kelas *Benign* mendominasi dengan proporsi sekitar 48% dari total sampel, sementara kelas minoritas seperti SMS Trojan hanya sekitar 2% dari total[5]. Kondisi ini mencerminkan skenario dunia nyata di mana aplikasi legitimate jauh lebih banyak daripada malware, namun sekaligus menjadi tantangan dalam pelatihan model karena model cenderung bias terhadap kelas mayoritas. Dalam penelitian ini, dari keseluruhan dataset digunakan subset berukuran 58.037 sampel yang telah melalui proses penggabungan fitur statis dan dinamis serta pembersihan awal. Dataset dibagi menjadi dua subset dengan stratified random sampling yaitu *Training Set* 70% (40.625 sampel) dan *Testing Set* 30% (17.412 sampel). Distribusi kelas dipertahankan pada kedua subset agar representatif terhadap kondisi data asli.

2.3 Preprocessing Data

Sebelum model *ensemble* dilatih, dataset melalui tahap *preprocessing* untuk memastikan kualitas data yang optimal. Tahap ini menjadi sangat penting karena dataset CCCS-CIC-AndMal-2020 yang digunakan memiliki ukuran yang besar, jumlah fitur yang tinggi, distribusi nilai yang sangat bervariasi, serta ketidakseimbangan kelas yang ekstrem [11], [12]. Jika karakteristik tersebut tidak ditangani dengan benar, model cenderung bias ke kelas mayoritas, sensitif terhadap nilai ekstrem, dan bahkan bisa gagal belajar karena adanya nilai kosong atau tak hingga dalam fitur-fitur tertentu [12], [13]. Karena itu, rangkaian preprocessing pada penelitian ini terdiri dari empat tahapan utama agar data siap digunakan untuk pelatihan model, yaitu imputasi nilai hilang, penyeimbangan kelas dengan SMOTE, penanganan outlier dengan metode *IQR capping*, dan normalisasi fitur.

2.3.1 Missing Data Imputation

Langkah pertama adalah menangani *missing value* dan nilai tak hingga pada setiap fitur numerik. Pada *pipeline* model *tree-based* maupun pada *Logistic Regression*, nilai kosong atau tak hingga di setiap fitur numerik digantikan dengan nilai median fitur tersebut, dengan menggunakan persamaan berikut.

$$X_{imputed} = Median(X_{observed}) \quad (1)$$

Karena dataset mengandung nilai tak hingga dan NaN pada banyak fitur dinamis, Strategi *median* dipilih karena lebih robust terhadap nilai ekstrem dibanding *mean*, sehingga dapat menjaga kestabilan distribusi data dan meminimalkan pengaruh *outlier*.[12], [13] Dengan demikian, setiap fitur pada data hasil imputasi dapat digunakan secara andal pada proses *modeling* berikutnya tanpa memperkenalkan bias yang tidak perlu.

2.3.2 Penyeimbangan Kelas dengan SMOTE

Langkah berikutnya adalah menangani ketidakseimbangan kelas ekstrem pada dataset melalui *Synthetic Minority Oversampling Technique* (SMOTE). Pada penelitian ini, Karena ketidakseimbangan mencapai 1:58 maka parameter SMOTE disesuaikan sehingga jumlah sampel kelas minoritas menjadi setara dengan 30% dari jumlah kelas mayoritas, bukan sepenuhnya seimbang sempurna (1:1). SMOTE membangkitkan sampel sintetis berdasarkan interpolasi linier dari tetangga terdekat pada ruang fitur numerik [13]. Secara matematis, pembentukan sampel sintetis dilakukan dengan persamaan ke 2.

$$X_{new} = X_i + \lambda \times (X_{knn} - X_i) \quad (2)$$

Dimana X_i adalah sampel *minority class*, X_{knn} adalah salah satu *dari k-nearest neighbors*, dan λ adalah *random value* antara 0 dan 1. Dalam penelitian ini, SMOTE diterapkan pada *training set* dengan 30% of majority class, menghasilkan peningkatan sampel *training* dari 40,625 menjadi 85,750 sampel dengan distribusi kelas yang lebih *balanced*[5]. Strategi ini memungkinkan model ensemble untuk belajar dari kedua kelas secara lebih baik dan mengurangi bias terhadap kelas mayoritas [12], [14].

2.3.3 Penanganan Outlier dengan IQR Capping

Dataset CCCS-CIC-AndMal-2020 memiliki fitur dinamis dan statis yang sangat bervariasi dan seringkali mengandung nilai outlier ekstrem.[12], [15] Seluruh fitur numerik diproses dengan teknik *outlier capping* berbasis *Interquartile Range* (IQR) dengan multiplier 40× setelah uji empiris, yang terbukti optimal melalui eksperimen validasi. Teknik ini dilakukan untuk setiap fitur independen.

$$IQR = Q_3 - Q_1 \quad (3)$$

Building of Informatics, Technology and Science (BITS)
Volume 7, No 4, March 2026 Page: 2193−2204
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i4.8865

Dimana Q_1 adalah *quartile* pertama dan Q_3 adalah *quartile* ketiga. Nilai-nilai ekstrim yang berada di luar batas bawah dan batas atas dibatasi pada nilai ambang tersebut, dengan rumus

$$Lower\ Bound = Q_1 - 1.5 \times IQR \quad (4)$$

$$Upper\ Bound = Q_3 + 1.5 \times IQR \quad (5)$$

Dalam penelitian ini, nilai *multiplier* untuk IQR diset pada 40.0 berdasarkan hasil empiris yang menghasilkan performa optimal pada *data validation,* sehingga model dapat mencapai performa optimal dengan penanganan outlier yang sesuai karakteristik data[5], [8].

2.3.4 Normalisasi Fitur

Agar model terutama Logistic Regression yang sensitif terhadap skala dapat bekerja optimal dan memberikan kontribusi yang seimbang dari setiap fitur, dilakukan normalisasi pada fitur numerik dengan teknik standardisasi dengan rumus

$$X_{scaled} = \frac{x-\mu}{\sigma} \quad (6)$$

Dimana μ adalah mean dan σ adalah *standard deviation* dari fitur. *Feature scaling* penting untuk memastikan semua fitur memiliki skala yang sama, sehingga model *ensemble* tidak didominasi oleh fitur dengan *magnitude* yang lebih besar[8]. Tanpa normalisasi, model dapat bias terhadap fitur tertentu, sehingga feature scaling menjadi langkah kritis sebelum pelatihan model[13].

