# LAPORAN PROYEK MATA KULIAH

# 10S3001 - KECERDASAN BUATAN

## Network Traffic Analysis for Android Malware Detection using Support Vector Machine (SVM) Classification



## Disusun Oleh :

12S21027 Rebecca Yulyartha Bulawan Sihombing

12S21037 Immanuella Eklesia Lumbantobing

12S21039 Widya Indah Sari Manurung

12S21053 Chesya Ivana J. M. Sitorus

12S21058 Grace Christina Yohanna Situmorang


**Tautan GitHub** : CERTAN-23-01-Network Traffic Analysis for Android

**Tautan Kaggle** : CERTAN-23-01-Network Traffic Analysis for Android

---


**PROGRAM STUDI SARJANA SISTEM INFORMASI**

**FAKULTAS INFORMATIKA DAN TEKNIK ELEKTRO**

**INSTITUT TEKNOLOGI DEL**


**NOVEMBER 2023**

---


Nama Dokumen: LP-PBDSI-23-01 Tanggal : 5 December 2023 Jumlah Halaman : 67

# DAFTAR ISI

DAFTAR ISI ..... 2
DAFTAR TABEL ..... 4
DAFTAR GAMBAR ..... 5
1. Pendahuluan ..... 6
1.1 Latar Belakang ..... 6
1.2 Tujuan ..... 7
1.3 Manfaat ..... 8
1.4 Ruang Lingkup ..... 8
1.5 Istilah dan Singkatan ..... 9
2. Studi Literatur ..... 11
2.1 Malware ..... 11
*2.2 SVM (Support Vector Machine)* ..... 11
2.3 Kernel RBF ..... 12
2.4 Penelitian Terdahulu ..... 13
*2.5* Tingkat Keberhasilan *Support Vector Machine Classification* ..... 15
3. Metode ..... 16
3.1 Perumusan Masalah ..... 17
3.2 Studi Literatur ..... 17
3.3 Pengumpulan Data ..... 17
3.4 Pemrosesan Data ..... 18
3.5 Pemisahan Data ..... 18
3.6 Normalisasi Data ..... 19
3.7 Model *Support Vector Machine (SVM)* ..... 20
3.7.1 Penentuan Rentang Nilai Parameter untuk Penalaan Hiperparameter ..... 20
3.7.2 Pembuatan Model SVM dan Penggunaan Grid Search ..... 22
3.7.3 Inisialisasi dan Pelatihan Model SVM dengan Parameter Terbaik ..... 23
3.8 Evaluasi *(Results)* ..... 23
3.8.1 Kriteria Evaluasi Model ..... 23
3.8.2 Pengukuran Akurasi ..... 24
3.8.3 Evaluasi Klasifikasi ..... 24

3.8.4 Analisis Hasil dan Penyempurnaan Model ..... 26
3.9 Visualisasi Data ..... 26
3.9.1 Matriks Kebingungan *(Confusion Matrix)* ..... 26
3.9.2 Kurva ROC *(Receiver Operating Characteristic)* ..... 27
4. Hasil Pengujian ..... 29
5. Analisis ..... 52
5.1 Analisis Data Sebelum Implementasi SVM ..... 52
5.1.1 Visualisasi Data Heatmap ..... 53
5.1.2 Analisis Hasil Visualisasi Data Confusion Matrix ..... 54
5.1.3 Analisis Visualisasi Data Receiver Operating Characteristic (ROC) Kurve ..... 55
5.1.4 Analisis Visualisasi Data Precision-Recall Curve ..... 56
5.2 Analisis Data Setelah Implementasi SVM ..... 58
5.2.1 Analisis Data pada Visualisasi Data Hyperplane 2D ..... 58
5.2.2 Analisis Data pada Visualisasi Data Hyperplane 3D ..... 60
6 Kesimpulan ..... 62
6.1 Kesimpulan ..... 62
6.2 Saran ..... 62
REFERENSI ..... 63
LAMPIRAN ..... 67
Lampiran 1. Pembagian Tugas Kelompok 10 ..... 67

# DAFTAR TABEL

Tabel 1. Istilah atau Singkatan Proyek..... 9
Tabel 2. Pembagian Tugas Anggota Kelompok ..... 67

## DAFTAR GAMBAR

Gambar 1. Diagram Tahapan Pengembangan Model Klasifikasi Android Malware Menggunakan SVM ..... 16
Gambar 2. Hyperplanes di tampilan 2D dan 3D ..... 21
Gambar 3. Batas Keputusan dan Klasifikasi Poin Pelatihan Hyperplane ..... 22
Gambar 4. Visual Pembagian data Dimensi tinggi ..... 23
Gambar 5. Optimal Hyperplane using the SVM Algorithm ..... 23
Gambar 6. Heatmap Korelasi Dataset Malware ..... 34
Gambar 7. Scatter plot Dataset Klasifikasi Malware ..... 40
Gambar 8. Hyperplane SVM in 2D ..... 42
Gambar 9. SVM Hyperplane in 3D ..... 45
Gambar 10. Confusion Matrik Klasifikasi Dataset Malware Android Traffic ..... 47
Gambar 11. Kurva ROC Klasifikasi Data Android Traffic Malware ..... 49
Gambar 12. Grafik 3D Grid Search Kombinasi Parameter C dan Gamma Pada Model SVM Untuk Mencari Nilai Optimal ..... 51
Gambar 13. Scatter Plot Visualisasi Dataset Klasifikasi Malware ..... 52
Gambar 14. Heatmap Korelasi Malware ..... 53
Gambar 15. Visualisasi Data Confusion Matrix Klasifikasi Malware ..... 54
Gambar 16. Kurva ROC Klasifikasi Dataset Android Malware ..... 55
Gambar 17. Precision-Recall Curve Klasifikasi Data Malware ..... 56
Gambar 18. Hyperplane SVM in 2D ..... 58
Gambar 19. Hyperplane SVM in 3D ..... 60

# 1. Pendahuluan

## 1.1 Latar Belakang

Di era digital yang terus berkembang, internet menjadi alat utama dalam mempermudah berbagai aktivitas sehari-hari. Namun, sayangnya, semakin banyak orang yang menggunakan internet, semakin banyak juga kejahatan yang terjadi [1]. Salah satu bentuk serangan yang umum adalah penggunaan perangkat lunak jahat, yang lebih dikenal sebagai malware [2]. Hal ini ditandai dengan semakin beragamnya serangan malware yang digunakan oleh para penyerang, seperti *virus, worm, trojan horse, rootkit, spyware, adware,* dan lainnya, memiliki tujuan merusak atau mengganggu sistem perangkat pengguna [3]. Masing-masing jenis malware memiliki cara kerja yang berbeda, tetapi semuanya memiliki tujuan untuk membahayakan pengguna.

Pentingnya keamanan digital tidak bisa diabaikan mengingat dampak negatif serangan malware, seperti risiko pencurian data pribadi, pengambilalihan kendali perangkat, dan penyebaran konten berbahaya [4]. Saat ini, para penyerang malware banyak mencari celah untuk menyembunyikan malware mereka dan menghindari deteksi oleh sistem keamanan. Salah satu cara yang sering digunakan adalah dengan menyembunyikan malware di dalam file atau program lain yang tampak tidak berbahaya, seperti dalam file PDF, dokumen Microsoft Office, atau bahkan file musik [5]. Cara lain yang sering digunakan adalah dengan menggunakan teknik enkripsi untuk menyembunyikan kode malware [6]. Ancaman serangan malware juga dilakukan dengan penyebaran konten berbahaya melalui aplikasi dari sumber yang tidak resmi, menciptakan saluran untuk serangan siber yang lebih luas [7]. Aplikasi-aplikasi ini dapat diunduh dari situs web atau toko aplikasi yang tidak terpercaya dan berisi malware yang dapat menginfeksi perangkat pengguna. Hal ini tentunya menjadi masalah serius yang berdampak pada ekosistem perangkat pengguna. Hal ini terjadi karena diversifikasi malware membuat perangkat lunak jahat semakin sulit dideteksi dikarenakan serangan malware terus berkembang dan lebih luas menyerang perangkat pengguna.

Dalam menghadapi masalah ini, perlu dilakukan analisis data lebih lanjut untuk mengetahui pola dinamika serangan malware yang banyak terjadi. Salah satu caranya dengan menggunakan metode klasifikasi. Metode klasifikasi menjadi esensi penting untuk

memahami jenis-jenis malware yang mengancam. Klasifikasi dapat digunakan untuk mengelompokkan malware ke dalam kategori-kategori berdasarkan karakteristik dan perilakunya. Namun, metode klasifikasi mengadaptasi berbagai algoritma untuk mengelompokkan data. Salah satu metode klasifikasi yang paling efektif untuk mengklasifikasi adalah algoritma *Support Vector Machine (SVM).* Algoritma SVM mampu membangun model yang efektif dalam mengenali pola-pola khas dari berbagai jenis malware [8].

SVM merupakan metode klasifikasi yang efektif untuk menyelesaikan masalah pengelompokan, terutama ketika kita berurusan dengan data yang memiliki banyak fitur atau karakteristik [9]. Algoritma ini bukan hanya efektif dalam konteks klasifikasi malware, tetapi juga telah terbukti berhasil dalam berbagai penelitian lainnya. Hal ini dikarenakan algoritma SVM mempunyai akurasi klasifikasi yang lebih konstan dibandingkan dengan algoritma lainnya [10]. Penelitian sebelumnnya dilakukan oleh Maglogiannis dan Zafiropolous (2007) melakukan diagnosis dan prognosis breast cancer dengan menggunakan SVM [4]. Hasil penelitian tersebut menunjukkan bahwa ketepatan klasifikasi menggunakan SVM mencapai 97%. Rachman dan Purnami (2012) melakukan penelitian terkait klasifikasi tingkat keganasan kanker menggunakan metode regresi logistik dan SVM. Hasilnya menunjukkan bahwa penggunaan SVM memberikan tingkat akurasi yang lebih tinggi, mencapai 98,11% [11].

Berdasarkan hasil kontribusi penelitian yang positif dan pemahaman atas masalah yang telah diuraikan sebelumnya, diharapkan proyek ini akan mengembangkan model untuk mengklasifikasi berbagai jenis malware yang mengancam pengguna Android dengan menggunakan algoritma *Support Vektor Machine* (SVM). Dengan demikian, proyek ini diharapkan dapat memberikan solusi yang efektif dalam menganalisis ancaman keamanan digital pada platform Android melalui penerapan SVM dalam mengklasifikasi malware.

## 1.2 Tujuan

Penelitian ini memiliki tujuan, antara lain:

a. Mengembangkan model untuk mengklasifikasi berbagai jenis malware yang mengancam pengguna Android dengan menggunakan algoritma Support Vektor Machine (SVM).

b. Mengevaluasi penerapan klasifikasi malware Android menggunakan algoritma SVM dengan mengukur keakuratan pemodelan SVM.

### 1.3 Manfaat

Penelitian ini diharapkan dapat memberikan kontribusi, diantaranya:

1. Memberikan pemahaman bagi pengguna Android terkait jenis malware yang mengancam keamanan digital pada platform Android.
2. Menyediakan landasan untuk pengembangan sistem keamanan lebih lanjut yang dapat secara proaktif mengatasi ancaman malware Android.
3. Menjadi referensi bagi peneliti lain dalam mengembangkan metode klasifikasi malware yang lebih efektif.

### 1.4 Ruang Lingkup

Ruang lingkup proyek ini mencakup beberapa aspek yang akan dibahas dan diimplementasikan. Adapun ruang lingkup proyek ini meliputi:

1. Dataset yang digunakan dalam penelitian ini menggunakan data public yang tersedia melalui halaman website Dataset Kaggle yang diakses melalui link berikut: https://www.kaggle.com/datasets/xwolf12/network-traffic-android-malware
2. Algoritma SVM yang digunakan adalah algoritma SVM dengan kernel linear.
3. Dataset yang digunakan adalah dataset yang bersifat numerik. Fitur-fitur atau kolom-kolom dalam dataset ini berisi nilai numerik yang mencerminkan berbagai aspek aktivitas jaringan pada sistem Android.
4. Pembatasan variabel yang tidak diperlukan adalah yang terdaftar dalam variabel unwanted_columns**,** seperti kolom duracion, avg_local_pkt_rate, avg_remote_pkt_rate, source_app_packets.
5. Pembagian data menjadi set pelatihan dan pengujian dilakukan dengan proporsi tetap (test_size=0.2) dan klasifikasi target hanya terdiri dari dua kelas (“benign” dan “malware”). Tipe kelas benign termasuk ancaman yang tidak mengandung malware. Sebaliknya, malware yang paling berbahaya.
6. Evaluasi kinerja akan melibatkan metrik-metrik standar seperti akurasi, presisi, recall, dan F1 score untuk mengukur dampak malware pada jaringan Android.

## 1.5 Istilah dan Singkatan

Istilah atau singkatan yang digunakan dalam pengembangan proyek, dijabarkan pada tabel berikut:

**Tabel 1.** Istilah atau Singkatan Proyek

| Istilah atau Singkatan | Definisi atau Kepanjangan |
|---|---|
| Agen Cerdas | Entitas pemrosesan yang dapat membuat keputusan cerdas berdasarkan informasi yang diterimanya. |
| Akurasi | Persentase data uji yang diklasifikasikan dengan benar oleh model. |
| Android | Sistem operasi seluler |
| Dataset | Kumpulan data yang digunakan untuk pelatihan model klasifikasi |
| F1 Score | Harmonic mean dari presisi dan recall, memberikan gambaran keseluruhan kinerja model klasifikasi. |
| Kaggle | Platform kompetisi dan sumber daya untuk data ilmiah dan proyek analisis data. |
| Kernel RBF | Kernel yang digunakan dalam algoritma SVM untuk merepresentasikan data |
| Malware | Perangkat lunak berbahaya yang dirancang untuk merusak, mengakses, atau mengambil alih sistem atau data tanpa izin. |
| Metode Klasifikasi | Teknik yang digunakan untuk mengelompokkan data ke dalam kategori tertentu |
| Perangkat Lunak | Serangkaian instruksi atau program komputer yang dirancang untuk melakukan fungsi-fungsi tertentu, seperti pengolah kata dan peramban web, hingga sistem operasi yang mengontrol operasi dasar komputer. |
| Preprocessing Data | Serangkaian langkah untuk membersihkan, mengubah, atau mengorganisir data agar sesuai dengan kebutuhan analisis. |
| Presisi | Rasio antara jumlah *true positive* dan total hasil yang diprediksi sebagai positif oleh model klasifikasi. |

| Istilah atau Singkatan | Definisi atau Kepanjangan |
|---|---|
| Recall | Rasio antara jumlah true positive dan total jumlah sampel yang sebenarnya positif. |
| SVM | *Support Vector Machine*, algoritma pembelajaran mesin untuk klasifikasi dan regresi. |

Tabel di atas berisi istilah dan singkatan yang mungkin dijumpai dalam dokumen ini, beserta definisi atau kepanjangannya masing-masing. Hal ini bertujuan untuk memastikan pemahaman yang konsisten dan jelas terhadap terminologi yang digunakan dalam konteks proyek pengklasifikasian malware pada jaringan Android.

## 2. Studi Literatur

### 2.1 Malware

Malware, singkatan dari *malicious software*, merujuk pada perangkat lunak berbahaya yang dirancang untuk merusak, mengakses, atau mengambil alih sistem atau data tanpa izin [12]. Jenis malware melibatkan berbagai ancaman, termasuk *virus, worm, trojan horse, rootkit, spyware*, dan *adware.* Masing-masing jenis memiliki karakteristik dan tujuan yang berbeda, tetapi semuanya dapat menimbulkan dampak negatif terhadap perangkat pengguna, seperti pencurian data pribadi, pengambilalihan kendali perangkat, dan penyebaran konten berbahaya [8].

Malware dirancang dengan menyisipkan kode-kode secara tersembunyi oleh penyerang, memungkinkan mereka untuk tetap beroperasi di dalam sistem komputer tanpa pemilik sistem menyadarinya selama periode waktu tertentu. Aktivitas mencurigakan dari malware seringkali melibatkan pemanfaatan lalu lintas jaringan sebagai sarana untuk mengirimkan informasi rahasia seperti data rekening bank, pesan pribadi, dan kata sandi. Selain itu, malware juga dapat menggunakan lalu lintas jaringan sebagai pintu belakang, memungkinkan malware lain untuk masuk dan berkembang [13]. Hal ini menandakan bahwa malware memiliki kemampuan untuk meresahkan keamanan sistem dengan memanfaatkan komunikasi melalui jaringan, yang dapat mengakibatkan potensi kebocoran data dan kerentanan terhadap serangan lanjutan. Keberadaan malware sangat berpotensi merugikan individu atau perusahaan karena dapat menyebabkan pencurian data dan informasi pribadi, meretas keamanan program dan sistem operasi, menghapus data yang penting, serta melakukan pemantauan terhadap perilaku online korban, dan ancaman lainnya. Kehadiran malware membawa dampak serius dan kompleks, menjadikannya sebagai ancaman serius yang perlu mendapat perhatian dan tindakan pencegahan secara menyeluruh [14].

### *2.2 SVM (Support Vector Machine)*

*Support Vector Machine (SVM)* adalah algoritma pembelajaran mesin yang digunakan untuk melakukan klasifikasi dan regresi. SVM bekerja dengan mencari batas keputusan yang optimal, atau hyperplane, yang dapat memisahkan dua kelas data dengan margin maksimal [8]. SVM bekerja dengan membangun model berdasarkan data pelatihan yang

dikategorikan. Model ini kemudian digunakan untuk mengklasifikasikan data baru ke dalam salah satu kelas yang telah ditentukan. Pada algoritma *Support Vector Machine* (SVM), hal ini dikenal sebagai optimalisasi *hyperplane*. *Hyperplane* ini berfungsi sebagai batas pemisah antara *support vector* dari kelas satu dan kelas lainnya. Tujuan utamanya adalah mengoptimalkan *support vector*, khususnya yang berdekatan antara kelas satu dan kelas lainnya. *Support vector* ini menjadi acuan untuk menentukan batas klasifikasi agar *hyperplane* yang dibuat menjadi seoptimal mungkin. *Support vector* berasal dari dataset yang telah diubah menjadi nilai vektor setelah proses ekstraksi fitur. Sebagai contoh, dalam dataset pelatihan, kita memiliki pasangan nilai x dan y dalam bentuk $\{(x1, y1), \dots, (xn, yn)\}$, di mana x adalah vektor dan y adalah label kelasnya [15].

Dalam konteks klasifikasi malware pada platform Android, SVM dapat digunakan untuk mengenali pola-pola khas dari berbagai jenis malware berdasarkan fitur-fitur tertentu, seperti perilaku jaringan, pola akses, dan tanda-tanda khas lainnya [16]. Melalui proses ini, SVM dapat membantu dalam membedakan antara perangkat lunak yang aman dan malware.

### 2.3 Kernel Linear

Sebelum memulai proses klasifikasi, banyak peneliti memilih untuk menggunakan kernel linear. Pemilihan salah satu jenis fungsi kernel yang digunakan dalam algoritma Support Vector Machine (SVM) untuk memproses dan memahami data [17]. Fungsi kernel ini memiliki karakteristik khusus yang membuatnya efektif untuk menangani kasus-kasus di mana hubungan antar fitur data bersifat linier atau sejaja, seperti halnya klasifikasi pada data malware ini. Dengan kata lain, kernel linear cocok digunakan ketika pola atau hubungan antar data dapat dijelaskan melalui garis lurus. Kelebihan kernel linear terletak pada kemampuannya dalam menangani dataset dengan fitur-fitur yang memiliki ketergantungan linier, sehingga dapat memberikan hasil yang optimal dalam beberapa kasus pemodelan dan klasifikasi data [18].

