JTIM : Jurnal Teknologi Informasi dan Multimedia

p-ISSN : 2715-2529
e-ISSN : 2684-9151
https://journal.sekawan-org.id/index.php/jtim

Deteksi Malware pada Perangkat Android Menggunakan *Ensemble Learning*

Muhamad Azwar [1]*, Lilik Widyawati [1], Raisul Azhar [1], Kartarina [2], Tanwir [1], Andi Sofyan Anas[2]

[1] Program Studi Ilmu Komputer, Universitas Bumigora; Indonesia.
[2] Program Studi Rekayasa Perangkat Lunak, Universitas Bumigora, Indonesia

* Korespondensi: muha.azwar@gmail.com

Sitasi: Azwar, M.; Widyawati, L.; Azhar, R.; Kartarina, K.; Tanwir, T.; and Anas, A. S. (**2025**). Deteksi Malware pada Perangkat Android Menggunakan Ensemble Learning. JTIM: Jurnal Teknologi Informasi Dan Multimedia, 7(3), 408-419. https://doi.org/10.35746/jtim.v7i3.573

Diterima: 19-07-2025
Direvisi: 24-09-2024
Disetujui: 31-05-2025

Abstract: The increasing use of permission-based applications on mobile platforms has raised concerns regarding privacy and security. Android, being one of the most widely used operating systems for interacting with mobile applications, is particularly susceptible to various security risks that must be promptly addressed. Low digital literacy and a lack of user awareness about security risks—especially when installing applications from unofficial sources or without paying attention to access permissions—make users vulnerable to malware attacks. Uninformed users can easily become victims of malware insertion by irresponsible parties, turning them into targets for data manipulation and even data theft, which may then be sold on illegal forums. Attackers exploit the permission system, allowing them to freely access the target smartphone. This lack of awareness among users increases their vulnerability to malware injection and subsequent threats such as data manipulation and the theft of personal information, which can be traded on underground markets. One approach to detecting malicious behavior in mobile applications is the use of machine learning techniques. These techniques can analyze application patterns and behaviors based on features such as requested permissions. Popular algorithms for malware detection include Support Vector Machine (SVM) and Random Forest (RF), both of which have demonstrated strong performance in various studies. However, to further improve accuracy and reduce classification errors, ensemble learning approaches such as Adaptive Boosting (AdaBoost) are increasingly being adopted. Ensemble learning combines multiple predictive models to produce more reliable classification results compared to single models. This study evaluates the performance of several classification algorithms in detecting malicious Android applications. The results show that AdaBoost achieved a high accuracy rate of 91.65% and an AUC value of 95%, effectively distinguishing between safe applications and malware. Therefore, the use of machine learning algorithms—particularly ensemble methods like AdaBoost—can serve as a promising solution to enhance the security and privacy of Android-based mobile application users.

Keywords: Mobile Security; Android; Machine Learning; Malware Detection

Abstrak: Meningkatnya penggunaan aplikasi berbasis izin pada aplikasi mobile menimbulkan kekhawatiran terhadap masalah privasi dan keamanan. Sistem Operasi *Android* merupakan salah satu sistem operasi yang paling banyak digunakan sebagai media interaksi dengan aplikasi *mobile*, sehingga berpotensi menimbulkan berbagai risiko keamanan yang harus segera diatasi. Rendahnya literasi digital dan kurangnya pemahaman pengguna terhadap risiko keamanan, terutama saat menginstal aplikasi dari sumber tidak resmi atau tanpa memperhatikan izin akses, menjadikan mereka rentan terhadap serangan *malware*. Pengguna yang tidak sadar akan bahaya ini dapat dengan mudah disisipkan *malware* oleh pihak yang tidak bertanggung jawab, sehingga menjadi target serangan untuk manipulasi data bahkan pencurian data yang kemudian dijual di forum ilegal. Penyerang memanfaatkan sistem izin yang ada sehingga sangat leluasa keluar masuk pada

smartphone target. Pengguna yang tidak sadar akan bahaya ini dapat dengan mudah menjadi korban penyisipan malware oleh pihak yang tidak bertanggung jawab. Hal ini menjadikan mereka target serangan seperti manipulasi data dan pencurian informasi pribadi, yang kemudian dapat diperjualbelikan di forum-forum ilegal. Salah satu pendekatan yang digunakan untuk mendeteksi perilaku berbahaya pada aplikasi *mobile* adalah dengan memanfaatkan teknik machine learning. Teknik ini dapat menganalisis pola dan perilaku aplikasi berdasarkan fitur-fitur seperti izin akses (*permission*) yang diminta oleh aplikasi. Beberapa algoritma populer dalam deteksi *malware* adalah *Support Vector Machine* (SVM) dan *Random Forest* (RF), yang telah terbukti memberikan performa yang baik dalam berbagai penelitian. Namun, untuk meningkatkan akurasi dan mengurangi kesalahan klasifikasi, pendekatan *ensemble learning* seperti *Adaptive Boosting* (*AdaBoost*) mulai banyak digunakan. *Ensemble learning* menggabungkan beberapa model prediktif untuk menghasilkan hasil klasifikasi yang lebih andal dibandingkan model tunggal. Penelitian ini mengevaluasi kinerja beberapa algoritma klasifikasi dalam mendeteksi aplikasi berbahaya pada *Android*. Hasil menunjukkan bahwa *AdaBoost* memiliki tingkat akurasi yang cukup tinggi, yaitu sebesar 91,65% dengan nilai AUC mencapai 95%, dapat membedakan antara aplikasi aman dan *malware* secara efektif. Oleh karena itu, penggunaan *algoritma machine learning*, khususnya metode *ensemble* seperti *AdaBoost*, dapat menjadi solusi dalam meningkatkan keamanan dan privasi pengguna aplikasi *mobile* berbasis *Android*.

