Studi Classifier Based 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

2020

Lembar Pengesahan

Studi Classifier Based 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, 29 Agustus 2022
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 sebuah perangkat lunak atau program berbahaya yang dapat menyebabkan kerusakan pada sitem operasi komputer dan jaringan. banyak penelitian yang telah melakukan pengembangan untuk mencegah serangan malware yang dapat menimbulkan banyak kerugian. Para peneliti telah mengembangkan metode deteksi dan klasifikasi malware dengan berbasis metode statis dan dinamis. namun metode statis dan dinamis banyak menuai pro dan kontra.metode statis mememiliki kekurangan dalam mendeteksi jenis file baru sedangkan metode dinamis memiliki kekurangan mengkonsumsi lebih banyak sumber daya dan juga memiliki biaya tinggi. Para peneliti juga telah mengembangkan berbagai macam teknik untuk mendeteksi malware berdasarkan pada fitur fitur pe-probe. Untuk menyelesaikan masalah-masalah di atas, tugas akhir ini mengusulkan pengembangan algoritma deteksi malware menggunakan metode berbasis feature-feature Pe-Probe(Pe-File) dengan menggunakan machine learning untuk meningkatkan akurasi deteksi malware dan klasifikasi. Metode yang digunakan dalam penelitian tugas akhir ini adalah 1. Studi literatur tentang deteksi dan klasifikasi malware, 2. Pengembangan algoritma klasifikasi untuk deteksi malware berbasis metode Pe-Probe(Pe-file), 3. Pengembangan prototype, 4. Pengujian performansi dan analisis. Penelitian ini menggunakan dataset sebanyak 19611 file pe probe yang terdiri dari data pe-probe terinfeksi malware dan data pe-probe tidak terinfeksi malware dan dengan nilai accuracy terhadap data train dan test sebesar 0.999 dan 0.978, dengan model terbaik hasil validasi menggunakan kfold cross validation yaitu machine learning stacking dengan menggunkan parameter.

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 Classifier Based 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 menjadi salah satu ancaman cyber utama saat ini. mulai dari perangkat lunak yang melakukan tindakan berbahaya, pencurian data dan informasi, dan spionase. tindakan tersebut dapat disebut malware. Kaspersky Labs (2017) mendefinisikan malware sebagai sejenis program komputer yang dirancang untuk menginfeksi komputer pengguna yang dan menimbulkan kerusakan di dalamnya dengan berbagai cara.

Perlindungan malware pada sistem komputer adalah salah satu tugas keamanan *cyber* yang penting bagi pengguna, karena satu serangan dapat mengakibatkan data yang disusupi dan mendaptakan kerugian yang cukup berdampak besar. saat ini popular metode dalam mendeteksi malware menggunakan metode klasifikasi statis dan dinamis yang berdasarkan algoritma yang mempelajari *signature based* dari malware Chumachenko (2017).

Namun metode statis dan dinamis banyak menuai pro dan kontra, Kelebihan teknik analisa statik adalah malware tidak kita jalankan, Sehingga mengurangi resiko sistem kita terinfeksi malware. Walaupun begitu terdapat banyak keterbatasan dari metode ini seperti tidak bisa mendeteksi jenis malware baru. Saat ini penulis malware banyak menggunakan teknik yang membuat proses analisa malware statik semakin sulit. Misalnya menggunakan packer untuk melakukan modifikasi kode secara otomatis.

Selain itu Analisis dinamis lebih efisien dan tidak perlu executable untuk dibongkar atau didekripsi. Namun, ini memakan waktu dan sumber daya yang intensif dan banyak (L. Nataraj, S. Karthikeyan, 2011)

(Sungtaek OH, Woong Go, and Taejin Lee, 2016) juga melakukan penelitian deteksi malware berbasis *signature based* dengan mengidentifikasi ciri-ciri malware yang berada di database, namun ketika terjadi serangan *zero-day* metode *tersebut* tidak sepenuhnya optimal dan banyak malware yang tidak terdeteksi.

Pada tahun (2017, Vyas dkk.) menyelidiki fitur statis untuk mengusulkan sistem deteksi malware jaringan waktu nyata. Fitur diekstraksi dari header

dan bagian file PE dan dikelompokkan kembali menjadi empat kategori. Metadata file, file pengepakan, DLL yang diimpor, dan fungsi yang diimpor. Ini fitur tersebut kemudian digunakan untuk melatih beberapa pembelajaran mesin pengklasifikasi.

Rezaei, Manavi and Hamzeh (2021) juga melakukan Penelitian deteksi Malware Berbasis Pe-Probe (Pe file) yang berdasarkan 9 feature dari pe header dengan tingkat akurasi 95,5% menggunakan *Random Forest*.

