BAB 2
LANDASAN TEORI

Pada bab ini diuraikan landasan teori yang mendasari penelitian secara komprehensif, mencakup tinjauan terhadap penelitian terdahulu yang relevan, konsep-konsep teoretis yang menjadi fondasi analisis, serta deskripsi *tools* dan *framework* yang digunakan dalam penelitian ini. Pemaparan teori dipilih secara selektif berdasarkan relevansinya terhadap permasalahan deteksi *ransomware* menggunakan pendekatan *machine learning*, khususnya perbandingan algoritma *Random Forest* dan XGBoost.

2.1 Penelitian Terdahulu

Kajian literatur ini mencakup delapan penelitian kunci yang relevan dengan penerapan *machine learning* dalam keamanan siber, karakteristik *ransomware* modern, serta efektivitas RF dan XGBoost pada berbagai domain klasifikasi, dipilih berdasarkan relevansi metodologi, konteks aplikasi, dan kualitas publikasi.

Tabel 2.1. Ringkasan penelitian terdahulu terkait deteksi *ransomware* menggunakan *machine learning*

Referensi ke-1	
Judul	*Ransomware Detection Based on Machine Learning Using Memory Features*
Nama Penulis	Aljabri et al. [21]
Sumber Jurnal	*Egyptian Informatics Journal*
Tahun	2024
Permasalahan	Mendeteksi aktivitas *ransomware* berbasis fitur *memory dump* dengan tingkat akurasi tinggi.
Framework	XGBoost, *memory feature extraction*, dan klasifikasi berbasis *machine learning*.

Bersambung ke halaman berikutnya

Tabel 2.1 Ringkasan penelitian terdahulu terkait deteksi *ransomware* menggunakan *machine learning* (lanjutan)

Temuan	XGBoost mencapai akurasi sebesar 97,85% dan ROC-AUC sebesar 0,9788.
Pembahasan	Fitur berbasis memori mampu merepresentasikan perilaku *ransomware* secara efektif dan meningkatkan performa klasifikasi.
Relevansi	Menjadi dasar penggunaan XGBoost pada penelitian ini karena menunjukkan performa tinggi pada deteksi *ransomware*.
Referensi ke-2	
Judul	*Ransomware Detection and Classification Using Random Forest: A Case Study with the UGRansome2024 Dataset*
Nama Penulis	Azugo et al. [20]
Sumber Jurnal	*arXiv Preprint*
Tahun	2024
Permasalahan	Mendeteksi aktivitas *ransomware* menggunakan fitur jaringan dan transaksi *blockchain*.
Framework	*Random Forest*, analisis fitur jaringan, dan klasifikasi berbasis *machine learning*.
Temuan	Model *Random Forest* memperoleh akurasi sebesar 96% dan *precision* sebesar 97%.
Pembahasan	Fitur jaringan dan transaksi BTC memiliki kontribusi tinggi terhadap performa model deteksi *ransomware*.
Relevansi	Menjadi acuan utama penggunaan dataset UGRansome2024 dalam penelitian ini.
Referensi ke-3	
Judul	*Machine Learning and Deep Learning Approaches for CyberSecurity: A Review*

Bersambung ke halaman berikutnya

Tabel 2.1 Ringkasan penelitian terdahulu terkait deteksi *ransomware* menggunakan *machine learning* (lanjutan)

Nama Penulis	Halbouni et al. [25]
Sumber Jurnal	*IEEE Access*
Tahun	2022
Permasalahan	Meninjau dan mengevaluasi efektivitas berbagai algoritma *machine learning* dan *deep learning* pada sistem deteksi ancaman siber.
Framework	*Survey* algoritma klasifikasi, *ensemble learning*, dan IDS berbasis *machine learning*.
Temuan	Metode *ensemble* seperti *Random Forest* dan XGBoost mendominasi performa deteksi ancaman siber pada berbagai dataset IDS.
Pembahasan	Model *ensemble* memiliki kemampuan generalisasi dan stabilitas yang tinggi pada data kompleks dan berdimensi tinggi.
Relevansi	Menjadi landasan teoretis penggunaan pendekatan *ensemble learning* dan dasar pemilihan algoritma dalam penelitian ini.
Referensi ke-4	
Judul	*Addressing Behavioral Drift in Ransomware Early Detection Through Weighted Generative Adversarial Networks*
Nama Penulis	Urooj et al. [26]
Sumber Jurnal	*IEEE Access*
Tahun	2024
Permasalahan	Menangani *behavioral drift* pada deteksi dini *ransomware* yang menyebabkan penurunan akurasi model dari waktu ke waktu.

Bersambung ke halaman berikutnya

Tabel 2.1 Ringkasan penelitian terdahulu terkait deteksi *ransomware* menggunakan *machine learning* (lanjutan)

Framework	*Weighted Generative Adversarial Networks* (WGAN), analisis perilaku jaringan, dan klasifikasi berbasis *machine learning*.
Temuan	Pendekatan WGAN mampu menstabilkan performa deteksi *ransomware* meskipun terjadi pergeseran perilaku pada data baru.
Pembahasan	Fitur jaringan terbukti efektif untuk mendeteksi perilaku *ransomware* modern secara *real-time* dan adaptif terhadap varian baru.
Relevansi	Menjadi landasan penggunaan fitur jaringan dan motivasi pengembangan model yang adaptif pada penelitian ini.
Referensi ke-5	
Judul	*Enhancing IDS Performance Through a Comparative Analysis of Random Forest, XGBoost, and Deep Neural Networks*
Nama Penulis	Rahal et al. [27]
Sumber Jurnal	*Machine Learning with Applications (Elsevier)*
Tahun	2025
Permasalahan	Membandingkan performa algoritma klasifikasi *Random Forest*, XGBoost, dan DNN pada sistem deteksi intrusi.
Framework	*Random Forest*, XGBoost, DNN, dan optimasi *hyperparameter*.
Temuan	*Random Forest* mencapai akurasi 99,80% dengan AUC 0,9988, melampaui XGBoost (99,79%) dan DNN (98,66%) pada dataset NSL-KDD.
Pembahasan	Hasil menunjukkan keunggulan *ensemble* berbasis pohon keputusan dalam menangani hubungan kompleks antar fitur keamanan siber.

Bersambung ke halaman berikutnya

Tabel 2.1 Ringkasan penelitian terdahulu terkait deteksi *ransomware* menggunakan *machine learning* (lanjutan)

Relevansi	Menjadi dasar metodologi komparatif antara *Random Forest* dan XGBoost pada penelitian ini.
Referensi ke-6	
Judul	*STB: Synthetic Minority Oversampling Technique for Tree-Boosting Models for Imbalanced Datasets of Intrusion Detection Systems*
Nama Penulis	Li et al. [28]
Sumber Jurnal	*PeerJ Computer Science*
Tahun	2023
Permasalahan	Mengatasi ketidakseimbangan kelas pada dataset IDS menggunakan SMOTE yang diintegrasikan dengan algoritma *tree-boosting*.
Framework	SMOTE, LightGBM, XGBoost, CatBoost, dan klasifikasi berbasis *machine learning*.
Temuan	Penerapan SMOTE pada model *tree-boosting* meningkatkan nilai F1-*score* dan sensitivitas pada kelas minoritas secara signifikan.
Pembahasan	SMOTE efektif dalam meningkatkan representasi kelas minoritas pada data keamanan siber yang tidak seimbang.
Relevansi	Menjadi dasar penggunaan SMOTE pada penelitian ini untuk menangani *class imbalance* pada dataset UGRansome2024.
Referensi ke-7	
Judul	*Redundancy Coefficient Gradual Up-weighting-based Mutual Information Feature Selection Technique for Crypto-ransomware Early Detection*
Nama Penulis	Al-rimy et al. [29]
Sumber Jurnal	*Future Generation Computer Systems (Elsevier)*

Bersambung ke halaman berikutnya

Tabel 2.1 Ringkasan penelitian terdahulu terkait deteksi *ransomware* menggunakan *machine learning* (lanjutan)

Tahun	2021
Permasalahan	Mengembangkan metode seleksi fitur berbasis mutual information untuk deteksi dini *crypto-ransomware*.
Framework	Seleksi fitur berbasis *mutual information*, *machine learning*, dan analisis perilaku *ransomware*.
Temuan	Metode seleksi fitur yang diusulkan meningkatkan akurasi deteksi dini *crypto-ransomware* secara signifikan dibanding metode konvensional.
Pembahasan	Fitur perilaku jaringan dan pola enkripsi merupakan indikator yang sangat relevan untuk mendeteksi *ransomware* modern.
Relevansi	Menjadi landasan penggunaan fitur jaringan dan motivasi *feature engineering* pada penelitian ini.
Referensi ke-8	
Judul	*Machine Learning-Based Ransomware Classification of Bitcoin Transactions*
Nama Penulis	Al-Haija dan Alsulami [30]
Sumber Jurnal	*Applied Computational Intelligence and Soft Computing (Hindawi)*
Tahun	2023
Permasalahan	Menganalisis transaksi Bitcoin untuk identifikasi dan klasifikasi aktivitas *ransomware*.
Framework	Analisis transaksi BTC, *Random Forest*, XGBoost, dan klasifikasi berbasis *machine learning*.
Temuan	Fitur transaksi BTC mampu meningkatkan akurasi deteksi *ransomware* dengan kombinasi *Random Forest* dan XGBoost.
Pembahasan	Karakteristik transaksi *blockchain* dapat digunakan sebagai indikator perilaku *ransomware* yang andal.

Bersambung ke halaman berikutnya

Tabel 2.1 Ringkasan penelitian terdahulu terkait deteksi *ransomware* menggunakan *machine learning* (lanjutan)

Relevansi	Menjadi dasar penggunaan fitur transaksi BTC pada penelitian ini.

Berdasarkan Tabel 2.1, algoritma *ensemble learning*, khususnya *Random Forest* dan XGBoost, terbukti unggul pada domain keamanan siber dan deteksi *ransomware*, mampu menangani data yang kompleks, non-linear, dan berdimensi tinggi [25]. Azugo et al. [20] mencapai akurasi 96% dengan *Random Forest* pada dataset UGRansome2024, sedangkan Aljabri et al. [21] mencapai 97,85% dengan XGBoost pada deteksi *ransomware* berbasis *memory dump*.

