Deteksi Adware Berbasis Machine Learning Menggunakan Cluster-Aware Stacking Ensemble

Adware Detection Based on Machine Learning Using Cluster-Aware Stacking Ensemble

Karfindo*[1], **Muhammad Diponegoro**[2], **Yusril Eka Mahendra**[3], **Mohamad Arifin**[4]
[1,2,3]*Teknik Elektro, Teknik Informatika, Politeknik Negeri Pontianak, Pontianak, Indonesia*
[4]*Teknik Elektro, Teknologi Rekayasa Sistem Elektronika, Politeknik Negeri Pontianak, Pontianak, Indonesia*
E-mail : karfindo@polnep.ac.id[1], muhammaddiponegoro@gmail.com[2], yusrilekamahendra@polnep.ac.id[3], mohamad.arifin@polnep.ac.id[4]*
Received 16 January 2026; Revised 20 February 2026; Accepted 21 February 2026

**Corresponding author*

Abstrak - Adware pada Android mengganggu pengalaman pengguna serta meningkatkan risiko privasi dan keamanan. Sebagian studi deteksi adware masih dievaluasi dengan pembagian data acak, sehingga kurang merepresentasikan pergeseran distribusi (out-of-distribution/OOD). Penelitian ini mengevaluasi Cluster-Aware Stacking (CAS) dengan validasi silang berbasis klaster (C_ClusterOOD) agar estimasi kinerja lebih mendekati kondisi penerapan. Dataset CICMalDroid 2020 difilter menjadi dua kelas, adware dan benign, menghasilkan 3.045 sampel dengan 470 fitur numerik (1.253 adware; 1.792 benign). Pembagian hold-out 80:20 menggunakan skema anti-kebocoran berbasis grup. Model dasar meliputi Logistic Regression, Random Forest, Gradient Boosting, dan XGBoost, dengan Soft Voting sebagai pembanding. Pada evaluasi berbasis klaster, CAS memberikan performa yang kompetitif pada metrik F1 adware dan MCC. Pada uji hold-out, konfigurasi terbaik mencapai akurasi 0,977, F1 adware 0,972, dan MCC 0,952, dengan ROC-AUC mendekati 1,0. Hasil ini menunjukkan bahwa evaluasi berbasis klaster membantu memilih model yang lebih robust untuk deteksi adware Android.

Kata Kunci – *Android adware, machine learning, ensemble learning, stacking ensemble, out-of-distribution*

Abstract – *Android adware disrupts user experience and increases privacy and security risks. Many adware-detection studies are still evaluated using random data splits, which do not adequately reflect distribution shift (out-of-distribution/OOD). This study evaluates Cluster-Aware Stacking (CAS) with cluster-based cross-validation (C_ClusterOOD) to obtain performance estimates that better match real-world deployment. The CICMalDroid 2020 dataset was filtered into two classes—adware and benign—yielding 3,045 samples with 470 numerical features (1,253 adware; 1,792 benign). An 80:20 hold-out split was applied using a group-based anti-leakage scheme. Base learners include Logistic Regression, Random Forest, Gradient Boosting, and XGBoost, with Soft Voting as a baseline. Under cluster-based evaluation, CAS achieves competitive performance in terms of adware F1-score and MCC. On the hold-out test set, the best configuration attains 0.977 accuracy, 0.972 adware F1-score, and 0.952 MCC, with ROC-AUC close to 1.0. These results indicate that cluster-based evaluation helps select more robust models for Android adware detection.*

Keywords – *Android adware, machine learning, ensemble learning, stacking ensemble, out-of-distribution*

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15641

1. PENDAHULUAN

Pertumbuhan ekosistem aplikasi Android telah menjadi pendorong utama transformasi layanan digital global. Dengan lebih dari dua miliar pengguna aktif, Android mendominasi pasar sistem operasi mobile dan secara inheren menjadi target utama serangan siber berbasis aplikasi [1], [2]. Salah satu ancaman paling dominan dalam ekosistem ini adalah *adware*, yaitu perangkat lunak yang mengeksploitasi mekanisme periklanan untuk memperoleh keuntungan finansial dengan cara yang agresif dan sering kali tanpa persetujuan pengguna [3], [4].

Berbeda dengan malware konvensional seperti trojan atau ransomware, adware sering diklasifikasikan sebagai *greyware* atau *potentially unwanted application* (PUA). Namun, berbagai penelitian menunjukkan bahwa adware modern tidak hanya menampilkan iklan, tetapi juga melakukan *ad fraud*, pelacakan perilaku pengguna, kebocoran data pribadi, hingga pemuatan payload berbahaya tambahan [5], [6]. Karakteristik ini menempatkan adware sebagai ancaman serius terhadap privasi dan keamanan pengguna Android.

Studi longitudinal menunjukkan bahwa adware mengalami evolusi berkelanjutan selama lebih dari satu dekade. Penelitian lain menunjukkan bahwa adware berevolusi dari sekadar penyalahgunaan *advertising library* menjadi sistem kompleks yang memanfaatkan *dynamic code loading*, teknik *obfuscation* tingkat lanjut, serta eksploitasi jaringan iklan yang sah [7]. Evolusi ini menyebabkan batas antara aplikasi *benign* dan adware semakin kabur, sehingga menyulitkan deteksi berbasis aturan atau tanda tangan statis.

Selain kompleksitas perilaku, distribusi adware juga bersifat sangat heterogen. Penelitian berbasis dataset skala besar menunjukkan bahwa karakteristik adware sangat dipengaruhi oleh wilayah geografis, waktu rilis, serta ekosistem distribusi aplikasi [8], [9]. Heterogenitas ini memperbesar risiko kegagalan model deteksi ketika diterapkan pada data dengan distribusi yang berbeda (*out-of-distribution*).

