BAB II
TINJAUAN PUSTAKA

2.1 Landasan Teori

2.1.1 *Malware Android*

Malware android adalah perangkat lunak berbahaya yang dirancang khusus untuk merusak atau mengakses perangkat *android* tanpa izin pengguna. *Android* sebagai sistem operasi ponsel pintar yang sangat populer, sering menjadi target utama bagi para penjahat *cyber* yang mencoba menyebarkan *malware* untuk mencuri data pribadi, menyebabkan kerusakan, atau menghasilkan keuntungan finansial. Penyebaran *malware* dapat terjadi melalui berbagai metode, termasuk serangan *phishing* melalui *email*, rekayasa sosial, dan pengunduhan perangkat lunak yang tidak aman. Pernyataan tersebut menunjukkan bahwa bahkan perusahaan seperti *Google*, yang sering berada di garis depan dalam menjaga ekosistem *android* bebas dari *malware*, telah gagal mengambil tindakan untuk mencegah *malware* (Ma et al., 2019).

Berdasarkan prinsip serangannya, *malware android* dapat dibedakan menjadi tiga kategori (Rana et al., 2018):

a Serangan berbasis perangkat keras, biasanya penyerang akan menggunakan perintah atau operasi tertentu untuk merusak perangkat keras, memasukkan *firmware* atau mengubah perangkat keras agar menjadi tidak normal. Tindakan tersebut biasanya terjadi dengan membuat *backdoor*, *intercepting data*, manipulasi perangkat keras, memasukkan *firmware* atau melakukan *cloning* perangkat keras dan layanan.

b Serangan berbasis perangkat lunak, merupakan serangan yang biasanya dilakukan dengan mengunggah aplikasi *android* yang mengandung *malware* ke *mobile app market*, yang nantinya aplikasi tersebut akan di *install* oleh pengguna *android* kedalam perangkat. Tindakan tersebut biasanya bertujuan untuk penyadapan data, membuat *botnet* atau mendapatkan keuntungan berupa uang dari korban.

c Serangan berbasis *firmware*, merupakan motif serangan yang biasanya dilakukan oleh penyerang dengan memodifikasi atau mengubah program perangkat dengan mendapatkan hak kontrol atau *backdoor* sehingga mereka dapat membuat *crash* atau mengambil alih kendali sistem.

Perkembangan teknologi yang cukup cepat menjadi salah satu faktor maraknya *cybercrime* dengan menggunakan *malware*. Jenis-jenis *malware android* semakin beragam seiring berkembangnya teknologi dan banyaknya pengguna *android*. Secara umum, berikut adalah beberapa jenis *malware* yang dapat menyerang sistem operasi *android* (Sinambela et al., 2020):

a *Ransomware*

Malware yang termasuk dalam kategori sangat berbahaya apabila sampai masuk dan menginfeksi perangkat pengguna. Pasalnya, untuk jenis *malware* yang satu ini ketika berhasil masuk kedalam perangkat korban, maka akan langsung mengenkripsi data pada perangkat tersebut kemudian penyerang akan meminta tebusan untuk membuka kunci enkripsi tersebut.

b *Adware*

Adware merupakan *malware* iklan yang bekerja dengan menampilkan iklan di perangkat pengguna secara masif. Tujuannya, tentu saja keuntungan bagi penyebar *malware* dari klik iklan yang muncul di perangkat korbannya, karena *malware* ini menampilkan iklan yang tidak diinginkan pada perangkat secara terus-menerus. *Malware* ini mengumpulkan informasi pribadi korban, seperti nomor telepon, alamat *email*, dan lokasi geografis, tanpa sepengetahuan korban. *Malware* ini merupakan salah satu jenis *malware* yang juga paling banyak dipakai oleh penjahat *cyber*, karena mampu mencuri informasi sensitif dari perangkat korban.

c *Spyware*

Malware yang menyamar sebagai aplikasi yang bermanfaat untuk pengguna, namun sebenarnya memiliki tujuan jahat, seperti mengambil alih kontrol perangkat korban atau mengakses data pribadi korban. *Malware* ini

umumnya terpasang pada perangkat dengan menyerupai aplikasi data atau *game* populer.

d *Worm*

Malware yang menyebar dari satu perangkat ke perangkat lainnya melalui jaringan, seperti *WiFi* atau *Bluetooth*. Perangkat yang kerap mengakses akses internet dengan *WiFi* publik sangat rentan menjadi korban *malware* seperti ini.

e *Trojan*

Merupakan *malware* yang menyamar menjadi aplikasi yang berguna. Setelah di *install*, *Trojan* dapat mencuri data sensitif, merusak sistem, atau membuka pintu bagi serangan lanjutan.

f *Virus*

Malware yang menyebar melalui aplikasi atau file yang terinfeksi dan dapat merusak sistem operasi pada perangkat. *Virus* dapat menyebabkan kerusakan pada data dan program yang ada di perangkat korban.

g *Banking Trojan*

Merupakan *malware* yang dirancang untuk mencuri informasi keuangan pengguna, seperti nomor kartu kredit, nomor rekening bank, dan kata sandi. *Malware* ini biasanya menyebar melalui *email phishing* atau situs web palsu.

h *Backdoor*

Malware yang memungkinkan penyerang untuk mengambil alih kontrol perangkat dari jarak jauh tanpa sepengetahuan korban. *Malware* ini biasanya menyebar melalui aplikasi yang diunduh dari sumber yang tidak terpercaya.

