Studi Pengklasifikasi Berdasarkan Machine Learning Untuk Mendeteksi Malware Berbasis Metode PE Probe

Tugas Akhir

Kelompok Keahlian: Cyber Physical System

Aria Fajar Pratama
NIM: 1301164473

Program Studi Sarjana Teknik Informatika

Fakultas Informatika

Universitas Telkom

Bandung

2022

Lembar Pengesahan

Studi Pengklasifikasi Berdasarkan Machine Learning Untuk Mendeteksi Malware Berbasis Metode PE Probe

Aria Fajar Pratama
NIM: 1301164473

Tugas Akhir ini diterima dan disahkan untuk memenuhi sebagian dari syarat
untuk memperoleh gelar sarjana Sarjana Komputer
Program Studi Sarjana Teknik Informatika
Fakultas Informatika Universitas Telkom

Bandung, 3 Maret 2023
Menyetujui

Pembimbing 1

Satria Mandala, PhD
NIP: 16730040

Mengesahkan,
Kepala Program Studi Teknik Informatika

Erwin Budi Setiawan, S.Si., M.T.
NIP: 00760045-1

Abstrak

Malware adalah perangkat lunak atau program berbahaya yang dapat merusak sistem operasi komputer dan jaringan. Banyak penelitian telah dilakukan untuk mencegah serangan malware yang dapat menimbulkan kerugian. Para peneliti telah mengembangkan metode deteksi dan klasifikasi malware berdasarkan metode statis dan dinamis. Namun, kedua metode ini memiliki kekurangan masing-masing, seperti metode statis yang kurang efektif dalam mendeteksi jenis file baru, dan metode dinamis yang lebih memakan sumber daya dan memiliki biaya yang lebih tinggi.

Para peneliti juga telah mengembangkan berbagai teknik untuk mendeteksi malware berdasarkan fitur-fitur Pe-Probe. Untuk mengatasi masalah tersebut, tugas akhir ini mengusulkan pengembangan algoritma deteksi malware berbasis fitur Pe-Probe menggunakan machine learning untuk meningkatkan akurasi deteksi dan klasifikasi. Metode yang digunakan dalam penelitian ini adalah studi literatur tentang deteksi dan klasifikasi malware, pengembangan algoritma klasifikasi untuk deteksi malware berbasis metode Pe-Probe, pengembangan prototype, pengujian performansi, dan analisis.

Penelitian ini menggunakan dataset sebanyak 19611 file Pe-Probe, yang terdiri dari data Pe-Probe yang terinfeksi malware dan tidak terinfeksi malware. Dengan menggunakan model machine learning stacking dan parameter tuning, hasil validasi menunjukkan nilai akurasi sebesar 0.978, detection rate sebesar 0.9905, dan false alarm rate sebesar 0.057.

Kata Kunci: Malware, Machine Learning, Pe-probe.

Lembar Persembahan

Alhamdulillah,segala puji Allah SWT dengan kemurahan dan ridho-Nya, Tugas Akhir ini dapat ditulis dengan baik dan lancar hingga selesai. Dengan ini akan kupersembahkan Tugas Akhir ini kepada :

1. Nabi ku, Nabi Muhammad SAW sebagai panutan umat muslim yang penuh dengan kemuliaan dan ketaatan kepada Allah SWT memberiku motivasi tentang kehidupan dan mengajari ku hidup melalui sunnah-sunnahnya. Kedua orang tua ku tersayang yang selalu memberikan ku ketenangan, kenyamanan, motivasi, doa terbaik dan menyisihkan finansial nya, sehingga aku bisa menyelesaikan studi ku. Kalian sangat berarti bagiku.

2. Kedua orang tua ku tersayang yang selalu memberikan ku ketenangan, kenyamanan, motivasi, doa terbaik dan menyisihkan finansial nya, sehingga aku bisa menyelesaikan studi ku. Kalian sangat berarti bagiku.

3. Guruku sekaligus orang tua kedua ku di kampus (pembimbing tugas akhir) Bapak Satria Mandala, Ph.D yang telah sabar membimbing ku untuk menyelesaikan tugas akhirku. Jasamu takkan pernah kulupakan.

Kata Pengantar

Dengan mengucapkan puji syukur kehadirat Allah SWT Tuhan Yang Maha Esa, karena kasih dan karunia-Nya sehingga penulis dapat menyelesaikan Tugas Akhir ini yang berjudul "Studi Pengklasifikasi Berdasarkan Machine Learning Untuk Mendeteksi Malware Berbasis Metode PE Probe". Tugas Akhir penelitian ini disusun untuk memenuhi salah satu syarat memperoleh gelar Sarjana Komputer di Telkom University. Dalam penyusunan proposal penelitian ini, penulis mengalami kesulitan dan penulis menyadari dalam penulisan proposal penelitian ini masih jauh dari kesempurnaan. Untuk itu, penulis sangat mengharapkan kritik dan saran yang membangun demi kesempurnaan proposal penelitian ini. Maka, dalam kesempatan ini pula penulis menyampaikan ucapan terima kasih yang sebesar-besarnya kepada bapak Satria Mandala, Ph.D selaku dosen pembimbing tugas akhir yang telah banyak memberikan arahan dan bimbingan kepada penulis selama proses penyelesaian proposal penelitian ini. Penulis sangat berharap semoga proposal penelitian ini bermanfaat bagi kita semua. Akhir kata, penulis mengucapkan terima kasih.

Daftar Isi

Daftar Gambar

Daftar Tabel

Bab I

Pendahuluan

1.1 Latar Belakang

Dengan pesatnya perkembangan internet, malware telah menjadi salah satu ancaman utama di dunia cyber saat ini. Malware dapat berupa perangkat lunak berbahaya, pencurian data dan informasi, serta spionase. Menurut definisi Kaspersky Labs (2017), malware adalah program komputer yang dirancang untuk menginfeksi komputer pengguna dan menimbulkan kerusakan di dalamnya dengan berbagai cara.

Perlindungan dari malware pada sistem komputer adalah tugas keamanan cyber yang penting bagi pengguna karena satu serangan saja dapat mengakibatkan kerusakan yang cukup besar. Saat ini, metode populer dalam mendeteksi malware menggunakan teknik klasifikasi statis dan dinamis yang berdasarkan algoritma yang mempelajari signature based dari malware, seperti yang dijelaskan oleh Chumachenko (2017).

Namun, terdapat banyak pro dan kontra mengenai metode statis dan dinamis. Kelebihan teknik analisis statik adalah malware tidak dijalankan, sehingga mengurangi risiko terinfeksi malware. Meskipun begitu, metode ini memiliki keterbatasan, seperti tidak dapat mendeteksi jenis malware baru. Saat ini, penulis malware sering menggunakan teknik yang membuat proses analisis malware statik semakin sulit, seperti menggunakan packer untuk melakukan modifikasi kode secara otomatis. Di sisi lain, analisis dinamis lebih efisien dan tidak memerlukan executable untuk dibongkar atau didekripsi. Namun, metode ini memakan waktu dan sumber daya yang banyak, seperti yang dijelaskan oleh Nataraj, Karthikeyan, Jacob and Manjunath (2011).

Penelitian lainnya, seperti yang dilakukan oleh Oh, Go and Lee (2017), menggunakan deteksi malware berbasis signature-based dengan mengidentifikasi ciri-ciri malware yang berada di database. Namun, metode ini tidak sepenuhnya optimal ketika terjadi serangan zero-day, dan banyak malware yang tidak terdeteksi.

Pada tahun Vyas, Luo, McFarland and Justice (2017). melakukan penelitian dengan menginvestigasi fitur statis untuk mengusulkan sistem deteksi malware jaringan waktu nyata. Fitur-fitur tersebut diekstraksi dari header

dan bagian file PE dan dikelompokkan menjadi empat kategori: metadata file, file pengepakan, DLL yang diimpor, dan fungsi yang diimpor. Fitur-fitur ini kemudian digunakan untuk melatih beberapa mesin pengklasifikasi pembelajaran.

Penelitian lainnya, seperti yang dilakukan oleh Rezaei, Manavi and Hamzeh (2021) menggunakan teknik deteksi malware berbasis Pe-Probe (Pe file) yang berdasarkan 9 fitur dari pe header dengan tingkat akurasi 95,5% menggunakan Random Forest.

Dari permasalahan itu, penelitian ini berfokus pada pengembangan deteksi dan klasifikasi akurasi malware berbasis feature-feature yang terdapat dalam pe-probe (pe file), dengan membuat sebuah prototype untuk deteksi malware dengan menggunakan machine learning untuk mendeteksi dan mengklasifikasikan akurasi dari PE-Probe (pe file). Diharapkan penelitian ini dapat menjadi metode optimal dalam mendeteksi malware.

1.2 Latar belakang di atas, rumusan masalah tugas akhir ini adalah sebagai berikut:

1. Bagaimana cara algoritma machine learning dapat mengklasifikasikan malware ?

2. Bagaimana cara mengembangkan *prototype* untuk mendeteksi dan kalsifikasi malware berdasarkan studi algoritma klasifikasi yang telah dilakukan?

3. Bagaimana cara algoritma machine learning dapat mendeteksi malware berdasarkan fitur fitur pe-probe?

1.3 Tujuan

1. Melakukan studi algoritma untuk meningkatkan akurasi klasifikasi malware.

2. Melakukan analisis dan membuat pengembangan deteksi malware menggunakan machine learning berbasis *PE-Probe(Pe-File.*

3. membuat validasi deteksi malware pada prototype

1.4 Batasan Masalah

Berikut adalah ruang lingkup yang ada pada penulisan tugas akhir ini :

1. Sistem yang dibuat hanya untuk membandingkan machine learning yang terbaik untuk mendeteksi malware yang terdapat dalam pe-probe.

2. Klasifikasi dan deteksi malware hanya berbasis dataset yang sudah dipelajari oleh machine learning.

3. Pengujian deteksi malware hanya bisa menggunakan jenis file berbentuk .exe, dll.

4. sistem yang dibuat tidak untuk mendeteksi jenis family malware

1.5 Hipotesis

1. Algoritma deteksi malware menghasilkan akurasi yang tinggi dan akurat

2. Metode yang diusulkan menjadi metode yang lebih baik dalam mendeteksi malware

1.6 Sistematika Penulisan

Tugas Akhir ini disusun dengan sistematika penulisan sebagai berikut :

- **BAB I Pendahuluan**. Bab ini membahas mengenai latar belakang, rumusan masalah, dan tujuan pengerjaan Tugas Akhir ini.

- **Bab II Kajian Pustaka**. Bab ini membahas fakta dan teori yang berkaitan dengan perancangan sistem untuk mendirikan landasan berfikir. Dengan menggunakan fakta dan teori yang dikemukakan pada bab ini penulis menganalisis kebutuhan akan rancangan arsitektur sistem yang dibangun.

- **BAB III Metodologi dan Desain Sistem**. Bab ini menjelaskan metode penelitian, rancangan sistem dan metode pengujian yang dilakukan dalam penelitian.

- **BAB IV Hasil dan Pembahasan**. Bab ini menjelaskan hasil uji dan analisa hasil uji dari yang telah dilakukan sesuai teori dan metode yang digunakan

- **BAB IV Hasil dan Pembahasan**. Bab ini menjelaskan Kesimpulan dan saran dari keseluruhan penelitian ini

Bab II

Kajian Pustaka

Bab ini menjelaskan riset terkait tugas akhir dan landasan teori pendukung yang digunakan. Riset Terkait diuraikan di Sub Bab 2.1, sedangkan landasan teori dapat ditemukan pada Sub Bab 2.2 dan 2.3. Ringkasan disajikan pada bagian terakhir dari Bab 2.