Kata kunci: Keamanan *Mobile*; *Android*; *Machine Learning*; *Malware Detection*

1. Pendahuluan

Perangkat *mobile* telah menjadi bagian integral dari kehidupan sehari-hari. *Android*, sebagai salah satu sistem operasi *mobile* yang paling dominan, menawarkan berbagai kemudahan dan fungsionalitas bagi penggunanya [1]. Namun, popularitas ini juga menarik perhatian para penyerang siber yang terus berusaha mengeksploitasi kerentanan keamanan yang ada [2]. Salah satu ancaman terbesar yang dihadapi pengguna *Android* adalah *malware*, yaitu perangkat lunak berbahaya yang dirancang untuk merusak, mengganggu, atau mendapatkan akses tidak sah ke perangkat [3]. *Malware* pada perangkat *Android* dapat mengakibatkan berbagai konsekuensi negatif, mulai dari pencurian data pribadi, kerugian finansial, hingga pelanggaran privasi yang serius [4].

Meskipun *Google Play Store* dan sumber aplikasi resmi lainnya berupaya untuk meminimalkan risiko ini melalui berbagai kebijakan keamanan, aplikasi yang diunduh dari sumber tidak resmi masih menjadi jalur utama masuknya *malware* ke dalam perangkat pengguna [5]. Ketidaktahuan dan kurangnya kesadaran pengguna tentang risiko ini semakin memperburuk situasi, karena banyak dari mereka yang mengizinkan akses yang tidak semestinya tanpa memahami potensi bahayanya. Dalam upaya untuk mendeteksi dan mencegah serangan *malware*, berbagai metode telah dikembangkan, salah satunya adalah penggunaan algoritma *machine learning*. Pendekatan ini memanfaatkan kemampuan *machine learning* untuk menganalisis pola dan perilaku aplikasi, sehingga mampu mengidentifikasi karakteristik yang menunjukkan adanya aktivitas berbahaya. Di antara berbagai teknik *machine learning*, *ensemble learning* telah menunjukkan potensi yang signifikan dalam meningkatkan akurasi deteksi *malware* [6].

Ensemble learning adalah pendekatan yang menggabungkan beberapa model pembelajaran mesin untuk menghasilkan model prediksi yang lebih kuat dan andal [7]. Dengan menggabungkan kekuatan dari berbagai algoritma, *ensemble learning* mampu menangani kompleksitas dan variabilitas data yang tinggi, serta mengurangi kemungkinan kesalahan prediksi yang disebabkan oleh kelemahan dari model individu. Algoritma *ensemble* yang populer termasuk *Adaptive Boosting* (*AdaBoost*), *Bagging*, dan *Random Forest*, yang semuanya telah terbukti efektif dalam berbagai aplikasi keamanan siber [8][9]. Penelitian ini bertujuan untuk mengevaluasi efektivitas metode *ensemble learning* dalam

mendeteksi *malware* pada perangkat *android*. Dengan membandingkan kinerja berbagai *algoritma ensemble*, maka dapat ditemukan solusi yang paling optimal dalam memberikan perlindungan yang lebih baik bagi pengguna *android*. Penelitian ini juga bertujuan untuk memberikan pemahaman yang lebih mendalam tentang bagaimana teknik *ensemble learning* dapat diterapkan dalam konteks keamanan *mobile*, serta memberikan rekomendasi bagi pengembang aplikasi dan penyedia layanan untuk meningkatkan praktik keamanan mereka.

Berbagai penelitian telah dilakukan untuk meningkatkan efektivitas deteksi malware pada perangkat Android menggunakan pendekatan machine learning. Salah satu penelitian oleh El Attaoui et al. (2024) menunjukkan bahwa algoritma Random Forest mampu mendeteksi malware dengan akurasi mencapai 98,47%, sensitivitas 98,60%, dan F-score 98,60% [10]. Hasil ini mengindikasikan bahwa metode ensemble dapat memberikan hasil klasifikasi yang sangat andal. Studi lain oleh Alhadethy et al. (2025) mengembangkan framework berbasis stacked classifiers, yaitu kombinasi dari beberapa algoritma seperti SVM, KNN, dan Random Forest, yang berhasil meningkatkan akurasi deteksi hingga lebih dari 98% [11]. Pendekatan ini membuktikan bahwa kombinasi model mampu menangkap kompleksitas pola serangan malware lebih baik dibandingkan satu algoritma tunggal. Selain itu, pendekatan ensemble convolutional neural network (CNN-Ensemble) juga dikaji untuk mengidentifikasi malware berdasarkan karakteristik aplikasi. Penelitian oleh Singh et al. (2023) memanfaatkan arsitektur CNN yang dikombinasikan secara ensemble, yang terbukti mampu meningkatkan akurasi dan mengurangi kesalahan klasifikasi pada dataset CICMalDroid 2020 [12].

Penelitian lainnya oleh Jiang et al. (2024) mengeksplorasi analisis lalu lintas jaringan aplikasi Android untuk mendeteksi aktivitas berbahaya menggunakan teknik ensemble learning, menunjukkan peningkatan kinerja deteksi dibandingkan metode tradisional [13]. Temuan-temuan ini mengindikasikan bahwa dalam kurun waktu lima tahun terakhir, ensemble learning menjadi salah satu pendekatan yang paling menjanjikan dalam mitigasi ancaman malware Android. Namun, tantangan seperti kompleksitas komputasi, adaptasi terhadap varian malware baru, dan kebutuhan terhadap dataset terkini masih menjadi perhatian utama dalam pengembangan metode yang lebih optimal.