Meskipun demikian, sebagian besar penelitian terdahulu memiliki keterbatasan: sebagian hanya berfokus pada satu algoritma tanpa pembanding [20], sebagian lain belum menerapkan penanganan *class imbalance*, *hyperparameter tuning*, maupun validasi statistik [28], dan sebagian hanya menggunakan fitur tertentu (memori atau *API call*) tanpa mengeksplorasi kombinasi fitur jaringan dan transaksi secara mendalam.

Oleh karena itu, penelitian ini mengisi kesenjangan tersebut dengan membangun analisis komparatif RF dan XGBoost pada dataset UGRansome2024, menerapkan *RandomizedSearchCV* untuk optimasi parameter, 10-*fold Stratified K-Fold Cross-Validation* untuk validitas evaluasi, SMOTE untuk ketidakseimbangan kelas, *feature engineering* dengan fitur derivatif tambahan, serta uji statistik Wilcoxon *signed-rank test* untuk menganalisis signifikansi perbedaan performa secara objektif [27, 28].

2.2 Tinjauan Teori

Bagian ini menguraikan konsep-konsep teoretis yang menjadi dasar pijakan penelitian, dimulai dari pemahaman mengenai *ransomware* sebagai ancaman keamanan siber, karakteristik dataset yang digunakan, hingga prinsip kerja algoritma *machine learning* yang diterapkan. Selain itu, dibahas pula teknik-teknik pendukung seperti penanganan ketidakseimbangan kelas, *feature engineering*, optimasi *hyperparameter*, validasi model melalui *cross-validation*, uji signifikansi

statistik, serta metrik evaluasi dan interpretabilitas yang digunakan untuk menilai performa model secara menyeluruh.

2.2.1 *Ransomware* dan Ancaman Keamanan Siber

Ransomware merupakan kategori perangkat lunak berbahaya (*malware*) yang mengenkripsi data atau memblokir akses sistem korban, kemudian menuntut tebusan (*ransom*) sebagai imbalan pemulihan akses [27]. Ancaman ini telah berkembang dari serangan oportunistik sederhana menjadi operasi kriminal terorganisir yang menarget infrastruktur kritis, institusi kesehatan, pemerintahan, dan perusahaan multinasional [29]. Kerugian global akibat *ransomware* diperkirakan mencapai miliaran dolar per tahun, dengan frekuensi serangan yang terus meningkat secara eksponensial [25].

A Tahapan Serangan *Ransomware*

Secara teknis, *ransomware* modern beroperasi melalui serangkaian tahapan yang terstruktur dan saling berkaitan, dan pemahaman terhadap setiap tahapan ini penting sebagai landasan ilmiah dalam merancang sistem deteksi yang efektif [29, 31]:

1. Infiltrasi (*Initial Access*)

   Ransomware masuk ke sistem target melalui *phishing*, eksploitasi kerentanan yang belum ditambal, *brute force* terhadap RDP, atau kredensial yang telah dikompromikan [5, 31]. Komponen awal (*dropper/loader*) ditempatkan tanpa menimbulkan kecurigaan.

2. Eksekusi dan Persistensi (*Execution and Persistence*)

   Muatan (*payload*) dieksekusi dan persistensi ditetapkan melalui modifikasi *registry*, entri *startup*, atau *scheduled task*, sekaligus menonaktifkan solusi keamanan yang terpasang [32].

3. Pengintaian dan Pergerakan Lateral (*Reconnaissance and Lateral Movement*)

   Pelaku memetakan jaringan korban, mencuri kredensial tambahan (misalnya menggunakan Mimikatz), dan bergerak lateral antar sistem, proses yang pada serangan *enterprise-level* dapat berlangsung berminggu-minggu sebelum enkripsi diaktifkan [31].

4. Komunikasi *Command and Control* (C2)

   Ransomware membangun saluran C2 untuk mengambil kunci enkripsi dan menerima instruksi [20, 30]. Fase ini menghasilkan pola lalu lintas khas, meliputi volume *Netflow_Bytes* yang tidak normal, protokol tertentu, dan transaksi *blockchain*, yang menjadi sinyal deteksi utama berbasis jaringan.

5. Eksfiltrasi Data (*Data Exfiltration*)

   Pada skema *double extortion*, data sensitif diekstraksi ke server eksternal sebelum enkripsi dimulai [32], menjadi senjata tekanan tambahan jika korban menolak membayar.

6. Enkripsi (*Encryption*)

   File dienkripsi menggunakan kriptografi asimetris yang kuat seperti kombinasi RSA-2048 dan AES-256 [29] pada ekstensi bernilai tinggi (`.docx`, `.sql`, `.jpg`, `.bak`), dirancang berlangsung secepat mungkin sebelum terdeteksi.

7. Permintaan Tebusan dan Pembayaran (*Ransom Demand*)

   Catatan tebusan (*ransom note*) ditampilkan berisi jumlah tebusan dalam mata uang kripto (umumnya Bitcoin atau Monero) dan batas waktu [30]. Transaksi Bitcoin dari tahap ini merepresentasikan jejak keuangan yang terekam pada dataset UGRansome2024, menjadikannya fitur kritis dalam sistem deteksi berbasis jaringan.

Ketujuh tahapan tersebut membentuk siklus serangan *ransomware* yang komprehensif. Tahapan ke-3 hingga ke-4 (pergerakan lateral dan komunikasi C2) menghasilkan artefak jaringan paling kaya yang dapat dieksploitasi sebagai fitur klasifikasi, karena pada fase inilah lalu lintas jaringan yang anomali dan transaksi *blockchain* mulai terekam secara konsisten [20].

2.2.2 Dataset UGRansome2024

Bagian ini menguraikan dataset UGRansome2024 secara mendalam, dimulai dari asal-usul dan latar belakang pengembangannya, sifat data yang digunakan, komponen konseptual yang menyusunnya, hingga struktur atribut dan definisi kelas target secara rinci. Pemahaman menyeluruh terhadap karakteristik dataset ini penting sebagai dasar interpretasi hasil eksperimen pada bab selanjutnya,

mengingat dataset ini memiliki sifat khusus yang membedakannya dari dataset lalu lintas jaringan pada umumnya.

A Asal-usul dan Latar Belakang Dataset

Dataset UGRansome2024 merupakan hasil pengembangan lanjutan dari dataset UGRansome1819 yang pertama kali diperkenalkan oleh Nkongolo et al. [20] sebagai sumber daya keamanan siber untuk mendeteksi *ransomware* dan serangan *zero-day*, khususnya yang menunjukkan perilaku *cyclostationary* (pola statistik yang berulang secara periodik pada lalu lintas jaringan). Dataset awal ini dibangun untuk mengatasi kelangkaan data publik yang dapat membedakan lalu lintas jaringan normal dari lalu lintas *ransomware*, sebuah kendala utama yang menghambat pengembangan strategi deteksi proaktif pada masa itu [20].

Mengingat beberapa atribut dataset ini berasal dari data transaksi Bitcoin, dataset ini bukan murni berbasis *blockchain* yang diekstraksi langsung dari *node blockchain* atau *ledger* penuh jaringan Bitcoin, melainkan dataset lalu lintas jaringan yang menyertakan sebagian atribut turunan dari transaksi *blockchain* sebagai bagian dari fiturnya. Empat atribut, yaitu `BTC`, `SeedAddress`, `ExpAddress`, dan `Clusters`, berasal dari domain *blockchain* dan merepresentasikan jejak finansial transaksi tebusan *ransomware* pada ekosistem Bitcoin, sementara atribut lainnya (`Time`, `Protocol`, `Family`, `Threats`, `Netflow_Bytes`, dan sejenisnya) sepenuhnya berasal dari domain lalu lintas jaringan konvensional. Karakteristik hibrida inilah yang mendasari perancangan fitur derivatif seperti `BTC_per_Byte` dan `BTC_x_Clusters` pada tahap *feature engineering*.

B Sifat Data

UGRansome2024 bukan murni hasil tangkapan (*capture*) lalu lintas jaringan nyata, melainkan kombinasi antara perilaku jaringan aktual dengan komponen *synthetic signatures*, yaitu sinyal yang secara sengaja dibangkitkan untuk keperluan pengujian dan simulasi pertahanan keamanan siber [20]. Sifat semi-sintetis ini tercermin pada kelas target *Synthetic Signature* (SS), yang dirancang sebagai sinyal buatan menyerupai *Lightning Network*. Performa tinggi model pada dataset ini sebagian mencerminkan kemampuannya mengenali pola sintetis yang memang dirancang untuk dapat dibedakan, sehingga generalisasi terhadap lalu lintas *ransomware* nyata tetap perlu divalidasi lebih lanjut pada data operasional.

C Komponen Konseptual Dataset

Azugo et al. [20] merancang UGRansome untuk mencakup sebelas komponen informasi yang saling melengkapi, yaitu *timestamps* untuk pelacakan waktu serangan, *flags* untuk kategorisasi jenis serangan, data protokol untuk wawasan vektor serangan, detail aliran jaringan untuk perilaku anomali, klasifikasi keluarga *ransomware*, analisis *malware*, *numeric clustering* untuk pola serangan umum, kuantifikasi kerugian finansial dalam USD maupun BTC, komponen *machine learning* untuk membangkitkan *signature* serangan, *synthetic signatures* untuk pengujian sistem pertahanan, serta deteksi anomali untuk dokumentasi penyimpangan pola. Kesebelas komponen ini direalisasikan sebagai atribut konkret pada dataset, dirinci pada Bagian 4.1.