2.1.2 *Dynamic Analysis*

Dynamic analysis atau analisis dinamis merupakan salah satu teknik dalam analisa *malware* yang digunakan untuk mempelajari perilaku *malware* dengan menjalankannya pada lingkungan yang diawasi dengan tujuan yaitu untuk memahami cara kerja *malware*. Analisa ini berfokus pada identifikasi pola aktivitas aplikasi dan jaringan mencurigakan seperti identifikasi *logcat error*,

shared memory dirty, system calls, process, dll. yang disebabkan oleh adanya serangan *malware* (Islam et al., 2023). Terdapat berbagai teknik pengamatan yang dilakukan pada analisis dinamis, diantaranya (Almomani et al., 2022):

a *Function Call Monitoring*, yaitu pengamatan dilakukan dengan menangkap *function call*. Beberapa informasi yang coba dipelajari dari proses *hooking* ini adalah tentang API (*Application Programm Interface*), *System Calls*, dan *Windows Native* API.

b *Function Parameter Analysis*, merupakan pengamatan yang dilakukan dengan mempelajari parameter atau *return value* dari sebuah *function*. Dengan cara ini maka akan diketahui keterhubungan antara *function-function* yang ada pada sampel *malware*.

c *Information Flow Tracking*. Pengamatan ini dilakukan untuk mempelajari alur data pada sampel *malware*. Data yang akan dipelajari diberi tanda terlebih dahulu (*tainted*). Pengamatan *information flow tracking* ini dilakukan untuk mempelajari informasi seperti *direct data dependencies*, *address dependencies*, *control flow dependencies*, *implicit information flow*, *Implementation of information flow tracking systems*, dll.

d *Instruction Trace*, merupakan pengamatan yang dilakukan terhadap instruksi apa saja yang dijalankan sampel pada saat sampel dianalisa.

e *Autostart Extensibility Points*: *Autostart extensibility points* (ASEPs) merupakan mekanisme pada sistem yang mengijinkan sebuah program untuk dijalankan secara otomatis setiap kali sistem operasi melakukan *booting*. Pengamatan terhadap ASEPs sangat penting untuk mengumpulkan informasi tentang sampel *malware*.

2.1.3 *Preprocessing Data*

Preprocessing data adalah serangkaian langkah dan teknik yang dilakukan pada data sebelum dimasukkan ke dalam model *machine learning* atau algoritma analisis data. *Preprocessing data* merupakan bagian kritis dalam rangkaian tahap *data mining* yang perlu dilakukan sebelum dilakukan analisis data. Tahapan ini melibatkan sejumlah langkah esensial seperti pembersihan data, transformasi data,

integrasi data, normalisasi data, imputasi data yang hilang, mengatasi kelas yang tidak seimbang, *feature scaling*, identifikasi *noise*, dan berbagai tugas reduksi data, seperti pemilihan fitur, pemilihan sampel, diskritisasi, dan sebagainya (García et al., 2015).

Preprocessing data yang cermat dapat meningkatkan performa model klasifikasi dengan memastikan bahwa data yang digunakan untuk melatih dan menguji model adalah data yang berkualitas dan sesuai dengan kebutuhan analisis. Berikut adalah beberapa tahapan secara umum dalam *preprocessing data* pada proses klasifikasi (Kamiran & Calders, 2012):

a Pembersihan Data (*Data Cleaning*)

Identifikasi dan penanganan data yang hilang, data yang tidak valid, atau data yang tidak relevan. Ini melibatkan penghapusan atau penggantian nilai yang hilang dan penanganan *outliers*.

b Transformasi Data (*Data Transformation*)

Transformasi variabel-variabel tertentu untuk memenuhi asumsi model atau untuk meningkatkan interpretabilitas. Contoh termasuk normalisasi, pengkodean kategori, atau transformasi logaritmik.

c Integrasi Data (*Data Integration*)

Penggabungan data dari berbagai sumber atau tabel menjadi satu dataset yang utuh. Hal ini diperlukan ketika data terdistribusi di beberapa tempat atau memiliki format yang berbeda.

d Normalisasi Data

Memastikan bahwa nilai-nilai dalam variabel memiliki skala yang seragam. Ini dapat mencegah variabel dengan skala besar mendominasi proses pembelajaran model.

e Penanganan Data yang Hilang (*Missing Data Handling*)

Identifikasi dan penanganan nilai-nilai yang hilang dalam dataset. Ini dapat melibatkan penghapusan baris atau kolom yang memiliki nilai yang hilang, atau mengisi nilai yang hilang dengan teknik imputasi.

f Identifikasi dan Penanganan *Noise*

Identifikasi dan penanganan *noise* atau gangguan dalam data yang dapat mempengaruhi performa model. *Noise* dapat berasal dari data yang tidak akurat atau ekstrem.

g Reduksi Dimensi (*Dimensionality Reduction*)