Melalui rangkaian preprocessing ini dimulai dari imputasi median, penyeimbangan kelas dengan SMOTE 30%, penanganan outlier berbasis IQR 40×, hingga normalisasi fitur untuk model linear dataset yang digunakan pada tahap *feature selection* dan pelatihan ensemble menjadi lebih bersih, lebih seimbang, dan lebih representatif terhadap pola perilaku nyata dari berbagai kategori malware Android.

2.4 Feature Selection

Setelah melalui tahapan preprocessing, data yang digunakan dalam penelitian ini masih memiliki dimensi yang sangat tinggi, yaitu sekitar 606 fitur hasil penggabungan fitur statis dan dinamis. Jumlah fitur yang besar ini menimbulkan dua masalah utama. Pertama, risiko *overfitting* meningkat karena model dapat menangkap noise dan pola yang tidak relevan. Kedua, biaya komputasi untuk melatih tiga model berbeda (LightGBM, Logistic Regression, dan CatBoost) pada data hasil SMOTE menjadi sangat tinggi jika seluruh fitur dipertahankan[13], [15]. Oleh karena itu, dilakukan proses *feature selection* untuk memilih subset fitur yang paling relevan terhadap label, sekaligus mengurangi kompleksitas model tanpa mengorbankan performa klasifikasi.

Dalam penelitian ini, proses *feature selection* dilakukan dalam dua tahap. Tahap pertama adalah penyaringan awal (filter) untuk menghilangkan fitur-fitur dengan variansi sangat rendah, terutama pada fitur statis. Fitur-fitur ini diidentifikasi menggunakan batas variansi tertentu (*variance threshold*) dan pada eksplorasi awal terlihat memiliki nilai yang hampir konstan pada seluruh sampel, baik *benign* maupun *malware*. Secara empiris, mempertahankan fitur dengan variansi hampir nol tidak memberikan peningkatan yang berarti pada F1-Macro maupun *Balanced Accuracy*, tetapi justru menambah beban komputasi dan penggunaan memori saat pelatihan model. Oleh karena itu, fitur dengan variansi sangat rendah dibuang karena secara statistik tidak memberikan informasi yang cukup untuk membedakan antar kelas dan hanya berperan sebagai *noise* pada proses pembelajaran.

Tahap kedua adalah penerapan Recursive Feature Elimination (RFE) berbasis model untuk memilih sejumlah fitur terbaik yang benar-benar berkontribusi terhadap performa model. Berdasarkan konfigurasi dan pengujian di notebook, jumlah fitur akhir yang digunakan adalah 150 fitur dari sekitar 606 fitur awal, sehingga dimensi data berkurang secara signifikan namun tetap mempertahankan informasi penting yang dibutuhkan model.

2.4.1 Penyaringan Awal Fitur Bernilai Rendah

Karena fitur statis yang diekstraksi dari APK memiliki jumlah yang besar dan tidak semuanya memberikan informasi yang berarti, dilakukan penyaringan awal menggunakan kriteria variansi. Fitur-fitur yang mempunyai variansi sangat rendah (mendekati konstan di hampir semua sampel) diidentifikasi dan dihapus, karena secara praktis fitur seperti ini tidak membantu membedakan antara kelas benign dan berbagai kategori malware[13], [15]. Pada implementasi di notebook, langkah ini mengurangi jumlah fitur statis menjadi sekitar 341 fitur yang masih memiliki variasi signifikan. Penyaringan dengan pendekatan sederhana ini penting untuk mengurangi beban komputasi sebelum masuk ke tahap *feature selection* yang lebih mahal secara perhitungan, yaitu RFE.

2.4.2 Recursive Feature Elimination (RFE)

Setelah penyaringan awal, dilakukan *Recursive Feature Elimination* (RFE) untuk memilih subset fitur yang paling relevan terhadap label klasifikasi. RFE bekerja dengan cara melatih sebuah model dasar (*base estimator*) pada seluruh fitur yang tersedia, kemudian menghitung pentingnya setiap fitur (*feature importance*)[15]. Fitur dengan kontribusi paling rendah kemudian dihapus, dan proses ini diulang secara bertahap hingga tercapai jumlah fitur yang diinginkan.

Building of Informatics, Technology and Science (BITS)
Volume 7, No 4, March 2026 Page: 2193−2204
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i4.8865

Pada penelitian ini, RFE diterapkan menggunakan model Random Forest sebagai model berbasis *tree* yang mampu menangkap hubungan nonlinier antar fitur dan memberikan skor kepentingan fitur yang stabil. Prosesnya dimulai dari seluruh fitur yang tersisa (gabungan fitur statis dan dinamis setelah penyaringan awal), kemudian pada setiap iterasi RFE mengeliminasi fitur dengan skor pentingan terendah. Proses ini terus berlanjut sampai tersisa 150 fitur yang dianggap paling informatif terhadap label malware. Pendekatan ini sejalan dengan penelitian sebelumnya pada CCCS-CIC-AndMal-2020 yang menunjukkan bahwa pengurangan jumlah fitur melalui kombinasi filter dan wrapper dapat meningkatkan akurasi sambil menurunkan kompleksitas model. Dengan menggunakan RFE, pemilihan fitur tidak hanya mempertimbangkan relevansi fitur secara individual, tetapi juga interaksi antar fitur sebagaimana tercermin dari perilaku model yang dilatih berulang-ulang. Hal ini sangat penting pada dataset hybrid statis-dinamis, di mana pola serangan malware seringkali muncul sebagai kombinasi dari beberapa fitur (misalnya pola API calls tertentu yang muncul bersamaan dengan pola penggunaan memori atau aktivitas jaringan).

