DETEKSI *INSIDER ATTACK* MENGGUNAKAN PENDEKATAN BERBASIS ALGORITMA *XGBOOST* DENGAN *TUNING HYPERPARAMETER GRID SEARCH*

TUGAS AKHIR

Disusun Oleh :

IGNASIUS SATRIA WICAKSANA

123210020

PROGRAM STUDI INFORMATIKA

JURUSAN INFORMATIKA

FAKULTAS TEKNIK INDUSTRI

UNIVERSITAS PEMBANGUNAN NASIONAL "VETERAN"

YOGYAKARTA

2026

DETEKSI INSIDER ATTACK MENGGUNAKAN PENDEKATAN BERBASIS ALGORITMA XGBOOST DENGAN TUNING HYPERPARAMETER GRID SEARCH

TUGAS AKHIR

Tugas Akhir ini sebagai salah satu syarat untuk memperoleh gelar sarjana Informatika
Universitas Pembangunan Nasional "Veteran" Yogyakarta

Disusun Oleh :

IGNASIUS SATRIA WICAKSANA

123210020

PROGRAM STUDI INFORMATIKA

JURUSAN INFORMATIKA

FAKULTAS TEKNIK INDUSTRI

UNIVERSITAS PEMBANGUNAN NASIONAL "VETERAN"

YOGYAKARTA

2026

HALAMAN PENGESAHAN PEMBIMBING

DETEKSI *INSIDER ATTACK* MENGGUNAKAN PENDEKATAN BERBASIS ALGORITMA XGBOOST DENGAN *TUNING HYPERPARAMETER GRID SEARCH*

Disusun Oleh:
Ignasius Satria Wicaksana
123210020

Telah diuji dan dinyatakan lulus oleh pembimbing

pada tanggal: 7 Mei 2026

Menyetujui,

Pembimbing

Andrey Ferriyan S.T., M.Cs., Ph.D.
NUPTK. 1255761662130193

Mengetahui,

Koordinator Program Studi

Dessyanto Boedi Prasetyo, S.T., M.T.
NIDN. 0505127501

HALAMAN PENGESAHAN PENGUJI

DETEKSI *INSIDER ATTACK* MENGGUNAKAN PENDEKATAN BERBASIS ALGORITMA XGBOOST DENGAN *TUNING HYPERPARAMETER GRID SEARCH*

Disusun Oleh:
Ignasius Satria Wicaksana
123210020

Telah diuji dan dinyatakan lulus oleh penguji
pada tanggal: 7 Mei 2026

Menyetujui,

Penguji 1

Andrey Ferriyan, S.T., M.Cs., Ph.D.
NUPTK. 1255761662130193

Penguji 2

Simon Pulung Nugroho, S.Kom., M.Cs.
NIDN. 0518028401

Penguji 3

Rifki Indra Perwira, S.Kom., M.Eng.
NIDN. 0508078301

Penguji 4

Dessyanto Boedi Prasetyo, S.T., M.T.
NIDN. 0505127501

SURAT PERNYATAAN
KARYA ASLI TUGAS AKHIR

Sebagai mahasiswa Program Studi Informatikai Fakultas Teknik Industri Universitas Pembangunan Nasional "Veteran" Yogyakarta, yang bertanda tangan di bawah ini, saya:

Nama : Ignasius Satria Wicaksana
NIM : 123210020

Menyatakan bahwa karya ilmiah saya yang berjudul:
Deteksi *Insider Attack* menggunakan Pendekatan Berbasis Algoritma *XGBoost* dengan *Tuning Hyperparameter Grid Search*
Merupakan karya asli saya dan belum pernah dipublikasikan dimanapun. Apabila di kemudian hari, karya saya disinyalir bukan merupakan karya asli saya, maka saya bersedia menerima konsekuensi apa pun yang diberikan Program Studi Informatika Fakultas Teknik Industri Universitas Pembangunan Nasional "Veteran" Yogyakarta kepada saya.

Demikian surat pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Yogyakarta
Pada Tanggal : 10 Februari 2026

Yang Menyatakan

Ignasius Satria Wicaksana
NIM. 123210020

PERNYATAAN BEBAS PLAGIASI

Saya yang bertanda tangan di bawah ini:

Nama : Ignasius Satria Wicaksana
NIM : 123210020
Prodi : Informatika

Dengan ini saya menyatakan bahwa judul Tugas Akhir
Deteksi *Insider Attack* menggunakan Pendekatan Berbasis Algoritma *XGBoost* dengan *Tuning Hyperparameter Grid Search*
adalah hasil kerja saya dan benar bebas dari plagiasi kecuali cuplikan serta ringkasan yang terdapat di dalamnya telah saya jelaskan sumbernya (Sitasi) dengan jelas. Apabila pernyataan ini terbuktu tidak benar maka saya bersedia menerima sanksi sesuai peraturan Mendiknas RI No 17 Tahun 2010 dan Peraturan Perundang-undangan yang berlaku.

Demikian surat pernyataan ini saya buat dengan penuh tanggung jawab.

Dibuat di : Yogyakarta
Pada Tanggal : 12 Maret 2026

Yang Menyatakan

Ignasius Satria Wicaksana
NIM. 123210020

ABSTRAK

Ancaman *insider attack* merupakan salah satu risiko keamanan paling kompleks dalam lingkungan organisasi karena pelakunya berasal dari internal dan memiliki akses sah terhadap sistem. Pola serangan insider threat sering kali bersifat tersembunyi, bertahap, dan sulit dibedakan dari aktivitas normal pengguna, sehingga pendekatan berbasis aturan statis tidak lagi memadai. Selain itu, karakteristik data keamanan yang tidak seimbang serta heterogenitas sumber log, seperti aktivitas *login*, akses *file*, penggunaan perangkat, dan komunikasi jaringan, semakin menambah tantangan dalam mendeteksi serangan *insider* secara akurat. Oleh karena itu, diperlukan pendekatan analitis yang mampu menangkap pola perilaku kompleks serta memiliki kemampuan generalisasi yang baik terhadap data dunia nyata.

Penelitian ini mengusulkan pendekatan deteksi insider threat menggunakan algoritma *Extreme Gradient Boosting* (*XGBoost*) yang dikenal efektif dalam menangani data berdimensi tinggi dan tidak seimbang. Dataset yang digunakan berasal dari *CERT Insider Threat Dataset*, yang mencakup berbagai jenis aktivitas pengguna dalam lingkungan organisasi. Tahapan penelitian meliputi pengumpulan data, *preprocessing*, pembagian data, serta pelatihan dan pengujian model. Evaluasi model dilakukan menggunakan skema *Stratified K-Fold Cross Validation* untuk menjaga proporsi kelas pada setiap *fold*, dengan metrik evaluasi yang berfokus pada kemampuan model dalam menekan tingkat *false positive* dan *false negative*.

Hasil pengujian menunjukkan bahwa model *XGBoost* mampu mencapai performa yang sangat baik ketika dilatih dan diuji pada distribusi data yang sama, khususnya pada dataset r4.2 dan r5.2 dengan nilai *F1-Score* dan *AUC-ROC* yang tinggi. Penerapan *SMOTE* terbukti efektif dalam meningkatkan kemampuan model mendeteksi kelas minoritas melalui peningkatan *recall*, meskipun terdapat *trade-off* berupa penurunan *precision*. Namun, pada dataset r6.2 yang memiliki ketidakseimbangan kelas lebih ekstrem, model gagal menghasilkan prediksi positif pada *threshold* default meskipun nilai AUC-ROC tinggi, yang mengindikasikan masalah pada kalibrasi threshold dan perbedaan distribusi probabilitas. Pengujian lintas versi dataset juga memperlihatkan penurunan performa yang signifikan, menandakan bahwa model masih sensitif terhadap perubahan distribusi data dan belum memiliki kemampuan generalisasi yang memadai untuk skenario deteksi *insider threat* pada lingkungan yang dinamis.

Kata Kunci: XGBoost, SMOTE, *Insider Attack*, CERT *Insider Threat*

ABSTRACT

Insider threats represent one of the most challenging security risks in organizational environments because the attackers originate from within the organization and possess legitimate system access. Insider attack behaviors are often subtle, gradual, and difficult to distinguish from normal user activities, rendering traditional rule-based detection approaches ineffective. Furthermore, the inherent characteristics of security data, such as class imbalance and the heterogeneity of log sources—including login activities, file access, device usage, and network communications—further complicate accurate detection. Consequently, there is a strong need for analytical approaches capable of modeling complex behavioral patterns while maintaining robust generalization performance in real-world scenarios.

This study proposes an insider threat detection approach based on the Extreme Gradient Boosting (XGBoost) algorithm, which has demonstrated effectiveness in handling high-dimensional and imbalanced data. The dataset used in this research is derived from the CERT Insider Threat Dataset, encompassing various types of user activities within an organizational environment. The research methodology includes data collection, preprocessing stages such as data cleaning, feature mapping, and class imbalance handling, followed by data partitioning, model training, and testing. Model evaluation is conducted using a Stratified K-Fold Cross-Validation scheme to preserve class distribution across folds, with evaluation metrics emphasizing the reduction of false positive and false negative rates.

Experimental results indicate that the XGBoost model achieves strong performance when trained and evaluated on datasets with similar distributions, particularly on versions r4.2 and r5.2, where high F1-scores and AUC-ROC values were consistently observed. The application of SMOTE proved effective in improving the detection of minority-class instances by increasing recall, although this was accompanied by a moderate reduction in precision, reflecting a typical trade-off in imbalanced classification problems. However, on dataset version r6.2, which exhibits more extreme class imbalance, the model failed to produce positive predictions at the default decision threshold despite achieving relatively high AUC-ROC values. This discrepancy suggests issues related to threshold calibration and skewed probability distributions rather than an inability of the model to learn discriminative patterns. Cross-version evaluation further revealed significant performance degradation, indicating that the model remains sensitive to distributional shifts and lacks sufficient generalization capability for deployment in dynamic real-world insider threat detection scenarios.