Pemilihan fitur yang relevan atau reduksi dimensi data untuk mengurangi kompleksitas model dan mempercepat proses pembelajaran. Metode ini mencakup pemilihan fitur dan pemampatan dimensi.

h *Sampling Data*

Jika data tidak seimbang antara kelas-kelas dalam tugas klasifikasi biner, teknik *sampling* seperti *oversampling* atau *undersampling* dapat digunakan untuk mencapai keseimbangan yang lebih baik.

i Pemilihan Fitur (*Feature Selection*)

Memilih subset fitur yang paling informatif dan relevan untuk tugas klasifikasi. Hal ini membantu meningkatkan efisiensi model dan mencegah *overfitting*.

j Diskritisasi

Merubah variabel kontinu menjadi variabel kategori untuk keperluan tertentu.

k Ekstraksi Fitur (*Feature Extraction*)

Merubah data menjadi representasi fitur yang lebih kompak dan informatif. Contoh teknik ekstraksi fitur termasuk *Principal Component Analysis* (PCA) atau teknik ekstraksi informasi lainnya.

2.1.4 *Split Data*

Split data atau pembagian data merupakan suatu metode dalam *machine learning* yang digunakan untuk membagi dataset menjadi tiga kelompok, yakni data pelatihan, data pengujian, dan data validasi. Proses pemisahan dataset menjadi tiga bagian ini dilakukan guna mengevaluasi performa model dan kemampuan algoritma dalam memahami pola dari data. Sementara beberapa penelitian menyatakan bahwa pemisahan dataset hanya perlu dilakukan menjadi data pelatihan dan data pengujian (Birba, 2020).

Data pelatihan digunakan untuk memberikan model pemahaman yang baik terhadap pola dan tren dalam dataset, kemudian data pengujian digunakan untuk menguji performa model pada data yang belum pernah dilihat sebelumnya, memberikan ukuran sejauh mana model dapat menghasilkan prediksi yang akurat. Adanya data validasi, dalam beberapa kasus, membantu dalam proses *fine-tuning* model dan mengoptimalkan parameter agar model dapat menggeneralisasi dengan baik. Proses pembagian data ini sering menggunakan perbandingan tertentu, seperti 80% data pelatihan - 20% data pengujian atau 70% data pelatihan - 30% data pengujian, untuk memastikan adanya representasi yang seimbang antara data pelatihan dan pengujian. Teknik *cross-validation* juga digunakan sebagai alternatif, membagi data menjadi beberapa lipatan untuk menghasilkan estimasi kinerja model yang lebih konsisten dan dapat dipercaya (Nguyen et al., 2021).

2.1.5 *Classification*

Classification atau klasifikasi merupakan teknik pendekatan *machine learning* yang digunakan untuk memprediksi kelas pada suatu data. Model klasifikasi mempelajari pola dari data pelatihan yang telah diberikan dan kemudian menggunakan pengetahuan ini untuk memprediksi kelas atau label dari data baru. Klasifikasi dikategorikan sebagai salah satu masalah yang paling banyak dipelajari oleh para peneliti di bidang *machine learning*, namun di sisi lain teknik ini memiliki tantangan tersendiri misalnya *imbalance class, missing value, noise* ataupun fitur yang tidak relevan (Soofi & Awan, 2017).

. Proses klasifikasi melibatkan tahap pelatihan, validasi, dan pengujian, di mana model mempelajari pola dari data pelatihan, diuji pada data validasi untuk memastikan kinerjanya, dan akhirnya digunakan untuk memprediksi kelas pada data yang tidak pernah dilihat sebelumnya. Evaluasi model klasifikasi melibatkan metrik seperti *confusion matrix, precision, recall, F1-score,* dan *Receiver Operating Characteristic* (ROC) *curve*. Klasifikasi tetap menjadi alat yang efektif dalam membuat prediksi dan pengambilan keputusan berbasis data, meskipun terdapat beberapa tantangan dalam prosesnya. Metode klasifikasi dibagi kedalam tiga jenis yaitu (Neeraj et al., 2020):

a *Discriminant or Discriminator Analysis*, jenis analisis ini menemukan ciri pembeda yang jelas di antara ciri-ciri yang diberikan untuk klasifikasi. Linear dan kuadrat adalah dua jenis analisis diskriminan utama yang digunakan untuk klasifikasi

b *Probabilistic Methods*, merupakan metode yang mengkategorikan fitur-fitur berdasarkan probabilitasnya untuk tetap berada di kelas tertentu. Metode ini dibagi menjadi *Logistic Regression* dan *Naive-Bayesian Classifiers*.

