Jutisi: Jurnal Ilmiah Teknik Informatika dan Sistem Informasi
Jl. Ahmad Yani, K.M. 33,5 - Kampus STMIK Banjarbaru
Loktabat – Banjarbaru (Tlp. 0511 4782881), e-mail: puslit.stmikbjb@gmail.com
e-ISSN: 2685-0893
p-ISSN: 2089-3787

Komparasi Algoritma *Naïve Bayes, Logistic Regression* Dan *Support Vector Machine* pada Klasifikasi File *Application Package Kit* Android *Malware*

Diana[1*], Richardus Eko Indrajit[2], Erick Dazki[3]
Magister Teknologi Informasi, Universitas Pradita
Jl. Gading Serpong Boulevard No.1, Curug Sangereng, Tangerang, Banten, Indonesia
*e-mail Corresponding Author: *diana@student.pradita.ac.id*

Abstrak

Fenomena *malware* yang terus bertumbuh pada sistem Android menjadikan peneliti berfokus untuk menganalisa *malware* dengan memanfaatkan teknologi kecerdasan buatan. Tujuan dari penelitian ini adalah menganalisa file-file APK (*Application Package Kit*) Android dengan mengklasifikasi keluarga *malware*. File *malware* akan dijadikan dataset untuk dilakukan training menggunakan algoritma pembelajaran mesin. Pembelajaran mesin yang digunakan adalah *Naïve Bayes*, *Logistic Regression* dan *Support Vector Machine*. Pengukuran performansi dan akurasi juga disajikan dalam perbandingan antara algoritma *Naïve bayes, Logistic Regression* dan *Support Vector Machine* yang merupakan algoritma *Machine Learning* dan bagian dari kecerdasan buatan. 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%. Meskipun akurasi tidak setinggi penelitian sebelumnya, teknik analisa statis dengan fitur *Permission* dan fitur *Intent* cukup sederhana untuk mendeteksi file APK Android adalah *malware* atau bukan *malware*.

Kata kunci: *Malware Android; Naïve Bayes; Logistic Regression; Support Vector Machine*

Abstarct

The phenomenon of malware that continues to grow on the Android system makes researchers focus on analyzing malware by utilizing artificial intelligence technology. The purpose of this research is to analyze Android APK (Application Package Kit) files by classifying malware families. The malware files will be used as a dataset for training using machine learning algorithms. The machine learning used is Naïve Bayes, Logistic Regression and Support Vector Machine. Performance and accuracy measurements are also presented in a comparison between the Naïve Bayes algorithm, Logistic Regression and Support Vector Machine which is a Machine Learning algorithm and part of artificial intelligence. The accuracy test results show that the Naive Bayes algorithm is able to classify malware families with an accuracy rate of 97.75%, while the Logistic Regression algorithm has an accuracy of 88.75% and an accuracy of Support Vector Machine reaches 96.75%. Although the accuracy is not as high as previous studies, the static analysis technique with the Permission feature and the Intent feature is quite simple to detect Android APK files are malware or not malware.

Keyword: *Malware Android; Naïve Bayes; Logistic Regression; Support Vector Machine*

1. Pendahuluan

Mengunduh *file Application Package Kit (APK)* [1], tanpa melakukan proses pengecekan dan melakukan instalasi *file APK* ke dalam sistem *Android* sangat membahayakan. Ada beberapa situs pengecekan *file APK* seperti virustotal.com. Situs ini melakukan *scanning file APK*, hasil pengecekannya file tersebut *Malware* atau bukan *Malware*. Famili *Malware* banyak sekali seperti famili *Malware* yang menyerang industri perbankan [2], famili *Malware* yang menyerang dengan media SMS, famili *Malware* dengan cara meminta tebusan kepada user target, yang dikenal dengan *Ransomware*.

Android memberikan *Source Code* [3] dan *Open Source* [4] dibawah lisensi *Apache* versi 2.0. Pembaruan *Kernel Linux*, dibawah Lisensi GNU. Karena sistem operasi Android bersifat *Open Source* [3]. Memungkinan peretas yang mempunyai maksud dan tujuan tertentu, menciptakan aplikasi APK tertentu untuk menguntungkan peretas. Selain itu pihak *Google Play Protect* membantu melakukan proteksi terhadap aplikasi APK yang dicurigai mengandung *Malware.* Keamanan dari Google Play [5] sudah memberikan layanan yang mumpuni dalam melakukan proteksi Aplikasi Android (APK).

Pada dasarnya analisa *Malware* terbagi menjadi analisa statis, analisa dinamis dan analisa gabungan atau Hybrid. Analisa statis Malware adalah analisa tanpa melakukan instalasi *Malware* ke sistem operasi, tetapi melakukan *Reverse Engineering* terhadap file-file APK *Malware*. Hasil *Reverse Engineering* [6] berupa file *Source Code* dan siap untuk dilakukan analisa. Sedangkan analisa dinamis tanpa melakukan Reverse Engineering dan menjalankan file *Malware*, kemudian menganalisa *Behavior Malware* [7] menggunakan log dari sistem Android. Analisa *Malware* menggunakan metode gabungan atau Hybrid adalah dengan melakukan analisa *Source Code* dan menganalisa log dari *system Android*.

Banyak penelitian dengan menggunakan fitur-fitur *APK Android* dengan menggunakan analisa statis. Fitur *permission* memang banyak dipilih karena fitur permission menjadi *trending* topik penelitian. Fitur-fitur yang jarang dilakukan penelitian adalah *Activities, Information Flow, Intent, Java Classes, Native Codes* [8]*, Network Protocols, Network Traffic, Opcodes, Services dan System Resources*. Fitur Intent menjadi menantang (*Challenging*), jika dilakukan penelitian. Penelitian ini akan menambahkan fitur permission dan fitur intent. Dimana fitur intent itu bertujuan mengaktifkan action dari module Android. Karena jika tidak digunakan fitur action maka sebuah APK Android tidak akan berjalan. *Research Gap* penelitian ini adalah penelitian sebelumnya yang jarang memasukkan fitur *Intent* kedalam dataset *malware*. Dengan alasan tersebut penelitian ini dilanjutkan untuk menjadi penelitian, karena fitur permission akan berhubungan erat dengan fitur intent.

Tujuan dari penelitian ini adalah mendapatkan model klasifikasi dengan algoritma *Naïve Bayes, Logistic Regression* dan *Support Vector Machine*. Akurasi, presisi, *recall* dan *F1-score* juga dilakukan. Pengukuran kinerja model dari ketiga algoritma tersebut menggunakan *ROC Curve*. Urgensinya untuk mengukur kinerja dari model klasifikasi sangatlah penting, karena model tersebut dapat digunakan untuk melakukan deteksi apakah file APK Android tersebut terinfeksi *Malware* atau tidak.