Dalam analisis menggunakan kernel linear, parameter yang dapat disesuaikan untuk optimalisasi adalah C atau Cost. Proses penyetelan optimal parameter C biasanya dilakukan melalui percobaan berulang dan eksperimen. Pada penelitian yang dilakukan oleh Pangestu

dan Anna (2020), mereka menggunakan tiga jenis kernel berbeda, yaitu kernel linear, kernel polynomial, dan kernel rbf. Dari ketiga jenis kernel tersebut, hasil penelitian menunjukkan bahwa kernel linear memberikan hasil yang paling baik dibandingkan dengan kernel lainnya. Dengan kata lain, penggunaan kernel linear memberikan kinerja yang optimal dalam konteks penelitian yang dilakukan oleh para peneliti tersebut [19].

## 2.4 Penelitian Terdahulu

Beberapa penelitian terdahulu telah dilakukan dalam mengimplementasikan SVM untuk klasifikasi dan deteksi malware. Watson et al. (2015) melakukan penelitian bertujuan untuk mengembangkan pendekatan deteksi malware di dalam infrastruktur *cloud computing* dengan menggunakan algoritma *Support Vector Machines (SVM)* satu kelas, teknik analisis statis dan dinamis, serta pendekatan adaptif untuk mengklasifikasi ancaman baru secara real-time dengan biaya komputasi yang minim. Para peneliti juga melakukan eksperimen dengan sampel malware, seperti *Trojan.Kelihos-5, Trojan.Zbot-1433, Trojan.Zbot-1023, Trojan.Zbot-18,* dan *Trojan.Zbot-385* untuk menganalisis perilaku dan karakteristiknya di lingkungan cloud. Dalam penelitian ini, para peneliti menciptakan cara baru untuk mendeteksi aktivitas mencurigakan di tingkat server cloud menggunakan algoritma *Support Vector Machines (SVM)*. SVM ini seperti detektif pintar yang diajarkan untuk mengenali perangkat lunak jahat dan serangan. Hasil pengujian menunjukkan bahwa SVM ini sangat baik dalam mengenali perangkat lunak jahat dengan tingkat keakuratan di atas 90% [20].

Berdasarkan hasil penelitian yang dilakukan oleh *Deutsche Telekom Laboratories*, metode deteksi malware menggunakan pendekatan *Support Vector Machines (SVM)* satu kelas dijelaskan dengan rinci. SVM dilatih dengan data normal dan mampu mengklasifikasikan anomali secara real-time dengan biaya komputasi yang minim. Metode ini juga melibatkan teknik analisis statis dan dinamis serta seleksi fitur untuk membangun model deteksi malware. Hasil penelitian menunjukkan bahwa sistem ini dapat diandalkan, melatih dengan cepat, dan tahan terhadap kegagalan komponen. Dengan demikian, sistem ini terbukti efisien dalam mendeteksi anomali dalam jaringan seluler dengan keandalan dan ketahanan yang baik terhadap kegagalan komponen [21].

Sejumlah penelitian terdahulu telah berhasil menggambarkan aplikasi SVM dalam konteks keamanan siber dan deteksi malware. Sementara, penelitian oleh Turnip et al. (2020) fokus pada deteksi malware Android menggunakan teknik machine learning, termasuk SVM dan penggunaan eXtreme Gradient Boosting *(XGBoost)* dalam mendeteksi dan mengklasifikasikan malware berbasis Android berdasarkan kategori izin *(permission)* APK. [22]. Model ini digunakan untuk mengklasifikasikan enam kelas malware, seperti *Trojan-Ransom, Trojan-SMS, Trojan-Banker, RiskTool, Trojan,* dan lainnya. Secara keseluruhan, hasil evaluasi model menunjukkan bahwa model klasifikasi SVM memberikan kinerja lebih baik dengan F1-score sebesar 74,40% [22].

Penelitian yang dilakukan oleh Moh Yamin pada tahun 2014 membahas klasifikasi tuberkulosis menggunakan metode Support Vector Machine (SVM). Tujuan penelitian tersebut adalah untuk menemukan hyperplane terbaik yang dapat memisahkan dua kelas pada ruang input. Penulis menjelaskan bahaya penyakit tuberkulosis, terutama jika tidak diobati dengan cepat. Hasil klasifikasi menggunakan metode SVM menunjukkan tingkat akurasi sebesar 98%, membuktikan bahwa algoritma SVM memberikan solusi yang sangat baik dalam mengidentifikasi kasus *tuberculosis* [23].

Selanjutnya, penelitian dilakukan oleh Zuriel et.al (2021), mengklasifikasi Support Vector Machine untuk menganalisa sentimen pengguna Twitter terhadap kebijakan PSBB. Hasil penelitian ini menunjukkan bahwa performa model klasifikasi SVM dengan kernel RB. Model klasifikasi dengan kernel RBF ini memberikan mengklasifikasikan 11.764 (52.7%) data tweet ke dalam kelas positif dan 10.571 (47.3%) data tweet ke dalam kelas negative. Hasil tersebut memberikan kesimpulan bahwa pengguna Twitter cenderung bersentimen positif terhadap kebijakan PSBB [24]

Hasil dari penelitian-penelitian tersebut memberikan dukungan untuk penggunaan SVM dalam konteks deteksi malware, termasuk pada platform Android. Dengan merinci cara kerja SVM dan memahami penelitian terdahulu, proyek ini dapat memanfaatkan keunggulan algoritma ini untuk mengidentifikasi dan mengklasifikasikan jenis-jenis malware pada platform Android dengan lebih efektif.

### *2.5* Tingkat Keberhasilan *Support Vector Machine Classification*

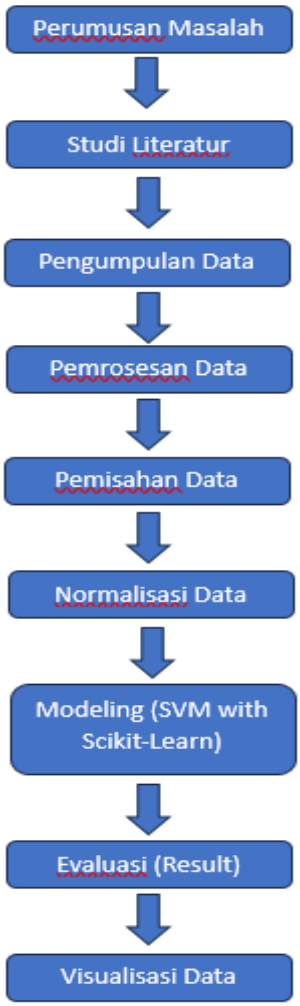
Implementasi SVM satu kelas dalam konteks deteksi malware telah membuktikan keunggulannya dengan mencapai tingkat keberhasilan yang luar biasa. Dengan pelatihan dan penyesuaian yang cermat, SVM mampu memberikan tingkat akurasi deteksi yang melebihi 90% untuk semua sampel malware, termasuk dalam kondisi analisis statis maupun migrasi. Keberhasilan ini menandakan kemampuan SVM dalam mengenali pola yang kompleks dan bervariasi yang terkait dengan aktivitas malware pada platform Android [16].

Lebih lanjut, keunggulan SVM tidak hanya terbatas pada akurasi tinggi, namun juga pada kemampuannya untuk mendeteksi anomali secara online dengan biaya waktu minimal. Dalam situasi di mana respons cepat terhadap ancaman malware sangat penting, SVM menunjukkan kinerja yang luar biasa dengan tingkat akurasi deteksi secara keseluruhan yang tetap melampaui 90% dalam kebanyakan kasus [8]. Kemampuan SVM dalam deteksi malware tidak hanya terletak pada tingkat keberhasilan tinggi, tetapi juga pada fleksibilitasnya dalam menangani berbagai kondisi dan metode analisis. SVM mampu mempertahankan kinerjanya yang unggul baik dalam situasi analisis statis, di mana karakteristik statis malware dianalisis tanpa eksekusi, maupun dalam skenario migrasi di mana perilaku malware dipantau selama aktivitas di dalam sistem [12].

Secara keseluruhan, prestasi SVM dalam mengklasifikasikan dan mendeteksi malware pada platform Android menciptakan landasan yang solid untuk pengembangan sistem keamanan yang andal. Keberhasilannya tidak hanya terletak pada tingkat akurasi tinggi, tetapi juga pada kemampuannya untuk beradaptasi dengan perubahan kondisi dan taktik yang digunakan oleh malware. Dengan mempertimbangkan kecepatan responsnya yang tinggi dan biaya waktu yang minimal dalam mendeteksi anomali secara online, SVM muncul sebagai pilihan yang kuat dalam upaya melawan ancaman malware yang terus berkembang [25].

## 3. Metode

Dalam rangka mengklasifikasikan malware ke dalam kategori malware tertentu, penelitian ini merancang sebuah model *Machine Learning* (ML) yang inovatif. Adapun model klasifikasi *Machine Learning* yang digunakan pada penelitian ini adalah menggunakan Support Vector Machine (SVM). Dalam menerapkannya, terdapat tahapan-tahapan utama yang akan dijalankan untuk mengklasifikasikan Malware menggunakan Support Vector Machine (SVM). Adapun tahapan-tahapan tersebut mencakup Perumusan Masalah, Studi Literatur, Pengumpulan Data, Pemrosesan Data, Pemisahan Data, Normalisasi Data, Modeling ( SVM with Scikit-Learn), Evaluasi (Result), dan Visualisasi Data..




Gambar 1. Diagram Tahapan Pengembangan Model Klasifikasi Android Malware Menggunakan SVM

### 3.1 Perumusan Masalah

Pada tahap ini, peneliti akan mendefinisikan masalah yang akan diselesaikan, yaitu mengklasifikasikan malware ke dalam kategori malware tertentu. Selain itu, peneliti juga akan menentukan tujuan dari penelitian, yaitu untuk mengembangkan model klasifikasi malware yang akurat dan efektif. Rumusan masalah dilakukan langkah awal dalam sebuah penelitian atau proyek yang melibatkan identifikasi, penjelasan, dan pembatasan permasalahan atau isu tertentu yang akan diteliti atau diselesaikan pada pengklasifikasian malware algoritma SVM. Proses ini membantu peneliti untuk secara jelas merinci permasalahan yang akan diinvestigasi, memberikan landasan yang kuat, dan mengarahkan arah penelitian atau proyek tersebut [26].

### 3.2 Studi Literatur

Pada tahap ini, peneliti akan melakukan kajian terhadap literatur yang relevan dengan penelitian, seperti penelitian-penelitian terdahulu yang telah dilakukan dalam bidang klasifikasi malware. Kajian literatur ini bertujuan untuk mendapatkan pemahaman yang lebih baik tentang masalah yang akan diselesaikan, serta untuk mendapatkan referensi dalam pengembangan model klasifikasi malware menggunakan algoritma *Support Vector Machine* (SVM). Studi literatur dilakukan karena melibatkan analisis, sintesis, dan evaluasi berbagai sumber informasi, termasuk jurnal ilmiah, buku, artikel, dan publikasi lainnya yang relevan dengan topik penelitian, yakni pengembangan model klasifikasi algoritma SVM. Tujuan dari studi literatur adalah untuk memahami perkembangan pengetahuan terkini, mengidentifikasi celah penelitian, dan menyusun dasar teoritis yang kokoh untuk mendukung penelitian yang akan dilakukan [27].

### 3.3 Pengumpulan Data

Pengumpulan data adalah proses sistematis untuk menghimpun informasi atau fakta yang relevan dengan tujuan penelitian atau analisis. Proses ini melibatkan metode dan teknik tertentu untuk mengakuisisi data yang diperlukan, termasuk survei, wawancara, observasi, pengukuran, atau pemanfaatan sumber data yang sudah ada. Tujuan pengumpulan data adalah untuk mendukung analisis, verifikasi hipotesis, atau memberikan dasar empiris bagi

keputusan dan penelitian [28]. Pada tahap ini, peneliti akan mengumpulkan data yang akan digunakan untuk pelatihan dan pengujian model klasifikasi malware. Data yang akan dikumpulkan adalah data malware yang telah dikategorikan ke dalam kategori-kategori tertentu. Dataset yang digunakan dalam penelitian ini menggunakan data public yang tersedia melalui halaman website Dataset Kaggle.

### 3.4 Pemrosesan Data

Pada tahap ini, data lalu lintas jaringan dari perangkat Android diproses untuk mendapatkan fitur-fitur yang dapat digunakan untuk membedakan antara lalu lintas normal dan lalu lintas malware. Fitur-fitur tersebut dapat berupa jenis protokol port yang digunakan, ukuran paket, dan sebagainya. Pemrosesan data pada tahap awal ini menjadi langkah kritis dalam konstruksi model klasifikasi malware menggunakan algoritma Support Vector Machine (SVM). Tujuan utamanya adalah untuk memastikan bahwa data yang digunakan memiliki kualitas yang baik dan sesuai sebelum diterapkan ke dalam model SVM. Oleh karena itu, dengan melakukan langkah pemrosesan data yang tepat, kita dapat secara signifikan mempengaruhi kualitas dan keandalan penemuan dan keputusan otomatis selanjutnya[29].

Proses pemrosesan data dapat dimulai dari pengambilan dataset yang tersedia dalam format csv. Untuk pengklasifikasian malware pada penelitian ini, dataset diambil dari website Kaggle yang berjudul “android_traffic.csv”. Setelah itu, informasi dasar mengenai dataset tersebut akan ditampilkan guna memulai penerimaan wawasan awal tentang karakteristik dan struktur data yang akan diolah. Selanjutnya akan dilakukan penghapusan kolom yang tidak relevan atau tidak diinginkan agar dataset tetap terfokus pada fitur-fitur yang benar-benar berkontribusi pada pemodelan SVM. Pada pemrosesan data ini juga akan dilakukan penanganan nilai yang hilang untuk memastikan kebersihan dan integritas data.

### 3.5 Pemisahan Data

Setelah data diproses, data tersebut kemudian dibagi menjadi dua set, yaitu set pelatihan dan set pengujian. Set pelatihan digunakan untuk melatih model SVM, sedangkan set pengujian digunakan untuk mengevaluasi kinerja model. Pemisahan data adalah desain studi yang banyak digunakan dalam pengaturan dimensi tinggi dengan membagi kumpulan

data menjadi set pelatihan dan set uji sebagai sarana untuk memperkirakan akurasi klasifikasi. Pengklasifikasi dikembangkan pada set pelatihan dan diterapkan pada setiap sampel dalam set uji [30].

Metode pemisahan yang digunakan untuk membagi data menjadi set pelatihan dan set pengujian pada penelitian ini adalah dengan menggunakan salah satu library pada aplikasi machine learning yang digunakan. Library ini bernama “sklearn.model_selection”, library ini menyediakan berbagai fungsi untuk membagi data menjadi set pelatihan dan pengujian. Salah satu fungsinya ada fungsi “train_test_split”, yang secara acak mempartisi data menjadi dua subset dengan proporsi yang ditentukan oleh parameter “test_size”. Seperti yang sudah dijelaskan sebelumnya, parameter “test_size” akan menentukan proporsi data yang akan dialokasikan ke set pengujian. Selain parameter ini, terdapat juga parameter yang bernama “random_state”. Parameter ini akan menentukan seed acak untuk proses pemisahan data.

Pemisahan data menjadi set pelatihan dan set pengujian tidak hanya terletak pada kemampuan model untuk mempelajari data pelatihan, tetapi juga pada kemampuannya untuk menggeneralisasi dan mengklasifikasikan data yang belum pernah dilihat sebelumnya. Dengan ukuran set pelatihan yang memadai, distribusi kelas yang serupa, dan metode pembagian data yang tepat, model SVM dapat meningkatkan akurasi, menghindari overfitting, dan memberikan evaluasi kinerja yang handal pada set pengujian [31].

### 3.6 Normalisasi Data

Pada tahap ini, data dalam set pelatihan dan set pengujian dinormalisasi agar nilainya berada dalam skala yang sama. Hal ini dilakukan agar model SVM dapat bekerja dengan lebih efektif. Proses normalisasi data ditandai dengan mengubah skala data sehingga semua fitur memiliki nilai yang mirip. Normalisasi ini penting untuk dilakukan karena dapat meningkatkan kinerja model SVM. SVM dapat sensitif terhadap skala data, sehingga jika data tidak ternormalisasi, maka model SVM dapat belajar dengan lebih baik pada fitur-fitur yang memiliki skala yang lebih besar [32]. Ada beberapa metode yang dapat digunakan untuk normalisasi data. Salah satu metode yang paling umum adalah normalisasi Min-Max. Metode ini mengubah nilai setiap fitur sehingga nilainya berada dalam rentang tertentu [25].

Dalam mengenali pentingnya normalisasi data, metode normalisasi Min-Max sering kali menjadi pilihan yang umum. Metode ini mengubah nilai setiap fitur sehingga nilainya terdapat dalam rentang tertentu, contohnya dari 0 hingga 1. Langkah-langkah konkret untuk menerapkan normalisasi Min-Max melibatkan perhitungan nilai minimum dan maksimum untuk setiap fitur, kemudian mengurangkan setiap nilai fitur dengan nilai minimumnya, dan membagi setiap nilai fitur dengan selisih nilai maksimum dan minimumnya.

$$Normalisasi\ Value = \frac{Real\ value - Min\ value}{Max\ value - Min\ value} \tag{3.3.1}$$

Keterangan:

| | | |
|---|---|---|
| Real value | : | Nilai asli dari variabel yang akan dinormalisasi |
| Min value | : | Nilai minimum dari variabel |
| Max value | : | Nilai maksimum dari variable |
| Normalisasi value | : | Nilai data yang telah dinormalisasi |

## 3.7 Model *Support Vector Machine (SVM)*

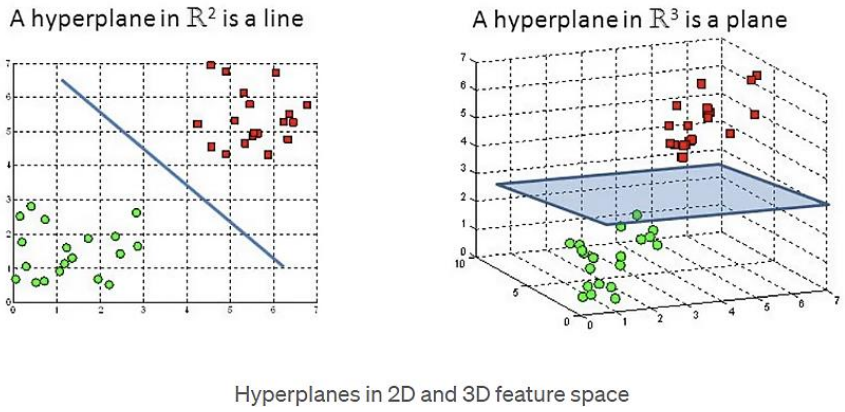
### 3.7.1 Penentuan Rentang Nilai Parameter untuk Penalaan Hiperparameter

Dalam membangun model Support Vector Machine (SVM), langkah pertama yang kritis adalah menentukan rentang nilai parameter yang akan diuji untuk penalaan hiperparameter. Pada tahap ini, model SVM dilatih menggunakan set pelatihan. Model SVM akan mempelajari hubungan antara fitur-fitur data dan kelasnya (normal atau malware). Model SVM yang digunakan dalam penelitian ini adalah Support Vector Machine (SVM) dengan kernel *Radial Basis Function (RBF).* Hiperparameter pada SVM melibatkan elemen-elemen seperti berikut ini:

*a. Hyperplane*

Untuk memahami logika SVM, pemahaman akan *Hyperplane* sangat ditekankan. Adapun demikian, Hyperplane dalam ruang Euclidean berdimensi n adalah himpunan bagian datar berdimensi n-1 dari ruang tersebut yang membagi ruang menjadi dua bagian yang tidak terhubung [33]. Untuk dua dimensi dapat dilihat bahwa garis pemisahnya adalah *hyperplane*. Demikian pula, untuk tiga dimensi, sebuah bidang dengan dua dimensi membagi ruang 3d menjadi dua bagian dan dengan demikian bertindak sebagai bidang

hiper. Jadi untuk ruang berdimensi n akan dimiliki bidang hiper berdimensi n-1 yang memisahkannya menjadi dua bagian.




