Klasifikasi Malware Dengan Machine Learning

Natnanael Andra Wijaya 1st Penulis
Teknik Informatika
Universitas Telkom
Purwokerto, Indonesia
nathanaeladr@student.telkomuniversity.ac.id

Wahyu Adi Prabowo 2nd Penulis
Teknik Informatika
Universitas Telkom
Purwokerto, Indonesia
wahyuadi@ittelkom-pwt.ac.id

Trihastuti Yuniati 3rd Penulis
Teknik Informatika
Universitas Telkom
Purwokerto, Indonesia
trihastuti@ittelkom-pwt.ac.id

Abstrak*— Perkembangan malware yang pesat menjadi tantangan besar bagi keamanan sistem informasi. Berdasarkan data G Data Security Labs, pada 2022 ditemukan lebih dari 50 juta jenis malware baru. Penelitian ini mengimplementasikan tiga algoritma machine learning yaitu Logistic Regression, XGBoost, dan Convolutional Neural Network (CNN) untuk mengklasifikasikan malware melalui analisis data statis. Dataset terdiri dari 130.046 sampel malware dan benign yang diperoleh dari Kaggle serta VirusShare. Proses penelitian meliputi pra-pemrosesan data, pembagian dataset menjadi 80% data latih dan 20% data uji, pelatihan model, serta evaluasi kinerja menggunakan metrik akurasi, presisi, recall, dan F1-score. Hasil pengujian menunjukkan XGBoost sebagai model dengan performa terbaik dengan akurasi 99,31%, precision 99,30%, recall 99,33%, dan F1-score 99,31%. CNN berada di posisi kedua dengan akurasi 98,89%, precision 98,91%, recall 98,88%, dan F1-score 98,89%. Logistic Regression mencatat akurasi 96,11%, precision 96,07%, recall 96,15%, dan F1-score 96,11%. XGBoost terbukti menjadi model paling efektif dalam klasifikasi malware pada penelitian ini sehingga berpotensi meningkatkan akurasi dan efisiensi sistem deteksi malware.

Kata kunci*— Machine Learning, Logistic Regression, XGBoost, Convolutional Neural Network , Malware

I. PENDAHULUAN

Perkembangan teknologi digital membawa manfaat besar, namun juga diiringi ancaman serius dari perangkat lunak berbahaya atau *malware* [1]. *Malware* merupakan program, aplikasi, atau kode yang dirancang untuk menginfeksi sistem komputer serta dapat mencuri atau menghapus data yang tersimpan [2]. Laporan dari Deep Instinct mencatat bahwa serangan siber berbasis *malware* meningkat 358% pada 2020 dibandingkan tahun sebelumnya [3]. Lebih mengkhawatirkan lagi, varian *malware* modern memiliki kemampuan mengubah perilaku dan kodenya untuk menghindari deteksi [4]. Berdasarkan data G Data Security Labs, tahun 2022 saja tercatat sekitar 50 juta *malware* baru [5].

Salah satu pendekatan yang dinilai efektif untuk mengatasi ancaman ini adalah penerapan *machine learning* (ML). ML merupakan cabang dari *Artificial Intelligence* (AI) yang mengembangkan algoritma berdasarkan pengamatan, penilaian, serta generalisasi data melalui metode statistik [6]. Keunggulan ML terletak pada kemampuannya beradaptasi dalam mendeteksi varian *malware* baru dengan memanfaatkan beragam algoritma dan fitur yang ada [7]. Selain itu, proses analisisnya cepat dan efisien, sehingga mampu mengolah data dalam jumlah besar. Dalam klasifikasi *malware*, berbagai algoritma ML dapat digunakan untuk mencari metode dengan akurasi tertinggi seperti algoritma XGBoost (*eXtreme Gradient Boosting*), CNN (*Convolutional Neural Network*), dan *Logistic Regression*. Pemilihan ketiga algoritma tersebut didasari oleh tingkat akurasi yang tinggi ketiga akurasi dalam klasifikasi *malware*. Penelitian terdahulu oleh Almaleh et al. (2023) melaporkan *Logistic Regression* memiliki akurasi 96% [8]. Aziz (2025) dengan metode *XGBoost* mencapai akurasi 98% [9], sedangkan penelitian Lad & Adamuthe (2020) mencatat CNN mampu mencapai akurasi 99.59% [10].

Rangkaian penelitian dimulai dari persiapan dataset *malware* yang diunduh dari Kaggle. Kemudian, model ML dibangun menggunakan ketiga algoritma tersebut, diikuti evaluasi menggunakan metrik akurasi, sensitivitas (*recall*), dan *F1-score*. Tahap terakhir adalah menentukan algoritma dengan performa terbaik untuk menghasilkan sistem klasifikasi *malware* yang akurat dan andal.

II. KAJIAN TEORI

A. *Malware*

Malware atau virus adalah istilah yang merujuk pada perangkat lunak yang dirancang untuk menginfeksi dan merusak sistem komputer. Jenis *malware* sangat beragam, mencakup *viruses, worms, spyware, adware, trojans, bots, rootkits, backdoors, ransomware,* dan *spam* [11]. Semakin canggih perkembangan teknologi mendorong kompleksitas *malware* bertambah. Salah satu bentuk umum *malware* adalah *packet executable* (PE) berformat .exe yang dapat menyamarkan kode berbahaya, dengan cara mengompresi ukuran file dan mengaburkan isi kode sehingga sulit terdeteksi maupun dianalisis [12]. Urgensi pengembangan sistem anti-*malware* menjadi sangat penting karena sifat *malware* yang merugikan karena dapat menyebabkan kerusakan perangkat, kehilangan dana, maupun pencurian data pribadi.