Untuk mengatasi tantangan tersebut, berbagai pendekatan *machine learning* telah diusulkan. Model klasik seperti Random Forest, Gradient Boosting, dan XGBoost terbukti efektif dalam mendeteksi adware berbasis fitur statis maupun dinamis [10], [11]. Pendekatan berbasis *network traffic flow* bahkan mampu mencapai akurasi tinggi dengan overhead komputasi rendah, sehingga dinilai cocok untuk lingkungan perangkat mobile dengan keterbatasan sumber daya [11].

Pendekatan lain memanfaatkan analisis struktural aplikasi Android melalui *Control Flow Graph* (CFG). Penelitian lain menunjukkan bahwa pola CFG yang diekstraksi menggunakan kerangka kerja Soot mampu merepresentasikan perilaku adware secara efektif tanpa memerlukan analisis dinamis yang mahal [12]. Meskipun demikian, pendekatan ini masih rentan terhadap teknik *obfuscation* tingkat lanjut yang banyak digunakan oleh adware modern.

Seiring berkembangnya teknik representasi data, metode berbasis *computer vision* dan *transfer learning* mulai digunakan dalam deteksi adware. Penelitian lain mencoba mengonversi file DEX menjadi citra grayscale dan memanfaatkan jaringan saraf konvolusional pralatih seperti MobileNet dan ResNet, yang menunjukkan bahwa representasi visual mampu menangkap karakteristik adware yang tidak eksplisit pada fitur tradisional [13].

Pendekatan *deep learning* murni juga menunjukkan potensi besar. MadDroid membuktikan bahwa CNN berbasis izin aplikasi mampu mendeteksi adware secara efektif [6]. Namun, penelitian-penelitian tersebut juga mencatat bahwa model *deep learning* umumnya membutuhkan sumber daya komputasi besar dan belum tentu *robust* terhadap pergeseran distribusi data yang sering terjadi dalam domain keamanan Android.

Masalah krusial lain dalam penelitian deteksi adware adalah skema evaluasi model. Sebagian besar penelitian terdahulu masih menggunakan pembagian data acak (*random split*), yang berisiko menghasilkan estimasi performa yang terlalu optimistis akibat kemiripan atau duplikasi sampel lintas subset data [14]. Fenomena ini dikenal sebagai *data leakage* dan dapat mengaburkan kemampuan generalisasi model pada data yang benar-benar baru.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15641

Dalam konteks keamanan Android, pergeseran distribusi (*out-of-distribution*) merupakan kondisi yang lazim karena keluarga adware terus berkembang dan bermunculan. Studi terbaru menunjukkan bahwa model dengan performa tinggi pada evaluasi konvensional dapat mengalami penurunan performa yang signifikan ketika diuji pada generasi adware yang berbeda [7].

Untuk meningkatkan stabilitas prediksi, teknik *ensemble learning* banyak diadopsi dalam deteksi malware dan adware. Metode voting terbukti mampu mengurangi varians dan meningkatkan performa rata-rata model [15]. Penelitian sebelumnya menunjukkan bahwa *soft voting ensemble* mampu meningkatkan akurasi deteksi malware Android dibandingkan model tunggal, namun masih bergantung pada skema evaluasi konvensional berbasis *random split* [16].

Pendekatan *stacking ensemble* menawarkan fleksibilitas yang lebih tinggi melalui penggunaan *meta-learner* untuk menggabungkan prediksi beberapa model dasar. Namun, efektivitas stacking sangat bergantung pada kualitas prediksi *out-of-fold*. Tanpa kontrol distribusi data yang memadai, stacking berisiko memperkuat bias pada *meta-learner*, terutama pada data keamanan yang bersifat tidak stasioner [17].

Berdasarkan keterbatasan tersebut, penelitian ini mengusulkan *Cluster-Aware Stacking* (CAS), yaitu pendekatan *stacking ensemble* yang mengintegrasikan validasi silang berbasis klaster untuk mensimulasikan kondisi *out-of-distribution* secara eksplisit. Dengan memanfaatkan klaster sebagai proksi variasi distribusi data, CAS menghasilkan prediksi *out-of-fold* yang lebih realistis dan meminimalkan risiko *data leakage*.

Eksperimen dilakukan menggunakan dataset CICMalDroid 2020, yang prinsip konstruksi, kategorisasi malware, serta metodologi pelabelannya dijelaskan secara rinci dalam [18], [19], dan telah digunakan secara luas dalam penelitian deteksi malware Android [10]. Hasil evaluasi menunjukkan bahwa CAS secara konsisten mengungguli model dasar dan ensemble konvensional, khususnya pada metrik yang sensitif terhadap ketidakseimbangan kelas seperti F1-score dan Matthews Correlation Coefficient (MCC). Temuan ini menegaskan bahwa evaluasi berbasis distribusi merupakan aspek krusial dalam pengembangan sistem deteksi adware Android yang andal dan berkelanjutan.

2. METODE PENELITIAN

Bagian ini menjelaskan tahapan metodologi yang digunakan untuk membangun dan mengevaluasi model deteksi adware. Metodologi dirancang untuk menghasilkan evaluasi yang andal dan mendekati kondisi penerapan nyata, khususnya ketika terjadi variasi pola adware dan pergeseran distribusi data.

Secara umum, penelitian ini mengombinasikan teknik *machine learning* dan *ensemble learning* dengan skema validasi yang mempertimbangkan potensi *out-of-distribution* (OOD). Data diproses melalui beberapa tahap utama: (1) pra-pemrosesan dan pembersihan data (termasuk penanganan nilai hilang serta deduplikasi untuk mengurangi risiko kebocoran), (2) pembagian data *hold-out* dengan mekanisme pencegahan *information leakage*, (3) pembentukan klaster secara *unsupervised* (mis. K-Means) pada ruang fitur untuk merepresentasikan variasi distribusi data, dan (4) pelatihan model dasar serta *stacking ensemble* untuk klasifikasi.