2. Tinjauan Pustaka

Dari pengumpulan data dari berbagai jurnal dengan topik Android *Malware* kebanyakan berfokus pada penelitian *Machine Learning, Feature Selection, Mobile Malware Detection* [9] [10] [11]. Berikut topik penelitian Android *Malware* yang banyak dibahas:

1. *File AndroidManifest* diekstraksi, Algoritma TF-IDF digunakan untuk menghitung nilai permission dari setiap permission dan nilai sensitivitas APK (SVOA) dari setiap aplikasi. SVOA dan jumlah *permission* yang digunakan dan diuji dengan *machin learning*. Hasil eksperimen menunjukkan bahwa hanya menggunakan fitur *permission Malware* untuk membedakan *malware* dan bukan *malware*. Untuk deteksi *malware*, pendekatan yang diusulkan mencapai akurasi hingga 99,5% [12].
2. Deteksi aplikasi yang terinfeksi malware dan kerentanan terindikasi pada fitur permission file APK Android. Melakukan percobaan menggunakan fitur permission pada aplikasi Android. Melakukan fitur seleksi yang terbaik dari Dataset, mengembangkan model deteksi malware dengan memanfaatkan pendekatan pembelajaran LSSVM (*Least Square Support Vector Machine*) yang terhubung melalui tiga fungsi kernel yang berbeda yaitu, linear, radial basis dan polinomial. Dengan metode tersebut mencapai akurasi 98,8% [13].
3. Metode analisis hibrid untuk mendeteksi malware Android dan mengklasifikasikan kelompok *malware* disajikan dalam makalah ini, dan sebagian dioptimalkan untuk data multi-fitur. Untuk analisis statis, penggunaan fitur *Permission* dan Intent sebagai fitur statis dan menggunakan tiga metode pemilihan fitur untuk membentuk subset dari tiga kandidat fitur. Dibandingkan dengan berbagai model, termasuk *K-Nearest Neighbor* dan *Random Forest*, hasil dengan algoritma Random Forest adalah yang terbaik, dengan tingkat deteksi 95,04%, sedangkan uji *chi-square* adalah metode seleksi fitur terbaik. Dalam analisis dinamis berdasarkan lalu lintas jaringan, tidak seperti yang berfokus pada arus lalu lintas

satu arah dan bekerja pada protokol HTTP, protokol lapisan transport, lapisan protokol. Hasil eksperimen menunjukkan terjadi akurasi dari 74% menjadi 99% [14].

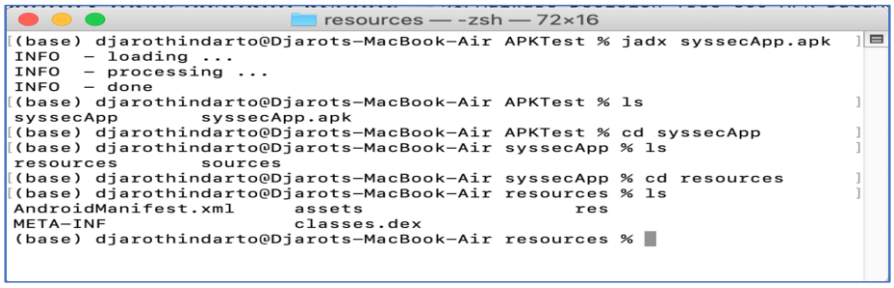
4. SmartMal menyajikan kerangka kerja deteksi malware perilaku berorientasi layanan baru untuk perangkat seluler. *SmartMal* memperkenalkan konsep *service-oriented architecture* (SOA) dan *behavior* analisis ke dalam paradigma deteksi *malware*. *Framework* yang diusulkan bergantung pada arsitektur *client-server, client* terus-menerus mengekstrak berbagai fitur dan mentransfernya ke server, dan tugas utama server adalah mendeteksi anomali menggunakan algoritme deteksi. Beberapa server terdistribusi secara bersamaan menganalisis vektor fitur menggunakan berbagai detektor dan fusi informasi digunakan untuk menggabungkan hasil detektor. Pendekatan statistik berbasis siklus untuk deteksi anomali perangkat seluler, menganalisis pola penggunaan reguler pengguna. Hasil empiris menunjukkan bahwa *Framework* yang diusulkan dengan algoritma deteksi anomali sangat efektif dalam mendeteksi *malware* pada perangkat Android [7].
5. Penelitian ini memanfaatkan tingkat positif palsu yang rendah dari deteksi penyalahgunaan dan kemampuan deteksi anomali untuk mendeteksi *malware zero-day*. Usulan sistem deteksi hibrid berdasarkan *tools CuckooDroid*, yang memungkinkan penggunaan fitur *Cuckoo Sandbox* untuk menganalisis malware Android melalui analisis dinamis dan statis. Sistemnya terdiri dari dua bagian: mesin pendeteksi anomali yang melakukan deteksi aplikasi abnormal melalui analisis dinamis; mesin pendeteksi *Signature* yang melakukan deteksi dan klasifikasi malware dengan kombinasi analisis statis dan dinamis. Eksperimen menunjukkan bahwa mesin pendeteksi anomali mampu mendeteksi *malware zero-day* dengan tingkat negatif palsu yang rendah (1,16 %) dan tingkat positif palsu yang dapat diterima (1,30 %). Akurasi klasifikasi sampel *malware* dengan tingkat positif rata-rata 98,94% [15].
6. Penelitian ini mengusulkan model deteksi *Malware* berdasarkan fitur ketidakpastian. Algoritma logistik regresi menggambarkan hubungan *input* (*Permission*) dan *output* (label). Algoritma *Markov Chain Monte Carlo* (MCMC) untuk menyelesaikan ketidakpastian fitur. Setelah bereksperimen dengan 2037 sampel, untuk deteksi *malware*, mencapai akurasi 95,5%, dan False Positive Rate sebesar 1,2%. Akurasi deteksi lebih tinggi dari akurasi langsung menggunakan 24 fitur Permission. Hasil akurasi deteksi sampel *Malware* baru adalah 92,7%. Dibandingkan dengan pendekatan mutakhir lainnya, metode ini efektif dalam mendeteksi malware [16].

Sedangkan penelitian tentang *Android Malware Classification* [17], [18] masih tergolong sedikit. Fitur *Intent* berguna untuk mengaktifkan modul-modul action dari APK Android. Menjadikan alasan penulis untuk memasukan fitur Intent ke dalam dataset *malware. State-of-the-art* pada penelitian ini, melakukan klasifikasi *Malware* dengan analisa static. Fitur *Permission* dan fitur Intent sebagai parameter input. Klasifikasi menggunakan algoritma *Naïve Bayes, Logistic Regression* dan *Support Vector Machine*.

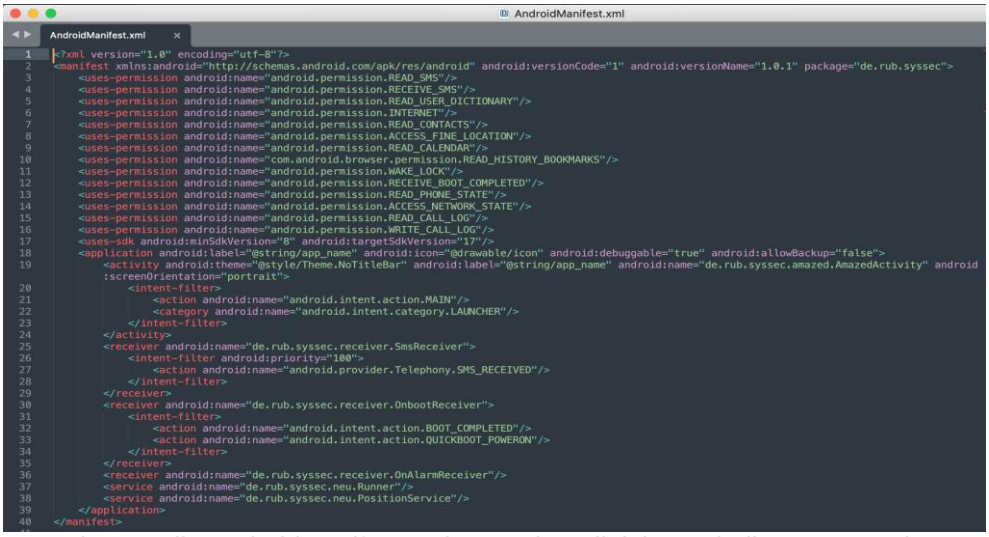
3. Metodologi

3.1. Reverse Engineering

Reverse Engineering adalah metode *decompiled file Execute* menjadi file *Source Code*. Salah satu *tools* yang digunakan untuk *Android Reverse Engineering* adalah *Jadx. Tools* tersebut mengubah *file APK Android* menjadi *Folder Source code, Folder Resources* dan *APK Signature*. Module Jadx ada dua macam, yaitu Jadx, dijalankan lewat *Command Line (CLi)* dan Jadx-gui, berbasis *Graphical User Interface (GUI).*