Dari permasalahan itu, pada penelitian ini berfokus pada pengembangan deteksi dan klasifikasi akurasi malware bebrbasis feature-feature yang berada didalam pe-probe(pe-file), dengan membuat sebuah prototype untuk deteksi malware dengan menggunakan *machine learning* untuk mendeteksi dan klasifikasi akurasi dari *PE-Probe*(pe file). Sehingga diharapkan penelitian ini menjadi metode yang 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 Pengembangan deteksi malware menggunakan machine learning berbasis *PE-Probe(Pe-File)* di prototype.

3. Membuat protoype untuk mendeteksi pe-probe untuk mendeteksi malware.

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 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,2020) 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%.

Dong Hee-Kim (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*

(R.J.Mangialardo,J. C.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 (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 (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%

Wuchner (2019) 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%.

Shiming Xia (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. (Horton and Seberry 1997).

2. **Worm**
   Worm merupakan jenis malware yang sangat mirip dengan virus, dapat melipat gandakan dirinya sendiri, melalui jaringan tanpa arahan dari pembuat malware. (Smith, et al. 2009)

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. (Chuvakin 2003).

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 (Chien 2005).

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. (Moffie, et al. 2006)

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. (Kateryna Chumachenko , 2017)

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 (Judith Hurwitz,Daniel Kirsch. 2018). 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 (DI XUE, 2019). *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.

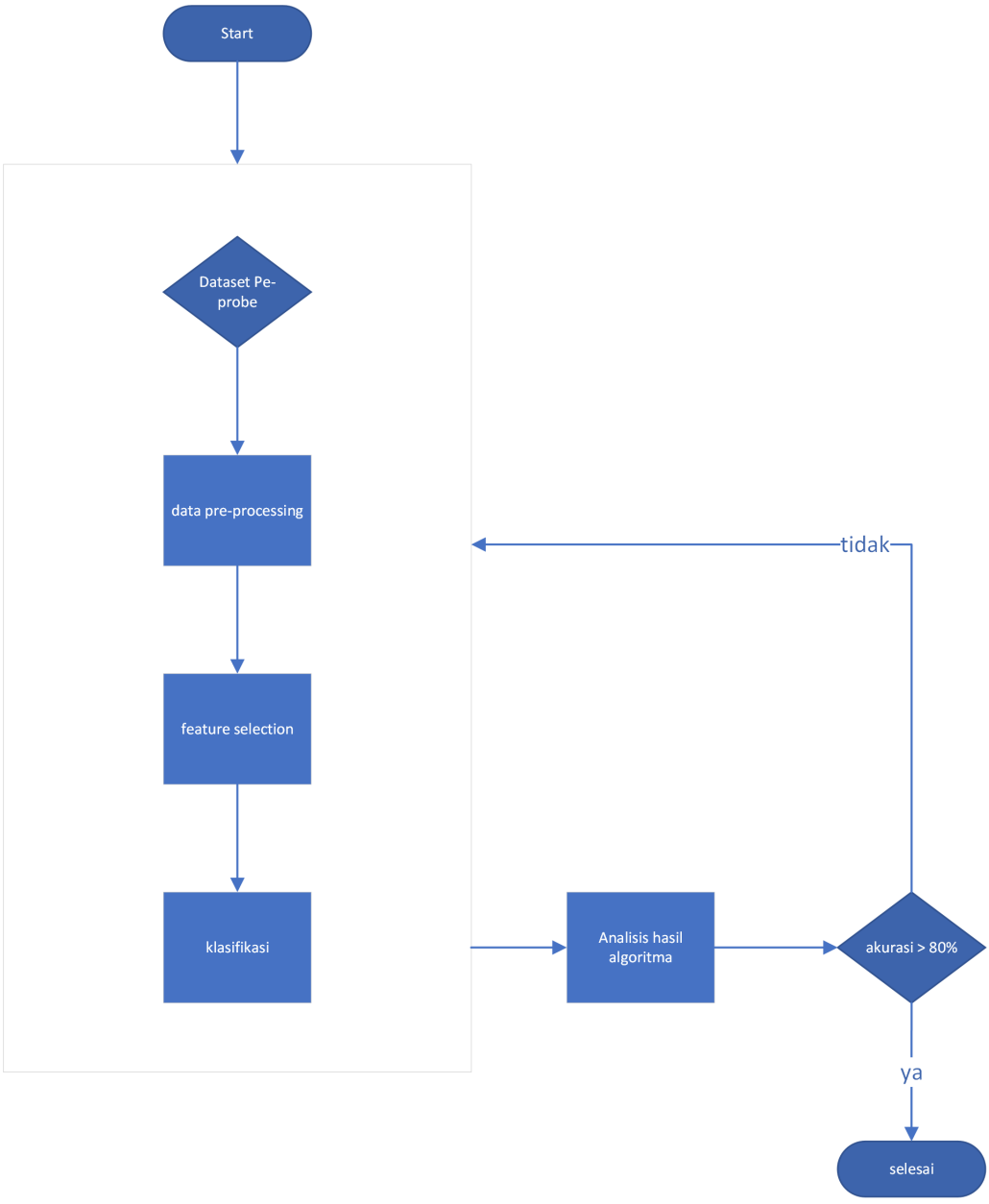
3. **Pembangunan Sistem**
   Pembangunan sistem dilakukan penulis untuk mendapatkan hasil analisis dari data yang didapatkan dari prototype mulai dari preprocessing, ekstraksi fitur, seleksi fitur dan klasifikasi.