Berbeda dari pendekatan konvensional yang mengandalkan pembagian data acak, penelitian ini menerapkan validasi berbasis klaster (*ClusterOOD*), yaitu validasi di mana data validasi berasal dari klaster yang tidak digunakan selama pelatihan (klaster diperlakukan sebagai *group*). Dengan demikian, evaluasi memaksa perbedaan distribusi antara data latih dan data validasi, sehingga estimasi performa menjadi lebih realistis dan lebih robust terhadap pergeseran distribusi.

Tahapan metodologi disusun secara sistematis dan dijelaskan berurutan pada subbagian berikutnya, dimulai dari deskripsi dataset dan pra-pemrosesan, dilanjutkan skema pembagian data dan pembentukan klaster, hingga proses pelatihan model dan evaluasi performa.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15641

2.1. Dataset dan Pra-pemrosesan Data

Dataset yang digunakan dalam penelitian ini adalah CICMalDroid 2020, yaitu dataset malware Android yang dikembangkan oleh *Canadian Insitute for Cybersecurity* (CIC). Dataset ini banyak digunakan dalam penelitian keamanan siber karena menyediakan fitur aplikasi Android (statis dan dinamis) dengan label malware yang beragam.

Pada penelitian ini, dataset difilter sehingga hanya mencakup dua kelas, yaitu adware dan benign, untuk memfokuskan studi pada deteksi adware yang umum ditemukan dan memiliki karakteristik bervariasi. Label pada dataset dipetakan ke format biner dengan definisi yang konsisten: 1 untuk adware dan 0 untuk benign.

Seluruh fitur non-numerik dihapus untuk memastikan kompatibilitas dengan algoritma pembelajaran mesin yang digunakan. Setelah pemfilteran dan pembersihan awal, dilakukan deduplikasi berdasarkan seluruh fitur numerik untuk menghapus sampel duplikat yang berpotensi menyebabkan *information leakage* pada evaluasi. Hasil akhir setelah deduplikasi menghasilkan 3.045 sampel dengan 470 fitur numerik, terdiri dari 1.253 sampel adware dan 1.792 sampel benign.

Selain itu, nilai tak hingga (*infinite values*) pada beberapa fitur digantikan dengan nilai kosong (NaN) dan kemudian ditangani menggunakan imputasi median. Untuk langkah yang sensitif terhadap skala fitur, terutama pembentukan klaster (K-Means) dan model berbasis regresi, dilakukan standardization. Agar tidak terjadi kebocoran informasi, parameter imputasi dan standardization di-fit hanya pada data pelatihan (pada setiap skema pelatihan/validasi), lalu diterapkan pada data validasi dan data uji.

2.2. Skema Pembagian Data dan Pencegahan Kebocoran

Pembagian data dilakukan menggunakan skema hold-out dengan rasio 80% untuk data pelatihan dan 20% untuk data pengujian. Untuk meminimalkan kebocoran informasi (*data leakage*) antara data pelatihan dan data pengujian, digunakan GroupShuffleSplit dengan *group identifier* yang dibentuk dari fingerprint/hash seluruh fitur numerik setiap sampel. Dengan demikian, sampel yang identik (duplikat)—dan sampel yang berpotensi sangat mirip akibat proses ekstraksi fitur—cenderung tetap berada dalam subset yang sama, sehingga mengurangi risiko kebocoran informasi lintas subset.

Skema ini membantu memastikan bahwa evaluasi pada data uji dilakukan pada data yang lebih independen dari data pelatihan, sehingga hasil pengujian akhir lebih representatif terhadap skenario penerapan nyata. Setelah hold-out split ditetapkan, seluruh proses pemilihan model dan penalaan parameter dilakukan pada data pelatihan (melalui skema validasi yang dijelaskan pada subbagian berikut), sedangkan data uji digunakan hanya untuk evaluasi final.

2.3. Pembentukan Klaster untuk Validasi Out-of-Distribution

Untuk mensimulasikan kondisi out-of-distribution (OOD) secara terkontrol, subset data pelatihan dikelompokkan menggunakan algoritma K-Means. Sebelum proses clustering, fitur dinormalisasi menggunakan StandardScaler; parameter penskalaan dihitung (*fit*) hanya pada data pelatihan, kemudian diterapkan (*transform*) pada data pelatihan untuk mencegah kebocoran informasi.

Jumlah klaster K dievaluasi melalui analisis sensitivitas pada beberapa nilai, yaitu $K \in \{5, 10, 20\}$, guna menilai stabilitas performa model terhadap granularitas pembentukan sub-populasi. Setiap klaster diperlakukan sebagai representasi sub-distribusi data (yakni kelompok sampel dengan pola fitur yang serupa, mencakup benign dan adware) yang berpotensi mencerminkan variasi karakteristik aplikasi serta pergeseran distribusi yang mungkin terjadi pada skenario penerapan.

Selanjutnya diterapkan skema validasi silang ClusterOOD menggunakan GroupKFold, dengan label klaster sebagai *group*. Pada setiap fold, satu klaster digunakan sebagai data validasi, sedangkan klaster lainnya digunakan untuk pelatihan. Dengan demikian, model dievaluasi pada kelompok sampel yang tidak digunakan pada pelatihan fold tersebut, sehingga menghasilkan estimasi performa yang lebih robust terhadap pergeseran distribusi berbasis klaster. Nilai K

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15641

terbaik kemudian dipilih berdasarkan metrik evaluasi yang ditetapkan (mis. kombinasi F1-Score dan MCC, atau skema multi-objective), sebelum dilakukan pengujian final pada data uji hold-out.

