P-ISSN : 2549-7480
E-ISSN : 2549-6301

JURNAL ILMIAH INFORMATIKA
E-ISSN : 2549-6301
ISSN : 2549-7480
Faculty of Sains and Technology, Ibrahimy University
S4-Accredited : 105/E/KPT/2022
Journal page is available to : https://journal.ibrahimy.ac.id/index.php/JIMI

ANALISIS EFEKTIVITAS ALGORITMA *MACHINE LEARNING* DALAM DETEKSI *MALWARE ANDROID* DENGAN *STATISTICAL TESTS*

Faisal Abdussalam[1)], Alam Rahmatulloh[*1)]

[1] *Informatika, Fakultas Teknik, Universitas Siliwangi*

email: [1] 217006090@student.ac.id, [1] alam@unsil.ac.id

ARTICLE INFO

Article History:
Recieved : 25 Mei 2024
Accepted : 24 November 2024
Published : 13 Desember 2024

Keywords:

Decision Tree
Machine learning
Malware
Neural Network
K-Nearest NeighborI

IEEE style in citing this article:
F. Abdussalam, A. Rahmatulloh, "Analisis Efektivitas Algoritma *Machine Learning* Dalam Deteksi *Malware Android* Dengan *Statistical Tests*", *jurnal.ilmiah.informatika*, vol. 9, no. 2, pp. 124-134, Des. 2024.

ABSTRACT

Because there are so many harmful apps on the market, Android malware detection has emerged as a crucial cybersecurity concern. Using the Drebin dataset, which comprises 5,560 malicious and 9,476 benign applications, this study attempts to assess how well three machine learning algorithms—Neural Network, K-Nearest NeighborI (K-NN), and Decision Tree—detect malware on the Android operating system. The method used involves testing the performance of the three algorithms based on Accuracy, Precision, Recall, and F1-Score. The results show that Neural Network has the highest Accuracy at 98.7%, followed by K-NN with 97.7%, and Decision Tree with 97.5%. Neural Network also excels in Precision, Recall, and F1-Score with values of 98.4%, 98.1%, and 98.2%, respectively. Although the performance differences between the algorithms are not statistically significant, the results indicate that Neural Network offers the best solution for Android malware detection. This study provides guidance in selecting the appropriate algorithm for Android -based security systems.

P-ISSN : 2549-7480
F. Abdussalam dkk/ JIMI 9 (2) pp. 124-133 E-ISSN : 2549-6301

1. PENDAHULUAN

Setiap tahun, teknologi terus berkembang untuk menjadikan kehidupan manusia lebih mudah. Perkembangan termasuk kemampuan untuk terhubung melalui internet melalui sistem operasi *Android*, yang saat ini merupakan sistem operasi mobile yang paling umum, dengan lebih dari 2 miliar perangkat yang aktif. Pada tahun 2019, *Android* menguasai lebih dari 74% pengguna *smartphone* di seluruh dunia, memberikan kontribusi yang signifikan terhadap pasar seluler. Namun, popularitas *Android* juga memiliki efek negatif, salah satunya adalah serangan *malware* [1].

Malware, juga dikenal sebagai *malicious malicious*, adalah program atau kode berbahaya yang dapat mengakibatkan kerugian bagi orang atau perusahaan. *Malware* dapat menyebabkan kerugian materi atau moneter. Dengan memahami cara kerja dan karakteristik *malware*, analisis *malware* dapat digunakan untuk mencegah insiden *malware* [2]. Serangan *phishing* melalui *email*, rekayasa sosial, dan pengunduhan perangkat lunak yang tidak aman adalah beberapa cara *malware* dapat menyebar.

Malware merupakan masalah yang terus berkembang, terutama pada platform seluler, karena meningkatnya toko aplikasi dan perangkat seluler terkait. Terlalu banyak aplikasi baru untuk memeriksa secara manual setiap aplikasi untuk mengetahui perilaku jahatnya [3]. Serangan *phishing*, rekayasa sosial, dan pengunduhan perangkat lunak yang tidak aman adalah beberapa cara *malware* dapat menyebar.

Malware Android dapat menyerang dengan berbagai cara, termasuk dengan memanfaatkan fitur izin aplikasi *Android* [4]. *Permissions* pada *Android* merupakan serangkaian aturan dan batasan yang diterapkan untuk menentukan apakah sebuah aplikasi dapat mengakses sumber daya dan fitur tertentu pada perangkat. Namun, banyak pengguna tidak memahami arti dari setiap izin, sehingga mereka sering kali memberikan izin aplikasi untuk mengakses data pribadi tanpa pertimbangan yang matang [5].