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 :

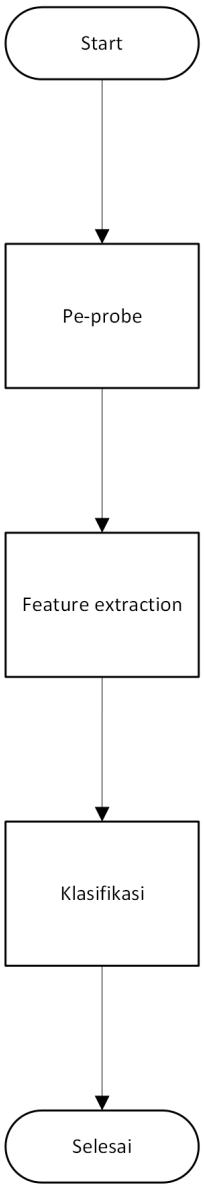
(a) **Dataset PE-Probe**
Dataset PE-Probe(Pe file) adalah sebuah dataset yang berisikan fitur-fitur data mengenai informasi sebuah aplikasi yang dapat di-jalankan atau dieksekusi oleh sistem operasi windows.

(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 :

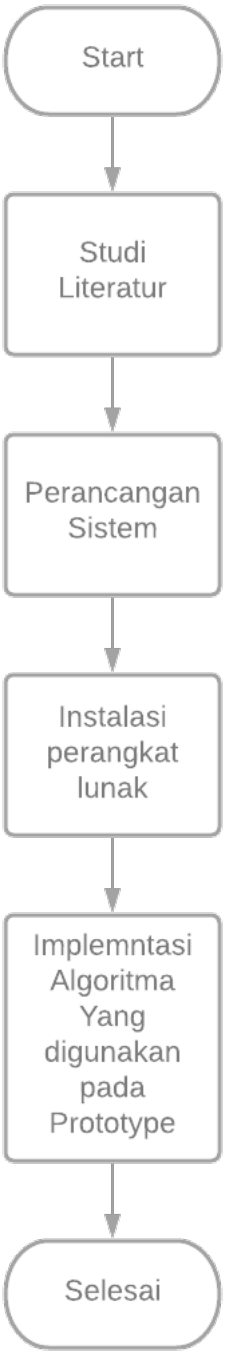
(a) **Pe-Probe**
Pe-Probe adalah file yang dapat di jalankan dan di eksekusi oleh sistem komputer dan berformat .exe dan .dll.

(b) **Feature Extarction**
Pada tahap ini algoritma membaca isi fitur-fitur dalam sebuah pe-probe(pe-file) yang nantinya akan dibandingkan dengan hasil data

feature yang sudah dipelajari oleh machine learning.

(c) **Klasifikasi dan Deteksi**
Pada tahap ini dilakukan klasifikasi dalam sebuah data test peprobe untuk mendeteksi file berisi malware atau tidak.

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 seperti yang diinginkan.

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 untuk mendeteksi file Malware, FN mendeteksi file bukan malware namun sebenarnya malware, FP mendeteksi file bukan malware namun dianggap malware, dan TN 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 hasil algortima Klasifikasi**
   Pada skenario ini dilakukan pengujian terhadap hasil algoritma deteksi dengan menganalisis hasil klasifikasi dan deteksi malware yang diharapkan mencapai nilai akurasi lebih dari 80%.

2. **Skenario 2 : Pengujian deteksi pe-pobe**
   Pada skenario ini, dilakukan pengujian deketeksi terhadap pe-probe, yang nantinya akan menghasilkan persentase seberapa banyak malware terhadap dalam satu file pe probe, dan mengklasifikasikan mengandung malware atau tidak yang terdapat dalam pe-probe.