2.4. Model Klasifikasi Dasar

Penelitian ini menggunakan beberapa algoritma klasifikasi sebagai model dasar (base learners), yaitu Random Forest, Gradient Boosting, Logistic Regression, dan XGBoost. Pemilihan keempat model ini didasarkan pada kemampuan yang saling melengkapi dalam menangani data berdimensi tinggi, pola non-linear, serta kebutuhan menghasilkan probabilitas prediksi yang stabil untuk digunakan pada tahap ensemble (soft voting dan stacking).

1. Forest, yang mampu menangani data berdimensi tinggi dan ketidakseimbangan kelasRandom Forest digunakan karena robust pada fitur berdimensi tinggi dan relatif tahan terhadap noise; ketidakseimbangan kelas ditangani melalui pengaturan class_weight = balanced.
2. Gradient Boosting digunakan untuk menangkap hubungan non-linear melalui pembelajaran berurutan (additive model) yang memperbaiki kesalahan prediksi pada iterasi berikutnya.
3. Logistic Regression digunakan sebagai pembanding model linear yang efisien dan interpretabel; ketidakseimbangan kelas ditangani melalui class_weight = balanced.
4. XGBoost digunakan karena performanya kuat pada data tabular serta mendukung optimasi komputasi; ketidakseimbangan kelas ditangani menggunakan scale_pos_weight yang disesuaikan dengan rasio kelas pada data pelatihan.

Seluruh model dasar dilatih menggunakan parameter yang ditetapkan secara eksplisit dan dievaluasi menggunakan skema validasi berbasis klaster (ClusterOOD) untuk memastikan perbandingan yang adil pada kondisi pergeseran distribusi. Untuk menjaga reprodusibilitas, parameter *random seed* digunakan secara konsisten pada komponen yang bersifat stokastik.

2.5. Stacking Ensemble Berbasis Out-of-Fold

Untuk meningkatkan performa klasifikasi, penelitian ini menerapkan stacking ensemble dengan membangun meta-fitur berbasis prediksi out-of-fold (OOF). Berbeda dengan stacking konvensional yang umumnya menggunakan pembagian acak, penelitian ini menghasilkan OOF dengan skema validasi berbasis klaster (ClusterOOD) sehingga pemisahan pelatihan–validasi tetap merepresentasikan skenario *out-of-distribution.*

Pada setiap fold validasi (di mana satu klaster ditahan sebagai data validasi), prosedurnya adalah sebagai berikut:

1. Model dasar dilatih menggunakan data dari klaster-klaster pelatihan (seluruh klaster selain klaster yang ditahan).
2. Model dasar menghasilkan prediksi probabilitas untuk sampel pada klaster yang ditahan.
3. Probabilitas prediksi dari seluruh model dasar kemudian digabungkan sebagai meta-fitur (level-1 features) untuk sampel pada klaster tersebut.
4. Setelah seluruh fold selesai, seluruh meta-fitur OOF yang terkumpul membentuk dataset level-1 yang digunakan untuk melatih meta-learner.

Pendekatan Pendekatan ini memastikan bahwa meta-fitur untuk setiap sampel selalu dihasilkan oleh model yang tidak pernah dilatih pada klaster asal sampel tersebut, sehingga meminimalkan risiko kebocoran informasi lintas distribusi dan memberikan estimasi performa yang lebih realistis terhadap pergeseran distribusi.

2.6. Kalibrasi Probabilitas dan Meta-Learner

Sebelum digunakan sebagai meta-fitur, probabilitas prediksi dari model dasar dikalibrasi menggunakan metode kalibrasi probabilitas sesuai konfigurasi eksperimen (misalnya sigmoid atau isotonic). Kalibrasi bertujuan untuk meningkatkan reliabilitas estimasi probabilitas, menyeragamkan skala probabilitas antar model dasar, serta mengurangi kecenderungan prediksi yang terlalu optimistis.

Untuk mencegah kebocoran informasi, proses kalibrasi dilakukan di dalam skema validasi: model kalibrasi dipelajari hanya dari data pelatihan pada setiap fold dan kemudian

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15641

diterapkan pada data validasi (klaster yang ditahan). Probabilitas yang telah dikalibrasi tersebut kemudian digunakan sebagai meta-fitur dalam stacking.

Sebagai meta-learner digunakan Logistic Regression dengan penyeimbangan kelas (class-weight balancing) yang dilatih pada meta-fitur hasil prediksi OOF. Pemilihan model linear pada level meta bertujuan untuk menjaga stabilitas pembelajaran dan mengurangi risiko overfitting terhadap meta-fitur.

Dalam pengambilan keputusan kelas, probabilitas keluaran meta-learner dikonversi menjadi label menggunakan ambang keputusan (threshold). Nilai threshold ditentukan melalui pencarian pada data validasi (dan/atau secara adaptif per-klaster) sesuai objektif evaluasi (misalnya memaksimalkan kombinasi F1 dan MCC, serta/atau memenuhi batas minimum recall untuk kelas adware).

2.7. Metrik Evaluasi

Evaluasi performa model dilakukan menggunakan beberapa metrik yang relevan terhadap permasalahan deteksi adware dan ketidakseimbangan kelas, yaitu:

1. Accuracy,
2. Balanced Accuracy,
3. Precision dan Recall untuk kelas adware,
4. F1-score kelas adware,
5. Matthews Correlation Coefficient (MCC),
6. Area Under Curve untuk ROC (ROC-AUC),
7. Area Under Curve untuk Precision–Recall (PR-AUC).