Selain itu, banyak pengembang aplikasi *Android* tidak memanfaatkan Antarmuka Program Aplikasi (API) dengan aman, sehingga aplikasi tersebut berpotensi menyalahgunakan data pribadi. Ditambah lagi, beberapa fitur keamanan bawaan *Android* tidak cukup kuat untuk melindungi data pribadi pengguna, yang menyebabkan aplikasi yang sebenarnya tidak berbahaya bisa secara tidak sengaja mengekspos data sensitif [6]. Algoritma pembelajaran mesin adalah metode umum untuk mengidentifikasi dan mendeteksi *malware Android*. Kemampuannya untuk belajar dari data menggunakan algoritma dan kemudian membuat keputusan berdasarkan pola yang ditemukan dalam data membuat machine learning menjadi cabang kecerdasan buatan (AI) yang paling populer [7].

Machine learning (ML) memungkinkan sistem untuk mempelajari data secara mandiri dan membuat keputusan berdasarkan pola yang ada. Proses pembelajaran ini bergantung pada data pelatihan untuk menghasilkan prediksi yang akurat [8]. Sebagai konsep komputasi, *Machine learning* menggunakan algoritma matematika dan komputer untuk mengenali pola dalam data serta memprediksi hasil di masa depan. Prosesnya terdiri dari dua tahap utama, yaitu pelatihan (*training*) dan pengujian (*testing*), dengan tiga kategori utama *Supervised Learning, Unsupervised Learning*, dan *Reinforcement Learning*.

Berbagai penelitian telah dilakukan untuk mengatasi ancaman *malware* pada

DOI : https://doi.org/10.35316/jimi.v9i2.124-133

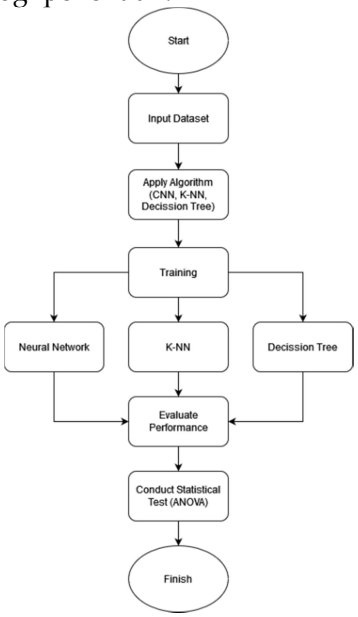
perangkat *Android* menggunakan metode *Machine learning* dan *deep learning*. Purnama et al. (2024) mengusulkan penggunaan metode *Deep Neural Network* (DNN) untuk mendeteksi *malware* ransomware pada perangkat *Android* [9]. Dengan menggunakan dataset *CIC-InvesAndMal2019*, metode ini berhasil mencapai tingkat akurasi hingga 96,6% dalam mendeteksi *malware Android*.

Selain itu, Karfindo et al. (2024) mengeksplorasi teknik *ensemble learning* menggunakan pendekatan *soft voting* yang menggabungkan model seperti *Random Forest*, *Gradient Boosting*, dan *XGBoost* [10]. Penelitian ini menunjukkan bahwa pendekatan *ansambel* mampu meningkatkan akurasi deteksi *malware* hingga 90% menggunakan dataset *KronoDroid*, membuktikan efektivitasnya dalam menangani kompleksitas *malware Android*.

Studi ini membandingkan *Neural Network, K-Nearest Neighbor*, dan *Decision Tree* dalam mendeteksi *malware* pada dataset aplikasi *Android*. Penelitian ini mengevaluasi efektivitas setiap algoritma pada kumpulan data yang terdiri dari aplikasi *Android* yang dikategorikan sebagai *benign* (tidak berbahaya) atau *malware*. Diharapkan penelitian ini akan memberikan kontribusi yang signifikan terhadap pengembangan sistem keamanan *Android* serta meningkatkan perlindungan pengguna dari *malware*. Penentuan algoritma yang paling efektif diharapkan menjadi dasar untuk pengembangan solusi keamanan yang lebih baik pada platform *Android*.

2. METODE PENELITIAN

Tujuan penelitian dicapai melalui serangkaian tindakan sistematis yang dikenal sebagai metode penelitian, yaitu menganalisis dan membandingkan efektivitas tiga algoritma *Machine learning* dalam mendeteksi *malware Android*. Proses perancangan penelitian dimulai dengan membuat diagram blok sistem yang mencakup seluruh sistem. Diagram blok yang lengkap membantu memastikan bahwa sistem beroperasi dengan benar. Gambar 1 menunjukkan *flowchart* metodologi penelitian.



Gambar 1. Metodologi Penelitian

A. *Data Mining*

Data mining proses eksplorasi data yang digunakan untuk menggali pola, membuat prediksi, membuat estimasi, atau mengelompokkan data dengan menggunakan metode dan algoritma tertentu. Berbagai disiplin ilmu sering menggunakan *Data Mining* untuk membantu menyelesaikan masalah yang berkaitan dengan pengolahan data. *Data Mining* sangat penting dalam situasi seperti ini untuk menemukan wawasan baru dan memanfaatkan potensi data yang ada [11].