Perangkat *android*, dengan popularitasnya yang tinggi, menjadi target utama bagi serangan *malware* yang mengancam keamanan dan privasi pengguna [14]. Meski upaya telah dilakukan untuk memitigasi risiko ini, aplikasi yang diunduh dari sumber tidak resmi masih menjadi pintu utama masuknya *malware* [15]. Penggunaan algoritma *machine learning*, khususnya *ensemble learning*, menawarkan solusi yang menjanjikan dalam mendeteksi *malware* secara efektif [16]. Dengan menggabungkan beberapa model pembelajaran mesin, *ensemble learning* dapat meningkatkan akurasi deteksi dan mengurangi kesalahan prediksi yang mungkin terjadi pada model individu [17][18][19]. Penelitian ini bertujuan untuk mengevaluasi kinerja berbagai *algoritma ensemble* dalam mendeteksi *malware* pada perangkat *android*, guna menemukan solusi yang optimal untuk melindungi pengguna dari ancaman siber yang semakin kompleks.

2. Bahan dan Metode

Bagian ini menjelaskan metodologi yang diusulkan serta komponen-komponen yang digunakan untuk uji coba dan analisis. Algoritma ini mencakup *Support Vector Machine* (SVM), *Naïve Bayes* dan *Decision Tree*. Tahapan metodologi ini meliputi:

2.1. Kumpulan Data

Data yang digunakan dalam penelitian ini diambil dari univeristas *Brunswich* dengan url resmi: https://www.unb.ca/cic/datasets/index.html tertanggal 16 Juli 2024 Kumpulan data CICMalDroid terdiri dari 5.000 lebih sampel *malware* yang capture (426 *malware* dan 5.065 *benign*) yang ditemukan di perangkat *Android*. Sampel *malware* diklasifikasikan ke dalam 4 kategori: *Adware*, *Ransomware*, *Scareware*, dan *malware* SMS [20].

2.2. Pemilihan Fitur

Terdapat 428 jenis izin *android* pada tahun 2024 jenis izin *android* ini dapat digunakan sebagai label fitur dalam kumpulan data, pada penelitian ini ada 12 izin *android* yang akan digunakan yaitu:

Android.permission.STORAGE
Android.permission.READ_EXTRERNAL_STORAGE
Android.permission.READ_CONTACTS
Android.permission.WRITE_CONTACTS
Android.permission.GET_ACCOUNTS
Android.permission.ACCESS_FINE_LOCATION
Android.permission.CAMERA
Android.permission.USE_FINGERPRINT
Android.permission.READ_CALENDAR
Android.permission.WRITE_CALENDAR
Android.permission.ACCESS_CORSE_LOCATION
Android.permission.PREVENT_POWER_KEY

Teks diatas menunjukan 12 daftar izin *android* yang sangat merugikan pengguna apabila bila di aktifkan pada perangkat *smartphone* tanpa seizin pengguna yang sah, adapun dampak ketiga dari 12 izin tersebut sebagai berikut:

- *STORAGE*: Memberikan akses ke penyimpanan eksternal berarti aplikasi bisa membaca dan mengakses semua file yang disimpan di perangkat, termasuk foto, dokumen, dan data pribadi lainnya. Ini bisa digunakan oleh aplikasi berbahaya untuk mencuri data pengguna [21].
- *CAMERA*: Memberikan akses ke kamera perangkat. Aplikasi berbahaya bisa mengambil foto atau merekam video tanpa sepengetahuan pengguna [22].
- USE_*FINGERPRINT*: Memberikan akses untuk menggunakan sensor sidik jari pada perangkat. Jika dikombinasikan dengan izin lain, ini bisa digunakan untuk *bypass* keamanan perangkat atau mencuri data sidik jari [23].

2.3. Seleksi Data

Dari 5000 lebih sampel *malware* yang terdapat pada *CICDataset*, namun yang digunakan pada penelitian ini hanya 431 sampel yang kemudian melalui tahap ekstraksi ke dalam file CSV. Tabel 1 menampilkan jumlah setiap sampel yang diambil.

Tabel 1. Jumlah data *malware*

Tipe	Jumlah	Keterangan
Adware	20	*Adware* adalah jenis perangkat lunak yang secara otomatis menampilkan atau mengunduh iklan ke komputer atau perangkat lain, sering kali tanpa persetujuan pengguna
Scareware	46	*Scareware* adalah jenis perangkat lunak berbahaya (*malware*) yang dirancang untuk menakut-nakuti pengguna dengan menampilkan pesan palsu atau menyesatkan, yang biasanya mengklaim bahwa komputer atau perangkat mereka telah terinfeksi virus atau mengalami masalah serius
SMS *Malware*	39	SMS *Malware* adalah jenis perangkat lunak berbahaya (*malware*) yang dirancang untuk menyebar atau mengeksekusi serangan melalui pesan teks (SMS) pada perangkat seluler
Ransomware	21	*Ransomware* adalah jenis *malware* (perangkat lunak berbahaya) yang menyerang dengan mengenkripsi file atau mengunci akses pengguna ke sistem komputer mereka

Tipe	Jumlah	Keterangan
Beign	305	*Benign* adalah istilah yang berarti tidak berbahaya atau tidak berisiko
Total	**431**	

2.4. Pra-Pemrosesan Data

Untuk menyiapkan kumpulan data sebagai data yang akan di latih dan diuji dengan model pembelajaran *ansambel*, maka digunakan skrip *Python* untuk menulis ke file CSV. Program ini melakukan perulangan setiap sampel dan mengekstrak nama-nama izin yang ditemukan dari APK. Jika ditemukan izin maka akan diisi pada kolom dengan nilai 1 jika tidak ditemukan izin pada file APK maka pada kolom akan diisi nilai 0.