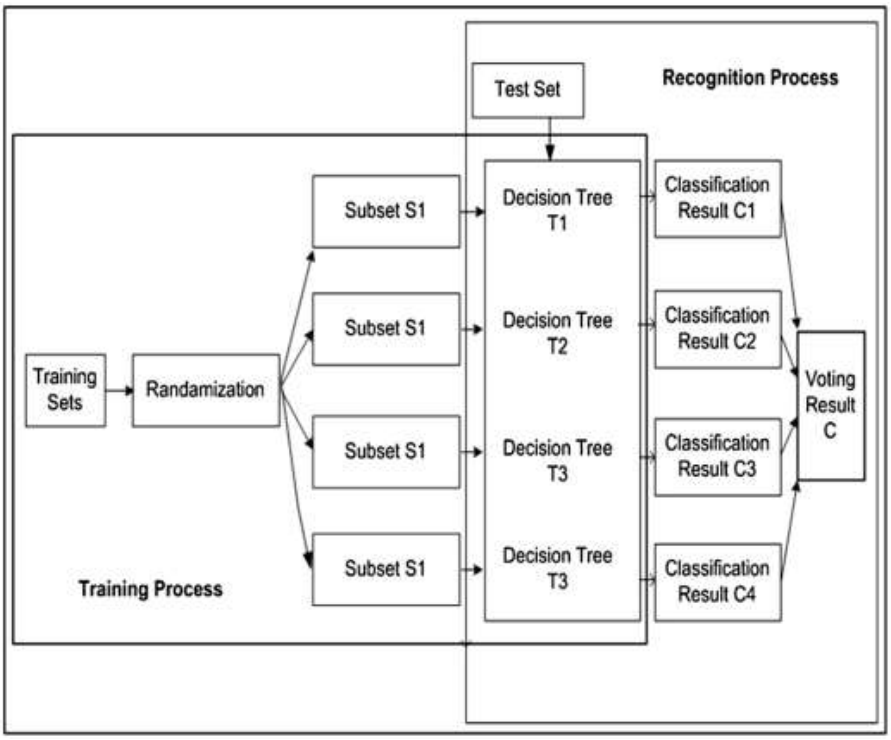
c *Model Based Classifier*, merupakan metode klasifikasi dengan pemodelan atau membangun algoritma *machine learning*. Model *machine learning* yang biasa digunakan untuk klasifikasi adalah *K-Nearest Neighbor* (KNN), *Decision Tree*, *Support Vector Machine* (SVM), *Random Forest* dan *Artificial Neural Network* (ANN).

2.1.6 Algoritma *Random Forest*

Algoritma *Random Forest* pertama kali dikemukakan oleh Leo Breiman dari *University of California* pada tahun 2001. Algoritma ini terdiri dari gabungan beberapa pengklasifikasian dasar (*decision tree*) yang sepenuhnya bersifat independen antara satu sama lain. Operasi acak (*random operation*) diperkenalkan dalam proses pembangunan model, termasuk pemilihan subset sampel dan subset fitur, untuk menjamin independensi setiap pohon keputusan, meningkatkan akurasi klasifikasi dan mendapatkan kemampuan generalisasi yang lebih baik. Penggunaan operasi acak dalam pemilihan subset sampel adalah dengan menarik subset sebagai set pelatihan dari sampel asli dengan metode *bagging*. Operasi acak di hutan acak secara signifikan meningkatkan kinerja model klasifikasi, karena proses pembuatan setiap pohon keputusan sangat cepat, paralelisasi dalam pembuatan hutan acak dapat diwujudkan, sehingga meningkatkan kecepatan klasifikasi secara signifikan (Breiman, 2020).

Algoritma *Random Forest* termasuk kedalam model *ensemble learning* yang merupakan gabungan dari pohon keputusan (*decision tree*) yang bertujuan untuk meningkatkan performa model. Algoritma ini menggunakan metode *bagging* (*bootstrap aggregating*) dimana beberapa model klasifikasi serupa dibangun

secara independen pada subset acak dari data pelatihan. Prediksi dari masing-masing model dihasilkan, dan hasil akhirnya diambil melalui *voting* atau *averaging* (Aung et al., 2009). Kemampuannya dalam menggabungkan hasil prediksi dari berbagai pohon, *Random Forest* mencapai tingkat ketangguhan yang tinggi terhadap *overfitting* dan performa yang baik dalam berbagai tugas klasifikasi dan regresi. Secara praktiknya, keunggulan *Random Forest* terletak pada kemampuannya untuk menangani sejumlah besar fitur dan data dengan baik, kemudian memiliki parameter-parameter seperti jumlah pohon, kedalaman pohon, dan jumlah fitur yang digunakan dapat disesuaikan untuk meningkatkan performa dan fleksibilitas model (Parmar et al., 2019). Berikut merupakan konsep klasifikasi pada model *Random Forest* :



Gambar 2. 1 Konsep *Random Forest Classifier* (Parmar et al., 2019)

Gambar 4.4 merupakan konsep dari proses klasifikasi pada model *Random Forest*, untuk penjelasan secara detailnya adalah sebagai berikut :

a. Dataset akan dibagi menjadi data *training* dan *testing*. Data *training* akan digunakan untuk membangun model *Random Forest*, sedangkan untuk data *testing* akan digunakan untuk pengujian model.
b. Data dibagi kedalam beberapa subset untuk menguji beberapa model *decision tree* sesuai dengan nilai parameter *n_estimators* atau jumlah pohon keputusan yang sudah ditetapkan.
c. Hasil klasifikasi dari beberapa model *decision tree* akan melalui tahap *voting* untuk menentukan *class* atau label dari data hasil klasifikasi.
d. Data *testing* kemudian akan dimasukkan untuk menguji performa model yang sudah dibangun sebelumnya.