Keywords: *XGBoost, SMOTE, Insider Attack, CERT Insider Threat*

KATA PENGANTAR

Puji syukur kepada Tuhan Yesus Kristus berkat kasih, dan karunia-Nya penulis dapat menyelesaikan Tugas Akhir dengan judul "Deteksi *Insider Attack* Menggunakan Pendekatan Berbasis Algoritma XGBoost dengan *Tuning Hyperparameter Grid Search*" ini dengan baik. Proses penulisan skripsi ini tidak lepas dari bantuan dan dukungan berbagi pihak. Oleh karena itu, penulis hendak mengucapkan rasa terima kasih kepada:

1. Bapak Dr. Heriyanto, A.Md.,S.Kom.,M.Cs. selaku Ketua Jurusan Informatika UPN "Veteran" Yogyakarta,
2. Bapak Andrey Ferriyan S.T., M.Cs., Ph.D. selaku Dosen Pembimbing Tugas Akhir atas segala bentuk bimbingan, arahan, serta motivasi yang telah diberikan kepada penulis,
3. Bapak Simon Pulung Nugroho, S.Kom., M.Cs., Bapak Rifki Indra Perwira, S.Kom., M.Eng., dan Bapak Dessyanto Boedi Prasetyo, S.T, M.T. selaku dosen penguji Tugas Akhir atas arahan dan saran yang telah diberikan kepada penulis,
4. Bapak Dr. Awang Hendrianto Pratomo., S.T., M.T. selaku dosen wali yang telah banyak memberikan bantuan selama perjalanan studi penulis di UPN "Veteran" Yogyakarta,
5. Seluruh Staf Pengajar Jurusan Informatika UPN "Veteran" Yogyakarta atas segala ilmu pengetahuan dan bantuan yang telah diberikan kepada penulis,
6. Bapak Yohanes Nunung Dwi Saputro, Ibu Laurentia Twedi Lusiani, Adik Christina Anindya Kartika yang senantiasa mendukung baik secara material maupun non material sehingga penulis dapat menjalani kegiatan studi hingga Tugas Akhir ini.
7. Keluarga besar UKM Seni UPN "Veteran" Yogyakarta dan Kelompok Studi Robotik Universitas Pembangunan "Veteran" Yogyakarta atas pengalaman berharga yang diberikan selama penulis menjalani kegiatan studi.
8. Rizqi Ananta Ekta Putra, Faris Dani Rizkianto, Rayhan Fairuz Sakha, dan Daniel Febrian Eka Wijaya yang selalu menemani setiap proses penulis selama di UPN "Veteran" Yogyakarta serta mendukung proses pengerjaan Tugas Akhir.
9. Seluruh teman-teman dan semua pihak yang tidak bisa disebutkan satu persatu yang secara langsung maupun tidak langsung membantu dan memberikan dukungan kepada penulis selama menjalani kegiatan studi.

Penulis menyadari bahwa tugas akhir ini masih memiliki berbagai keterbatasan. Oleh karena itu, penulis mengharapkan saran dan kritik yang membangun agar karya ini dapat bermanfaat dan dikembangkan lebih lanjut.

Yogyakarta, 10 Februari 2026

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang Masalah

Di era digital yang terus berkembang, jaringan komunikasi komputer merupakan salah satu penopang bagi hampir seluruh kegiatan organisasi, baik dalam sektor pemerintahan, pendidikan, bisnis, kesehatan, maupun industri.

Perkembangan yang pesat dapat membawa dampak positif maupun negatif terhadap keamanan jaringan computer (Cahyawati et al. 2023). Ancaman keamanan jaringan dapat bersumber dari luar seperti serangan siber dan juga dapat berasal dari dalam organisasi yang dapat kita sebut dengan istilah *insider attack* (Nurse et al. 2014). *Insider attack* terjadi ketika seseorang yang memiliki hak akses menyalahgunakan hak nya untuk mencuri, merusak, menyabotase, atau menyebarkan informasi sensitif yang biasanya bersifat rahasia (Al-shehari and Alsowail 2021). Kenyataannya, serangan ini sering kali lebih sulit dideteksi dan dicegah karena pelaku *insider attack* memiliki pengetahuan tentang struktur jaringan dan keamanan yang digunakan serta hak akses legal yang dapat membantu pelaku melakukan serangan secara tersembunyi dan efektif (Inayat et al. 2024). Hal ini mengakibatkan proses penanganan dan pencegahan *insider attack* menjadi lebih sulit dalam menjaga integritas, ketersediaan, serta keamanan data dalam suatu jaringan komputer.

Kebutuhan penelitian mengenai *insider attack* semakin mendesak seiring dengan meningkatnya kompleksitas jaringan internal dan keterbatasan metode deteksi tradisional. Penelitian terdahulu oleh (Tao et al. 2023) telah menggunakan deteksi berbasis perilaku pengguna melalui biometrik dinamis seperti mouse dynamics. Penelitian tersebut mengusulkan metode autentikasi pengguna orang dalam berbasis *Efficient Channel Attention - Temporal Convolutional Network* (ECA-TCN) yang efektif dalam menangkap ketergantungan waktu serta sifat nonlinier data perilaku pengguna dengan dikombinasikan dengan *One-Class* SVM bagi setiap pengguna sehingga mencapai akurasi hingga 96% AUC. Pendekatan behavioral analytics dan deep evidential clustering untuk mendeteksi ancaman internal secara *real-time*, mencapai akurasi deteksi hingga 94,7% dan mengurangi *false positive* sebesar 38% dibandingkan metode konvensional (Ali, Husain, and Hans 2025). Pendekatan berbasis *graph neural networks* (GNN) telah menunjukkan efektivitas dalam menangani data jaringan yang kompleks dan heterogen (Gong et al. 2024), yang menekankan bahwa struktur graf lebih adaptif terhadap data intranet dibandingkan metode pembelajaran mesin tradisional.

Salah satu tantangan utama dalam penelitian *insider attack* adalah keterbatasan akses terhadap data nyata yang bersifat sensitif dan rahasia. Data aktivitas internal organisasi yang mencerminkan perilaku pengguna umumnya tidak tersedia secara publik karena alasan privasi dan keamanan. Oleh karena itu, banyak penelitian memanfaatkan dataset sintetis yang dirancang menyerupai kondisi nyata, seperti dataset *Computer Emergency Response Team* (CERT) *Insider Threat* yang dikembangkan oleh *Carnegie Mellon University* (CMU).

Dataset ini mencakup berbagai jenis log aktivitas pengguna, seperti *logon*, akses *file*, *email*, *HTTP*, dan perangkat, yang merepresentasikan perilaku normal maupun serangan orang dalam dalam suatu organisasi. Penggunaan dataset ini memungkinkan evaluasi model secara terstandarisasi, meskipun memiliki keterbatasan dalam merepresentasikan kompleksitas penuh dari lingkungan dunia nyata.

Di antara berbagai algoritma pembelajaran mesin yang ada, XGBoost (Extreme Gradient Boosting) telah terbukti unggul dalam banyak studi terkait klasifikasi dalam hal keamanan jaringan. XGBoost merupakan algoritma *boosting* yang banyak diakui karena kombinasi efisiensi komputasi dan akurasi prediksinya yang tinggi. XGBoost dikenal sebagai salah satu algoritma pembelajaran mesin berbasis pohon yang sangat efisien dan akurat. Keunggulannya telah menjadikannya standar dalam banyak aplikasi pembelajaran mesin dan kompetisi data science, di mana XGBoost dapat secara iteratif meningkatkan model dengan menambahkan pohon keputusan yang berfokus pada prediksi yang salah dari iterasi sebelumnya (Ke et al. 2017). Selain itu, XGBoost dilengkapi dengan fitur regularization untuk menghindari overfitting, serta mendukung pemrosesan paralel dan menangani missing values, yang menjadikannya sangat cocok untuk diterapkan pada data besar dan kompleks seperti data perilaku pengguna.

Dalam konteks deteksi *insider attack*, kemampuan XGBoost dalam mengenali pola tersembunyi serta fleksibilitasnya dalam menangani ketidakseimbangan kelas data menjadi alasan utama pemilihannya sebagai inti dari penelitian ini. Meskipun demikian, performa XGBoost sangat bergantung pada konfigurasi *hyperparameter* yang digunakan. Pemilihan *hyperparameter* yang tidak tepat dapat menyebabkan model mengalami *overfitting* atau *underfitting*, terutama pada *dataset* dengan distribusi kelas yang tidak seimbang. Berbagai pendekatan optimasi seperti *grid search*, *random search*, maupun metode berbasis metaheuristik dapat digunakan untuk meningkatkan performa model. Studi terdahulu oleh (Kan et al. 2023) menunjukkan bahwa dengan menggunakan XGBoost dipadukan dengan strategi penyesuaian data, seperti penyaringan data mencurigakan dan *oversampling* dengan SMOTE serta optimasi *hyperparameter* dengan *Ethnic Randomized Particle Swarm Optimization* (ERPSO) dapat meningkatkan akurasi serta stabilitas model secara signifikan dalam deteksi ancaman orang dalam. Untuk memperoleh performa yang optimal, pendekatan *tuning hyperparameter* dan penanganan ketidakseimbangan kelas dengan SMOTE menjadi aspek penting dalam penelitian ini.

Diharapkan dengan adanya penelitian ini dapat ikut berkontribusi dalam mengembangkan model deteksi *insider attack* yang lebih adaptif dan akurat. Dengan memanfaatkan kemajuan dalam analisis perilaku serta pembelajaran mendalam, diharapkan dapat mengurangi false positive dan meningkatkan akurasi dibandingkan dengan metode konvensional. Penggunaan algoritma pembelajaran mesin seperti XGBoost telah terbukti dalam deteksi *insider attack*. Diharapkan model ini dapat memperkuat sistem keamanan jaringan dan menjadi langkah strategis untuk menghadapi ancaman internal dan melindungi integritas data jaringan.

1.2 Rumusan Masalah

Dari latar belakang diatas, maka rumusan masalah yang ada adalah bagaimana melakukan implementasi dan optimasi model XGBoost menggunakan *grid search* untuk mendeteksi *insider attack* pada *CERT Insider Threat Test Dataset*.