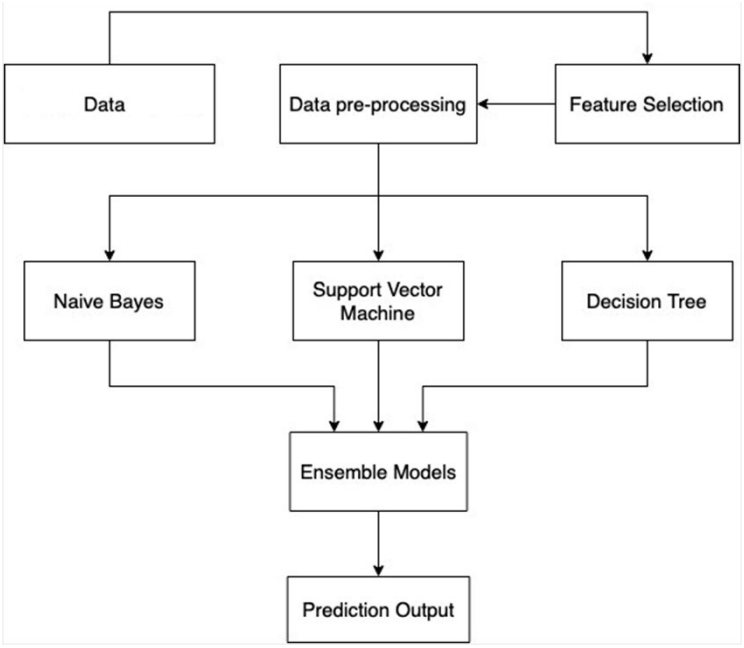
2.5. Pembelajaran Ansamble

Pembelajaran *ansambel* menggabungkan keputusan beberapa algoritma klasifikasi menjadi satu, baik dengan hasil rata-rata atau suara mayoritas. Metodologi ini didasarkan pada gagasan bahwa keputusan dari berbagai sumber lebih andal daripada keputusan dari satu sumber. Metode *ansambel* bertujuan mengurangi bias dan varians dengan menggabungkan beberapa metode yang lemah untuk mencapai kinerja yang lebih baik.

Ada tiga metode pembelajaran *ansambel*, antara lain:

1. *Bagging* (*Bootstrap Aggregating*): Metode ini menggabungkan beberapa model yang dilatih secara terpisah pada subset data yang berbeda yang diambil melalui *bootstrap* sampling. Hasil akhir diperoleh dengan menggabungkan prediksi dari semua model, biasanya dengan rata-rata (untuk regresi) atau suara mayoritas (untuk klasifikasi). *Bagging* membantu mengurangi *varians* dan mencegah model kehilangan kemampuan generalisasi, sehingga tidak bisa membuat prediksi yang akurat untuk situasi atau data yang berbeda [24][25].
2. *Boosting*: Metode ini menggabungkan beberapa model dengan melatih setiap model baru untuk memperbaiki kesalahan dari model sebelumnya. Setiap model diberi bobot berdasarkan kinerjanya, dan model yang lebih baik mendapatkan bobot lebih tinggi [26]. Contoh algoritma *boosting* yang populer adalah *AdaBoost* dan *Gradient Boosting*. *Boosting* bertujuan mengurangi bias dan meningkatkan kinerja secara keseluruhan.
3. *Stacking* (*Stacked Generalization*): Metode ini menggabungkan beberapa model dasar dengan menggunakan model *meta*. Model dasar dilatih pada data yang sama, dan prediksi mereka digunakan sebagai input untuk model meta yang membuat prediksi akhir. *Stacking* dapat menangkap informasi dari berbagai model dasar dan menggabungkannya untuk meningkatkan kinerja prediksi [27].

Berikut adalah alur ketiga algoritma tersebut dalam melakukan prediksi, dimulai dari seleksi fitur dan pra-pemrosesan data sebelum menghasilkan keluaran:



Gambar 1. Alur atau proses pada sistem

Pada gambar 1 menunjukan alur atau proses bagaimana sistem bekerja, dengan keterangan sebagai berikut:

2.5.1. Dataset

Mengumpulkan data yang akan digunakan dalam penelitian, misalnya dataset CICEV2023 yang terdiri dari sampel *malware* dan *benign* pada perangkat *Android* dan Memisahkan *dataset* menjadi set pelatihan (training set) dan set pengujian (testing set), biasanya dengan rasio 80:20 atau 70:30 untuk memastikan model dapat diuji dengan data yang tidak terlihat sebelumnya.

2.5.2. Feature Selection

Menentukan fitur-fitur yang relevan untuk digunakan dalam model, seperti izin aplikasi, aktivitas, layanan dan menggunakan teknik pengurangan dimensi seperti *Principal Component Analysis* (PCA) untuk mengurangi jumlah fitur tanpa kehilangan informasi penting, jika diperlukan.

2.5.3. Data Pre-Processing

Menghapus atau memperbaiki data yang hilang atau tidak valid kemudian Mengubah data ke dalam skala yang seragam untuk memastikan konsistensi dan meminimalisasi bias dan memastikan data terbagi dengan proporsi yang sesuai untuk pelatihan dan pengujian.

2.5.4. Proses Algoritma

Menggunakan dataset pelatihan untuk melatih model *Naive Bayes*. Algoritma ini mengasumsikan independensi antar fitur dan menghitung probabilitas setiap kelas berdasarkan fitur yang diberikan, melatih model SVM dengan dataset pelatihan untuk menemukan *hyperplane* optimal yang memisahkan kelas-kelas yang berbeda dan melatih model *Decision Tree* dengan dataset pelatihan untuk membuat pohon keputusan berdasarkan fitur-fitur yang relevan.

2.5.5. Ensemble Models

Menggabungkan prediksi dari beberapa model (*Naive Bayes*, SVM, dan *Decision Tree*) menggunakan teknik *ensemble* seperti *bagging*, *boosting*, atau *stacking*, melatih model

ensemble dengan dataset pelatihan, di mana hasil dari model individu digunakan sebagai input untuk model meta (dalam kasus *stacking*).