B. Data Set

Dataset *Drebin*, yang diakses melalui *Kaggle*, digunakan dalam penelitian ini, terdiri dari 5.560 aplikasi *Android* yang dikategorikan sebagai *malware* dan 9.476 aplikasi *benign* (tidak berbahaya). Dataset ini berisi berbagai fitur yang mencerminkan karakteristik aplikasi

P-ISSN : 2549-7480
F. Abdussalam dkk/ JIMI 9 (2) pp. 124-133 E-ISSN : 2549-6301

Android yang digunakan untuk mengklasifikasikan apakah suatu aplikasi adalah *malware* atau bukan. Fitur-fitur tersebut mencakup berbagai aspek seperti izin aplikasi (*Permissions*), panggilan *API* (*API calls*), dan banyak lagi, yang akan membantu algoritma untuk mempelajari pola yang relevan. Dataset ini menjadi fondasi untuk melatih dan menguji efektivitas tiga algoritma utama dalam mendeteksi *malware*, yaitu *Neural Network*, *K-Nearest NeighborI* (K-NN), dan *Decision Tree*. Setelah dataset dimasukkan, tahap selanjutnya adalah mengaplikasikan algoritma untuk memulai proses pelatihan.

C. *Apply Algorithm*

Pada tahap ini, dataset yang telah dimasukkan diproses menggunakan tiga algoritma pembelajaran mesin utama: *Neural Network*, *K-Nearest NeighborI* (K-NN), dan *Decision Tree*. Masing-masing algoritma bekerja dengan pendekatan yang berbeda untuk memproses dan menganalisis data dalam rangka mendeteksi *malware Android*. Berikut penjelasan lebih detail mengenai setiap algoritma.

D. *Training*

Pembelajaran mesin adalah proses pelatihan yang menggunakan algoritma yang parameternya disesuaikan dengan data yang diberikan. Otak manusia mengalami perubahan sebagai hasil dari belajar, jadi analoginya mirip. Untuk memastikan bahwa model yang dibuat berfungsi dengan baik, pelatihan pembelajaran mesin sangat penting. Untuk memahami informasi yang terkandung dalam data, model pembelajaran mesin belajar atau melatih diri dari data yang diberikan. Penelitian ini menggunakan metode split *training-test*, yang membagi dataset menjadi dua bagian 80% data digunakan untuk melatih model (set pelatihan), dan 20% digunakan untuk menguji model. Tujuan dari pembagian ini adalah memastikan bahwa model dapat belajar dari cukup banyak data sambil memberikan set data yang berbeda untuk evaluasi kinerjanya.

Setiap algoritma, yaitu *Neural Network*, *K-Nearest NeighborI*, dan *Decision Tree*, dilatih menggunakan data *training* dan dievaluasi berdasarkan data *testing*. Hasil evaluasi dari setiap algoritma kemudian dibandingkan untuk menentukan algoritma mana yang paling efektif dalam mendeteksi *malware* pada aplikasi *Android*.

E. *Neural Network*

Algoritma bekerja dengan sistem pemrosesan informasi yang mirip dengan kotak yang menerima masukan dan mengeluarkan keluaran. Beberapa komponen pemrosesan dan bobotnya saling terhubung satu sama lain [12]. Metode pembelajaran mesin yang menggunakan algoritma jaringan saraf berdasarkan bagaimana jaringan *neuron* disusun dan bergerak di otak. Metode-metode ini menggunakan model *neuron* yang sangat idealis.

Pada Penelitian ini *Neural Network*, melalui proses pelatihan, menggunakan jaringan saraf tiruan yang terdiri dari *neuron-neuron* yang saling terhubung. Pada fase prediksi, data uji yang dimasukkan ke dalam jaringan akan diproses melalui lapisan-lapisan *neuron* yang telah dioptimalkan. Setiap *input* data dipetakan ke *output* prediksi apakah aplikasi tersebut *malware* atau *benign*, berdasarkan bobot yang dihasilkan dari pelatihan.

F. *K-Nearest NeighborI*

Untuk melakukan klasifikasi, algoritma *K-Nearest NeighborI* (KNN) menggunakan jarak yang ada antara data uji dan pelatihan. KNN mencari k data pelatihan yang berada di jarak terdekat dengan data uji, dan kemudian melakukan klasifikasi berdasarkan

DOI : https://doi.org/10.35316/jimi.v9i2.124-133

mayoritas kelas dari data terdekat tersebut. Untuk menghitung jarak, rumus seperti jarak *Euclidean* dan *Minkowski* digunakan. Rumus *Euclidean* biasanya digunakan karena sangat akurat [13].

Algoritma K-NN siap untuk memprediksi kelas aplikasi dalam data uji dengan menghitung jarak antara data uji dan seluruh data pelatihan. Data latihan dan kelasnya biasanya disimpan dalam KNN. Algoritma ini mengklasifikasikan objek baru berdasarkan nilai atributnya dan data latihan saat menerima data baru. Tujuan algoritma ini adalah untuk mengklasifikasikan objek baru berdasarkan atribut nilai dan informasi yang ada pada data latihan saat ini [14]. K-NN akan mencari k-tetangga terdekat untuk setiap aplikasi yang diuji dan menentukan klasifikasinya (*malware* atau *benign*) berdasarkan mayoritas kelas dari tetangga terdekat tersebut. Semakin banyak tetangga yang termasuk ke dalam kelas yang sama, semakin besar kemungkinan aplikasi uji diklasifikasikan sebagai kelas tersebut.