Gambar 2. Hyperplanes di tampilan 2D dan 3D

Sumber: SupportVectorMachines(SVM)—AnOverview

b. Tuning Parameters

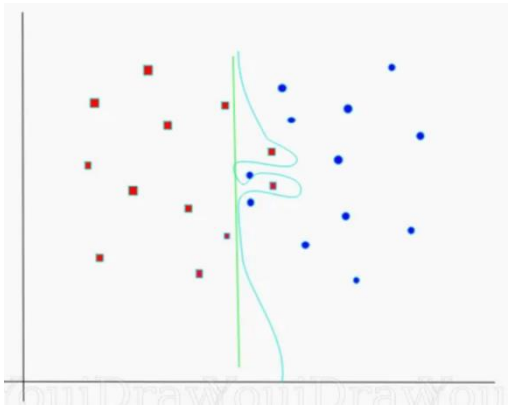
Parameter adalah argumen yang digunakan untuk membuat pengklasifikasian. Terdapat 2 jenis parameter yaitu:

1. Kernel

Parameter Kernel berfungsi untuk menentukan fungsi yang digunakan untuk mengubah data input menjadi bentuk yang lebih sesuai dengan pemisahan non-linear. Adapun demikian, terdapat beberapa jenis kernel yang salah satunya akan digunakan pada penelitian ini. Kernel tersebut adalah ‘Linear’ yang digunakan untuk data yang dapat dipisahkan secara linear seperti halnya dataset android_traffic yang akan diklasifikasikan.

2. C

Parameter C berfungsi untuk mengontrol *trade off* antara batas keputusan yang mulus dan mengklasifikasikan poin pelatihan dengan benar. Nilai c yang besar berarti model akan mendapatkan lebih banyak poin pelatihan dengan benar.



Gambar 3. Batas Keputusan dan Klasifikasi Poin Pelatihan Hyperplane

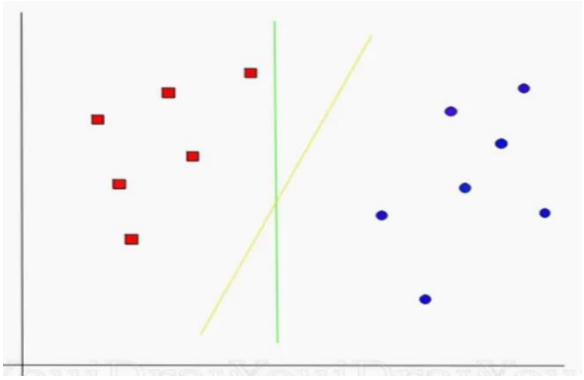
Sumber: SupportVectorMachines(SVM)—AnOverview

3. Gamma

Parameter Gamma berfungsi untuk menentukan seberapa jauh pengaruh satu contoh pelatihan. Jika nilainya rendah berarti setiap titik mempunyai jangkauan yang jauh dan sebaliknya nilai gamma yang tinggi berarti setiap titik mempunyai jangkauan yang dekat. Dengan kata lain, jika nilai gamma yang dihasilkan rendah menunjukkan bahwa sampel data yang jauh dari garis pemisah memiliki pengaruh yang signifikan dalam perhitungan untuk menentukan posisi garis pemisah. Sebaliknya, jika nilai Gamma tinggi, sampel data yang dekat dengan garis pemisah akan memiliki pengaruh yang lebih besar dalam proses perhitungan untuk menentukan garis pemisah yang optimal. Besarnya nilai Gamma yang dihasilkan memengaruhi seberapa ketat atau luas garis pemisah dalam klasifikasi data [34].

### 3.7.2 Pembuatan Model SVM dan Penggunaan Grid Search

Setelah rentang nilai parameter ditentukan, langkah selanjutnya adalah membangun model SVM dan menggunakan Grid Search untuk menemukan kombinasi parameter terbaik. Grid Search adalah metode pencarian sistematis yang mengevaluasi kombinasi berbagai nilai hiperparameter untuk menemukan kombinasi yang memberikan kinerja optimal. Pada tahap ini, kita dapat menggunakan modul GridSearchCV dari pustaka Scikit-learn untuk secara otomatis melakukan pencarian parameter terbaik.

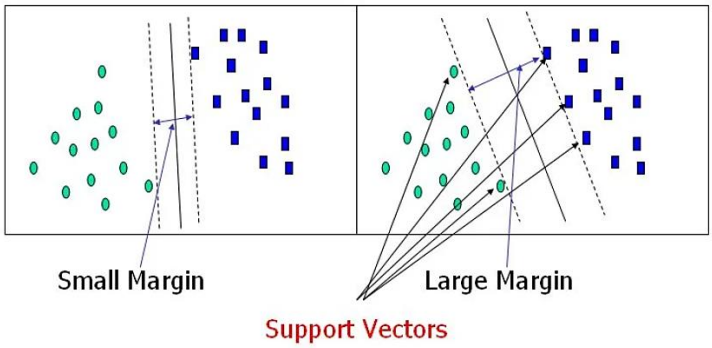


Gambar 4. Visual Pembagian data Dimensi tinggi

Sumber: SupportVectorMachines(SVM)—AnOverview

### 3.7.3 Inisialisasi dan Pelatihan Model SVM dengan Parameter Terbaik

Setelah Grid Search selesai, kita dapat mengakses parameter terbaik yang ditemukan dan menginisialisasi ulang model SVM menggunakan parameter tersebut. Langkah ini memastikan bahwa model yang dibangun mengoptimalkan kinerja berdasarkan parameter terbaik yang diidentifikasi.




Gambar 5. Optimal Hyperplane using the SVM Algorithm

Sumber: SupportVectorMachine—IntroductiontoMachineLearningAlgorithms

### 3.8 Evaluasi *(Results)*

Selanjutnya, kinerja model SVM dievaluasi menggunakan berbagai metrik, seperti akurasi, presisi, *recall*, dan f1-*score.*

### 3.8.1 Kriteria Evaluasi Model

Setelah berhasil melatih model SVM dengan parameter terbaik yang diidentifikasi melalui *Grid Search*, langkah selanjutnya adalah mengevaluasi kinerjanya. Evaluasi model adalah tahap kritis untuk memahami seberapa baik model dapat menggeneralisasi informasi dari data pelatihan ke data pengujian. Beberapa metrik evaluasi yang dapat digunakan mencakup akurasi, matriks kebingungan, dan laporan klasifikasi.

### 3.8.2 Pengukuran Akurasi

Pada tahap ini dilakukan pengukuran akurasi untuk mengukur sejauh mana model dapat memprediksi kelas dengan benar. Dengan kata lain, mengindikasikan sejauh mana suatu teknik klasifikasi dapat memberikan hasil yang tepat. Semakin tinggi nilai akurasi klasifikasi, semakin baik pula kinerja teknik klasifikasi tersebut secara keseluruhan. Dengan kata lain, tingkat akurasi mencerminkan seberapa tepat suatu metode dapat mengklasifikasikan data, dan semakin tinggi nilai akurasi, semakin baik pula performansi teknik klasifikasi yang digunakan [35]. Akurasi dihitung sebagai rasio prediksi yang benar (positif dan negatif) terhadap jumlah total data. Namun, untuk masalah klasifikasi yang tidak seimbang, akurasi saja mungkin tidak mencerminkan kinerja yang sebenarnya.

### 3.8.3 Evaluasi Klasifikasi

Laporan klasifikasi memberikan ringkasan komprehensif dari berbagai metrik evaluasi untuk setiap kelas. Ini mencakup presisi, recall, dan F1-score,

a. Presisi adalah metrik yang mengukur sejauh mana prediksi positif yang dibuat oleh model adalah benar.

   Rumus presisi:

$$\frac{True\ Positives}{True\ Positives + False\ Positives} \qquad (3.5.3.1)$$

   Keterangan:

   True Positives : Jumlah instance yang benar-benar termasuk dalam kelas yang diinginkan dan diprediksi dengan benar oleh model.

   False Positives : Jumlah instance yang sebenarnya bukan termasuk dalam kelas yang diinginkan, tetapi diprediksi sebagai anggota kelas tersebut oleh model.

Presisi memberikan informasi tentang seberapa baik model mengidentifikasi instance yang sebenarnya termasuk dalam kelas tersebut.

b. *Recall* mengukur sejauh mana model mampu mengidentifikasi semua *instance* positif yang sebenarnya.

Rumus recall

$$\frac{True\ Positives}{True\ Positives + False\ Negatives} \quad (3.5.3.2)$$

Keterangan:

True Positives : Jumlah instance yang benar-benar termasuk dalam kelas yang diinginkan dan diprediksi dengan benar oleh model.

False Negatives : Jumlah instance yang sebenarnya bukan termasuk dalam kelas yang diinginkan, tetapi diprediksi sebagai anggota kelas tersebut oleh model.

Recall memberikan informasi tentang seberapa baik model dapat menangkap semua instance positif yang sebenarnya.

c. F1-*score* adalah metrik gabungan yang menggabungkan presisi dan recall dalam satu nilai.

Rumus F1-s*core*

$$F1 - score = 2\ x\ \frac{Presisi\ x\ Recall}{Presisi + Recall} \quad (3.5.3.3)$$

Keterangan:

Precision : Metrik yang mengukur sejauh mana prediksi positif yang dibuat oleh model adalah benar.

Recall : Metrik yang mengukur sejauh mana model mampu mengidentifikasi semua instance positif yang sebenarnya.

F1-score memberikan keseluruhan ukuran kinerja model dengan mempertimbangkan kedua *false positives* dan *false negatives*. F1-score merupakan harmonic mean dari presisi dan *recall*. Karena menggunakan harmonic mean, F1-score cenderung

memberikan bobot lebih besar pada nilai yang lebih rendah dari kedua metrik tersebut. Oleh karena itu, F1-score baik digunakan ketika keseimbangan antara presisi dan recall diinginkan, terutama dalam situasi di mana ada ketidakseimbangan antara kelas positif dan negatif.

Nilai F1-*score* berkisar antara 0 dan 1, di mana nilai 1 menunjukkan kinerja model yang sempurna, dan nilai 0 menunjukkan kinerja yang sangat buruk. F1-score berguna untuk menyatukan informasi presisi dan recall menjadi satu nilai yang dapat memberikan gambaran holistik tentang kemampuan model dalam tugas klasifikasi.

### 3.8.4 Analisis Hasil dan Penyempurnaan Model

Setelah mendapatkan metrik evaluasi, langkah berikutnya adalah melakukan analisis mendalam terhadap hasilnya. Dengan mendetail dalam menganalisis hasil evaluasi, dapat diambil kesimpulan untuk memahami kekuatan dan kelemahan model SVM yang telah dikembangkan. Ini memberikan landasan untuk mengambil tindakan selanjutnya, baik itu peningkatan model atau penerapannya dalam skenario produksi. Evaluasi yang cermat adalah kunci untuk memastikan bahwa model yang dikembangkan dapat memberikan solusi yang andal dan efektif terhadap masalah klasifikasi malware.

## 3.9 Visualisasi Data

Pada tahap ini, data divisualisasi untuk membantu memahami karakteristik data dan kinerja model SVM. Visualisasi data dapat dilakukan dengan menggunakan berbagai teknik, seperti grafik, histogram, dan sebagainya.Visualisasi data dapat membantu peneliti untuk memahami pola yang ada dalam data dan untuk menganalisis kinerja model SVM.

### 3.9.1 Matriks Kebingungan *(Confusion Matrix)*

Ketika hasil pengukuran data dan pemodelan SVM telah dilakukan, maka selanjutnya akan diidentifikasi *confusion matrix*. Matriks kebingungan ini menyajikan informasi tentang seberapa baik model dapat memprediksi setiap kelas [13]. Matriks ini terdiri dari empat elemen: *True Positive (TP), True Negative (TN), False Positive (FP),* dan *False Negative (FN).* Dengan menggunakan matriks kebingungan, kita dapat menghitung metrik seperti presisi, recall, dan tingkat kesalahan tipe I dan II pada klasifikasi malware berdasarkan

implementasi algoritma SVM. Matriks konfusi merupakan alat evaluasi kinerja dalam pembelajaran mesin, yang mewakili keakuratan model klasifikasi. Matriks ini menampilkan jumlah positif benar, negatif benar, positif palsu, dan negatif palsu. Model matriks ini akan menganalisis kinerja model, mengidentifikasi kesalahan klasifikasi, dan meningkatkan akurasi prediksi [36].

Matriks Kebingungan dapat divisualisasikan untuk memberikan gambaran yang lebih intuitif tentang kinerja model. Visualisasi ini dapat membantu dalam memahami sejauh mana model mampu membedakan antara kelas-kelas tertentu. Misalnya, visualisasi ini dapat menggunakan pustaka seperti `seaborn` dan `matplotlib` untuk membuat heatmap matriks kebingungan:

```
import seaborn as sns
import matplotlib.pyplot as plt
# Membuat heatmap matriks kebingungan
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', cbar=False,
            xticklabels=['Prediksi Negatif', 'Prediksi Positif'],
            yticklabels=['Aktual Negatif', 'Aktual Positif'])
plt.xlabel('Prediksi')
plt.ylabel('Aktual')
plt.title('Matriks Kebingungan')
plt.show()
```

Visualisasi ini memberikan representasi grafis tentang seberapa baik model dapat mengklasifikasikan data ke dalam kategori positif dan negatif.

### 3.9.2 Kurva ROC *(Receiver Operating Characteristic)*

Kurva karakteristik operasi penerima (ROC) merupakan grafik yang memetakan sensitivitas uji terhadap spesifisitas 1 atau tingkat positif palsu (FPR) terhadap tingkat negatif palsu sebagai koordinat x. ROC digunakan sebagai metode yang efektif dalam mengevaluasi kinerja uji diagnostik. Kurva Karakteristik Operasi Penerima (ROC) adalah alat visualisasi yang digunakan untuk menggambarkan kinerja model pada berbagai tingkat ambang batas *(threshold)* untuk klasifikasi. ROC menggambarkan trade-off antara Tingkat Positif Benar *(True Positive Rate)* dan Tingkat Positif Salah *(False Positive Rate)*. Kurva ROC memberikan gambaran visual tentang seberapa baik model dapat membedakan antara

kelas positif dan negatif. Area di bawah kurva ROC (AUC) adalah metrik yang berguna semakin besar nilai AUC, semakin baik kinerja model [37].

```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

# Menghitung nilai prediksi probabilitas positif
y_prob = best_svm_model.predict_proba(X_test)[:, 1]

# Menghitung nilai False Positive Rate (FPR) dan True Positive Rate
(TPR) untuk berbagai ambang batas
fpr, tpr, thresholds = roc_curve(y_test, y_prob)

# Menghitung area di bawah kurva ROC (AUC)
roc_auc = auc(fpr, tpr)

# Membuat kurva ROC
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'AUC =
{roc_auc:.2f}')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate (FPR)')
plt.ylabel('True Positive Rate (TPR)')
plt.title('Kurva ROC')
plt.legend(loc="lower right")
plt.show()
```

# 4. Hasil Pengujian

Pada tahap ini, duraikan hasil pengujian terhadap klasifikasi data menggunakan algoritma SVM untuk mengklasifikasikan lalu lintas jaringan Android menjadi berbagai jenis (baik atau jahat).

## 4.1 Necessary Imports and Dataset Information

### 4.1.1 Import Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Langkah pertama untuk mengimpor library yang akan digunakan dalam analisis data dan pembuatan model.

### 4.1.2 Load Data from CSV File

```
# Read the CSV file with the correct delimiter
df = pd.read_csv('android_traffic.csv', sep=';')

# Display the first few rows of the DataFrame
df.head()
```

Selanjutnya, memuat data dengan menggunakan pandas, membaca data dari file CSV dengan delimiter ';' dan menyimpannya dalam DataFrame **df**.

## 4.2 Data Exploration

### 4.2.1. Display the first few rows of the DataFrame

Selanjutnya, menampilkan beberapa baris pertama (biasanya lima baris pertama secara default) dari DataFrame yang disimpan dalam variabel **df**. Dengan menggunakan metode **head()** menampilkan beberapa baris pertama dari DataFrame untuk memahami format data, tipe kolom, dan nilai-nilai awalnya.