2.5.6. Prediction Output

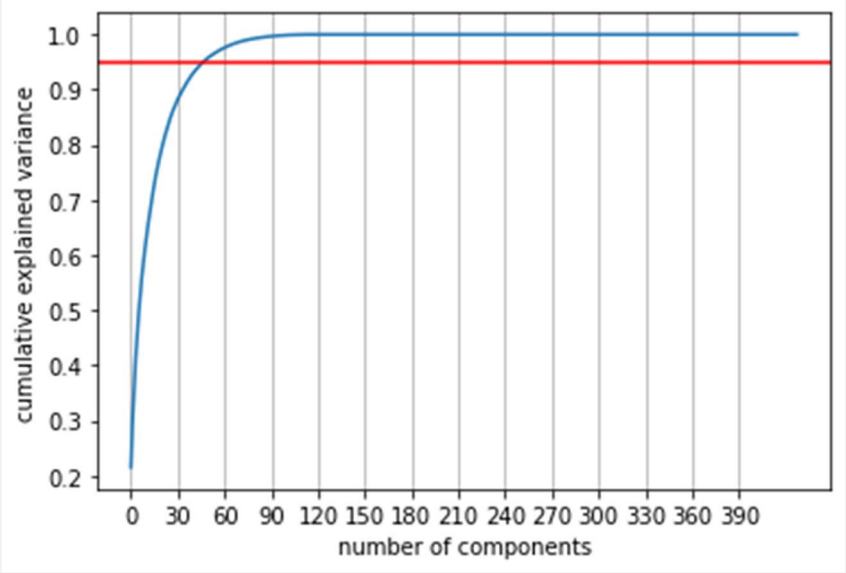
Menggunakan model yang telah dilatih untuk membuat prediksi pada data baru atau data pengujian dan mengukur kinerja model menggunakan metrik seperti akurasi, *precision*, *recall*, dan AUC (*Area Under the Curve*) untuk menentukan efektivitas model dalam mendeteksi *malware*.

3. Hasil

3.1. Pengurangan Fitur

Analisis Komponen Utama (PCA) adalah teknik reduksi fitur yang populer untuk mengurangi dimensi dataset besar dengan mengubah variabel menjadi variabel independen linier (vektor eigen). Vektor eigen ini, juga dikenal sebagai komponen utama, dapat digunakan sebagai fitur baru dalam model pembelajaran mesin [28]. PCA bertujuan mengatasi overfitting dengan menghapus data yang berisik atau tidak relevan. Langkah awal sebelum melakukan PCA adalah menentukan jumlah komponen utama yang tepat untuk digunakan. Aturan umumnya adalah menggunakan jumlah komponen utama yang diperlukan untuk memperhitungkan jumlah varian terbesar dalam kumpulan data.

Dengan rasio varians yang dijelaskan, kita dapat memvisualisasikan jumlah komponen utama yang diperlukan untuk memperhitungkan setidaknya 95% varians. Gambar 2 menunjukkan bahwa diperlukan setidaknya 45-50 komponen utama.



Gambar 2. Jumlah Komponen Utama yang dibutuhkan sehubungan dengan jumlah varians yang diperhitungkan

Pada gambar 2 warna merah menunjukan jumlah komponen utama dan warna biru menunjukan varian, warna biru dan merah bertemu antara nilai 30 dan 60 yang artinya diperluakan setidaknya 45-50 komponen utama untuk bisa mencapai 95%, Setelah melakukan analisis komponen utama (PCA) pada kumpulan data dan menyesuaikan setiap model dengan komponen utama sebagai fitur, disimpulkan bahwa kinerja setiap model tidak baik di semua aspek. Hal ini disebabkan oleh jumlah data yang masih sedikit dan kurangnya fitur penting untuk klasifikasi yang lebih baik.

3.2. Evaluasi Kinerja

Tabel 2 menampilkan evaluasi kinerja untuk model pembelajaran mesin *SVM*, *Naïve Bayes*, dan *Decision Tree*. *Cross validation* K-fold dengan k=5 digunakan untuk membagi dataset menjadi beberapa lipatan guna meminimalkan *overfitting*, jika menggunakan K=2 dalam cross-validation, evaluasi model mungkin kurang akurat atau representatif dibandingkan dengan K=5 atau K yang lebih tinggi. Sebab, dengan lebih banyak lipatan (*folds*), lebih banyak variasi data dapat diuji, sehingga memberikan hasil evaluasi yang lebih stabil dan dapat diandalkan untuk model seperti *SVM, Naïve Bayes, dan Decision Tree.*

Tabel 2. Evaluasi kinerja metode 1 – 3.

Metode	Accuracy	Precision	Recall	F1-Score
Decision Tree	78.93%	96.15%	71.43%	81.97%
Naïve Bayes	78.93%	93.33%	40%	55.99%
SVM	90.01%	89.29%	71.43%	79.37%

Presisi menunjukkan proporsi prediksi positif yang benar dari seluruh prediksi positif, sedangkan *recall* menunjukkan proporsi prediksi positif yang benar dari semua kasus positif sebenarnya. Baik presisi maupun *recall* mengukur seberapa akurat model dalam mengidentifikasi data dengan benar.

Rumus penghitungan *recall* adalah sebagai berikut, dimana TP = *True Positive* dan FN.= *False Negative* (salah prediksi *positif*):

$$Recall = \frac{TP}{TP + FN} \tag{1}$$

F1-Score didefinisikan sebagai rata-rata presisi.

$$F1 - Score = 2 * \frac{precision * recall}{precision + recall} \tag{2}$$

Lebih jelas untuk mendapatkan keempat komponen tersebut yaitu menggunakan *source code* python sebagai berikut:

```
for name, model in models.items():
    model.fit(X_train, y_train)
    y_prediksi = model.predict(X_test)

    accuracy = accuracy_score(y_test, y_ prediksi)
    precision = precision_score(y_test, y_ prediksi, average='weighted')
    recall = recall_score(y_test, y_ prediksi, average='weighted')
    f1 = f1_score(y_test, y_ prediksi, average='weighted')
```

Metode klasifikasi terdiri dari tiga model yang dievaluasi: SVM, *Decision Tree*, dan *Naïve Bayes*. Metode *bagging* menggunakan SVM sebagai model dasar, yang menunjukkan kinerja baik kedua. Metode *AdaBoost* menggunakan *Decision Tree* sebagai model dasar, yang diberi peringkat berkinerja baik di antara semua metode pembelajaran ansambel yang diuji. Hasilnya dapat dilihat pada Tabel 3.