B. Machine Learning

Machine learning adalah cabang kecerdasan buatan yang memungkinkan sistem mempelajari data tanpa pemrograman eksplisit. Dalam klasifikasi *malware, machine learning* menganalisis fitur atau atribut file serta proses mencurigakan untuk mengenali pola yang mengindikasikan keberadaan *malware*. Algoritma digunakan untuk mengelompokkan data menjadi "*malware*" atau "bukan *malware*" berdasarkan pengamatan. Pendekatan ini bervariasi, mulai dari metode sederhana seperti analisis statistik dan heuristik, hingga teknik lebih kompleks seperti *Logistic Regression, XGBoost,*

Random Forest, Support Vector Machines (SVM), Naive Bayes, serta model *deep learning* seperti CNN. Berbagai metode ini memungkinkan deteksi malware dilakukan secara otomatis, efisien, dan akurat, sehingga sistem keamanan dapat mengidentifikasi ancaman lebih cepat.

C. *Convolutional Neural Network* 1 Dimensi

Convolutional Neural Network (CNN) 1-Dimensi (1D CNN) adalah arsitektur jaringan saraf tiruan untuk memproses data satu dimensi berurutan, seperti deret numerik atau urutan API call pada analisis *malware*. Model ini terdiri dari lapisan konvolusi, pooling, aktivasi, dan *fully connected*. Proses dimulai dengan normalisasi dan augmentasi data, dilanjutkan konvolusi untuk mengekstraksi pola lokal, aktivasi ReLU untuk sifat non-linear, serta max pooling untuk mereduksi dimensi. Representasi fitur kemudian diproses di lapisan *fully connected* guna menghasilkan prediksi malware atau benign, dengan evaluasi menggunakan binary *cross-entropy loss* untuk mengukur selisih antara prediksi dan label sebenarnya.

D. *XGBoost*

XGBoost adalah algoritma gradient boosting yang membangun decision tree secara berurutan, di mana tiap pohon baru memperbaiki kesalahan pohon sebelumnya. Dengan memanfaatkan gradien dan hessian dari loss function serta regularisasi, *XGBoost* mengurangi *overfitting*. Prosesnya meliputi prediksi awal, perhitungan residual, pelatihan pohon untuk memprediksi residual, dan penggabungan hasil dengan *learning rate*, berulang hingga target tercapai. Algoritma ini efisien, mendukung penanganan *missing value*, paralelisasi, dan mampu mendeteksi pola kompleks dengan akurasi tinggi.

E. *Logistic Regression*

Logistic Regression adalah algoritma klasifikasi linier yang memprediksi probabilitas kelas menggunakan fungsi logistik (sigmoid). Model ini mencari koefisien terbaik yang memaksimalkan likelihood data pelatihan, sehingga memetakan kombinasi linier fitur menjadi nilai probabilitas antara 0 dan 1. Berdasarkan ambang batas (umumnya 0,5), data diklasifikasikan ke kelas positif atau negatif. *Logistic Regression* sederhana, cepat dilatih, mudah diinterpretasikan, dan efektif pada data dengan hubungan linier, sehingga sering digunakan sebagai baseline dalam klasifikasi.

F. *Confusion Matrix*

Confusion matrix adalah metode evaluasi yang digunakan untuk menganalisis keakuratan model klasifikasi dengan membandingkan hasil prediksi terhadap label aktual [13]. Matriks ini terdiri dari empat komponen, yaitu *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN), yang merepresentasikan jumlah prediksi benar atau salah untuk setiap kelas. Berdasarkan nilai-nilai tersebut, dihitung metrik evaluasi seperti *accuracy* (proporsi prediksi benar), *precision* (proporsi prediksi positif yang benar), *recall* (proporsi kasus positif yang terdeteksi), dan *F1-score* (rata-rata harmonik precision dan recall). *Confusion matrix* memberikan gambaran menyeluruh mengenai kemampuan model dalam membedakan kelas secara tepat, serta membantu mengidentifikasi potensi perbaikan performa model [13].

III. METODE

Penelitian ini menggunakan pendekatan kuantitatif untuk mengevaluasi kinerja tiga algoritma machine learning, yaitu Logistic Regression, XGBoost, dan CNN dalam klasifikasi *malware* berbasis analisis data statis. Tahapan mencangkup pengumpulan data, pra-pemrosesan, pembagian dataset, pelatihan model, dan evaluasi. Sistem diawali dengan pengumpulan dan pra-pemrosesan data *malware* dan non-*malware* yang telah diverifikasi. Data dibagi menjadi data latih dan uji (80:20) [14]. *Logistic Regression* digunakan sebagai *baseline* linier, *XGBoost* sebagai model boosting untuk meningkatkan akurasi, dan CNN untuk mengekstraksi fitur kompleks secara otomatis. Evaluasi pada data uji menggunakan akurasi, presisi, *recall,* dan *F1-score*, kemudian hasil ketiga model dibandingkan untuk menentukan detektor *malware* terbaik.

A. Pengumpulan Data

Dataset yang digunakan diperoleh dari platform Kaggle untuk sample berisi file - file aman (benign) dan sample *malware* yang bersumber dari VirusShare dengan total sampel sebanyak 130.046 sampel. Data mencakup fitur-fitur statis dari file PE (*Portable Executable*) yang telah melalui proses verifikasi dan validasi untuk memastikan kualitas dan konsistensi [15].