G. *Decision Tree*

Dalam model prediksi yang diawasi, pohon keputusan digunakan pohon ini terdiri dari node yang menunjukkan keputusan dan cabang yang menunjukkan konsekuensi dari keputusan tersebut.[15]. Model *Decision Tree* yang sudah dilatih memiliki struktur pohon keputusan yang siap digunakan untuk memprediksi aplikasi di data uji. Ketika data uji dimasukkan ke dalam model ini, algoritma akan mengikuti cabang-cabang dalam pohon berdasarkan fitur-fitur aplikasi, seperti izin yang diminta, aktivitas yang dilakukan, atau sumber daya yang diakses. Setiap keputusan di simpul pohon akan membawa data ke cabang berikutnya hingga mencapai daun (*leaf node*), di mana prediksi *malware* atau *benign* ditentukan.

H. *Evaluate Performance*

Tujuan evaluasi secara keseluruhan adalah untuk menguji kemampuan sistem untuk mendeteksi *malware* pada platform *Android* dengan menggunakan teknologi pengajaran mesin dan bahasa pemrograman *Python*. Tujuan evaluasi adalah untuk memastikan bahwa program berjalan sesuai dengan perancangan yang telah dirancang dan untuk mengukur tingkat kemungkinan kesalahan program.

Tabel 1. *Confusion Matrix*

	***False Positive* (FP)**	***False Negative* (FN)**
True Positive	TP	FN
True Negative	FP	TN

Komponen utama dalam menentukan akurasi, *presisi*, dan *recall* adalah *Confusion Matrix*. Akurasi adalah perbandingan dari prediksi yang benar yang diklasifikasikan oleh sistem. Untuk membandingkan kinerja ketiga algoritma, digunakan beberapa metrik evaluasi, antara lain:

1. Akurasi

Persentase prediksi yang benar dari semua prediksi yang dibuat model menggunakan rumus:

$$\text{Akurasi} = \frac{\text{T+TN}}{\text{TP+TN+FPFN}} \quad (1)$$

2. *Precision*

Jumlah *malware* yang terdeteksi dengan benar (*true positives*) dibandingkan dengan semua deteksi *malware* dihitung dengan rumus berikut:

$$Precision = \frac{\text{TP}}{\text{TP+FN}} \quad (2)$$

3. *Recall* (*Sensitivity*)

Dengan menggunakan rumus ini, rasio antara jumlah *malware* yang benar-benar terdeteksi dan total jumlah *malware* dalam dataset dapat dihitung sebagai berikut:

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

DOI : https://doi.org/10.35316/jimi.v9i2.124-133

P-ISSN : 2549-7480
F. Abdussalam dkk/ JIMI 9 (2) pp. 124-133 E-ISSN : 2549-6301

4. *F1-Score*

Harmonic mean of Recall and Precision, yang menunjukkan keseimbangan antara keduanya dengan rumus:

$$F1 - Score = 2X \frac{Precision\ X\ Recall}{Precision + Recall} \quad (4)$$

I. Conduct Statistical Test (ANOVA)

Setelah kinerja dari masing-masing algoritma diukur, dilakukan uji statistik, seperti *ANOVA (Analysis of Variance*), untuk membandingkan apakah ada perbedaan signifikan antara performa ketiga algoritma tersebut. Uji statistik ini digunakan untuk menentukan apakah perbedaan hasil kinerja dari masing-masing algoritma bersifat signifikan atau hanya kebetulan. Untuk menentukan apakah terdapat perbedaan signifikan antara kinerja ketiga algoritma, dilakukan uji statistik menggunakan *Analysis of Variance (ANOVA)*. Langkah-langkah dalam uji *ANOVA*:

1. Penentuan Hipotesis

Hipotesis Nol (H0) menyatakan bahwa kinerja ketiga algoritma tidak berbeda secara signifikan dari rata-rata. Hipotesis Alternatif (H1) menyatakan bahwa setidaknya satu dari ketiga algoritma memiliki perbedaan yang signifikan.

2. Penghitungan *ANOVA*

ANOVA membandingkan varians dalam kelompok (dalam setiap algoritma) dan varians antar kelompok. Rumus dasar *ANOVA* adalah sebagai berikut:

$$\mathrm{F1} = \frac{\text{Varians antar kelompok}}{\text{Varians dalam kelompok}} \quad (5)$$

3. *P-Value*

Hasil *ANOVA* menghasilkan nilai p, juga dikenal sebagai nilai p, yang menunjukkan kemungkinan bahwa hasil yang diamati terjadi secara kebetulan. Jika nilai p lebih kecil dari tingkat signifikansi yang ditetapkan, misalnya 0,05, hipotesis nol ditolak.