D Struktur dan Definisi Atribut

Dataset yang digunakan dalam penelitian ini memuat 149.043 baris dan 14 kolom. Definisi lengkap serta tipe data setiap kolom telah dirangkum pada Tabel 4.1 (Bab 4), dengan penjelasan konseptual sebagai berikut:

1. `Time` merepresentasikan stempel waktu (*timestamp*) terjadinya transaksi atau aktivitas jaringan, digunakan untuk pelacakan temporal pola serangan.
2. `Protocol` menunjukkan protokol komunikasi jaringan yang digunakan (TCP, UDP, atau ICMP), yang memberikan wawasan terhadap vektor dan metodologi serangan.
3. `Flag` berfungsi sebagai penanda kategori jenis koneksi atau serangan, membantu proses klasifikasi dan pemahaman awal terhadap jenis aktivitas jaringan.
4. `Family` mengindikasikan nama keluarga *ransomware* yang teridentifikasi pada suatu transaksi, mencakup 17 keluarga berbeda seperti *WannaCry*, *Locky*, dan *SamSam* (dibahas rinci pada Bagian F).
5. `Threats` mengategorikan sembilan jenis ancaman jaringan, yaitu *Blacklist*, *Botnet*, *DoS*, *NerisBotnet*, *Port Scanning*, *SSH*, *Scan*, *Spam*, dan *UDP Scan*, merepresentasikan keragaman vektor serangan pada skenario keamanan jaringan nyata.

6. `IPaddress` berisi kelas kategoris alamat IP (dikodekan sebagai A/B/C/D), bukan alamat IP geografis yang sesungguhnya, sehingga hanya merepresentasikan pengelompokan sumber/tujuan lalu lintas secara abstrak.

7. `Port` menunjukkan nomor *port* jaringan yang digunakan dalam komunikasi.

8. `BTC` merepresentasikan nilai nominal transaksi dalam satuan Bitcoin, menjadi indikator kuantifikasi dampak finansial aktivitas *ransomware*.

9. `USD` merupakan nilai konversi dari `BTC` ke mata uang dolar Amerika Serikat berdasarkan kurs pasar pada saat transaksi.

10. `Netflow_Bytes` mengukur volume data (dalam *byte*) yang ditransfer dalam suatu sesi aliran jaringan (*netflow*), menjadi indikator intensitas komunikasi.

11. `SeedAddress` merepresentasikan alamat Bitcoin pengirim (*seed address*) yang menjadi titik asal transaksi.

12. `ExpAddress` merepresentasikan alamat Bitcoin tujuan (*expended address*) yang menerima transaksi.

13. `Clusters` menunjukkan hasil pengelompokan (*numeric clustering*) alamat *blockchain* berdasarkan kemiripan pola aktivitas, digunakan untuk mengenali pola serangan yang berulang.

14. `Prediction` adalah kolom label kelas target yang menjadi variabel yang diprediksi oleh model, dijelaskan lebih rinci pada subbagian berikut.

E Definisi Kelas Target

Variabel target `Prediction` bersifat multi-kelas dengan tiga nilai kategori, yang penamaannya merupakan hasil adaptasi terminologi *blockchain* yang dilakukan oleh Azugo et al. [20] terhadap label asli pada UGRansome1819:

1. S (*SegWit*/Normal): merupakan hasil pembaruan dari label asli *Signature*, direpresentasikan ke istilah *SegWit* (*Segregated Witness*) untuk mencerminkan keterkaitannya dengan identifikasi segmen transaksi *blockchain* yang bersifat normal/reguler [20].

2. A (*Anomaly*/Tidak Normal): merupakan hasil pembaruan dari label asli *Anomaly*, yang oleh Azugo et al. juga diistilahkan sebagai *Unusual Transaction*, yaitu transaksi yang menyimpang dari pola umum dan berpotensi mengindikasikan aktivitas mencurigakan [20].

3. SS (*Synthetic Signature*/Lightning): merupakan kelas sintetis yang oleh Azugo et al. di-*rebranding* menjadi *Lightning Network*, merepresentasikan pola transaksi yang menyerupai karakteristik jaringan *Lightning Network* pada ekosistem Bitcoin [20].

Selain ketiga kelas utama tersebut, Azugo et al. [20] juga mencatat adanya diferensiasi pada jenis alamat transaksi *blockchain* menjadi tiga tipe, yaitu *Pay-to-Public-Key-Hash* (P2PKH), *Pay-to-Script-Hash* (P2SH), dan *Pay-to-Witness-Public-Key-Hash* (P2WPKH), yang memberikan konteks tambahan mengenai jenis transaksi *blockchain* yang direkam pada kolom `SeedAddress` dan `ExpAddress`.

Berdasarkan uraian di atas, dapat disimpulkan bahwa UGRansome2024 adalah dataset keamanan siber multi-domain yang menggabungkan tiga dimensi informasi sekaligus, yaitu perilaku lalu lintas jaringan, karakteristik transaksi finansial *blockchain*, serta atribut identitas *malware*. Sebagian komponen data pada dataset ini, khususnya kelas SS, bersifat sintetis untuk keperluan pengujian sistem deteksi [20, 30].

F Keluarga *Ransomware* dalam Dataset UGRansome2024

Dataset UGRansome2024 memuat rekaman lalu lintas jaringan dari 17 keluarga *ransomware* yang berbeda, dengan total 149.043 sampel. Berikut penjelasan singkat karakteristik dan pola serangan masing-masing keluarga beserta jumlah sampelnya.

1. *WannaCry* (16.110 sampel)

   Menyebar masif Mei 2017 melalui eksploitasi *EternalBlue* pada kerentanan SMBv1 Windows, menginfeksi lebih dari 200.000 sistem di 150 negara [33, 34]. Mengenkripsi dengan RSA-2048/AES-128, meminta tebusan $300–$600 dalam Bitcoin, dan merepresentasikan ancaman berbasis TCP dengan pola komunikasi C2 khas dalam dataset ini.

2. *Locky* (25.062 sampel)

Teridentifikasi sejak Februari 2016, menyebar via kampanye *spam* dengan *macro* Word, memakai infrastruktur *fast-flux* untuk C2, dan mengenkripsi dengan RSA-2048/AES-128 pada ekstensi `.locky`/`.zepto` [35].

3. *SamSam* (19.657 sampel)

   Dioperasikan secara manual sejak akhir 2015, menargetkan institusi kesehatan dan pemerintah kota melalui kerentanan server dan RDP, diikuti pengintaian manual sebelum enkripsi, menghasilkan tuntutan tebusan lebih besar dibanding *ransomware* massal [36].

4. *JigSaw* (13.712 sampel)

   Muncul April 2016, dikenal karena menghapus sebagian file setiap jam jika tebusan tidak dibayarkan, meskipun enkripsi AES-nya relatif mudah didekripsi karena kunci tersimpan di memori proses [37].

5. *Flyper* (12.014 sampel)

   Ransomware berbasis Python sejak 2016, didistribusikan via *exploit kit* dan tautan berbahaya, dengan pola komunikasi jaringan sederhana yang menjadikannya relatif mudah dideteksi [29].

6. *DMALocker* (11.360 sampel)

   Aktif sejak awal 2016, menyebar via *spam* dan *exploit kit* dengan target awal Eropa Timur, versi 3.0/4.0 memperbaiki kelemahan kriptografi awal dengan RSA-2048 dan mampu mengenkripsi tanpa koneksi internet [38].

7. *EDA2* (6.054 sampel)

   Berasal dari proyek *open-source* edukasi keamanan tahun 2016 yang disalahgunakan untuk membuat puluhan varian *ransomware* nyata, menggunakan AES-256 untuk konten file dan RSA untuk proteksi kunci [39].

8. *Globe* (7.373 sampel)

   Aktif sejak pertengahan 2016 dengan banyak varian ekstensi (`.globe`, `.purge`), didistribusikan via *exploit kit* Rig dan *malvertising*, menggunakan Blowfish lalu beralih ke AES-256 [40].

9. *CryptXXX* (9.335 sampel)

Terdeteksi sejak April 2016, didistribusikan eksklusif via *exploit kit* Angler/Neutrino, mengenkripsi dengan RSA-4096 sekaligus memiliki kemampuan *infostealer* untuk kata sandi dan dompet kripto [41].

10. *Cryptohitman* (4.134 sampel)

    Varian dari *ransomware* Stampado tahun 2016 dengan ekstensi `.porno` sebagai intimidasi psikologis, dipasarkan sebagai *Ransomware-as-a-Service* di forum *dark web* [42].

11. *NoobCrypt* (1.248 sampel)

    Teridentifikasi sejak Juli 2016, dikenal karena kunci enkripsi identik untuk semua korban pada varian awalnya sehingga cepat didekripsi peneliti keamanan, sebelum diperbaiki dengan kunci unik per infeksi [43].

12. *TowerWeb* (4.381 sampel)

    Teridentifikasi sekitar 2016, secara khusus dirancang untuk mengenkripsi file server web melalui eksploitasi kerentanan *web application/plugin* [29].

13. *Globev3* (73 sampel)

    Varian ketiga keluarga Globe dengan peningkatan mekanisme enkripsi dan kemampuan menghindari deteksi *sandbox* [40], penting sebagai representasi evolusi berkelanjutan keluarga ini meski jumlah sampelnya kecil.

14. *CryptoLocker* (788 sampel)

    Ransomware pelopor akhir 2013 yang memperkenalkan model bisnis berbasis kriptografi asimetris (RSA-2048/AES-256), menyebar via botnet Gameover Zeus, dan dihentikan Juni 2014 melalui Operasi Tovar [44].

15. *CryptoLocker2015* (150 sampel)

    Varian peniru (*copycat*) yang muncul setelah infrastruktur CryptoLocker asli dilumpuhkan, memanfaatkan reputasi pendahulunya dengan infrastruktur yang dimodifikasi [44].

16. *APT* (9.730 sampel) dan *Razy* (7.862 sampel)

    Label `APT` merujuk pada kategori ancaman kompleks dan bertahan lama yang mungkin melibatkan komponen *ransomware* [20], sementara Razy adalah *trojan* pencuri data dan dompet kripto yang pada beberapa varian

juga mengunduh *ransomware* tambahan [32], mencerminkan kaburnya batas antara *ransomware* murni dan *malware* lainnya.