Results:

Out[3]:

| | name | tcp_packets | dist_port_tcp | external_ips | vulume_bytes | udp_packets | tcp_urg_packet |
|---|---|---|---|---|---|---|---|
| 0 | AntiVirus | 36 | 6 | 3 | 3911 | 0 | 0 |
| 1 | AntiVirus | 117 | 0 | 9 | 23514 | 0 | 0 |
| 2 | AntiVirus | 196 | 0 | 6 | 24151 | 0 | 0 |
| 3 | AntiVirus | 6 | 0 | 1 | 889 | 0 | 0 |
| 4 | AntiVirus | 6 | 0 | 1 | 882 | 0 | 0 |

Out[3]:

| source_app_packets | remote_app_packets | source_app_bytes | remote_app_bytes | duracion | avg_local_ |
|---|---|---|---|---|---|
| 39 | 33 | 5100 | 4140 | NaN | |
| 128 | 107 | 26248 | 24358 | NaN | |
| 205 | 214 | 163887 | 24867 | NaN | |
| 7 | 6 | 819 | 975 | NaN | |
| 7 | 6 | 819 | 968 | NaN | |

Out[3]:

| ; | duracion | avg_local_pkt_rate | avg_remote_pkt_rate | source_app_packets.1 | dns_query_times | type |
|---|---|---|---|---|---|---|
| ) | NaN | NaN | NaN | 39 | 3 | benign |
| ; | NaN | NaN | NaN | 128 | 11 | benign |
| ' | NaN | NaN | NaN | 205 | 9 | benign |
| ; | NaN | NaN | NaN | 7 | 1 | benign |
| ; | NaN | NaN | NaN | 7 | 1 | benign |

### 4.2.2. Display the last few rows of the DataFrame

Berikutnya, menampilkan beberapa baris terakhir dari DataFrame yang disimpan dalam variabel **df**. Dengan menggunakan metode **tail()**, beberapa baris terakhir ditampilkan dari DataFrame untuk memeriksa nilai-nilai terakhir, dan ini juga membantu dalam mendapatkan gambaran umum tentang data yang dimiliki oleh DataFrame tersebut. Mirip dengan **head()**, metode **tail()** juga menampilkan sejumlah baris (biasanya lima) secara default.

```
df.tail()
```

Out[4]:

| | name | tcp_packets | dist_port_tcp | external_ips | vulume_bytes | udp_packets | tcp_urg_packet |
|---|---|---|---|---|---|---|---|
| **7840** | Zsone | 0 | 0 | 0 | 0 | 0 | 0 |
| **7841** | Zsone | 4 | 4 | 1 | 296 | 0 | 0 |
| **7842** | Zsone | 0 | 0 | 0 | 0 | 0 | 0 |
| **7843** | Zsone | 0 | 0 | 0 | 0 | 0 | 0 |
| **7844** | Zsone | 0 | 0 | 0 | 0 | 0 | 0 |

```
df.tail()
```

Out[4]:

| source_app_packets | remote_app_packets | source_app_bytes | remote_app_bytes | duracion | avg_local |
|---|---|---|---|---|---|
| 2 | 2 | 257 | 143 | NaN | |
| 5 | 1 | 86 | 382 | NaN | |
| 2 | 2 | 257 | 143 | NaN | |
| 2 | 2 | 257 | 143 | NaN | |
| 2 | 2 | 257 | 143 | NaN | |

```
df.tail()
```

Out[4]:

| duracion | avg_local_pkt_rate | avg_remote_pkt_rate | source_app_packets.1 | dns_query_times | type |
|---|---|---|---|---|---|
| NaN | NaN | NaN | 2 | 2 | malicious |
| NaN | NaN | NaN | 5 | 1 | malicious |
| NaN | NaN | NaN | 2 | 2 | malicious |
| NaN | NaN | NaN | 2 | 2 | malicious |
| NaN | NaN | NaN | 2 | 2 | malicious |

### 4.2.3. Display the total size of the DataFrame

Pada tahap ini, enampilkan ukuran total DataFrame dalam hal jumlah elemen yang mencakup seluruh sel atau elemen dalam DataFrame, termasuk nilai-nilai yang mungkin null.

```
df.size
```

Out[5]: 133365

### 4.2.4. Display the shape of the DataFrame

Kemudian, menampilkan dimensi DataFrame, yaitu jumlah baris dan kolom. Hasil eksekusi akan berupa tupel yang menyatakan (jumlah baris, jumlah kolom), seperti pada hasil eksekusi berikut:

```
df.shape
```

```
Out[6]: (7845, 17)
```

### 4.2.5. Display the count of non-null values in each column

Selanjutnya, menampilkan jumlah non-null *(non-missing)* nilai untuk setiap kolom dalam DataFrame. Hasil eksekusi akan berupa seri yang berisi jumlah non-null untuk setiap kolom.

```
df.count()
```

```
Out[7]: name                  7845
        tcp_packets           7845
        dist_port_tcp         7845
        external_ips          7845
        vulume_bytes          7845
        udp_packets           7845
        tcp_urg_packet        7845
        source_app_packets    7845
        remote_app_packets    7845
        source_app_bytes      7845
        remote_app_bytes      7845
        duracion                 0
        avg_local_pkt_rate       0
        avg_remote_pkt_rate      0
        source_app_packets.1  7845
        dns_query_times       7845
        type                  7845
        dtype: int64
```

### 4.2.6. Display the column names

Kemudian, menampilkan daftar nama kolom dalam DataFrame. Hasil eksekusi akan berupa objek yang berisi nama-nama kolom.

```
df.columns
```

```
Out[8]: Index(['name', 'tcp_packets', 'dist_port_tcp', 'external_ips', 'vulume_bytes',
               'udp_packets', 'tcp_urg_packet', 'source_app_packets',
               'remote_app_packets', 'source_app_bytes', 'remote_app_bytes',
               'duracion', 'avg_local_pkt_rate', 'avg_remote_pkt_rate',
               'source_app_packets.1', 'dns_query_times', 'type'],
              dtype='object')
```

### 4.2.7. Display the counts of each unique value in the 'type' column

```
df['type'].value_counts()
```

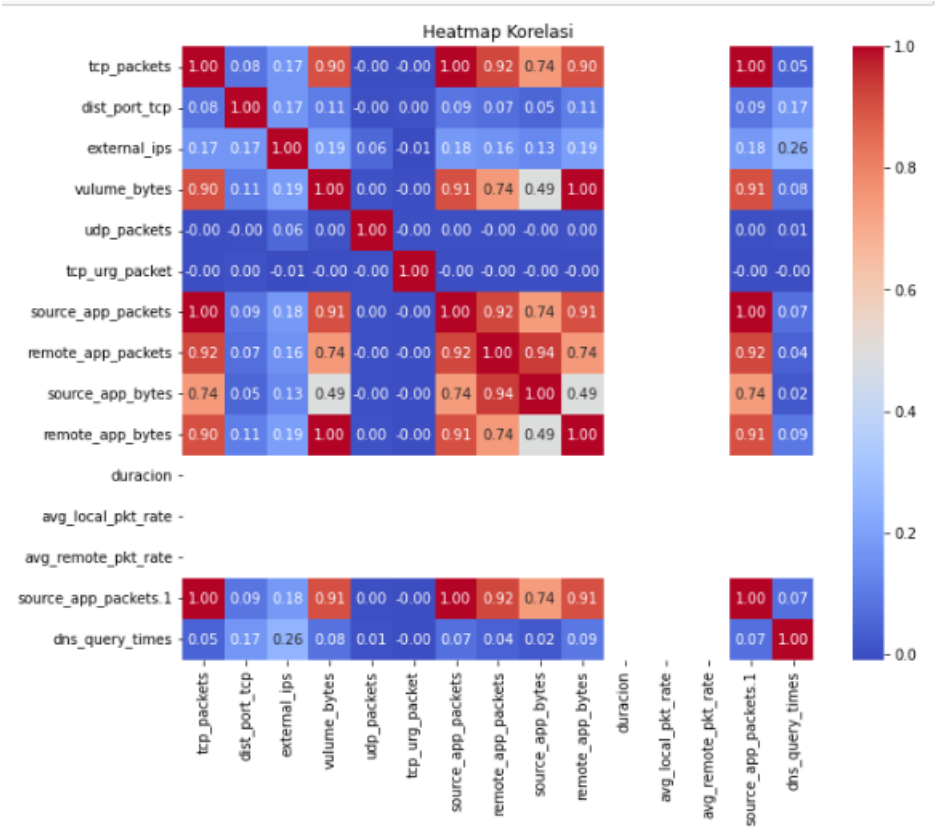
```
Out[9]: benign       4704
        malicious    3141
        Name: type, dtype: int64
```

### 4.3. Data Preprocessing

#### 4.3.1. Menentukan Korelasi Heatmap

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# matriks korelasi
numeric_columns = df.select_dtypes(include=['float64',
'int64']).columns
correlation_matrix = df[numeric_columns].corr()

# visualisasi matriks korelasi
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Heatmap Korelasi')
plt.show()
```

Langkah berikutnya adalah membuat visualisasi matriks korelasi dengan menggunakan heatmap. Visualisasi Heatmap akan menghasilkan plot warna yang menunjukkan kekuatan dan arah korelasi antar variabel. Visualisasi ini dilakukan dengan bantuan fungsi **heatmap** dari pustaka **seaborn**. Matriks korelasi dibuat untuk fitur numerik, dan visualisasi dilakukan menggunakan heatmap. Heatmap memberikan pandangan visual tentang sejauh mana fitur-fitur saling berkorelasi, membantu dalam pemilihan fitur untuk model.




Gambar 6. Heatmap Korelasi Dataset Malware

Selanjutnya, heatmap ini akan menampilkan nilai-nilai korelasi dengan warna, di mana warna yang lebih gelap menunjukkan korelasi yang lebih tinggi, dan warna yang lebih terang menunjukkan korelasi yang lebih rendah. Selain itu, nilai-nilai korelasi juga ditampilkan di dalam sel heatmap sebagai anotasi untuk memudahkan interpretasi.

### 4.3.2. Selection and Remove Unwanted Columns Base on Heatmap Korelasi, and Choose the Selected columns for SVM model label

Berdasarkan hasil heatmap korelasi, kita memilih kolom-kolom tertentu untuk digunakan dalam model SVM. Selanjutnya, dilakukan label encoding pada kolom 'type' untuk mengubah kelas menjadi format numerik. Data dibagi menjadi set pelatihan dan pengujian, dan dilakukan normalisasi menggunakan StandardScaler untuk memastikan skala yang seragam.

### 4.3.3. Selection Columns

Sebelumnya, kita perlu mendefinisikan variabel **selected_columns** yang berisi daftar nama kolom yang ingin dipilih dari dataframe (**df**). Kolom-kolom tersebut adalah 'tcp_packets', 'vulume_bytes', dan 'type'. Selanjutnya, kita menggunakan selected_columns untuk memilih subset dari dataframe (df). Dengan menggunakan df[selected_columns], kita membuat dataframe baru yang hanya berisi kolom-kolom yang telah dipilih. Langkah berikutnya adalah menghapus baris atau entri yang mengandung nilai-nilai yang hilang (NaN) dari dataframe yang baru dibuat. Ini dilakukan dengan menggunakan metode .dropna() pada dataframe data.

```
selected_columns = ['tcp_packets', 'vulume_bytes', 'type']
data = df[selected_columns].dropna()
```

```
data
```

Out[14]:

| | tcp_packets | vulume_bytes | type |
|---|---|---|---|
| 0 | 36 | 3911 | benign |
| 1 | 117 | 23514 | benign |
| 2 | 196 | 24151 | benign |
| 3 | 6 | 889 | benign |
| 4 | 6 | 882 | benign |
| ... | ... | ... | ... |
| 7840 | 0 | 0 | malicious |
| 7841 | 4 | 296 | malicious |
| 7842 | 0 | 0 | malicious |
| 7843 | 0 | 0 | malicious |
| 7844 | 0 | 0 | malicious |

7845 rows × 3 columns

Hasil pengujian atau hasil akhir disimpan dalam variabel **data**. Variabel ini berisi dataframe yang telah disusun ulang hanya dengan kolom-kolom yang telah dipilih, dan nilai-nilai NaN telah dihapus. Dengan demikian, kita memiliki dataset yang bersih dan siap untuk digunakan dalam analisis lebih lanjut.

### 4.3.4. Label Encoding

Selanjutnya, dilakukan pengkodean variabel kategori menjadi nilai numerik. Pertama, modul tersebut diimpor, kemudian objek **label_encoder** diinisialisasi. Langkah berikutnya melibatkan penggunaan metode **fit_transform** pada kolom 'type' dari DataFrame **data**,

yang secara bersamaan melibatkan proses pelatihan dan transformasi. Dalam proses pelatihan, model encoder disesuaikan dengan data kategori pada kolom 'type', sedangkan dalam transformasi, data kategori diubah menjadi nilai numerik berdasarkan pembelajaran yang telah dilakukan. Hasilnya adalah kolom 'type' pada DataFrame **data** sekarang berisi nilai-nilai numerik yang merepresentasikan kategori-kategori yang awalnya bersifat nominal.

```
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
data['type'] = label_encoder.fit_transform(data['type'])
```

### 4.3.5. Divide the Data as Train/Test Dataset

Selanjutnya, dilakukan pembagian dataset menjadi train dan test sets, serta melakukan penskalaan fitur menggunakan StandardScaler dari scikit-learn. Pada awalnya, modul **train_test_split** dari scikit-learn diimpor untuk memudahkan pembagian dataset menjadi dua bagian: train set dan test set. Variabel **features** didefinisikan untuk menentukan fitur-fitur yang akan digunakan dalam model, dan **X** dan **y** mewakili masing-masing fitur dan target variable. Selanjutnya, dataset dibagi menjadi train set dan test set menggunakan fungsi **train_test_split**. Parameter **test_size=0.2** menunjukkan bahwa 20% dari data akan dialokasikan sebagai test set, sementara 80% sisanya menjadi train set. **random_state=42** digunakan untuk memberikan reproduktibilitas pada pemisahan dataset. Setelah itu, modul **StandardScaler** dari scikit-learn digunakan untuk melakukan penskalaan fitur. Ini penting ketika kita memiliki fitur-fitur dengan rentang nilai yang berbeda, dan kita ingin menghindari pengaruh dominasi dari fitur-fitur tertentu pada hasil model. Data train diukur dan penskalanya dihitung menggunakan **fit_transform**, sementara data test hanya di-transform menggunakan skala yang sama yang telah diperoleh dari data train.

```
from sklearn.model_selection import train_test_split, GridSearchCV

# Features (X) and target variable (y)
features = ['tcp_packets', 'vulume_bytes']
X = data[features]
y = data['type']

# Split the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Hasilnya, kita memiliki empat variabel: **X_train** dan **X_test** sebagai fitur-fitur yang belum di-scala, serta **X_train_scaled** dan **X_test_scaled** sebagai fitur-fitur yang sudah di-scala. Proses ini mempersiapkan data untuk dijadikan input dalam proses pelatihan dan evaluasi model machine learning.

### 4.4. Model SVM with Scikit-Learn

```
from sklearn.svm import SVC

svm_model = SVC(kernel='linear', C=1.0)
svm_model.fit(X_train, y_train)
```

Pada tahap ini, kita akan melakukan pemodelan SVM pada klasifikasi data malware. Pada kasus ini, kita menggunakan Scikit-learn, jadi kita mengimpor kelas **SVC** dari modul **svm**. Selanjutnya, kita inisialisasi model SVM dengan menggunakan kelas **SVC**. Pada contoh di atas, kita menggunakan kernel linear dengan parameter **C=1.0**. Kernel linear biasanya digunakan untuk masalah klasifikasi dengan batas keputusan linear. Setelah inisialisasi model, kita melatihnya menggunakan data pelatihan. **X_train** adalah matriks fitur dari data pelatihan, dan **y_train** adalah label yang sesuai.

Results:

```
Out[19]: SVC(kernel='linear')
```

Setelah pelatihan selesai, model SVM siap untuk digunakan. Setelah melatih model, kita dapat melakukan prediksi pada data uji.

### 4.5. Evaluation (Results)

Model dievaluasi menggunakan data pengujian. Dilakukan prediksi pada data pengujian, dan metrik evaluasi seperti akurasi, matriks kebingungan, dan laporan klasifikasi ditampilkan. Pada tahapan awal, diimpor beberapa fungsi penting dari library scikit-learn, yaitu **accuracy_score**, **classification_report**, dan **confusion_matrix**. Selanjutnya, model SVM digunakan untuk membuat prediksi (**y_pred**) berdasarkan data uji (**X_test**).

```
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix

y_pred = svm_model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

print(f'Accuracy: {accuracy}')
print(f'Confusion Matrix:\n{conf_matrix}')
print(f'Classification Report:\n{class_report}')
```

```
Accuracy: 0.6896112173358827
Confusion Matrix:
[[829 109]
 [378 253]]
Classification Report:
              precision    recall  f1-score   support

           0       0.69      0.88      0.77       938
           1       0.70      0.40      0.51       631

    accuracy                           0.69      1569
   macro avg       0.69      0.64      0.64      1569
weighted avg       0.69      0.69      0.67      1569
```

Kemudian, akurasi model dihitung dengan membandingkan prediksi (**y_pred**) dengan label sebenarnya dari data uji (**y_test**). Hasilnya, akurasi sebesar 0.6896 atau 68.96%. Setelah itu, matriks kebingungan memberikan gambaran tentang kinerja model dalam memprediksi

kelas. Matriks tersebut memiliki empat sel: True Positive (TP), True Negative (TN), False Positive (FP), dan False Negative (FN). Berdasarkan hasil pengujian, terdapat 829 prediksi True Negative (TN), 253 prediksi True Positive (TP), 109 prediksi False Positive (FP), dan 378 prediksi False Negative (FN). Berikutnya, laporan klasifikasi memberikan informasi lebih lanjut tentang kinerja model untuk setiap kelas. Laporan tersebut mencakup precision, recall, dan f1-score untuk setiap kelas, serta rata-rata berponderasi (weighted avg) dan rata-rata tanpa bobot (macro avg). Laporan klasifikasi memberikan gambaran lebih rinci tentang kualitas prediksi model untuk setiap kelas, mencakup precision, recall, dan f1-score. Pada contoh di atas, kelas 0 memiliki recall yang lebih tinggi (0.88) dibandingkan dengan kelas 1 (0.40), yang menunjukkan bahwa model cenderung lebih baik dalam mengidentifikasi instans dari kelas 0.

## 4.6. Data Visualization

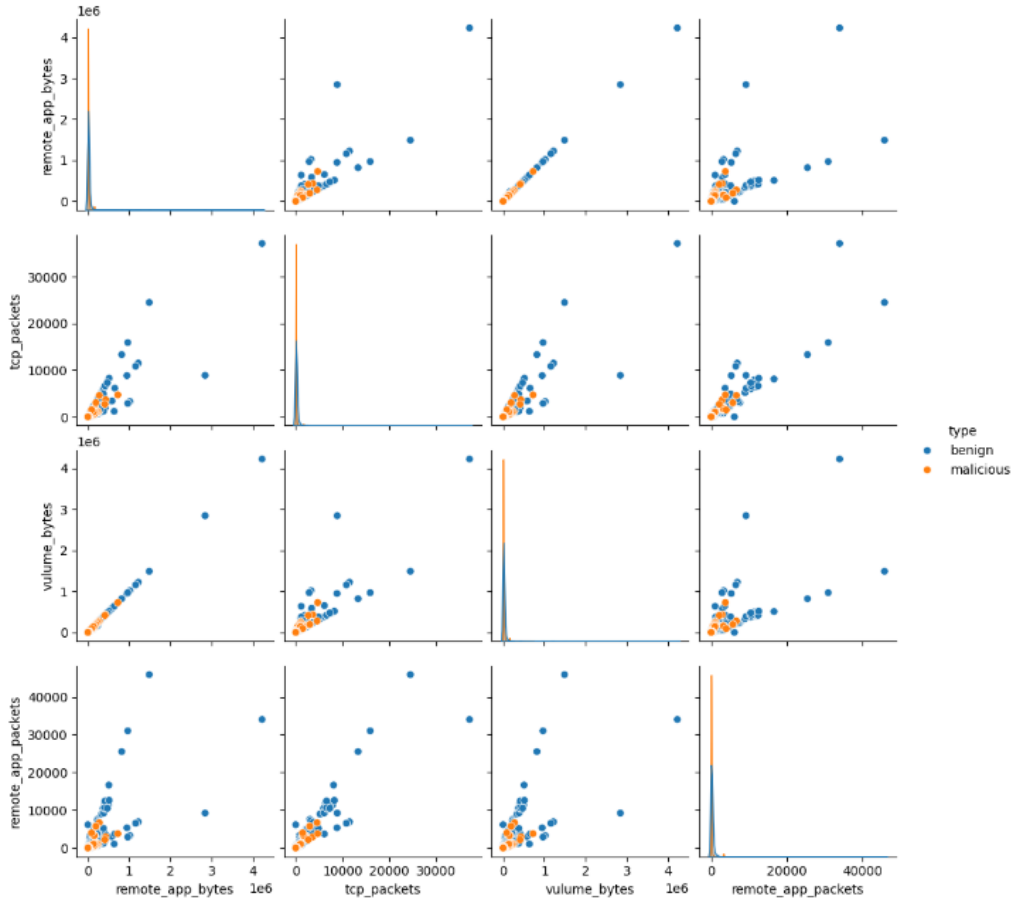
Visualisasi data dilakukan dengan menggunakan pairplot, yang memberikan pemahaman visual tentang hubungan antar fitur dalam dataset. Namun, perlu dicatat bahwa visualisasi ini dilakukan sebelum dan sesudah penerapan model SVM.