Gambar 1. *Reverse Engineering* dengan *Command Line (CLI), Jadx*



Gambar 2. *File AndroidManifest.xml* yang akan diolah menjadi *Dataset Malware*

Di *Folder Resources*, terdapat file *AndroidManifest.xml*, file inilah yang berisikan informasi mengenai nama paket aplikasi, komponen aplikasi, *Permission* dari aplikasi serta fitur perangkat keras dan fitur perangkat lunak dari aplikasi. Fitur *Permission* dan fitur *Intent* adalah dua komponen yang dibutuhkan *Android* dalam menjalankan *file* APK Android. Hasil dari *Reverse Engineering* yang dijadikan sebagai *Dataset* dalam proses klasifikasi adalah *file AndroidManifest.xml*. *File* tersebut berikan informasi mengenai permission, intent, dan lain-lain. Jadx merupakan module ekstraks *file APK Android* yang bersifat *Open-Source*.

3.2. DATASET MALWARE

Penggunaan klasifikasi menggunakan *Machine Learning* memerlukan Dataset untuk proses *probability* menghasilkan keputusan keluarga Malware. Dataset diambil dari pembacaan isi file APK Android, yang terbaca adalah parameter atau fitur permission dan fitur intent. Setelah itu disimpan kedalam Dataset *Malware*. Keluarga Malware yang dilakukan penelitian adalah: *Adware APK, Banking APK, Ransomware APK, Riskware APK, SMS APK, Benign APK.*

Tabel 1. *Malware* dan label (target)

No	Malware	Label
1	*Banking APK*	0
2	*Adware APK*	1
3	*Ransomware APK*	2
4	*Riskware APK*	3
5	*SMS APK*	4
6	*Benign APK*	5

Dataset di tabel 2, menggunakan fitur-fitur *Permission* yang digunakan untuk memberikan ijin dari perangkat seluler. Selain fitur Permission, fitur Intent juga dimasukan ke *Dataset Malware.* Fitur *Intent* ini yang melakukan *Action* terhadap perangkat yang sudah diberikan *Permission* melalu fitur *Permission*. Didalam penelitian ini menggunakan Dataset 600 file APK Android. Setalah dilakukan proses ekstrak dari file tersebut didapatkan 1277 fitur. *Malware Banking* adalah kelompok *Malware* yang sering menyerang pada sektor perbankan, dimana target utama ada mengambil informasi dari nasabah bank. Ransomware adalah kelompok *Malware* yang bertujuan meminta tebusan dari korban atau pengguna yang terinfeksi *Ransomware.*

3.3. NAIVE BAYES

Naïve Bayes Classifier adalah metoda klasifikasi berdasarkan teori Bayes. Metode ini berdasarkan probabilitas yang dipresentasikan oleh ilmuwan Inggris Thomas Bayes. Peluang

dari prediksi masa depan berdasarkan pengalaman di masa sebelumnya. Asumsi yang kuat (Naive) terhadap indepensi parameter merupakan ciri-ciri dari metode *Naive Bayes*.
Salah satu kelebihan dari *Naive Bayes*, tidak membutuhkan Dataset yang besar, mudah dipahami dan masih banyak kelebihan-kelebihan teori *Naive Bayes*.
Sebagai dasar teori *Naive Bayes*

$$P(A|B) = (P(B|A) * P(A)) / P(B) \quad \text{..... (1)}$$

Dimana,
P (A|B) = Likelihood.
P(B) = Evidence.
P(A) = Prior Probability.
P(A|B) = Probabilitas terjadinya A sebagai syarat B telah terjadi.
Dalam penggunaan teori Bayes, dapat mencari peluang terjadinya A, sebagai syarat B telah terjadi. Penggunaan nilai B sebagai bukti dan nilai A sebagai hipotesis. Dengan asumsi bahwa nilai prediktor/fiturnya independen. Dapat diartikan bahwa kehadiran satu fitur tertentu tidak mempengaruhi fitur yang lain. Maka hal ini disebut sebagai Naive. Rumus ini berubah menjadi:

$$P(y \mid X) = (P(X \mid y) * P(y)) / P(X) \quad \text{..... (2)}$$

Sebagai contoh, sekolah mengadakan acara kegiatan penyelenggaraan perkemahan di alam. Sebelum melakukan kegiatan tersebut sudah melihat tentang ramalan cuaca. Hasil dari informasi cuaca sebagai berikut: Hari ini hujan dan mulai berawan = 50%, hari ini meulai berawan = 40%, hari ini pada bulan september kemungkinan hujan = 10%. Bagaimana menyelesaikan masalah tersebut berdasarkan teorim Bayes? Apakah acara perkemahan sekolah jadi dilaksanakan?
Berdasarkan teori Bayes:
P (hujan|berawan) = (P(hujan) * P(berawan|hujan)) / P(berawan)
P (hujan|berawan) = (0,1 * 0,5) / 0,4
P (hujan|berawan) = 0,125 x 100%
P (hujan|berawan) = 12,5%
Dari hasil perhitungan dengan menggunakan teori Bayes, maka didapatkan 12,5% kemungkinan hujan. Maka dapat diputuskan bahwa acara perkemahan sekolah di alam, tetep dapat dilaksanakan. Masih banyak permasalahan untuk menyelesaikan dengan teori Bayes, seperti penyelesaian teks dalam dokumen, masalah spam didalam email, deteksi *Malware* APK Android dan lain-lain. Persamaannya adalah:

$$V_{MAP} = \arg\max P(V_j \mid a_1, a_2, \ldots\ldots a_n) \quad \text{..... (3)}$$

Menurut persamaan (3), maka persamaan (1) dapat ditulis:

$$V_{MAP} = \underset{v_j \in V}{\arg\max} \frac{P(a_1, a_2, \ldots a_n \mid v_j)\, P(v_j)}{P(a_1, a_2, \ldots a_n)} \quad \text{..... (4)}$$

$P(a_1, a_2, \ldots\ldots a_n)$ sebagai konstan, sehingga dapat dihilangkan dan persamaan menjadi:

$$V_{MAP} = \underset{v_j \in V}{\arg\max}\, P(a_1, a_2, \ldots a_n \mid v_j)\, P(v_j) \quad \text{..... (5)}$$

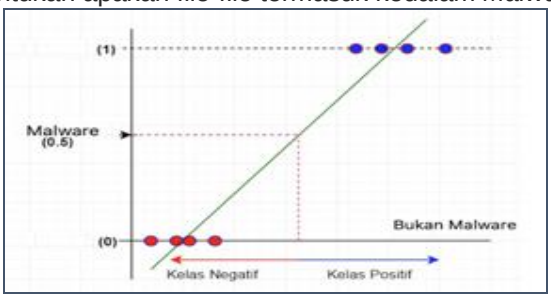
Nilai $P(a_1,a_2,\ldots..a_n|v_j)$ sulit dilakukan perhitungan, maka diasumsikan bahwa setiap *Malware* pada file APK tidak mempunyai keterkaitan.