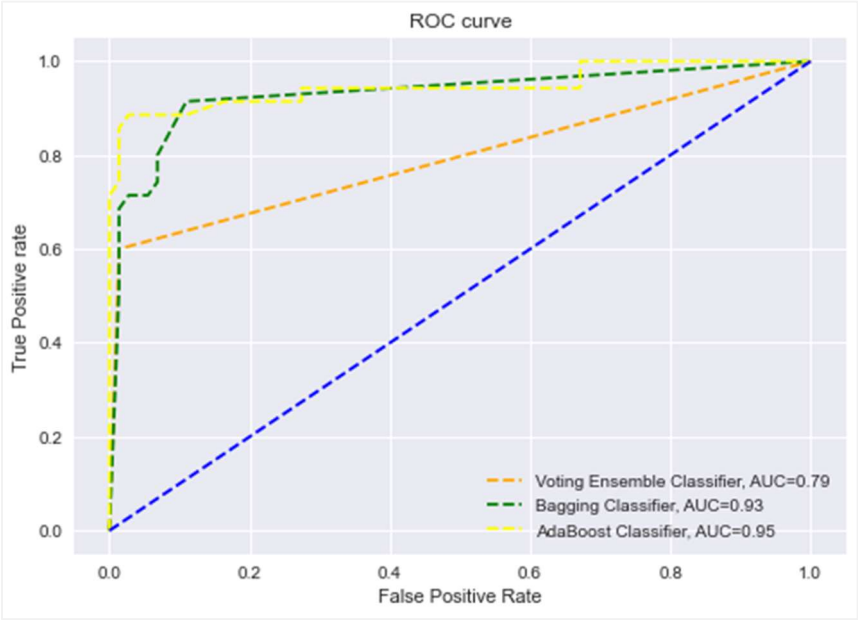
Tabel 3. Evaluasi kinerja metode 4 - 6

Metode	*Accuracy*	*Precision*	*Recall*	*F1-Score*
Voting Classifier	89.77%	95.45%	60%	73.68%
Bagging	90.70%	92.59%	71.43%	80.65%
AdaBoost	91.65%	96.77%	85.71%	90.90%

3.3. Evaluasi Kinerja Model

Memvisualisasikan AUC (Area Under the Curve) dapat membantu menentukan kinerja model klasifikasi biner. AUC adalah ukuran yang baik untuk menilai kinerja

model, dengan nilai berkisar antara 0 hingga 1. Nilai yang mendekati 1 menunjukkan kinerja yang baik, sedangkan nilai yang mendekati 0 menunjukkan kinerja yang buruk.



Gambar 3. Hasil area *under the curve*

Pada Gambar 3 terlihat bahwa model klasifikasi AdaBoost mempunyai nilai model ansambel terbaik dengan nilai AUC tertinggi sebesar 0,95. Penggunaan area under curve ini sangat membantu dalam memilih model terbaik. Dari hasil yang didapatkan dapat ditentukan bahwa metode AdaBoost mempunya nilai tertinggi yang dapat digunakan untuk melakukan deteksi malware menggunakan ensamble learning.

4. Pembahasan

Penelitian ini mengusulkan penggunaan algoritma pembelajaran *ansambel AdaBoost* untuk mendeteksi *malware* pada aplikasi *Android* berbasis izin. Hasil penelitian menunjukkan bahwa *AdaBoost* mencapai akurasi 91,65%, menjadikannya sebagai algoritma dengan performa baik dibandingkan dengan metode klasifikasi lainnya seperti SVM, *Naïve Bayes*, dan *Decision Tree*.

Penelitian sebelumnya telah menunjukkan bahwa metode pembelajaran *ansambel*, seperti *AdaBoost*, sering kali unggul dalam mengatasi masalah klasifikasi yang kompleks karena kemampuannya untuk menggabungkan kekuatan berbagai algoritma. Hasil ini konsisten dengan temuan sebelumnya yang menunjukkan bahwa *AdaBoost* efektif dalam meningkatkan akurasi klasifikasi dengan mengurangi bias dan *varians*.

Hipotesis kerja dalam penelitian ini adalah bahwa menggabungkan beberapa algoritma klasifikasi yang memiliki kinerja individu rendah dapat menghasilkan sistem deteksi *malware* yang lebih efektif. Hasil yang menunjukkan kinerja *AdaBoost* dengan akurasi 91,65% mendukung hipotesis ini, karena *AdaBoost* berhasil meningkatkan kinerja deteksi *malware* dengan menggabungkan kekuatan dari berbagai model dasar. Temuan ini juga menunjukkan bahwa pendekatan *ansambel* tidak hanya meningkatkan akurasi, tetapi juga memberikan solusi yang stabil terhadap data yang tidak seimbang dan fitur-fitur yang beragam, seperti izin aplikasi pada perangkat *Android*.

Dengan kinerja yang baik dalam mendeteksi *malware*, sistem berbasis *AdaBoost* dapat memberikan perlindungan yang lebih efektif terhadap ancaman keamanan di perangkat *Android* Dataset-nya adalah izin aplikasi Android, Proses ekstraksi fitur berdasarkan permissions hanya relevan untuk Android, Aplikasi Android lebih rentan terhadap malware

karena ekosistemnya lebih terbuka dan jika diterapkan di iOS kendalanya Dataset baru dari iOS (yang sulit diperoleh), Fitur baru selain izin (misalnya API behavior, network activity), Adaptasi framework ML ke CoreML dan tunduk pada aturan Apple yang ketat. . Hal ini sangat penting mengingat bahwa *Android* mendominasi pasar *smartphone* dan sering menjadi target serangan *malware*. Peningkatan akurasi deteksi *malware* dapat mengurangi risiko data pribadi pengguna yang dicuri atau disalahgunakan, sehingga meningkatkan keamanan dan kepercayaan pengguna terhadap aplikasi *mobile*.