### 4.6.1. Visualisasi Data Sebelum Implementasi SVM

#### 4.6.1.1. Visualisasi Data Pair Plot

```
# NOT SVM IMPLEMENTATION MODEL
import seaborn as sns
import matplotlib.pyplot as plt

# Memvisualisasikan Data
```

Selanjutnya, dilakukan visualisasi data menggunakan pairplot dari pustaka seaborn dan matplotlib. Setelah itu, kita menggunakan fungsi **pairplot** dari seaborn untuk membuat scatterplot matrix dari beberapa variabel yang dipilih. Dalam hal ini, variabel yang dipilih adalah 'remote_app_bytes', 'tcp_packets', 'vulume_bytes', dan 'remote_app_packets'. Data yang digunakan untuk visualisasi adalah DataFrame **df**. Harga variabel 'type' digunakan sebagai argumen **hue** untuk membedakan data berdasarkan kategori 'type'.




Gambar 7. Scatter plot Dataset Klasifikasi Malware

Hasil visualisasi ini adalah matriks scatterplot di mana setiap sel memuat scatterplot antara dua variabel yang dipilih. Warna plot berbeda-beda sesuai dengan kategori 'type'. Scatterplot memungkinkan kita untuk melihat hubungan dan distribusi antar variabel. Kemudian, kita menggunakan **plt.show()** untuk menampilkan plot yang sudah dibuat.Penggunaan pairplot sangat berguna dalam mengeksplorasi hubungan antar variabel, terutama ketika kita ingin melihat bagaimana variabel-variabel tersebut berinteraksi dan apakah ada pola-pola yang dapat diidentifikasi. Selain itu, dengan menambahkan hue berdasarkan kategori 'type', kita dapat melihat bagaimana pola-pola tersebut berkaitan dengan kelas atau jenis data yang sedang diamati.

### 4.6.2. Visualisasi Data Sesudah Implementasi SVM

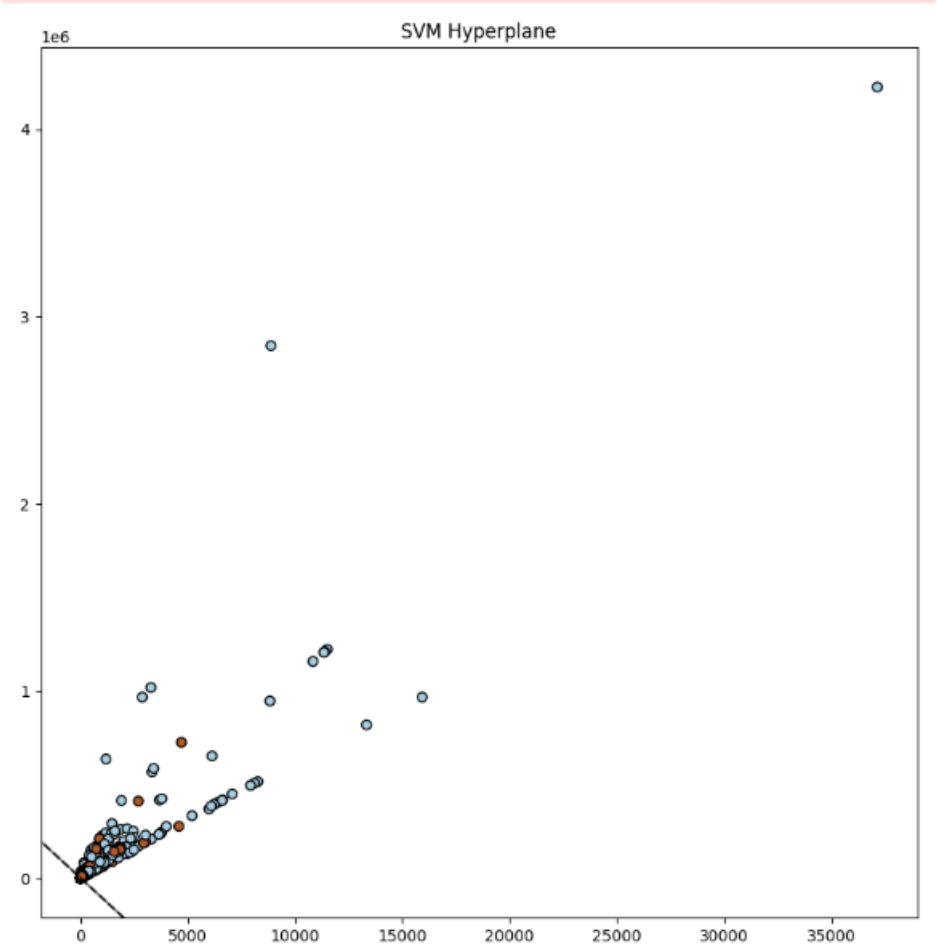
#### 4.6.2.1. Visualisasi Data Plot Hyperlane 2D

Selanjutnya, dilakukan visualisasi data plot Hyperplane di ruang dua dimensi dengan mengimpor dua library utama, yaitu NumPy untuk operasi numerik dan Matplotlib untuk visualisasi data.

```
import numpy as np
import matplotlib.pyplot as plt

def plot_hyperplane(X, y, model, title='SVM Hyperplane'):
    plt.figure(figsize=(10, 10))  # Atur ukuran gambar di sini
    plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Paired,
edgecolors='k')

    # plot the decision function
    ax = plt.gca()
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()

    # create grid to evaluate model
    xx, yy = np.meshgrid(np.linspace(xlim[0], xlim[1], 50),
                         np.linspace(ylim[0], ylim[1], 50))
    Z = model.decision_function(np.c_[xx.ravel(), yy.ravel()])

    # plot decision boundary and margins
    Z = Z.reshape(xx.shape)

    plt.contour(xx, yy, Z, colors='k', levels=[-1, 0, 1], alpha=0.5,
                linestyles=['--', '-', '--'])

    plt.title(title)
    plt.show()

# visualisasikan hyperplane
plot_hyperplane(X_train.values, y_train.values, svm_model)
```

Pada potongan kode di atas, terdapat implementasi visualisasi hyperplane dari model Support Vector Machine (SVM) yang telah dilatih. Pertama, dilakukan import library NumPy dan Matplotlib. Selanjutnya, fungsi **plot_hyperplane** digunakan untuk memvisualisasikan data input, dengan scatter plot yang menunjukkan sebaran data berdasarkan kelasnya. Selanjutnya, dilakukan perhitungan decision function model SVM di seluruh ruang fitur dan pembuatan grid untuk evaluasi model. Decision boundary dan

margins ditentukan melalui kontur plot dengan garis solid sebagai decision boundary dan garis putus-putus sebagai margins. Hasil visualisasi ini memberikan pemahaman yang jelas tentang bagaimana model SVM memisahkan kelas-kelas dalam ruang fitur dua dimensi. Pada bagian akhir, fungsi **plot_hyperplane** dipanggil dengan melewatkan data latih, label latih, dan model SVM yang telah dilatih, menghasilkan gambar hyperplane yang dapat digunakan untuk menganalisis performa model terhadap data latih tersebut.




Gambar 8. Hyperplane SVM in 2D

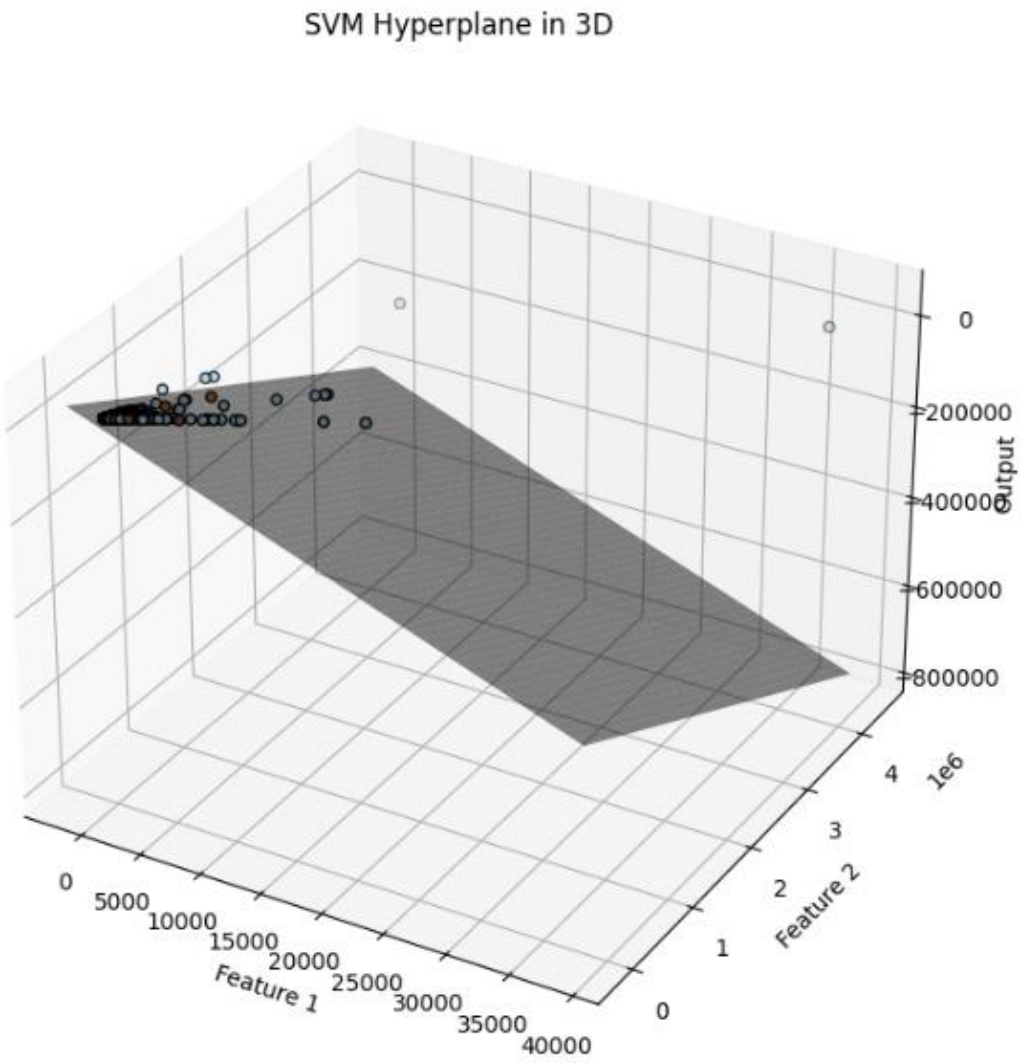
Visualisasi tersebut merepresentasikan bagaimana model Support Vector Machine (SVM) memisahkan kelas-kelas dalam ruang fitur dua dimensi. Hasilnya dapat dilihat dalam gambar hyperplane yang dihasilkan oleh fungsi **plot_hyperplane.**

#### 4.6.2.2. Visualisasi Data Hyperplane 3D

Selanjutnya, memvisualisasikan hyperplane dari model Support Vector Machine (SVM) dalam tiga dimensi. Visualisasi dimulai dengan mengimpor library yang diperlukan, yaitu

NumPy untuk operasi matematika, Matplotlib untuk visualisasi data, dan Axes3D dari mpl_toolkits.mplot3d untuk menggambar plot 3D. Selanjutnya, terdapat definisi fungsi **plot_hyperplane_3d** yang digunakan untuk membuat plot data dan menampilkan hyperplane SVM dalam tiga dimensi. Fungsi ini membutuhkan data (**X** dan **y**), model SVM (**model**), dan beberapa parameter opsional seperti judul (**title**). Di sisi lain, Fungsi **plot_hyperplane_3d** dijalankan dengan memberikan data latih (**X_train** dan **y_train**) serta model SVM (**svm_model**). Fungsi ini menciptakan sebuah plot 3D dengan melakukan scatter plot data, membuat meshgrid untuk fungsi keputusan SVM, dan menampilkan hyperplane sebagai permukaan 3D.

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Fungsi untuk membuat plot dan hyperplane 3D
def plot_hyperplane_3d(X, y, model, title='SVM Hyperplane in 3D'):
    fig = plt.figure(figsize=(10, 8))
    ax = fig.add_subplot(111, projection='3d')

    # Scatter plot data
    ax.scatter(X[:, 0], X[:, 1], y, c=y, cmap=plt.cm.Paired,
edgecolors='k')

    # Create meshgrid for decision function
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()
    xx, yy = np.meshgrid(np.linspace(xlim[0], xlim[1], 50),
                         np.linspace(ylim[0], ylim[1], 50))
    Z = model.decision_function(np.c_[xx.ravel(), yy.ravel()])

    # Plot decision surface in 3D
    Z = Z.reshape(xx.shape)
    ax.plot_surface(xx, yy, Z, color='k', alpha=0.5)

    # Set labels
    ax.set_xlabel('Feature 1')
    ax.set_ylabel('Feature 2')
    ax.set_zlabel('Output')

    plt.title(title)
    plt.show()

# Memvisualisasikan hyperplane dalam 3D
plot_hyperplane_3d(X_train.values, y_train.values, svm_model)
```



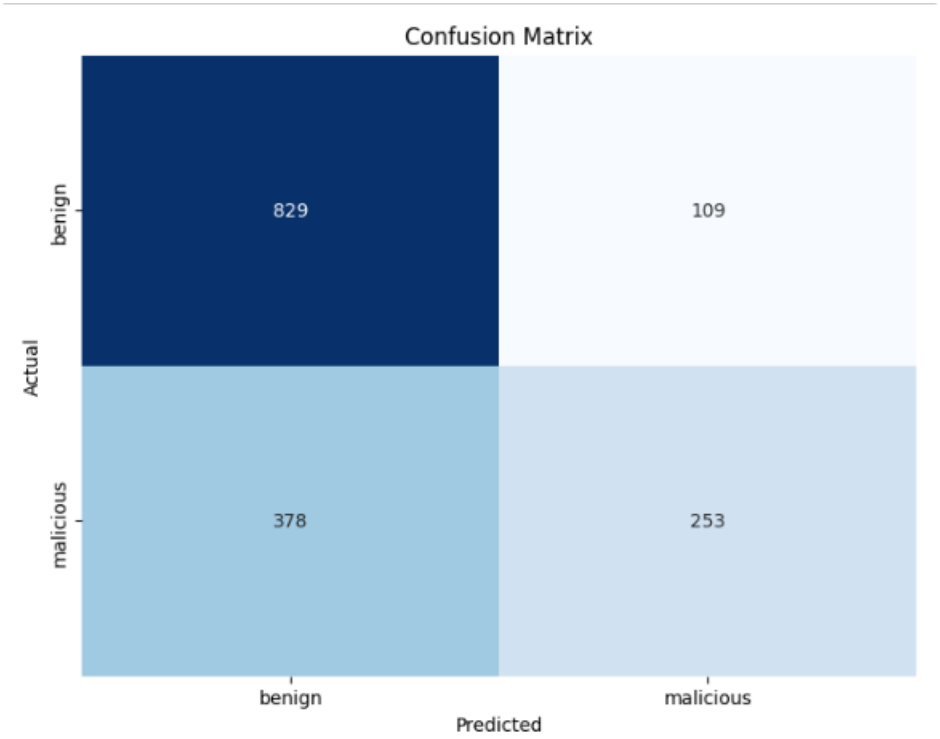

Gambar 9. SVM Hyperplane in 3D

Pengujian tersebut menghasilkan visualisasi data latih dan hyperplane SVM dalam tiga dimensi. Pada plot, titik-titik merepresentasikan data latih dengan warna yang sesuai dengan kelasnya. Hyperplane SVM ditampilkan sebagai permukaan yang memisahkan dua kelas. Permukaan ini dihasilkan berdasarkan fungsi keputusan SVM yang dihitung pada grid yang dibuat dari nilai-nilai fitur dalam dua dimensi. Selain itu, sumbu x dan y mewakili fitur-fitur, sementara sumbu z merepresentasikan keluaran dari model SVM. Hyperplane ini berfungsi sebagai batas keputusan yang memisahkan dua kelas. Plot tersebut memberikan pemahaman visual tentang bagaimana SVM membuat keputusan pada data yang diberikan. Pemanggilan fungsi plt.show() menampilkan plot tersebut, dan pengguna dapat melihat visualisasi hasilnya.

#### 4.6.2.3. Visualisasi Data Confusion Matrix

Selanjutnya, dilakukan implementasi untuk memvisualisasikan matriks kebingungan (confusion matrix) dari model klasifikasi pada data pengujian

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

# Compute confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Create a DataFrame from the confusion matrix
cm_df = pd.DataFrame(cm, index=['benign', 'malicious'],
columns=['benign', 'malicious'])

# Plot confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(cm_df, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```




Gambar 10. Confusion Matrik Klasifikasi Dataset Malware Android Traffic

Selanjutnya, mengevaluasi performa model klasifikasi pada data pengujian dengan memvisualisasikan matriks kebingungan. Pertama, beberapa pustaka seperti **matplotlib.pyplot**, **seaborn**, dan **confusion_matrix** dari **sklearn.metrics** diimpor untuk mendukung visualisasi dan perhitungan matriks kebingungan. Langkah berikutnya melibatkan perhitungan matriks kebingungan menggunakan label sebenarnya (**y_test**) dan hasil prediksi model (**y_pred**). Berikutnya, matriks kebingungan diorganisir ke dalam bentuk DataFrame menggunakan pustaka **pandas**. Ini memungkinkan hasil pengujian dapat lebih mudah dipahami dan divisualisasikan. Visualisasi dilakukan melalui sebuah heatmap, yang dihasilkan oleh fungsi **sns.heatmap**. Pada heatmap, warna dan angka di setiap sel mencerminkan seberapa baik model dapat memprediksi kelas sebenarnya dari data pengujian.

Sebagai tambahan, penyesuaian plot, seperti pengaturan ukuran dan penambahan label pada sumbu x dan y, diterapkan untuk meningkatkan keterbacaan. Hasil akhir berupa heatmap dari matriks kebingungan ditampilkan dengan menggunakan **plt.show()**.Melalui visualisasi ini, kita dapat mengevaluasi kinerja model dalam mengklasifikasikan data pengujian menjadi kategori "benign" atau "malicious". Intensitas warna pada heatmap memberikan

pandangan langsung tentang sejauh mana model mampu memprediksi kelas dengan benar, sementara angka di setiap sel memberikan informasi kuantitatif tentang jumlah sampel yang termasuk dalam kategori tertentu. Evaluasi ini menjadi penting dalam menilai kehandalan model klasifikasi dalam konteks aplikasinya.

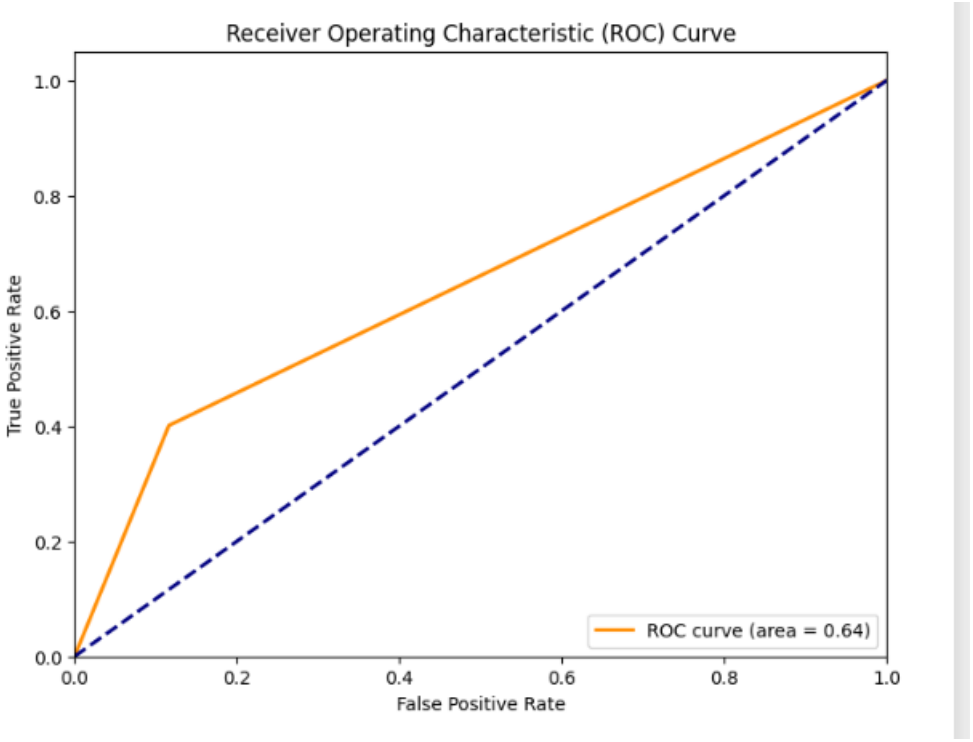
#### 4.6.2.4. Visualisasi Kurva ROC

Untuk mengukur kinerja model klasifikasi menggunakan Receiver Operating Characteristic (ROC) curve.