2.1 Penelitian Terkait

Penelitian tentang deteksi malware sudah bermula sejak tahun 2005. Berikut adalah 15 penelitian terkait yang sudah dipublikasikan sejak tahun 2015 sampai sekarang.

Han, Jin, Wang, Wang and Yuan (2021) melakukan penelitian dengan metode *4-LFE* dengan cara mengekstrak multi-fitur dari program jahat dengan menggabungkan fitur *piksel* dan *n-gram* untuk mengklasifikasi malware. Setelah melakukan pengujian kepada data publik terdiri dari 10.868 sampel malware, mencapai nilai akurasi sebesar 99,99%.

Kim, Woo, Lee and Chung (2016) melakukan penelitian deteksi malware menggunakan fitur *PE-Header*. Algoritma yang digunakan dalam penelitian deteksi malware dengan cara menggabungkan algoritma *Cart*, *SVC* dan *SGD*. Hasil penelitian ini menghasilkan akurasi sebesar 99,99%

Liu, sheng Wang, Yu and xi Zhong (2017) mengusulkan sistem analisis malware berbasis pembelajaran mesin, yang terdiri dari tiga modul: pemrosesan data, pengambilan keputusan, dan deteksi malware baru. Modul pemrosesan data menggunakan fitur *gray-scale images*, *Opcode n-gram*, dan *import functions*, untuk modul pengambilan keputusan menggunakan fitur klasifikasi dan mengindentifikasi malware, dan modul deteksi malware menggunakan fitur algoritma *SNN*. Pengujian menggunakan 20.000 contoh malware dengan hasil akurasi sebesar 86,7%.

Udayakumar, Saglani, Cupta and Subbulakshmi (2018) mengusulkan teknik baru dalam mendeteksi dan klasifikasi malware dengan teknik analisis menggunakan alat *Knime* dan *Orange*, menggunakan 6 algoritma yang berbeda. Dari hasil pengujian data set pada mongo Db algoritma Random Forest mencapai nilai akurasi sebesar 63,49% pada *Knime* dan 94,2% pada *Orange*

Mangialardo and Duarte (2015) melakukan percobaan dengan menggabungkan analisis statis dan dinamis dalam menganalisis klasifikasi dan indentifikasi malware. Pengujian ini menggunakan algoritma *C.50* dan *Random Forest* yang di implementasikan dalam *Framework*. Percobaan ini menghasilkan akurasi sebesar 95,75%.

Pai, Troia, Visaggio, Austin and Stamp (2017) melakukan penelitian klasifikasi malware dengan teknik *clustering* menggunakan algoritma *K-means* dan *Expectation Maximizational*. metode *clustering* yang digunakan berdasarkan perhitungan skor *Hidden Markov Model*, memvariasikan jumlah cluster dari 2 dan 10, dan jumlah dimensi (skor) dari 2 hingga 5. pengujian ini menggunakan 8000 sampel malware dan menhasilkan nilai akurasi sebesar 90%

Shafiq, Tabish, Mirza and Farooq (2009) juga melakukan penelitian deteksi malware serangan *Zero-day* dalam struktur *framework PE-Miner* menggunakan fitur yang distandarisasi oleh sistem operasi *Microsoft* untuk *file executable, DLLs* dan file objek. Pengujian di lakukan selama 1jam dapat mengeskusi kumpulan data sebanyak lebih dari 17ribu data dan menghasilkan nilai akurasi sebesar 99% .

Xue, Li, Lv, Wu and Wang (2019) juga melakukan percobaan dalam mengklasifikasi malware berdasarkan penilaian *probabilitas* dan *machine learning*. Tahap 1 menggunakan *neural networks* dengan *Spatial Pyramid Pooling* untuk menganalisi gambar *grayscale*(fitur dinamis), Tahap 2 menggunakan *variable n-gram* dan *machine learning*, dan malscore digunakan untuk menggabungkan tahap 1 dan tahap 2. Dari eksperimen pada 174.607 sampel malware dari 63 jenis malware, *malscore* dapat mengklasifikasi dengan nilai akurasi sebesar 98,82%.

Abijah Roseline, Geetha, Kadry and Nam (2020) juga mengusulkan sebuah metode sistem mendeteksi malware pendekatan visualisasi 2d dengan digabungkan dengan pembelajaran mesin. *(machine learning)*. Hasil dari pengujian sistem terhadap teknik yang lebih canggih seperti *Malimg, BIG2015,* dan *MaleVis malware datasets*, memiliki tingkat akurasi sebesar 98,65%, 97,2%, dan 97,43%.

Di tahun yang sama Chen, Su, Lee and Bair (2020) melakukan percobaan deteksi dan klasifikasi malware dengan mengimplementasikan teknik klasifikasi menggunakan *support vector machine (SVM)* dengan menggabungkan algortima pembelajaran aktif *(ALBL)* untuk melakukan klasifikasi malware. Pengujian dilakukan terhadap data malware yang sudah dikumpulkan oleh *Microsoft Malware Classification Challenge* (BIG 2015) di Kaggle dan mampu meningkatkan performa *machine learning* dalam mendeteksi dan klasifikasi malware.

Das, Liu, Zhang and Chandramohan (2016) juga melakukan percobaan dalam mendeteksi dan klasifikasi malware secara online dengan meningkatkan perangkat keras, dan *Guard OL* (gabungan *prosesor dan FPGA*). Algortima

yang digunakan adalah *multilayer perceptron* untuk melatih pendeteksian dini malware berbahaya atau jinak. metode tersebut menghasilkan nilai akurasi pada prediski awal sebesar 46% malware dalam 30% eksekusi pertama, sedangkan 97% sampel dari 100% setelah eksekusi.

Liu, Lai, Wang and Yan (2019)melakukan penelitian pengklasifikasian malware dengan cara memvisualisasikan malware. Metode yang digunakan adalah *LBP (Local Binary Patterns)* dan *(Scale-invariant feature transform)* dengan mengelompokan malware kedalam blok menggunakan model *bag-of-visual-words (BoVW)*. Pengujian dilakukan terhadap 3 database malware dan hasil pengujian dapat meningkatkan klasifikasi yang cukup baik dan canggih.

Sahu, Ahirwar and Shukla (2015) juga mengusulkan teknik klasifikasi dan deteksi malware berbasis grafik yang digunakan untuk mengumpulkan fitur malware yang berbeda. Metode yang digunakan adalah metode *hybrid*, berdasarkan *DAG* dan *Gaussian Support Vector Machines*, untuk klasifikasi malware. pengujian dilakukan berdasarkan data *KDD cup 1999* dan memberikan hasil akurasi yang tinggi dengan tingkat kesalahan pendeteksian di bawah 1%

Wuechner, Cisłak, Ochoa and Pretschner (2017) juga mengusulkan teknik deteksi malware menggunakan metode *compression-based mining* pada grafik aliran data kuantitatif. Dari hasil pengujian yang dilakukan terhadap kumpulan malware yang beragam, hasil pengujian mengungguli model deteksi berbasis frekuensi dalam hal efektifitas sebesar 600%.

Xia, Pan, Chen, Bai and Yang (2018) melakukan penelitian deteksi malware berbasis gambar. Metode yang digunakan menggunakan *SVM* untuk mendeteksi malware file, dan metode *Markov Transition Field (MTF)* untuk memeriksa dan klasifikasi malware yang nantinya akan di rubah kembali dalam gambar satu dimensi ke bentuk *vektor SVM*. Penelitian tersebut dilakukan didataset dan menghasilkan yang lebih baik di bandingkan metode berbasis gambar *grayscale byteplot*

Perbandingan hasil penelitian di atas dapat dilihat pada tabel di bawah ini:

Tabel 2.1: Ringkasan riset terkait

No.	Judul	Penulis / Tahun	Hasil Riset	Kelebihan	Kekurangan
1	Classification of malware for self-driving systems	Xiangyu Han, Fusheng Jin, Runan Wang, Shuliang Wang, Ye Yuan /2019	dapat diterapkan dengan 10 algoritma machine learning yang berbeda dan mencapai nilai akurasi klasifikasi yang efisien	memiliki nilai akurasi yang tinggi yaitu sebesar 99,99%.	Belum mampu untuk mendeteksi secara *real-time*
2	Static Detection of Malware and Benign Executable Using Machine Learning Algorithm	Dong-Hee Kim, Sang-Uk Woo, Dong-Kyu Lee, Tai-Myoung Chun / 2016	Tingkat kesalahan dalam mendeteksi sangat rendah ketika menggabung algoritma *CART*,*SVC*	Memiliki nilai efisiensi, akurasi yang tinggi sebesar 99,99%	Banyak memakan waktu dan sumber daya
3	Automatic malware classification and new malware detection using machine learning	Liu LIU, Bao-sheng WANG, Bo YU, Qiu-xi ZHONG / 2017	Meningkatkan akurasi dan deteksi secara efektif	Memiliki nilai akurasi yang cukup tinggi yaitu 86,7%	Pengujian hanya dilakukan terhadap malware yang telah dikumpulkan oleh Kingsoft, ESET NOD32, and Anubis
4	Malware Classification Using Machine Learning Algorithms and Tools	Ginika Mahajan,Bhavna Saini,Shivam Anand / 2013	Algoritma random forest memiliki nilai akurasi yang tinggi di bandingkan algoritma yang lain	Nilai akurasi cukup tinggi sebesar 63,49% pada *Knime* dan 94,2% pada *Orange*	Teknik ini belum diuji pada serangan malware secara langsung

Tabel 2.1: Ringkasan riset terkait

No.	Judul	Penulis / Tahun	Hasil Riset	Kelebihan	Kekurangan
5	Integrating Static and Dynamic Malware Analysis Using Machine Learning	R. J. Mangialardo, J. C. Duarte / 2015	Meningkatkan Teknik static dan dinamis dalam mendeteksi dan klasifikasi malware	Nilai akurasi cukup tinggi sebesar 95,75%	
6	Clustering for malware classification	Pai1, Fabio Di Troia2, Corrado Aaron Visaggio, Thomas H.Austin, Mark Stamp / 2016	dapat mendeteksi jenis malware baru secara otomatis	Pengujian deteksi memiliki akurasi 90%	Pengujian hanya dilakukan pada dataset *HMM*
7	PE-Miner: Mining Structural Information to Detect Malicious Executables in Realtime	M. Zubair Shafiq, S. Momina Tabish, Fauzan Mirza, Muddassar Farooq/ 2009	Dapat digunakan di sistem operasi *Unix* dan *non-Windows*	Dapat mendeteksi malware secara *real time* dan nilai akurasi yang tinggi	mengalami *overhead* saat mengeksekusi file tertentu selama 0,244 detik .
8	Malware classification using probability scoring and machine learning	DI XUE1,JINGMEI LI1,TU LV,WEIFEI WU,AND JIAXIANG WANG / 2019	metode penggabungan analisis klasifikasi malware statis dan dinamis	*Malscore* memakan waktu yang lebih rendah saat mengklasifikasi malware dan mendeteksi malware	.