Penggunaan metrik yang beragam bertujuan untuk memberikan gambaran performa model yang lebih komprehensif, khususnya pada konteks deteksi perangkat lunak berbahaya. Untuk pemilihan konfigurasi terbaik, metrik utama yang digunakan adalah F1-score adware dan MCC (atau skor gabungan yang didefinisikan pada eksperimen), sedangkan metrik lainnya digunakan sebagai pendukung interpretasi perform

2.8. Alur Penelitian

Secara ringkas, alur penelitian ini dapat dirangkum sebagai berikut:

1. Pra-pemrosesan dan pembersihan data,
2. Pembagian data dengan skema anti-kebocoran,
3. Pembentukan klaster pada data pelatihan,
4. Validasi silang berbasis klaster (ClusterOOD/GroupKFold) untuk mensimulasikan kondisi out-of-distribution,
5. Pelatihan model dasar, pembentukan meta-fitur OOF, kalibrasi probabilitas, serta pelatihan meta-learner stacking
6. Analisis sensitivitas jumlah klaster (misalnya $K \in \{5, 10, 20\}$) dan identifikasi klaster sulit.
7. Perbandingan dengan baseline validasi acak (KFold/StratifiedKFold).
8. Evaluasi akhir pada data uji hold-out.

3. HASIL DAN PEMBAHASAN

Bagian ini menyajikan hasil eksperimen dan pembahasan yang disusun mengikuti alur metodologi penelitian. Evaluasi dimulai dengan deskripsi dataset setelah pra-pemrosesan, dilanjutkan dengan analisis performa model melalui validasi silang berbasis klaster (ClusterOOD) untuk mensimulasikan skenario out-of-distribution, dan diakhiri dengan pengujian pada data uji hold-out yang sepenuhnya terpisah dari data pelatihan.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15641

3.1. Deskripsi Data Hasil Pra-pemrosesan

Setelah melalui tahapan pra-pemrosesan dan pembersihan data, dataset akhir yang digunakan dalam penelitian ini terdiri dari 3.045 sampel dengan 470 fitur numerik. Dataset mencakup dua kelas, yaitu 1.253 sampel adware dan 1.792 sampel benign, dengan rasio ketidakseimbangan benign/adware sebesar 1,43. Deduplication dilakukan untuk menghapus sampel duplikat yang berpotensi menyebabkan kebocoran informasi (3.048 → 3.045). Selanjutnya, dataset dibagi menggunakan skema hold-out dengan rasio 80:20, menghasilkan 2.436 sampel data pelatihan dan 609 sampel data pengujian, dengan mekanisme anti-leakage berbasis grup sehingga data uji sepenuhnya independen dari data pelatihan. Ringkasan distribusi kelas setelah pra-pemrosesan disajikan pada Tabel 1.

Tabel 1. Distribusi Kelas Dataset Setelah Pra-Pemrosesan

Subset	Jumlah Sampel (N)	Benign (0)	Adware (1)	Persentase Benign (%)	Persentase Adware (%)	Rasio Ketidakseimbangan (Benign/Adware)
Semua	3045	1792	1253	58.85	41.15	1.43
Train	2436	1428	1008	58.62	41.38	1.42
Test	609	364	245	59.77	40.23	1.49

Tabel 1 menunjukkan bahwa distribusi kelas pada data pelatihan dan data pengujian relatif konsisten dengan dataset keseluruhan. Kondisi ini penting untuk memastikan bahwa proses evaluasi model dilakukan secara adil dan representatif.

3.2. Hasil Validasi Silang Berbasis Klaster (C_ClusterOOD)

Evaluasi awal performa model dilakukan pada data pelatihan menggunakan skema validasi silang berbasis klaster (C_ClusterOOD). Dalam skema ini, data pelatihan terlebih dahulu dikelompokkan ke dalam beberapa klaster berdasarkan karakteristik fiturnya. Selanjutnya, setiap klaster secara bergantian dijadikan data validasi, sedangkan klaster lainnya digunakan sebagai data pelatihan. Pendekatan ini dirancang untuk mensimulasikan kondisi *out-of-distribution* (OOD), karena distribusi data validasi berasal dari kelompok (klaster) yang tidak disertakan saat pelatihan pada fold tersebut.

Model yang dievaluasi mencakup beberapa model dasar, yaitu Logistic Regression, Random Forest, Gradient Boosting, dan XGBoost, serta model ensemble konvensional Soft Voting. Selain itu, penelitian ini mengevaluasi pendekatan CAS yang membangun meta-fitur dari prediksi probabilitas *out-of-fold* (OOF) dengan tetap mempertahankan struktur klaster, sehingga proses stacking tidak mengandung kebocoran informasi antar-distribusi. Seluruh model dievaluasi menggunakan skema validasi yang sama untuk memastikan perbandingan yang adil.

Ringkasan hasil validasi silang berbasis klaster disajikan pada Tabel 2. Secara umum, seluruh model menunjukkan nilai ROC-AUC dan PR-AUC yang tinggi, yang mengindikasikan bahwa fitur yang digunakan memiliki daya pisah yang kuat antara kelas adware dan benign. Namun, perbedaan performa menjadi lebih terlihat pada metrik yang lebih sensitif terhadap ketidakseimbangan kelas, khususnya F1-score untuk kelas adware dan Matthews Correlation Coefficient (MCC).

Berdasarkan Tabel 2, Soft Voting memberikan nilai *balanced accuracy* sedikit lebih tinggi, sedangkan CAS menghasilkan nilai F1-score adware dan MCC tertinggi secara rata-rata. Mengingat tujuan utama penelitian ini adalah deteksi adware yang seimbang (meminimalkan *false negative* tanpa meningkatkan *false positive* secara berlebihan), F1-score adware dan MCC diprioritaskan sebagai dasar pemilihan kandidat model terbaik. Oleh karena itu, CAS dipilih untuk dievaluasi lebih lanjut pada data uji hold-out.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15641

Tabel 2. Performa Model pada Validasi Silang Berbasis Klaster (C_ClusterOOD)