2.2.3 *Machine Learning* untuk Klasifikasi

Machine learning adalah cabang kecerdasan buatan yang mengembangkan algoritma dengan kemampuan belajar dari data dan membuat prediksi tanpa diprogram secara eksplisit untuk setiap skenario [25]. Dalam konteks keamanan siber, paradigma *supervised learning* menjadi pilihan dominan karena tersedia dataset berlabel dari insiden *malware* historis [27].

Pendekatan *supervised learning* untuk klasifikasi bekerja dengan memetakan fitur input $\mathbf{x} \in \mathbb{R}^d$ ke label kelas $y \in \{1, 2, \ldots, K\}$ melalui fungsi $f : \mathbb{R}^d \to \mathscr{Y}$ yang dipelajari dari data pelatihan $\mathscr{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ [25], dengan pembelajaran dioptimalkan melalui minimisasi fungsi *loss*. Keunggulan utamanya dibanding deteksi berbasis *signature* konvensional adalah kemampuan mendeteksi ancaman *zero-day* yang belum pernah dilihat sebelumnya [29].

2.2.4 *Ensemble Learning*

Ensemble learning adalah paradigma pembelajaran mesin yang menggabungkan prediksi dari beberapa model dasar (*base learner*) untuk menghasilkan prediksi akhir yang lebih akurat dan *robust* dibandingkan model tunggal [25], berakar pada hukum besar bilangan: jika model-model dasar independen dan memiliki *error* yang tidak berkorelasi sempurna, agregasi prediksinya mengurangi varians total [27].

Secara formal, diberikan M model dasar $\{f_1, f_2, \ldots, f_M\}$, prediksi *ensemble* untuk klasifikasi diperoleh melalui strategi *majority voting* sebagaimana ditunjukkan pada Persamaan 2.1.

$$\hat{y} = \arg\max_k \sum_{m=1}^{M} \mathbb{1}[f_m(\mathbf{x}) = k] \tag{2.1}$$

Terdapat dua strategi *ensemble* utama yang relevan dengan penelitian ini: *bagging* (digunakan oleh *Random Forest*) dan *boosting* (digunakan oleh XGBoost) [27].

2.2.5 Algoritma *Random Forest*

Random Forest adalah algoritma *ensemble* berbasis *bagging* (*Bootstrap Aggregating*) yang diperkenalkan oleh Breiman [45]. Algoritma ini membangun sejumlah T pohon keputusan secara independen dan paralel, masing-masing dilatih pada subset data yang diambil secara acak dengan pengembalian (*bootstrap sampling*) [27]. Keunikannya adalah penerapan *random feature subspace*: pada setiap pemisahan simpul (*node split*), hanya $m = \lfloor \sqrt{d} \rfloor$ fitur yang dipilih secara acak dari total d fitur sebagai kandidat pemisahan [25].

Pemisahan setiap simpul pada pohon keputusan RF dilakukan dengan memilih fitur dan titik ambang (*threshold*) yang meminimalkan tingkat impuritas Gini pada simpul anak yang dihasilkan [45]. Impuritas Gini pada suatu simpul t dihitung sebagaimana ditunjukkan pada Persamaan 2.2, di mana p_k adalah proporsi sampel kelas k pada simpul t dan K adalah jumlah kelas.

$$Gini(t) = 1 - \sum_{k=1}^{K} p_k^2 \tag{2.2}$$

Kandidat pemisahan terbaik dipilih dengan memaksimalkan penurunan impuritas (*information gain*) antara simpul induk dan simpul anak, sebagaimana ditunjukkan pada Persamaan 2.3, di mana N_t, N_L, dan N_R berturut-turut adalah jumlah sampel pada simpul induk, anak kiri, dan anak kanan.

$$\Delta Gini = Gini(t) - \left(\frac{N_L}{N_t} Gini(t_L) + \frac{N_R}{N_t} Gini(t_R) \right) \tag{2.3}$$

Pencarian kandidat terbaik hanya dilakukan pada subset m fitur yang dipilih secara acak, diterapkan berulang secara rekursif pada setiap simpul hingga kriteria penghentian tercapai (kedalaman maksimum, jumlah sampel minimum, atau simpul telah murni), sehingga membentuk satu pohon keputusan $h_t(\mathbf{x})$ secara utuh.

Proses pembentukan *Random Forest*: diberikan dataset pelatihan $\mathscr{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, untuk setiap pohon ke-$t$ ($t = 1, 2, \ldots, T$), dibuat *bootstrap sample* $\mathscr{D}_t$ berukuran n dengan pengambilan acak dari $\mathscr{D}$, kemudian pohon keputusan $h_t(\mathbf{x})$ dilatih pada $\mathscr{D}_t$ dengan pemilihan fitur acak pada setiap pemisahan simpul. Prediksi akhir untuk sampel baru $\mathbf{x}$ ditunjukkan pada Persamaan 2.4.

$$\hat{y}_{RF}(\mathbf{x}) = \text{mode}\{h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots, h_T(\mathbf{x})\} \tag{2.4}$$

Mekanisme *bagging* dan pemilihan fitur acak secara sinergistis menciptakan keberagaman (*diversity*) antar-pohon, kunci pengurangan varians tanpa meningkatkan bias secara signifikan [27]. Keunggulan RF dalam konteks deteksi *ransomware* meliputi kemampuan menangani fitur berskala berbeda tanpa normalisasi, ketahanan terhadap *outlier* dan *noise*, serta kemampuan menghasilkan *feature importance* [20]. Mekanisme *Out-of-Bag* (OOB) *error estimation* juga memungkinkan estimasi generalisasi tanpa *validation set* terpisah [25].

2.2.6 Algoritma XGBoost

XGBoost (*eXtreme Gradient Boosting*) adalah implementasi *gradient boosting* yang dioptimalkan secara komputasional, diperkenalkan oleh Chen dan Guestrin [46]. Berbeda dengan RF yang membangun pohon secara paralel, XGBoost membangun pohon secara sekuensial: setiap pohon baru dilatih untuk memperbaiki *residual error* dari agregasi pohon sebelumnya menggunakan penurunan gradien [27].

Persamaan 2.5 menunjukkan formulasi fungsi prediksi *ensemble* XGBoost setelah M iterasi, di mana $\mathscr{F}$ adalah ruang pohon keputusan.

$$\hat{y}_i^{(M)} = \sum_{m=1}^{M} f_m(\mathbf{x}_i), \quad f_m \in \mathscr{F} \tag{2.5}$$

XGBoost mengoptimalkan fungsi objektif pada setiap iterasi m sebagaimana ditunjukkan pada Persamaan 2.6, di mana ℓ adalah fungsi *loss* yang dapat didiferensiasikan, dan $\Omega(f_m) = \gamma T + \frac{1}{2}\lambda \|\mathbf{w}\|^2$ adalah regularisasi yang mengontrol kompleksitas pohon (T adalah jumlah daun, $\mathbf{w}$ adalah bobot daun) [46].

$$\mathscr{L}^{(m)} = \sum_{i=1}^{n} \ell\left(y_i, \hat{y}_i^{(m-1)} + f_m(\mathbf{x}_i)\right) + \Omega(f_m) \tag{2.6}$$

Untuk menentukan pemisahan simpul terbaik, XGBoost menghitung gradien orde pertama $g_i = \partial_{\hat{y}^{(m-1)}} \ell(y_i, \hat{y}_i^{(m-1)})$ dan gradien orde kedua (Hessian) $h_i = \partial^2_{\hat{y}^{(m-1)}} \ell(y_i, \hat{y}_i^{(m-1)})$ untuk setiap sampel i [46]. Nilai *gain* dari kandidat pemisahan yang membagi simpul menjadi anak kiri (L) dan anak kanan (R) dihitung sebagaimana ditunjukkan pada Persamaan 2.7, di mana $G_L = \sum_{i \in L} g_i$, $H_L = \sum_{i \in L} h_i$ (serupa untuk R), λ adalah parameter regularisasi L_2, dan γ adalah

biaya kompleksitas untuk menambah satu daun baru.

$$Gain = \frac{1}{2}\left[\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda}\right] - \gamma \tag{2.7}$$

Suatu simpul hanya dipisahkan apabila nilai *gain* bernilai positif (melampaui biaya γ); jika tidak, simpul tersebut menjadi daun (*leaf*). Proses pencarian *split* ini diulang untuk membangun satu pohon f_m secara rekursif, yang kemudian ditambahkan ke model *ensemble* sesuai Persamaan 2.5.

Kontribusi inovatif XGBoost meliputi regularisasi L_1/L_2 terintegrasi untuk mencegah *overfitting*, aproksimasi kuantil berbasis *histogram* untuk mempercepat pencarian *split*, penanganan *sparse data* secara *native*, komputasi paralel berbasis kolom, serta dukungan regularisasi melalui parameter `gamma`, `lambda`, dan `alpha` [46, 27]. Dalam domain deteksi *ransomware*, XGBoost unggul menangkap pola non-linear kompleks dari kombinasi fitur jaringan dan transaksi kripto, terutama pada distribusi kelas yang tidak seimbang [21].

2.2.7 *Class Imbalance* dan Teknik SMOTE

Class imbalance terjadi ketika distribusi kelas dalam dataset pelatihan sangat tidak merata, sehingga model cenderung bias terhadap kelas mayoritas dan gagal mendeteksi kelas minoritas yang justru sering kali lebih penting secara praktis [28]. Tingkat ketidakseimbangan suatu dataset diukur menggunakan *imbalance ratio* (IR), yaitu perbandingan antara jumlah sampel kelas mayoritas dan kelas minoritas, sebagaimana ditunjukkan pada Persamaan 2.8.

$$IR = \frac{N_{mayoritas}}{N_{minoritas}} \tag{2.8}$$

Dalam konteks deteksi *ransomware*, kondisi ini lazim terjadi karena lalu lintas normal jauh mendominasi dibandingkan lalu lintas *malware* [27]. IR yang tinggi secara langsung menurunkan *recall* pada kelas minoritas, yang dalam konteks keamanan siber berarti meningkatnya jumlah *false negative*, yaitu kegagalan mendeteksi serangan aktual.