Tabel 2.1: Ringkasan riset terkait

No.	Judul	Penulis / Tahun	Hasil Riset	Kelebihan	Kekurangan
9	Intelligent Vision-Based Malware Detection and Classification Using Deep Random Forest Paradigm	S. ABIJAH ROSELINE, SEIFEDINE KADRY, S. GEETHA,Yunyoung Nam / 2020	dapat mendeteksi malware yang belum teridentifikasi dan serangan *zero-day*	dapat mendeteksi malware dengan visualisasi 2d *width*	Belum diujikan menggunakan data *real time*.
10	Malware Family Classification using Active Learning by Learning	Chin-Wei Chen, Ching-Hung Su, Kun-Wei Lee, Ping-Hao Bair / 2020	meningkatkan deteksi dan klasifikasi malware dengan menggabungkan jaringan *adversarial generatif (GAN)*	meningkatkan kinerja *machine learning* dalam hal klasifikasi dan deteksi malware	Algoritma dan metode masih belum bisa di terapkan dalam mendeteksi malware di dunia nyata.
11	Semantics-based Online Malware Detection: Towards Efficient Real-time Protection Against Malware	Sanjeev Das, Yang Liu, Wei Zhang, Mahintham Chandramohan / 2016	dapat mendeteksi malware yang berada di *software* dan *hardware*	nilai akurasi deteksi cukup tinggi dalam mendeteksi malware	Masih diperlukannya peningkatan algoritma dan metode dalam mendeteksi malware.
12	A New Learning Approach to Malware Classification using Discriminative Feature Extraction	YASHU LIU, YUKUN LAI, ZHIHAI WANG, HANBING YAN / 2016	fitur deskriptor dan klasifikasi malware yang kuat,tangguh dan canggih	klasifikasi malware cukup akurat	tidak terlalu akurat saat klasifikasi jika malware yang di*enksripsi*

Tabel 2.1: Ringkasan riset terkait

No.	Judul	Penulis / Tahun	Hasil Riset	Kelebihan	Kekurangan
13	Improved Malware Detection Technique Using Ensemble Based Classifier and Graph Theory	Manish Kumar Sahu, Manish Ahirwar,Piyush Kumar Shukla / 2015	menghasilkan hasil data malware yang akurat dalam pengelompokan klasifikasi	akurasi tinggi dan tingkat kesalahan deteksi rendah yang diukur di bawah 1%	belum di implementasikan untuk mendeteksi serangan malware secara *real-time*.
14	Leveraging Compression-based Graph Mining for Behavior-based Malware Detection	Tobias W¨uchner, Aleksander Cisłak, Mart´ın Ochoa, Alexander Pretschner / 2017	metode baru *compression-based mining* dapat mendeteksi secara efektif sebesar 600%	klasifikasi lebih baik di bandingkan menggunakan metode berbasis QDFG	kurang efisiensi dalam menentukan dan langkah mengklasifikasi malware
15	Malware Classification with Markov Transition Field Encoded Images	Shiming Xia, Zhisong Pan, Zhe Chen, Wei Bai, Haimin Yang / 2018	dapat mendeteksi malware berbasis gambar yang lebih baik dibandingkan metode *grayscale byteplot*	dapat digunakan secara real-time	masih diperlukannya pengembangan lebih lanjut agark lebih efisien.

2.2 Malware

Menurut Mohd Faizal Ab Razak (2016) *malware* adalah sebuah perangkat lunak berbahaya yang di rancang untuk memberikan efek kerugian dan kerusakan pada sistem operasi dan jaringan. Jenis malware pada umumnya yang menyebabkan kerusakan pada sitem operasi dan jaringan adalah *Rootkit, worm, spyware, virus, Adware* dan *trojan.*
Berikut beberapa informasi mengenai jenis jenis malware :

1. **Virus**
   Virus merupakan jenis malware yang paling sederhana dan umum, virus dapat melipatkan gandakan dirinya sendiri dan menginfeksi perangkat lunak dan file.

2. **Worm**
   Worm merupakan jenis malware yang sangat mirip dengan virus, dapat melipat gandakan dirinya sendiri, melalui jaringan tanpa arahan dari pembuat malware.

3. **Rootkit**
   Rootkit adalah sekumpulan perangkat lunak yang berfungsi untuk menyerang dan mengakses data administratif sebagai pengguna yang tidak sah. keberadaan rootkit seringkali tidak terlihat dan menyembuyinkan dirinya, sehingga rootkit sulit di deteksi dan di hapus keberadaanya.

4. **Spyware**
   Spyware digunakan untuk mencuri data, mengumpulkan informasi dan melakukan spionase kepada target tertentu tanpa sepengetahuannya.aktivitas tersebut biasanya di jual kepada pihak ketiga yang membutuhkan.

5. **Trojan**
   Trojan membuat rekayasa seolah-olah yang di unduh oleh pengguna berpikir sebagai perangkat lunak yang sah. Ketika mendapatkan akses ke sistem, trojan akan melancarkan serangan dengan merusak sebuah sistem dan jaringan.

6. **Adware**
   Adware dibuat untuk menampilkan iklan di browser dengan cara melacak browser pengguna dan riwayat unduhan untuk menampilkan iklan pop-up yang memikat pengguna untuk melakukan pembelian.

2.3 Machine Learning

Menurut Kajaree Das, Rabi Narayan Behera (2017)*Machine Learning* adalah sebuah konsep yang merujuk pada pembelajaran dari pengalaman masa

lalu (data) untuk meningkatkan pembelajaran untuk di masa yang akan datang. *Machine learning* merupakan cabang dari *artificial intelligence (AI)* yang berfokus pada pembelajaran dari data dan meningkatnya akurasi dari waktu ke waktu.

Machine Learning digunakan untuk berbagai jenis pekerjaan untuk membantu manusia dalam mendapatkan data dan informasi yang lebih akurat untuk memudahkan dalam pengambilan keputusan. Machine learning memiliki beberapa metode yang digunakan dalam mendapatkan data dan informasi.

Dalam beberapa dekade terakhir, metode machine learning telah digunakan dalam studi deteksi dan klasifikasi malware dan memberikan banyak pengembangan anti virus. Secara khusus, berbagai metode machine learning telah diusulkan untuk mendeteksi dan memahami perkembangan malware. *Machine learning* seperti *K-NN*, *Random Forest*, dan *SVM* telah diterapkan untuk mendeteksi dan klasifikasi malware. *Machine learning* yang digunakan dapat menghasilkan akurasi yang optimal terhadap deteksi dan klasifikasi malware.

2.4 Ringkasan

Ada beberapa hal yang perlu menjadi catatan dari Bab ini sebagai berikut:

1. Para peneliti sudah banyak melakukan penelitian tentang studi deteksi dan klasifikasi malware dengan berbagai metode dan machine learning yang digunakan dalam penelitian, dengan tingkat akurasi deteksi dan klasifikasi lebih dari 80%

2. malware merupakan sebuah program atau perangkat lunak yang menyebabkan banyak kerugian bagi pengguna baik dari segi ekonomi, keamanan privasi, dan informasi.

3. machine learning merupakan sebuah algoritma yang membantu manusia dalam menyelesaikan pekerjaan untuk mengambil keputusan untuk informasi dan data yang akan digunakan. machine learning memiliki beberapa metode yang memudahkan untuk menerapkan dalam algoritma yang akan diterapkan dalam berbagai bidang pekerjaan.

Bab III

Metodologi dan Desain Sistem

3.1 Metode Penelitian

3.1.1 Framework Penelitian

Metodologi yang dilakukan dalam menyelesaikan penelitian ini ditunjukan pada diagram alir 3.1 dibawah ini :

Gambar 3.1: Diagram Alir Riset *Framework*

Berikut penjelasan dari masing-masing tahapan riset :

1. **Studi Litaratur**
   Studi literatur dilakukan untuk mempelajari dan memahami permasalahan yang dilakukan sebagai objek penelitian sehingga masalah dan solusi didapatkan berdasarkan penelitian sebelumnya. Tahap ini penulis melakukan studi terhadap paper-paper, jurnal maupun artikel yang berhubungan dengan objek penelitian.

2. **Penulisan Proposal**
   Penulisan proposal dilakukan untuk sebagai rancangan dari penelitian yang akan dilakukan oleh penulis sehingga penelitian dapat berjalan baik.

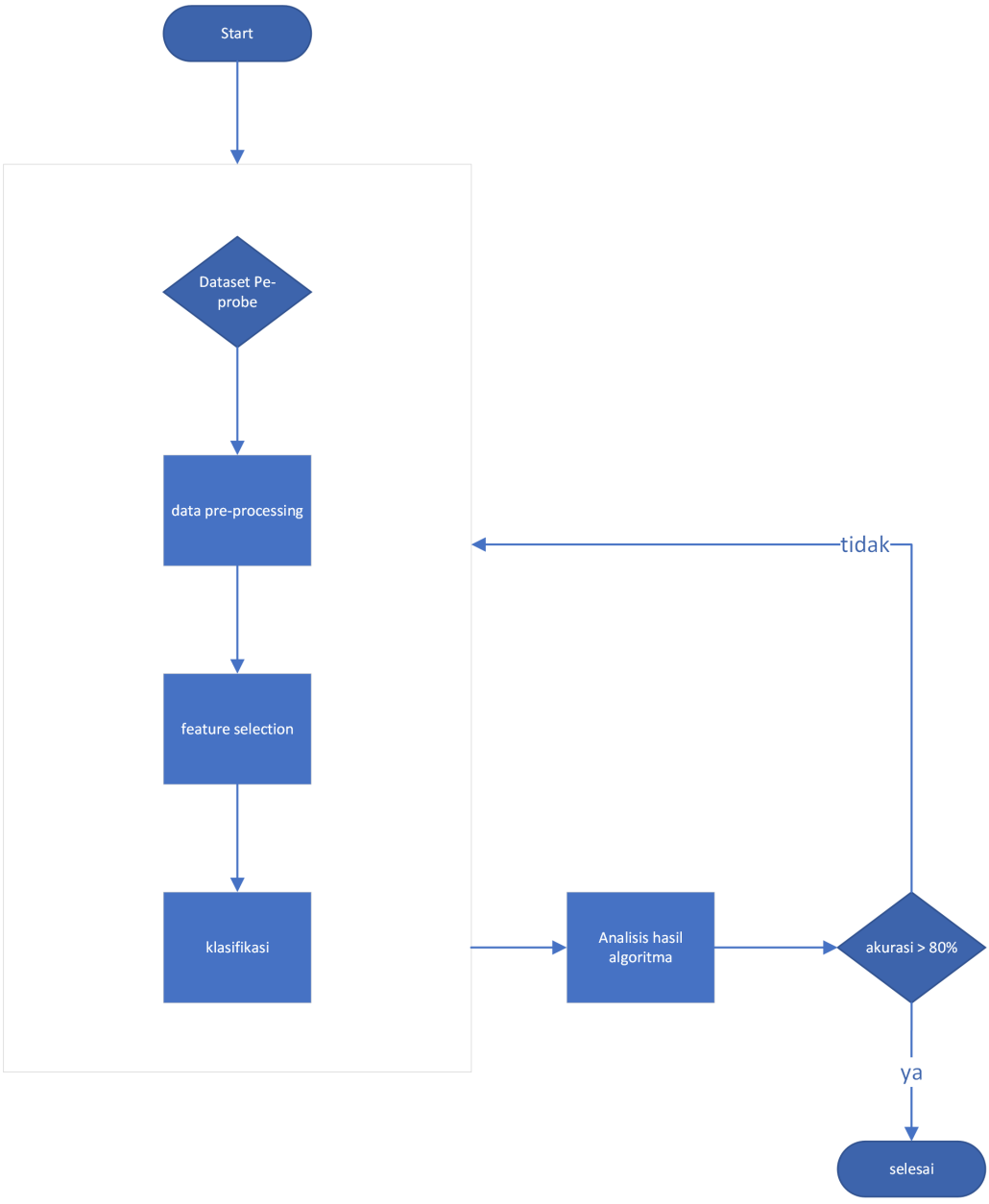
3. **Pembangunan Sistem**
   Pembangunan sistem dilakukan penulis untuk mendapatkan hasil analisis dari data yang didapatkan dari prototype mulai dari preprocessing, ekstraksi fitur, seleksi fitur dan klasifikasi.