4. Uji *Post-Hoc*

Langkah selanjutnya adalah melakukan uji *Post-Hoc*, seperti *Tukey's HSD (Honestly Significant Difference*), untuk mengetahui algoritma mana yang memiliki performa terbaik secara signifikan. Uji ini membandingkan semua pasangan algoritma untuk menunjukkan perbedaan rata-rata yang signifikan antara mereka. Hasil *ANOVA* menunjukkan perbedaan yang signifikan.

3. HASIL DAN PEMBAHASAN

Berdasarkan hasil pengujian yang telah dilakukan terhadap tiga model algoritma yaitu *Neural Network* (NN), *K-Nearest NeighborI* (KNN), dan *Decision Tree* (DT), didapatkan beberapa kesimpulan terkait performa masing-masing model serta hasil uji statistik untuk membandingkan mereka. Pada bagian ini, hasil yang diperoleh akan dibahas lebih lanjut dengan mengacu pada dua aspek utama yaitu hasil performa model dan hasil uji statistik.

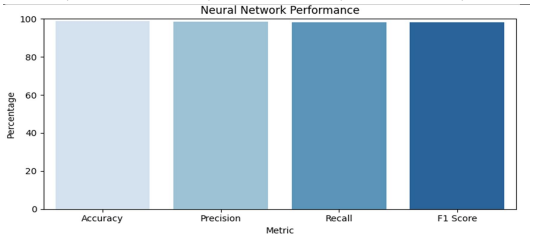
A. Perbandingan Kinerja Algoritma

Setelah menerapkan ketiga algoritma (*Neural Network*, *K-Nearest NeighborI*, dan *Decision Tree*) pada dataset yang telah diproses, hasil evaluasi kinerja untuk masing-masing algoritma diperoleh sebagai berikut:

1. Hasil Pengujian *Neural Network*

Tabel 2. *Classification Report Neural Network*

Accuracy	*Precision*	*Recall*	*Score*
98,7%	98,4%	98,1%	98,2%



Gambar 2. *Classification Report Neural Network*

P-ISSN: 2549-7480
E-ISSN: 2549-6301

Tabel 2 dan gambar 2 menyajikan laporan klasifikasi untuk algoritma *Neural Network* yang menunjukkan kinerja model dalam mendeteksi *malware*. Akurasi model mencapai 98,7%, yang berarti 98,7% dari total prediksi yang dibuat oleh model adalah benar. *Precision*, yang merupakan rasio deteksi *malware* yang benar terhadap semua deteksi *malware*, mencapai 98,4%, menunjukkan bahwa sebagian besar aplikasi yang terdeteksi sebagai *malware* benar-benar merupakan *malware*. *Recall*, atau sensitivitas, sebesar 98,1% menunjukkan bahwa dari semua aplikasi *malware* yang ada, 98,1% berhasil terdeteksi oleh model. Akhirnya, skor F1, yang mencerminkan keseimbangan antara *Precision* dan *Recall*, mencapai 98,2%, menegaskan bahwa model *Neural Network* memiliki kinerja yang sangat baik dalam klasifikasi aplikasi sebagai *benign* atau *malware* .

2. Hasil Pengujian *K- Nearest Neighbor*

Tabel 3. *Classification Report K- Nearest Neighbor*

Accuracy	*Precision*	*Recall*	*Score*
97,7%	97,1%	96,7%	96,9%

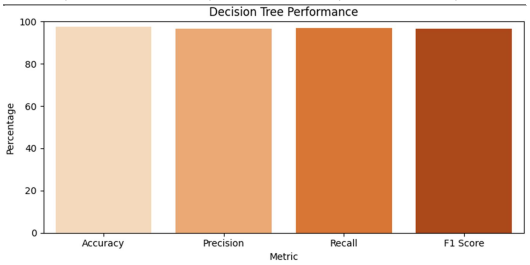
Gambar 3. *Classification Report K- Nearest Neighbor*

Tabel 3 dan gambar 3 menunjukkan hasil *Classification Report* untuk algoritma *K-Nearest NeighborI* (K-NN), dengan metrik evaluasi yang mencakup akurasi, *Precision, Recall*, dan *F1-Score*. Akurasi model mencapai 97,7%, menandakan bahwa persentase prediksi yang benar dari total prediksi yang dilakukan oleh model cukup tinggi. *Precision* sebesar 97,1% menunjukkan bahwa proporsi deteksi *malware* yang benar (*true positives*) terhadap semua deteksi *malware* yang dilakukan model adalah baik, sehingga mengindikasikan rendahnya jumlah *False Positives. Recall* yang dicapai adalah 96,7%, yang berarti model mampu mendeteksi 96,7% dari semua *malware* yang sebenarnya ada dalam dataset. Sementara itu, *F1-Score*, yang merupakan *harmonic mean* dari *Precision* dan *Recall*, berada di angka 96,9%, menggambarkan keseimbangan yang baik antara kedua metrik tersebut. Secara keseluruhan, hasil ini menunjukkan bahwa K-NN merupakan algoritma yang efektif dalam mendeteksi *malware* pada dataset yang digunakan.