SMOTE (*Synthetic Minority Over-sampling Technique*) merupakan teknik *oversampling* yang diperkenalkan oleh Chawla et al. [47] untuk mengatasi masalah ini. Berbeda dengan *random oversampling* yang sekadar menduplikasi sampel

minoritas, SMOTE membangkitkan sampel sintetis baru melalui interpolasi linear di ruang fitur. Untuk setiap sampel minoritas $\mathbf{x}_i$, dipilih k tetangga terdekat dari kelas yang sama, kemudian sampel sintetis baru dibangkitkan sebagaimana ditunjukkan pada Persamaan 2.9, di mana $\mathbf{x}_{nn}$ adalah salah satu dari k tetangga terdekat yang dipilih secara acak [47].

$$\mathbf{x}_{new} = \mathbf{x}_i + \delta \cdot (\mathbf{x}_{nn} - \mathbf{x}_i), \quad \delta \sim \text{Uniform}(0,1) \tag{2.9}$$

Pada penelitian ini, SMOTE diterapkan dengan $k = 5$ tetangga terdekat hanya pada data pelatihan (*training set*) untuk mencegah kebocoran informasi (*data leakage*) ke data pengujian [28].

2.2.8 *Feature Engineering*

Feature engineering adalah proses transformasi dan konstruksi fitur baru dari data mentah yang bertujuan meningkatkan kemampuan representasi informasi oleh model *machine learning* [25]. Penerapannya pada data jaringan dan finansial terbukti secara empiris meningkatkan performa model deteksi [48], dan pada dataset berdimensi tinggi yang tidak seimbang, kombinasinya dengan SMOTE menjadi krusial karena informasi redundan justru menurunkan akurasi *classifier* [49]. Dalam konteks data finansial dan transaksi *blockchain*, *feature engineering* bahkan disebut sebagai topik utama dalam deteksi *fraud* berbasis *machine learning* karena kemampuannya mengekstraksi informasi bermakna sekaligus menghilangkan redundansi [50]. Penelitian ini mengimplementasikan lima fitur derivatif baru berdasarkan pemahaman domain tentang karakteristik transaksi *ransomware*:

1. `BTC_per_Byte`: Rasio nilai transaksi Bitcoin terhadap volume *Netflow_Bytes*, merepresentasikan intensitas finansial per unit komunikasi jaringan, sebagaimana ditunjukkan pada Persamaan 2.10 (penambahan konstanta 1 menghindari pembagian dengan nol).

$$\texttt{BTC\_per\_Byte} = \frac{\texttt{BTC}}{\texttt{Netflow\_Bytes} + 1} \tag{2.10}$$

2. `Log_BTC` dan `Log_Bytes`: Transformasi logaritmik pada fitur `BTC` dan

`Netflow_Bytes` untuk mengurangi *skewness* distribusi yang ekstrem [25], sebagaimana ditunjukkan pada Persamaan 2.11.

$$\texttt{Log\_BTC} = \log(1 + \texttt{BTC}), \quad \texttt{Log\_Bytes} = \log(1 + \texttt{Netflow\_Bytes}) \tag{2.11}$$

3. `BTC_x_Clusters`: Fitur interaksi antara nilai BTC dan jumlah *cluster* alamat *blockchain*, merepresentasikan dampak finansial agregat dari jaringan distribusi *ransomware*. Al-Haija dan Alsulami [30] menunjukkan bahwa kombinasi dimensi finansial dan topologi jaringan *blockchain* meningkatkan diskriminasi antar keluarga *ransomware*. Fitur ini dihitung sebagai $\texttt{BTC} \times \texttt{Clusters}$.

4. `High_BTC`: Fitur biner yang mengindikasikan apakah nilai BTC suatu transaksi melampaui nilai median keseluruhan dataset, sebagai indikator *high-value transaction*, sebagaimana ditunjukkan pada Persamaan 2.12.

$$\texttt{High\_BTC} = \begin{cases} 1 & \text{jika } \texttt{BTC} > \text{median}(\texttt{BTC}) \\ 0 & \text{lainnya} \end{cases} \tag{2.12}$$

2.2.9 Analisis Korelasi Antar Fitur

Analisis korelasi merupakan teknik statistik yang mengukur kekuatan dan arah hubungan linear antara dua variabel numerik [25]. Salah satu ukuran korelasi yang paling umum digunakan adalah koefisien korelasi Pearson (r), yang dihitung sebagaimana ditunjukkan pada Persamaan 2.13, di mana x_i dan y_i adalah nilai observasi ke-i dari dua variabel, serta $\bar{x}$ dan $\bar{y}$ adalah nilai rata-ratanya.

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{2.13}$$

Nilai r berkisar antara -1 hingga $+1$, dengan nilai mendekati $+1$ menunjukkan korelasi positif kuat, mendekati -1 menunjukkan korelasi negatif kuat, dan mendekati 0 menunjukkan tidak adanya hubungan linear. Interpretasi

kekuatan korelasi mengacu pada skema klasifikasi yang diusulkan oleh Evans [51], sebagaimana dirangkum pada Tabel 2.2.

Tabel 2.2. Interpretasi Kekuatan Koefisien Korelasi menurut Evans (1996)

Rentang Nilai $\|r\|$	**Kategori**
0,00 – 0,19	Sangat lemah
0,20 – 0,39	Lemah
0,40 – 0,59	Sedang
0,60 – 0,79	Kuat
0,80 – 1,00	Sangat kuat

Sumber: [51]

Koefisien Pearson dipilih karena seluruh fitur yang dianalisis, baik fitur asli maupun hasil *feature engineering*, bersifat numerik kontinu, sehingga asumsi hubungan linear yang mendasari metode ini lebih wajar dibandingkan metode berbasis peringkat seperti Spearman [25]. Analisis ini ditambahkan karena penambahan lima fitur derivatif berpotensi menciptakan hubungan matematis langsung antar fitur, misalnya `BTC_x_Clusters` yang merupakan hasil perkalian langsung dari `BTC` dan `Clusters`, sehingga penting untuk mengidentifikasi potensi multikolinearitas sejak awal agar hasil *feature importance* dapat diinterpretasikan lebih tepat [20].

2.2.10 *Hyperparameter Tuning*

Hyperparameter adalah parameter model yang nilainya ditetapkan sebelum proses pelatihan dimulai dan tidak dapat dipelajari langsung dari data [27], sangat memengaruhi kapasitas model, kemampuan generalisasi, dan keseimbangan antara *bias* dan *variance*.

RandomizedSearchCV adalah metode *hyperparameter tuning* yang mengambil sampel konfigurasi secara acak dari distribusi parameter yang ditentukan, kemudian mengevaluasi setiap konfigurasi melalui *cross-validation* [27], lebih efisien secara komputasional dibandingkan *GridSearchCV* yang bersifat *exhaustive* [25].

Secara formal, diberikan ruang parameter $\Lambda = \Lambda_1 \times \Lambda_2 \times \ldots \times \Lambda_K$, *RandomizedSearchCV* memilih n_{iter} konfigurasi secara acak dari distribusi marginal setiap *hyperparameter*, kemudian mengembalikan konfigurasi terbaik sebagaimana

ditunjukkan pada Persamaan 2.14.

$$\boldsymbol{\lambda}^{*} = \arg\max_{\boldsymbol{\lambda}^{(j)}} \text{CV-Score}\left(\mathscr{M}_{\boldsymbol{\lambda}^{(j)}}\right) \tag{2.14}$$

Pada penelitian ini, *Random Forest* dituning dengan ruang parameter mencakup `n_estimators` $\in \{100, 200, 300, 500\}$, `max_depth` $\in \{10, 20, 30, \text{None}\}$, `min_samples_split` $\in \{2, 5, 10\}$, `min_samples_leaf` $\in \{1, 2, 4\}$, `max_features` $\in \{\text{sqrt}, \text{log2}\}$, dan `class_weight` $\in \{\text{balanced}, \text{None}\}$. Untuk XGBoost, parameter yang dituning meliputi `n_estimators` $\in \{100, 200, 300, 500\}$, `max_depth` $\in \{3, 5, 7, 9\}$, `learning_rate` $\in \{0.01, 0.05, 0.1, 0.2, 0.3\}$, `subsample` $\in \{0.6, 0.8, 1.0\}$, `colsample_bytree` $\in \{0.6, 0.8, 1.0\}$, `gamma` $\in \{0, 0.1, 0.2, 0.5\}$, `reg_alpha` $\in \{0, 0.1, 0.5, 1.0\}$, dan `reg_lambda` $\in \{1.0, 1.5, 2.0\}$ [46, 27].

Pemilihan rentang nilai didasarkan pada tiga pertimbangan. Pertama, `n_estimators`, `max_depth`, `min_samples_split`, `min_samples_leaf`, dan `max_features` pada RF dipilih karena paling memengaruhi performa model: Zhu et al. [52] menunjukkan melalui analisis *functional ANOVA* bahwa `max_depth` menyumbang lebih dari 78% dari total variansi performa RF, jauh melampaui `min_samples_leaf` (6,29%) dan `min_samples_split` (0,63%). Kedua, rentang `max_depth` XGBoost (3 hingga 9) dan `learning_rate` (0,01 hingga 0,3) mengikuti praktik pada penelitian deteksi intrusi jaringan sejenis: Songma et al. [53] menerapkan `learning_rate` sebesar 0,2 dan `max_depth` sebesar 5 pada dataset CSE-CIC-IDS-2018, tepat di dalam rentang pencarian penelitian ini. Ketiga, rentang `subsample` dan `colsample_bytree` (0,6 hingga 1,0) dipilih agar mencakup baik penggunaan seluruh data/fitur (nilai 1,0) maupun *subsampling* sebagian (nilai 0,6) sebagai regularisasi tambahan, sejalan dengan pendekatan Rahal et al. [27].