4. **Pengujian**
   Pengujian dilakukan untuk menguji terhadap prototype dan sistem yang telah dibangun sehingga mendapatkan akurasi dari penelitian ini.

5. **Penyusunan Laporan**
   Penyusunan laporan dilakukan penulis agar menyelesaikan penelitian ini sehingga menghasilkan dokumen paper berdasarkan hasil dan analisis yang telah dilakukan dari studi literatur sampai dengann pengujian.

3.1.2 Metodologi untuk Mencapai Tujuan Penelitian

A) **M**etodologi untuk mencapai objectif pertama
   Metodologi yang dilakukan dalam mencapai objektif pertama adalah sebagai berikut :



Gambar 3.2: Diagram Alir Metodologi Objektif Pertama

Berikut adalah penjelasan untuk setiap tahapan metodologi :

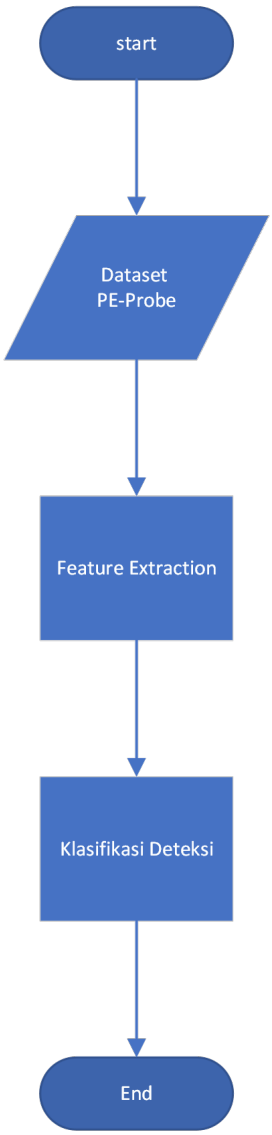
(a) **Dataset PE-Probe**
Dataset PE-Probe(Pe file) adalah sebuah dataset yang berisikan fitur-fitur data mengenai informasi sebuah aplikasi yang dapat di-jalankan atau dieksekusi oleh sistem operasi windows.

(b) **Data Preprocessing**
Data preproccesing digunakan untuk menginputkan data kepada algoritma yang akan dibangun dan menormalisasi data untuk mengurangi penyimpangan nilai data yang besar.

(c) **Feature Selection**
Feature selection digunakan untuk memilih feature yang paling berpengaruh yang akan digunakan untuk dipelajari oleh algoritma machine learning

(d) **Analisis Hasil Algoritma Klasifikasi**
Analisis hasil digunakan untuk mengetahui algoritma machine learning terbaik untuk pengklasifikasian deteksi malware

B) **M**etodologi untuk mencapai objectif kedua
Berikut adalah skema *prototype* yang akan dibangun untuk mencapai objektif kedua :



Gambar 3.3: Diagram Alir Metodologi Objektif Kedua

Berikut adalah penjelasan dari masing-masing tahapan :

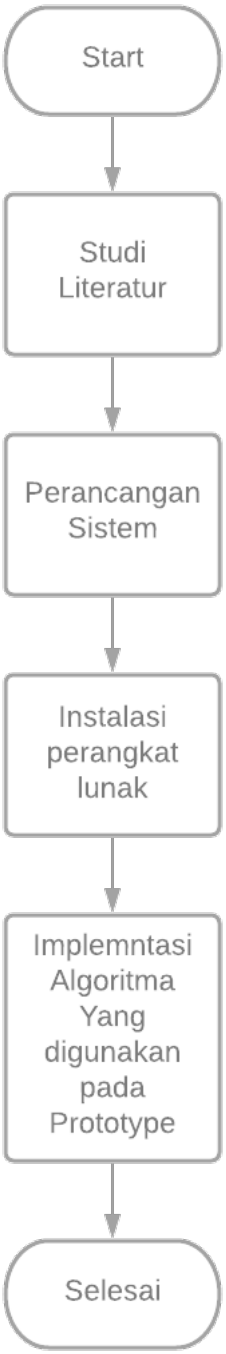
(a) **Dataset Pe-Probe**
Pe-Probe adalah file yang dapat di jalankan dan di eksekusi oleh sistem komputer dan berformat .exe dan .dll. Pada bagian ini akan mengambil data testing sebesar 20% yang akan dideteksi pada tahap selanjutnya.

(b) **Feature Extarction**
Pada tahap ini algoritma membaca isi fitur-fitur dalam sebuah pe-

probe(pe-file) yang nantinya akan dideteksi oleh model machine learning yang dibuat.

(c) **Klasifikasi dan Deteksi**
Pada tahap ini dilakukan deteksi pe-probe untuk mendeteksi file berisi malware atau tidak berisi malware berdasarkan dataset test dan menggunakan model machine learning yang telah dibuat, dan menentukan machine learning paling optimal yang akan digunakan pada protoype.

C) **M**etodologi untuk mencapai objektif ketiga Metodologi yang dilakukan dalam mencapai objektif ketiga adalah sebagai berikut :



Gambar 3.4: Diagram Alir Metodologi Objektif Ketiga

Berikut adalah penjelasan untuk setiap tahapan metodologi :

(a) **Studi Literatur**
Pada tahap ini dilakukan studi untuk melakukan pengembangan-pengembangan *prototype* sejenis yang telah dilakukan. Hal ini bertujuan untuk mempelajari bagaimana sistem deteksi malware bekerja pada umumnya, melakukan riset tentang perangkat lunak yang diperlukan dalam membangun sistem, dan batasan-batasan sistem.

(b) **Perancangan Sistem**
Pada tahap ini dilakukan perancangan sistem berdasarkan litaratur yang telah dipelajari antara lain, mekanisme bagaimana data diproses, dan bagaimana informasi dari data tersebut diberikan.

(c) **Instalasi Perangkat Lunak**
Pada tahap ini dilakukan implementasi dari hasil perancangan sistem, dengan membuat aplikasi yang dapat mendeteksi malware.

(d) **Implementasi Algoritma Pada Sistem**
Pada tahap ini dilakukan implementasi dari algoritma yang telah disiapkan untuk diterapkan dalam sistem. Hasil dari tahapan ini adalah sistem dapat menjalankan algoritma dengan baik dan memberi hasil prediksi probability berdasarkan validasi uji pe-probe.

3.1.3 Analisis Kebutuhan Sistem

A) **S**pesifikasi Perangkat Keras

- Laptop Processor Intel() Core(TM) i5-8250U @1.60GHz
- Memory 8GB
- Hard Drive 1TB

B) **S**pesifikasi Perangkat Lunak

- Windows 10 64bit
- Python 2.7
- Visual Studio Code

3.1.4 Data

Data yang digunakan dalam melakukan penelitian ini adalah data Malware yang berbentuk fitur ektrasi dalam pe-probe(pe-files) dengan jumlah data sebanyak 19610 jenis pe file yang mengandung malware dan tidak mengandung malware

3.1.5 Metrik Uji

Metrik pengujian yang digunakan dalam melakukan pengujian algoritma adalah metrik yang juga digunakan pada penelitian-penelitian sebelumnya Shafiq, Tabish and Farooq (2009) dan Radwan (2019). Meliputi akurasi dan spesifikasi.

Accuracy

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{3.1}$$

Detection Rate

$$DR = \frac{TP}{TP + FN} \tag{3.2}$$

False Alarm Rate

$$FAR = \frac{FP}{FP + TN} \tag{3.3}$$

Di mana TP(True Positive) untuk mendeteksi file Malware, FN(False Negative) mendeteksi file bukan malware namun sebenarnya malware, FP(False Positive) mendeteksi file bukan malware namun dianggap malware, dan TN(True Negative) mendeteksi file bukan malware.

3.1.6 Metode Pengujian

Untuk mengetahui keberahasilan seluruh rancangan diperlukan adanya pengujian, baik secara perangkat maupun algoritma. Hal ini ditujukan mengetahui apakah tujuan tugas akhir ini tercapai.

Tujuan Pengujian

Tujuan dari pengujian ini adalah untuk membuktikan akurasi dan deteksi malware dari algoritma yang dibangun dan menguji hasil algoritma dan sistem yang dibangun

Skenario Pengujian :

1. **Skenario 1 : Pengujian algortima Klasifikasi**
   Pada skenario ini dilakukan pengujian terhadap 3 algoritma machine learning yaitu stacking, bagging, dan boosting. Dengan menginputkan dataset training berisikan nilai nilai feature struktur dari pe-probe dan mengeluarkan nilai akurasi klasifikasi pe-probe malware dan bukan malware

2. **Skenario 2 : Pengujian deteksi pe-pobe**
   Pada skenario ini, dilakukan pengujian deteksi terhadap pe-probe, dengan mengekstrasi nilai fitur-fitur pe-probe yang akan diuji berdasarkan dataset test dan mengeluarkan nilai akurasi deteksi pe-probe yang terinfeksi malware dan tidak terinfeksi malware , yang nantinya untuk menentukan machine learning tebaik yang akan digunakan pada prototype.

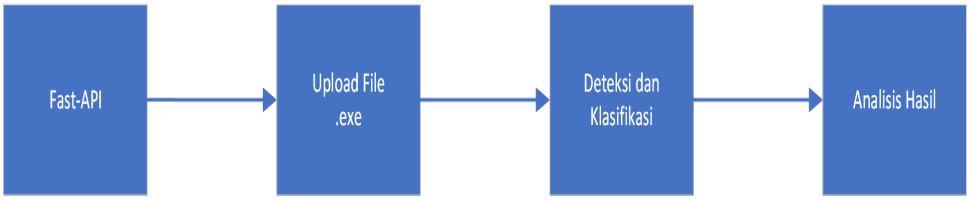
3. **Skenario 3 : Implementasi algoritma pada prototype**
   Pada skenario ini dilakukan implementasi hasil algoritma yang telah disiapkan untuk diterapkan pada sistem. hasil pada tahapan ini adalah sistem dapat menjalakan algoritma dengan baik dan mengeluarkan nilai probabilitas prediksi terhadap validasi pe-probe yang akan diuji.

3.1.7 Perbandingan Hasil Penelitian

Tugas Akhir ini melakukan perbandingan hasil yang didapat dengan penelitian sejenis yang telah dilakukan oleh Radwan (2019)

3.2 Desain Sistem

Gambar 3.5 adalah ilustrasi desain dari sistem dari tugas akhir ini.



Gambar 3.5: Desain Sistem yang direncanakan

3.3 Ringkasan

Ada beberapa hal yang perlu menjadi catatan dari Bab ini sebagai berikut:

1. Penelitian ini terdapat 3 metodelogi sebagai acuan gambaran kerangka kerja dari penelitian yang akan dilakukan, dan terdapat metrik uji sebagai pengujian algoritma penelitian.