B. Pra-pemrosesan Data

Pra-pemrosesan dilakukan untuk meningkatkan kualitas data dan kesiapan dalam pelatihan model. Tahap ini meliputi:

1. Pembersihan data, dengan memnghapus data yang terduplikasi, tidak konsisten, dan tidak lengkap.
2. Penanganan outlier, dengan menggunakan teknik capping atau teknik membatasi nilai-nilai ekstrem yang berada di luar batas wajar ke nilai maksimum atau minimum. Pemilihan batas bawah menggunakan 1st persentil dan 99th persentil.
3. Normalisasi dan *Encoding*, dengan menyamakan skala fitur numerik serta mengubah variabel kategorikal menjadi numerik (one-hot encoding).
4. Penyeimbangan Kelas dengan Synthetic Minority Over-sampling Technique (SMOTE) yang menghasilkan data sintetis untuk kelas minoritas dengan cara membentuk sampel baru secara interpolatif berdasarkan tetangga terdekat dari data minoritas yang ada.

C. Pembagian Dataset

Dataset dibagi menjadi dua subset utama yaitu yaitu training set dan testing set. Dataset dibagi menjadi 80% data latih dan 20% data uji dengan metode *Stratified K-Fold Cross-Validation*. Teknik ini mempertahankan proporsi kelas yang seimbang pada setiap fold, sehingga hasil evaluasi lebih stabil [13]. Fungsi utama pembagian dataset yaitu memungkinkan evaluasi kinerja model terhadap data yang tidak pernah dilihat sebelumnya (*unseen data*), sehingga dapat mengukur kemampuan generalisasi model secara objektif. Terdapat dua skenario pemisahan data yang disesuaikan dengan arsitektur model yang digunakan yaitu pengunaan *feature scaling (X_scaled)* yang dibagi menggunakan fungsi *train_test_split* dari library *scikit-learn* untuk algoritma *Logistic Regression* dan *XGBoost*,

sedangkan algoritma CNN menyesuaikan strukturnya untuk arsitektur CNN *(X_cnn*) dibagi dengan metode yang identik.

D. Pelatihan dan Pembuatan Model

Proses pelatihan dan pembuatan model dilakukan menggunakan bahasa pemrograman Python dengan memanfaatkan library *machine learning*, seperti *scikit-learn, xgboost,* dan *TensorFlow/Keras*. Meskipun implementasi bersifat berbasis *library*, rancangan model tetap mengikuti prinsip matematis dan rumus fundamental dari masing-masing algoritma.

1. CNN 1 Dimensi

Arsitektur CNN 1D dirancang mengikuti operasi konvolusi seperti pada Persamaan (1), fungsi aktivasi ReLU pada Persamaan (2), dan *cross-entropy loss* pada Persamaan (3). Data yang telah dipreproses diumpankan ke lapisan konvolusi untuk mengekstraksi pola lokal, diikuti lapisan pooling untuk reduksi dimensi, lalu dihubungkan ke lapisan *fully connected* guna menghasilkan prediksi akhir.

$$(f * g)(t) = \int_{-\infty}^{\infty} f(T)g(t-T)dT \quad (1)$$

Dimana :

- f : Fungsi input, yaitu data yang akan diproses oleh konvolusi.
- g : Filter atau kernel, yaitu matriks kecil yang digunakan untuk mendeteksi fitur spesifik dari input.
- t : Variabel yang menunjukkan posisi atau waktu di mana hasil konvolusi dihitung.
- τ : Variabel dummy yang digunakan dalam integral untuk menjumlahkan produk dari fungsi input dan filter yang digeser.

$$ReLU(x) = max(0, x) \quad (2)$$

Dimana :

- x : *Input* ke fungsi aktivasi, bisa berupa nilai setelah konvolusi dan *pooling*.

$$Loss = \sum_i yi\ log\ (pi) \quad (3)$$

Dimana :

- yi : adalah label sebenarnya dari data ke- i
- pi : adalah probabilitas prediksi untuk data ke- i

2. *XGBoost*

Implementasi *XGBoost* mengacu pada formulasi *gradient boosting* dengan regularisasi seperti pada Persamaan (4) dan (5). Setiap iterasi melibatkan pembentukan pohon keputusan baru yang dilatih untuk memprediksi residual dari iterasi sebelumnya menggunakan gradien (turunan pertama) dan hessian (turunan kedua) dari fungsi *loss*. Prediksi akhir diperoleh melalui penjumlahan berbobot hasil dari seluruh pohon.

$$\mathrm{L}(\phi) = \sum_{i=1}^{n} \mathrm{l}\left(\mathrm{yi}, \hat{y}_i^{(t)}\right) + \sum_{k=1}^{t} \Omega(\mathrm{fk}) \quad (4)$$

- $\hat{y}$: Prediksi akhir yang dihasilkan oleh *XGBoost* setelah menjumlahkan semua prediksi dari setiap pohon.$\hat{y}m$: Prediksi yang dihasilkan oleh pohon.
- $f_m(x)$: Prediksi yang dihasilkan oleh pohon ke- m untuk input x.
- m : Pohon ke- m di dalam model *boosting*.
- M : Jumlah total pohon dalam *XGBoost*.

$$\hat{y}^{(t)} = \hat{y}^{(t-1)} + ft(x) \quad (5)$$

Dimana :

- $\hat{y}^{(t)}$: Prediksi akhir pada iterasi ke-t.
- $\hat{y}^{(t-1)}$: Prediksi akhir pada iterasi sebelumnya.
- $ft(x)$: Output pohon ke-t yang menyesuaikan kesalahan (residual) dari prediksi sebelumnya.