Jenis Pengklasifikasi *Naive Bayes*:
1. *Multinomial Naive Bayes*:
Frekwensi kata yang ada didalam dokumen menggunakan parameter fitur atau prediktor sebagai pengklasifikasi [19], [20]. Metode ini digunakan untuk klasifikasi dokumen, yaitu apakah suatu dokumen termasuk dalam kategori teknologi, berita, olahraga, dll.
2. *Bernoulli Naive Bayes* [21]:
Metode ini hamper sama dengan Mulitnomial Naive Bayes multinomial, perbedaan di prediktornya adalah variabel *boolean*. Parameter yang digunakan untuk memprediksi variabel kelas dengan bernilai ya atau tidak [22].
3. *Gaussian Naive Bayes*:
Ketika nilai-nilai kontinu dan tidak diskrit diambil oleh predictor yang diasumsikan sebagai nilai-nilai dari distribusi Gaussian [23], [24]. Untuk lebih lanjut mengenai Naïve Bayes, penelitian ini membahas implementasi, menggunakan bahasa pemrograman *Python. Naive Bayes* juga

merupakan salah satu *Machine Learning.* Metode *Machine Learning* [25], [26] adalah metode yang sangat sederhana dalam memproses setiap fitur dan menghasilkan keputusan yang lebih baik. Sehingga metode ini sangat cocok untuk berbagai penyelesaian.

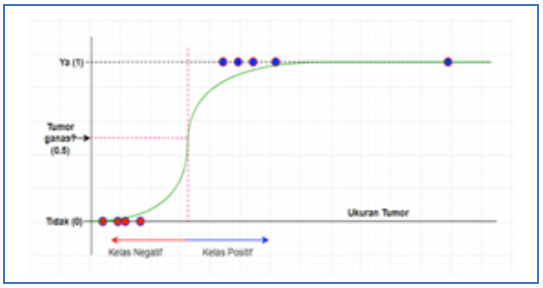
3.4. Logistic Regression

Sebelum membahas mengenai *Logistic Regression* [27], [28], [29], maka disampaikan mengenai *Linear Regression*. *Linear Regression* adalah suatu cara permodelan masalah keterhubungan antara suatu variabel independen terhadap variabel dependen [30], [31]. Contohnya adalah menentukan apakah file-file termasuk kedalam *Malware* atau bukan *Malware*.



Gambar 3. Klasifikasi dengan *Linear Regression*

Gambar 4, garis dari *Linear Regression* dapat melakukan klasifikasi *Malware* dengan baik. Tetapi, bermasalah jika terdapat *Outlier* Data. Terlihat pada gambar 3.



Gambar 4*. Logistic Regression*

Pada gambar 6, terlihat sebuah metode yang dapat menangani permasalahan klasifikasi, sehingga menghasilkan klasifikasi yang lebih baik dan tidak terjadi kegagalan terhadap 2 data kelas positip pada *Linear Regression*, seperti pada gambar 3. Gambar 4 adalah metode *Logistic Regression*, dimana metode tersebut mampu menangani permasalahan data *Outleir*. *Logistic Regression* merupakan algoritma untuk klasifikasi dengan mencari relasi antara fitur (*input*) *Discrete/Continue* dengan probabilitas output pada *Discrete/Continue* tertentu. Beberapa tipe dari *Logistic Regression*:

Binary Logistic Regression: merupakan *Logistic Regression* yang mengklasifikasikan dua output saja. Misalkan: Ya dan Tidak, *Malware* dan bukan *Malware*.

Multinomial Logistic Regression merupakan *Logistic Regression* yang melakukan klasifikasi dua output atau lebih. Misalkan kelas famili *Malware (Ransomware, Banking Malware, SMS Malware, Trojan).*

Ordinal Logistic Regression merupakan *Logistic Regression* yang melakukan klasifikasi dua kelas atau lebih, dengan menghasilkan output yang berurutan. Misalkan membagi kelas siswa ke dalam range Index Prestasi Kumulatif (IPK) seperti 1.0, 2.0, 3.0 sampai 4.0.

Logistic Function merupakan fungsi yang dihasilkan dari persamaan Y didalam Linear *Function* dan Y didalam *Sigmoid Function*. Yang bertujuan untuk menggambarkan data-data yang ke dalam bentuk fungsi Sigmoid.

3.5. Support Vector Machine

Support Vector Machine (SVM) merupakan algoritma pembelajaran mesin klasifikasi yang terawasi (*Supervised Learning*) dengan menggunakan dua garis *vector* (*Hyperplane*) dengan margin yang maksimal. *Hyperplane* merupakan garis pemisah data antar kelompok data. Margin merupakan jarak antara hyperplane dengan jarak terdekat pada

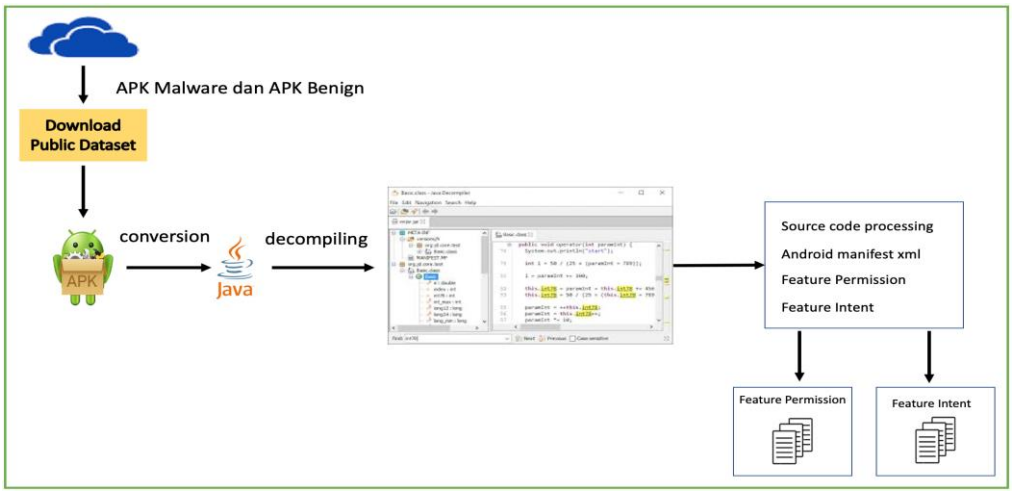
kelompok.Algoritma ini mampu menyelesaikan masalah linear dan non-linear dalam permasalahan klasifikasi. Algoritma SVM, sudah lama berkembang sejak 1960. Dipresentasikan kembali oleh Boser, Vapnik dan Guyon tahun 1992. SVM memiliki performansi yang baik untuk berbagai bidang keilmuan.

Tabel 2. Kelebihan dan kekurangan *Support Vector Machine*

Kelebihan	Kekurangan
Generalisasi difungsikan sebagai kemampuan Support Vector Machine di dalam mengklasifikasikan pola, dimana pola tersebut tidak termasuk dari dataset yang digunakan dalam pembelajaran mesin.	Sulit dipakai dalam problem berskala besar.
Feasibility, implement SVM relative mudah, karena proses penentuan support vector dapat menyelesaikan masalah.	Pada awalnya Support Vector Machine dikembangkan untuk dua klas, tetapi jika dipakai untuk lebih dari dua klas, mengakibatkan penurunan performansinya.