2. Penelitian nilai hasil akurasi klasifikasi dan deteksi malware diharapkan diatas 80%

Bab IV

Hasil dan Pembahasan

Pada bab ini akan dibahas hasil dari algoritma klasifikasi malware dan hasil pengujian skenario yang dilakukan terhadap sample pe-probe yang terinfeksi malware dan tidak terinfeksi Malware

4.1 Hasil Pengujian

Setelah melaksanakan pengujian sistem seperti yang telah dibahas pada bab sebelumnya sub bab ini akan memaparkan hasil dari percobaan.

4.1.1 Hasil Pencapaian Metodologi Objektif Pertama

Algoritma pengujian klasifikasi pe-probe yang diusulkan bertujuan untuk mempelajari dan membedakan pe-probe yang terinfeksi malware dan tidak terinfeksi malware berdasarkan dataset. Hasil dari algoritma klasifikasi pe-probe berdasarkan kfold cross validadion menggunakan 80% data training yang dapat dilihat sebagai berikut :

Tabel 4.1: Hasil validasi Kfold data train machine learning bagging tidak menggunakan parameter

K-fold Bagging tidak menggunakan parameter(tuning) dengan data train 80%				
Fold	Accuracy	Recall	Precission	F1-score
1	0.992	0.998	0.992	0.995
2	0.992	0.996	0.993	0.995
3	0.992	0.997	0.991	0.994
4	0.994	0.996	0.992	0.996
5	0.991	0.997	0.990	0.994
6	0.993	0.997	0.993	0.995
7	0.993	0.998	0.992	0.995
8	0.993	0.998	0.992	0.995
9	0.991	0.998	0.990	0.994
10	0.990	0.995	0.991	0.993

Tabel 4.2: Hasil validasi Kfold data train machine learning Boosting tidak menggunakan parameter

K-fold Boosting tidak menggunakan parameter(tuning) dengan data train 80%				
Fold	Accuracy	Recall	Precission	F1-score
1	0.990	0.997	0.989	0.993
2	0.993	0.996	0.991	0.995
3	0.991	0.998	0.990	0.994
4	0.991	0.998	0.989	0.994
5	0.991	0.996	0.992	0.994
6	0.990	0.998	0.988	0.993
7	0.990	0.998	0.988	0.993
8	0.989	0.998	0.988	0.993
9	0.990	0.997	0.989	0.993
10	0.990	0.998	0.991	0.994

Tabel 4.3: Hasil validasi Kfold data train machine learning Stacking tidak menggunakan parameter

K-fold Stacking tidak menggunakan parameter(tuning) dengan data train 80%				
Fold	Accuracy	Recall	Precission	F1-score
1	0.997	1.0	0.999	0.999
2	0.997	1.0	0.999 8	0.999
3	1.0	1.0	1.0	1.0
4	0.997	1.0	0.999	0.999
5	0.999	1.0	0.999	0.999
6	0.999	1.0	0.999	0.999
7	0.999	1.0	0.999	0.999
8	0.999	1.0	0.999	0.999
9	0.999	1.0	0.999	0.999
10	0.999	1.0	0.999	0.999

Berdasarkan Tabel 4.1, 4.2, dan 4.3 merupakan hasil validasi kfolds machine learning tidak menggunakan parameter terhadap data train. Machine learning Bagging mendapatkan nilai accuracy tertinggi yaitu 0.994, untuk nilai recall tertinggi yaitu 0.998, untuk nilai precission tertinngi sebesar 0.993, dan nilai f1-score tertinggi yaitu 0.996. Untuk machine learning Boosting berhasil mendapatkan score accuracy tertinggi sebesar 0.993, untuk precission tertingi yaitu dengan nilai 0.998, sedangkan untuk score tertinggi pada recall sebesar 0.998, dan untuk F1-score nilai tertinggi sebesar 0.995. Lalu untuk machine learning Stacking nilai accuracy tertinggi sebesar 1.0, recall tertinggi 1.0, precission 1.0, dan F1-score 1.0. Pada tahap selanjutnya dilakukan deteksi berdasarkan kfold cross validation menggunakan machine learning dengan parameter(tuning).

Tabel 4.4: Hasil validasi Kfold data train machine learning Bagging menggunakan parameter

K-fold Bagging menggunakan parameter(tuning) dengan data train 80%				
Fold	Accuracy	Recall	Precission	F1-score
1	0.991	0.997	0.991	0.994
2	0.992	0.997	0.992	0.994
3	0.994	0.998	0.993	0.996
4	0.994	0.998	0.993	0.995
5	0.993	0.996	0.993	0.996
6	0.994	0.998	0.992	0.995
7	0.992	0.998	0.992	0.995
8	0.992	0.998	0.992	0.995
9	0.993	0.998	0.993	0.995
10	0.993	0.998	0.991	0.995

Tabel 4.5: Hasil validasi Kfold data train machine learning Boosting menggunakan parameter

K-fold Boosting menggunakan parameter(tuning) dengan data train 80%				
Fold	Accuracy	Recall	Precission	F1-score
1	0.996	0.999	0.995	0.997
2	0.996	1.0	0.994	0.997
3	0.995	0.999	0.993	0.996
4	0.997	0.999	0.996	0.998
5	0.994	0.999	0.993	0.996
6	0.995	0.998	0.995	0.996
7	0.995	0.998	0.995	0.996
8	0.995	0.998	0.995	0.997
9	0.996	0.999	0.995	0.997
10	0.996	0.999	0.995	0.997

Tabel 4.6: Hasil validasi Kfold data train machine learning Stacking menggunakan parameter

K-fold Stacking menggunakan parameter(tuning) dengan data train 80%				
Fold	Accuracy	Recall	Precission	F1-score
1	0.996	0.995	1.0	1.0
2	0.996	0.994	1.0	1.0
3	0.995	0.993	1.0	1.0
4	0.997	0.996	1.0	1.0
5	0.994	0.993	0.999	0.999
6	0.995	0.995	0.999	0.999
7	0.995	0.995	0.999	0.999
8	0.995	1.0	1.0	1.0
9	0.996	1.0	1.0	1.0
10	0.996	1.0	1.0	1.0

Berdasarkan Tabel 4.4, 4.5, dan 4.6 dibawah ini merupakan hasil validasi kfolds machine learning menggunakan parameter terhadap data train. Machine learning Bagging mendapatkan nilai accuracy tertinggi yaitu 0.994, untuk nilai recall tertinggi yaitu 0.998, untuk nilai precission tertinngi sebesar 0.993, dan nilai f1-score tertinggi yaitu 0.996. Untuk machine learning Boosting berhasil mendapatkan score accuracy tertinggi sebesar 0.997, untuk precission tertingi yaitu dengan nilai 0.996, sedangkan untuk score tertinggi pada recall sebesar 0.997, dan untuk F1-score nilai tertinggi sebesar 0.998. Lalu untuk machine learning Stacking nilai accuracy tertinggi sebesar 1.0, recall tertinggi 1.0, precission 1.0, dan F1-score 1.0.

Tabel 4.7: Score rata rata k fold 80% training tidak menggunakan parameter(tuning)

Rata-rata score Kfold tidak menggunakan parameter(tuning) dengan data train 80%				
Machine Learning	Accuracy	Recall	Precission	F1-score
Bagging	0.992	0.997	0.992	0.995
Boosting	0.990	0.998	0.990	0.994
Stacking	0.999	1.0	0.999	0.999

Tabel 4.8: Score rata rata k fold 80% training menggunakan parameter(tuning)

Rata-rata score Kfold menggunakan parameter(tuning) dengan data train 80%				
Machine Learning	Accuracy	Recall	Precission	F1-score
Bagging	0.993	0.992	0.998	0.995
Boosting	0.995	0.995	0.999	0.997
Stacking	0.999	1.0	0.999	0.999

Berdasarkan Tabel 4.7 dan Tabel 4.8 merupakan hasil score rata rata yang didapat berdasarkan kfold terhadap 80% data train dari dataset peprobe yang terinfeksi malware dan tidak terinfeksi malware. Rata rata nilai score berdasarkan perhitungan matrix dari setiap machine learning dengan menggunakan rumus accuracy, recall, precession, dan f1-score. Untuk mendapatkan hasil berdasarkan tabel diatas, berikut langkah-langkah yang digunakan untuk mendapatkan hasil berdasarkan tabel diatas.

Data Preprocessing

langkah pertama dalam data preprocessing adalah dengan cara menginputkan dataset yang akan dipelajari, setelah menginputkan dataset yang akan di pelajari langkah berikutnya adalah melakukan normalisasi data menggunakan standar scaler agar tidak ada penyimpangan nilai data yang besar

Feature Selection

feature selection digunakan untuk mengoptimalkan feature yang paling berpengaruh dan menghilangkan feature yang tidak berpengaruh yang akan digunakan dan dipelajari oleh model machine learning dengan menggunakan principal component analysis (PCA). hasil dari Principal component analysis menghasilkan 35 feature yang digunakan oleh machine learning dari total jumlah 49 feature.

Klasifikasi

Untuk mendapatkan nilai akurasi seperti pada tabel diatas, digunakan 3 model machine learning ensemble yaitu Bagging, Boosting, dan Stacking de-

Tabel 4.9: Parameter(tuning) yang digunakan pada model algoritma machine learning

Machine learning	Parameter(Tuning)
Bagging	Random State = 0, N_estimator = 100, Max_Depth = 16, Max_Features = sqrt
Boosting	Random State = 0, N_estimator = 100, Max_Depth = 16, Max_Features = sqrt
Stacking	Random State = 0, N_estimator = 100, Max_Depth = 16, Max_Features = sqrt, Verbose = 10

ngan membandingkan algoritma yang menggunakan parameter(tuning) dan tidak menggunakan parameter(tuning). Parameter yang digunakan pada machine learning berdasarkan Tabel 4.9

4.1.2 Hasil Pencapaian Metodologi Objektif Kedua

Algoritma pengujian deteski pe-probe yang diusulkan bertujuan agar model machine learning dapat mendeteksi dan memprediksi pe-probe yang terinfeksi malware atau tidak terfinfeksi malware berdasarkan data test yang akan di uji dan menentukan model machine learning yang paling optimal dalam melakukan deteksi. Hasil dari algoritma prediski dan deteksi pe-probe yang dapat dilihat sebagai berikut :

Tabel 4.10: Hasil deteksi malware data test machine learning Bagging tidak menggunakan parameter

Machine learning Bagging tidak menggunakan parameter(tuning) dengan data test 20%				
Fold	Accuracy	Recall	Precission	F1-score
1	0.984	0.989	0.989	0.989
2	0.974	0.987	0.981	0.984
3	0.974	0.993	0.973	0.983
4	0.975	0.993	0.980	0.986
5	0.974	0.993	0.973	0.983
6	0.987	1.0	0.983	0.991
7	0.979	0.986	0.986	0.989
8	0.966	0.979	0.976	0.977
9	0.979	0.985	0.985	0.985
10	0.959	0.979	0.966	0.972

Tabel 4.11: Hasil deteksi malware data test machine learning Boosting tidak menggunakan parameter

Machine learning Boosting tidak menggunakan parameter(tuning) dengan data test 20%				
Fold	Accuracy	Recall	Precission	F1-score
1	0.974	0.982	0.982	0.982
2	0.974	0.987	0.981	0.984
3	0.977	0.996	0.974	0.985
4	0.974	0.993	0.973	0.983
5	0.969	0.993	0.966	0.979
6	0.984	1.0	0.980	0.989
7	0.989	0.996	0.989	0.993
8	0.969	0.976	0.982	0.979
9	0.977	0.989	0.979	0.984
10	0.961	0.975	0.972	0.974