Model	Balanced Acc (mean ± std)	F1_Adware (mean ± std)	MCC (mean ± std)	ROC_AUC (mean ± std)	PR_AUC (mean ± std)
CAS	0.9732 ± 0.0403	0.5735 ± 0.5245	0.5541 ± 0.5099	0.9938 ± 0.0098	0.9926 ± 0.0088
SoftVoting	0.9748 ± 0.0340	0.5704 ± 0.5212	0.5433 ± 0.4977	0.9933 ± 0.0107	0.9921 ± 0.0100
XGB	0.9744 ± 0.0326	0.5688 ± 0.5197	0.5421 ± 0.4962	0.9925 ± 0.0094	0.9867 ± 0.0115
GB	0.9543 ± 0.0504	0.5571 ± 0.5094	0.5124 ± 0.4708	0.9856 ± 0.0173	0.9807 ± 0.0168
RF	0.9583 ± 0.0622	0.5490 ± 0.5044	0.5199 ± 0.4812	0.9919 ± 0.0113	0.9887 ± 0.0107
LR	0.9397 ± 0.0701	0.5328 ± 0.4896	0.4854 ± 0.4491	0.9695 ± 0.0346	0.9091 ± 0.0903

Tabel 2 menyajikan ringkasan performa berbagai model klasifikasi pada skema validasi silang berbasis klaster (C_ClusterOOD). Nilai yang ditampilkan merupakan rata-rata dan simpangan baku dari setiap metrik evaluasi pada seluruh fold validasi, sehingga menggambarkan performa model pada skenario *out-of-distribution* antar klaster.

Berdasarkan hasil tersebut, metode Soft Voting menunjukkan nilai *balanced accuracy* sedikit lebih tinggi dibandingkan model lainnya. Namun demikian, pendekatan CAS menghasilkan nilai F1-score kelas adware dan Matthews Correlation Coefficient (MCC) tertinggi secara rata-rata. Mengingat fokus penelitian ini adalah deteksi adware pada dataset yang tidak sepenuhnya seimbang, metrik F1-score adware dan MCC diprioritaskan sebagai dasar evaluasi karena lebih sensitif terhadap keseimbangan performa antar kelas. Oleh karena itu, model CAS dipilih sebagai kandidat terbaik untuk evaluasi lanjutan pada data uji hold-out.

3.3. Analisis Sensitivitas Parameter K

Setelah Uji sensitivitas dilakukan terhadap parameter K (jumlah klaster) dengan mengevaluasi nilai $K \in \{5, 10, 20\}$. Hasil menunjukkan bahwa pemilihan K memiliki dampak signifikan terhadap performa dan stabilitas model.

Tabel 3. Ringkasan Uji Sensitivitas Parameter K

K	Model	F1 (mean±std)	MCC (mean±std)	Recall (mean±std)
CAS	0.9732 ± 0.0403	0.5735 ± 0.5245	0.5421±0.4962	0.5768±0.5266
SoftVoting	0.9748 ± 0.0340	0.5704 ± 0.5212	0.5147±0.4870	0.5980±0.5459
XGB	0.9744 ± 0.0326	0.5688 ± 0.5197	0.5078±0.4453	0.7998±0.3218
GB	0.9543 ± 0.0504	0.5571 ± 0.5094	0.5075±0.4409	0.8624±0.3108
RF	0.9583 ± 0.0622	0.5490 ± 0.5044	0.3903±0.4481	0.6286±0.4470
LR	0.9397 ± 0.0701	0.5328 ± 0.4896	0.3952±0.4519	0.6544±0.4541

K=10 memberikan trade-off optimal dengan F1-score 0.834 ± 0.301, dibandingkan K=5 (0.560 ± 0.515) dan K=20 (0.625 ± 0.431). K=5 menghasilkan variance yang sangat tinggi (std ≈ 0.5) karena clustering menghasilkan beberapa klaster dengan ukuran sangat kecil (1-3 sampel), menyebabkan hasil ekstrem yang meningkatkan variance. Sebaliknya, K=10 menghasilkan distribusi klaster yang lebih seimbang dengan ukuran berkisar 61-1361 sampel, menghasilkan evaluasi yang lebih stabil (std = 0.30).

3.4. Analisis Komposisi Klaster

Analisis komposisi klaster dilakukan untuk memahami karakteristik setiap klaster yang dihasilkan oleh algoritma K-Means serta memvalidasi bahwa skema validasi berbasis klaster memang merepresentasikan variasi distribusi (out-of-distribution/OOD). Pada penelitian ini, konfigurasi K=10 digunakan sebagai fokus analisis karena memberikan trade-off yang baik antara granularitas distribusi dan stabilitas evaluasi.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15641

Tabel 4. Komposisi Klaster untuk K=10

Klaster	N_total	Benign	Adware	Adware Rate	Purity	Entropy	Difficulty
0	61	38	23	0.3770	0.6230	0.9559	Hard
1	573	365	208	0.3630	0.6370	0.9451	Hard
2	116	84	32	0.2759	0.7241	0.8498	Hard
5	278	253	25	0.0899	0.9101	0.4362	Easy
7	1361	677	684	0.5026	0.5026	1.0000	Hard
8	12	10	2	0.1667	0.8333	0.6500	Medium

Hasil pada Tabel 4 menunjukkan heterogenitas klaster yang tinggi pada komposisi benign/adware. Klaster 7 (klaster terbesar dengan 1.361 sampel) memiliki distribusi kelas yang hampir seimbang (Adware Rate ≈ 0.50) dan entropy mendekati 1.0, yang mengindikasikan tingkat kesulitan tinggi karena pemisahan kelas menjadi lebih ambigu. Sebaliknya, klaster dengan purity tinggi (misalnya klaster 5, purity > 0.90) cenderung lebih mudah dipelajari karena dominasi satu kelas lebih kuat.