1.3 Batasan Masalah

Adapun batasan masalah pada penelitian ini berdasarkan masalah yang sudah dirumuskan sebelumnya yakni:

1. Penelitian ini menggunakan *CERT Insider Threat Test Dataset* yang diunduh dari repositori *dataset* Carnegie Mellon University. Versi *dataset* yang digunakan dalam penelitian ini adalah r4.2, r5.2 dan r6.2 dikarenakan dinilai memiliki jumlah serangan yang lebih ideal untuk digunakan dalam studi ini dibandingkan dengan versi yang lain.
2. Penelitian ini bertujuan untuk memberikan konsep dari deteksi *Insider Attack* dengan penerapan *machine learning* berbasis algoritma XGboost.
3. Penelitian ini mengklasifikasikan serangan menjadi dua kelas, yaitu ada serangan dan tidak ada serangan berdasarkan data *insider* resmi yang disediakan dalam *dataset* CERT.
4. Proses pelabelan pada penelitian ini dilakukan pada tingkat pengguna (*user-level labeling*), di mana seluruh aktivitas yang dimiliki oleh seorang pengguna diberi label *malicious* apabila pengguna tersebut teridentifikasi sebagai *insider* dalam *dataset*. Dengan demikian, penelitian ini tidak melakukan deteksi pada tingkat kejadian (*event-level*) atau waktu tertentu.

1.4 Tujuan Penelitian

- Mengimplementasikan model deteksi *insider attack* dengan pendekatan berbasis algoritma XGBoost.
- Menganalisis kinerja yang dihasilkan dari model algoritma XGBoost dalam hal identifikasi *insider attack*.

1.5 Manfaat Penelitian

- Merancang solusi untuk deteksi *insider attack* untuk meningkatkan keamanan jaringan organisasi.
- Berpartisipasi mengembangkan sistem deteksi serangan dari dalam berbasis algoritma XGBoost.

1.6 Tahapan Penelitian

Dalam bagian ini disampaikan tentang cara-cara yang digunakan dalam melakukan penelitian. Metode penelitian berisi :

1. Studi Literatur

   Melakukan studi literatur terkait *Insider Attack*, SMOTE, XGBoost dan Evaluasi untuk mencari state-of-the-art dan research gap.

2. Pengumpulan Data

   Penelitian ini menggunakan *dataset Carnegie Mellon University - Computer Emergency Response Team* (CMU-CERT). *Dataset* CMU-CERT merupakan *dataset* sintetis yang mencerminkan perilaku pengguna nyata dalam suatu organisasi.

3. Pra-Pemrosesan Data

   Data yang sudah dikumpulkan pada tahap sebelumnya tidak dapat langsung digunakan untuk pemodelan. Perlu dilakukan proses pengolahan data ekstraksi fitur serta pembagian data sebelum kemudian data digunakan untuk pemodelan.

4. Modeling

   Model XGBoost dengan kombinasi SMOTE dilatih untuk klasifikasi serangan dengan data yang sudah diproses pada tahap sebelumnya.

5. Evaluasi

   Tahap evaluasi digunakan untuk meninjau performa model yang sudah dibuat pada tahap modeling. Metrik yang digunakan pada tahap evaluasi adalah *precision, recall, f1-score*, *ROC curve*, dan *AUC ROC-score.*

1.7 Sistematika Penulisan

Bagian ini berisi struktur Tugas Akhir ini mulai Bab Pendahuluan sampai Bab Penutup dan deskripsi singkat dari masing-masing bab. Sistematika penulisan laporan pada penelitian ini adalah sebagai berikut :

1. BAB I PENDAHULUAN

   Bab I merupakan bagian pendahuluan yang berisi pembahasan mengenai research gap, metode yang digunakan, serta solusi yang ditawarkan untuk mengatasi masalah. Pada bab ini terdapat uraian mengenai latar belakang, rumusan masalah, batasan penelitian, tujuan dan manfaat penelitian, tahapan pelaksanaan, serta sistematika penulisan.

2. BAB II TINJAUAN LITERATUR

   Bab II berisi penjelasan dan pembahasan mengenai teori, konsep, model, metode, sistem, maupun analisis pustaka ilmiah yang relevan dan mendasar terhadap permasalahan yang diteliti. Dalam penelitian ini, teori yang relevan mencakup SMOTE, *Insider Attack* dan XGBoost.

3. BAB III METODOLOGI PENELITIAN

   Bab III menjelaskan tahapan pemanfaatan dataset, prapemrosesan data termasuk penerapan SMOTE untuk menangani ketidakseimbangan kelas, penerapan model XGBoost, serta rencana pengujian dan evaluasi.

4. BAB IV HASIL PENGUJIAN DAN PEMBAHASAN

   Bab IV memaparkan penerapan metodologi penelitian hingga menjadi produk akhir. Bab ini juga menyajikan hasil pelaksanaan serta pengujian metode penelitian secara sistematis dan logis. Selain itu, pembahasan implementasi mencakup analisis hasil penelitian yang diperoleh untuk menjawab rumusan masalah.

5. BAB V PENUTUP

Bab V memuat hasil penelitian yang menunjukkan kemampuan dalam menjawab rumusan masalah, sehingga dapat ditarik kesimpulan sesuai dengan tujuan penelitian. Pada bagian ini juga disampaikan saran yang dapat menjadi acuan untuk pengembangan penelitian selanjutnya.

BAB II
TINJAUAN LITERATUR

2.1 Insider Attack

Insider attack dapat didefinisikan sebagai ancaman keamanan yang dilakukan oleh individu dengan hak akses sah terhadap sistem komputer organisasi (Madan Kumar et al. 2025). Akses ini dapat digunakan oleh penyerang baik secara sengaja maupun tidak sengaja untuk merugikan kerahasiaan, integritas, atau ketersediaan data dan sumber daya digital. Ancaman ini bersifat unik karena pelaku umumnya memahami arsitektur sistem serta prosedur internal organisasi, sehingga mampu menghindari deteksi. Kondisi tersebut menjadikan *insider attack* sebagai salah satu ancaman yang paling sulit terdeteksi, khususnya pada lingkungan komputasi awan yang kompleks dan terus berubah.

Karakteristik *insider attack* mencakup beberapa aspek utama. Pertama, pelaku biasanya memiliki akses sah melalui autentikasi dan otorisasi resmi dari organisasi, sehingga dapat mengakses sistem atau data tanpa memicu kecurigaan awal. Kedua, serangan ini sulit terdeteksi karena aktivitas berbahaya yang dilakukan sering kali menyerupai perilaku normal pengguna, sehingga sulit dibedakan oleh sistem keamanan konvensional. Ketiga, pelaku umumnya memiliki pengetahuan mendalam tentang sistem, kebijakan, dan celah keamanan organisasi, yang dapat dimanfaatkan untuk melakukan serangan secara lebih efektif. Terakhir, motif pelaku beragam, mulai dari keuntungan finansial dan aksi balas dendam hingga kelalaian yang tidak disengaja, yang seluruhnya berpotensi mengancam keamanan informasi organisasi. (Singh, Pareek, and Sharma, n.d.).

Deteksi *insider attack* memiliki berbagai tantangan utama, seperti pola perilaku yang tidak terbatas dan jeda waktu yang bervariasi antar aktivitas, perubahan peran serta gaya kerja individu (non-stationarity dan individuality), serta tingginya dimensi dan interaksi antar fitur yang kompleks (Gheyas and Abdallah 2016). Selain itu, serangan berkelompok oleh beberapa orang dalam sulit terdeteksi. Deteksi serangan cenderung menghasilkan banyak alarm palsu, dan distribusi data yang tidak seimbang menyebabkan model bias terhadap aktivitas normal.

Berbeda dengan *external threat* yang berasal dari pihak luar organisasi seperti peretas atau kelompok kriminal siber yang berusaha menembus perimeter keamanan tanpa izin, *insider attack* dilakukan oleh individu yang memang sudah memiliki hak akses sah ke dalam sistem. Perbedaan mendasar ini menjadikan serangan eksternal umumnya dapat dideteksi melalui mekanisme pertahanan tradisional seperti *firewall*, *intrusion detection system*, atau autentikasi berlapis, karena pola serangannya cenderung menonjol dan berasal dari luar jaringan. Sebaliknya, aktivitas *insider attack* sering kali menyerupai perilaku pengguna biasa sehingga sulit dibedakan dari lalu lintas normal, bahkan oleh sistem keamanan canggih sekalipun. Dengan demikian, meskipun ancaman eksternal lebih banyak disorot karena sifatnya yang terbuka, *insider attack* justru lebih berbahaya karena menyamarkan diri dalam kerangka akses yang sah dan pengetahuan mendalam tentang prosedur internal organisasi, sehingga memerlukan pendekatan deteksi yang lebih khusus dan kontekstual.

2.2 *Dataset* CERT

Penelitian terkait deteksi insider threat umumnya memerlukan data historis yang berisi perilaku pengguna dalam suatu organisasi. Salah satu dataset yang paling banyak digunakan adalah CERT (*Computer Emergency Response Team*) *Insider Threat Dataset* yang dikembangkan oleh *Carnegie Mellon University* (CMU-CERT). Dataset ini tersedia dalam beberapa versi baik versi rilis mayor maupun minor yaitu r1, r2, r3.3, r3.2, r4.1, r4.2, r5.1, r5.2, r6.1, dan r6.2. Dataset CERT memuat catatan aktivitas pengguna seperti *logon*, akses *file*, *email*, penggunaan *web*, hingga interaksi dengan perangkat penyimpanan eksternal. Setiap versi memiliki skenario berbeda yang mensimulasikan perilaku normal maupun aktivitas berbahaya, sehingga dapat digunakan untuk pengembangan dan evaluasi metode deteksi *insider attack* (Glasser and Lindauer 2013).