Tabel 4.12: Hasil deteksi malware data test machine learning Stacking tidak menggunakan parameter

Machine learning Stacking tidak menggunakan parameter(tuning) dengan data test 20%				
Fold	Accuracy	Recall	Precission	F1-score
1	0.979	0.993	0.980	0.986
2	0.977	1.0	0.968	0.984
3	0.977	0.996	0.973	0.984
4	0.969	0.992	0.974	0.983
5	0.971	0.982	0.976	0.979
6	0.974	0.983	0.980	0.981
7	0.982	0.982	0.982	0.982
8	0.979	0.996	0.979	0.987
9	0.964	0.986	0.986	0.975
10	0.969	0.989	0.962	0.975

Berdasarkan Tabel 4.10, 4.11, dan 4.12 merupakan hasil deteksi malware machine learning tidak menggunakan parameter terhadap data test. Machine learning Bagging mendapatkan nilai accuracy tertinggi yaitu 0.987, untuk nilai recall tertinggi yaitu 1.0, untuk nilai precission tertinngi sebesar 0.989, dan nilai f1-score tertinggi yaitu 0.991. Untuk machine learning Boosting berhasil mendapatkan score accuracy tertinggi sebesar 0.989, untuk precission tertingi yaitu dengan nilai 0.989, sedangkan untuk score tertinggi pada recall sebesar 1.0, dan untuk F1-score nilai tertinggi sebesar 0.993. Lalu untuk machine learning Stacking nilai accuracy tertinggi sebesar 0.979, recall tertinggi 1.0, precission 0.986, dan F1-score 0.987. Pada tahap selanjutnya dilakukan deteksi malware menggunakan machine learning denga menggunakan parameter(tuning), betikut tabel score hasil deteksi malware berdasarkan machine learning dengan menggunakan parameter (tuning).

Tabel 4.13: Hasil deteksi malware data test machine learning Bagging menggunakan parameter

Machine learning Bagging menggunakan parameter(tuning) dengan data test 20%				
Fold	Accuracy	Recall	Precission	F1-score
1	0.989	0.992	0.992	0.992
2	0.974	0.987	0.981	0.984
3	0.977	0.993	0.977	0.985
4	0.977	0.989	0.989	0.984
5	0.974	0.993	0.976	0.983
6	0.984	1.0	0.998	0.989
7	0.982	0.986	0.989	0.988
8	0.966	0.979	0.976	0.977
9	0.982	0.982	0.992	0.987
10	0.961	0.979	0.969	0.974

Tabel 4.14: Hasil deteksi malware data test machine learning Boosting menggunakan parameter

Machine learning Boosting menggunakan parameter(tuning) dengan data test 20%				
Fold	Accuracy	Recall	Precission	F1-score
1	0.989	0.992	0.992	0.992
2	0.977	0.987	0.984	0.985
3	0.974	0.993	0.973	0.983
4	0.979	0.993	0.980	0.986
5	0.969	0.989	0.970	0.979
6	0.982	0.996	0.980	0.988
7	0.984	0.989	0.989	0.989
8	0.964	0.976	0.976	0.976
9	0.974	0.978	0.985	0.982
10	0.966	0.982	0.972	0.977

Tabel 4.15: Hasil deteksi malware data test machine learning Stacking menggunakan parameter

Machine learning Stacking menggunakan parameter(tuning) dengan data test 20%				
Fold	Accuracy	Recall	Precission	F1-score
1	0.989	0.992	0.992	0.992
2	0.977	0.990	0.981	0.985
3	0.977	0.996	0.974	0.985
4	0.982	0.996	0.980	0.988
5	0.971	0.989	0.973	0.981
6	0.984	1.0	0.980	0.989
7	0.987	0.993	0.989	0.991
8	0.966	0.979	0.976	0.977
9	0.982	0.982	0.992	0.987
10	0.966	0.982	0.972	0.977

Berdasarkan Tabel 4.13, 4.14, dan 4.15 merupakan hasil deteksi malware machine learning menggunakan parameter terhadap data test. Machine learning Bagging mendapatkan nilai accuracy tertinggi yaitu 0.989, untuk nilai recall tertinggi yaitu 1.0, untuk nilai precission tertinngi sebesar 0.998, dan nilai f1-score tertinggi yaitu 0.992. Untuk machine learning Boosting berhasil mendapatkan score accuracy tertinggi sebesar 0.989, untuk precission tertingi yaitu dengan nilai 0.992, sedangkan untuk score tertinggi pada recall sebesar 0.992, dan untuk F1-score nilai tertinggi sebesar 0.992. Lalu untuk machine learning Stacking nilai accuracy tertinggi sebesar 0.989, recall tertinggi 1.0, precission 0.992, dan F1-score 0.992. Pada tahap selanjutnya adalah menghitung nilai score rata rata accuracy berdasarkan nilai matrix, berikut nilai score accuracy rata-rata pada setiap machine learning.

Tabel 4.16: Score rata rata deteksi malware 20% data test tidak menggunakan parameter(tuning)

Rata-Rata Machine learning tidak menggunakan parameter(tuning) dengan data test 20%				
Machine Learning	Accuracy	Recall	Precission	F1-score
Bagging	0.976	0.988	0.979	0.984
Boosting	0.975	0.989	0.978	0.983
Stacking	0.977	0.990	0.976	0.983

Tabel 4.17: Score rata rata deteksi malware 20% data test menggunakan parameter(tuning)

Rata-Rata Machine learning menggunakan parameter(tuning) dengan data test 20%				
Machine Learning	Accuracy	Recall	Precission	F1-score
Bagging	0.977	0.988	0.981	0.984
Boosting	0.976	0.988	0.980	0.984
Stacking	0.978	0.990	0.981	0.985

Berdasarkan Tabel 4.16 dan Tabel 4.17 merupakan hasil score rata-rata deteksi malware terhadap 20% data test dari dataset peprobe yang terinfeksi malware dan tidak terinfeksi malware. Agar mendapatkan hasil berdasarkan pada tabel 4.16 dan 4.17 digunakan rumus accuracy, recall, precission, dan f1-score berdasarkan hasil perhitungan confusion matrix pada setiap model machine learning. Berikut hasil nilai confusion matrix pada setiap model machine learning:

Machine Learning	TP	TN	FP	FN
Bagging	2914	913	61	35
Boosting	2917	909	65	32
Stacking	2920	918	56	29

Tabel 4.18: Hasil deteksi malware confusion matrix berdasarkan machine learning tanpa parameter(tuning)

Machine Learning	TP	TN	FP	FN
Bagging	2914	914	60	35
Boosting	2914	916	58	35
Stacking	2921	918	56	28

Tabel 4.19: Hasil deteksi malware confusion matrix berdasarkan machine learning menggunakan parameter(tuning)

Berdasarkan Tabel 4.18 dan Tabel 4.19 didapatkan nilai confusion matrix hasil prediksi malware dari model algoritma machine learning yang dibuat. Dapat dilihat terdapat penuruan dan kenaikan pada nilai confusion matrix TP, TN, FP, dan FN pada setiap prediksi model machine learning yang menggunakan parameter(tuning) dan tidak menggunakan parameter(tuning).

Pada machine learning bagging mengalami kenaikan nilai TN menjadi 914 di machine learning yang menggunakan parameter(tuning), yang tadinya bernilai 913 pada machine learning tanpa parameter(tuning), yang dimana TN(True Negative) untuk mendeteksi file bukan malware. Untuk nilai FP mengalami penurunan pada machine learning yang menggunakan parameter(tuning) menjadi 60 yang tadinya bernilai 61 pada machine learning tanpa parameter(tuning), yang dimana FP(False Positive) mendeteksi file bukan malware namun dianggap malware. Nilai TN dan TP berpengaruh pada hasil nilai Precission , Recall dan FAR seperti pada Tabel 4.1 dan Tabel 4.2.

Nilai TP pada machine learning boosting yang menggunakan parameter(tuning) mengalami penurunan menjadi 2914 yang sebelumnya bernilai 2917 pada machine learning yang tidak menggunakan parameter(tuning). Nilai TN juga mengalami peningkatan pada machine learning yang menggunakan parameter(tuning) sebesar 916 yang tadinya bernilai 909 di machine learning yang tidak menggunakan parameter(tuning). Untuk nilai FP dan penurunan pada machine learning yang menggunakan parameter menjadi 58 di machine learnin tanpa paremeter(tuning). Sedangkan untuk nilai FN mengalami peningkatan pada machine learning menggunakan parameter dengan nilai 35 yang tadinya bernilai 32 pada machine learning tanpa parameter.

Sedangkan pada machine learning stacking mengalami peningkatan pada nilai TP sebesar 2921 pada machine learning yang menggunakan parameter(tuning), yang sebelumnya bernilai 2920 pada machine learning yang tidak menggunakan parameter(tuning). Sedangkan untuk nilai FN mengalami penurunan pada machine learning yang menggunakan parameter dengan nilai 28 yang sebelumnya 29 pada machine learnin tanpa parameter(tuning). Nilai TP dan FN berpengaruh untuk menghitung nilai accuracy, precission, dan detection rate seperti pada Tabel 4.1 dan Tabel 4.2. Semakin meningkatnya nilai TP dan TN maka algoritma yang dibuat akan semakin baik dalam mendeteksi dan klasifikasi malware, dan semakin kecil nilai FP dan FN maka algortima yang dibuat akan menjadi lebih baik karena mengurangi kesalahan dalam mendeteksi dan klasifikasi malware.

4.1.3 Hasil Pencapaian Metodologi Objektif Ketiga

Algoritma pengujian diusulkan bertujuan agar prototype yang dibuat dapat menampilkan hasil prediksi validasi pe-prpobe yang telah dimplementasikan di algortima machine learning yang dibuat. Hasil algortima pengujian impelementasi pada prototype sebagai berikut.

Nama File	Nilai Prediksi
Whatsappsetup.exe	11.3
Steamapi64.dll	15.9
Glib-2.dll	31.9
Obs-Studio-27.2.4.exe	1.53
Icudt71.dll	46.7
GModule2.dll	26.9
GithubDesktopSetup.exe	6.74
Overwatch Launcher.exe	25.1
Git-2.35.1.2-64bit.exe	54.5
Battlenet.exe	28.9

Tabel 4.20: Hasil pengujian deteksi pe-probe yang tidak terinfeksi malware.

Berdasarkan Tabel 4.20 terdapat kesalahan prediksi pada pengujian pe-probe yang tidak terinfeksi malware. Kesalahan terjadi pada nama file Git-2.35.1.2-64bit.exe yang dimana jika tidak terinfeksi malware nilai prediksi probabilty seharusnya di bawah 50.

Berdasarkan Tabel 4.21 dibawah ini terdapat kesalahan pada pengujian pe-probe yang terinfeksi malware. Kesalahan terjadi pada nama file Wildfire-test-pe-file.exe yang dimana jika terinfeksi malware nilai prediksi probabilty seharusnya di atas 50.

Nama File	Nilai Prediksi
00d1.exe	99.1
00d6.exe	98.8
00de7	99.1
LojaxSmallAgent.exe	96.4
00a0.exe	99.1
Tsetup-x64.3.5.1.2.exe	54.5
Wildfire-test-pe-file.exe	20.1
00e0.exe	99.1
00db.exe	99.1
00d94	99.1

Tabel 4.21: Hasil pengujian deteksi pe-probe yang terinfeksi malware .