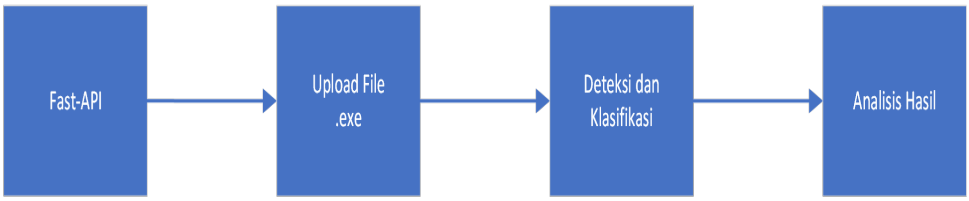
3. **Skenario 3 : Implementasi algoritma pada prototype**
   Pada skenario ini dilakukan implementasi hasil algoritma yang telah disiapkan untuk diterapkan pada sistem. hasil pada tahapan ini adalah sistem dapat menjalakan algoritma dengan baik memberikan hasil yang diinginkan.

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 Pengujian hasil algortima Klasifikasi

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 dan deteksi pe-probe yang dapat dilihat sebagai berikut :

Machine Learning	Accuracy	Precision	Recall	F1-score	Detection Rate	False Alarm Rate
Boosting	0.974	0.988	0.988	0.983	0.9911	0.051
Bagging + Random Forest	0.976	0.980	0.988	0.984	0.9935	0.037
Stacking + Random Forest + Logistic Regresion	0.977	0.980	0.990	0.985	0.9925	0.039

Gambar 4.1: Machine Learning tidak menggunakan Parameter terhadap test

Machine Learning	Accuracy	Precision	Recall	F1-score	Detection Rate	False Alarm Rate
Boosting	0.990	0.990	0.998	0.994	0.9911	0.0513
Bagging + Random Forest	0.992	0.992	0.997	0.995	0.9935	0.037
Stacking + Random Forest + Logistic Regresion	0.999	0.999	0.999	0.999	0.9925	0.039

Gambar 4.2: Machine Learning tidak menggunakan Parameter terhadap train

Machine Learning	Accuracy	Precision	Recall	F1-score	Detection Rate	False Alarm Rate
Boosting	0.9783	0.981	0.989	0.985	0.9928	0.040
Bagging + Random Forest	0.976	0.979	0.989	0.984	0.9928	0.040
Stacking + Random Forest + Logistic Regresion	0.9785	0.981	0.990	0.985	0.9935	0.041

Gambar 4.3: Machine Learning menggunakan Parameter terhadap test

Machine Learning	Accuracy	Precision	Recall	F1-score	Detection Rate	False Alarm Rate
Boosting	1.0	1.0	1.0	1.0	0.9932	0.036
Bagging + Random Forest	0.992	0.992	0.997	0.995	0.9928	0.040
Stacking + Random Forest + Logistic Regresion	0.999	0.999	1.0	0.999	0.9935	0.041

Gambar 4.4: Machine Learning menggunakan Parameter terhadap train

4.1.2 Hasil Pengujian deteksi pe-pobe

Algoritma Pengujian deteski Pe - probe yang diusulkan bertujuan agar dapat mendeteksi dan memprediksi pe-probe yang terinfeksi malware atau tidak terfinfeksi malware. Hasil dari algoritma prediski dan deteksi pe-probe yang dapat dilihat sebagai berikut :

Nama File	Stacking + Random Forest + Logistic Regresion
00d1.exe	99.1
000d6.exe	98.8
000e7.exe	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.exe	99.1

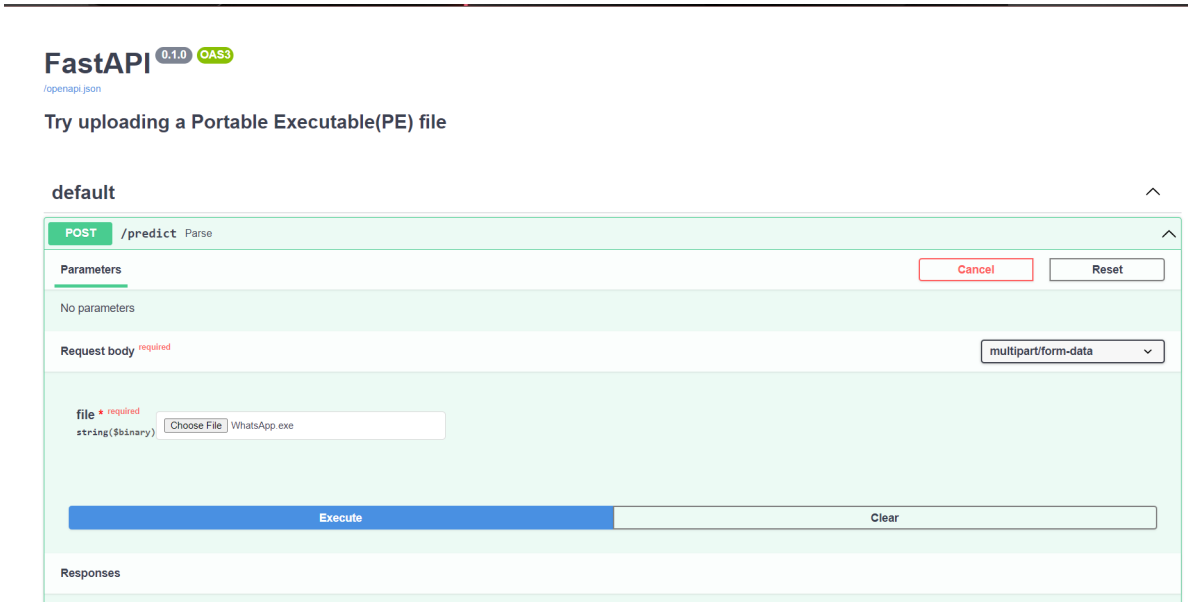
Gambar 4.5: Prediksi pe-probe terinfeksi malware