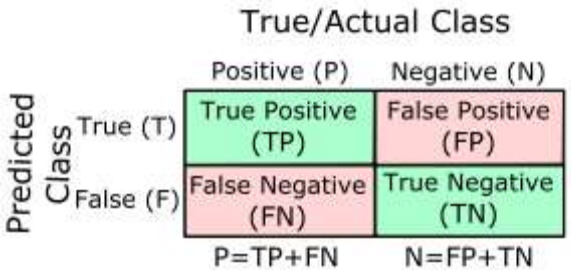
Rumus yang digunakan untuk memutuskan bagaimana node pada pohon keputusan bercabang. Rumus ini menggunakan kelas dan probabilitas untuk menentukan Gini setiap cabang pada sebuah node, menentukan cabang mana yang lebih mungkin terjadi. Di sini, p_i mewakili frekuensi relatif kelas yang Anda amati dalam kumpulan data dan *c* mewakili jumlah kelas. Berikut persamaan 2.1 yang merupakan rumus dalam memutuskan bagaimana node pada pohon keputusan bercabang.

$$\text{Gini} = 1 - \sum_{i=1}^{C} (p_i)^2 \qquad (2.1)$$

2.1.7 *Confusion Matrix*

Confusion Matrix merupakan suatu metode evaluasi kinerja yang umum digunakan dalam konteks masalah klasifikasi untuk mengukur akurasi suatu model dengan output berupa dua kelas atau lebih, dengan tujuan untuk menentukan kebenaran atau kesalahan prediksi. *Confusion Matrix* terdiri dari empat kategori, meliputi *True Positive* (TP), *False Positive* (FP), *True Negative* (TN), dan *False Negative* (FN), yang mencakup nilai aktual dan nilai prediksi. Hal

ini digunakan untuk menyajikan hasil evaluasi performa model dengan memberikan gambaran lebih rinci tentang keberhasilan atau kegagalan model dalam mengklasifikasikan instan data ke dalam berbagai kategori (Tharwat, 2018).



Gambar 2. 2 *Confusion Matrix*

True Positive (TP) merujuk pada data aktual yang memiliki nilai benar (P) dan diprediksi dengan hasil positif, sementara *False Positive* (FP) mengacu pada data yang memiliki nilai salah (N) namun diprediksi sebagai hasil positif. *True Negative* (TN) adalah data yang memiliki nilai benar (P) dan diprediksi sebagai hasil negatif, sedangkan *False Negative* (FN) merupakan data dengan nilai salah (N) dan diprediksi sebagai hasil negatif. Setelah memperoleh nilai untuk keempat kategori dalam *confusion matrix*, langkah selanjutnya adalah menghitung nilai *accuracy*, *precision*, *recall*, dan *F-measure*. Formulanya dapat ditemukan sebagai berikut.

$$Accuracy = \frac{TP+TN}{(TP+TN+FP+FN)} \qquad (2.\ 2)$$

$$Precision = \frac{TP}{(TP + FP)} \qquad (2.\ 3)$$

$$Recall = \frac{TP}{(TP+FN)} \qquad (2.\ 4)$$

$$F - Measure = 2 \times \frac{Precision \times Recall}{Precision+Recall} \qquad (2.\ 5)$$

2.2 Penelitian Terkait

2.2.1 *State of The Art*

Berdasarkan rumusan masalah dan tujuan penelitian, maka dilakukan literatur review dari penelitian sebelumnya yang berkaitan dengan klasifikasi malware android dengan pendekatan algoritma machine learning. Beberapa penelitian sebelumnya yang sudah dilakukan adalah, sebagai berikut:

Tabel 2. 1 *State of The Art*

No	Penulis, Tahun Penelitian	Judul	Algoritma	Dataset	Hasil
1.	(Hadiprakoso et al., 2022)	Analisis Statis Deteksi Malware Android Menggunakan Algoritma Supervised Machine Learning	SVM, Naive Bayes, Decision Tree, KNN	Malware DREBIN	Studi ini melakukan deteksi *malware* menggunakan metode analisis statis dan algoritma *machine learning* dengan dataset malware DREBIN. Beberapa algoritma *supervised learning* digunakan pada penelitian ini. Hasilnya, model dengan performa terbaik dihasilkan oleh algoritma SVM dengan akurasi mecapai 96,94% dan nilai AUC 95%.

2.	(Tjahjadi & Santoso, 2023)	Klasifikasi Malware Menggunakan Teknik Machine Learning	Random Forest	pe-files-malwares	Penelitian ini bertujuan untuk mengetahui performa algoritma *Random Forest* untuk klasifikasi. Dataset yang diambil dalam penelitian ini diperoleh dari internet dengan judul pe-files-malwares yang terdiri dari 19611 baris dan 79 kolom. Hasil penelitian menunjukan performa model dengan mendapat nilai *accuracy, precision, recall, f1-score* sebesar 99%.
3.	(Chitayae et al., 2023)	Identifikasi Malware pada Android menggunakan Algoritma K-Nearest Neighbor	KNN	Malware / Benign Permission Android	Penelitian ini menerapkan algoritma *K-Nearest Neighbor* (KNN) untuk klasifikasi *malware* pada aplikasi *android*. Penelitian ini menggunakan data *Android Malware/Benign Permissions* berupa file CSV yang diperoleh dari Kaggle.com. Hasil penelitian menunjukkan bahwa klasifikasi *malware* dan bukan *malware* pada izin aplikasi *android* dapat dilakukan dengan baik menggunakan algoritma KNN