3. Hasil Pengujian *Decision Tree*

Tabel 4. *Classification Report Decision Tree*

Accuracy	*Precision*	*Recall*	*Score*
97,5 %	96,5 %	96,9 %	96,7 %



Gambar 4. *Classification Report Decision Tree*

Tabel 4 dan gambar 4 menunjukkan hasil *Classification Report* untuk algoritma *Decision Tree*, yang mencakup metrik evaluasi akurasi, *Precision, Recall*, dan *F1-Score*. Akurasi model tercatat sebesar 97,5%, menunjukkan bahwa persentase prediksi yang benar dari total prediksi yang dilakukan oleh model cukup tinggi. *Precision* mencapai 96,5%, yang berarti proporsi deteksi *malware* yang benar (*true positives*) dibandingkan dengan total deteksi *malware* menunjukkan bahwa model ini memiliki tingkat kesalahan yang rendah dalam mengklasifikasikan aplikasi *benign* sebagai *malware*. Sementara itu, *Recall* berada di angka 96,9%, mengindikasikan bahwa model mampu mendeteksi sebagian besar *malware* yang ada dalam dataset. *F1-Score*, yang merupakan *harmonic mean* dari *precision* dan *recall*, tercatat 96,7%, mencerminkan

P-ISSN : 2549-7480
F. Abdussalam dkk/ JIMI 9 (2) pp. 124-133 E-ISSN : 2549-6301

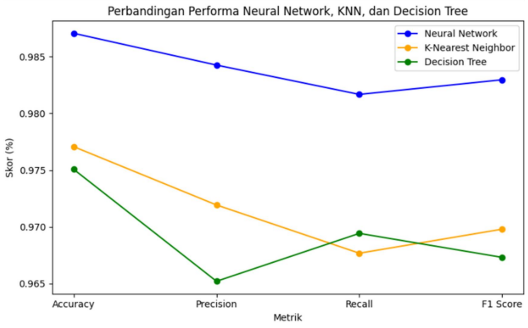
keseimbangan yang baik antara kedua metrik tersebut.

4. Hasil Perbandingan Kinerja Tiga Algoritma

Setelah laporan klasifikasi dan hasil pengujian diterima, langkah berikutnya adalah membandingkan hasil dari ketiga laporan. Tabel 5 menampilkan hasil perbandingan metrik kinerja utama antara tiga algoritma yang digunakan untuk mendeteksi *malware* Neural Network (KNN), dan Decision Tree. Akurasi, presisi, recall, dan skor F1 adalah metrik kinerja yang dibandingkan, dan hasilnya digambarkan dengan lebih jelas di Gambar 5.

Tabel 5. Hasil Perbandingan *Classification Report*

	Neural Network	**KNN**	***Decision Tree***
Accuracy	98,7%	97,7%	97,5%
Precision	98,4%	97,1%	96,5%
Recall	98,1%	96,7%	96,9%
F1 Score	98,2%	96,9%	96,7%



Gambar 5. Hasil Perbandingan *Classification Report*

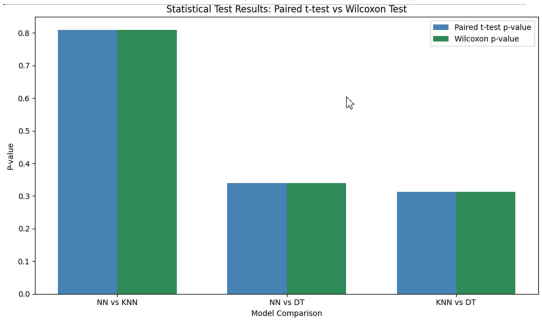
Hasil perbandingan kinerja tiga algoritma, yaitu *Neural Network*, *K-Nearest NeighborI* (KNN), dan *Decision Tree*, dalam deteksi *malware Android* menunjukkan bahwa *Neural Network* memiliki performa terbaik dengan akurasi 98,7%, *Precision* 98,4%, *Recall* 98,1%, dan *F1 Score* 98,2%. KNN berada di posisi kedua dengan akurasi 97,7%, *Precision* 97,1%, *Recall* 96,7%, dan *F1 Score* 96,9%, sementara *Decision Tree* menunjukkan performa paling rendah dengan akurasi 97,5%, *Precision* 96,5%, *Recall* 96,9%, dan *F1 Score* 96,7%. Grafik perbandingan menunjukkan bahwa *Neural Network* unggul di semua metrik, sementara *Decision Tree* berada di posisi terbawah.