Nama File	Stacking + Random Forest + Logistic Regresion
Whatsappsetup.exe	11.3
Steap_api64.dll	15.9
Glib-2.dll	31.9
OBS-STUDIO-27.2.4.exe	1.53
Icudt71.dll	46.7
Gmodule-2.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

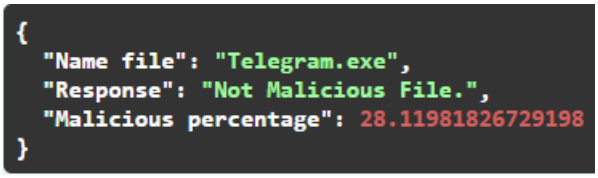
Gambar 4.6: Prediksi pe-probe tidak terinfeksi malware

4.1.3 Pengujian hasil implementasi algoritma pada prototype

Algoritma pengujian diusulkan bertujuan agar prototype yang dibuat dapat menampilkan data prediksi yang telah dimplementasikan di algortima machine learning yang dibuat. hasil algortima pengujian impelementasi pada prototype sebagai berikut.



Gambar 4.7: tampilan prototype yang akan menguji pe-probe



Gambar 4.8: tampilan prototype mengeluarkan hasil pengujian deteksi terhadap pe-probe yang tidak terinfeksi malware

Gambar 4.9: tampilan prototype mengeluarkan hasil pengujian deteksi terhadap pe-probe yang terinfeksi malware

4.2 Pembahasan

4.2.1 Algoritma Klasifikasi

Berdasarkan hasil analisis pada skenario-skenario di atas penulis menggunakan teknik ensemble machine learning untuk mengklasifikasi pe-probe yang terinfeksi malware atau tidak terinfeksi malware berdasarkan dataset pe-probe, diantaranya adalah ensemble bagging, gradient boost, ensemble stacking

data preprocessing

langkah pertama dalam data preprocessing adalah cengan 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 50 feature

klasifikasi

Untuk mendapatkan nilai akurasi dan prediksi dari model machine learning terbaik yang nantinya digunakan untuk memprediksi dan mengklasifikasi

pe-probe, penulis menggunakan 2 cara yaitu machine learning menggunakan parameter dan tidak menggunakan parameter dengan dipadukan cross validation dengan nilai kfolds = 10. fungsi kfolds sendiri digunakan untuk memilih model validasi terbaik yang nantinya diimplementasikan pada prototype.

Machine Learning	Parameter
Bagging + Random Forest	n_estimator , random_state, max_depth, max_features
Gradient Boost	n_estimator, learning_rate, max_depth, random_state, max_features
Stacking + Random Forest + Logistic Regresion	n_estimator , random_state, max_depth, max_features, verbose

Gambar 4.10: paraemeter yang digunakan dalam machine learning

4.2.2 Deteksi pe-probe

feature extraction

Untuk dapat menghasilkan nilai prediksi seperti pada gambar 4.1 dan 4.2 langkah yang digunakan dengan melakukan feature extraction dari pe-probe agar system dapat membaca isi feature dari pe-probe yang akan di uji.

```
def createDataframeFromPEdump(pe):

    dosHeaders = ['e_magic', 'e_cblp', 'e_cp', 'e_crlc', 'e_cparhdr',
            'e_minalloc', 'e_maxalloc', 'e_ss', 'e_sp', 'e_csum', 'e_ip', 'e_cs',
            'e_lfarlc', 'e_ovno', 'e_oemid', 'e_oeminfo', 'e_lfanew']
    fileHeaders=['Machine',
            'NumberOfSections', 'PointerToSymbolTable',
            'NumberOfSymbols', 'SizeOfOptionalHeader', 'Characteristics']
    optionalHeaders=['Magic',
            'MajorLinkerVersion', 'MinorLinkerVersion', 'SizeOfCode',
            'SizeOfInitializedData', 'SizeOfUninitializedData','AddressOfEntryPoint', 'BaseOfCode',
             'ImageBase', 'SectionAlignment','FileAlignment', 'MajorOperatingSystemVersion',
            'MinorOperatingSystemVersion', 'MajorImageVersion', 'MinorImageVersion',
            'MajorSubsystemVersion', 'MinorSubsystemVersion', 'SizeOfHeaders',
            'CheckSum', 'SizeOfImage', 'Subsystem', 'DllCharacteristics',
            'SizeOfStackReserve', 'SizeOfStackCommit', 'SizeOfHeapReserve',
            'SizeOfHeapCommit', 'LoaderFlags', 'NumberOfRvaAndSizes']
```