3. *Logistic Regression*

Model ini dibangun berdasarkan fungsi logistik (sigmoid) yang memetakan kombinasi linier dari fitur input menjadi probabilitas kelas target sebagaimana ditunjukkan pada Persamaan (6). Selama pelatihan, parameter β dioptimalkan untuk meminimalkan *cross-entropy loss* sesuai formulasi matematis, sehingga model dapat memisahkan kelas malware dan benign secara optimal.

$$\hat{y} = \frac{1}{1+e-(\beta 0+\beta 1x1+\beta 2x2+\cdots+\beta pxp)} \quad (6)$$

Dimana :

- $\hat{y}$: adalah probabilitas kelas target (biasanya antara 0 dan 1).
- $\beta 0$: adalah intercept.
- $\beta 1, \beta 2, \ldots, \beta p$: adalah prediksi dari model ke-i.
- $x1, x2, \ldots, xp$: adalah fitur-fitur input.

E. Evaluasi Model

Kinerja model dievaluasi pada data uji menggunakan confusion matrix. Confusion Matrix adalah suatu proses yang dipakai dalam menganalisa keakuratan dari model klasifikasi

yang dibuat untuk mengidentifikasi data dengan kelas yang berbeda serta mengetahui performa dari suatu model klasifikasi tersebut [13]. Confusion metrix dapat menghitung metrix evaluasi seperti *accuracy, presision, recall,* dan *f1-scrore* [16].

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (7)$$

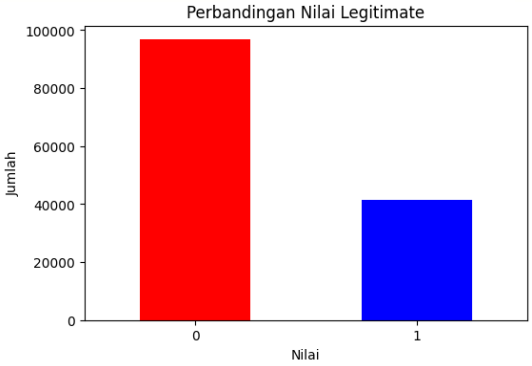
$$Presicion = \frac{True\ Positives\ (TP)}{True\ Positives\ (TP)+False\ Positive(FP)} \quad (8)$$

$$Recall = \frac{True\ Positives\ (TP)}{True\ Positives\ (TP)+False\ Negative(FN)} \quad (9)$$

$$F1-Score = 2x\left(\frac{Precision\ x\ Recall}{Precision+Recall}\right) \quad (10)$$

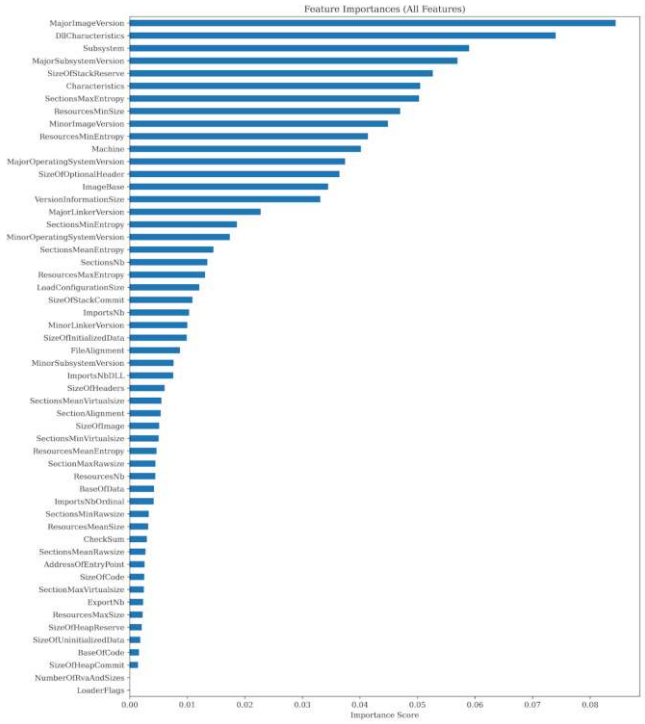
IV. HASIL DAN PEMBAHASAN

Dataset yang digunakan dalam penelitian ini terdiri dari berbagai file executable dengan kolom-kolom yang berperan sebagai atribut prediktor (feature). Terdapat 1 kolom yang berperan sebagai variabel target, yaitu legitimate, yang hanya memiliki dua nilai: 0 untuk *malware* dan 1 untuk file bukan *malware* (legitimate). Hasil perbandingan nilai legitimate 0 dan 1 menunjukan ada perbedaan bahwa nilai label 0 lebih banyak kurang dari 100,000 data dibandingkan data label 1 hanya pada kisaran 40,000 – 50,000 data (Gambar 1 (a)).



GAMBAR 1
(PERBANDINGAN NILAI LEGITIMATE)

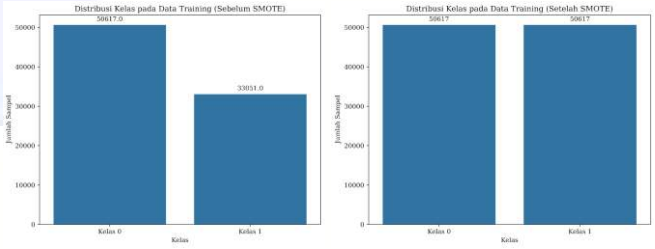
Berdasarkan visualisasi feature importance dengan Decision Tree (Gambar 2 (b)), fitur *MajorImageVersion* memiliki pengaruh terbesar dengan skor 0.08 dalam membedakan *malware* dan non-*malware*, diikuti *DllCharacteristics* dengan skor 0.07 dan *Subsystem* dengan skor 0.06. Fitur-fitur ini terkait versi header PE, karakteristik DLL, dan jenis subsistem, yang sering dimodifikasi *malware* untuk menyamarkan diri. Pengaruh sedang terlihat pada *SizeOfStackReserve, Characteristics,* dan *SectionsMaxEntropy* dengan skor antara 0.03 hingga 0.05, yang berkaitan dengan alokasi memori, struktur file, dan tingkat entropy. Sementara itu, *LoaderFlags* dan *MinorLinkerVersion* memiliki skor di bawah 0.01 sehingga kontribusinya minimal. Hasil ini menunjukkan atribut struktur utama file dan versi PE lebih relevan dibanding fitur statistik umum, dan model dapat disederhanakan dengan sekitar 15 fitur teratas yang memiliki skor di atas 0.02 untuk efisiensi tanpa menurunkan akurasi.