Variansi performa yang lebih besar pada skema ClusterOOD (misalnya simpangan baku F1 sekitar 0.30 pada konfigurasi tertentu) bukan merupakan kelemahan, melainkan konsekuensi yang diharapkan dari evaluasi OOD yang lebih realistis. Hal ini terutama disebabkan oleh, Heterogenitas klaster (komposisi kelas dan tingkat kesulitan berbeda), Tantangan OOD, karena model divalidasi pada klaster yang tidak pernah dilihat saat pelatihan, dan Ukuran klaster yang timpang (contoh: 61 hingga 1.361 sampel), yang dapat memperbesar fluktuasi metrik antar fold/klaster.

3.5. Perbandingan dengan Baseline Cross-Validation

Untuk menegaskan bahwa validasi berbasis klaster memberikan estimasi yang lebih realistis, dilakukan perbandingan dengan metode cross-validation standar (KFold dan StratifiedKFold). Tabel 5 menyajikan perbandingan antara ClusterOOD dan baseline.

Tabel 5. Perbandingan ClusterOOD dengan Baseline

Scheme	**Model**	**F1 (mean±std)**	**MCC**	**Recall**
ClusterOOD(K=10)	CAS-Enhanced	0.8345±0.3008	0.5075±0.4409	0.8624±0.3108
ClusterOOD(K=10)	XGB	0.8275±0.3083	0.5078±0.4453	0.7998±0.3218
KFold	XGB	0.9589±0.0128	0.9305±0.0194	0.9574±0.0045
StratifiedKFold	XGB	0.9540±0.0058	0.9217±0.0099	0.9574±0.0184

Perbandingan menunjukkan bahwa evaluasi ClusterOOD menghasilkan metrik yang lebih rendah (misalnya F1 sekitar 0.83) dibandingkan random CV (sekitar 0.95–0.96). Perbedaan ini expected dan desirable karena:

1. Random CV menguji pada distribusi yang secara praktik masih sangat mirip dengan data pelatihan (asumsi IID), sehingga cenderung menghasilkan estimasi optimistis.
2. ClusterOOD menguji pada distribusi berbeda (OOD), sehingga memberikan estimasi yang lebih konservatif namun lebih realistis untuk skenario deployment.
3. Simpangan baku lebih besar pada ClusterOOD mencerminkan variasi kondisi dunia nyata yang biasanya tidak tertangkap oleh random CV.

Dengan demikian, model yang dipilih melalui ClusterOOD cenderung lebih robust terhadap adware baru dengan karakteristik yang tidak identik dengan data pelatihan, meskipun nilai validasinya lebih konservatif.

Model yang dipilih melalui ClusterOOD lebih generalizable untuk mendeteksi adware baru dengan karakteristik berbeda dari training data, meskipun metrik validasinya lebih konservatif.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15641

3.6. Hasil Evaluasi pada Data Uji Hold-out

Berdasarkan hasil validasi silang, konfigurasi terbaik digunakan untuk evaluasi final pada data uji hold-out yang tidak pernah dilihat selama proses pelatihan maupun validasi. Pada eksperimen ini, konfigurasi terbaik adalah SoftVoting dengan K=10, yang kemudian diuji pada subset hold-out.

Tabel 6. Confusion Matrix pada Data Uji Hold-out

	Predicted Benign	**Predicted Adware**
Actual Benign	356	8
Actual Adware	6	239

Dari 364 sampel benign, 356 (97.8%) diprediksi benar dan 8 (2.2%) merupakan false positive. Dari 245 sampel adware, 239 (97.6%) diprediksi benar dan 6 (2.4%) merupakan false negative. Recall yang tinggi (97.55%) penting dalam konteks deteksi adware karena meminimalkan adware yang lolos deteksi.

3.7. Pembahasan Umum

Hasil eksperimen menunjukkan bahwa penggunaan validasi berbasis klaster (ClusterOOD) memberikan evaluasi yang lebih realistis dibandingkan cross-validation standar. Meskipun nilai metrik pada ClusterOOD cenderung lebih rendah daripada random CV, hal ini mencerminkan tantangan deployment yang sesungguhnya, yaitu model menghadapi variasi distribusi data yang berbeda dari data pelatihan

Selain itu, analisis komposisi klaster menunjukkan adanya klaster “hard” dengan entropy tinggi dan distribusi kelas yang lebih ambigu, yang turut menjelaskan variasi performa antar fold/klaster. Secara keseluruhan, performa final pada data uji hold-out memvalidasi bahwa konfigurasi model yang dipilih melalui evaluasi berbasis klaster tetap memberikan kemampuan generalisasi yang baik.

4. KESIMPULAN

Penelitian ini mengusulkan pendekatan deteksi adware berbasis machine learning dengan menekankan evaluasi yang lebih realistis melalui validasi silang berbasis klaster (C_ClusterOOD) serta pemanfaatan teknik ensemble untuk meningkatkan kemampuan generalisasi model pada kondisi pergeseran distribusi data. Evaluasi dilakukan menggunakan dataset CICMalDroid 2020 yang difilter menjadi dua kelas (adware dan benign) serta menggunakan skema pembagian data yang dirancang untuk meminimalkan kebocoran informasi antara data pelatihan dan pengujian.

Hasil eksperimen pada skema C_ClusterOOD menunjukkan bahwa pendekatan Cluster-Aware Stacking (CAS/CAS-Enhanced) memberikan performa yang kompetitif dan mampu mempertahankan kinerja pada skenario out-of-distribution, yang tercermin dari capaian metrik berbasis ketidakseimbangan kelas (misalnya F1-score adware dan MCC) pada beberapa konfigurasi. Di sisi lain, pemilihan konfigurasi terbaik untuk evaluasi akhir ditentukan berdasarkan hasil komprehensif pada seluruh kandidat model.