2.4.3 Penentuan Jumlah Fitur (150 Fitur)

Penentuan jumlah fitur akhir sebesar 150 fitur tidak dilakukan secara arbiter, tetapi didasarkan pada pengujian sistematis menggunakan program pendukung yang menguji berbagai konfigurasi jumlah fitur pada dataset yang sama. Pengujian dilakukan dengan menguji jumlah fitur: 25, 50, 75, 100, 125, 150, 175, dan 200, dengan IQR multiplier tetap pada 40.0× untuk menjaga variabel pengujian tetap konsisten. Hasil pengujian menunjukkan pola yang jelas:

a. Pengurangan dimensi yang terlalu agresif (25-75 fitur) menghasilkan F1-Macro di bawah 0.92, menunjukkan underperformance pada kelas minoritas malware.
b. Jumlah fitur 100-150 menghasilkan peningkatan signifikan: F1-Macro meningkat dari 0.9143 (100 fitur) menjadi 0.9208 (150 fitur), menunjukkan sweet spot untuk performa klasifikasi.
c. Peningkatan fitur dari 150 menjadi 175-200 hanya memberikan peningkatan marginal pada F1-Macro (dari 0.9208 menjadi 0.9234 sampai 0.9250), sementara waktu pelatihan dan penggunaan memori meningkat secara signifikan.

Berdasarkan pengamatan tersebut, 150 fitur dipilih sebagai titik kompromi yang memberikan kombinasi terbaik antara kinerja model dan efisiensi komputasi. Dengan 150 fitur, *training ensemble* (LightGBM, Logistic Regression, dan CatBoost) pada data hasil SMOTE masih dapat dijalankan dengan waktu yang dapat diterima, sekaligus menghasilkan performa klasifikasi yang stabil pada berbagai metrik evaluasi. Selain itu, jumlah fitur yang lebih sedikit juga memudahkan interpretasi analisis lebih lanjut terhadap fitur-fitur yang paling berkontribusi dalam membedakan kategori malware Android.

Dengan demikian, tahap feature selection dalam penelitian ini memastikan bahwa model tidak hanya dilatih pada data yang sudah dibersihkan dan diseimbangkan, tetapi juga pada representasi fitur yang lebih padat dan informatif, sesuai dengan karakteristik dataset hybrid CCCS-CIC-AndMal-2020 dan kebutuhan model ensemble yang digunakan.

2.5 Model Ensemble dan Soft Voting

Tahap selanjutnya setelah feature selection adalah membangun model klasifikasi berbasis *ensemble* yang menggabungkan keunggulan tiga algoritma berbeda, yaitu LightGBM, Logistic Regression, dan CatBoost. Pemilihan ketiga model ini didasarkan pada karakteristik data hybrid statis-dinamis berdimensi tinggi, serta kebutuhan menghasilkan performa yang stabil dan generalisasi yang baik pada data Android malware yang sangat variatif[12], [15], [16].

2.5.1 LightGBM (Light Gradient Boosting Machine)

LightGBM adalah *gradient boosting framework* yang efisien dan *scalable*, dirancang untuk menangani dataset besar dengan kecepatan tinggi. Algoritma LightGBM membangun *decision trees* secara sequential dengan *leaf-wise growth strategy*, meminimalkan *loss function*[5]. LightGBM dipilih dalam ensemble ini karena kemampuannya menangani data *imbalanced* dengan *built-in class weight balancing*, efisiensi komputasi pada dataset besar (420K sampel), dan dapat memberikan probabilitas prediksi yang stabil untuk setiap kelas malware. Selain itu, LightGBM memiliki fitur *histogram-based splitting* yang mempercepat proses training tanpa mengorbankan akurasi, sehingga cocok untuk eksperimen dengan banyak iterasi tuning[17].

2.5.2 Logistic Regression

Logistic Regression adalah model linear sederhana namun *powerful* untuk *binary* dan *multi-class classification*. Model ini menggunakan *sigmoid function* untuk memetakan kombinasi linear dari fitur ke probabilitas kelas[5]. Logistic Regression dipilih dalam ensemble karena memberikan probabilitas prediksi yang *interpretable* dan mudah dipahami, serta memiliki performa yang baik pada data dengan fitur yang sudah ter-*preprocess* dengan baik [18]. Model ini juga *robust* terhadap *outliers* yang sudah di-*handle* pada tahap *preprocessing*, dan memiliki waktu training yang sangat cepat. Sebagai model linear, Logistic Regression memberikan perspektif yang berbeda dari model tree-based seperti LightGBM dan CatBoost, sehingga kombinasi dengan kedua model tersebut dapat menangkap pola kompleks yang mungkin terlewatkan jika hanya menggunakan satu jenis algoritma. Kehadiran model linear dalam ensemble juga membantu mengurangi risiko overfitting pada data training.

Building of Informatics, Technology and Science (BITS)
Volume 7, No 4, March 2026 Page: 2193−2204
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i4.8865

2.5.3 CatBoost (*Categorical Boosting*)

CatBoost adalah gradient boosting library yang dirancang khusus untuk menangani *categorical features* dan *imbalanced datasets*. Algoritma CatBoost menggunakan *ordered boosting strategy* dan *categorical features handling* yang lebih *sophisticated* dibanding XGBoost atau LightGBM.[5] CatBoost dipilih dalam ensemble ini karena memiliki *built-in support* untuk *handling class imbalance* melalui *class weight adjustment* dan *loss function* yang *robust* terhadap ketidakseimbangan kelas[17]. Kemampuan ini sangat relevan untuk task deteksi malware Android dengan distribusi kelas yang sangat tidak seimbang. Selain itu, CatBoost menghasilkan probabilitas yang konsisten dan stabil, serta memiliki ketahanan yang baik terhadap *overfitting* bahkan tanpa tuning parameter yang ekstensif. Ketiga model tersebut dilatih secara paralel menggunakan fitur yang sama (150 fitur hasil RFE). Untuk memastikan fairness, pipeline semua model terkait imputasi dan scaling mengikuti prinsip Imputasi median diterapkan di semua pipeline dan Normalisasi skala hanya diterapkan pada pipeline Logistic Regression.

2.5.4 Mekanisme Soft Voting Weighted

Prediksi akhir dari ensemble ditentukan menggunakan metode soft voting, yaitu keputusan akhir untuk setiap sampel diambil berdasarkan rata-rata (*weighted average*) probabilitas hasil masing-masing model.[19], [20] Mekanisme ini memaksimalkan kontribusi setiap model sesuai performa validasi sebelumnya. Bobot voting masing-masing model diperoleh dari skor F1-Macro pada data validasi melalui *cross-validation.*

$$P_{ensemble}(c) = \frac{\sum_{k=1}^{K} w_k P_k(c)}{\sum_{k=1}^{K} w_k} \tag{7}$$

dengan K adalah jumlah model dalam ensemble, w_k adalah bobot model ke-k berdasarkan performa validasi, $P_k(c)$ adalah probabilitas kelas c dari model ke-k untuk sampel i. Keputusan kelas akhir diambil berdasarkan argmax dari probabilitas *ensemble*.