GAMBAR 2
(FEATURE IMPORTANCE DATASET)

Ketidakseimbangan kelas pada penelitian ini terjadi karena kelas "0" (*malware*) jauh lebih dominan dibanding kelas "1" (non-*malware*), sehingga berpotensi menurunkan deteksi kelas minoritas. Masalah ini diatasi dengan *Synthetic Minority Over-sampling Technique* (SMOTE), yang menghasilkan data sintetis kelas minoritas secara interpolatif untuk menciptakan variasi lebih realistis.

Sebelum penerapan SMOTE, dilakukan preprocessing berupa imputasi nilai hilang pada top_features menggunakan median dengan *SimpleImputer*, kemudian normalisasi fitur menggunakan *StandardScaler* agar memiliki skala seragam. SMOTE diterapkan hanya pada data latih untuk mencegah *data leakage*, menghasilkan distribusi seimbang masing-masing 50.617 sampel per kelas, sehingga meningkatkan sensitivitas model terhadap kelas minoritas tanpa menurunkan akurasi.



GAMBAR 3
(DISTRIBUSI KELAS PADA DATA TRAINING SEBELUM DAN SESUDAH SMOTE)

Pembuatan model *machine learning* dalam klasifikasi malware menggunakan *feature selection* agar model tidak terbebani akan informasi yang tidak relevan sehingga kemampuan dalam membedakan antara kelas *malware* dan non-*malware* meningkat. Hasil model *Logistic Regresion* (LR) menunjukan hasil yang cukup baik dilihat dari laporan

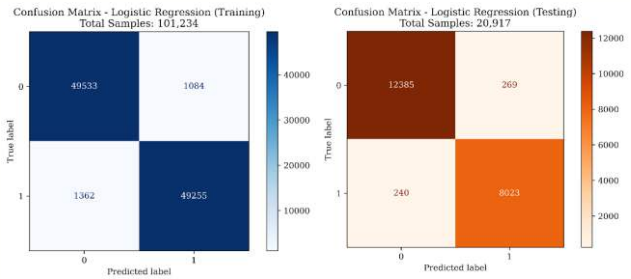
klasifikasi dan *confusion matrix* model ini. *Confusion matrix* model *Logistic Regression* menunjukkan performa klasifikasi yang cukup baik pada data training maupun testing. Pada data testing berjumlah 20.917 sampel, model berhasil mengidentifikasi 12.385 *malware* (label 0) sebagai True Positive (TP) dan 8.023 non-*malware* (label 1) sebagai True Negative (TN). Kesalahan yang terjadi meliputi 240 False Positive (FP) dan 269 False Negative (FN). Akurasi pada data testing tercatat sekitar 97,57%, diperoleh dari (12.385 + 8.023) / 20.917 ≈ 0,9757. Meski tergolong tinggi, nilai FN menunjukkan sensitivitas deteksi *malware* masih dapat ditingkatkan. Pada data training yang terdiri dari 101.234 sampel, performa model sedikit lebih baik. Sebanyak 49.533 non-*malware* terdeteksi benar (TP) dan 49.255 *malware* juga terklasifikasi tepat (TN), dengan kesalahan 1.362 FP dan 1.084 FN. Akurasi training mencapai 97,59%, dihitung dari (49.533 + 49.255) / 101.234 ≈ 0,9759.

```
[LogReg] Accuracy: 0.97567
[LogReg] Classification Report:
              precision    recall  f1-score   support

           0    0.98099   0.97874   0.97986     12654
           1    0.96756   0.97095   0.96925      8263

    accuracy                        0.97567     20917
   macro avg    0.97427   0.97485   0.97456     20917
weighted avg    0.97568   0.97567   0.97567     20917
```

GAMBAR 4
(LAPORAN KLASIFIKASI MODEL LR)



Gambar 5
(*Confusion Matrix LR Training* dan Testing Data)

Model *machine learning* menggunakan CNN mengadaptasi arsitektur CNN-1D. Arsitektur ini digunakan untuk klasifikasi malware terdiri dari satu lapisan Conv1D dengan 64 filter, kernel size 3, dan aktivasi ReLU, diikuti Max Pooling 1D untuk mengurangi dimensi. Dropout 0,3 diterapkan untuk mencegah *overfitting*, kemudian Flatten menghubungkan ke lapisan Dense berisi 128 unit dengan ReLU. Lapisan output menggunakan *softmax* untuk menghasilkan probabilitas prediksi antara kelas malware dan non-*malware*. Model dikompilasi dengan optimizer Adam, *loss sparse_categorical_crossentropy*, dan metrik akurasi. Pelatihan dilakukan selama 20 epoch dengan *batch size* 64. Hasil evaluasi pada data uji menunjukkan akurasi dan *classification report* yang kompetitif, menandakan arsitektur ini efektif dalam mengekstraksi fitur lokal dari data statis file executable, menyederhanakan representasi data sekuensial, dan membedakan secara akurat antara malware dan non-malware.