3.6. Download Dataset APK *Malware*

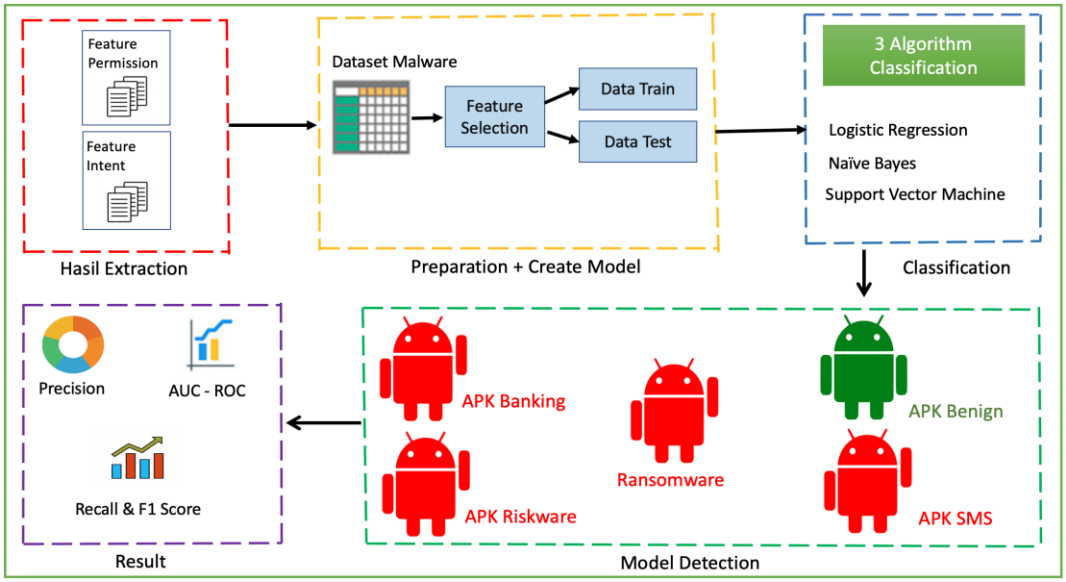
Dataset *Malware* diunduh dari *Canadian Institute for Security, Google Play* dan *APK Pure*. Jenis-jenis Malware yang dipilih adalah Riskware, SMS, Ransomware, Banking dan Benign (Normal APK). Data-data APK, selanjutnya dikumpulkan ke dalam folder-folder sesuai dengan jenis dari kelompok Malware tersebut.



Gambar 5. Download APK Android.

Pada gambar 7, proses ekstraksi dilakukan menggunakan modul Jadx dari APKTOOL, dimana file APK dilakukan *reverse engineering* dan menghasilkan file-file kode sumber, sebelum dilakukan proses kompilasi menjadi *Application Package Kit (APK*). File AndroidManifest.xml hasil dari reverse engineering, dilakukan parse untuk mencari fitur-fitur permission dan disimpan ke dalam data permission. Proses selanjutnya mencari fitur-fitur intent dan disimpan kedalam data intent. Kedua dataset tersebut disimpan dalam bentuk *Comma Separated Values (CSV).*

3.7. Proses Klasifikasi APK *Malware*



Gambar 6. Metodologi yang diusulkan

Pada gambar 8, memperlihatkan rangkaian proses pembelajaran mesin untuk mengklasifikasikan *APK Android Malware* atau bukan *Malware*. Proses dataset yang dilakukan adalah 60:40 (60% dataset training dan 40% dataset testing). Pembagian dataset training dan testing antara lain 70:30, 80:20 dan 90:10. Setelah proses training dilakukan, selanjutnya dilakukan perbandingan kinerja algoritma pembelajaran mesin, seperti *Logistic Regression, Naïve Bayes dan Support Vector Machine.*

3.8. Receiver Operating Characteristic (ROC)

Kinerja pada algoritma *KNN, Naive Bayes dan Decision Tree* dievaluasi menggunakan analisis *Receiver Operating Characteristic (ROC)* untuk mengetahui tingkat akurasi, sensitivitas dan *specificity*.

Sensitifitas = Benar Positip / (Benar Positip + Salah Negatif).

Specificity = Benar Negatif / (Benar Negatif + Salah Positif).

Tabel 3. Kriteria ROC

Nilai AUC	Interpretasi
0.90 - 1.00	*excellent classification*
0.80 - 0.90	*good classification*
0.70 - 0.80	*fair classification*
0.60 - 0.70	*poor classification*
0.50 - 0.60	*failure*

Receiver Operating Characteristic (ROC) merupakan alat untuk mengukur kinerja dari model yang dihasilkan dari klasifikasi.

4. Hasil dan Pembahasan

Pada bagian ini percobaan menggunakan bahasa pemrograman python 3.8, library pandas, numpy, sklearn. Hasil percobaan dengan dataset malware menggunakan algoritma Naive Bayes, dengan perbandingan dataset training : dataset *testing*, 60% : 40%, 70% : 30%, 80% : 20% dan 90%:10%.

Definisi: BP = Benar Positif, BN = Benar Negatif, SP = Salah Positif, SN = Salah *Negative*.

Akurasi adalah rasio prediksi yang benar (positif dan negatif) terhadap seluruh dataset. Akurasi dan jawab pertanyaan “Berapa persentase file APK Android dengan benar memprediksi *Malware* dan Benign dari seluruh kumpulan data file APK Android?”.

Akurasi = (BP + BN) / (BP + SP + SN + BN). Akurasi dapat dilihat pada tabel 4.

Tabel 4. Akurasi dari hasil algoritma klasifikasi

Algorithm	Accuracy			
	60:40	70:30	80:20	90:10
Logistic Regression	86%	93%	85%	91%
Naïve Bayes	94%	98%	99%	100%
Support Vector Machine (SVM)	96%	96%	95%	100%

Presisi merupakan pembagian antara perkiraan true postive dengan hasil perkiraan poitive total. Presisi menjawab pertanyaan “Berapa persentase file APK Android yang benar-benar *Malware* dari total dataset yang diprediksi *Malware*?”
Presisi = (BP) / (BP + SP). Presisi dapat dilihat pada tabel 5.

Tabel 5 *Precision* dari hasil algoritma klasifikasi

Algorithm	Precision			
	60:40	70:30	80:20	90:10
Logistic Regression	86%	96%	86%	93%
Naïve Bayes	95%	98%	99%	100%
Support Vector Machine (SVM)	96%	96%	96%	100%

F1 Score merupakan perbandingan tertimbang dari rata-rata *Precision* dan *Recall*.
F1 Score = 2 * (Recall * Presisi) / (Recall + Presisi). F1-Score dapat dilihat pada tabel 6.