Dataset CERT merupakan himpunan data sintetik multilog berupa log masuk-keluar, web, *email*, *file*, perangkat *USB*, dan LDAP yang meniru aktivitas organisasi besar serta menyisipkan skenario serangan orang dalam. Setiap rilis dibedakan per versi dan dilengkapi berkas answer key berisi uraian skenario serta identitas pengguna yang terlibat sehingga menyediakan label kebenaran untuk evaluasi. Rilis yang paling sering digunakan adalah r4.2 dan r6.2. Berikut ini merupakan fitur utama yang dimiliki oleh *dataset* CERT:

1. Logon Data
   Menunjukkan *log* aktivitas *logon* pengguna pada suatu komputer.
2. Device Data
   Menunjukkan *log* aktivitas user menyambungkan atau melepaskan perangkat dari komputer seperti *flash drive.*
3. HTTP (*Hypertext Transfer Protocol*) Data
   Berisi *log* akses *http* oleh user dalam suatu komputer.
4. *E-mail* Data
   Berisi *log e-mail* yang dikirim atau diterima oleh tiap pengguna.
5. File Data
   Memuat *log* penyalinan data dari suatu komputer ke perangkat lain.
6. LDAP (*Lightweight Directory Access Protocol*)
   Detail data karyawan yang ada dalam organisasi.

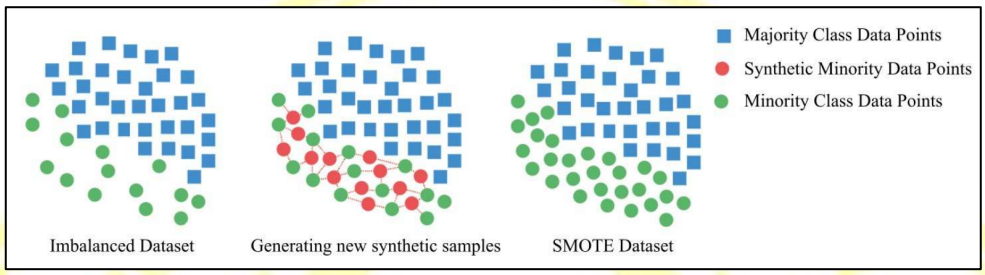
Fitur-fitur ini menggambarkan aktivitas dan perilaku pengguna dalam suatu organisasi yang memuat kondisi normal dan kondisi serangan. Data ini sangat bermanfaat dalam membangun model deteksi *insider attack* untuk mengklasifikasikan aktivitas normal dan serangan.

Dataset sintetik CERT memberikan kontrol penuh, dapat dibagikan ke publik, serta memiliki label yang jelas, namun realisme data terbatas pada dimensi yang dimodelkan dan berisiko mengandung skenario bias yang tidak selalu menggeneralisasi ke skenario dunia nyata. *Dataset* ini sangat ideal untuk digunakan dalam penelitian dan pengujian model, namun kurang disarankan untuk menyimpulkan kinerja model di lingkungan operasional yang sebenarnya.

2.3 *Synthetic Minority Over-sampling Technique*

Synthetic Minority Oversampling Technique (SMOTE) adalah salah satu teknik *oversampling* yang banyak digunakan untuk mengatasi masalah ketidakseimbangan kelas pada *dataset*. Kondisi ini terjadi apabila jumlah data pada kelas mayoritas jauh lebih banyak dibandingkan kelas minoritas.

SMOTE dikembangkan sebagai solusi atas keterbatasan metode *oversampling* tradisional yang bekerja dengan melakukan duplikasi sampel minoritas (Chawla et al. 2002). Alih-alih hanya menggandakan data minoritas, SMOTE menghasilkan sampel sintetis baru dengan cara melakukan interpolasi antar titik dari data minoritas yang ada.



Gambar 2.1 Representasi Visual proses SMOTE (Soundrapandiyan et al. 2023)

Salah satu data pada kelas minoritas diambil secara acak dan ditetapkan sebagai sampel sebagai titik acuan, dapat ditulis sebagai x_i. Titik ini digunakan sebagai dasar pembuatan sampel sintetis baru. Setelah ada satu titik dari kelas minoritas yang ditetapkan, algoritma mencari beberapa tetangga terdekat dari titik tersebut dengan metode *k-nearest neighbors* (k-NN). Nilai k dapat ditentukan terlebih dahulu, apabila dimisalkan k = 5, maka akan dicari lima tetangga terdekat dalam kelas yang sama dari titik yang sudah ditetapkan sebelumnya. Tetangga terdekat yang dipilih ini dapat didefinisikan sebagai x_{i_k} sehingga selanjutnya dapat dibuat titik sintetis baru yang terletak pada sepanjang garis yang menghubungkan titik x_i dan titik x_{i_k} (Syahwaluddin and Alita 2024). Proses ini dapat ditulis dengan rumus berikut:

$$x' = x_i + \delta \cdot (x_{i_k} - x_{i_k}) \quad \text{.....(2.1)}$$

Dalam persamaan tersebut, x' merupakan sampel sintetis baru yang dibuat berdasarkan sampel yang sudah ditetapkan sebelumnya. Variabel δ merupakan bilangan antara 0 dan 1 yang berfungsi untuk memastikan pembuatan sampel sintetis baru berada pada garis yang menghubungkan titik x_i dan titik x_{i_k}.

Proses ini dapat diulang beberapa kali sehingga menghasilkan lebih banyak variasi sampel sintetis pada kelas minoritas, sehingga dapat memperbanyak wilayah keputusan untuk kelas minoritas. Dengan demikian, SMOTE dapat membantu model dalam mengenali pola data sehingga dapat meningkatkan akurasi dan sensitivitas model dalam mendeteksi kelas minoritas.

2.4 Extreme Gradient Boosting

Extreme Gradient Boosting (XGBoost) merupakan salah satu model pembelajaran mesin yang dikenal efektif dalam mengatasi variabel yang kompleks serta dianggap sangat baik dalam menangani data dengan ketidakseimbangan kelas (Velarde et al. 2023). XGBoost merupakan implementasi dari algoritma *gradient boosting* yang dioptimasi dalam hal kinerja serta kecepatan (Bentéjac, Csörgő, and Martínez-Muñoz 2019). XGBoost digunakan dalam klasifikasi dan regresi dengan menggabungkan teknik *boosting* dan regularisasi dalam menciptakan model yang dapat mengakomodasi *overfitting* dengan baik (Chen and Guestrin 2016). Algoritma XGBoost memiliki keunggulan dalam kecepatan pelatihan bahkan dalam kelas data yang tidak seimbang, seperti pada data di dunia nyata (Nugraha 2021).

Proses pelatihan model XGBoost dimulai dengan penentuan prediksi untuk setiap observasi. Predikisi ini dihitung untuk setiap kelas dalam *dataset*, dapat dirumuskan dengan persamaan berikut ini

$$F_0 = \ln(\frac{N_1}{N_0}) \quad \text{.....(2.2)}$$

Dimana F_0 merupakan prediksi mula mula untuk semua sampel, N_1 merupakan jumlah sampel pada kelas positif, dan N_1 merupakan jumlah sampel dalam pada kelas negatif. Rumus ini menghitung nilai *log-odds*, yaitu logaritma dari rasio antara jumlah sampel kelas positif terhadap jumlah sampel kelas negatif dalam dataset. Nilai *log-odds* tersebut berfungsi sebagai dasar estimasi awal yang merepresentasikan distribusi kelas pada data.

Selanjutnya, probabilitas awal setiap sampel terhadap kelas positif P_i dihitung dengan menerapkan fungsi logistik (sigmoid) terhadap nilai prediksi awal, yang dijabarkan pada rumus 2.3.

$$P_i = \frac{1}{1 + e^{-F_0}} \quad \text{.....(2.3)}$$

Pada persamaan 2.3, fungsi sigmoid digunakan untuk mentransformasi nilai *log-odds* F_0 menjadi probabilitas kelas positif P_i dengan rentang nilai antara 0 hingga 1. Berbeda dengan fungsi *softmax* yang digunakan pada kasus multikelas, fungsi sigmoid hanya melibatkan satu output karena model klasifikasi biner memiliki dua kemungkinan kelas (positif dan negatif). Fungsi ini memastikan bahwa probabilitas kedua kelas saling melengkapi, yaitu P_i untuk kelas positif dan $1-P_i$ untuk kelas negatif.

Pada persamaan 2.4, *gradien* didefinisikan sebagai selisih antara probabilitas yang diprediksi dan label aktual.

$$g_{ik} = p_{ik} - y_{ik} \quad \text{.....(2.4)}$$

Rumus 2.4 mendefinisikan p_{ik} sebagai hasil prediksi probabilitas untuk kelas k, dan y_{ik} merupakan label kebenaran yang bernilai 0 atau 1 pada kelas k sampel i. *Hessian* yang digunakan untuk memperbarui parameter dihitung berdasarkan probabilitas kelas, sebagaimana ditunjukkan pada persamaan 2.5 berikut

$$h_{ik} = 2p_{ik}(1 - p_{ik}) \quad \text{.....(2.5)}$$

Persamaan 2.5 mendefinisikan *Hessian* untuk sampel i pada kelas k, dengan p_{ik} merepresentasikan probabilitas prediksi untuk kelas tersebut.

Tahap berikutnya dalam pelatihan model adalah perhitungan *split* pada pohon keputusan. Proses ini melibatkan pembagian data ke dalam dua *node* berdasarkan nilai suatu fitur. *Left node* mencakup sampel yang memenuhi kriteria, sedangkan *right node* mencakup sampel yang tidak memenuhi kriteria tersebut. Setelah pemisahan, total *gradien* dan *Hessian* dihitung secara terpisah untuk masing-masing *node*.

Selanjutnya, dilakukan perhitungan bobot daun pada setiap node. Bobot ini digunakan untuk memperbarui estimasi model serta mengoptimalkan parameter pohon keputusan. Setiap *leaf node* memiliki bobot yang diturunkan dari total *gradien* dan *Hessian* seluruh sampel yang teralokasi pada *node* tersebut. Perhitungan bobot daun mengikuti persamaan 2.6 berikut:

$$w_j = -\frac{G_j}{H_j + \lambda} \quad \text{.....(2.6)}$$

Persamaan 2.6 mendefinisikan bobot daun pada *leaf* j, dengan G_j merepresentasikan total gradien dan H_j total *Hessian* pada leaf tersebut. Total *gradien* dan *Hessian* dihitung dari agregasi *gradien* serta *Hessian* seluruh sampel yang dialokasikan pada *node* bersangkutan. Parameter regularisasi λ, yang umumnya bernilai 1, digunakan untuk mengendalikan kompleksitas model dan mencegah *overfitting*. Bobot daun yang diperoleh berperan dalam memperbarui nilai prediksi, sehingga meminimalkan kesalahan sekaligus meningkatkan akurasi model secara iteratif.