Gambar 6 menunjukkan hasil pelatihan dan evaluasi CNN-1D selama 20 epoch. Akurasi meningkat stabil dari 97,18% (loss 0,0809) pada epoch pertama menjadi 98,98% (loss 0,0322) di epoch terakhir, menandakan konvergensi yang baik. Evaluasi pada data uji menghasilkan akurasi 98,83% dengan laporan klasifikasi dua kelas.Kelas 0 memiliki *precision* 99,39%, *recall* 98,67%, dan *F1-score* 99,03%, sedangkan kelas 1 mencapai precision 97,99%, recall 99,67%, dan F1-score 98,53%. Recall tinggi pada kelas 1 menunjukkan kemampuan deteksi positif yang sangat baik. Rata-rata makro *F1-score* tercatat 98,78% dan akurasi tertimbang 98,83%, menandakan model seimbang tanpa bias signifikan terhadap salah satu kelas.

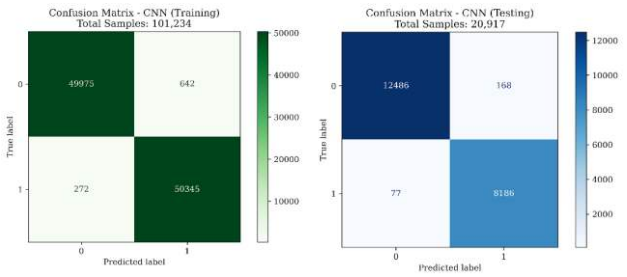
```
Epoch 1/20
/usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv.py:
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
1582/1582 ━━━━━━━━━━━━━━━━━━━━ 12s 7ms/step - accuracy: 0.9718 - loss: 0.0809
Epoch 2/20
1582/1582 ━━━━━━━━━━━━━━━━━━━━ 21s 7ms/step - accuracy: 0.9843 - loss: 0.0485
Epoch 3/20
1582/1582 ━━━━━━━━━━━━━━━━━━━━ 12s 7ms/step - accuracy: 0.9850 - loss: 0.0466
Epoch 4/20
1582/1582 ━━━━━━━━━━━━━━━━━━━━ 11s 7ms/step - accuracy: 0.9856 - loss: 0.0438
Epoch 5/20
1582/1582 ━━━━━━━━━━━━━━━━━━━━ 11s 7ms/step - accuracy: 0.9869 - loss: 0.0412
Epoch 6/20
1582/1582 ━━━━━━━━━━━━━━━━━━━━ 9s 6ms/step - accuracy: 0.9871 - loss: 0.0407
Epoch 7/20
1582/1582 ━━━━━━━━━━━━━━━━━━━━ 18s 11ms/step - accuracy: 0.9878 - loss: 0.0384
Epoch 8/20
1582/1582 ━━━━━━━━━━━━━━━━━━━━ 10s 6ms/step - accuracy: 0.9881 - loss: 0.0370
Epoch 9/20
1582/1582 ━━━━━━━━━━━━━━━━━━━━ 11s 7ms/step - accuracy: 0.9882 - loss: 0.0368
Epoch 10/20
1582/1582 ━━━━━━━━━━━━━━━━━━━━ 11s 7ms/step - accuracy: 0.9879 - loss: 0.0377
Epoch 11/20
1582/1582 ━━━━━━━━━━━━━━━━━━━━ 11s 7ms/step - accuracy: 0.9882 - loss: 0.0367
Epoch 12/20
1582/1582 ━━━━━━━━━━━━━━━━━━━━ 21s 7ms/step - accuracy: 0.9879 - loss: 0.0365
Epoch 13/20
1582/1582 ━━━━━━━━━━━━━━━━━━━━ 11s 7ms/step - accuracy: 0.9886 - loss: 0.0356
Epoch 14/20
1582/1582 ━━━━━━━━━━━━━━━━━━━━ 11s 7ms/step - accuracy: 0.9888 - loss: 0.0351
Epoch 15/20
1582/1582 ━━━━━━━━━━━━━━━━━━━━ 10s 6ms/step - accuracy: 0.9889 - loss: 0.0340
Epoch 16/20
1582/1582 ━━━━━━━━━━━━━━━━━━━━ 11s 7ms/step - accuracy: 0.9890 - loss: 0.0334
Epoch 17/20
1582/1582 ━━━━━━━━━━━━━━━━━━━━ 21s 7ms/step - accuracy: 0.9892 - loss: 0.0334
Epoch 18/20
1582/1582 ━━━━━━━━━━━━━━━━━━━━ 19s 6ms/step - accuracy: 0.9894 - loss: 0.0330
Epoch 19/20
1582/1582 ━━━━━━━━━━━━━━━━━━━━ 11s 7ms/step - accuracy: 0.9896 - loss: 0.0329
Epoch 20/20
1582/1582 ━━━━━━━━━━━━━━━━━━━━ 20s 7ms/step - accuracy: 0.9898 - loss: 0.0322
654/654 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
[CNN] Accuracy: 0.98829
[CNN] Classification Report:
              precision    recall  f1-score   support

           0    0.99387   0.98672   0.99028     12654
           1    0.97989   0.99068   0.98526      8263

    accuracy                        0.98829     20917
   macro avg    0.98688   0.98870   0.98777     20917
weighted avg    0.98835   0.98829   0.98830     20917
```

Gambar 6
(Laporan Klasifikasi Model XGBoost)

Confusion matrix CNN (Gambar 7) menunjukan hasil yang lebih baik dibandingkan model sebelumnya. Pada data training dengan 101.234 sampel, model CNN mencapai akurasi 99,1% dengan 49.975 *true negative* (TN) dan 50.345 *true positive* (TP). Kesalahan klasifikasi minim, yakni 642 *false positive* (FP) dan 272 *false negative* (FN), menunjukkan model mampu mempelajari pola data secara optimal dengan risiko *overfitting* yang rendah.Pada data testing berjumlah 20.917 sampel, akurasi tetap 99,1% dengan 12.886 TN dan 8.186 TP, serta hanya 77 FP dan 108 FN. Konsistensi hasil pada kedua dataset mengindikasikan kemampuan generalisasi yang sangat baik, sementara perbedaan kecil pada jumlah FP dan FN menegaskan efektivitas arsitektur CNN dalam menangkap pola penting secara stabil.