Untuk mendapatkan score predict probabilty berdasarkan Tabel 4.21 dan Tabel 4.22, dibuatlah algoritma sebagai berikut

Feature Extraction

Berdasarkan Tabel 4.22 dibawah ini merupakan algoritma untuk melakukan feature extraction terhadap validasi file pe-probe yang akan diuji, pe-probe yang akan diuji berformat .exe dan .dll. Setelah melakukan feature extracion berdasarkan Tabel 4.22 langkah berikutnya adalah menyimpan hasil feature extraction kedalam array, yang nantinya akan di prediksi berdasarkan model terbaik machine learning yang telah dibuat.

Tabel 4.22: feature extraction terhadap validasi pe-probe yang akan di uji

No.	Feature	Decsription
1	e_magic	Magic number
2	e_cblp	Bytes on last page of file
3	e_cp	Pages in file
4	e_crlc	Relocations
5	e_cparhdr	Size of header in paragraphs
6	e_minalloc	Minimum extra paragraphs needed
7	e_maxalloc	Maximum extra paragraphs needed
8	e_ss	Initial (relative) SS value
9	e_sp	Initial SP value
10	e_csum	Checksum
11	e_ip	Initial IP value
12	e_cs	Initial (relative) CS value
13	e_lfarlc	File address of relocation table
14	e_ovno	Overlay number
15	e_oemid	OEM identifier (for e_oeminfo)
16	e_oeminfo	OEM information; e_oemid specific
17	e_lfanew	File address of new exe header
18	Machine	This is a number that indicates the type of machine (CPU Architecture).
19	NumberOfSections	This field holds the number of sections
20	PointerToSymbolTable	hold the file offset to the COFF symbol table and the number of entries in that symbol table
21	NumberOfSymbol	hold the file offset to the COFF symbol table and the number of entries in that symbol table
22	SizeOfOptionalHeader	The size of the optional header, which is required for executable files but not for object files
23	Characteristics	The flags that indicate the attributes of the file.
24	Magic	The unsigned integer that identifies the state of the image fil.
25	MajorLinkerVersion	The linker major version number
26	MinorLinkerVersion	The linker minor version number
27	SizeOfCode	The size of the code (text) section, or the sum of all code sections if there are multiple sections.
28	SizeOfInitializedData	The size of the initialized data section, or the sum of all such sections . if there are multiple data sections

29	SizeOfUninitializedData	The size of the uninitialized data section (BSS), or the sum of all such sections . if there are multiple BSS sections.
30	AddressOfEntryPoint	The address of the entry point relative to the image base when the executable file is loaded into memory.
31	BaseOfCode	The address that is relative to the image base of the beginning-of-code section when it is loaded into memory.
31	ImageBase	The preferred address of the first byte of image when loaded into memory; must be a multiple of 64 K.
32	SectionAlignment	This field holds the preferred address of the first byte of image when loaded into memory (the preferred base address), this value must be a multiple of 64K.
33	FileAlignment	The alignment factor (in bytes) that is used to align the raw data of sections in the image file. The value should be a power of 2 between 512 and 64 K, inclusive. The default is 512.
34	MajorOperating SystemVersion	The major version number of the required operating system.
35	MinorOperating SystemVersion	The minor version number of the required operating system.
35	MajorImageVersion	The major version number of the image.
36	MinorImageVersion	The minor version number of the image.
37	MajorSubsystemVersion	The major version number of the subsystem.
38	MinorSubsystemVersion	The minor version number of the subsystem.
39	SizeOfHeaders	The combined size of an MS-DOS stub, PE header, and section headers rounded up to a multiple of FileAlignment.
40	CheckSum	A checksum of the image file, it's used to validate the image at load time..
41	SizeOfImage	The size (in bytes) of the image, including all headers, as the image is loaded in memory. It must be a multiple of SectionAlignment.

42	Subsystem	The subsystem that is required to run this image.
43	DllCharacteristics	his field defines some characteristics of the executable image file, like if it's NX compatible and if it can be relocated at run time.
44	SizeOfStackReserve	The size of the stack to reserve. Only SizeOfStackCommit is committed; the rest is made available one page at a time until the reserve size is reached.
45	SizeOfStackCommit	The size of the stack to commit.
46	SizeOfHeapReserve	The size of the local heap space to reserve. Only SizeOfHeapCommit is committed; the rest is made available one page at a time until the reserve size is reached
47	SizeOfHeapCommit	The size of the local heap space to commit.
48	LoaderFlags	Reserved, must be zero.
49	NumberOfRvaAndSizes	The number of data-directory entries in the remainder of the optional header. Each describes a location and size.

Prediksi

Berdasarkan Algorithm 1 merupakan algoritma untuk memprediksi berdasarkan hasil feature extraction pe-probe yang telah dibuat sebelumnya, dengan cara melakukan load file.pkl model machine learning terbaik yang telah dilatih untuk mendeteksi malware.

Algorithm 1 Pseudecode algoritma prediksi

```
1: def getPredictions(df):
2:     load_scaler = joblib.load(open(r"../components/boostingscalerTP.pkl","rb"))
3:     load_skpca = joblib.load(open(r"../components/boostingpcaTP.pkl","rb"))
4:     load_model = joblib.load(open(r"../components/boostingrf_modelTP.pkl","rb")
5:     pipe = Pipeline([("scale",load_scaler),("pca",load_skpca),("rf_model",load_model)])
6:     df = np.array(df)
7:     df = df.reshape(1,-1)
8:     results = pipe.predict_proba(df)
9:     pred = pipe.predict(df)
10:    return (results[0],pred[0])
```

Algortima Get dan Post

Berdasarkan pada Algorithm 2 terdapat fungsi get dan post, dimana fungsi get berfungsi untuk menampilkan seluruh tampilan. Untuk fungsi post ber-

fungsi untuk menampilkan hasil prediksi pe probe dari implementasi machine learning yang dibuat.

Algorithm 2 Pseudecode algoritma get dan post

```
app = FastAPI(description = app_desc)
templates = Jinja2Templates(directory="html")
@app.get("/",include_in_schema=False)
def index():
    return RedirectResponse(url="/docs")
@app.post("/predict")
def parse(file: UploadFile = File(...)):
     extension = os.path.splitext(file.filename)[1]
     _, path = tempfile.mkstemp(prefix='parser_', suffix=extension)
     with open(path, 'ab') as f:
    for chunk in iter(lambda: file.file.read(10000), b"):
            .write(chunk)
```

Algoritma Untuk Menampilkan Prediksi Pengujian Validasi Pe-probe

Berdasarkan Algorithm 3 merupakan algoritma untuk menampilkan hasil pengujian validasi pe-probe seperti pada 4.20 dan 4.21 diatas, keluaran hasil pengujian dari hasil prediksi pe probe yang dimana jika nilai prediksi adalah kelas malware prototype akan menampilkan hasil julmah prediksi probability di atas 50. Sedangkan jika nilai prediksi adalah kelas bukan malware maka prototype akan menampilkan hasil jumlah nilai prediksi probability di bawah 50.

Algorithm 3 Pseudecode algoritma untuk menampilkan prediksi validasi pe probe

```
content = pefile.PE(path,fast_load=True)
dataframe = createDataframeFromPEdump(content)
binary_preds = getPredictions(dataframe)
if binary_preds[1] == 1:
  return 'Namefile':file.filename,'response':'Maliciousfile', 'predictions':binary_preds[0][1]*100
else:
  return 'Name file' : file.filename, 'Response': 'NotMaliciousFile','prediction':binary_preds[0][1]*100
```

4.2 Pembahasan

Machine Learning	Accuracy	Recall	Precission	F1-Score	DR	FAR
Bagging	0.976	0.988	0.979	0.984	0.9881	0.062
Boosting	0.975	0.989	0.978	0.983	0.9891	0.066
Stacking	0.977	0.990	0.976	0.983	0.9901	0.057

Tabel 4.23: Hasil Algoritma Machine Learning klasifikasi dan deteksi malware tanpa parameter(tuning)

Machine Learning	Accuracy	Recall	Precission	F1-Score	DR	FAR
Bagging	0.976	0.988	0.981	0.984	0.9881	0.061
Boosting	0.976	0.988	0.980	0.984	0.9888	0.066
Stacking	0.978	0.990	0.981	0.985	0.9905	0.057

Tabel 4.24: Hasil Algoritma Machine Learning klasifikasi dan deteksi malware menggunakan parameter(tuning)

Machine learning stacking, bagging, dan boosting adalah teknik-teknik ensemble learning dalam machine learning. Ensemble learning adalah sebuah metode yang menggabungkan beberapa model machine learning yang berbeda untuk meningkatkan kinerja prediksi. Meskipun ada beberapa kesamaan antara ketiga machine learning ini, namun mereka memiliki perbedaan utama dalam cara bagaimana model-model individual digabungkan untuk membuat prediksi akhir.

Berikut adalah beberapa keunggulan machine learning stacking dibandingkan dengan machine learning bagging dan boosting:

1. machine learning stacking seringkali menghasilkan model yang lebih akurat dibandingkan machine learning bagging dan boosting. Hal ini disebabkan karena machine learning stacking dapat menggabungkan berbagai jenis model machine learning yang berbeda dan menggabungkan kelebihan dari masing-masing model untuk menghasilkan prediksi akhir yang lebih akurat.

2. Machine learning stacking lebih fleksibel dibandingkan machine learning bagging dan boosting karena dapat menggabungkan berbagai jenis model machine learning, termasuk model yang kompleks seperti neural network dan deep learning. Selain itu, dengan menggabungkan berbagai jenis model, machine learning stacking dapat meningkatkan kemampuan model untuk menangani data yang kompleks dan beragam.

3. machine learning stacking dapat menyesuaikan diri dengan data yang berubah-ubah, sehingga mampu menghasilkan prediksi yang lebih akurat pada data yang berubah-ubah tersebut. Sementara itu, machine learning bagging dan boosting cenderung menghasilkan model yang lebih statis dan sulit beradaptasi dengan data yang berubah-ubah.

Namun, meskipun machine learning stacking memiliki beberapa keunggulan, machine learning staking juga memiliki kelemahan. Salah satu kelemahan utama dari mmachine learning stacking adalah kompleksitasnya yang lebih tinggi dibandingkan machine learning bagging dan boosting. Machine learning stacking memerlukan lebih banyak waktu dan sumber daya untuk menggabungkan berbagai jenis model dan membangun model ensemble yang akurat.

Pada penelitian ini machine learning stacking dengan paraemeter(tuning) lebih unggul dibandingkan dengan machine learning boosting dan machine learning bagging yang menggunakan parameter(tuning) dan tidak menggunakan parameter(tuning). Keunggulan tersebut berdasarkan hasil percobaan ketiga model machine learning dengan menggunakan paramater(tuning) dan tidak menggunakan parameter(tuning). Berdasarkan Tabel 4.23 dan Tabel 4.24 machine learning Stacking dengan parameter(tuning) menghasilkan nilai akurasi tertinggi yaitu sebesar 0.9785, detection rate tertinggi yaitu 0.9905, dan false alarm rate sebesar 0.057 dibandingkan model machine learning lain, sehingga machine learning stacking diimplentasikan pada prototype untuk menguji validasi pe-probe yang terinfeksi malware dan tidak terinfeksi malware. Hasil nilai accuracy, detection rate, dan false alarm rate didapatkan berdasrkan hasil uji matrix. Dalam penelitian sebelumnya oleh Sanjeev kumar (2022) bahwa machine learning stacking dapat meningkatkan akurasi klasifikasi dalam mendeteksi jenis malware, dibandingkan dengan penggunaan algoritma klasifikasi tunggal, Selain itu Vijayakumar Ramalingam dan Dr. R. Saravanan (2017) menjelaskan bahwa detection rate yang tinggi sangat penting dalam deteksi malware dan membantu mencegah penyebaran malware.