Setelah bobot daun dihitung dan kontribusinya terhadap prediksi ditentukan, XGBoost melakukan pembaruan prediksi model pada setiap iterasi dengan cara menambahkan keluaran pohon keputusan baru ke prediksi kumulatif dari iterasi sebelumnya. Mekanisme ini secara bertahap mengurangi error dan mengoptimalkan performa model. Rumus pembaruan prediksi ditunjukkan sebagai berikut:

$$\hat{y}_{ik}^{(t)} = \hat{y}_{ik}^{(t-1)} + \eta \cdot ft(xi) \quad \text{.....(2.7)}$$

Persamaan 2.7 mendefinisikan prediksi akhir untuk sampel iii pada iterasi ke-t, dengan $\hat{y}_{ik}^{(t-1)}$ sebagai prediksi pada iterasi sebelumnya. Parameter η, atau learning rate, berfungsi mengendalikan besarnya kontribusi pembaruan terhadap prediksi. Fungsi $ft(xi)$ merepresentasikan keluaran pohon keputusan pada iterasi ke-t, yang terdiri atas bobot daun

hasil perhitungan sebelumnya. Proses pembaruan prediksi ini dirancang untuk meminimalkan error dengan menambahkan kontribusi pohon keputusan baru ke prediksi kumulatif, sehingga akurasi klasifikasi dapat ditingkatkan.

Dengan formulasi tersebut, XGBoost melakukan pembangunan model klasifikasi secara iteratif, mengoreksi kesalahan pada setiap tahap, serta menghasilkan struktur pohon yang semakin optimal pada tiap iterasi. Secara keseluruhan, algoritma ini menghasilkan model yang lebih akurat, stabil, dan mampu menangani data berskala besar maupun dengan kompleksitas tinggi secara efisien.

2.5 *Hyperparameter* pada XGBoost

Hyperparameter dalam model pembelajaran mesin merupakan sekumpulan parameter yang ditetapkan secara manual sebelum proses pelatihan dan memiliki pengaruh langsung terhadap performa serta perilaku model. Penentuan hyperparameter yang optimal sangat penting karena dapat memengaruhi kualitas hasil prediksi secara signifikan. Proses pemilihan biasanya dilakukan melalui *hyperparameter tuning* atau *hyperparameter optimization*, yakni pengujian kombinasi *hyperparameter* untuk memperoleh performa terbaik pada *dataset* tertentu. Berikut merupakan *hyperparameter* yang digunakan dalam algoritma XGBoost ditampilkan pada Tabel 2.1.

Tabel 2.1 Parameter XGBoost

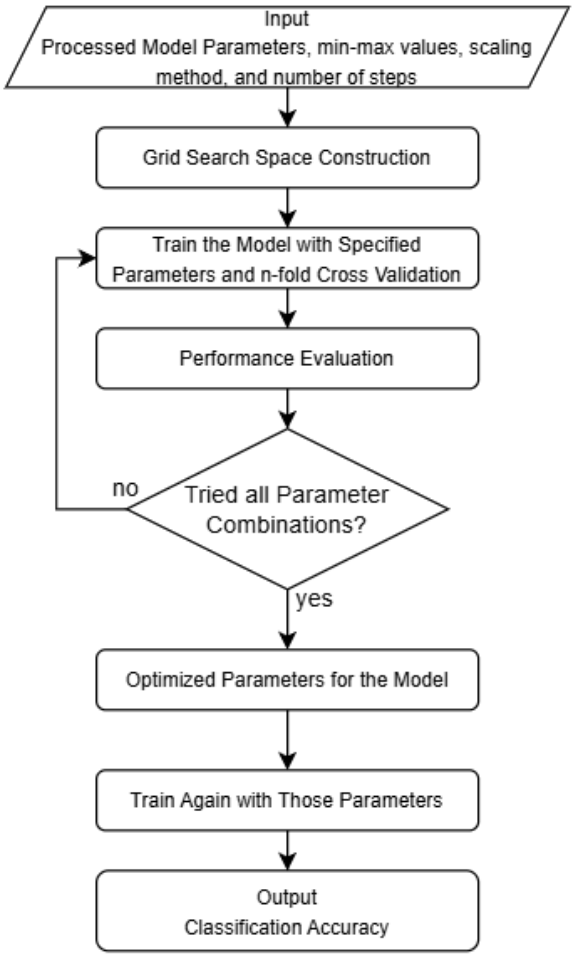
Parameter	Keterangan
learning_rate	Seberapa besar kontribusi setiap pohon terhadap pembaruan model.
max_depth	Kedalaman maksimum setiap pohon keputusan.
n_estimators	Jumlah pohon keputusan yang dibangun dalam model.
subsample	Persentase jumlah sampel data pada setiap iterasi pembentukan pohon.
min_child_weight	jumlah minimum total hessian dari semua instance di sebuah leaf.
gamma	Batas minimum loss reduction agar suatu split pada pohon dapat dilakukan.
colsample_bytree	Persentase fitur yang dipilih secara acak untuk membentuk pohon.

2.6 Hyperparameter Tuning

Pada model machine learning, terdapat sekumpulan parameter yang ditentukan sebelum proses pelatihan dan disebut sebagai *hyperparameter*. Penentuan nilai optimalnya, yang dikenal dengan istilah *hyperparameter tuning*, bertujuan untuk memperoleh performa model terbaik pada dataset tertentu (Chong and Shah, n.d.). Salah satu pendekatan yang sering digunakan adalah *Grid Search*, yaitu metode pencarian menyeluruh terhadap seluruh kombinasi *hyperparameter* yang ditentukan pengguna. Setiap kombinasi diuji dengan melatih model dan dievaluasi menggunakan metrik kinerja, yang umumnya diukur melalui cross-validation.

Tahapan *Grid Search* dimulai dengan pemilihan kandidat nilai *hyperparameter*, lalu model dilatih pada setiap kombinasi dalam ruang produk Kartesian. Evaluasi dilakukan pada data validasi terpisah maupun dengan *cross-validation* pada data pelatihan. Pada akhirnya, metode ini mengidentifikasi kombinasi hyperparameter dengan kinerja tertinggi, yang

kemudian diadopsi sebagai konfigurasi akhir model. Algoritma *Grid Search* ditunjukkan pada Gambar 2.2.



Gambar 2.2 Algoritma *Grid Search* dengan *Cross Validation* (Uddin et al. 2021).

2.7 Evaluasi

Evaluasi model merupakan tahapan penelitian yang penting guna meninjau performa model dalam mengklasifikasi dan memprediksi data dengan akurat. Pemilihan metrik evaluasi yang tepat sangat penting karena metrik harus sesuai dengan tujuan bisnis dan karakteristik data. Menggunakan metrik akurasi semata sering kali menyesatkan, sebab model dapat mencapai akurasi tinggi hanya dengan memprediksi mayoritas kelas (normal activity), namun gagal mendeteksi aktivitas berbahaya yang justru menjadi fokus utama (He and Garcia 2009). Oleh karena itu, berbagai metrik lain digunakan untuk memberikan gambaran yang lebih menyeluruh terhadap performa model.

Precision mengukur jumlah prediksi yang benar-benar positif dibandingkan seluruh hasil prediksi positif yang dihasilkan oleh model. Dalam konteks *insider attack, precision* menunjukkan aktivitas yang benar berbahaya dibandingkan dengan aktivitas yang terdeteksi sebagai berbahaya.

$$Precision = \frac{TP}{TP + FP} \quad \text{.....(2.8)}$$

Recall, sering disebut sebagai *sensitivity* atau *true positive rate* mengukur kemampuan model dalam mendeteksi seluruh kasus positif yang sebenarnya ada. Pada deteksi *insider attack, recall* dinilai lebih penting karena kesalahan melewatkan serangan (*false negative*) dapat berdampak lebih serius dibandingkan kesalahan mendeteksi aktivitas normal (*false positive*).

$$Recall = \frac{TP}{TP + FN} \quad \text{.....(2.9)}$$

Metrik *F1-score* merupakan salah satu metrik evaluasi yang menyeimbangkan nilai precision dan recall dalam bentuk rata-rata harmonis dari keduanya. *F1-score* memberikan evaluasi yang lebih seimbang terutama pada dataset yang tidak seimbang, sehingga dapat menunjukkan keseimbangan antara kemampuan model dalam menghindari *false positive* dan *false negative* (Saito and Rehmsmeier 2015).

$$\text{F1-score} = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad \text{.....(2.12)}$$

Evaluasi kinerja model juga dapat dilakukan dengan menggunakan kurva *Receiver Operating Characteristic* (ROC), yaitu grafik yang memplot nilai *True Positive Rate* (TPR) terhadap *False Positive Rate* (FPR) pada berbagai *threshold* klasifikasi. Area di bawah kurva ROC yang dikenal sebagai *Area Under the Curve* (AUC-ROC) digunakan sebagai indikator keseluruhan performa model. Nilai AUC-ROC mendekati 1 menandakan bahwa model memiliki kemampuan yang baik dalam membedakan antara kelas positif dan kelas negatif (Bradley 1997).

$$\text{AUC-ROC} = \int_0^1 TPR(FPR)d(FPR) \quad \text{.....(2.13)}$$

Dengan demikian, kombinasi metrik seperti *precision, recall*, *F1-score*, kurva ROC, dan skor AUC-ROC dinilai cukup efektif dalam mengevaluasi efektivitas model XGBoost untuk deteksi *insider attack*. Penggunaan metrik yang beragam ini memastikan bahwa model tidak hanya tampak baik secara akurasi, tetapi juga benar-benar mampu mendeteksi ancaman yang jarang terjadi namun berisiko tinggi.

2.8 Penelitian Terdahulu

Studi terdahulu menjadi landasan utama yang menegaskan urgensi dilakukannya penelitian ini. Hasil penelitian terdahulu dapat dijadikan acuan sekaligus pembanding, sehingga penelitian ini diharapkan mampu memberikan kontribusi dalam pengembangan ilmu pengetahuan dan teknologi, khususnya pada bidang deteksi *insider attack* dengan penerapan SMOTE dan algoritma XGBoost pada dataset CERT. Rangkuman penelitian terdahulu ditampilkan pada Tabel 2.2.