Sebagai validasi tambahan, konfigurasi optimal yang ditemukan Praharsha et al. [54] pada deteksi *botnet* jaringan IoT (`n_estimators` = 100, `min_samples_split` = 10, `min_samples_leaf` = 4, `max_features` = *sqrt* untuk RF, serta `n_estimators` = 100, `subsample` = 1,0, `learning_rate` = 0,1 untuk XGBoost) seluruhnya berada di dalam ruang pencarian penelitian ini, memperkuat keyakinan bahwa ruang pencarian yang ditetapkan cukup representatif dan cukup luas untuk mengakomodasi karakteristik khusus dataset UGRansome2024.

Nilai-nilai spesifik di atas tidak dimaksudkan sebagai satu-satunya konfigurasi optimal yang mungkin ada, melainkan representasi rentang yang secara umum dianggap masuk akal (*reasonable search space*) [52, 53, 54, 27]. Nilai

yang lebih ekstrem, seperti `max_depth` XGBoost di atas 9 atau `n_estimators` di atas 500, sengaja tidak dimasukkan karena pertimbangan efisiensi komputasi, mengingat proses *RandomizedSearchCV* dengan 30 iterasi dan 5-*fold cross-validation* sudah memerlukan total 150 kali pelatihan model untuk setiap algoritma. Nilai `learning_rate` di bawah 0,01 juga tidak disertakan karena akan memerlukan jumlah pohon yang jauh lebih banyak untuk konvergensi, kurang efisien dalam konteks *RandomizedSearchCV* yang bersifat pencarian acak dengan iterasi terbatas.

Setiap konfigurasi dievaluasi menggunakan 30 iterasi acak dengan validasi 5-*fold*. Jumlah *fold* yang lebih kecil dipilih khusus untuk tahap pencarian *hyperparameter* ini karena pertimbangan efisiensi komputasi: dengan 30 iterasi, total pelatihan model yang harus dijalankan adalah $30 \times 5 = 150$ kali, dibandingkan $30 \times 10 = 300$ kali apabila digunakan 10-*fold*. Karena tujuan tahap ini hanya untuk menemukan kandidat kombinasi parameter yang mendekati optimal, bukan estimasi performa akhir yang presisi, 5-*fold* dinilai memadai tanpa membebani waktu komputasi [27]. Estimasi performa akhir yang lebih presisi kemudian diperoleh melalui *10-fold Stratified Cross-Validation* yang terpisah, sebagaimana diuraikan pada Bagian 2.2.11.

2.2.11 *Cross-Validation* dan Stratified K-Fold

Cross-validation adalah teknik evaluasi yang mempartisi dataset menjadi *k fold* yang sama besar, kemudian secara iteratif melatih model pada $k-1$ *fold* dan mengevaluasinya pada *fold* yang tersisa, hingga setiap *fold* digunakan tepat sekali sebagai data evaluasi [25]. Metode ini memberikan estimasi generalisasi yang lebih reliabel dibandingkan pembagian data tunggal (*holdout*), terutama pada dataset dengan ukuran terbatas.

Stratified K-Fold adalah varian *cross-validation* yang mempertahankan proporsi kelas pada setiap *fold* sesuai distribusi kelas di dataset asli [28], kritis pada dataset *imbalanced* seperti UGRansome2024. Pada penelitian ini diterapkan *10-fold Stratified K-Fold* dengan metrik *accuracy* sebagai skor utama, menghasilkan 10 nilai performa independen untuk uji signifikansi statistik [27]. Jumlah *fold* yang lebih besar (10-*fold*) dipilih khusus untuk tahap evaluasi akhir ini, berbeda dari 5-*fold* pada tahap *hyperparameter tuning*, karena dua alasan. Pertama, dengan proporsi data uji per *fold* yang lebih kecil, estimasi performa memiliki varians yang lebih rendah dan lebih stabil. Kedua, skema 10-*fold* menghasilkan tepat 10 pasang skor akurasi yang menjadi jumlah sampel input bagi uji statistik *Wilcoxon signed-*

rank test [27].

Secara ringkas, perbedaan antara kedua tahap terletak pada tujuan, data, dan keluarannya. 5-*fold* pada tahap *RandomizedSearchCV* berfungsi sebagai validasi internal untuk memilih kandidat *hyperparameter* terbaik, dilakukan pada data latih hasil SMOTE, dan menghasilkan satu skor rata-rata. Sementara itu, 10-*fold Stratified Cross-Validation* berfungsi sebagai evaluasi akhir performa model dengan konfigurasi tetap, dilakukan pada dataset asli tanpa SMOTE agar tetap objektif, dan menghasilkan sepuluh skor individual sebagai masukan uji *Wilcoxon signed-rank test*.

2.2.12 Uji Statistik Wilcoxon *Signed-Rank*

Uji Wilcoxon *signed-rank* adalah uji statistik non-parametrik yang digunakan untuk membandingkan dua sampel berpasangan tanpa mengasumsikan distribusi normal pada data [27], membandingkan skor *cross-validation* dari dua algoritma pada *fold* yang sama sehingga perbedaan antar-*fold* dikontrol secara eksperimental [25].

Perbandingan akurasi rata-rata antar model belum cukup untuk menyimpulkan bahwa satu algoritma secara konsisten lebih unggul, karena selisihnya berpotensi dipengaruhi variasi acak pada partisi data tertentu. Uji Wilcoxon *signed-rank test* memeriksa konsistensi arah perbedaan performa pada setiap pasangan *fold* secara individual, memberikan bukti yang lebih meyakinkan dibandingkan sekadar membandingkan dua nilai rata-rata akhir. Penerapan uji signifikansi statistik semacam ini masih relevan pada penelitian keamanan siber terkini [55], sekaligus menjadi kontribusi metodologis penelitian ini terhadap *gap* penelitian terdahulu yang hanya melaporkan akurasi mentah tanpa validasi statistik [20, 21].

Hipotesis Statistik yang diuji adalah:

1. H_0: Tidak terdapat perbedaan yang signifikan secara statistik antara performa *Random Forest* dan XGBoost ($p \geq 0.05$).
2. H_1: Terdapat perbedaan yang signifikan secara statistik antara performa *Random Forest* dan XGBoost ($p < 0.05$).

H_0 dan H_1 di atas merupakan hipotesis statistik yang melekat pada prosedur uji Wilcoxon *signed-rank test*, berbeda dari hipotesis penelitian (*research hypothesis*) yang umumnya dinyatakan pada Bab 1 untuk penelitian yang bersifat

asosiatif atau korelasional. Karena penelitian ini bersifat komparatif-eksperimental, rumusan masalah dan tujuan penelitian pada Bab 1 sudah cukup untuk mengarahkan penelitian, sehingga hipotesis statistik ini lebih tepat diuraikan di sini sebagai bagian dari landasan teori metode Wilcoxon.

Statistik uji W dihitung dari selisih berpasangan $d_i = x_i - y_i$ antara skor *cross-validation* kedua algoritma pada *fold* ke-i. Setiap $|d_i|$ diberi peringkat (*rank*) R_i, kemudian W dihitung sebagai jumlah peringkat terkecil di antara jumlah peringkat pada selisih positif (W^+) dan negatif (W^-), sebagaimana ditunjukkan pada Persamaan 2.15.

$$W = \min(W^+, W^-), \quad W^+ = \sum_{d_i>0} R_i, \quad W^- = \sum_{d_i<0} R_i \tag{2.15}$$

Statistik ini dihitung berdasarkan peringkat dari selisih absolut antar pasangan, dengan menghiraukan pasangan yang selisihnya nol [27]. Besaran efek (*effect size*) diukur menggunakan koefisien Cohen's d sebagaimana ditunjukkan pada Persamaan 2.16, di mana $\bar{\Delta}$ adalah rata-rata selisih performa antar-*fold* dan σ_Δ adalah deviasi standarnya.

$$d = \frac{\bar{\Delta}}{\sigma_\Delta} \tag{2.16}$$

Interpretasi nilai Cohen's d adalah sebagai berikut: $|d| < 0.2$ menunjukkan efek kecil, $0.2 \leq |d| < 0.5$ menunjukkan efek sedang, dan $|d| \geq 0.5$ menunjukkan efek besar [25].

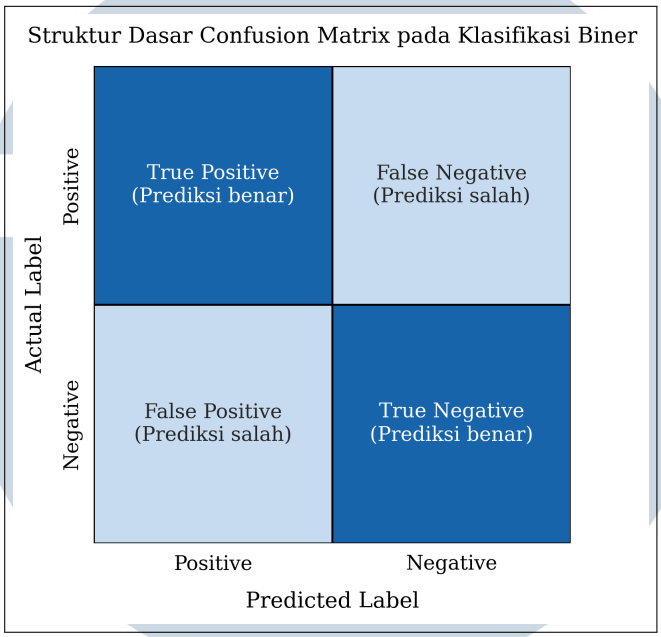
2.2.13 Metrik Evaluasi Klasifikasi

Evaluasi komprehensif model klasifikasi memerlukan penggunaan beberapa metrik yang melengkapi satu sama lain, terutama pada kondisi *class imbalance* di mana *accuracy* tunggal dapat menyesatkan [27].

A *Confusion Matrix*

Confusion matrix adalah tabel yang merangkum hasil prediksi model klasifikasi dengan membandingkan label sebenarnya (*actual/true label*) terhadap label hasil prediksi (*predicted label*) [25]. Pada klasifikasi biner, *confusion*

matrix menghasilkan empat komponen utama, yaitu *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN), yang menjadi dasar perhitungan seluruh metrik evaluasi pada bagian berikutnya [27].