Gambar 4.11: feature extraction pe probe

klasifikasi dan deteksi

Setelah melakukan feature extraction langkah berikutnya adalah membandingkan hasil model klasifikasi machine learning terbaik dengan membandingkan feature extraction setelah system melakukan upload file. untuk membandingkan hasil klasifikasi machine learning dengan hasil feature extraction pe-probe dengan cara melakukan load file.pkl dari model machine learning menggunakan parameter yaitu ensemble stacking dan memprediksi file pe-probe

yang akan di uji. Pemilihan model machine learning terbaik mengacu pada nilai recall, accuracy, dan detection rate.

```
def getPredictions(df):
    load_scaler = joblib.load(open(r'../components/scaler.pkl','rb'))
    load_skpca = joblib.load(open(r'../components/pca.pkl','rb'))
    load_model = joblib.load(open(r'../components/rf_model.pkl','rb'))
    pipe = Pipeline([('scale', load_scaler),('pca', load_skpca),('rf_model', load_model)])
    df = np.array(df)
    df = df.reshape(1,-1)
    results = pipe.predict_proba(df)
    pred = pipe.predict(df)
    return (results[0],pred[0])
```

Gambar 4.12: Algoritma untuk memprediksi hasil feature extraction dengan membandingkan hasil algoritma klasifikasi machine learning

Untuk dapat membedakan pe-probe yang terinfeksi malware penulis membuat sistem yang dimana jika nilai prediksi pe-probe diatas 50 maka akan dikategorikan terinfeksi malware, sedangkan jika nilai prediksi dibawah 50 tidak dikategorikan terinfeksi malware seperti pada gambar 4.5 dan 4.6.

Dari hasil gambar 4.5 terdapat kesalahan dalam mendeteksi file yang terinfeksi malware, yang seharusnya Wildfire.exe merupakan file yang terinfeksi malware dengan nilai 20.1 yang seharusnya jika file pe-probe terinfeksi malware harus bernilai score prediksi diatas 50. Sedangkan hasil pada gambar 4.6 pada file Git-2.35.1.2.exe terdapat kesalahan juga dengan nilai score 54.5, yang dimana jika bernilai score diatas 50 maka nilai prediksi dikategorikan sebagai malware.

4.2.3 implementasi algoritma pada prototype

```
#creating fastApi app
from fileinput import filename


app_desc = """<h2> Try uploading a Portable Executable(PE) file"""
app = FastAPI(description = app_desc)
templates = Jinja2Templates(directory="html")
@app.get("/",include_in_schema=False)
async 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''):
            f.write(chunk)
```

Gambar 4.13: algoritma pembuatan sistem fast api

pada gambar 4.14 terdapat fungsi get dan post, dimana fungsi get berfungsi untuk menampilkan seluruh tampilan seperti pada gambar 4.7. Untuk fungsi post berfungsi untuk menampilkan hasil prediksi pe probe dari implementasi machine learning yang dibuat

```
# extract content
content = pefile.PE(path,fast_load=True)
dataframe = createDataframeFromPEdump(content)
binary_preds = getPredictions(dataframe)
if binary_preds[1] == 1:
    return {'Name file' : file.filename, 'response':'Malicious file','predictions':binary_preds[0][1]*100}
else:
    return {'Name file' : file.filename, 'Response': 'Your file is same from malware.','Malicious percentage':binary_pre
#remove temp file
os.close(_)
os.remove(path)
```

Gambar 4.14: algoritma pembuatan sistem fast api

pada gambar 4.14 merupakan algoritma untuk menampilkan hasil pengujian seperti pada gambar 4.8 dan 4.9, keluaran hasil pengujian dari hasil prediksi pe probe yang dimana jika nilai prediksi di atas 50 maka akan di kategorikan sebagai malware sedangkan jika nilai prediksi di bawah 50 maka akan di kategorikan non 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

Bab V

Kesimpulan dan Saran

5.1 Kesimpulan

Tugas akhir ini telah mencapai semua obyektif yang disebutkan pada Bab I, sebagai berikut:

1. Obyektif Pertama sudah tercapai. Bukti capaian dapat dilihat pada bab 4 hasil dan pembahasan pada bagian 4.1.1 pengujian hasil algoritma klasifikasi dan 4.2.1 Algoritma klasifikasi .