Tabel 2.2 Penelitian Terdahulu

No.	Penelitian	Metode	Hasil Penelitian
1	*Insider Threat Detection Using Machine Learning Approach (Bin Sarhan and Altwaijry 2023)*	*Neural Network* SVM AdaBoost *Random Forest*	Penelitian ini menggunakan CERT *Insider Threat Dataset* r4.2 dalam upaya mendeteksi ancaman internal dengan pendekatan *machine learning*. Data diolah melalui tahapan *Deep Feature Synthesis* (DFS), *Principal Component Analysis* (PCA), serta teknik SMOTE sehingga menghasilkan representasi fitur perilaku yang relevan. Dari hasil pengujian, pendekatan klasifikasi, khususnya *Support Vector Machine* (SVM), menunjukkan kinerja terbaik dengan akurasi, *precision, recall*, dan F1-score mencapai 100%. Sebaliknya, metode *anomaly detection* (OCSVM dan iForest) hanya mampu memperoleh akurasi pada kisaran 86–91%. Temuan ini menegaskan bahwa integrasi DFS, PCA, dan SVM sangat efektif dalam mengidentifikasi insider threat, meskipun model tetap memerlukan pelatihan ulang apabila terjadi perubahan struktur log maupun penambahan pengguna baru.
2	*Detecting an Insider Threat and Analysis of XGBoost using Hyperparameter tuning (Mamidanna, Reddy, and Gujju 2022)*	XGBoost *Decision Tree* *Random Forest*	Penelitian ini menggunakan CERT r4.2 *dataset* untuk mendeteksi *insider threat* dengan algoritma XGBoost. Setelah melalui *preprocessing* dan penyeimbangan data dengan SMOTE, model dibandingkan dengan *Decision Tree* dan *Random Forest*. Hasilnya, XGBoost dengan hyperparameter tuning mencapai akurasi 98,6% dan recall 98,9%, lebih tinggi dibanding metode lain. Model ini juga diimplementasikan dalam aplikasi web berbasis Flask untuk prediksi real-time, menunjukkan potensi besar dalam deteksi insider threat yang akurat dan cepat.

Tabel 2.3 Lanjutan Penelitian Terdahulu

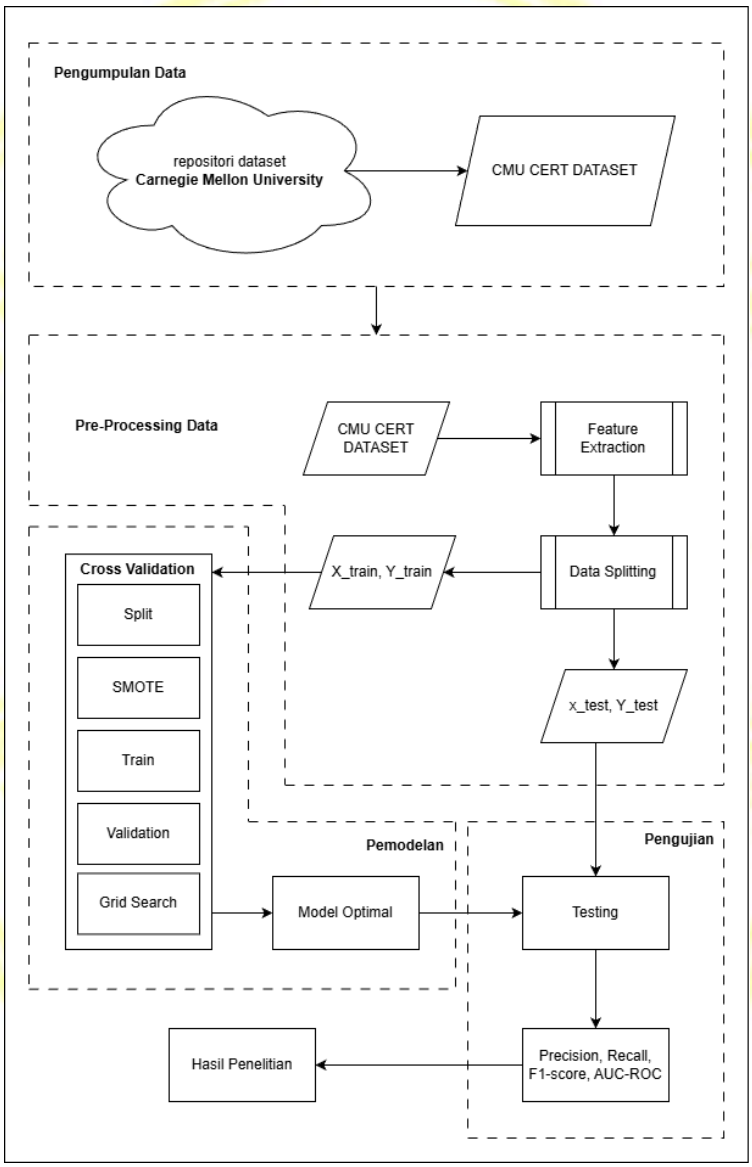
No.	Penelitian	Metode	Hasil Penelitian
3	*Data adjusting strategy and optimized XGBoost algorithm for novel insider threat detection model (Kan et al. 2023)*	ERPSO-XGBoost	Penelitian ini mengusulkan model deteksi insider threat menggunakan strategi penyesuaian data (DA) dan XGBoost yang dioptimalkan dengan ERPSO. Dengan dataset CERT r4.2 yang sangat tidak seimbang, DA digunakan untuk menyaring sampel mencurigakan, menyeimbangkan data dengan undersampling dan SMOTE. Hasilnya, model ERPSO-XGBoost mencapai Precision 0,87, Recall 0,98, F1-score 0,99, dan AUC 0,99, melampaui XGBoost standar dan Random Forest. Pendekatan ini terbukti meningkatkan akurasi, mengurangi missed detection, serta memberi wawasan perilaku karyawan untuk mendukung strategi manajemen keamanan.
4	*Ensemble Strategy for Insider Threat Detection from User Activity Logs (Zou et al. 2020)*	*Data Adjusment* - XGBoost	Penelitian ini mengembangkan strategi *ensemble* untuk deteksi *insider threat* dengan menggabungkan *Data Adjustment* (DA) dan XGBoost menggunakan *dataset* CERT r6.2. Hasilnya, model XGBoost + DA mencapai akurasi 99,51% dan *recall* 98,16%, lebih baik 8,76% dibanding metode lain. Pendekatan ini efektif mengatasi ketidakseimbangan data, meningkatkan akurasi deteksi, serta mengurangi *false positive* sehingga meringankan beban analis keamanan.

Penelitian tersebut memiliki beberapa kesamaan metode serta proses pengolahan data yang dilakukan sebelumnya pada tabel 2.2. Optimasi yang dilakukan menggunakan pendekatan berbeda tetapi seluruhnya berhasil menangani masalah dengan menurunkan tingkat *false positve* atau *false negatif* . Kombinasi SMOTE dan XGBoost diharapkan menghasilkan model yang lebih efektif dalam mendeteksi *insider attack* terutama pada *dataset* CERT *insider threat*.

BAB III
METODOLOGI PENELITIAN

3.1 Metode Penelitian

Penelitian ini dilaksanakan melalui serangkaian tahapan yang dirancang untuk mengembangkan, mengoptimalkan, serta mengevaluasi performa model berbasis XGBoost dalam mendeteksi serangan pada CERT *Insider Threat Dataset* versi r4.2, r5.2, dan r6.2. Rangkaian tahapan tersebut digambarkan dalam diagram alir pada Gambar 3.1.



Gambar 3.1 Metode Penelitian

Alur penelitian dibagi ke dalam empat tahap utama, yaitu pengumpulan data, *pre-processing* data, pelatihan model menggunakan XGBoost, serta perancangan dan pelaksanaan pengujian hingga diperoleh hasil penelitian.

3.2 Pengumpulan Data

Dalam penelitian ini digunakan dataset sekunder, yaitu CERT *Insider Threat Dataset* yang dikembangkan oleh Carnegie Mellon University (CMU-CERT). Dataset ini merupakan dataset sintetis yang disusun untuk merepresentasikan aktivitas dalam suatu organisasi, baik perilaku normal maupun aktivitas serangan dari orang dalam. Data tersebut merupakan hasil simulasi aktivitas organisasi selama 18 bulan, mulai Desember 2009 hingga Mei 2011.

Dalam lingkungan organisasi nyata, data aktivitas pengguna seperti pada dataset penelitian ini umumnya dikumpulkan melalui mekanisme audit logging terpusat yang diintegrasikan dengan sistem *Security Information and Event Management* (SIEM). Setiap komputer pengguna, server, dan perangkat jaringan dikonfigurasi untuk menghasilkan log aktivitas yang mencatat autentikasi, penggunaan perangkat eksternal, akses web, aktivitas email, serta interaksi terhadap berkas. Log autentikasi seperti *logon* dan *logoff* yang dapat diperoleh dari Windows Security Event Log pada domain controller dan workstation, sementara aktivitas perangkat seperti koneksi USB dicatat melalui *system event* yang mencatat identitas perangkat dan waktu koneksi.

Aktivitas *email* pada lingkungan nyata dapat dikumpulkan dari *mail server* organisasi, seperti Microsoft Exchange atau sistem SMTP *gateway*, yang mencatat *metadata* pesan termasuk pengirim, penerima, ukuran pesan, dan keberadaan lampiran. Sementara itu, aktivitas akses *web* dikumpulkan melalui *proxy server* atau *next-generation firewall (NGFW)* seperti Fortinet FortiGate,yang merekam URL yang dikunjungi, waktu akses, serta identitas pengguna berdasarkan mekanisme autentikasi jaringan.

Interaksi terhadap *file*, termasuk penyalinan atau pemindahan *file*, diperoleh dari audit sistem *file* pada *endpoint* atau melalui solusi *Data Loss Prevention* (DLP) yang dirancang untuk memantau pergerakan data sensitif. Seluruh *log* tersebut kemudian dikirimkan secara berkala atau *real-time* ke *server* kolektor log menggunakan agen atau protokol seperti *syslog*, sebelum dinormalisasi dan disimpan dalam repositori terpusat untuk keperluan analisis keamanan.