```
from sklearn.metrics import roc_curve, auc
from sklearn.preprocessing import LabelBinarizer
import matplotlib.pyplot as plt

# Convert labels to binary format
lb = LabelBinarizer()
y_test_bin = lb.fit_transform(y_test)
y_pred_bin = lb.transform(y_pred)

# Compute ROC curve and ROC area
fpr, tpr, _ = roc_curve(y_test_bin, y_pred_bin)
roc_auc = auc(fpr, tpr)

# Plot ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area =
{:.2f})'.format(roc_auc))
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.show()
```

Pertama, potongan kode diatas mengimpor beberapa modul yang diperlukan dari pustaka scikit-learn, yaitu **roc_curve** dan **auc** dari **sklearn.metrics**, serta **LabelBinarizer** dari **sklearn.preprocessing**. Modul matplotlib juga diimpor untuk keperluan visualisasi.

Langkah pertama setelah impor modul adalah mengonversi label (y_test dan y_pred) ke dalam format biner menggunakan **LabelBinarizer**. **LabelBinarizer** digunakan untuk mengubah label kelas menjadi format biner, yang diperlukan untuk menghitung ROC curve. Setelah label dikonversi, dilakukan perhitungan ROC curve dan Area Under the Curve (AUC) menggunakan fungsi **roc_curve** dan **auc** dari scikit-learn. Fungsi ini mengembalikan nilai False Positive Rate (FPR), True Positive Rate (TPR), dan AUC.




Gambar 11. Kurva ROC Klasifikasi Data Android Traffic Malware

Hasil perhitungan ROC curve dan AUC digunakan untuk membuat plot. Sebuah plot dengan sumbu x sebagai False Positive Rate dan sumbu y sebagai True Positive Rate dibuat menggunakan modul matplotlib. Kurva ROC aktual, bersama dengan garis putus-putus yang merepresentasikan kinerja model acak, ditampilkan. Selain itu, area di bawah kurva ROC (AUC) juga dicetak sebagai label di sudut kanan bawah plot. Plot ROC ini memberikan pandangan visual tentang kinerja model klasifikasi. Idealnya, kurva ROC mendekati sudut kiri atas, dan nilai AUC mendekati 1, menunjukkan kinerja yang sangat baik. Garis putus-putus menunjukkan kinerja model acak, dan semakin jauh kurva ROC dari garis ini, semakin baik model tersebut. Selanjutnya, plot ditampilkan menggunakan

plt.show(). Seluruh proses ini memberikan pemahaman yang lebih baik tentang sejauh mana model dapat membedakan antara kelas positif dan negatif, serta seberapa baik model tersebut dibandingkan dengan model acak.

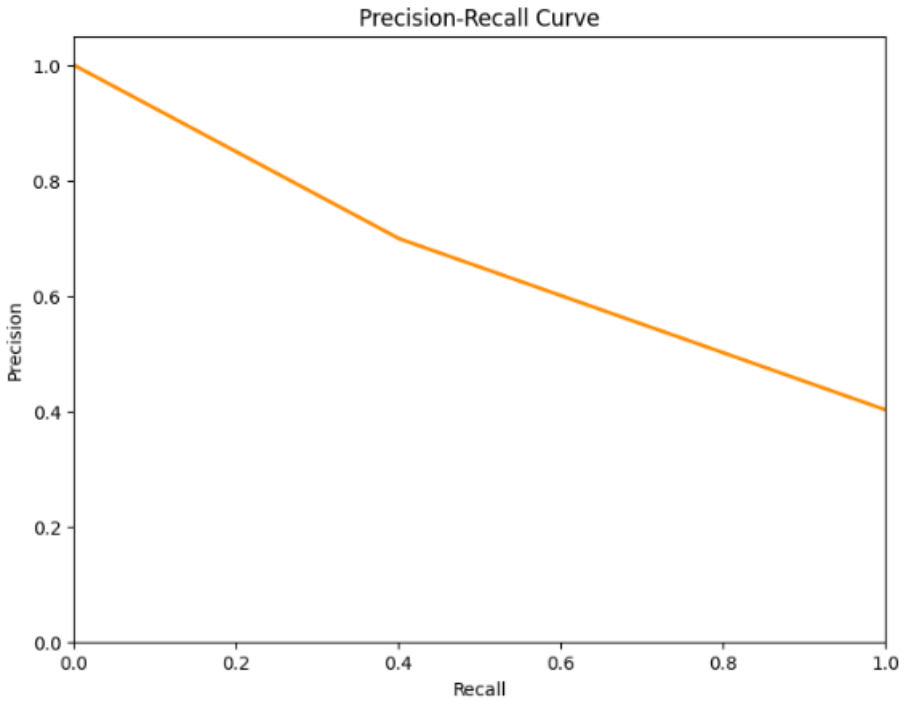
#### 4.6.2.5. Precision-Recall Curve