					yang menghasilkan akurasi sebesar 77% dengan penerapan *feature selection.*
4.	(Diana et al., 2022)	Komparasi Algoritma Naïve Bayes, Logistic Regression Dan Support Vector Machine pada Klasifikasi File Application Package Kit Android Malware	Naïve Bayes, Logistic Regression Dan Support Vector Machine	Canadian Institute for Security, Google Play dan APK Pure	Pembelajaran mesin yang pada penelitian ini adalah *Naïve Bayes, Logistic Regression* dan *Support Vector Machine*. Hasil uji akurasi menunjukkan algoritma *Naive Bayes* mampu mengklasifikasi keluarga *malware* dengan tingkat akurasi 97.75%, sedangkan algoritma *Logistic Regression* akurasinya 88.75% dan akurasi *Support Vector Machine* mencapai 96,75%.
5.	(Efriyani & Panjaitan, 2021)	Klasifikasi Malware Dengan Menggunakan Recurrent Neural Network	RNN	DasmalWerk	Klasifikasi *malware android* dengan algoritma RNN dan penerapan fitur *N-grams*, didapatkan hasil *accuracy* sebesar 86% dan *f1-score* 85% dari jumlah data sebanyak 215 data *malware* dan bukan *malware*.

6.	(Refhaldo et al., 2022)	Klasifikasi Aplikasi Malware Android Menggunakan Algoritma C5.0	C5.0	Android Malware Dataset for ML	Penelitian ini bertujuan untuk mendapatkan hasil analisa menggunakan algoritma C5.0 dalam klasifikasi sebuah aplikasi yang teridentifikasi sebagai *malware*. Dengan teknik pengujian *split validation* 80:20 dimana 80% sebagai data *training* dan 20% sebagai data *testing*, maka didapatkan hasil akurasi sebesar 94,96% pada data training dan 94,23% pada data testing.

7.	(Turnip et al., 2023)	Klasifikasi Malware Android Aplikasi Menggunakan Random Forest Berdasarkan Fitur Statik	Random Forest	Virusshare	Kerangka pada penelitian ini meliputi pra pemrosesan data, klasifikasi menggunakan algoritma *Random Forest*, dan *test* APK terhadap model yang diperoleh. Pada penelitian ini, *Synthetic Minority Over-Sampling Technique* (SMOTE) diterapkan untuk menyelesaikan masalah ketidakseimbangan kelas pada dataset. Berdasarkan hasil penelitian, akurasi terbaik dihasilkan pada kombinasi SMOTE sebesar 92.26% dan dapat mengklasifikasi APK yang mengandung *malware* ke dalam 13 kelas jenis *malware*

8.	(Ramadhan et al., 2023)	Komparasi Algoritma Neural Network dan K-Nearest Neighbor Dalam Mendeteksi Malware Android	Neural Network dan KNN	Android Permission Dataset	Penelitian ini bertujuan untuk membandingkan performa dua algoritma *machine learning*, yaitu *Neural Network* dan *K-Nearest Neighbors* (KNN). Hasil penelitian menunjukkan bahwa *Neural Network* mencapai performa terbaik dengan akurasi 97%, presisi 97%, *recall* 97%, dan *F1-score* 97%. Sementara itu, KNN memiliki performa yang sedikit lebih rendah dengan akurasi 95%, presisi 96%, *recall* 95%, dan *F1-score* 95%
9.	(Rafrastara et al., 2023)	Deteksi Malware menggunakan Metode Stacking berbasis Ensemble	Stacking Methods (Neural Network, Random Forest, KNN dan Logistic Regression)	Malware static and dynamic features VxHeaven and Virus Total Data Set	Penelitian ini menerapkan metode *stacking* dalam klasifikasi data dengan tujuan untuk meningkatkan performa. Hasilnya, metode *stacking* dengan meta *classifier Logistic Regression* berhasil mengungguli 4 algoritma yang dieksekusi secara individu. Kemudian *stacking* dengan *logistic*

					regression juga dibandingkan dengan metode *stacking* lain, namun dengan meta *classifier* yang berbeda-beda, yaitu *Random Forest*, KNN dan *Neural Network*. Hasilnya, metode *stacking* dengan meta *classifier Logistic Regression* masih menjadi yang terbaik, dengan tingkat akurasi serta *recall* sebesar 98.7%
10.	(Sitorus et al., 2021)	Analisis Deteksi Malware Android menggunakan metode Support Vector Machine & Random Forest	SVM dan Random Forest	Dataset Malware/Benign permission Android	Penelitian ini melakukan perbandingan antara model SVM dengan model *Random Forest* dalam proses klasifikasi data, serta membandingkan hasil akurasi dengan penelitian sebelumnya. Klasifikasi menggunakan model SVM menghasilkan nilai *precision* 97%, nilai *recall* 97%, dan nilai *f1-score* 97%, dan akurasi 96,23%, kemudian pada model *Random Forest* menghasilkan nilai *precision* 99%, nilai *recall* 99%,nilai *f1-score* 99%, dan akurasi