Dengan pendekatan ini, organisasi dapat membangun rekam jejak aktivitas digital setiap pengguna secara kronologis dan lintas sistem, sehingga memungkinkan analisis perilaku pengguna, korelasi antar peristiwa yang berpotensi mengindikasikan aktivitas *insider threat*. *Log* yang dihasilkan dari berbagai sumber tersebut secara konseptual sejalan dengan skema data pada penelitian ini, yang memisahkan aktivitas ke dalam kategori *logon, device, email, web,* dan *file* untuk memudahkan proses ekstraksi fitur dan analisis perilaku.

Seluruh versi dataset memuat berbagai bentuk aktivitas pengguna, seperti logon, penggunaan perangkat, *email*, akses *file*, akses HTTP, serta LDAP. Rincian lebih lanjut mengenai dataset dan jumlahnya ditampilkan pada Tabel 3.1 hingga Tabel 3.7.

Tabel 3.1 Rincian *device*.csv

No.	Nama Kolom	Keterangan
1	*id*	Identitas unik setiap catatan aktivitas perangkat pada dataset.
2	*date*	Waktu terjadinya aktivitas perangkat pada komputer pengguna.
3	*user*	Identitas pengguna yang melakukan aktivitas pada perangkat tersebut.
4	*pc*	Identitas komputer tempat aktivitas perangkat terjadi.
5	*activity*	Jenis aktivitas perangkat yang terjadi, yaitu *Connect* atau *Disconnect*.

Tabel 3.2 Rincian *email*.csv

No.	Nama Kolom	Keterangan
1	*id*	Identitas unik untuk setiap catatan aktivitas pengiriman email.
2	*date*	Waktu ketika email dikirim oleh pengguna.
3	*user*	Identitas pengguna yang mengirim email dari sistem organisasi.
4	*pc*	Identitas komputer yang digunakan pengguna untuk mengirim email.
5	*to*	Alamat *email* penerima utama dari pesan tersebut.
6	*cc*	Alamat penerima salinan carbon copy yang juga menerima *email*.
7	*bcc*	Alamat penerima *blind carbon copy* yang tidak terlihat oleh penerima lain.
8	*from*	Alamat *email* pengirim yang tercatat pada pesan *email*.
9	*size*	Ukuran email dalam satuan *byte*.
10	*attachments*	Jumlah lampiran yang disertakan dalam *email*.
11	*content*	Isi teks dari *email* yang dikirim oleh pengguna.

Tabel 3.3 Rincian *file*.csv

No.	Nama Kolom	Keterangan
1	*id*	Identitas unik untuk setiap aktivitas penyalinan *file* ke perangkat lain.
2	*date*	Waktu ketika pengguna melakukan aktivitas terhadap *file* tersebut.
3	*user*	Identitas pengguna yang melakukan interaksi terhadap *file*.
4	*pc*	Identitas komputer tempat aktivitas *file* dilakukan.
5	*filename*	Nama *file* yang disalin oleh pengguna.
6	*content*	Representasi isi *file* yang terdiri dari *header file* dalam format heksadesimal yang menunjukkan tipe *file*, diikuti daftar kata kunci konten yang menggambarkan topik isi *file*.

Tabel 3.4 Rincian *decoy_file*.csv

No.	Nama Kolom	Keterangan
1	*decoy_filename*	*Path decoy file* yang ditempatkan pada sistem untuk mendeteksi akses yang mencurigakan oleh pengguna.
2	*pc*	Identitas komputer tempat *file* umpan tersebut berada.

Tabel 3.5 Rincian *http*.csv

No.	Nama Kolom	Keterangan
1	*id*	Identitas unik untuk setiap catatan aktivitas akses web oleh pengguna.
2	*date*	Waktu ketika pengguna mengakses halaman *web* tersebut.
3	*user*	Identitas pengguna yang melakukan akses ke situs *web*.
4	*pc*	Identitas komputer yang digunakan untuk mengakses situs *web*.
5	*url*	Alamat URL lengkap dari halaman *web* yang dikunjungi pengguna.
6	*content*	Kata kunci yang terkait topik atau isi halaman *web* yang diakses.

Tabel 3.6 Rincian *logon*.csv

No.	Nama Kolom	Keterangan
1	*id*	Identitas unik untuk setiap catatan aktivitas autentikasi pada komputer.
2	*date*	Waktu ketika pengguna melakukan aktivitas *logon* atau *logoff* pada sistem.
3	*user*	Identitas pengguna yang melakukan *logon* atau *logoff*pada komputer.
4	*pc*	Identitas komputer tempat pengguna melakukan login atau logout.
5	*activity*	Jenis aktivitas yang terjadi, yaitu *Logon* atau *Logoff*.

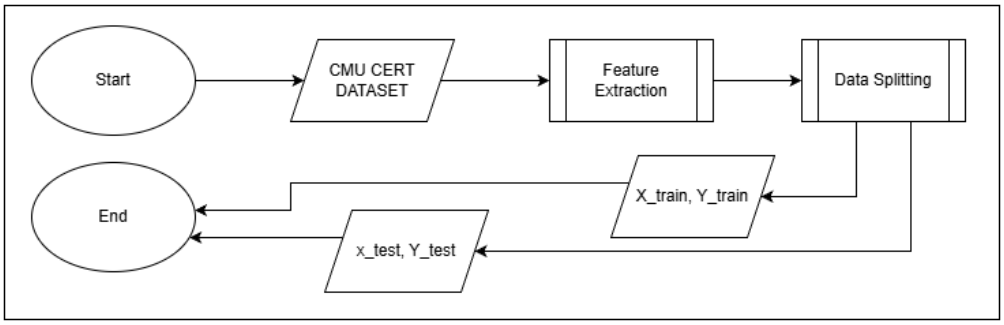
Tabel 3.7 Jumlah Distribusi Serangan Tiap Dataset

No.	Dataset	Jumlah Pengguna Normal	Jumlah Pengguna Abnormal
1	r4.2	930	70
2	r5.2	1901	99
3	r6.2	3995	5

Dataset CERT memberikan label resmi yang dapat digunakan untuk membagi data menjadi 2 kelas yaitu aktivitas pengguna normal dan aktivitas pengguna abnormal. Pada versi r4.2, dataset mencakup 930 pengguna dengan aktivitas normal dan 70 pengguna dengan aktivitas serangan. Sementara itu, pada versi r5.2 terdapat 1901 pengguna normal serta 99 pengguna dengan aktivitas serangan, dan pada versi r6.2 terdiri atas 3995 pengguna normal serta 5 pengguna dengan aktivitas serangan.

3.3 Pre-Processing

Pre-processing data merupakan serangkaian langkah untuk menyiapkan data mentah agar lebih bersih, terstruktur, dan sesuai untuk digunakan oleh algoritma XGBoost. Rangkaian tahapan *pre-processing* data yang digunakan ditampilkan pada diagram alir pada gambar 3.2.



Gambar 3.2 Tahapan *Pre-Processing*

Pada Gambar 3.2, proses *pre-processing* terdiri dari beberapa tahapan, yaitu *feature extraction*, *splitting*, dan *oversampling* menggunakan SMOTE. Penjelasan lebih lanjut mengenai masing-masing tahap diberikan di bawah ini.

3.3.1 *Feature Extraction*

Tahap *Feature Extraction* digunakan untuk mengekstrak fitur-fitur penting yang dapat digunakan dalam *dataset* CMU CERT yang berupa *logon.csv*, *device.csv*, *http.csv*, *email.csv*, *file.csv*, serta *decoy file.csv.* Berikut penjelasan dari masing-masing fitur yang diekstraksi.

a. Ekstraksi Fitur Logon

Dalam penelitian ini, fitur *logon* diturunkan menjadi beberapa kolom yang merepresentasikan pola aktivitas pengguna. Kolom *Weekday_Logon_After* menunjukkan jumlah aktivitas logon pada hari kerja di luar jam kerja normal, yaitu sebelum pukul 07.00 atau setelah pukul 18.00. Kolom *Weekday_Logon_Normal* merepresentasikan jumlah

aktivitas *logon* pada hari kerja yang dilakukan dalam rentang jam kerja normal. Kolom *Weekend_Logon* menggambarkan jumlah aktivitas *logon* yang dilakukan pada akhir pekan. Pemisahan variabel-variabel tersebut dilakukan untuk mengidentifikasi perbedaan perilaku logon antara aktivitas normal dengan aktivitas yang berpotensi menyimpang.

b. Ekstraksi Fitur Device

Kolom *device_insert* merepresentasikan jumlah perangkat eksternal yang disambungkan oleh pengguna ke sistem. Aktivitas ini dapat mencakup penggunaan media penyimpanan portabel, seperti USB *flash drive* atau *hard disk* eksternal, yang berpotensi digunakan untuk memindahkan maupun menyalin data. Pencatatan fitur ini penting karena tingginya frekuensi penyambungan perangkat eksternal dapat mengindikasikan adanya perilaku abnormal atau aktivitas mencurigakan yang berhubungan dengan risiko kebocoran data.