B. Hasil *Statistical Test*

Hasil uji statistik menggunakan analisis *ANOVA* dan pengujian statistik lainnya, seperti *paired t-test* dan *Wilcoxon test*, menunjukkan perbandingan kinerja antara ketiga algoritma: *Neural Network* (NN), *K-Nearest NeighborI* (KNN), dan *Decision Tree* (DT).

Tabel 6. *Statistical Test Results*:

Model Comparison	***Paired t-test P-Value***	**Wilcoxon *P-Value***
NN vs KNN	80,8411%	80,8365%
NN vs DT	33,9064%	33,8980%
KNN vs DT	31,2502%	31,2422%



Gambar 6. *Statistical Test Results*

Hasil uji statistik menggunakan *Paired t-test* dan *Wilcoxon Signed-Rank Test* yang membandingkan kinerja *Neural Network* (NN), *K-Nearest NeighborI* (KNN), dan *Decision Tree* (DT) menunjukkan bahwa tidak ada perbedaan signifikan antara ketiga algoritma tersebut. *P-Value* untuk setiap perbandingan (NN vs KNN, NN vs DT, dan KNN vs DT) berada di atas 31%, yang berarti variasi dalam kinerja yang diamati tidak signifikan secara statistik. Dari ketiga pengujian statistik yang dilakukan, *Neural Network* (NN) menunjukkan kinerja yang sedikit lebih unggul dibandingkan dengan *K-Nearest NeighborI* (KNN) dan *Decision Tree* (DT),

meskipun perbedaannya tidak signifikan secara statistik. Hal ini terlihat dari *P-Value* pada perbandingan NN vs KNN yang paling tinggi, yakni 80,8411% untuk *paired t-test* dan 80,8365% untuk *Wilcoxon test*.

Dengan demikian, meskipun NN memiliki kinerja terbaik, pilihan algoritma yang akan digunakan tetap dapat disesuaikan dengan konteks dan kebutuhan spesifik dari aplikasi deteksi *malware* yang ingin dikembangkan.

C. Perbandingan dengan Penelitian Sebelumnya

Evaluasi dilakukan dengan membandingkan beberapa model dari penelitian sebelumnya dalam memprediksi *malware Android*. Model yang dibandingkan mencakup DNN, *Decision Tree*, *XGBoost*, NN, KNN, dan *Random Forest*. Hasil perbandingan menunjukkan bahwa model NN memiliki kinerja yang lebih baik dibandingkan dengan model lain yang digunakan dalam penelitian sebelumnya; ini terbukti dengan peningkatan yang signifikan pada nilai akurasi, *presisi, recall*, dan nilai f1, yang menghasilkan persentase yang lebih tinggi daripada model lain.

Tabel 7. Perbandingan dengan Penelitian Sebelumnya

Research	**[9]**	**[10]**	**[10]**	***Our Research***	***Our Research***	***Our Research***
Model	DNN	*Random Forest*	*XGBoost*	NN	KNN	*Decision Tree*
Accuracy	96.6 %	85 %	85 %	98,7%	97,7%	97,5%
Precision	96,1 %	0.86	0.86	98,4%	97,1%	96,5%
Recall	96,2 %	0.84	0.84	98,1%	96,7%	96,9%
F1-Score	96.2 %	0.85	0.85	98,2%	96,9%	96,7%

4. KESIMPULAN

Perkembangan *malware* yang besar dengan efek yang semakin mengkhawatirkan memaksa para peneliti untuk berpartisipasi. Salah satu kontribusi mereka adalah dengan membuat model-model yang berfungsi untuk menentukan apakah sebuah file termasuk *malware* atau tidak. Dataset publik digunakan dalam penelitian ini dengan melakukan pre-processing agar data siap digunakan. Penelitian ini menunjukkan bahwa algoritma *Neural Network* adalah yang terbaik dalam mendeteksi *malware Android*. Ini mencapai akurasi tertinggi 98,7%, serta skor F1 yang lebih tinggi, *presisi*, dan *Recall*, dibandingkan dengan *K-Nearest NeighborI* (K-NN) dan *Decision Tree*. Meskipun *Neural Network* masih memiliki kinerja yang lebih baik, hasil pengujian statistik menggunakan *Paired t-test* dan *Wilcoxon Signed-Rank Test* menunjukkan bahwa tidak ada perbedaan statistik yang signifikan antara ketiga algoritma. Meskipun pemilihan algoritma harus disesuaikan dengan konteks dan kebutuhan aplikasi tertentu, *Neural Network* disarankan sebagai algoritma utama untuk deteksi *malware* dalam sistem keamanan *Android*. Penelitian ini memberikan wawasan penting tentang seberapa efektif algoritma pembelajaran mesin dalam deteksi *malware*. Ini juga memberi arahan untuk penelitian lebih lanjut yang akan meneliti kombinasi algoritma, pengoptimalan parameter, dan pengujian pada dataset yang lebih beragam untuk meningkatkan deteksi *malware* secara keseluruhan.

5. REFERENSI

[1] A. Nur Iman, A. Budiyono, And A. Almaarif, "Analisis *Malware* Pada Sistem Operasi *Android* Menggunakan Permission-Based *Malware* Analysis In *Android* Operation System Using Permission-Based."