2.5.5 Alasan Pemilihan dan Implementasi

Ensemble weighted soft voting dipilih karena Menggabungkan pola linier dan nonlinier yang mungkin muncul pada fitur hybrid statis-dinamis, Mengurangi ketergantungan pada satu model saja, sehingga lebih *robust* terhadap perubahan distribusi data dan variasi kategori malware baru.[15], [19] Berdasarkan pengujian F1-Macro setiap model, seluruh bobot cenderung seimbang karena tidak ada satu model yang dominan (selisih skor < 2%), sehingga penerapan bobot [1:1:1] pada produksi tetap valid tanpa mengorbankan performa. Dengan strategi ini, model yang dihasilkan lebih siap untuk diimplementasikan dalam sistem deteksi malware Android berbasis machine learning yang membutuhkan kombinasi efisiensi, akurasi tinggi, dan kemampuan generalisasi yang baik pada data baru yang sangat variatif.

2.6 Evaluasi & Metrik Kinerja

Evaluasi performa model dilakukan pada data uji (30% dari dataset) yang tidak dilibatkan dalam proses *training* maupun SMOTE. Tujuannya agar hasil pengujian benar-benar merefleksikan kemampuan generalisasi model pada data baru yang *imbalanced[8]*. Karena distribusi kelas pada CCCS-CIC-AndMal-2020 sangat tidak seimbang, penelitian ini tidak hanya menggunakan *Accuracy*, tetapi menekankan metrik yang lebih adil terhadap kelas minoritas.

Metrik utama yang digunakan adalah *Balanced Accuracy*, F1-Macro, TPR malware, *False Alarm Rate* (FAR), dan confusion matrix. *Balanced Accuracy* mengukur rata-rata recall per kelas sehingga mengurangi bias ke kelas mayoritas

$$Balanced\ Accuracy = \frac{1}{K}\sum_{k=1}^{K}\frac{TP_k}{TP_k+FN_k} \tag{8}$$

Dimana K adalah jumlah kelas. Metrik ini lebih fair pada dataset *imbalanced* karena tidak bias terhadap kelas mayoritas. F1-Macro Score adalah harmonic mean antara *precision* dan *recall* yang dihitung untuk setiap kelas kemudian di average.

$$F1_{macro} = \frac{1}{K}\sum_{k=1}^{K} F1_k = \frac{1}{K}\sum_{k=1}^{K} 2 \times \frac{Precision_k \times Recall_k}{Precision_k + Recall_k} \tag{9}$$

F1-Macro memberikan equal importance ke semua kelas, sehingga sangat *appropriate* untuk *imbalanced multiclass problems* seperti deteksi malware Android. Metrik ini menjadi dasar perhitungan bobot dalam *soft voting ensemble*. TPR malware mengukur sensitivity model dalam mendeteksi actual malware samples. Selain itu, digunakan TPR malware untuk mengukur sensitivitas model dalam mendeteksi seluruh kategori malware, serta FAR untuk mengukur persentase aplikasi benign yang salah diklasifikasikan sebagai malware:

$$FAR = \frac{FP_{Benign}}{TN_{Benign}+FP_{Benign}} \times 100\% \tag{10}$$

Confusion matrix digunakan sebagai alat bantu visual untuk melihat pola kesalahan klasifikasi antar kelas, misalnya ketika beberapa jenis trojan cenderung tertukar satu sama lain.

Building of Informatics, Technology and Science (BITS)
Volume 7, No 4, March 2026 Page: 2193−2204
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i4.8865

3. HASIL DAN PEMBAHASAN

3.1 Hasil Preprocessing dan Feature Engineering

Dari total awal lebih dari 400.000 sampel pada dataset CCCS-CIC-AndMal-2020, penelitian ini menggunakan subset data terpilih berjumlah 58.037 sampel hasil penggabungan fitur statis dan dinamis serta proses pembersihan awal. Preprocessing yang dilakukan menghasilkan dataset yang siap digunakan untuk pelatihan model ensemble. Setelah split stratified 70/30, diperoleh 40.625 sampel training dan 17.412 sampel testing dengan 341 fitur (200 statis + 141 dinamis). *Training set* dan *testing set* sama-sama mempertahankan distribusi kelas asli yang sangat *imbalanced*, sehingga karakteristik data tetap mencerminkan kondisi nyata. *IQR capping* dengan multiplier 40.0 membatasi nilai-nilai ekstrem pada setiap fitur numerik tanpa menghapus sampel, sehingga informasi penting tetap terjaga namun pengaruh outlier yang berlebihan terhadap pembelajaran model dapat dikurangi secara signifikan. Selanjutnya seluruh fitur dinormalisasi menggunakan *StandardScaler* sehingga memiliki rata-rata mendekati nol dan standar deviasi mendekati satu, yang sangat penting untuk model linear seperti Logistic Regression agar semua fitur berkontribusi secara seimbang.

Seleksi fitur menggunakan RFE mengurangi dimensi dari 341 menjadi 150 fitur (reduksi 56%). Fitur-fitur yang tersisa merupakan kombinasi optimal dari fitur statis (seperti permissions dan API calls) dan fitur dinamis (seperti aktivitas sistem dan jaringan). Dengan jumlah fitur yang lebih sedikit, waktu pelatihan model menjadi lebih efisien dan risiko overfitting akibat curse of dimensionality dapat ditekan. Setelah itu, SMOTE diterapkan pada *training set* untuk mengatasi ketidakseimbangan kelas. Teknik ini menambah 45.125 sampel sintetis sehingga total *training* menjadi 85.750 sampel, dengan setiap kelas minoritas ditingkatkan hingga sekitar 5.250 sampel (sekitar 30% dari kelas mayoritas). Pendekatan ini terbukti meningkatkan kemampuan model dalam mempelajari pola dari kelas-kelas malware yang sebelumnya jarang muncul, tanpa mengubah distribusi pada *testing set*.