Gambar 7
(*Confusion Matrix XGBoost Training* dan Testing Data)

Model Terakhir yaitu XGBoost yang tersusun atas objek XGBClassifier() untuk membangun model klasifikasi berbasis algoritma tersebut. Hasil model XGBoost menjadi model terbaik dalam klasifikasi malware karena menghasilkan nilai *accuracy, precision, recall*, dan *f1-score* diatas 99% untuk klasifikasi *malware* maupun benign (Gambar 7). Pada data training dengan 101.234 sampel, model *XGBoost* mencatat performa sangat tinggi. Sebanyak 50.529 sampel label 0 (*malware*) terklasifikasi benar sebagai *True Positive* (TP) dan 50.585 sampel label 1 (*non-malware*) terdeteksi tepat sebagai *True Negative* (TN). Kesalahan sangat minim, hanya 32 *False Positive* (FP) dan 88 *False Negative* (FN), menghasilkan akurasi 99,88% ((50.529 + 50.585) / 101.234 ≈ 0,9988). Tingginya akurasi ini menunjukkan model mampu mempelajari pola data dengan baik tanpa indikasi *overfitting*.Pada data testing berjumlah 20.917 sampel, kinerja tetap unggul dengan 12.569 TP dan 8.203 TN, serta hanya 60 FP dan 85 FN. Akurasi testing mencapai 99,31% ((12.569 + 8.203) / 20.917 ≈ 0,9931). Selisih kecil antara akurasi training (99,88%) dan testing (99,31%) menandakan kemampuan generalisasi yang baik. Secara keseluruhan, XGBoost memberikan hasil konsisten dengan akurasi mendekati sempurna pada kedua dataset.

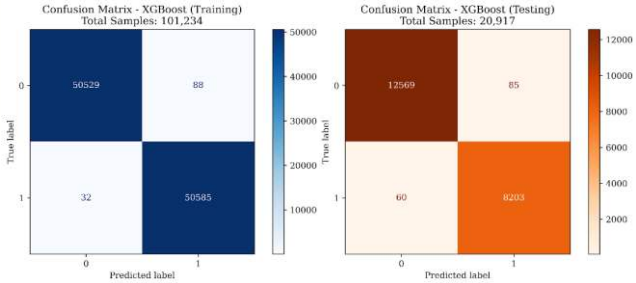
```
[XGBoost] Evaluation:
Accuracy: 0.99307

Classification Report:
              precision    recall  f1-score   support

           0    0.99525   0.99328   0.99426     12654
           1    0.98974   0.99274   0.99124      8263

    accuracy                        0.99307     20917
   macro avg    0.99250   0.99301   0.99275     20917
weighted avg    0.99307   0.99307   0.99307     20917
```

Gambar 8
(Laporan Klasifikasi Model CNN)



Gambar 8
(*Confusion Matrix CNN Training* dan Testing Data)

Evaluasi model difokuskan pada metrik utama, yaitu akurasi, presisi, *recall*, dan *F1-score*, guna menentukan algoritma paling efektif dalam klasifikasi malware. Tiga metode dibandingkan untuk merepresentasikan pendekatan yang beragam: model linear melalui *Logistic Regression, ensemble learning* berbasis pohon dengan XGBoost, dan *deep learning* berbasis CNN 1D yang memanfaatkan pola sekuensial pada data. Hasil evaluasi disajikan dalam tabel evaluasi pada tabel 1.

Tabel 1
(Tabel Evaluasi Model LR, XGBoost, dan CNN)

Skenario	Label	Precision	Recall	F1 Score	Accuracy
Logistic Regresion	0	0.98099	0.97874	0.97986	0.97567
	1	0.96756	0.97095	0.96925	
	Mean	**0.97427**	**0.97485**	**0.97456**	
XGBoost	0	0.99525	0.99328	0.99426	0.99307
	1	0.98974	0.99274	0.99124	
	Mean	**0.99250**	**0.99301**	**0.99275**	
Convulation Neural Network 1D	0	0.99387	0.98672	0.99028	0.98829
	1	0.97989	0.99068	0.98526	
	Mean	**0.98688**	**0.98870**	**0.98777**	

Hasil evaluasi menunjukkan seluruh model mencapai akurasi di atas 97,5% dalam klasifikasi biner. *XGBoost* menempati posisi teratas dengan akurasi 99,31%, diikuti CNN 1D sebesar 98,83%, dan *Logistic Regression* 97,57%. *XGBoost* unggul di semua metrik, dengan precision tertinggi 0,995 pada kelas 0 dan recall tertinggi 0,993 pada kelas 1, menandakan kemampuan deteksi positif yang sangat akurat dan konsisten di kedua kelas. CNN 1D menunjukkan keseimbangan precision dan recall dengan selisih kecil antar kelas, sehingga stabil dalam mengenali malware dan non-malware secara proporsional. *Logistic Regression*, meskipun memiliki performa baik, menunjukkan precision kelas 1 yang lebih rendah dibanding kelas 0, menandakan kelemahan dalam mengidentifikasi instance positif.