```
from sklearn.metrics import precision_recall_curve

# Compute precision-recall curve
precision, recall, _ = precision_recall_curve(y_test_bin.ravel(),
y_pred_bin.ravel())

# Plot Precision-Recall curve
plt.figure(figsize=(8, 6))
plt.plot(recall, precision, color='darkorange', lw=2)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve')
plt.show()
```

Selanjutnya, menggunakan pustaka **scikit-learn** untuk menghitung dan memplot kurva presisi-recall (Precision-Recall curve) dalam konteks evaluasi kinerja model klasifikasi. Mari kita bahas langkah-langkah dan hasil pengujian tersebut. Langkah pertama adalah mengimpor fungsi **precision_recall_curve** dari pustaka **scikit-learn**. Fungsi ini digunakan untuk menghitung nilai presisi, recall, dan ambang batas yang diperlukan untuk menggambar kurva presisi-recall. Ambang batas tersebut kemudian digunakan dalam prediksi model untuk menghasilkan nilai prediksi biner, y_pred_bin. Setelah itu, nilai presisi, recall, dan ambang batas dihitung menggunakan fungsi **precision_recall_curve**, dengan memberikan parameter y_test_bin (kelas target yang sebenarnya) dan y_pred_bin (prediksi model). Hasil dari fungsi ini disimpan dalam variabel precision, recall, dan _ (ambang batas, tetapi tidak digunakan dalam kode ini). Langkah selanjutnya adalah memplot kurva presisi-recall menggunakan pustaka matplotlib.

Fungsi **plot** digunakan untuk menggambar kurva dengan warna 'darkorange' dan lebar garis (lw) sebesar 2. Sumbu x dan y diberi label 'Recall' dan 'Precision', masing-masing. Batas

sumbu x dan y diatur dari 0 hingga 1.05 untuk menyesuaikan plot. Judul plot diberi label 'Precision-Recall Curve'. Terakhir, fungsi **show** digunakan untuk menampilkan plot.

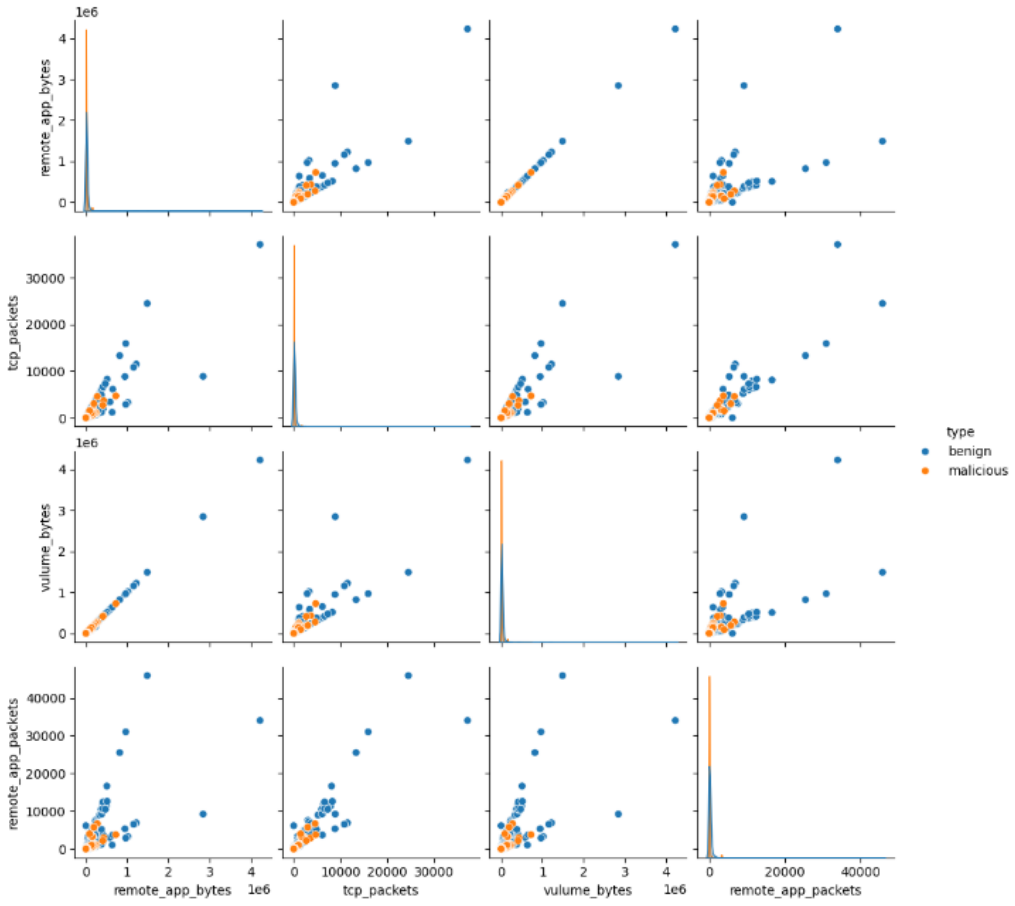



Gambar 12. Grafik 3D Grid Search Kombinasi Parameter C dan Gamma Pada Model SVM Untuk Mencari Nilai Optimal.

Hasilnya adalah kurva presisi-recall yang menggambarkan hubungan antara presisi dan recall model pada berbagai ambang batas. Biasanya, Anda akan melihat bahwa dengan menaikkan ambang batas prediksi, presisi meningkat tetapi recall menurun, dan sebaliknya. Kurva ini memberikan gambaran komprehensif tentang kinerja model klasifikasi dalam memprediksi kelas positif. Semakin tinggi area di bawah kurva (AUC-PR), semakin baik model dapat mempertahankan tingkat presisi yang tinggi dengan tingkat recall yang sesuai.

# 5. Analisis

## 5.1 Analisis Data Sebelum Implementasi SVM

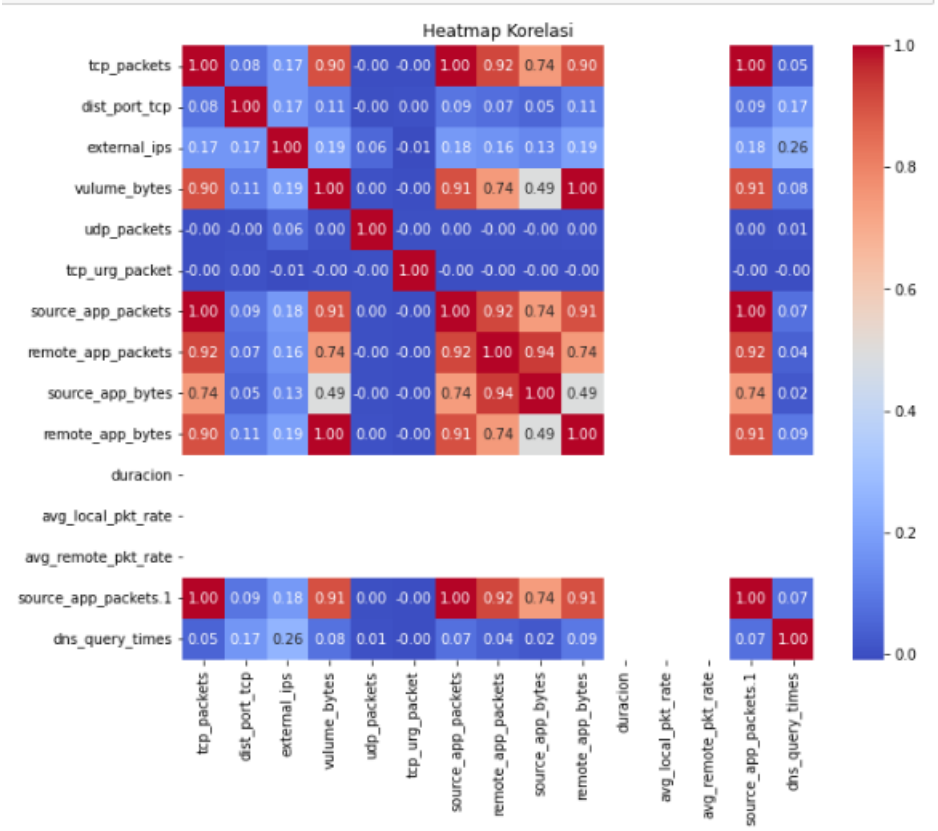



Gambar 13. Scatter Plot Visualisasi Dataset Klasifikasi Malware

Pada tahapan awal pemrosesan data, dataset malware yang dimiliki terdiri atas berbagai macam fitur-fitur yang dijelaskan dalam kolom-kolom pada tabel. Diantara data-data tersebut terdapat beberapa fitur yang secara analitis dapat lebih akurat menampilkan visualisasi data untuk menampilkan class benign (0), dan malicious(1). Fitur-fitur tersebut adalah remote_app_bytes, tcp_packets, vulume_bytes, dan remote_app_packets. Gambar diatas adalah hasil visualisasi data dari relasi keempat fitur tersebut dimana data-data yang ditampilkan masih belum mengalami pengklasifikasian menggunakan model SVM. Dapat dilihat secara visual kalau data yang termasuk dalam tipe malicious berada pada rentang yang mendekati angka 0, sedangkan data yang termasuk dalam tipe benign lebih mengarah ke angka-angka yang lebih besar.

Namun, secara visual memang fitur-fitur tersebut lebih menunjukkan perbedaan antara masing-masing kelas. Tapi tidak menutup kemungkinan, bahwa bukan fitur-fitur tersebutlah yang akan menjadi korelasi dua fitur terbaik untuk menciptakan hasil model SVM yang optimal. Dengan demikian, digunakanlah Heatmap korelasi. Heatmap korelasi adalah representasi visual dari matriks korelasi menggunakan warna. Matriks korelasi mengukur sejauh mana dua variabel berkaitan satu sama lain. Nilai korelasi ini berkisar antara -1 hingga 1. Dimana 1 menunjukkan korelasi positif sempurna (ketika satu variabel naik, yang lain juga naik secara proporsional), -1 menunjukkan korelasi negatif sempurna (ketika satu variabel naik, yang lain turun secara proporsional), dan -1 menunjukkan korelasi negatif sempurna (ketika satu variabel naik, yang lain turun secara proporsional).

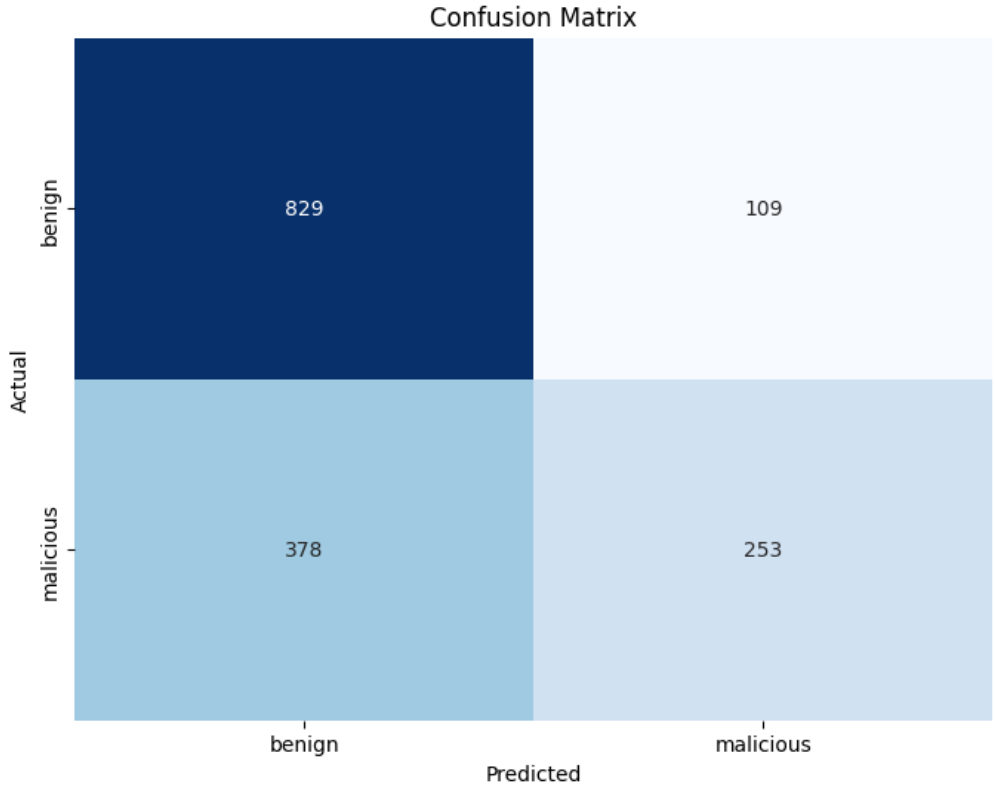
### 5.1.1 Visualisasi Data Heatmap




Gambar 14. Heatmap Korelasi Malware

Pada Heatmap Korelasi yang telah dilakukan terhadap fitur-fitur pada antroid_traffic dataset, terdapat beberapa fitur yang memiliki korelasi yang kuat. Salah satu korelasi yang berdampak besar pada penentuan label(target) model SVM adalah tcp_packets dan vulume_bytes dengan korelasi 0.90. Walaupun masih terdapat korelasi yang lebih tinggi dari korelasi ini, pengambilan keputusan fitur harus ikut serta memperhatikan hubungan korelasi tersebut dalam penentuan  klasifikasi kelas benign dan malicious. Sehingga tidak menutup kemungkinan bahwa fitur lainnya pun dapat digunakan dalam visualisasi model SVM.

### 5.1.2 Analisis Hasil Visualisasi Data Confusion Matrix

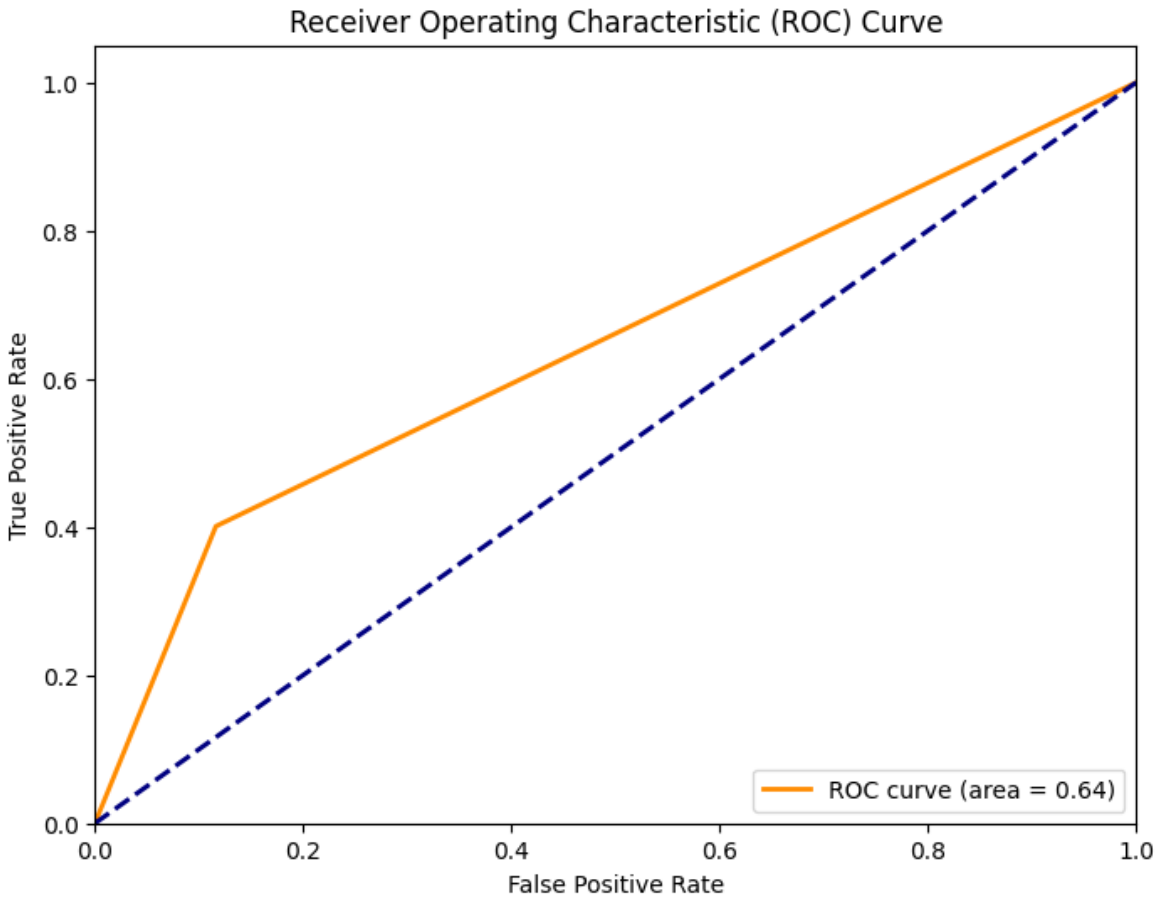



Gambar 15. Visualisasi Data Confusion Matrix Klasifikasi Malware

Berdasarkan hasil pengujian yang telah dilakukan untuk mengklasifikasikan malware menggunakan model SVM (Support Vector Machine), keakuratan penggunaan model ini dalam mengklasifikasikan malware menjadi dua kelas yaitu benign dan malicious adalah sebesar 0.6896112173358827 atau sekitar 68,96% yang dimana hal tersebut sudah cukup baik untuk memprediksi klasifikasi malware dengan optimal. Dapat diperhatikan pada

gambar Confusion Matrix diatas, Terdapat visualisasi Matriks Konfusi yang berguna untuk mengukur kinerja model klasifikasi dengan membandingkan prediksi model dengan nilai sebenarnya dari dataset yang digunakan. True Negative (TN) sejumlah 829 mengindikasikan bahwa model dengan benar mengklasifikasikan 829 sampel sebagai negatif. Sebaliknya, False Positive (FP) sebanyak 109 menunjukkan bahwa ada 109 sampel yang seharusnya negatif namun keliru diprediksi sebagai positif. False Negative (FN) sebanyak 378 menandakan bahwa 378 sampel positif salah diklasifikasikan sebagai negatif. Di sisi lain, True Positive (TP) sebanyak 253 menyiratkan bahwa model berhasil mengklasifikasikan 253 sampel sebagai positif dengan benar.

### 5.1.3 Analisis Visualisasi Data Receiver Operating Characteristic (ROC) Kurve

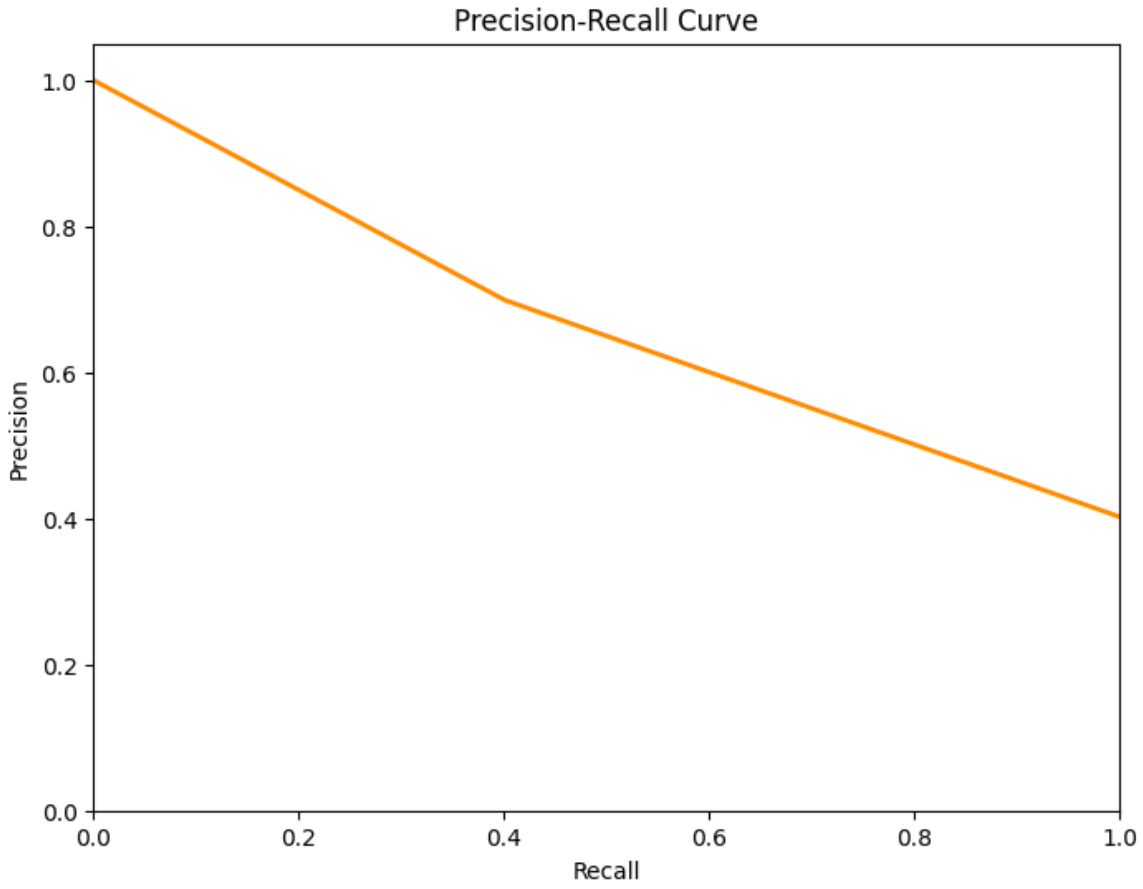



Gambar 16. Kurva ROC Klasifikasi Dataset Android Malware

Salah satu cara untuk mengetahui dan memahami lebih baik kinerja model klasifikasi dapat melalui ROC (Receiver Operating Characteristic). ROC Curve diatas memvisualisasikan

hubungan antara tingkat True Positive Rate dan tingkat False Positive Rate. Kurva ROC yang berada di atas garis acak (biru putus-putus) menunjukkan kinerja yang lebih baik, dan nilai *Area Under the Curve* (AUC) memberikan gambaran agregat tentang seberapa baik model dapat membedakan antara kelas positif dan negatif. Berdasarkan gambar tersebut, dapat dilihat garis orange yang merupakan implementasi dari kurva ROC yang berada tepat diatas garis biru putus-putus. Ini menunjukkan bahwa model memiliki tingkat sensitivitas yang baik dibandingkan dengan spesifisitas acak.

### 5.1.4 Analisis Visualisasi Data Precision-Recall Curve




Gambar 17. Precision-Recall Curve Klasifikasi Data Malware

Penganalisisan model tidak akan lengkap tanpa adanya penganalisisan terkait precision dan recall. Kedua metrix evaluasi klasifikasi ini memberikan informasi terkait performa model tentang data yang tidak seimbang *(imbalance class).* Gambar diatas merupakan diagram

yang menampilkan relasi antara Precision dan Recall. Sehubungan dengan itu, diagram ini berasal dari hasil eksekusi test set dengan penjelasan yang dapat diambil adalah :

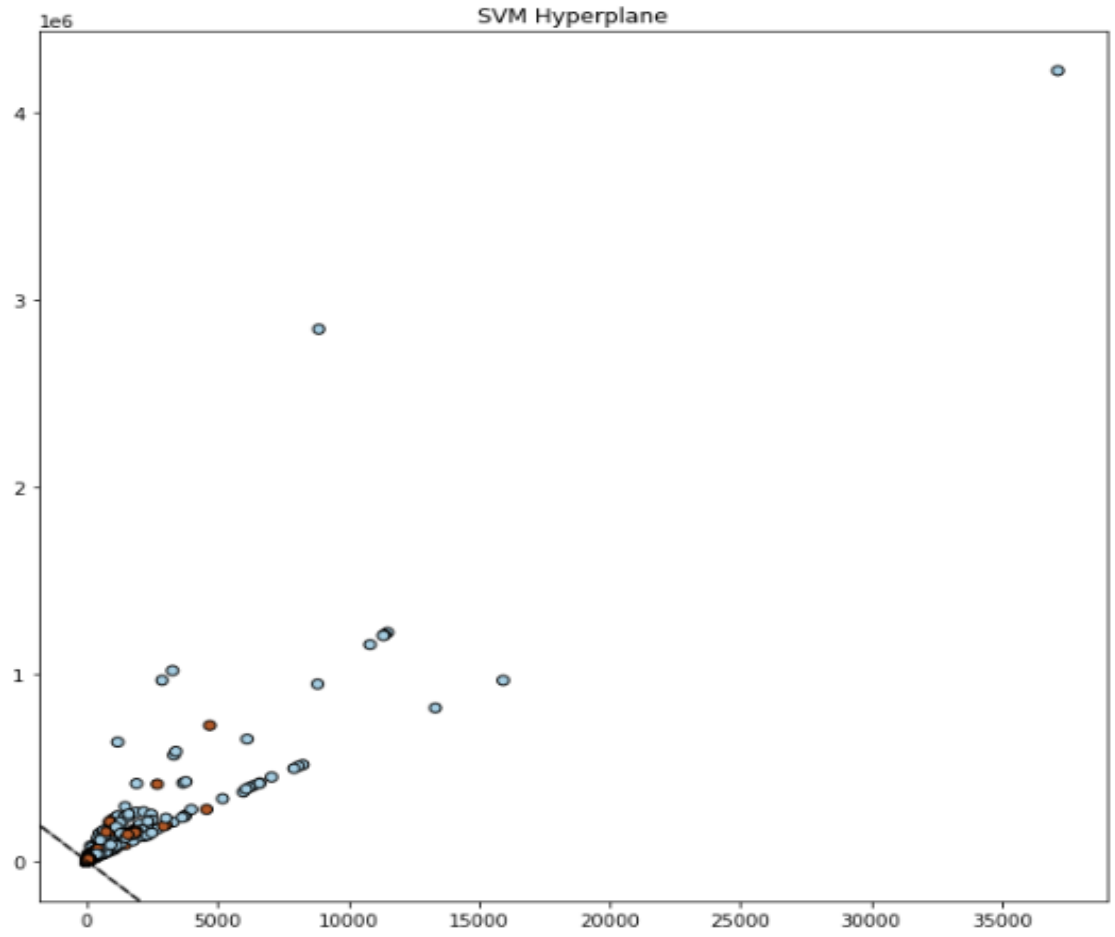
- Pada kelas benign (0), precision sebesar 0,69 menunjukkan bahwa 69% dari prediksi positif untuk kelas benign adalah benar, sementara 31% adalah false positive. Kemudian pada bagian recall yang sebesar 0.88 menunjukkan bahwa model mampu mengidentifikasi semua sampel positif dari kelas benign sebesar 88%
- Pada kelas malicious (1), Precision sebesar 0.70 menunjukkan bahwa 70% dari prediksi positif untuk kelas malicious adalah benar, tetapi 30% adalah false positive. Kemudian, pada bagian recall yang sebesar 0.40 menunjukkan bahwa model gagal mengidentifikasi sebagian besar sampel positif dari Kelas 1.
- Melalui relasi antara precision dan recall pada kelas benign, dapat disimpulkan bahwa F1-score yang tinggi sebesar 0,77 atau 77% adalah harmonic mean yang artinya precision dan recall tersebut memberikan gambaran keseimbangan antara keduanya.
- Melalui relasi antara precision dan recall pada kelas malicious, dapat disimpulkan bahwa F1-score yang sedang sebesar 0,51 atau 51% mencerminkan keseimbanga antara precision dan recall berada dalam kondisi cukup untuk dinyatakan seimbang.

Secara umum, model yang memiliki nilai recall yang tinggi dianggap sebagai model yang baik. Hal ini karena model tersebut mampu mengidentifikasi sebanyak mungkin sampel positif dari kelas yang dimaksud [38]. Dalam kasus ini, nilai recall untuk kelas benign adalah tinggi, yaitu 0,88. Hal ini menunjukkan bahwa model mampu mengidentifikasi sebagian besar sampel positif dari kelas benign. Nilai presisi untuk kelas benign juga cukup tinggi, yaitu 0,69. Hal ini menunjukkan bahwa model juga cukup akurat dalam memprediksi kelas yang benar. Model ini dapat dianggap sebagai model yang baik untuk kelas benign. Sementara, nilai recall untuk kelas malicious adalah 0,40, sedangkan nilai presisi adalah 0,70. Hal ini menunjukkan bahwa model gagal mengidentifikasi sebagian besar sampel positif dari kelas malicious, tetapi hanya 70% dari prediksi positif untuk kelas malicious yang benar. Nilai recall yang rendah pada kelas malicious menunjukkan bahwa model tidak mampu mengidentifikasi sebagian besar serangan malware. Hal ini dapat menyebabkan serangan malware tidak terdeteksi dan dapat menyebabkan kerugian. Secara keseluruhan,

model ini memiliki kinerja yang buruk untuk kelas malicious. Hal ini perlu diperbaiki agar model dapat lebih efektif dalam mendeteksi serangan malware.

## 5.2 Analisis Data Setelah Implementasi SVM

### 5.2.1 Analisis Data pada Visualisasi Data Hyperplane 2D



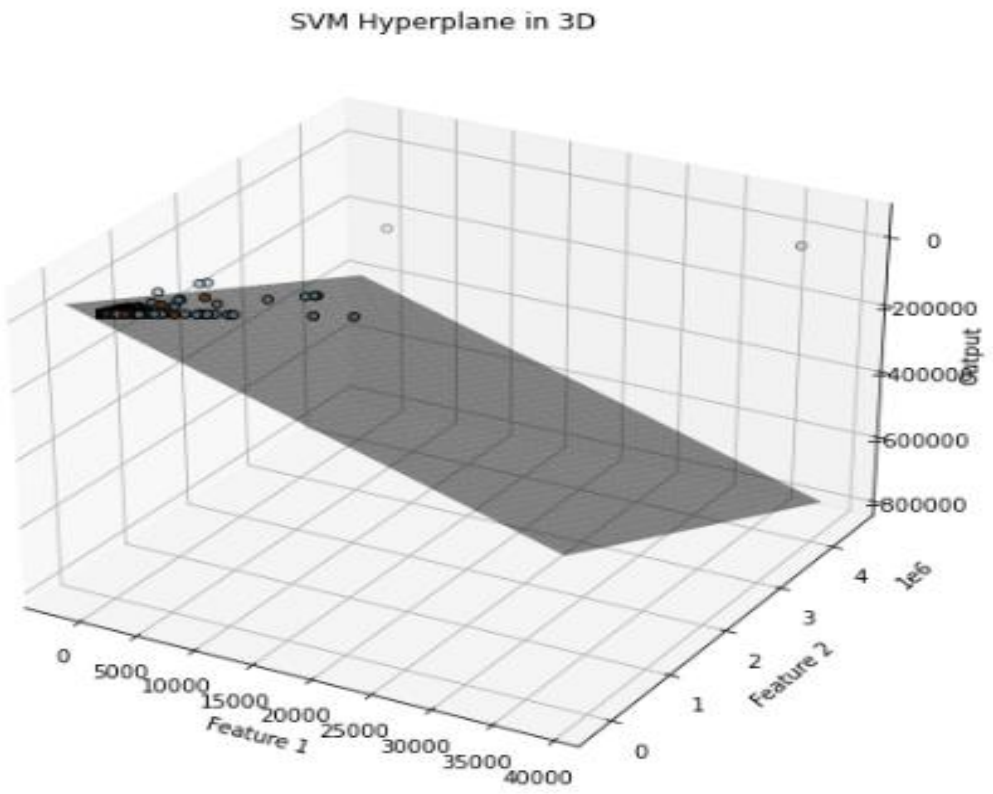

Gambar 18. Hyperplane SVM in 2D

Hyperplane adalah suatu bidang yang memisahkan data menjadi dua kelas. Dalam konteks SVM, hyperplane ini dipilih sedemikian rupa sehingga jarak *(margin)* antara hyperplane dan titik-titik terdekat dari masing-masing kelas (yang disebut sebagai vektor dukungan atau support vectors) adalah maksimum. Pada visualisasi tersebut dapat diamati bahwa terdapat beberapa data yang terletak dekat dengan garis hyperplane. Terdapat grafik sebaran *(scatter plot)* dari jumlah objek di dalam gambar. Objek-objek tersebut memiliki warna dan

bentuk yang berbeda-beda, dan semuanya diatur sedemikian rupa sehingga terlihat seperti melayang di udara.

Hiperplane SVM memisahkan dua kelas objek, yaitu objek dengan jumlah objek yang sedikit (kelas 0) dan objek dengan jumlah objek yang banyak (kelas 1). Data-data ini disebut yang paling dekat ini akan disebut sebagai support vectors. Dimana, support vectors memiliki peran penting dalam menentukan posisi garis hyperplane. Data yang semakin dekat dengan support vectors masing-masing akan memisahkan kelas dengan lebih baik. Namun, pada penerapannya di dataset android_traffic ini, hasil yang didapatkan masih belum dapat memisahkan kedua kelas yang diharapkan menjadi dua bagian yang jelas dikarenakan oleh datanya yang terlalu banyak dan tidak tersebar sehingga tidak cocok untuk pemodelan menggunakan SVM. Objek-objek yang berada di bawah hiperplane diklasifikasikan sebagai kelas 0, sedangkan objek-objek yang berada di atas hiperplane diklasifikasikan sebagai kelas 1 Hiperplane ini memisahkan dua kelas objek, yaitu objek dengan jumlah objek yang sedikit *benign* (kelas 0) dan objek dengan jumlah objek yang banyak *malicious* (kelas 1). Berdasarkan grafik tersebut, dapat dilihat bahwa objek-objek dengan jumlah objek yang sedikit cenderung berada di bawah hiperplane, sedangkan objek-objek dengan jumlah objek yang banyak cenderung berada di atas hiperplane. Hal ini pun semakin diperkuat dengan adanya beberapa data yang terletak di luar garis hyperplane. Data-data ini disebut sebagai *outliers* yang dapat mempengaruhi akurasi klasifikasi model SVM. Sebab, Outlier dapat mengubah struktur margin dan pada gilirannya, mempengaruhi kemampuan SVM untuk menggeneralisasi dengan baik pada data baru.

### 5.2.2 Analisis Data pada Visualisasi Data Hyperplane 3D




Gambar 19. Hyperplane SVM in 3D

Secara keseluruhan, gambar ini menunjukkan bagaimana hyperplane dapat digunakan untuk memisahkan dua kelas data dalam ruang berdimensi 3. Pembentukan visualisasi model SVM tidak menutup kemungkinan divisualisasikan dalam bentuk 3D seperti gambar tersebut. Walaupun masih belum cukup optimal bagaimana hyperplane tersebut dapat memisahkan kelas berdasarkan support vektor yang diterapkannya. Tetapi dapat kita lihat bahwasanya gambar tersebut menunjukkan grafik sebaran *(scatter plot)* dari dua fitur data, yaitu feature 1 dan feature 2. Titik-titik di dalam grafik mewakili data dari dua kelas, yaitu kelas 0 dan kelas 1. Posisi kelas-kelas yang ingin diklasifikasikan, sebagian besar