Tabel 6. *F1-Score* dari hasil algoritma klasifikasi

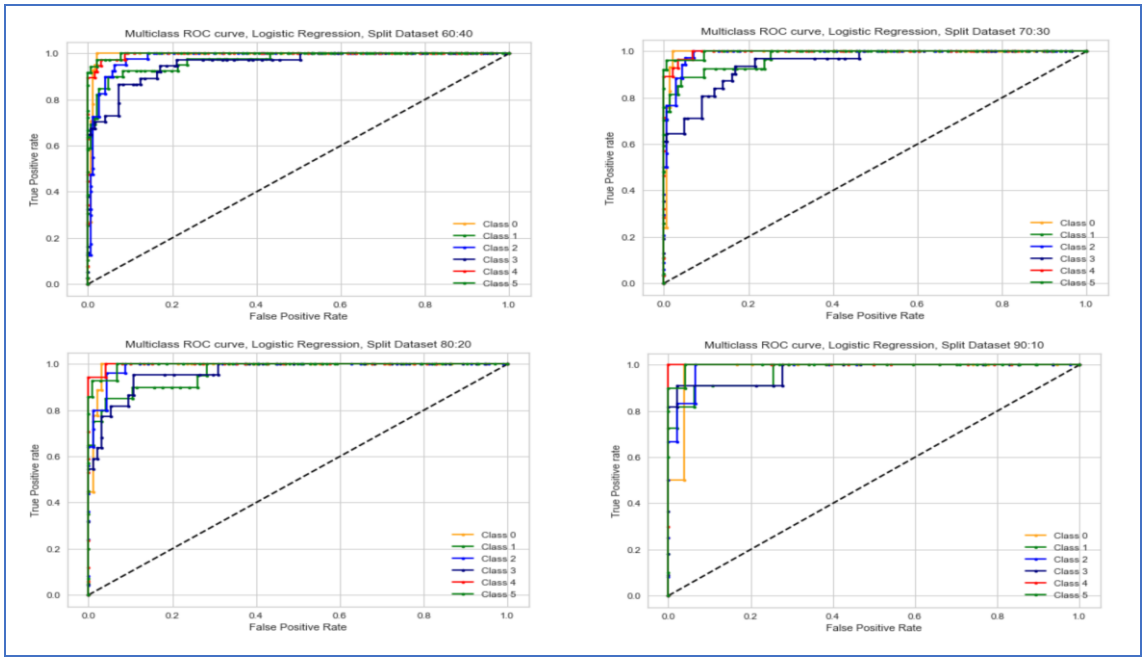
Algorithm	F1-Score			
	60:40	70:30	80:20	90:10
Logistic Regression	85%	91%	83%	90%
Naïve Bayes	94%	98%	99%	100%
Support Vector Machine (SVM)	95%	95%	95%	100%

Recall merupakan rasio prediksi positif benar dibandingkan dengan jumlah total data positif benar. *Recall* menjawab pertanyaan "Berapa persentase file APK Android yang diprediksi sebagai *Malware* dibandingkan dengan malware yang sebenarnya".
Recall = (BP) / (BP + SN). *Recall* dapat dilihat pada tabel 7.

Tabel 7. *Recall* dari hasil algoritma klasifikasi

Algorithm	Recall			
	60:40	70:30	80:20	90:10
Logistic Regression	85%	89%	82%	89%
Naïve Bayes	94%	98%	99%	100%
Support Vector Machine (SVM)	95%	95%	95%	100%

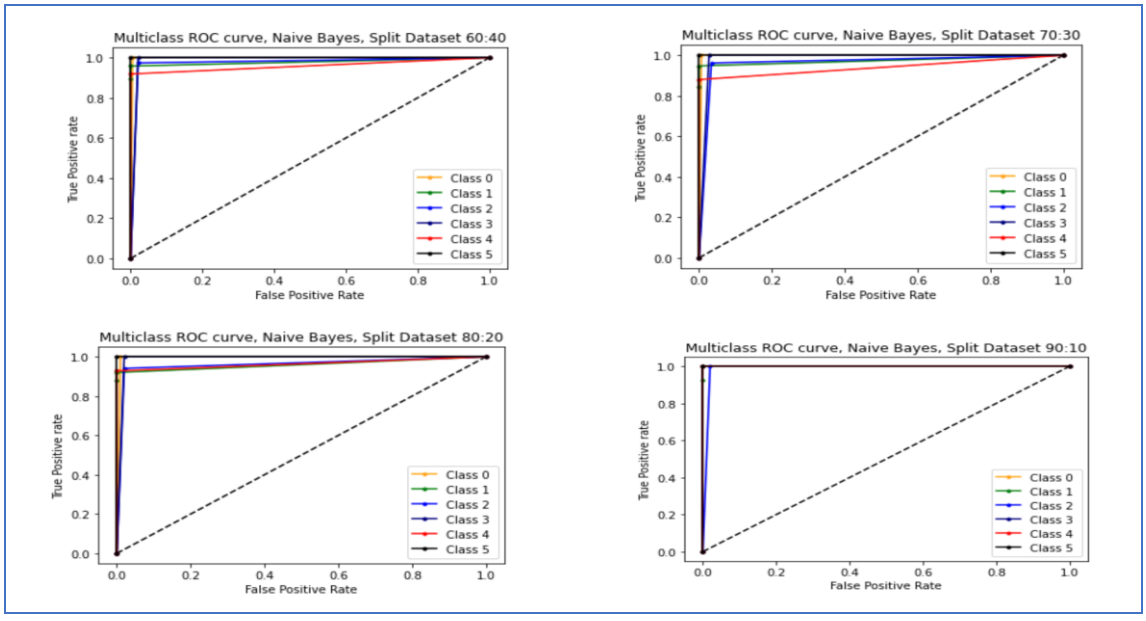
Pengukuran kinerja dari algortima *Logistic Regression* dengan menggunakan *Receiver Operating Characteristic (ROC) Curve pada gambar 7*.



Gambar 7. ROC Logistic Regression

Skor *ROC AUC Logistic Regression* untuk perbandingan *Split Dataset* 60:40 adalah 98,24%, kriteria *Excellent Classification* berdasarkan tabel 3. Skor *ROC AUC Logistic Regression* untuk perbandingan *Split Dataset* 70:30 adalah 98,18%, kriteria *Excellent Classification*. Skor *ROC AUC Logistic Regression* untuk perbandingan *Split Dataset* 80:20 adalah 98,17%, kriteria *Excellent Classification*. Skor *ROC AUC Logistic Regression* untuk perbandingan *Split Dataset* 90:10 adalah 98,40%, kriteria *Excellent Classification.* Rata-rata *True Positive Rate* = 98,25%, rata-rata *False Positive Rate* = 1,75 %

Pengukuran kinerja dari algortima *Naïve Bayes* dengan menggunakan *Receiver Operating Characteristic (ROC) Curve pada gambar 8*.