Pada pengujian data uji hold-out, konfigurasi terbaik yang diperoleh adalah Soft Voting dengan K=10, dengan akurasi 97,70%, F1-score adware 97,15%, MCC 95,23%, dan recall adware 97,55%, serta ROC-AUC 0,9976. Hasil ini menunjukkan bahwa model mampu memberikan kinerja tinggi dan seimbang dalam mendeteksi adware sekaligus menjaga tingkat kesalahan yang rendah pada kelas benign, sehingga relevan untuk skenario penerapan deteksi adware di lingkungan Android yang dinamis.

Secara keseluruhan, penelitian ini menegaskan bahwa validasi berbasis klaster (ClusterOOD) penting untuk memperoleh estimasi performa yang lebih mendekati kondisi nyata, terutama ketika data yang dihadapi saat deployment berpotensi berbeda dari data pelatihan. Kombinasi evaluasi berbasis klaster dan ensemble learning menghasilkan sistem yang lebih robust dan berpotensi menjadi alternatif yang andal untuk deteksi adware pada aplikasi Android.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15641

DAFTAR PUSTAKA

[1] Y. Zhou and X. Jiang, "Dissecting Android malware: Characterization and evolution," in *Proceedings - IEEE Symposium on Security and Privacy*, Institute of Electrical and Electronics Engineers Inc., 2012, pp. 95–109. doi: 10.1109/SP.2012.16.

[2] D. Arp, M. Spreitzenbarth, M. Hübner, H. Gascon, and K. Rieck, "Drebin: Effective and Explainable Detection of Android Malware in Your Pocket," 2014. [Online]. Available: http://dx.doi.org/

[3] S. Suresh, F. Di Troia, K. Potika, and M. Stamp, "An analysis of Android adware," *Journal of Computer Virology and Hacking Techniques*, vol. 15, no. 3, pp. 147–160, Sep. 2019, doi: 10.1007/s11416-018-0328-8.

[4] Xingyu. Wang, David. Lo, and Emad. Shihab, *Should You Consider Adware as Malware in You Study?* IEEE, 2019.

[5] R. A. Alzahrani and M. Aljabri, "AI-Based Techniques for Ad Click Fraud Detection and Prevention: Review and Research Directions," Feb. 01, 2023, *MDPI*. doi: 10.3390/jsan12010004.

[6] S. Seraj, M. Pavlidis, M. Trovati, and N. Polatidis, "MadDroid: malicious adware detection in Android using deep learning," *Journal of Cyber Security Technology*, vol. 8, no. 3, pp. 163–190, 2024, doi: 10.1080/23742917.2023.2247197.

[7] C. Wang *et al.*, "The arts and crafts of android adware across a decade," *Automated Software Engineering*, vol. 33, no. 1, Jun. 2026, doi: 10.1007/s10515-025-00575-9.

[8] H. Huang *et al.*, "A Large-Scale Study of Android Malware Development Phenomenon on Public Malware Submission and Scanning Platform," *IEEE Trans. Big Data*, vol. 7, no. 2, pp. 255–270, Jan. 2018, doi: 10.1109/tbdata.2018.2790439.

[9] H. Wang, J. Si, H. Li, and Y. Guo, "RmvDroid: Towards a reliable android malware dataset with app metadata," in *IEEE International Working Conference on Mining Software Repositories*, IEEE Computer Society, May 2019, pp. 404–408. doi: 10.1109/MSR.2019.00067.

[10] U. Farooq, S. S. Khurana, P. Singh, and M. Kumar, "An Empirical Study on Detection of Android Adware Using Machine Learning Techniques," *Multimed. Tools Appl.*, vol. 83, no. 13, pp. 38753–38792, Apr. 2024, doi: 10.1007/s11042-023-16920-7.

[11] M. M. Alani and A. I. Awad, "AdStop: Efficient flow-based mobile adware detection using machine learning," *Comput. Secur.*, vol. 117, Jun. 2022, doi: 10.1016/j.cose.2022.102718.

[12] J. Park and S. Jung, "Android Adware Detection using Soot and CFG," *J. Wirel. Mob. Netw. Ubiquitous Comput. Dependable Appl.*, vol. 13, no. 4, pp. 94–104, 2022, doi: 10.58346/JOWUA.2022.I4.006.

[13] K. Katutwa, D. E. Banda, and J. Shabani, "Detection of Android Adware Using Transfer Learning with Computer Vision," *Computer Science and Engineering*, vol. 12, pp. 30–38, 2022, doi: 10.5923/j.computer.20221201.03.

[14] M. Sebastián, R. Rivera, P. Kotzias, and J. Caballero, "AVCLASS: A Tool for Massive Malware Labeling," 2016. [Online]. Available: https://github.com/malicialab/avclass

[15] T. G. Dietterich, "Ensemble Methods in Machine Learning," 2000. [Online]. Available: http://www.cs.orst.edu/~tgd

[16] Karfindo, R. Turaina, and R. Saputra, "Optimalisasi Deteksi Malware pada Platform Android dengan Pendekatan Ensemble Machine Learning," *Jurnal Nasional Komputasi dan Teknologi Informasi (JNKTI)*, vol. 7, no. 3, 2024.

[17] D. H. Wolpert, "STACKED GENERALIZATION," 1992.

[18] Mahdavifar Samaneh, Fitriah Abdul Kadir Andi, Fatemi Rasool, Alhadidi Dima, and A. Ghorbani Ali, *Dynamic Android Malware Category Classification using Semi-Supervised Deep Learning*. IEEE, 2020.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15641

[19] S. Mahdavifar, D. Alhadidi, and A. A. Ghorbani, "Effective and Efficient Hybrid Android Malware Classification Using Pseudo-Label Stacked Auto-Encoder," *Journal of Network and Systems Management*, vol. 30, no. 1, Jan. 2022, doi: 10.1007/s10922-021-09634-4.

e-ISSN : 2356-2579 | p-ISSN : 1412-2693, DOI : 10.62411/tc.v25i1.15641