Temuan ini mendemonstrasikan potensi metode *ansambel* dalam meningkatkan akurasi model klasifikasi dalam konteks deteksi *malware*. Ini dapat mendorong pengembangan lebih lanjut dari teknik *ansambel* dan penerapannya dalam berbagai domain lain yang memerlukan deteksi ancaman atau anomali.

Meskipun *AdaBoost* dapat meningkatkan kinerja model dengan menekankan contoh yang sulit, namun dapat terpengaruh oleh kelas yang tidak seimbang. Ketika satu kelas mendominasi, *AdaBoost* cenderung memfokuskan perhatian pada kelas mayoritas,saran penelitian berikut yaitu dapat mengeksplorasi penggunaan teknik *oversampling* atau *undersampling* untuk mengatasi masalah ketidakseimbangan data sebelum menerapkan *AdaBoost*. Selain itu, mengembangkan modifikasi *AdaBoost* yang secara eksplisit dirancang untuk menangani ketidakseimbangan kelas dapat menjadi fokus penelitian.

5. Kesimpulan

Sistem Sistem operasi Android, yang digunakan oleh sekitar 70% pasar smartphone global, menghadapi risiko ancaman keamanan berupa malware yang memanfaatkan izin aplikasi. Penelitian ini berhasil mengembangkan sistem deteksi malware berbasis izin menggunakan algoritma pembelajaran ansambel AdaBoost yang menunjukkan akurasi sebesar 91,65%, menandakan performa yang unggul dalam mendeteksi malware pada aplikasi Android. Implikasi dari hasil ini adalah bahwa metode ansambel seperti AdaBoost dapat meningkatkan efektivitas sistem keamanan dengan memberikan deteksi malware yang lebih akurat dan andal, sehingga sistem ini berpotensi diterapkan sebagai solusi proteksi bagi pengguna dan pengembang aplikasi Android. Namun, untuk pengembangan lebih lanjut diperlukan pengujian pada dataset yang lebih besar dan beragam, termasuk aplikasi dari berbagai sumber dan jenis izin, agar model dapat digeneralisasikan ke aplikasi dunia nyata yang lebih luas. Selain itu, diperlukan upaya untuk mengatasi masalah ketidakseimbangan data dengan teknik oversampling atau undersampling serta eksplorasi penggunaan fitur tambahan selain izin aplikasi, seperti perilaku runtime atau metadata aplikasi, untuk meningkatkan akurasi deteksi. Pengembangan juga dapat diarahkan pada adaptasi sistem ke platform lain seperti iOS dengan penyesuaian fitur yang relevan serta implementasi sistem secara real-time untuk menguji performa dalam lingkungan operasional nyata. Dengan langkah-langkah tersebut, sistem deteksi malware dapat menjadi lebih efektif, adaptif, dan mampu memberikan perlindungan optimal terhadap ancaman keamanan yang terus berkembang di dunia smartphone.

Kekurangan dari penelitian ini adalah dalam deskripsi metode machine learning belum disertakan informasi terkait pengaturan hyperparameter yang digunakan pada setiap model. Penjelasan mengenai hyperparameter ini penting untuk memberikan gambaran lebih lengkap tentang proses pelatihan model dan memungkinkan reproduksi serta optimasi lebih lanjut oleh peneliti lain

Ucapan Terima Kasih: Kami mengucapkan terima kasih yang sebesar-besarnya kepada semua pihak yang telah mendukung dan berkontribusi dalam penelitian ini mengenai deteksi *malware* pada perangkat *Android* menggunakan algoritma pembelajaran *ansambel AdaBoost* dan kami berharap masukan dari para ahli di bidang keamanan siber dan pembelajaran mesin untuk mendukung kualitas penelitian dan sistem yang lebih baik dan handal untuk menciptakan Indonesia yang lebih maju dalam keamanan siber dan teknologi.

Referensi

[1] L. Lutfiah, "Aplikasi Kamus Simplisia Dan Resep Obat Tradisional (Sidota) Berbasis Android," *J. Sains dan Inform.*, vol. 8, no. 1, pp. 61–69, 2022, https://doi.org/10.34128/jsi.v8i1.369.

[2] J. Hutagalung, P. S. Ramadhan, and S. J. Sihombing, "Keamanan Data Menggunakan Secure Hashing Algorithm (SHA)-256 dan Rivest Shamir Adleman (RSA) pada Digital Signature," *J. Teknol. Inf. dan Ilmu Komput.*, vol. 10, no. 6, 2023, https://doi.org/10.25126/jtiik.1067319.

[3] R. Cabral, J. T. McDonald, L. M. Hively, and R. G. Benton, "Profiling CPU Behavior for Detection of Android Ransomware," in *Conference Proceedings - IEEE SOUTHEASTCON*, 2022. https://doi.org/10.1109/SoutheastCon48659.2022.9764053.

[4] F. Ullah, A. Alsirhani, M. M. Alshahrani, A. Alomari, H. Naeem, and S. A. Shah, "Explainable Malware Detection System Using Transformers-Based Transfer Learning and Multi-Model Visual Representation," *Sensors*, vol. 22, no. 18, 2022, https://doi.org/10.3390/s22186766.

[5] V. Kouliaridis, K. Barmpatsalou, G. Kambourakis, and S. Chen, "A survey on mobile malware detection techniques," *IEICE Trans. Inf. Syst.*, vol. E103D, no. 2, 2020, https://doi.org/10.1587/transinf.2019INI0003.

[6] S. H. Khan *et al.*, "A new deep boosted CNN and ensemble learning based IoT malware detection," *Comput. Secur.*, vol. 133, 2023, https://doi.org/10.1016/j.cose.2023.103385.