					98,99%. Kesimpulannya, model *Random Forest* lebih unggul dari model SVM.
11.	(Yogaswara, 2021)	Klasifikasi Malware Family Menggunakan Metode K-Nearest Neighbor (K-NN)	KNN	CICInversAndMal 2019	Model KNN digunakan untuk klasifikasi *malware*. Metode C5.0 digunakan dalam *preprocessing* data kemudian data di uji pada model klasifikasi KNN. Pembagian data sebesar 90% untuk data *training* dan 10% data *testing* mendapat hasil terbaik pada performa model dengan nilai *recall* 65% dan *precision* 83%.
12.	(Zakariya & Ramli, 2023)	Desain Penilaian Risiko Privasi Pada Aplikasi Seluler Melalui Model Machine Learning Berbasis Ensembel Learning Dan Multiple Application Attributes	Ensemble methods (Decision Tree, KNN, Random Forest)	CIC-AndMal2017	Penggunaan *ensemble methods* bertujuan untuk meningkatkan performa model dalam proses klasifikasi data. Hasil penelitian menunjukkan bahwa penerapan *ensemble learning* dengan algoritma klasifikasi *Decision Tree* (DT), *K-Nearest Neighbor* (KNN), dan *Random Forest* (RF) memiliki performa model yang lebih baik dibandingkan dengan menggunakan

					algoritma klasifikasi tunggal, dengan *accuracy* sebesar 95.2%, nilai *precision* 93.2%, nilai *F1-score* 92.4%, dan *True Negative Rate* (TNR) sebesar 97.6%.
13.	(Alomari et al., 2023)	*A Comparative Analysis of Machine Learning Algorithms for Android Malware Detection*	LightGBM, RF, Extra Trees Classifier, Gradient Boosting Classifier, DT, KNN, Ada Boost Classifier, Linear Discriminant Analysis, Ridge Classifier, Logistic Regression, Naive Bayes, SVM, Quadratic Discriminant Analysis	CICMalDroid2020 (Dynamic Features)	Penelitian ini melakukan perbandingan 13 model dalam pengklasifikasian pada dataset CICMalDroid2020. Selain itu, penelitian ini menerapkan teknik SMOTE, normalisasi skor-z pada fitur numerik dan menerapkan PCA untuk memperbaiki masalah pada data dan mencapai akurasi maksimum. Hasilnya menunjukkan bahwa model *LightGBM* adalah model dengan akurasi terbaik di antara algoritma lain yang diuji pada semua tahapan. Skor F1 terbaik yang dicapai adalah 95,47% oleh algoritma *LightGBM*.

14.	(Islam et al., 2023)	*Android Malware Classification Using Optimum Feature Selection And Ensemble Machine Learning*	Ensemble Methods (Random Forest (RF), K-nearest Neighbor (KNN), Multi-layer Perceptron (MLP), Decision Tree (DT), Support Vector Machine (SVM) and Logistic Regression (LR))	CCCS-CIC-AndMal-2020 (Dynamic Analysis)	Penelitian ini menggunakan *ensemble learning* dengan menggabungkan 6 algoritma klasifikasi serta menerapkan beberapa tahapan pada *preprocessing* data seperti *missing data imputation, random oversampling, outlier handling, feature scaling, transformation, feature selection,* dan *dimensionality reduction*. Hasil akhir dari evaluasi model *ensemble learning i*ni didaptkan nilai akurasi sebesar 95% setelah mengecualikan 60,2% fitur.
15.	(Batouche & Jahankhani, 2021)	*A Comprehensive Approach to Android Malware Detection Using Machine Learning*	SVM, Tree Classifier, Random Forest, Naive Bayes, KNN	CCCS-CIC-AndMal-2020 (Static Analysis)	Model SVM, *Tree Classifier*, *Random Forest*, *Naive Bayes*, dan KNN di gunakan dalam proses klasifikasi. Penelitian ini bertujuan untuk membandingkan performa model dalam mengklasifikasi data. Hasilnya, performa model yang terbaik didapatkan oleh algoritma *Random Forest* dengan nilai akurasi 89%.

16.	(Shatnawi et al., 2022)	*An Android Malware Detection Approach Based on Static Feature Analysis Using Machine Learning Algorithms*	SVM, KNN, Naive Bayes	CICInversAndMal 2019 (Static Feature)	Penelitian ini membandingkan beberapa algoritma *machine learning* yiatu SVM, KNN, dan *Naive Bayes* untuk pengklasifikasian pada dataset CICInversAndMal2019 berdasarkan *static feature*. Tahap *preprocessing, feature extraction,* dan *feature selection* diterapkan pada penelitian ini. Hasilnya model SVM menghasilkan performa terbaik dibandingkan model lain, dengan nilai akurasi sebesar 94,36%.

17.	(Almahmoud et al., 2021)	*ReDroidDet: Android Malware Detection Based on Recurrent Neural Network*	RNN	Kombinasi dari dataset CIC-AndMal2017, CIC-InvesAndMal2019, dan CIC-MalDroid2020. (Static Feature)	Penelitian ini mengusulkan algoritma RNN untuk deteksi *malware* berdasarkan *static feature* pada dataset yang sudah di kombinasi. *Feature extraction* dan *selection* diterapkan pada *preprocessing data*. Hasilnya model RNN menghasilkan performa lebih baik dari penelitian sebelumnya dengan nilai akurasi mencapai 98,58%.