Gambar 2.1. Struktur Dasar *Confusion Matrix* pada Klasifikasi Biner

Pada klasifikasi multi-kelas seperti yang diterapkan dalam penelitian ini (kelas S, A, dan SS), *confusion matrix* berbentuk matriks $K \times K$, dengan K adalah jumlah kelas. Elemen diagonal merepresentasikan prediksi yang benar untuk kelas tertentu, sedangkan elemen di luar diagonal merepresentasikan *misclassification* antar pasangan kelas [25]. Nilai TP, TN, FP, dan FN untuk setiap kelas dihitung menggunakan pendekatan *One-vs-Rest* [27].

B Akurasi (*Accuracy*)

Akurasi mengukur proporsi keseluruhan prediksi yang benar terhadap total sampel [25], sebagaimana ditunjukkan pada Persamaan 2.17.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.17}$$

C *Precision*

Precision mengukur presisi prediksi positif, yaitu proporsi prediksi positif yang benar-benar merupakan kelas positif [25], sebagaimana ditunjukkan pada Persamaan 2.18 untuk satu kelas k.

$$Precision_k = \frac{TP_k}{TP_k + FP_k} \tag{2.18}$$

Karena penelitian ini bersifat klasifikasi multi-kelas, nilai *Precision* akhir yang dilaporkan merupakan rata-rata tertimbang (*weighted average*) dari *Precision* setiap kelas, dengan bobot berupa proporsi jumlah sampel (*support*) kelas k terhadap total sampel n, sebagaimana ditunjukkan pada Persamaan 2.19 [27].

$$Precision_{weighted} = \sum_{k=1}^{K} \frac{n_k}{n} \cdot Precision_k \tag{2.19}$$

D *Recall*

Recall (sensitivitas) mengukur kemampuan model mendeteksi seluruh sampel positif yang sesungguhnya [25], sebagaimana ditunjukkan pada Persamaan 2.20 untuk satu kelas k.

$$Recall_k = \frac{TP_k}{TP_k + FN_k} \tag{2.20}$$

Serupa dengan *Precision*, nilai *Recall* akhir yang dilaporkan juga merupakan rata-rata tertimbang berdasarkan *support*-nya, sebagaimana ditunjukkan pada Persamaan 2.21.

$$Recall_{weighted} = \sum_{k=1}^{K} \frac{n_k}{n} \cdot Recall_k \tag{2.21}$$

E F1-*Score*

F1-*score* adalah *harmonic mean* dari *precision* dan *recall*, memberikan ukuran seimbang antara keduanya [25], sebagaimana ditunjukkan pada

Persamaan 2.22 untuk satu kelas k.

$$F1\text{-}Score_k = 2 \cdot \frac{Precision_k \times Recall_k}{Precision_k + Recall_k} \tag{2.22}$$

Untuk klasifikasi multi-kelas, versi *weighted* digunakan dengan pembobotan berdasarkan *support* setiap kelas k, sejalan dengan perhitungan *Precision* dan *Recall* sebelumnya [27], sebagaimana ditunjukkan pada Persamaan 2.23.

$$F1\text{-}Score_{weighted} = \sum_{k=1}^{K} \frac{n_k}{n} \cdot F1\text{-}Score_k \tag{2.23}$$

F ROC-AUC

Area Under the Receiver Operating Characteristic Curve (ROC-AUC) mengukur kemampuan diskriminasi model terhadap pasangan kelas, dengan nilai 1.0 merepresentasikan diskriminasi sempurna dan 0.5 prediksi acak [21]. Untuk klasifikasi multi-kelas, strategi *One-vs-Rest* (OvR) digunakan untuk menghitung ROC-AUC rata-rata tertimbang di semua kelas [27].

Tabel 2.3. Interpretasi nilai ROC-AUC sebagai panduan evaluasi model.

Nilai ROC-AUC	**Interpretasi**
$0.90 - 1.00$	*Excellent*
$0.80 - 0.90$	*Good*
$0.70 - 0.80$	*Fair*
$0.60 - 0.70$	*Poor*
$0.50 - 0.60$	*Failed*

Sumber: [25]

G *Average Precision* (AP)

Average Precision (AP) merupakan metrik yang merangkum kurva *Precision-Recall* ke dalam satu nilai tunggal, dihitung sebagai jumlah tertimbang dari *precision* pada setiap ambang batas n, sebagaimana ditunjukkan pada Persamaan 2.24, di mana P_n dan R_n adalah *precision* dan *recall* pada ambang batas ke-n [27].

$$AP = \sum_{n} (R_n - R_{n-1}) \cdot P_n \tag{2.24}$$

AP lebih informatif dibandingkan ROC-AUC pada kondisi *class imbalance*, karena secara eksplisit memperhitungkan proporsi prediksi positif yang benar pada kelas minoritas, sedangkan ROC-AUC dapat memberikan gambaran yang terlalu optimistis pada dataset yang tidak seimbang [25].

2.2.14 *Feature Importance* dan Analisis Interpretabilitas

Interpretabilitas model merupakan aspek kritis dalam konteks keamanan siber karena analis keamanan perlu memahami alasan di balik keputusan klasifikasi untuk mengambil tindakan respons yang tepat [27]. Dua pendekatan *feature importance* digunakan dalam penelitian ini.

Built-in feature importance pada *Random Forest* dihitung berdasarkan penurunan rata-rata impuritas Gini yang dikontribusikan setiap fitur di seluruh pohon dalam *ensemble*, sebagaimana ditunjukkan pada Persamaan 2.25, di mana $p(v)$ adalah proporsi sampel yang mencapai simpul v dan $\Delta I(v)$ adalah penurunan impuritas pada simpul tersebut [45].

$$\text{Importance}(j) = \frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{\text{node } v: \\ \text{split on } j}} p(v) \cdot \Delta I(v) \tag{2.25}$$

Untuk XGBoost, *built-in importance* dihitung berdasarkan frekuensi penggunaan fitur dalam pemisahan pohon atau rata-rata penurunan *gain* yang dikontribusikannya [46].

Permutation importance adalah metode yang lebih *robust* dan tidak bias terhadap kardinalitas fitur [27], mengukur penurunan kinerja model ketika nilai suatu fitur diacak (*permuted*), sehingga memutus hubungan antara fitur tersebut dengan variabel target. Kelebihan utamanya adalah bebas dari asumsi distribusi fitur dan lebih dapat dipercaya pada fitur dengan kardinalitas tinggi [25].

Secara formal, *permutation importance* untuk fitur j dihitung sebagai selisih antara skor performa model pada data asli dan skor performa setelah nilai fitur j diacak, sebagaimana ditunjukkan pada Persamaan 2.26, di mana s adalah metrik performa (misalnya *accuracy*), $\mathscr{D}$ adalah data uji asli, dan $\mathscr{D}_{\pi_j}$ adalah data uji dengan nilai kolom j diacak.

$$PI_j = s(\mathscr{D}) - s(\mathscr{D}_{\pi_j}) \tag{2.26}$$

Nilai PI_j dihitung berulang sebanyak n_{repeats} kali untuk menghasilkan estimasi rata-rata yang lebih stabil.

2.3 Framework dan Metodologi

Bagian ini menjelaskan kerangka kerja menyeluruh yang menjadi acuan dalam menjalankan tahapan penelitian secara sistematis, mulai dari pemilihan data hingga interpretasi hasil model. Kerangka kerja ini berfungsi sebagai peta jalan yang memastikan setiap tahapan penelitian, mulai dari eksplorasi data hingga evaluasi akhir, dilakukan secara terstruktur dan dapat dipertanggungjawabkan secara metodologis.

2.3.1 *Knowledge Discovery in Databases* (KDD)

Knowledge Discovery in Databases (KDD) pertama kali didefinisikan secara formal oleh Fayyad, Piatetsky-Shapiro, dan Smyth [56] sebagai proses non-trivial untuk mengidentifikasi pola yang valid, baru (*novel*), berpotensi berguna, dan pada akhirnya dapat dipahami (*understandable*) dari data. Suatu pola dapat dikategorikan sebagai pengetahuan (*knowledge*) apabila memenuhi keempat kriteria tersebut secara bersamaan [56]. Fayyad et al. [56] juga menegaskan bahwa KDD merupakan proses menyeluruh untuk menemukan pengetahuan yang berguna dari data, sedangkan *data mining* hanyalah satu langkah di dalam keseluruhan proses tersebut, yaitu langkah penerapan algoritma untuk mengekstraksi pola.

KDD dipilih sebagai kerangka kerja penelitian ini berdasarkan tiga pertimbangan. Pertama, KDD dirancang khusus untuk konteks riset akademik yang berorientasi pada penemuan pengetahuan secara sistematis dan dapat direproduksi [56], sejalan dengan tujuan penelitian ini yang tidak hanya membangun model klasifikasi, tetapi juga mengungkap pola dan pengetahuan baru mengenai karakteristik deteksi *ransomware*. Kedua, KDD telah banyak diadopsi pada penelitian keamanan siber berbasis *machine learning* yang serupa [25, 27], memudahkan perbandingan metodologis dengan literatur yang ada. Ketiga, sifat data penelitian ini yang multi-domain (menggabungkan fitur jaringan dan transaksi *blockchain*) serta memerlukan persiapan data yang ekstensif sangat sesuai dengan

penekanan KDD terhadap tahapan persiapan data yang matang sebelum *data mining* [56].