[2] Y. Dwi *Et Al.*, "Analisis *Malware* Menggunakan Metode Analisis

DOI : https://doi.org/10.35316/jimi.v9i2.124-133

P-ISSN : 2549-7480
E-ISSN : 2549-6301

Statis Dan Dinamis Untuk Pembuatan Ioc Berdasarkan Stix Versi 2.1."

[3] N. Mclaughlin *Et Al.*, "Deep *Android Malware* Detection," In *Codaspy 2017 - Proceedings Of The 7th Acm Conference On Data And Application Security And Privacy*, Association For Computing Machinery, Inc, Mar. 2017, Pp. 301–308. Doi: 10.1145/3029806.3029823.

[4] P. Agrawal And B. Trivedi, "Unstructured Data Collection From Apk Files For *Malware* Detection," 2020.

[5] D. Hindarto, R. Eko Indrajit, And E. Dazki, "Perbandingan Kinerja Akurasi Klasifikasi K-Nn, Nb Dan Dt Pada Apk *Android*," Vol. 9, No. 1, Pp. 486–503, 2022, [Online]. Available: Http://Jurnal.Mdp.Ac.Id

[6] Ebtesam J. Alqahtani, Rachid Zagrouba, And Abdullah Almuhaideb, *2019 Sixth International Conference On Malicious Defined Systems (Sds) : 10-13 June, 2019, Rome, Italy*. Ieee, 2019.

[7] D. T. Ananto *Et Al.*, "Prosiding Seminar Nasional Lppm Umj Website: Http://Jurnal.Umj.Ac.Id/Index.Php/ Semnaskat E-Issn: 2714-6286 Edukasi Dan Pelatihan Pengenalan *Machine learning* Dan Computer Vision Untuk Mengeksplorasi Potensi Visual." [Online]. Available: Http://Jurnal.Umj.Ac.Id/Index.Php/ Semnaskat

[8] A. Ramadhan, L. Lindawati, And M. M. Rose, "Komparasi Algoritma *Neural Network* Dan *K-Nearest Neighbor* Dalam Mendeteksi *Malware Android*," *Building Of Informatics, Technology And Science (Bits)*, Vol. 5, No. 1, Jun. 2023, Doi: 10.47065/Bits.V5i1.3538.

[9] B. Purnama *Et Al.*, "Jepin (Jurnal Edukasi Dan Penelitian Informatika) Deteksi *Malware* Ransomware Menggunakan *Deep Neural Network*," 2024.

[10] R. Turaina, R. Saputra, U. Metamedia, And J. Khatib Sulaiman Dalam No, "Optimalisasi Deteksi *Malware* Pada Platform *Android* Dengan Pendekatan Ensemble *Machine learning*," *Jurnal Nasional Komputasi Dan Teknologi Informasi (Jnkti)*, Vol. 7, No. 3, 2024.

[11] N. Nursobah, S. Lailiyah, B. Harpad, And M. Fahmi, "Penerapan Data *Mining* Untuk Prediksi Perkiraan Hujan Dengan Menggunakan Algoritma *K-Nearest Neighbor*," *Building Of Informatics, Technology And Science (Bits)*, Vol. 4, No. 3, Dec. 2022, Doi: 10.47065/Bits.V4i3.2564.

[12] N. Hadianto, H. B. Novitasari, And A. Rahmawati, "Klasifikasi Peminjaman Nasabah Bank Menggunakan Metode *Neural Network*," *Jurnal Pilar Nusa Mandiri*, Vol. 15, No. 2, Pp. 163–170, Sep. 2019, Doi: 10.33480/Pilar.V15i2.658.

[13] V. Alvian, D. Hidayatullah, A. Nilogiri, H. Azizah, And A. Faruq, "Klasifikasi Siswa Berprestasi Menggunakan Metode *K-Nearest Neighbor* (Knn) Pada Sma Negeri 2 Situbondo Classification Of Achieving Students Using *K-Nearest Neighbor* (Knn) Method At Sma Negeri 2 Situbondo," 2022. [Online]. Available: Http://Jurnal.Unmuhjember.Ac.Id/Index.Php/Jst

[14] P. Putra, A. M. H Pardede, And S. Syahputra, "Analisis Metode K-

DOI : https://doi.org/10.35316/jimi.v9i2.124-133

Nearest Neighbour (Knn) Dalam Klasifikasi Data Iris Bunga," *Jurnal Teknik Informatika Kaputama (Jtik)*, Vol. 6, No. 1, 2022.

[15] R. N. Ramadhon, A. Ogi, A. P. Agung, R. Putra, S. S. Febrihartina, And U. Firdaus, "Implementasi Algoritma *Decision Tree* Untuk Klasifikasi Pelanggan Aktif Atau Tidak Aktif Pada Data Bank," 2024.

DOI : https://doi.org/10.35316/jimi.v9i2.124-133