4.3 Ringkasan

Ada beberapa hal yang perlu menjadi catatan dari Bab ini sebagai berikut:

1. Untuk mendapatkan nilai prediksi deteksi malware dengan cara feature extraction file pe probe yang sudah diupload sistem dan membandingan hasil klasifikasi malware yang telah dipelajari oleh machine learning

2. Algoritma machine learning yang digunakan menggunakan 2 teknik yaitu menggunakan parameter dan tidak menggunakan parameter

3. Algoritma yang digunakan pada prototype adalah algoritma machine learning stacking

Bab V

Kesimpulan dan Saran

5.1 Kesimpulan

Pada penelitian kali ini penulis menyajikan sebuah sistem deteksi malware berdasarkan fitur-fitur pe-probe dengan menggunakan machine learning ensemble. Sistem ini mampu mendeteksi dan membedakan pe-probe yang terinfeksi malware dan tidak terinfeksi malware, Machine learning yang digunakan untuk membangun sistem deteksi malwware yaitu machine learning Bagging, Boosting, dan Stacking. Penulis juga menemukan kekurangan dalam sistem yang dibangun, yaitu ketika mendeteksi validasi pe-probe tidak berdasarkan fitur-fitur yang telah dipelajari oleh machine learning, sehingga kesalahan dalam deteksi malware dan bukan malware akan semakin besar. Berdasarkan hasil penelitian, machine learning Stacking menjadi machine learning yang paling optimal dalam melakukan deteksi malware berdasarkan nilai accuracy 0.978, detection rate 0.9905, dan false alarm rate 0.0057 mendapatkan nilai paling besar dibandingkan machine learning yang lain seperti Boosting dan Bagging.

5.2 Saran

Berdasarkan proses perancangan dan pengujian sistem, penulis melihat beberapa pengembangan rancangan dan langkah pengujian yang dapat dilakukan, antara lain:

1. Memilih fitur dan klasifikasi lain untuk meningkatkan kehandalan akurasi deteksi dan klasifikasi
2. Lebih mendalam untuk mempelajari struktur dari pe-probe
3. Lebih mempelajari cara kerja malware dalam menginfeksi pe-probe
4. Membuat sistem yang dapat mengklasifikasikan jenis family malware

Daftar Pustaka

Abijah Roseline, S., Geetha, S., Kadry, S. and Nam, Y. (2020), 'Intelligent Vision-based Malware Detection and Classification using Deep Random Forest Paradigm', *IEEE Access* pp. 1–1.

Chen, C. W., Su, C. H., Lee, K. W. and Bair, P. H. (2020), 'Malware Family Classification using Active Learning by Learning', *International Conference on Advanced Communication Technology, ICACT* **2020**, 590–595.

Chumachenko, K. (2017), 'Machine Learning Methods for Malware Detection and Classification', *Proceedings of the 21st Pan-Hellenic Conference on Informatics - PCI 2017* p. 93.

Das, S., Liu, Y., Zhang, W. and Chandramohan, M. (2016), 'Semantics-based online malware detection: Towards efficient real-time protection against malware', *IEEE Transactions on Information Forensics and Security* **11**(2), 289–302.

Han, X., Jin, F., Wang, R., Wang, S. and Yuan, Y. (2021), 'Classification of malware for self-driving systems', *Neurocomputing* **428**, 352–360.

Kim, D., Woo, S., Lee, D. and Chung, T. (2016), Static detection of malware and benign executable using machine learning algorithm, *in* 'INTERNET 2016: The Eighth International Conference on Evolving Internet', pp. 14–19.

Liu, L., sheng Wang, B., Yu, B. and xi Zhong, Q. (2017), 'Automatic malware classification and new malware detection using machine learning', *Frontiers of Information Technology and Electronic Engineering* **18**(9), 1336–1347.

Liu, Y. S., Lai, Y. K., Wang, Z. H. and Yan, H. B. (2019), 'A New Learning Approach to Malware Classification Using Discriminative Feature Extraction', *IEEE Access* **7**(c), 13015–13023.

Mangialardo, R. J. and Duarte, J. C. (2015), 'Integrating static and dynamic malware analysis using machine learning', *IEEE Latin America Transactions* **13**(9), 3080–3087.

Nataraj, L., Karthikeyan, S., Jacob, G. and Manjunath, B. S. (2011), Malware images: visualization and automatic classification, *in* 'Proceedings of the 8th international symposium on visualization for cyber security', pp. 1–7.

Oh, S., Go, W. and Lee, T. (2017), A study on the behavior-based malware detection signature, *in* 'Advances on Broad-Band Wireless Computing, Communication and Applications: Proceedings of the 11th International Conference On Broad-Band Wireless Computing, Communication and Applications (BWCCA–2016) November 5–7, 2016, Korea', Springer, pp. 663–670.

Pai, S., Troia, F. D., Visaggio, C. A., Austin, T. H. and Stamp, M. (2017), 'Clustering for malware classification', *Journal of Computer Virology and Hacking Techniques* **13**(2), 95–107.

Radwan, A. M. (2019), Machine learning techniques to detect maliciousness of portable executable files, *in* '2019 International Conference on Promising Electronic Technologies (ICPET)', IEEE, pp. 86–90.

Rezaei, T., Manavi, F. and Hamzeh, A. (2021), 'A pe header-based method for malware detection using clustering and deep embedding techniques', *Journal of Information Security and Applications* **60**, 102876.

Sahu, M. K., Ahirwar, M. and Shukla, P. K. (2015), Improved malware detection technique using ensemble based classifier and graph theory, *in* '2015 IEEE International Conference on Computational Intelligence & Communication Technology', IEEE, pp. 150–154.

Shafiq, M. Z., Tabish, S. and Farooq, M. (2009), Pe-probe: leveraging packer detection and structural information to detect malicious portable executables, *in* 'Proceedings of the Virus Bulletin Conference (VB)', Vol. 8.

Shafiq, M. Z., Tabish, S. M., Mirza, F. and Farooq, M. (2009), 'PE-Miner : Mining Structural Information to Detect Malicious Executables in Realtime Agenda Introduc1on to Domain Problem Defini1on Proposed Solu1on', pp. 121–141.
URL: *http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.663.7013&rep=rep1&type=*

Udayakumar, N., Saglani, V. J., Cupta, A. V. and Subbulakshmi, T. (2018), 'Malware Classification Using Machine Learning Algorithms', *Proceedings of the 2nd International Conference on Trends in Electronics and Informatics, ICOEI 2018* pp. 1007–1012.

Vyas, R., Luo, X., McFarland, N. and Justice, C. (2017), Investigation of malicious portable executable file detection on the network using supervised

learning techniques, *in* '2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)', IEEE, pp. 941–946.

Wuechner, T., Cisłak, A., Ochoa, M. and Pretschner, A. (2017), 'Leveraging compression-based graph mining for behavior-based malware detection', *IEEE Transactions on Dependable and Secure Computing* **16**(1), 99–112.

Xia, S., Pan, Z., Chen, Z., Bai, W. and Yang, H. (2018), Malware classification with markov transition field encoded images, *in* '2018 Eighth International Conference on Instrumentation & Measurement, Computer, Communication and Control (IMCCC)', IEEE, pp. 1–5.

Xue, D., Li, J., Lv, T., Wu, W. and Wang, J. (2019), 'Malware classification using probability scoring and machine learning', *IEEE Access* **7**, 91641–91656.

Lampiran A

Jadwal Kegiatan

The table 5.2 is an example of referenced LaTeXelements. Laporan proposal ini akan dijadwalkan sesuai dengan tabel yang diberikna berikutnya.

Tabel 5.1: Jadwal kegiatan proposal tugas akhir

No	Kegiatan	Bulan ke-																					
		1				2				3				4				5				6			
1	Studi Literatur																						
2	Pengumpulan Data																						
3	Analisis dan Perancangan Sistem																						
4	Implementasi Sistem																						
5	Analisa Hasil Implementasi																						
6	Penulisan Laporan																						

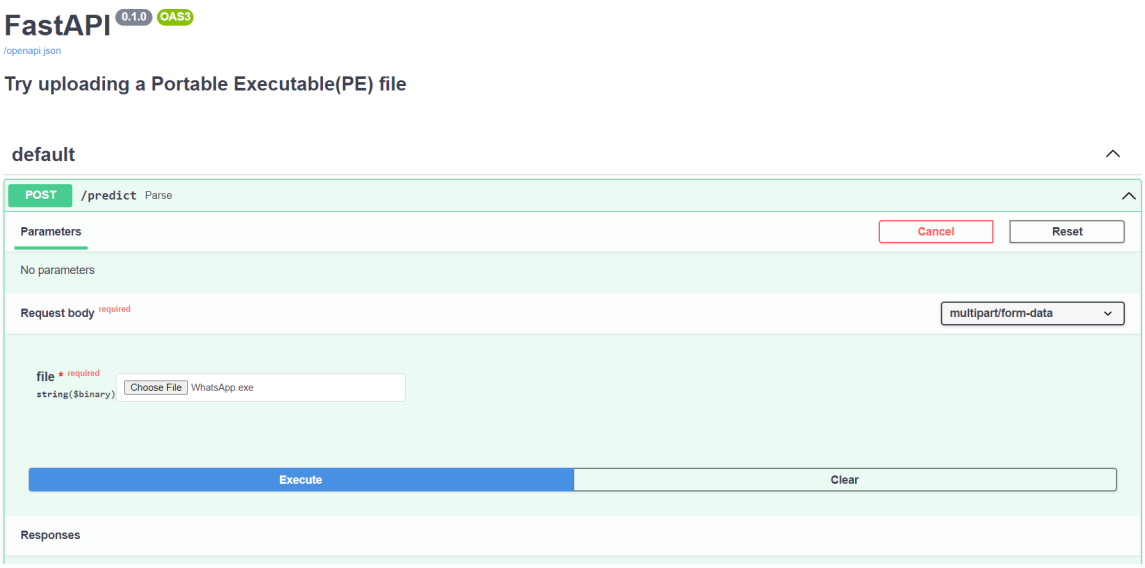
Lampiran B

Jadwal Kegiatan

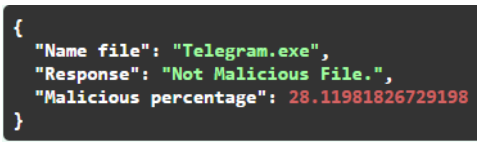
The table 5.2 is an example of referenced LaTeXelements. Laporan proposal ini akan dijadwalkan sesuai dengan tabel yang diberikna berikutnya.

No	Kegiatan	Bulan ke-																					
		1				2				3				4				5				6			
1	Studi Literatur																						
2	Pengumpulan Data																						
3	Analisis dan Perancangan Sistem																						
4	Implementasi Sistem																						
5	Analisa Hasil Implementasi																						
6	Penulisan Laporan																						

Tabel 5.2: Jadwal kegiatan proposal tugas akhir



Gambar 5.1: Tampilan saat akan menginputkan pe-probe yang diuji



Gambar 5.2: Tampilan saat mengeluarkan hasil pengujian pe-probe yang tidak terinfeksi malware

Gambar 5.3: Tampilan saat mengeluarkan hasil pengujian pe-probe yang terinfeksi malware