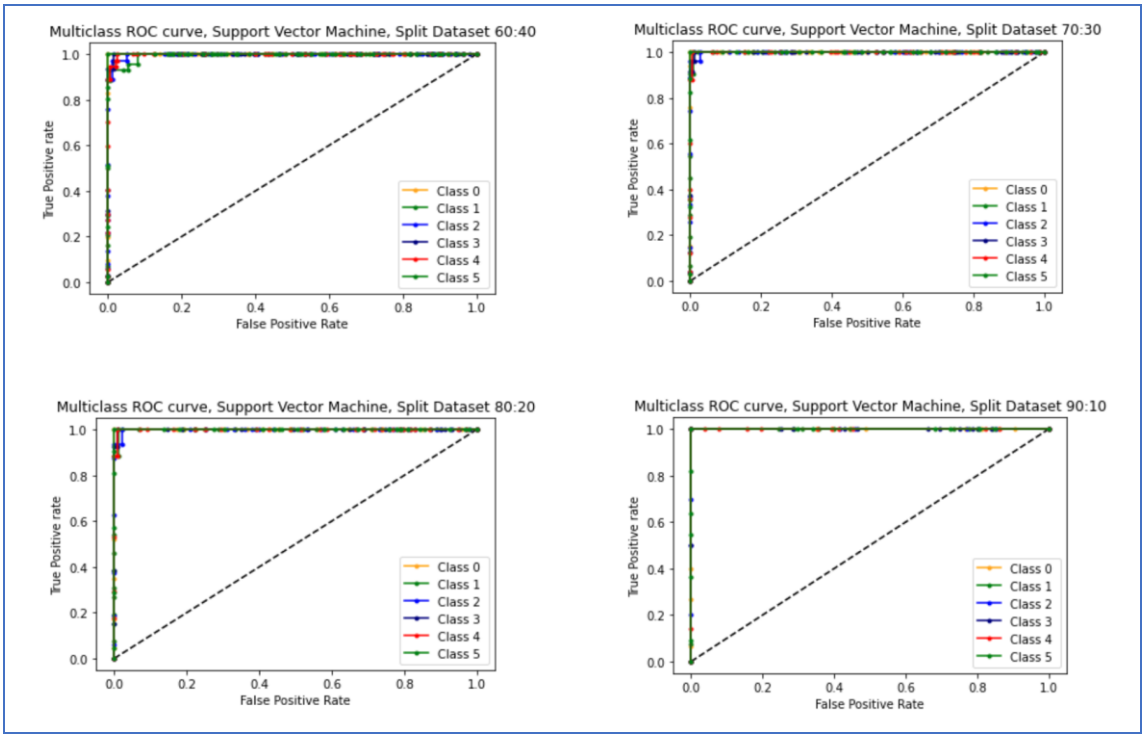


Gambar 8. ROC Naïve Bayes

Skor *ROC AUC Naïve Bayes* untuk perbandingan *Split Dataset* 60:40 adalah 98,33%, kriteria *Excellent Classification* berdasarkan tabel 3. Skor *ROC AUC Naïve Bayes* untuk perbandingan

Split Dataset 70:30 adalah 97,76%, kriteria *Excellent Classification*. Skor *ROC AUC Naïve Bayes* untuk perbandingan *Split Dataset* 80:20 adalah 97,93%, kriteria *Excellent Classification*. Skor *ROC AUC Naïve Bayes* untuk perbandingan *Split Dataset* 90:10 adalah 99,88%, kriteria *Excellent Classification.* Rata-rata *True Positive Rate* = 98,48%, rata-rata *False Positive Rate* = 1,52%.

Pengukuran kinerja dari algortima *Support Vector Machine* dengan menggunakan *Receiver Operating Characteristic (ROC) Curve pada gambar 8*.



Gambar 9. ROC Support Vector Machine

Skor *ROC AUC Support Vector Machine* untuk perbandingan *Split Dataset* 60:40 adalah 99,83%, kriteria *Excellent Classification* berdasarkan tabel 3. Skor *ROC AUC Support Vector Machine* untuk perbandingan *Split Dataset* 70:30 adalah 99,95%, kriteria *Excellent Classification.* Skor *ROC AUC Support Vector Machine* untuk perbandingan *Split Dataset* 80:20 adalah 99,93%, kriteria *Excellent Classification.* Skor *ROC AUC Support Vector Machine* untuk perbandingan *Split Dataset* 90:10 adalah 100%, kriteria *Excellent Classification.* Rata-rata True Positive Rate = 99,93%, rata-rata *False Positive Rate* = 0,07%.

Komparasi hasil penelitian sebelumnya dan metode yang diusulkan peneliti dapat dilakukan sebagai berikut:

Tabel 8. Komparasi dengan penelitian sebelumnya

Penelitian terdahulu	Fitur	Algoritma	Performansi
[12]	*Permission*	*J48*	*Accuracy 99,5%*
[13]	*Permission*	*LSSVM (Least Square Support Vector Machine)*	*Accuracy 98,8%*
[14]	*Permission, Intent*	*Random Forest* *Res7LSTM*	*Accuracy 95,04%* *Accuracy 99,98%*
[7]	*Intent*	*C4,5 Decision Tree*	*Accuracy 99,1%,*
[15]	*permission*	*SVM*	*Average TPR = 98.94 %*

Penelitian terdahulu	Fitur	Algoritma	Performansi
[16]	*Permission*	*Logistic Regression*	*Accuracy 95,5%*
Usulan metode penelitian	*Permission, Intent*	*Logistic Regression*	*Accuracy 88,75%, Average TPR = 98,25%, FPR = 1,75%*
		Naïve Bayes	*Accuracy 97,75%, Average TPR = 98,48%, FPR = 1,52%*
		Support Vector Machine	*Accuracy 96,75%, Average TPR = 99,93%, FPR = 0,07%*

Dari hasil perbandingan dengan penelitian sebelumnya, penelitian dengan judul dari paper "*A hybrid analysis-based approach to android malware family classification*" [14] mirip, sama-sama mengklasifikasi fitur *permission* dan fitur *intent.* Algoritma yang digunakan adalah *Random Forest* dengan akurasi 95,04% dan *deep learning Res7LSTM*. Pada penelitian ini penggunaan algoritma pembelajaran mesin, *Naïve Bayes* dan *Support Vector Machine*, penelitian ini menghasilkan akurasi lebih tinggi 97,75% dan 96,75%.

5. Kesimpulan

Penelitian ini sangat berbeda dengan beberapa penelitian, dimana ekstraksi fitur *Permission* dan fitur *Intent* dengan menggunakan statis analisis, mampu mendeteksi keluarga *Malware*. Penggunaan 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%. Meskipun akurasi tidak setinggi penelitian sebelumnya, teknik analisa statis dengan fitur *Permission* dan fitur *Intent* cukup sederhana untuk mendeteksi file APK Android adalah *Malware* atau bukan *Malware*.

Model *Malware* yang dihasilkan dari training dataset *Malware* dengan ketiga algortima sudah cukup dalam melakukan identifkasi terhadap *Malware* dan keluarga *Malware*. Model dari *Naive Bayes* menjadi peringkat pertama, disusul oleh *Support Vector Machine* dan *Logistic Regression*.

Perbandingan dengan penelitian sebelumnya, akurasinya tidak jauh berbeda. Penelitian ini mirip dengan penelitian dengan fitur permission dan fitur intent menggunakan random *forest* dan *Res7LSTM*. Untuk penggunaan algortima *Machine Learning* penelitian ini unggul dibandingkan dengan penggunaan random forest. Penggunaan *Naïve Bayes* akurasi mendapai 97,75 dan *Support Vector Machine* akurasi mencapai 96,75%.

Daftar Referensi

[1] M. Hussain *et al.*, "Conceptual framework for the security of mobile health applications on Android platform," *Telemat. Informatics*, vol. 35, no. 5, pp. 1335–1354, 2018, doi: 10.1016/j.tele.2018.03.005.

[2] P. Black, I. Gondal, and R. Layton, "A survey of similarities in banking malware behaviours," *Comput. Secur.*, vol. 77, pp. 756–772, 2018, doi: 10.1016/j.cose.2017.09.013.

[3] S. Aonzo, G. C. Georgiu, L. Verderame, and A. Merlo, "Obfuscapk: An open-source black-box obfuscation tool for Android apps," *SoftwareX*, vol. 11, p. 100403, 2020, doi: 10.1016/j.softx.2020.100403.