[7] P. Mishra, A. Biancolillo, J. M. Roger, F. Marini, and D. N. Rutledge, "New data preprocessing trends based on ensemble of multiple preprocessing techniques," 2020. https://doi.org/10.1016/j.trac.2020.116045.

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

[9] A. Mohammed and R. Kora, "A comprehensive review on ensemble deep learning: Opportunities and challenges," 2023. https://doi.org/10.1016/j.jksuci.2023.01.014.

[10] S. F. R. Roradi and I. R. Mutiaz, "Design of Borneo Virtual Tour Website as a Media for Promotion of Dayak Cultural Tourism Objects, Pampang Village Samarinda," in *Proceedings of the ICON ARCCADE 2021: The 2nd International Conference on Art, Craft, Culture and Design (ICON-ARCCADE 2021)*, 2022. https://doi.org/10.2991/assehr.k.211228.039.

[11] J. Q. Guan, L. H. Wang, Q. Chen, K. Jin, and G. J. Hwang, "Effects of a virtual reality-based pottery making approach on junior high school students' creativity and learning engagement," *Interact. Learn. Environ.*, vol. 31, no. 4, 2023, https://doi.org/10.1080/10494820.2021.1871631.

[12] P. Bhattacharya *et al.*, "Coalition of 6G and Blockchain in AR/VR Space: Challenges and Future Directions," *IEEE Access*, vol. 9, 2021, https://doi.org/10.1109/ACCESS.2021.3136860.

[13] V. B. -, Z. A. -, and E. P. -, "A Study on the Impact of Destination Image on Customer Value, Tourism Satisfaction, and Behavioral Intention in Jatim Park 2," *Int. J. Multidiscip. Res.*, vol. 6, no. 1, 2024, https://doi.org/10.36948/ijfmr.2024.v06i01.12899.

[14] A. Sangal and H. K. Verma, "A Static Feature Selection-based Android Malware Detection Using Machine Learning Techniques," in *Proceedings - International Conference on Smart Electronics and Communication, ICOSEC 2020*, 2020. https://doi.org/10.1109/ICOSEC49089.2020.9215355.

[15] M. Gopinath and S. C. Sethuraman, "A comprehensive survey on deep learning based malware detection techniques," 2023. https://doi.org/10.1016/j.cosrev.2022.100529.

[16] H. Alamro, W. Mtouaa, S. Aljameel, A. S. Salama, M. A. Hamza, and A. Y. Othman, "Automated Android Malware Detection Using Optimal Ensemble Learning Approach for Cybersecurity," *IEEE Access*, vol. 11, 2023,: https://doi.org/10.1109/ACCESS.2023.3294263.

[17] A. Taha and O. Barukab, "Android Malware Classification Using Optimized Ensemble Learning Based on Genetic Algorithms," *Sustain.*, vol. 14, no. 21, 2022, https://doi.org/10.3390/su142114406.

[18] X. Wang, L. Zhang, K. Zhao, X. Ding, and M. Yu, "MFDroid: A Stacking Ensemble Learning Framework for Android Malware Detection," *Sensors*, vol. 22, no. 7, 2022, https://doi.org/10.3390/s22072597.

[19] A. O. Christiana, B. A. Gyunka, and A. N. Oluwatobi, "Optimizing android malware detection via ensemble learning," *Int. J. Interact. Mob. Technol.*, vol. 14, no. 9, 2020, https://doi.org/10.3991/ijim.v14i09.11548.

[20] 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, pp. 1–34, 2022, https://doi.org/10.1007/s10922-021-09634-4.

[21] T. Yuniati, A. R. Tambunan, and Y. A. Setyoko, "Implementasi Static Analysis Dan Background Process Untuk Mendeteksi Malware Pada Aplikasi Android Dengan Mobile Security Framework," *LEDGER J. Inform. Inf. Technol.*, vol. 1, no. 2, 2022, https://doi.org/10.20895/ledger.v1i2.848.

[22] N. Anwar, S. A. Akbar, A. Azhari, and I. Suryanto, "Ekstraksi Logis Forensik Mobile pada Aplikasi E-Commerce Android," *Mob. Forensics*, vol. 2, no. 1, 2020, https://doi.org/10.12928/mf.v2i1.1791.

[23] E. Prabakaran and K. Pillay, "Nanomaterials for latent fingerprint detection: A review," 2021. https://doi.org/10.1016/j.jmrt.2021.03.110.

[24] N. E. Goldameir, A. M. Yolanda, A. Adnan, and L. Febrianti, "Classification of the Human Development Index in Indonesia Using the Bootstrap Aggregating Method," *SinkrOn*, vol. 6, no. 1, 2021, https://doi.org/10.33395/sinkron.v6i1.11173.

[25] Y. Resti, C. Irsan, J. F. Latif, I. Yani, and N. R. Dewi, "A Bootstrap-Aggregating in Random Forest Model for Classification of Corn Plant Diseases and Pests," *Sci. Technol. Indones.*, vol. 8, no. 2, 2023, https://doi.org/10.26554/sti.2023.8.2.288-297.
[26] R. Sibindi, R. W. Mwangi, and A. G. Waititu, "A boosting ensemble learning based hybrid light gradient boosting machine and extreme gradient boosting model for predicting house prices," *Eng. Reports*, vol. 5, no. 4, 2023, https://doi.org/10.1002/eng2.12599.
[27] S. R. Sharma, B. Singh, and M. Kaur, "A Novel Approach of Ensemble Methods Using the Stacked Generalization for High-dimensional Datasets," *IETE J. Res.*, vol. 69, no. 10, 2023, https://doi.org/10.1080/03772063.2022.2028582.
[28] S. Marukatat, "Tutorial on PCA and approximate PCA and approximate kernel PCA," *Artif. Intell. Rev.*, vol. 56, no. 6, 2023, https://doi.org/10.1007/s10462-022-10297-z.