2.2.2 Matriks Penelitian

Tabel 2.2 mencakup matriks penelitian yang berfokus dalam klasifikasi *malware android* dengan menggunakan pendekatan algoritma machine learning. Matriks tersebut dapat memberikan informasi tentang perbedaan penelitian yang akan dilakukan dengan penelitian terdahulu.

Tabel 2. 2 Matriks Penelitian

No	Penulis, Tahun Penelitian	Judul	Ruang Lingkup								
			Analisis		Metode Klasifikasi		Parameter Uji				
			Statis	Dinamis	*Ensemble Learning*	Model Klasifikasi Tunggal	*Accuracy*	*Precision*	*Recall*	*F- Measure*	*ROC Curve / AUC*
1.	(Fauzi, 2023)	Klasifikasi Serangan Malware Android Berdasarkan Hasil Analisis Fitur Dinamis Menggunakan Algoritma Random Forest	–	√	√	–	√	√	√	√	–
2.	(Hadiprakoso et al., 2022)	Analisis Statis Deteksi Malware Android Menggunakan Algoritma Supervised Machine Learning	√	–	–	√	√	–	–	–	√
3.	(Tjahjadi & Santoso, 2023)	Klasifikasi Malware Menggunakan Teknik Machine Learning	√	–	√	–	√	√	√	√	–

4.	(Chitayae et al., 2023)	Identifikasi Malware pada Android menggunakan Algoritma K-Nearest Neighbor	√	–	–	√	√	√	√	√	–
5.	(Diana et al., 2022)	Komparasi Algoritma Naïve Bayes, Logistic Regression Dan Support Vector Machine pada Klasifikasi File Application Package Kit Android Malware	√	–	–	√	√	√	√	√	√
6.	(Efriyani & Panjaitan, 2021)	Klasifikasi Malware Dengan Menggunakan Recurrent Neural Network	√	–	–	√	√	–	–	√	–
7.	(Refhaldo et al., 2022)	Klasifikasi Aplikasi Malware Android Menggunakan Algoritma C5.0	√	–	–	√	√	√	√	–	–
8.	(Turnip et al., 2023)	Klasifikasi Malware Android Aplikasi Menggunakan Random Forest Berdasarkan Fitur Statik	√	–	√	–	√	√	√	√	–
9.	(Ramadhan et al., 2023)	Komparasi Algoritma Neural Network dan K-Nearest Neighbor Dalam Mendeteksi Malware Android	√	–	–	√	√	√	√	√	–

10.	(Rafrastara et al., 2023)	Deteksi Malware menggunakan Metode Stacking berbasis Ensemble	–	√	√	–	√		√	–	–
11.	(Sitorus et al., 2021)	Analisis Deteksi Malware Android menggunakan metode Support Vector Machine & Random Forest	√	–	√	–	√	√	√	√	–
12.	(Yogaswara, 2021)	Klasifikasi Malware Family Menggunakan Metode K-Nearest Neighbor (K-NN)	√	–	–	√	–	√	√	–	–
13.	(Zakariya & Ramli, 2023)	Desain Penilaian Risiko Privasi Pada Aplikasi Seluler Melalui Model Machine Learning Berbasis Ensembel Learning Dan Multiple Application Attributes	√	–	√	–	√	√	√	√	–
14.	(Alomari et al., 2023)	*A Comparative Analysis of Machine Learning Algorithms for Android Malware Detection*	√	–	–	√	√	√	√	√	√
15.	(Islam et al., 2023)	*Android Malware Classification Using Optimum Feature Selection And Ensemble Machine Learning*	–	√	√	–	√	√	√	√	–

16.	(Batouche & Jahankhani, 2021)	*A Comprehensive Approach to Android Malware Detection Using Machine Learning*	√	–	√	–	√	√	√	√	–
17.	(Shatnawi et al., 2022)	*An Android Malware Detection Approach Based on Static Feature Analysis Using Machine Learning Algorithms*	√	–	–	√	√	√	√	√	–
18.	(Almahmoud et al., 2021)	*ReDroidDet: Android Malware Detection Based on Recurrent Neural Network*	√	–	–	√	√	√	√	√	–

Berdasarkan matriks penelitian pada Tabel 2.2, penelitian yang akan dilakukan adalah klasifikasi *malware android* dengan menggunakan metode *ensemble learning Random Forest* dengan parameter uji yaitu *accuracy, precicion, recall,* dan *F-Measure*. Perbedaan antara penelitian yang akan dilakukan dengan penelitian sebelumnya, yaitu terletak pada penggunaan dataset yang merupakan dataset *malware android* terbaru berdasarkan fitur hasil analisis dinamis. Metode klasifikasi yang digunakan pada penelitian ini yaitu *ensemble learning* dan merupakan saran dari penelitian sebelumnya. Penelitian ini akan berfokus pada perbaikan performa model *ensemble learning* (Random Forest) dalam pengklasifikasian malware android berdasarkan analisis fitur dinamis pada dataset CCCS-CIC-AndMal2020.