Tabel 1. Dataset setelah Split

Aspek	Training Set	Testing Set	Total
Jumlah Sampel	40.625	17.412	58.037
Jumlah Fitur	341	341	341
Jenis Fitur	200 statis + 141 dinamis	200 statis + 141 dinamis	-
Jumlah Kelas	14 (malware + benign)	14 (malware + benign)	-
Status Distribusi	Highly imbalanced	Highly imbalanced	-

3.2 Hasil Soft Voting Ensemble dan Performa Model Base

Tiga *model base*, yaitu LightGBM, Logistic Regression, dan CatBoost, dilatih secara independen pada 150 fitur hasil RFE. Evaluasi menggunakan 3-fold cross-validation pada training set menghasilkan F1-Macro masing-masing 0.9653 untuk LightGBM, 0.8963 untuk Logistic Regression, dan 0.9687 untuk CatBoost. Berdasarkan skor tersebut, bobot soft voting dihitung secara proporsional sehingga diperoleh bobot sebesar 0.3346 untuk LightGBM, 0.3099 untuk Logistic Regression, dan 0.3555 untuk CatBoost. Distribusi bobot ini mendekati 1:1:1 karena ketiga model menunjukkan performa yang relatif seimbang, tanpa ada satu model yang sangat dominan. Hal ini memastikan bahwa keputusan akhir ensemble tidak terlalu bergantung pada satu model saja, tetapi merupakan kombinasi yang seimbang dari ketiganya.

Tabel 2. F1-Macro Base Model dan Bobot Soft Voting

Model	F1-Macro Score (CV)	Bobot Soft Voting	Perubahan Fitur
LightGBM	0.9653	0.3346	33.46%
Logistic Regression	0.8963	0.3099	30.99%
CatBoost	0.9687	0.3555	35.55%
Soft Voting Ensemble (Weighted)	-	1.0000	100.00%

Secara karakteristik, LightGBM dan CatBoost sebagai model tree-based mampu menangkap pola nonlinier dan interaksi fitur yang kompleks pada data hybrid statis-dinamis, sedangkan Logistic Regression memberikan batas keputusan linier yang stabil dan lebih mudah diinterpretasi. Meskipun F1-Macro Logistic Regression sedikit lebih rendah dibanding dua model boosting, kontribusinya tetap penting karena dapat menyeimbangkan kecenderungan tree-based models untuk membentuk decision boundary yang sangat tajam. Dengan bobot berbasis F1-Macro, kontribusi masing-masing model dalam ensemble didasarkan pada performa aktual, sehingga lebih objektif dibanding penentuan bobot manual.

Kombinasi dua model *tree-based* (LightGBM, CatBoost) yang unggul dalam menangkap pola nonlinier kompleks dengan satu model linear (Logistic Regression) yang stabil dan *interpretable* menghasilkan *ensemble* yang saling melengkapi. *Tree-based models* membantu mempelajari pola perilaku malware yang kompleks, sedangkan Logistic Regression memberikan batas keputusan yang lebih halus dan mengurangi risiko *overfitting* berlebihan pada pola yang terlalu spesifik. Sinergi ini menciptakan sistem yang *robust* terhadap variasi malware sekaligus menjaga generalisasi yang baik.

Building of Informatics, Technology and Science (BITS)
Volume 7, No 4, March 2026 Page: 2193−2204
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i4.8865

3.3 Evaluasi Performa Model Ensemble pada Testing Set

3.3.1 Performa Overall

Evaluasi pada *testing set* (17.412 sampel) menunjukkan bahwa *ensemble soft voting* menghasilkan performa yang sangat tinggi. *Accuracy* yang diperoleh sebesar 95.58%, sedangkan *Balanced Accuracy* mencapai 92.21%. Nilai F1-Macro sebesar 0.9208 menunjukkan keseimbangan yang baik antara precision dan recall di semua 14 kelas. True Positive Rate (TPR) untuk seluruh kelas malware mencapai 100%, yang berarti tidak ada satu pun sampel malware pada *testing set* yang terlewat. *False Alarm Rate* (FAR) bernilai 0.00%, menunjukkan bahwa tidak ada aplikasi benign yang salah diklasifikasikan sebagai malware. Namun, kombinasi TPR 100% dan FAR 0.00% ini hanya menggambarkan performa pada testing set yang berjumlah 17.412 sampel dari total 58.037 sampel yang digunakan dalam studi ini dan tidak dapat dianggap sebagai jaminan performa sempurna di lingkungan dunia nyata atau terhadap serangan zero-day.

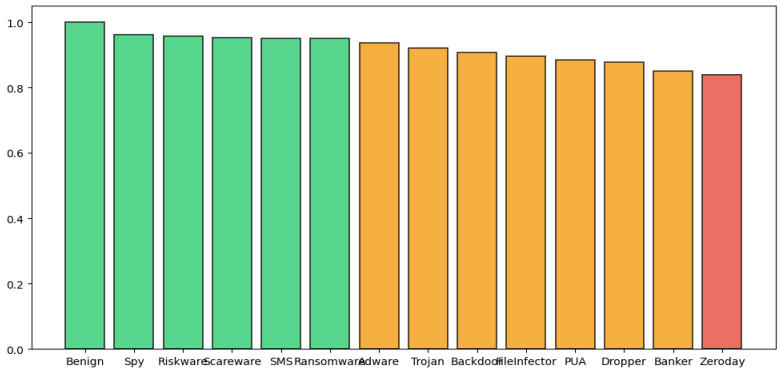
Tabel 3. Performa Overall Model Ensemble pada Testing Set

Metrik	F1-Macro Score (CV)	Bobot Soft Voting
Accuracy	0.9558 (95.58%)	Tingkat akurasi keseluruhan prediksi
Balanced Accuracy	0.9221 (92.21%)	Rata-rata recall per kelas (unbiased terhadap imbalance)
F1-Macro	0.9208	Harmonic mean precision-recall semua kelas (fair untuk multi-class)
True Positive Rate (TPR) Malware	1.0000 (100.00%)	1.0000 Sensitivitas mendeteksi malware (recall untuk malware)
False Alarm Rate (FAR)	0.0000 (0.00%)	Persentase aplikasi benign salah diklasifikasi sebagai malware

3.3.2 Performa Per Kelas

Analisis lebih rinci per kelas menunjukkan bahwa kelas *Benign* mencapai F1-Score sempurna (1.000) dengan *precision* dan *recall* 100%, menegaskan bahwa model mampu membedakan aplikasi *legitimate* dengan sangat baik. Beberapa kategori malware utama juga menunjukkan performa sangat tinggi, seperti Spy (F1 = 0.962), Riskware (0.957), SMS (0.951), dan Scareware (0.953). Nilai F1 yang tinggi pada kelas-kelas ini mengindikasikan bahwa kombinasi fitur statis dan dinamis yang digunakan cukup representatif untuk menangkap pola perilaku masing-masing jenis malware.