A Sembilan Langkah Asli Proses KDD

Fayyad et al. [56] menguraikan proses KDD secara rinci ke dalam sembilan langkah berikut:

1. *Developing an understanding of the application domain*: mengembangkan pemahaman terhadap domain aplikasi dan pengetahuan awal yang relevan, serta mengidentifikasi tujuan proses KDD dari sudut pandang pengguna.
2. *Creating a target data set*: memilih dataset atau berfokus pada subset variabel/sampel data yang menjadi objek penemuan.
3. *Data cleaning and preprocessing*: membersihkan data dari *noise*, menangani *missing data*, serta memperhitungkan informasi *time-sequence*.
4. *Data reduction and projection*: menemukan fitur yang berguna untuk merepresentasikan data, melalui reduksi dimensi atau transformasi.
5. *Matching the goals of the KDD process to a particular data-mining method*: mencocokkan tujuan proses KDD dengan metode *data mining* tertentu (klasifikasi, regresi, *clustering*, dan sebagainya).
6. *Exploratory analysis and model and hypothesis selection*: memilih algoritma dan metode pencarian pola yang akan digunakan.
7. *Data mining*: pencarian pola yang sesungguhnya dalam bentuk representasional tertentu, seperti aturan klasifikasi, pohon keputusan, regresi, atau *clustering*.
8. *Interpreting mined patterns*: menginterpretasikan pola yang ditemukan, yang dapat melibatkan kembali ke langkah 1 hingga 7 untuk iterasi lebih lanjut, termasuk visualisasi pola dan model.
9. *Acting on the discovered knowledge*: menindaklanjuti pengetahuan yang ditemukan, baik dengan menggunakannya secara langsung, mengintegrasikannya ke sistem lain, maupun mendokumentasikan dan melaporkannya kepada pihak-pihak yang berkepentingan.

B Penyesuaian Tahapan KDD pada Penelitian Ini

Kesembilan langkah di atas pada praktiknya jarang diterapkan secara harfiah satu per satu dalam penelitian terapan, karena beberapa langkah bersifat konseptual/perencanaan (langkah 1, 5, dan 6) yang secara alami menyatu dengan langkah-langkah operasional di sekitarnya. Kurgan dan Musilek [57] menemukan bahwa sembilan langkah asli Fayyad et al. umum diadaptasi menjadi tahapan yang lebih spesifik sesuai kebutuhan domain penelitian, tanpa mengurangi esensi maupun urutan logis dari proses aslinya. Oleh karena itu, penelitian ini mengadaptasi kesembilan langkah tersebut menjadi sembilan tahap operasional yang sesuai dengan alur penelitian yang dijalankan (*Data Collection*, *Exploratory Data Analysis*, *Data Preprocessing*, *Feature Engineering*, *Data Preparation*, *Model Training*, *Model Evaluation*, *Result Analysis*, dan *Research Conclusion & Recommendation*), sebagaimana dirangkum pemetaannya pada Tabel 2.4.

Tabel 2.4. Pemetaan Sembilan Langkah Asli KDD ke Tahapan Penelitian Ini

No.	**Langkah Asli (Fayyad et al., 1996)**	**Tahap Penelitian Ini**	**Alasan Penyesuaian**
1	*Developing an understanding of the application domain*	*(mendasari seluruh tahap)*	Sudah tercakup dalam Latar Belakang (Bab 1) dan Tinjauan Teori (Bab 2), sehingga tidak diulang sebagai tahap operasional terpisah
2	*Creating a target data set*	*Data Collection*	Diadopsi sebagai tahap pemuatan dan validasi awal dataset UGRansome2024

Bersambung ke halaman berikutnya

Tabel 2.4 Pemetaan Sembilan Langkah Asli KDD ke Tahapan Penelitian Ini (lanjutan)

No.	**Langkah Asli (Fayyad et al., 1996)**	**Tahap Penelitian Ini**	**Alasan Penyesuaian**
3	*Data cleaning and preprocessing*	*Exploratory Data Analysis* dan *Data Preprocessing*	Dipecah menjadi dua tahap terpisah: EDA untuk mengidentifikasi kebutuhan pembersihan data (*missing value*, duplikat, anomali), dan *Data Preprocessing* untuk eksekusi pembersihan serta *label encoding*
4	*Data reduction and projection*	*Feature Engineering* dan *Data Preparation*	Diadaptasi menjadi pembangunan fitur derivatif baru (*Feature Engineering*) serta analisis korelasi untuk identifikasi multikolinearitas (*Data Preparation*)
5	*Matching goals to data-mining method*	*(mendasari Model Training)*	Bersifat perencanaan konseptual (pemilihan RF/XGBoost sebagai metode klasifikasi) yang sudah dijustifikasi pada Bab 1–2, sehingga tidak diulang sebagai tahap terpisah
6	*Exploratory analysis and model/hypothesis selection*	*(mendasari Model Training)*	Bersifat perencanaan (desain ruang pencarian *hyperparameter*) yang menyatu dengan tahap *Model Training* sebagai bagian tak terpisahkan dari pembangunan model

Bersambung ke halaman berikutnya

Tabel 2.4 Pemetaan Sembilan Langkah Asli KDD ke Tahapan Penelitian Ini (lanjutan)

No.	Langkah Asli (Fayyad et al., 1996)	Tahap Penelitian Ini	Alasan Penyesuaian
7	*Data mining*	*Model Training*	Direalisasikan sebagai tahap pembangunan model *baseline*, *hyperparameter tuning*, dan *cross-validation* untuk RF dan XGBoost
8	*Interpreting mined patterns*	*Model Evaluation* dan *Result Analysis*	Dipecah menjadi dua tahap: *Model Evaluation* untuk perhitungan metrik dan visualisasi, serta *Result Analysis* untuk sintesis dan perbandingan hasil
9	*Acting on the discovered knowledge*	*Research Conclusion & Recommendation*	Direalisasikan secara eksplisit sebagai tahap akhir yang mendokumentasikan pengetahuan (*knowledge*) yang ditemukan beserta rekomendasi tindak lanjut

Dengan demikian, sembilan tahap operasional yang dipakai penelitian ini bukan pengabaian terhadap langkah 1, 5, dan 6 dari proses asli, melainkan pengintegrasian langkah-langkah tersebut ke dalam tahap operasional yang paling relevan: langkah 1 telah dijalankan pada Bab 1 dan Bab 2, sedangkan langkah 5 dan 6 menyatu dengan tahap *Model Training* karena bersifat perencanaan pemodelan. Langkah 9 (*acting on the discovered knowledge*) direalisasikan secara eksplisit sebagai tahap tersendiri, yaitu *Research Conclusion & Recommendation*, yang menjadi wujud konkret dari esensi *knowledge engineering* dalam KDD bahwa proses tidak berhenti pada evaluasi model semata, melainkan harus berlanjut hingga pendokumentasian dan pemanfaatan pengetahuan yang ditemukan.

Pada tahap *Research Conclusion & Recommendation*, pengetahuan

(*knowledge*) yang dihasilkan dari keseluruhan proses KDD penelitian ini berupa simpulan keunggulan algoritma serta rekomendasi tindak lanjut yang telah dijawab secara rinci pada Bab 4 (Hasil dan Pembahasan) dan Bab 5 (Simpulan dan Saran). Realisasi tahap ini pada kedua bab tersebut memenuhi keempat kriteria pengetahuan menurut Fayyad et al. [56], yaitu *valid* (dikonfirmasi melalui uji statistik), *novel* (temuan baru pada dataset yang diteliti), *useful* (memberi acuan praktis), dan *understandable* (disajikan melalui metrik dan visualisasi yang mudah diinterpretasikan).

2.4 *Tools* yang Digunakan

Bagian ini menguraikan perangkat lunak dan pustaka (*library*) yang digunakan dalam mendukung keseluruhan proses penelitian, mulai dari pengolahan data, pembangunan model, penanganan ketidakseimbangan kelas, hingga pengujian statistik. Pemilihan setiap *tools* didasarkan pada kesesuaiannya dengan kebutuhan teknis penelitian serta dukungan luas yang dimilikinya dalam komunitas ilmiah dan industri.

2.4.1 Python

Python adalah bahasa pemrograman tingkat tinggi yang bersifat *open source*, dikenal luas dalam komunitas ilmiah dan industri karena keterbacaan sintaksnya dan ekosistem pustaka yang kaya [25]. Dalam penelitian ini, Python versi 3.x digunakan sebagai lingkungan pemrograman utama.

2.4.2 Scikit-learn

Scikit-learn adalah pustaka *machine learning* komprehensif untuk Python yang menyediakan implementasi efisien dari ratusan algoritma pembelajaran, termasuk *Random Forest*, berbagai metode evaluasi, dan utilitas *preprocessing* [27]. Penelitian ini memanfaatkan `RandomForestClassifier` dengan dukungan paralelisasi (`n_jobs=-1`), `StratifiedKFold`, `RandomizedSearchCV`, serta modul `permutation_importance` untuk analisis interpretabilitas.

2.4.3 XGBoost

XGBoost adalah pustaka *gradient boosting* yang dioptimalkan secara komputasional dan tersedia sebagai pustaka Python mandiri [46]. Implementasinya mendukung paralelisasi *multi-threading* dan penanganan *sparse data* secara *native*. Parameter `eval_metric='mlogloss'` digunakan untuk kompatibilitas dengan versi terbaru dan dataset multi-kelas.

2.4.4 Imbalanced-learn

Imbalanced-learn adalah pustaka Python yang terintegrasi dengan ekosistem Scikit-learn dan menyediakan berbagai teknik penanganan *class imbalance*, termasuk SMOTE dan variannya [28]. Pustaka ini memungkinkan integrasi SMOTE dalam *pipeline* yang mencegah kebocoran data antara *fold* dalam *cross-validation* [47].

2.4.5 Jupyter Notebook

Jupyter Notebook adalah aplikasi *web-based* berbasis *open source* yang memungkinkan pembuatan dokumen interaktif yang menggabungkan kode eksekusi, visualisasi, dan narasi ilmiah dalam satu berkas [25]. Format `.ipynb` yang digunakan mendukung reprodusibilitas penelitian dan memfasilitasi dokumentasi alur kerja analisis secara komprehensif.

2.4.6 SciPy

SciPy adalah pustaka komputasi ilmiah untuk Python yang menyediakan modul-modul untuk optimasi, integrasi, interpolasi, dan statistik [25]. Dalam penelitian ini, modul `scipy.stats` dimanfaatkan untuk implementasi uji Wilcoxon *signed-rank* melalui fungsi `stats.wilcoxon()` [27].