2. Obyektif Kedua berhasil dicapai. Bukti dari capaian ada pada bab 4 hasil dan pembahasan bagian 4.1.2 hasil pengujian deteksi pe-probe dan bagian 4.2.2 deteksi pe-probe

3. Obyektif Ketiga sukes dicapai dengan bukti ada pada Bab 4 hasil dan pembahasan bagian deteksi pe-probe pada bagian 4.1.3 dan 4.2.3

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.

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.

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.

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.

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

	fit_time	score_time	test_precision	train_precision	test_recall	train_recall	test_accuracy	train_accuracy	test_f1	train_f1
0	7.401972	0.016000	0.992857	1.0	0.992857	1.0	0.989822	1.0	0.992857	1.0
1	7.326790	0.016540	0.981073	1.0	0.990446	1.0	0.977099	1.0	0.985737	1.0
2	6.858364	0.016619	0.974026	1.0	0.996678	1.0	0.977099	1.0	0.985222	1.0
3	7.060916	0.016997	0.976821	1.0	0.993266	1.0	0.977041	1.0	0.984975	1.0
4	6.810374	0.016606	0.970297	1.0	0.996610	1.0	0.974490	1.0	0.983278	1.0
5	7.101663	0.019998	0.983389	1.0	1.000000	1.0	0.987245	1.0	0.991625	1.0
6	7.075171	0.015961	0.989967	1.0	0.996633	1.0	0.989796	1.0	0.993289	1.0
7	6.827976	0.015995	0.976271	1.0	0.979592	1.0	0.966837	1.0	0.977929	1.0
8	7.266158	0.016977	0.989362	1.0	0.982394	1.0	0.979592	1.0	0.985866	1.0
9	6.983827	0.017000	0.979522	1.0	0.986254	1.0	0.974490	1.0	0.982877	1.0

Gambar 5.1: nilai akurasi cross validation setiap iterasi nilai k pada algoritma stacking yang tidak menggunakan parameter

	fit_time	score_time	test_precision	train_precision	test_recall	train_recall	test_accuracy	train_accuracy	test_f1	train_f1
0	8.880923	0.136997	0.992832	0.991061	0.989286	0.997003	0.987277	0.990935	0.991055	0.994023
1	8.935621	0.155329	0.981073	0.991708	0.990446	0.998482	0.977099	0.992635	0.985737	0.995083
2	8.569645	0.134276	0.974026	0.993607	0.996678	0.997734	0.977099	0.993484	0.985222	0.995666
3	8.829190	0.147995	0.973597	0.992135	0.993266	0.998869	0.974490	0.993203	0.983333	0.995490
4	8.530385	0.140016	0.970297	0.992876	0.996610	0.997739	0.974490	0.992920	0.983278	0.995302
5	8.628238	0.137998	0.983389	0.993998	1.000000	0.998869	0.987245	0.994619	0.991625	0.996428
6	8.817333	0.166083	0.993220	0.991379	0.986532	0.997360	0.984694	0.991504	0.989865	0.994361
7	8.620162	0.142242	0.979592	0.993623	0.979592	0.997740	0.969388	0.993486	0.979592	0.995678
8	8.743794	0.150020	0.989362	0.992913	0.982394	0.998874	0.979592	0.993769	0.985866	0.995885
9	8.411416	0.135048	0.972789	0.992138	0.982818	0.996990	0.966837	0.991787	0.977778	0.994558

Gambar 5.2: nilai akurasi cross validation setiap iterasi nilai k pada algoritma bagging yang tidak menggunakan parameter

	fit_time	score_time	test_precision	train_precision	test_recall	train_recall	test_accuracy	train_accuracy	test_f1	train_f1
0	7.715717	0.130000	0.982332	0.989223	0.992857	0.997377	0.982188	0.989802	0.987567	0.993284
1	5.606781	0.007999	0.981073	0.993202	0.990446	0.998102	0.977099	0.993484	0.985737	0.995646
2	5.750542	0.006000	0.974026	0.990633	0.996678	0.998489	0.977099	0.991785	0.985222	0.994546
3	6.209297	0.012657	0.979933	0.992495	0.986532	0.997360	0.974490	0.992353	0.983221	0.994922
4	4.664375	0.003592	0.966887	0.990658	0.989831	0.998870	0.966837	0.992070	0.978224	0.994747
5	4.605058	0.004036	0.980132	0.992498	1.000000	0.997361	0.984694	0.992353	0.989967	0.994924
6	4.064024	0.004001	0.989865	0.991004	0.986532	0.996983	0.982143	0.990937	0.988196	0.993985
7	4.828130	0.004580	0.979592	0.992135	0.979592	0.997740	0.969388	0.992353	0.979592	0.994930
8	4.574613	0.004001	0.975524	0.990320	0.982394	0.998124	0.969388	0.991221	0.978947	0.994207
9	4.346042	0.003971	0.975945	0.992512	0.975945	0.997366	0.964286	0.992353	0.975945	0.994933