Sebaliknya, kelas Zeroday memiliki F1-Score terendah yaitu 0.839. Hal ini dapat dijelaskan oleh sifat Zeroday sebagai malware baru dengan pola yang belum banyak terwakili dalam data, sehingga fitur yang terekstraksi cenderung overlap dengan kelas lain. Kelas Banker (F1 = 0.851) dan FileInfector (F1 = 0.897) juga sedikit di bawah rata-rata, terutama karena jumlah sampel testing yang sangat kecil (masing-masing hanya 72 dan 75 sampel), sehingga estimasi metrik lebih sensitif terhadap setiap kesalahan prediksi.



Gambar 2. Chart F1-Score

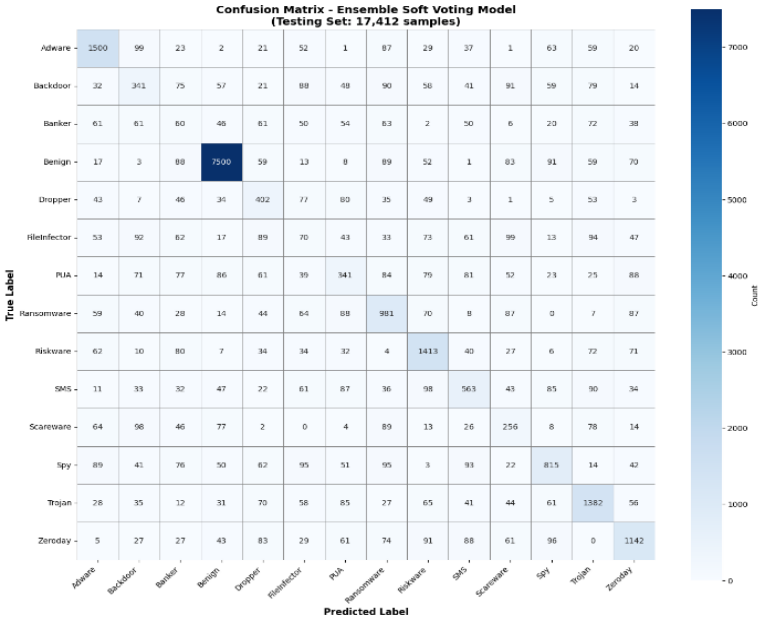
Grafik F1-Score per kelas pada Gambar 2 memperjelas variasi performa ini secara visual. Batang untuk kelas Benign dan sebagian besar kelas malware berada di atas 0.90, sementara beberapa kelas minoritas seperti Zeroday dan Banker tampak sedikit lebih rendah. Pola ini konsisten dengan analisis tekstual dan menegaskan bahwa tantangan utama terletak pada kelas dengan jumlah sampel sedikit dan karakteristik yang lebih sulit dibedakan.

3.3.2 Confusion Matrix dan Pola Kesalahan

Confusion matrix pada Gambar 3 menunjukkan bahwa mayoritas prediksi berada pada diagonal utama, yang berarti sebagian besar sampel berhasil diklasifikasikan dengan benar sesuai kelasnya. Kolom untuk kelas Benign bersih tanpa *false positive*, mengkonfirmasi bahwa tidak ada aplikasi normal yang keliru terdeteksi sebagai malware. Kesalahan klasifikasi yang terjadi tersebar tipis pada beberapa kelas minoritas, terutama Banker, FileInfector, dan Zeroday, yang sesekali tertukar dengan kelas malware lain yang memiliki pola fitur mirip. Tidak terlihat adanya bias sistematis

Building of Informatics, Technology and Science (BITS)
Volume 7, No 4, March 2026 Page: 2193−2204
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i4.8865

terhadap satu pasangan kelas tertentu; pola kesalahan relatif tersebar dan proporsinya kecil terhadap jumlah sampel masing-masing kelas.



Gambar 3. Confusion Matrix

3.4 Analisis Pengaruh Jumlah Fitur

Untuk memvalidasi pemilihan 150 fitur pada tahap *feature selection*, dilakukan eksperimen dengan hanya mengubah jumlah fitur hasil RFE. Jumlah fitur yang diuji adalah 25, 50, 75, 100, 125, 150, 175, dan 200 fitur.

Tabel 4. Pengaruh Jumlah Fitur terhadap F1-Macro

Jumlah fitur	F1-Macro	Time
25	0.8815	0h 12m 13s
50	0.8927	0h 24m 22s
75	0.9119	0h 36m 46s
100	0.9143	0h 47m 15s
125	0.9158	0h 50m 07s
150	0.9208	0h 53m 52s
175	0.9234	1h 18m 41s
200	0.9250	1h 34m 27s

Dari Tabel 4 terlihat bahwa penambahan jumlah fitur dari 25 hingga 150 secara konsisten meningkatkan F1-Macro, dengan waktu pelatihan yang juga ikut meningkat. Setelah sekitar 125 fitur, kenaikan F1-Macro mulai melambat: dari 0.9158 (125 fitur) menjadi 0.9208 (150 fitur). Penambahan fitur di atas 150 (175 dan 200 fitur) hanya memberikan kenaikan F1-Macro yang sangat kecil (0.9234 dan 0.9250), sementara waktu pelatihan meningkat cukup tajam, dari 0h 53m 52s (150 fitur) menjadi 1h 18m 41s dan 1h 34m 27s.