Secara komparatif, *XGBoost* menawarkan kombinasi akurasi dan konsistensi tertinggi, CNN 1D menjadi pilihan yang solid untuk kestabilan performa antar kelas, sedangkan *Logistic Regression* lebih cocok sebagai *baseline* model yang sederhana namun cukup efektif. Pemilihan model terbaik akan bergantung pada prioritas: *XGBoost* untuk akurasi maksimal, CNN 1D untuk keseimbangan deteksi, dan Logistic Regression untuk efisiensi komputasi.

V. KESIMPULAN

Penelitian ini berhasil menghasilkan tiga model klasifikasi malware berdasarkan fitur – fitur statik pada file executable dengan algoritma *logistic regression,* CNN, dan *XGBoost*. Hasil evaluasi menunjukkan seluruh model memiliki akurasi tinggi. *XGBoost* menjadi yang terbaik dengan akurasi 99,31% serta precision dan recall tinggi dan seimbang untuk kedua kelas, sehingga efektif mengidentifikasi baik aplikasi normal maupun berbahaya. 1D meraih akurasi 98,83% dengan keunggulan dalam menangkap pola data numerik berurutan, menjaga keseimbangan precision dan recall. *Logistic Regression* memperoleh CNN akurasi 97,57%, tetap kompetitif dan layak sebagai baseline model. Temuan ini membuktikan bahwa machine learning berbasis fitur statis dapat secara efektif memprediksi malware terutama algoritma *XGBoost* yang terbukti paling andal pada penelitian ini.

REFERENSI

[1] W. Satria, J. Prayoga, and E. Dores, "Sosialisasi Keamanan Siber Bagi Pelajar Dalam Menghadapi Era Digital," Abdi Dalem: Jurnal Pengabdian Kepada Masyarakat, vol. 2, no. 1, pp. 9–17, 2025.

[2] E. S. Alomari, R. R. Nuiaa, Z. A. A. Alyasseri, H. J. Mohammed, N. S. Sani, M. I. Esa, and B. A. Musawi, "Malware detection using deep learning and correlation-based feature selection," Symmetry, vol. 15, no. 1, p. 123, 2023.

[3] Deep Instinct, "Deep Instinct Reports 358% Increase in Malware Attacks in 2020," 2021. [Online]. Available: https://www.deepinstinct.com/. [Accessed: Jan. 2, 2025].

[4] P. Maniriho, A. N. Mahmood, and M. J. M. Chowdhury, "A study on malicious software behaviour analysis and detection techniques: Taxonomy, current trends and challenges," Future Generation Computer Systems, vol. 130, pp. 1–18, 2022.

[5] G Data, "Attacks every few seconds: Around 100 malware variants per minute threaten IT security," 2023. [Online]. Available: https://presse.gdata.de/news-attacks-every-few-seconds-around-100-malware-variants-per-minute-threaten-it-security?id=174381&menueid=28982&l=english. [Accessed: Jan. 2, 2025].

[6] R. Alfred, "The rise of machine learning for big data analytics," in Proc. 2016 2nd Int. Conf. Science in Information Technology (ICSITech), Balikpapan, 2016, pp. 1–6.

[7] P. Kishore, B. P. Gond, and D. P. Mohapatra, "Enhancing Malware Classification with Machine Learning: A Comparative Analysis of API Sequence-Based Techniques," in Proc. 2024 IEEE Int. Conf. Smart Power Control and Renewable Energy (ICSPCRE), 2024, pp. 1–6.

[8] A. Almaleh, R. Almushabb, and R. Ogran, "Malware API calls detection using hybrid logistic regression and RNN model," Applied Sciences, vol. 13, no. 9, p. 5439, 2023.

[9] T. A. Aziz, "Klasifikasi Malware Android Dengan Menggunakan Metode XGBoost Algoritma," Jurnal Repositor, vol. 7, no. 1, pp. 103–110, 2025.

[10] S. Choudhary and A. Sharma, "Malware detection & classification using machine learning," in Proc. 2020 Int. Conf. Emerging Trends in Communication, Control and Computing (ICONC3), 2020, pp. 1–4.

[11] U. H. Rao, U. Nayak, U. H. Rao, and U. Nayak, "Malicious software and anti-virus software," in The InfoSec Handbook: An Introduction to Information Security, 2014, pp. 141–161.

[12] A. H. N. Al-ojaimi, "Advanced Framework for Detecting Malware in Portable Executable (PE) Files Using a Multi-Model," Journal of Information Systems Engineering and Management, vol. 10, no. 36, pp. 769–781, 2025.

[13] H. Cai, J. Cai, and L. Sun, "An adaptive gradient privacy-preserving algorithm for federated XGBoost," in Proc. 2023 2nd Asia Conf. Algorithms, Computing and Machine Learning, 2023, pp. 277–282.

[14] D. Z. Syeda and M. N. Asghar, "Dynamic malware classification and API categorisation of Windows portable executable files using machine learning," Applied Sciences, vol. 14, no. 3, p. 1015, 2024.

[15] D. Z. Syeda and M. N. Asghar, "Dynamic malware classification and API categorisation of Windows portable executable files using machine learning," Applied Sciences, vol. 14, no. 3, p. 1015, 2024.

[16] J. F. Balarezo, S. Wang, K. G. Chavez, A. Al-Hourani, and S. Kandeepan, "A survey on DoS/DDoS attacks mathematical modelling for traditional, SDN and virtual networks," Engineering Science and Technology, an International Journal, vol. 31, p. 101065, 2022.