Gambar 5.3: nilai akurasi cross validation setiap iterasi nilai k pada algoritma boosting yang tidak menggunakan parameter

	fit_time	score_time	test_precision	train_precision	test_recall	train_recall	test_accuracy	train_accuracy	test_f1	train_f1
0	8.851490	0.036996	0.996416	1.000000	0.992857	1.0	0.992366	1.000000	0.994633	1.000000
1	9.629454	0.029998	0.977987	1.000000	0.990446	1.0	0.974555	1.000000	0.984177	1.000000
2	7.751528	0.019999	0.977199	1.000000	0.996678	1.0	0.979644	1.000000	0.986842	1.000000
3	7.701917	0.021987	0.980066	0.999623	0.993266	1.0	0.979592	0.999717	0.986622	0.999811
4	8.215872	0.038998	0.970297	0.999623	0.996610	1.0	0.974490	0.999717	0.983278	0.999812
5	9.433470	0.030003	0.983389	1.000000	1.000000	1.0	0.987245	1.000000	0.991625	1.000000
6	8.634591	0.034000	0.989933	1.000000	0.993266	1.0	0.987245	1.000000	0.991597	1.000000
7	8.383462	0.024001	0.976271	1.000000	0.979592	1.0	0.966837	1.000000	0.977929	1.000000
8	9.761465	0.020998	0.989362	1.000000	0.982394	1.0	0.979592	1.000000	0.985866	1.000000
9	8.827033	0.035999	0.976190	1.000000	0.986254	1.0	0.971939	1.000000	0.981197	1.000000

Gambar 5.4: nilai akurasi cross validation setiap iterasi nilai k pada algoritma stacking yang menggunakan parameter

	fit_time	score_time	test_precision	train_precision	test_recall	train_recall	test_accuracy	train_accuracy	test_f1	train_f1
0	9.225655	0.152676	0.992832	0.991428	0.989286	0.996628	0.987277	0.990935	0.991055	0.994021
1	9.078296	0.183606	0.987302	0.990584	0.990446	0.998102	0.982188	0.991501	0.988871	0.994329
2	8.446884	0.137120	0.973941	0.993223	0.993355	0.996224	0.974555	0.992068	0.983553	0.994721
3	9.683423	0.143998	0.980066	0.992879	0.993266	0.998869	0.979592	0.993769	0.986622	0.995865
4	8.304103	0.175028	0.973510	0.994369	0.996610	0.998116	0.977041	0.994336	0.984925	0.996239
5	8.442626	0.129951	0.980132	0.992120	1.000000	0.996608	0.984694	0.991504	0.989967	0.994359
6	8.664748	0.147992	0.989933	0.992123	0.993266	0.997360	0.987245	0.992070	0.991597	0.994735
7	8.415048	0.133002	0.979592	0.993631	0.979592	0.998870	0.969388	0.994336	0.979592	0.996243
8	8.742231	0.139996	0.989362	0.991800	0.982394	0.998499	0.979592	0.992637	0.985866	0.995138
9	8.293250	0.142953	0.966102	0.991399	0.979381	0.997366	0.959184	0.991504	0.972696	0.994374

Gambar 5.5: nilai akurasi cross validation setiap iterasi nilai k pada algoritma bagging yang menggunakan parameter

	fit_time	score_time	test_precision	train_precision	test_recall	train_recall	test_accuracy	train_accuracy	test_f1	train_f1
0	3.870866	0.073340	0.996416	1.0	0.992857	1.0	0.992366	1.0	0.994633	1.0
1	3.880308	0.009023	0.984177	1.0	0.990446	1.0	0.979644	1.0	0.987302	1.0
2	3.619102	0.012001	0.973941	1.0	0.993355	1.0	0.974555	1.0	0.983553	1.0
3	3.256301	0.007014	0.976821	1.0	0.993266	1.0	0.977041	1.0	0.984975	1.0
4	3.765687	0.009009	0.970199	1.0	0.993220	1.0	0.971939	1.0	0.981575	1.0
5	3.441574	0.008001	0.983389	1.0	1.000000	1.0	0.987245	1.0	0.991625	1.0
6	3.887201	0.012002	0.986622	1.0	0.993266	1.0	0.984694	1.0	0.989933	1.0
7	3.249896	0.008001	0.976271	1.0	0.979592	1.0	0.966837	1.0	0.977929	1.0
8	3.791130	0.008987	0.989324	1.0	0.978873	1.0	0.977041	1.0	0.984071	1.0
9	3.699523	0.008041	0.976190	1.0	0.986254	1.0	0.971939	1.0	0.981197	1.0

Gambar 5.6: nilai akurasi cross validation setiap iterasi nilai k pada algoritma boosting yang menggunakan parameter