Dengan kata lain, konfigurasi 175 dan 200 fitur memang mampu memberikan F1-Macro sedikit lebih tinggi, tetapi tambahan performanya tidak sebanding dengan kenaikan waktu komputasi yang dibutuhkan. Sebaliknya, 150 fitur memberikan kombinasi yang lebih seimbang: F1-Macro sudah mendekati nilai tertinggi, tetapi dengan waktu pelatihan yang lebih rendah dibanding 175 dan 200 fitur. Berdasarkan *trade-off* ini, 150 fitur dipilih sebagai konfigurasi akhir pada program utama karena dianggap sebagai titik kompromi terbaik antara performa dan efisiensi komputasi.

3.5 Analisis Generalisasi dan Kesiapan Deployment

Perbandingan training vs testing menunjukkan gap F1-Macro sekitar 7.9% (0.9998 vs 0.9208), yang masih dalam rentang *acceptable* (<10%) untuk klasifikasi multi-kelas. Gap ini terkendali karena strategi *preprocessing* (*IQR capping, StandardScaler*, RFE, SMOTE) dan *ensemble soft voting* yang secara natural mengurangi *variance*. Model tidak mengalami *severe overfitting* dan siap untuk *deployment* pada skenario nyata.

Tabel 5. Perbandingan Performa Training vs Testing (Generalization Check)

Metrik	Training Set	Testing Set	Gap
F1-Macro	0.9998	0.9208	+0.0790 (7.9%)

Building of Informatics, Technology and Science (BITS)
Volume 7, No 4, March 2026 Page: 2193−2204
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i4.8865

Metrik	Training Set	Testing Set	Gap
Balanced Accuracy	~0.9980	0.9221	~0.0759 (7.59%)

Hasil penelitian menunjukkan efektivitas signifikan dari pendekatan *ensemble soft voting* untuk deteksi malware Android. Strategi *preprocessing multi-step* (IQR capping, *StandardScaler*, RFE, SMOTE) berhasil meningkatkan kualitas dataset dengan pengurangan dimensi 56% (341→150 fitur) sambil mempertahankan informasi penting. SMOTE meningkatkan training samples menjadi 85.750 dengan distribusi kelas lebih seimbang, memungkinkan model belajar pattern dari kelas minoritas dengan lebih baik. Performa *ensemble* (F1-Macro 0.9208) melampaui *individual models*, membuktikan *synergy* dari kombinasi tiga algoritma berbeda. Bobot berbasis F1-Macro (33.46% LightGBM, 30.99% Logistic Regression, 35.55% CatBoost) memastikan setiap model berkontribusi sesuai performa aktualnya. Integrasi *tree-based models* dengan linear model menciptakan *decision boundary* yang *robust* dan *stable*, dengan *soft voting inherently* lebih *resistant* terhadap *overfitting*.

Hasil evaluasi menunjukkan karakteristik ideal untuk *production deployment*: Accuracy 95.58%, *False Alarm Rate* 0.00% (*no false positive*), *True Positive Rate* 100% (semua malware terdeteksi), dan *Balanced Accuracy* 92.21%. Kombinasi FAR=0 dan TPR=1.0 adalah skenario ideal yang jarang dicapai dalam malware detection systems. Performa bervariasi antar kategori malware. Benign mencapai F1-Score sempurna (1.000), sementara kelas-kelas seperti Spy (0.962), Riskware (0.957), SMS (0.951) menunjukkan exceptional performance (>0.95). Zeroday menunjukkan performa terendah (0.839) likely karena karakteristik "*unknown*" dari novel malware. Mayoritas kelas mencapai F1 >0.85, menunjukkan model *robust* untuk berbagai malware categories. Gap 7.9% antara *training* dan *testing* F1-Macro menunjukkan *good generalization capability*. Model tidak mengalami *severe overfitting* meskipun training performance sangat tinggi (99.98%). Independensi *testing set* dari *preprocessing steps*, *combined* dengan SMOTE dan *ensemble regularization*, berkontribusi pada generalization yang baik. Performa testing tetap tinggi (92.08% F1-Macro) membuktikan model dapat *transfer learned patterns* ke *unseen data*. Limitasi penelitian mencakup

a. Zeroday *underperformance* memerlukan teknik *anomaly detection* atau *continuous model update*
b. *Small sample size* pada *minority classes* (Banker 72, FileInfector 75) membatasi *capability*
c. *Computational cost ensemble* adalah *acceptable trade-off* untuk *security-critical application*.

Penelitian ini berkontribusi dengan *demonstrating effectiveness* dari *hybrid approach* (*static+dynamic features*) dan *soft voting ensemble* untuk *Android malware detection* yang *suitable* untuk *production deployment*.

4. KESIMPULAN

Penelitian ini berhasil mengembangkan sistem deteksi malware Android berbasis *ensemble soft voting* yang mengintegrasikan LightGBM, Logistic Regression, dan CatBoost pada dataset CCCS-CIC-AndMal-2020, untuk mengatasi ketidakseimbangan kelas ekstrem, dimensionalitas tinggi, dan kebutuhan performa stabil pada kelas minoritas. Strategi preprocessing multi-tahap (IQR capping 40×, StandardScaler, RFE 150 fitur, SMOTE 30%) meningkatkan kualitas dataset dengan pengurangan dimensi 56% tanpa kehilangan informasi penting, sehingga ensemble dengan bobot berbasis F1-Macro mencapai *Accuracy* 95,58%, *Balanced Accuracy* 92,21%, F1-Macro 0,9208, TPR malware 100%, dan FAR 0,00%, dengan mayoritas kelas memiliki F1 di atas 0,85 dan performa terendah pada kelas Zeroday (0,839) yang mencerminkan tantangan deteksi novel malware. Hasil ini menunjukkan kemampuan generalisasi yang baik (gap F1-Macro *train-test* 7,9%) dan efektivitas integrasi fitur hybrid statis-dinamis dengan *ensemble learning*, namun tetap memiliki beberapa keterbatasan seperti evaluasi hanya dilakukan pada subset 58.037 sampel dari lebih 400.000 sampel sehingga generalisasi ke seluruh distribusi data asli masih terbatas, ukuran sampel testing yang kecil pada beberapa kelas minoritas (misalnya Banker dan FileInfector) mengurangi reliabilitas metrik per-kelas, serta *computational cost ensemble* meskipun masih dapat diterima untuk aplikasi security-critical tetap memerlukan optimisasi lebih lanjut, khususnya untuk implementasi pada perangkat dengan sumber daya terbatas. Oleh karena itu, penelitian lanjutan perlu mengeksplorasi evaluasi lintas dataset, skenario *time-based*, teknik *anomaly detection*, dan optimasi arsitektur untuk memperkuat *robustness* terhadap serangan *zero-day* dan *obfuscation*.