[4] C. Walls, "Open Source, Embedded Linux, and Android," *Embed. Softw.*, pp. 337–363, 2012, doi: 10.1016/b978-0-12-415822-1.00009-x.

[5] E. Bougiakiotis, "One law to rule them all? The reach of EU data protection law after the Google v CNIL case," *Comput. Law Secur. Rev.*, vol. 42, p. 105580, 2021, doi: 10.1016/j.clsr.2021.105580.

[6] G. Shrivastava and P. Kumar, "Android application behavioural analysis for data leakage," *Expert Syst.*, vol. 38, no. 1, pp. 1–12, 2021, doi: 10.1111/exsy.12468.

[7] C. Wang, Z. Wu, X. Li, X. Zhou, A. Wang, and P. C. K. Hung, "SmartMal: A Service-Oriented Behavioral Malware Detection Framework for Mobile Devices," *Sci. World J.*, vol. 2014, 2014, doi: 10.1155/2014/101986.

[8] E. M. B. Karbab, M. Debbabi, A. Derhab, and D. Mouheb, "MalDozer: Automatic framework for android malware detection using deep learning," *DFRWS 2018 EU - Proc.*

5th Annu. DFRWS Eur., vol. 24, pp. S48–S59, 2018, doi: 10.1016/j.diin.2018.01.007.
[9] X. Su, Q. Gong, Y. Zheng, X. Liu, and K. C. Li, "An Informative and Comprehensive Behavioral Characteristics Analysis Methodology of Android Application for Data Security in Brain-Machine Interfacing," *Comput. Math. Methods Med.*, vol. 2020, 2020, doi: 10.1155/2020/3658795.
[10] S. Garg and N. Baliyan, "Data on Vulnerability Detection in Android," *Data Br.*, vol. 22, pp. 1081–1087, 2019, doi: 10.1016/j.dib.2018.12.038.
[11] J. Abawajy, A. Darem, and A. A. Alhashmi, "Feature subset selection for malware detection in smart iot platforms," *Sensors (Switzerland)*, vol. 21, no. 4, pp. 1–19, 2021, doi: 10.3390/s21041374.
[12] H. Yuan, Y. Tang, W. Sun, and L. Liu, "A detection method for android application security based on TF-IDF and machine learning," *PLoS One*, vol. 15, no. 9 September, pp. 1–19, 2020, doi: 10.1371/journal.pone.0238694.
[13] A. Mahindru and A. L. Sangal, "FSDroid:- A feature selection technique to detect malware from Android using Machine Learning Techniques: FSDroid," *Multimed. Tools Appl.*, 2021, doi: 10.1007/s11042-020-10367-w.
[14] C. Ding, N. Luktarhan, B. Lu, and W. Zhang, "A hybrid analysis-based approach to android malware family classification," *Entropy*, vol. 23, no. 8, 2021, doi: 10.3390/e23081009.
[15] X. Wang, Y. Yang, and Y. Zeng, "Accurate mobile malware detection and classification in the cloud," *Springerplus*, vol. 4, no. 1, pp. 1–23, 2015, doi: 10.1186/s40064-015-1356-1.
[16] H. Yuan, "MADFU : An Improved Malicious Application," *Entropy*, 2020.
[17] M. Rashed and G. Suarez-Tangil, "An Analysis of Android Malware Classification Services," *Sensors*, vol. 21, no. 16, p. 5671, 2021, doi: 10.3390/s21165671.
[18] S. R. T. Mat, M. F. Ab Razak, M. N. M. Kahar, J. M. Arif, S. Mohamad, and A. Firdaus, *Towards a systematic description of the field using bibliometric analysis: malware evolution*, vol. 126, no. 3. Springer International Publishing, 2021.
[19] V. Balakrishnan and W. Kaur, "String-based multinomial naïve bayes for emotion detection among facebook diabetes community," *Procedia Comput. Sci.*, vol. 159, pp. 30–37, 2019, doi: 10.1016/j.procs.2019.09.157.
[20] L. Jiang, S. Wang, C. Li, and L. Zhang, "Structure extended multinomial naive Bayes," *Inf. Sci. (Ny).*, vol. 329, pp. 346–356, 2016, doi: 10.1016/j.ins.2015.09.037.
[21] M. Singh, M. Wasim Bhatt, H. S. Bedi, and U. Mishra, "Performance of bernoulli's naive bayes classifier in the detection of fake news," *Mater. Today Proc.*, no. xxxx, 2020, doi: 10.1016/j.matpr.2020.10.896.
[22] M. Artur, "Review the performance of the Bernoulli Naïve Bayes Classifier in Intrusion Detection Systems using Recursive Feature Elimination with Cross-validated selection of the best number of features," *Procedia Comput. Sci.*, vol. 190, no. 2019, pp. 564–570, 2021, doi: 10.1016/j.procs.2021.06.066.
[23] D. Petschke and T. E. M. Staab, "A supervised machine learning approach using naive Gaussian Bayes classification for shape-sensitive detector pulse discrimination in positron annihilation lifetime spectroscopy (PALS)," *Nucl. Instruments Methods Phys. Res. Sect. A Accel. Spectrometers, Detect. Assoc. Equip.*, vol. 947, no. September, p. 162742, 2019, doi: 10.1016/j.nima.2019.162742.
[24] M. Ontivero-Ortega, A. Lage-Castellanos, G. Valente, R. Goebel, and M. Valdes-Sosa, "Fast Gaussian Naïve Bayes for searchlight classification analysis," *Neuroimage*, vol. 163, pp. 471–479, 2017, doi: 10.1016/j.neuroimage.2017.09.001.
[25] C.-W. Tsai, Y.-P. Chen, T.-C. Tang, and Y.-C. Luo, "An efficient parallel machine learning-based blockchain framework," *ICT Express*, no. xxxx, pp. 0–7, 2021, doi: 10.1016/j.icte.2021.08.014.
[26] O. E. Gundersen, S. Shamsaliei, and R. J. Isdahl, "Do machine learning platforms provide out-of-the-box reproducibility?," *Futur. Gener. Comput. Syst.*, vol. 126, pp. 34–47, 2022, doi: 10.1016/j.future.2021.06.014.
[27] A. A. H. Ahmadini, "A novel technique for parameter estimation in intuitionistic fuzzy logistic regression model," *Ain Shams Eng. J.*, no. xxxx, 2021, doi: 10.1016/j.asej.2021.06.004.
[28] R. Verma, N. Bhardwaj, P. D. Singh, A. Bhavsar, and V. Sharma, "Estimation of sex

through morphometric landmark indices in facial images with strength of evidence in logistic regression analysis," *Forensic Sci. Int. Reports*, vol. 4, p. 100226, 2021, doi: 10.1016/j.fsir.2021.100226.

[29] W. Książek, M. Gandor, and P. Pławiak, "Comparison of various approaches to combine logistic regression with genetic algorithms in survival prediction of hepatocellular carcinoma," *Comput. Biol. Med.*, vol. 134, 2021, doi: 10.1016/j.compbiomed.2021.104431.

[30] P. Taraba, "Linear regression on a set of selected templates from a pool of randomly generated templates," *Mach. Learn. with Appl.*, vol. 6, no. May, p. 100126, 2021, doi: 10.1016/j.mlwa.2021.100126.

[31] K. Brzeziński, K. Józefiak, and A. Zbiciak, "On the interpretation of shear parameters uncertainty with a linear regression approach," *Meas. J. Int. Meas. Confed.*, vol. 174, no. May 2020, 2021, doi: 10.1016/j.measurement.2020.108949.