masih terdapat dibagian bawah perpotongan bidang hyperplane. Berdasarkan grafik tersebut, dapat dilihat bahwa titik-titik data dari kelas 0 cenderung berada di bawah hiperplane, sedangkan titik-titik data dari kelas 1 cenderung berada di atas hyperplane. Titik-titik data pada gambar menunjukkan dua kelas data, yaitu kelas positif (ditunjukkan dengan warna merah) dan kelas negatif (ditunjukkan dengan warna biru). Hyperplane membagi dua kelas data ini dengan benar, sehingga semua titik data pada satu kelas berada di satu sisi hyperplane dan semua titik data pada kelas lain berada di sisi lain hyperplane.

Margin adalah jarak antara hiperplane SVM dan titik-titik data terdekat dari masing-masing kelas. Margin yang besar menunjukkan bahwa hiperplane SVM dapat memisahkan dua kelas data dengan baik. Dalam gambar ini, margin marginnya cukup besar, sehingga hyperplane memisahkan dua kelas data dengan baik walaupun belum tervisualisasi dengan jelas. Berdasarkan heatmap korelasi untuk penentuan label model SVM ini, serta hasil dari visualisasi data hyperplane yang didapatkan dapat kita simpulkan bahwasanya kelas benign yang dataset-nya berwarna biru memiliki memiliki nilai korelasi terhadap tcp_packets dan vulume_bytes yang lebih tinggi dari kelas malicious yang berwarna orange. Hal ini ditandai pada nilai korelasi antara tcp_packets dan volume_bytes untuk kelas benign lebih besar daripada nilai korelasi untuk kelas malicious. Nilai korelasi untuk kelas benign adalah 0,7, sedangkan nilai korelasi untuk kelas malicious adalah 0,6.

# 6 Kesimpulan

## 6.1 Kesimpulan

Berdasarkan hasil pengujian yang telah dilakukan, diperoleh beberapa hasil penelitian sebagai berikut:

- Model SVM pada dataset Android_traffic memberikan akurasi 68.96%, menunjukkan kemungkinan penggunaan yang cukup optimal untuk pengklasifikasian malware.
- Hyperplane SVM berhasil memisahkan kelas benign dan malicious dalam ruang tiga dimensi, meskipun visualisasi belum optimal. Margin yang cukup besar menunjukkan kemampuan memisahkan kelas dengan baik.
- Terdapat data dekat garis hyperplane sebagai support vectors, menunjukkan kemampuan SVM menangani data berdekatan, namun pemodelan pada dataset Android_traffic masih perlu ditingkatkan.
- Outliers di luar garis hyperplane dapat mempengaruhi akurasi SVM. Penanganan outliers menjadi krusial untuk meningkatkan generalisasi model pada data baru.
- Heatmap korelasi menunjukkan kelas benign memiliki nilai korelasi lebih tinggi antara **tcp_packets** dan **volume_bytes** dibandingkan kelas malicious, indikasi keberhasilan model memahami pola data.
- Keseimbangan pengeksekusian model pada kelas benign dan malicious terlihat dari precision dan recall, memberikan persentase keseimbangan yang berkategori sedang.
- Tantangan dalam pemodelan SVM di dataset ini karena data terlalu banyak dan tidak tersebar secara optimal, mempengaruhi kemampuan SVM memisahkan kedua kelas dengan jelas.

## 6.2 Saran

Dalam pengembangan sistem kecerdasan buatan dengan model SVM dalam klasifikasi, pengambilan dataset untuk mengklasifikasikan harus dapat memenuhi kapasitas data untuk pengeksekusian berdasarkan model yang dikembangkan hal ini bertujuan untuk menghindari overfitting dan memastikan pengeksekusian model yang optimal. Adanya kesalahan atau ketidakseimbangan data seperti kapasitas yang terlalu besar akan memberikan dampak pada ketidakakuratan pengkalsifikasian data.

# REFERENSI


[1] G. R. Kanagachidambaresan, A. Ruwali, D. Banerjee, and K. B. Prakash, "Klasifikasi Malware Menggunakan MetodeRecurrent Neural Network," *EAI/Springer Innov. Commun. Comput.*, vol. 23, no. 3, pp. 53–61, 2021.

[2] Hafiz and Meidi Dwi, "Visualisasi dan Klasifikasi Malware Menggunakan Metode K-Nearest Neighbor (K-NN). ," *Indralaya Univ. Sriwij.*, 2020, [Online]. Available: https://repository.unsri.ac.id/39918/18/RAMA_56201_09011281520097_0003047905_01_front_ref.pdf.

[3] Y. Ilhamdi and Y. N. Kunang, "Analisis Malware Pada Sistem Operasi Windows Menggunakan Teknik Forensik," *Bina Darma Conf. Comput. Sci.*, vol. 3, pp. 256–264, 2021, [Online]. Available: https://conference.binadarma.ac.id/index.php/BDCCS/article/view/2124.

[4] B. Guc *et al.*, "Semantic annotation of GPS trajectories," *11th Agil. Int. Conf. Geogr. Inf. Sci. 2008*, vol. 38, no. January 2008, pp. 1–9, 2008.

[5] F. C. Venna, "Implementasi Steganografi Audio pada File Wav dengan metode Redundant Pattern Encoding (RPE) Berbasis Sndroid," *Repository.Uinjkt.Ac.Id*, 2019, [Online]. Available: http://repository.uinjkt.ac.id/dspace/handle/123456789/47958.

[6] M. F. Syawal, D. C. Fikriansyah, and N. Agani, "Implementasi Teknik Steganografi Menggunakan Algoritma Vigenere Cipher Dan Metode LSB," *J. TICOM*, vol. 4, no. 3, pp. 91–99, 2016.

[7] N. K. Gyamfi, N. Goranin, D. Ceponis, and H. A. Čenys, "Automated System-Level Malware Detection Using Machine Learning: A Comprehensive Review," *Appl. Sci.*, vol. 13, no. 21, p. 11908, 2023, doi: 10.3390/app132111908.

[8] E. S. Lamdompak Sistem Komputer and F. Ilmu Komputer, "Klasifikasi Malware Trojan Ransomware Dengan Algoritma Support Vector Machine (SVM)," vol. 2, no. 1, pp. 122–127, 2016, [Online]. Available: http://ars.ilkom.unsri.ac.id.

[9] D. I. Pushpita Anna Octaviani, Yuciana Wilandari, "Penerapan Metode SVM Pada Data Akreditasi Sekolah Dasar Di Kabupaten Magelang," *J. Gaussian*, vol. 3, no. 8, pp. 811–820, 2014.

[10] D. Zhang, H. Huang, Q. Chen, and Y. Jiang, "A Comparison Study of Credit

Scoring Models," *Proc. - Third Int. Conf. Nat. Comput. ICNC 2007*, vol. 1, no. June, pp. 15–18, 2007, doi: 10.1109/ICNC.2007.15.

[11] F. Rachman and Santi Wulan Purnami, "Perbandingan Klasifikasi Tingkat Keganasan Breast Cancer Dengan Menggunakan Regresi Logistik Ordinal Dan Support Vector Machine (SVM)," *Chest*, vol. 73, no. 2 suppl., pp. 293–299, 1978, doi: 10.1378/chest.73.2_supplement.293.

[12] E. V. Tjahjadi and B. Santoso, "Klasifikasi Malware Menggunakan Teknik Machine Learning," *J. Ilm. Ilmu Komput.*, vol. 2, no. 1, pp. 60–70, 2023.

[13] Ari Sandriana, Rianto, and Firmansyah Maulana, "Klasifikasi serangan Malware terhadap Lalu Lintas Jaringan Internet of Things menggunakan Algoritma K-Nearest Neighbour (K-NN)," *E-JOINT (Electronica Electr. J. Innov. Technol.*, vol. 3, no. 1, pp. 12–22, 2022, doi: 10.35970/e-joint.v3i1.1559.

[14] N. Chitayae and A. H. Muhammad, "Identifikasi Malware pada Android menggunakan Algoritma K-Nearest Neighbor," *J. Inf. Technol.*, vol. 3, no. 2, pp. 63–68, 2023, doi: 10.46229/jifotech.v3i2.752.

[15] A. S. Nugraha and K. K. Purnamasari, "Penerapan Metode Support Vector Machine Pada Part of Speech Tag Bahasa Indonesia," no. 112, 2019.

[16] N. A. Khalil and B. M. Khammas, "an Effective and Efficient Features Vectors for Ransomware Detection Via Machine Learning Technique," *Iraqi J. Inf. Commun. Technol.*, vol. 5, no. 3, pp. 23–33, 2022, doi: 10.31987/ijict.5.3.205.

[17] W. Widayani and H. Harliana, "Analisis Support Vector Machine Untuk Pemberian Rekomendasi Penundaan Biaya Kuliah Mahasiswa," *J. Sains dan Inform.*, vol. 7, no. 1, pp. 20–27, 2021, doi: 10.34128/jsi.v7i1.268.

[18] H. C. S. Ningrum, "Perbandingan Metode Support Vector Machine (SVM) Linear, Radial Basis Function (RBF), dan Polinomial Kernel dalam Klasifikasi Bidang Studi Lanjut Pilihan Alumni UII," *Tugas Akhir Stat. Univ. Islam Indones.*, pp. 1–90, 2018.

[19] P. Fremmuzar and A. Baita, "Uji Kernel SVM dalam Analisis Sentimen Terhadap Layanan Telkomsel di Media Sosial Twitter," *Komputika J. Sist. Komput.*, vol. 12, no. 2, pp. 57–66, 2023, doi: 10.34010/komputika.v12i2.9460.

[20] O. Article, "Malware Detection in Cloud Computing Infrastructures Manuscript," *Environ. Educ. Res.*, 2011.

[21] A. S. Shamili, C. Bauckhage, and T. Alpcan, “Malware detection on mobile devices using distributed machine learning,” *Proc. - Int. Conf. Pattern Recognit.*, pp. 4348–4351, 2010, doi: 10.1109/ICPR.2010.1057.

[22] T. N. Turnip, A. Situmorang, A. Lumbantobing, J. Marpaung, and S. I. G. Situmeang, “Android malware classification based on permission categories using extreme gradient boosting,” *ACM Int. Conf. Proceeding Ser.*, no. November, pp. 190–194, 2020, doi: 10.1145/3427423.3427427.

[23] M. Y. Darsyah, “KLASIFIKASI TUBERKULOSIS DENGAN PENDEKATAN METODE SUPPORTS VECTOR MACHINE (SVM),” *J. Kesehat.*, vol. 10, no. 3, p. 405, 2019, doi: 10.26630/jk.v10i3.1479.

[24] H. P. P. Zuriel and A. Fahrurozi, “Implementasi Algoritma Klasifikasi Support Vector Machine Untuk Analisa Sentimen Pengguna Twitter Terhadap Kebijakan Psbb,” *J. Ilm. Inform. Komput.*, vol. 26, no. 2, pp. 149–162, 2021, doi: 10.35760/ik.2021.v26i2.4289.

[25] W. Agustina, M. T. Furqon, and B. Rahayudi, “Implementasi Metode Support Vector Machine (SVM) Untuk Klasifikasi Rumah Layak Huni (Studi Kasus: Desa Kidal Kecamatan Tumpang Kabupaten Malang),” *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 2, no. 10, pp. 3366–3372, 2018, [Online]. Available: http://j-ptiik.ub.ac.id.

[26] B. Thomas, “Problem Formulation in Social Work Research: Issues and Concerns,” *Int. J. Soc. Econ. Res.*, vol. 5, no. 4, p. 65, 2015, doi: 10.5958/2249-6270.2015.00055.0.

[27] D. N. Boote and P. Beile, “Scholars Before Researchers: On the Centrality of the Dissertation Literature Review in Research Preparation,” *Educ. Res.*, vol. 34, no. 6, pp. 3–15, 2005, doi: 10.3102/0013189X034006003.

[28] D. Huyler and C. McGill, “Research Design: Qualitative, Quantitative, and Mixed Methods Approaches, by John Creswell and J. David Creswell. Thousand Oaks, CA: Sage Publication, Inc. 275 pages, $67.00 (Paperback).,” *New Horizons Adult Educ. Hum. Resour. Dev.*, vol. 31, pp. 75–77, Jul. 2019, doi: 10.1002/nha3.20258.

[29] S. Ramírez-Gallego, B. Krawczyk, S. García, M. Woźniak, and F. Herrera, “A survey on data preprocessing for data stream mining: Current status and future directions,” *Neurocomputing*, vol. 239, pp. 39–57, 2017, doi:

10.1016/j.neucom.2017.01.078.

[30] K. K. Dobbin and R. M. Simon, “Optimally splitting cases for training and testing high dimensional classifiers,” *BMC Med. Genomics*, vol. 4, 2011, doi: 10.1186/1755-8794-4-31.

[31] B. Raharjo, *Pembelajaran Mesin (Machine Learning)*. YAYASAN PRIMA AGUS TEKNIK, 2016.

[32] P. Adytia, W. Wahyuni, K. Sussolaikah, and Y. Satria, “Klasifikasi Penggunaan Data Trafik Internet Menggunakan Algoritma Support Vector Machine,” *J. Komput. dan Inform.*, vol. 11, no. 1, pp. 96–102, 2023, doi: 10.35508/jicon.v11i1.10039.

[33] R. Pupale, “Support Vector Machines(SVM) — An Overview,” 2018, [Online]. Available: https://towardsdatascience.com/https-medium-com-pupalerushikesh-svm-f4b42800e989.

[34] D. D. D. Kemala, “PERBANDINGAN EMPIRIS FUNGSI KERNEL PADA METODE KLASIFIKASI SUPPORT VECTOR MACHINE DAN PENERAPANNYA PADA DATA PENDERITA PENYAKIT JANTUNG,” pp. 1–14, 2023, [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK558907/.

[35] D. H. Anto Satriyo Nugroho, Arief Budi Witarto, “Support Vector Machine - Teori dan Aplikasinya dalam Bioinformatika1,” *Proc. 2011 Chinese Control Decis. Conf. CCDC 2011*, pp. 842–847, 2011, doi: 10.1109/CCDC.2011.5968300.

[36] Aniruddha Bhandari, “Understanding & Interpreting Confusion Matrix in Machine Learning (Updated 2023),” 2023. https://www.analyticsvidhya.com/blog/2020/04/confusion-matrix-machine-learning/.

[37] K. J. Radiol., “Receiver Operating Characteristic (ROC) Curve: Practical Review for Radiologists.” https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2698108/.

[38] N. Uly, H. Hendry, and A. Iriani, “CNN-RNN Hybrid Model for Diagnosis of COVID-19 on X-Ray Imagery,” *Digit. Zo. J. Teknol. Inf. dan Komun.*, vol. 14, no. 1, pp. 57–67, 2023, doi: 10.31849/digitalzone.v14i1.13668.

# LAMPIRAN

## Lampiran 1. Pembagian Tugas Kelompok 10

**Tabel 2.** Pembagian Tugas Anggota Kelompok

| No | NIM | Nama | Peran | Tugas |
|---|---|---|---|---|
| 1. | 12S21027 | Rebecca Yulyartha Bulawan Sihombing | Ketua | Bab III Metode Penelitian, Bab IV Implementasi, Bab V Analisis Hasil Pengujian |
| 2. | 12S21037 | Immanuella Eklesia Lumbantobing | Anggota | Bab IV Implementasi, Bab V Analisis Hasil Pengujian, PowerPoint |
| 3. | 12S21039 | Widya Indah Sari Manurung | Anggota | Bab III Metode Penelitian |
| 4. | 12S21053 | Chesya Ivana J.M Sitorus | Anggota | Bab II Studi Literatur, Bab IV Implementasi, Penulisan Dokumen |
| 5. | 12S21058 | Grace Christina Yohanna Situmorang | Anggota | Bab I Pendahuluan, Bab II Studi Literatur, Bab IV Implementasi, Penulisan Dokumen |

**Tabel 3.** Persentase Kontribusi Pengerjaan

| No | NIM | Nama | Kontribusi |
|---|---|---|---|
| 1. | 12S21027 | Rebecca Yulyartha Bulawan Sihombing | 23,75% |
| 2. | 12S21037 | Immanuella Eklesia Lumbantobing | 23,75% |
| 3. | 12S21039 | Widya Indah Sari Manurung | 5% |
| 4. | 12S21053 | Chesya Ivana J.M Sitorus | 23,75% |
| 5. | 12S21058 | Grace Christina Yohanna Situmorang | 23,75% |