REFERENCES

[1] T. A. Aziz, Z. Sari, C. Sri, and K. Aditiya, "Klasifikasi Malware android dengan menggunakan metode XGBoost Algoritma," *REPOSITOR*, vol. 7, no. 1, pp. 103–110, 2025, doi: 10.22219/repositor.v7i1.36564.

[2] D. B. Ansori, J. Slamet, M. Z. Ghufron, M. A. R. Putra, and T. Ahmad, "Android Malware Classification Using Gain Ratio and Ensembled Machine Learning," *International Journal of Safety and Security Engineering*, vol. 14, no. 1, pp. 259–266, Feb. 2024, doi: 10.18280/ijsse.140126.

[3] S. K. Smmarwar, G. P. Gupta, and S. Kumar, "Android malware detection and identification frameworks by leveraging the machine and deep learning techniques: A comprehensive review," *Telematics and Informatics Reports*, vol. 14, Jun. 2024, doi: 10.1016/j.teler.2024.100130.

[4] A. Alhogail and R. A. Alharbi, "Effective ML-Based Android Malware Detection and Categorization," *Electronics (Switzerland)*, vol. 14, no. 8, Apr. 2025, doi: 10.3390/electronics14081486.

[5] 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.

Building of Informatics, Technology and Science (BITS)
Volume 7, No 4, March 2026 Page: 2193−2204
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v7i4.8865

[6] L. Kaur, C. Singh Saroa, and J. Singh, "A review of Static Analysis of Android Malware," *IOSR Journal of Computer Engineering (IOSR-JCE)*, vol. 25, no. 6, pp. 72–78, Dec. 2023, doi: 10.9790/0661-2506027278.
[7] K. Khalda and D. K. Wibowo, "Malware Behavior Analysis Using Static and Dynamic Analysis Approaches," *Jurnal Sains, Nalar, dan Aplikasi Teknologi Informasi*, vol. 4, no. 1, pp. 1–8, Jan. 2025, doi: 10.20885/snati.v4.i1.1.
[8] P. Sumalatha and G. S. Mahalakshmi, "Machine Learning Based Ensemble Classifier For Android Malware Detection," *International Journal of Computer Networks and Communications*, vol. 15, no. 4, pp. 111–122, Jul. 2023, doi: 10.5121/ijcnc.2023.15407.
[9] J. Abawajy, A. Darem, and A. A. Alhashmi, "Feature subset selection for malware detection in smart iot platforms," *Sensors (Switzerland)*, vol. 21, no. 4, pp. 1–19, Feb. 2021, doi: 10.3390/s21041374.
[10] S. Aurangzeb and M. Aleem, "Evaluation and classification of obfuscated Android malware through deep learning using ensemble voting mechanism," *Sci Rep*, vol. 13, no. 1, Dec. 2023, doi: 10.1038/s41598-023-30028-w.
[11] A. Al-Sraratee and A. Al-Azawei, "Classifying Android Malware Categories Based On Dynamic Features: An Integration Of Feature Reduction And Selection Techniques," *Kufa Journal of Engineering*, vol. 16, no. 2, pp. 96–118, Apr. 2025, doi: 10.30572/2018/KJE/160206.
[12] N Anintha Devi, C Karthika, V Pradeepa, and C Sharmila, "Shap Based -Android Malware Detection Using Ensemble Learning," *International Research Journal on Advanced Science Hub*, vol. 7, no. 07, pp. 673–680, Jul. 2025, doi: 10.47392/irjash.2025.077.
[13] R. R. Sani, F. A. Rafrastara, and W. Ghozi, "Integrating Ensemble Learning and Information Gain for Malware Detection based on Static and Dynamic Features," *Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control*, Jan. 2025, doi: 10.22219/kinetik.v10i1.2051.
[14] M. Azwar, L. Widyawati, R. Azhar, K. Kartarina, T. Tanwir, and A. S. Anas, "Deteksi Malware pada Perangkat Android Menggunakan Ensemble Learning," *JTIM : Jurnal Teknologi Informasi dan Multimedia*, vol. 7, no. 3, pp. 408–419, Jun. 2025, doi: 10.35746/jtim.v7i3.573.
[15] S. S. Suhaila and K. S. Krishnan, "A novel end-to-end ensemble framework for enhanced android malware detection accuracy," *Egyptian Informatics Journal*, vol. 32, Dec. 2025, doi: 10.1016/j.eij.2025.100827.
[16] V. Kouliaridis and G. Kambourakis, "A comprehensive survey on machine learning techniques for android malware detection," *Information (Switzerland)*, vol. 12, no. 5, 2021, doi: 10.3390/info12050185.
[17] S. Widodo and F. S. Utomo, "A Comprehensive Evaluation of CatBoost and LightGBM Algorithms for Honorarium Prediction on Categorical Datasets with Class Imbalance," *JUITA: Jurnal Informatika*, vol. 13, no. 3, pp. 359–370, 2025, doi: 10.30595/juita.v13i3.27363.
[18] V. Jyothsna, K. P. Dasari, S. Inuguru, V. B. R. Gowni, J. T. R. Kudumula, and K. Srilakshmi, "Unified Approach for Android Malware Detection: Feature Combination and Ensemble Classifier," *Proceedings of the International Conference on Computational Innovations and Emerging Trends*, pp. 485–495, 2024, doi: 10.2991/978-94-6463-471-6_47.
[19] E. Y. Chaymae and C. Khalid, "Android Malware Detection Through CNN Ensemble Learning on Grayscale Images," *International Journal of Advanced Computer Science and Applications*, vol. 16, no. 1, p. 2025, 2025, doi: 10.14569/IJACSA.2025.01601116.
[20] K. Aziz, A. Wahyudi, and I. Palupi, "Mental Health Sentiment Analysis on Twitter using Ensemble Learning Algorithm," *Technology and Science (BITS)*, vol. 7, no. 2, pp. 1017–1027, 2025, doi: 10.47065/bits.v7i2.7763.