c. Ekstraksi Fitur HTTP

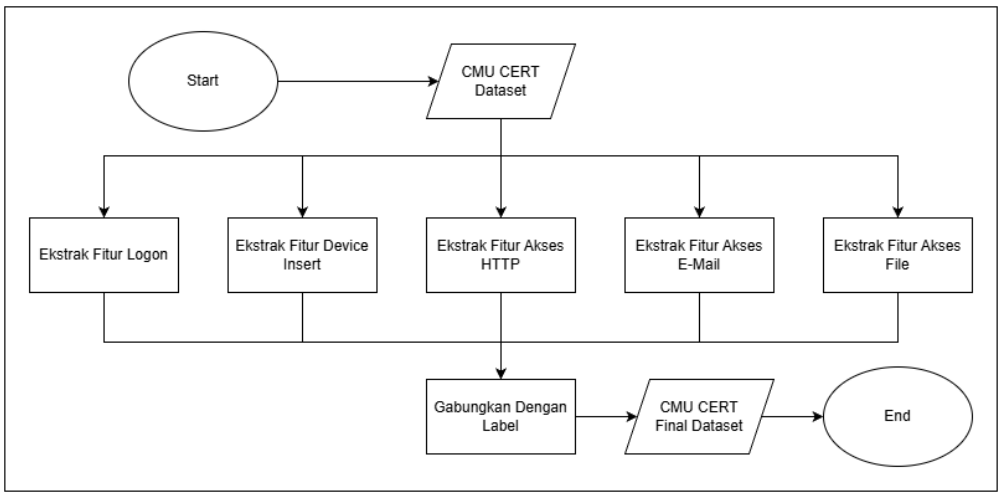
Kolom *http_access_count* menunjukkan total jumlah permintaan HTTP yang dilakukan oleh pengguna, yang mencerminkan tingkat aktivitas akses internet secara keseluruhan. Sementara itu, kolom *unique_domains* merepresentasikan jumlah domain unik yang diakses, sehingga dapat memberikan gambaran mengenai keragaman sumber informasi atau situs yang dikunjungi pengguna. Adapun kolom *avg_req_per_day* menggambarkan rata-rata jumlah permintaan HTTP yang dilakukan per hari, sehingga dapat membantu mengidentifikasi pola penggunaan internet harian. Kombinasi dari ketiga fitur ini dapat digunakan untuk membedakan aktivitas normal dengan potensi adanya aktivitas mencurigakan yang mungkin terkait dengan upaya serangan atau penyalahgunaan akses.

d. Ekstraksi Fitur *Email*

Kolom *email_sent_count* merepresentasikan jumlah *email* yang dikirim oleh pengguna, yang dapat mencerminkan intensitas komunikasi melalui saluran email dalam suatu periode waktu tertentu. Sementara itu, kolom *unique_receivers* menunjukkan jumlah alamat penerima unik dari *email* yang dikirimkan. Kedua fitur ini penting untuk dianalisis karena frekuensi pengiriman *email* yang tinggi atau distribusi pesan ke banyak penerima unik dapat menjadi indikator adanya aktivitas serangan, seperti penyebaran informasi sensitif atau upaya eksfiltrasi data melalui media email.

e. Ekstraksi Fitur File dan Decoy File

Kolom *file_copy_count* menggambarkan jumlah aktivitas penyalinan *file* yang dilakukan oleh pengguna, yang dapat menjadi indikator potensi upaya pemindahan atau replikasi data dalam jumlah besar. Selanjutnya, kolom *file_after_hours* merepresentasikan jumlah aktivitas terkait *file* yang terjadi di luar jam kerja normal, di mana aktivitas semacam ini dapat dianggap menyimpang dari penggunaan wajar. Adapun kolom *decoy_file_access* mencatat jumlah akses terhadap *decoy file*, yaitu file jebakan yang sengaja disediakan untuk mendeteksi adanya perilaku berbahaya. Ketiga fitur ini memberikan informasi penting mengenai aktivitas *file* yang berhubungan erat dengan potensi ancaman insider threat.

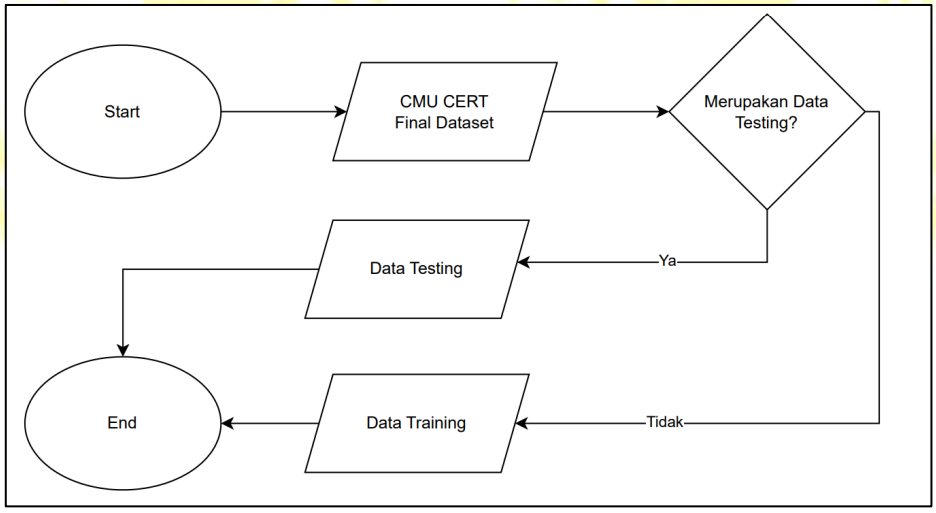


Gambar 3.3 Tahapan Feature Extraction

Seluruh fitur yang telah diekstraksi digabungkan berdasarkan *user_id* sehingga membentuk profil aktivitas masing-masing pengguna. Selanjutnya, penentuan label malicious dilakukan dengan merujuk pada *answer file* yang memuat daftar pengguna yang terindikasi terlibat dalam insiden. Dengan demikian, setiap pengguna dalam dataset memiliki representasi aktivitas yang lengkap beserta label yang diperlukan untuk proses pelatihan dan evaluasi model.

3.3.2 *Data Splitting*

Proses splitting data dilakukan sesuai dengan tahapan yang digambarkan pada Gambar 3.5.



Gambar 3.4 Proses *Splitting* Data

Pada penelitian ini, proses pembagian data dilakukan pada tahap *pre-processing*, bukan pada tahap akhir sebagaimana umumnya. Hal ini dimaksudkan agar data uji tetap merepresentasikan kondisi data asli tanpa melalui pemrosesan lanjutan. Penelitian ini menggunakan *splitting* data dengan proporsi 80% untuk data training dan 20% untuk data testing.

3.4 Pemodelan

Proses membangun dan melatih model menggunakan algoritma XGBoost dilakukan melalui beberapa langkah terstruktur yang dijelaskan sebagai berikut:

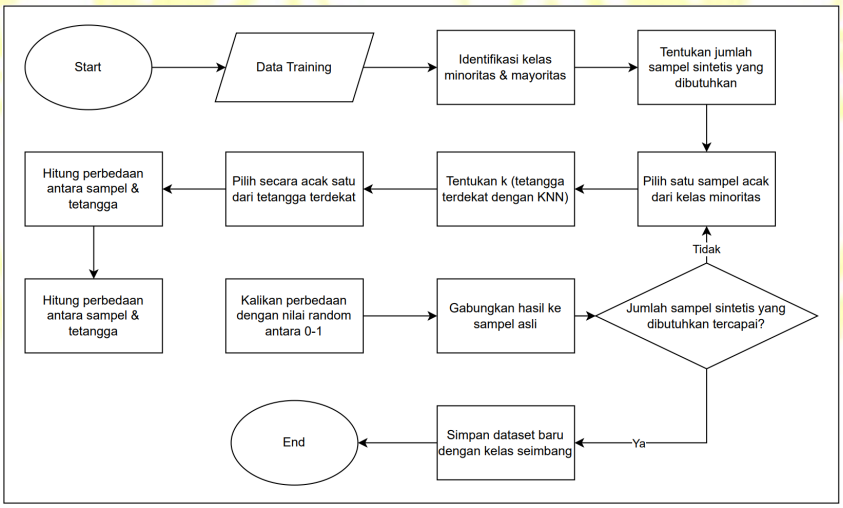
3.4.1 *Splitting Data Cross-Validation*

Pada proses ini, data pada *subset training* dibagi secara merata menjadi 3 *fold*. Masing-masing *fold* akan digunakan untuk melatih dan memvalidasi model melalui proses *cross validation*. Pada setiap iterasi, dua fold digunakan sebagai data latih (*training* set), sedangkan satu fold lainnya digunakan sebagai data validasi (*validation* set). Proses ini dilakukan secara bergantian hingga setiap fold berperan sebagai data validasi sebanyak satu kali. Dengan pendekatan tersebut, model dapat dievaluasi secara menyeluruh terhadap seluruh data tanpa mengubah proporsi kelas yang ada.

Setelah seluruh tiga iterasi selesai, nilai rata-rata dari hasil evaluasi setiap fold dihitung untuk memperoleh estimasi performa model yang lebih stabil serta bebas dari pengaruh pembagian data tertentu. Selain itu, metode ini juga berperan dalam mengurangi risiko overfitting, karena model divalidasi pada bagian data yang berbeda pada setiap iterasi.

3.4.2 *Oversampling Data*

Pada penelitian ini, proses oversampling dilakukan dengan menggunakan metode *Synthetic Minority Over-sampling Technique* (SMOTE). Metode ini melalui beberapa tahapan untuk menghasilkan data sintetis yang bertujuan mengatasi permasalahan ketidakseimbangan kelas pada dataset. Penjelasan lebih rinci mengenai alur proses tersebut ditunjukkan pada Gambar 3.5.



Gambar 3.5 Proses Oversampling Menggunakan SMOTE

Berdasarkan Gambar 3.5, tahapan awal dalam alur SMOTE adalah mengidentifikasi dataset dengan kelas minoritas serta menentukan jumlah sampel baru yang diperlukan untuk mengatasi ketidakseimbangan data. Selanjutnya, dilakukan pemilihan sampel acak dari kelas minoritas. Untuk meningkatkan variasi, urutan sampel dari kelas minoritas kemudian diacak. Pada tahap berikutnya, setiap sampel minoritas dihitung tetangga terdekatnya, yang akan

menjadi acuan dalam pembentukan data baru. Dari tetangga yang terpilih secara acak, SMOTE menghasilkan sampel sintetis melalui interpolasi linier antara sampel asli dengan tetangganya. Proses ini menambahkan data baru ke dalam dataset, dan jumlah oversampling yang tersisa akan terus berkurang hingga target jumlah yang ditentukan tercapai.

3.4.3 XGBoost *Training*

1. Melakukan prediksi awal dengan menggunakan nilai log-odds pada seluruh data.
2. Menghitung probabilitas untuk setiap kelas dan sampel dengan fungsi softmax.
3. Menentukan nilai gradien dan hessian sebagai dasar pembentukan struktur pohon keputusan.
4. Melakukan proses split pada setiap node dengan memanfaatkan nilai total gradien maupun hessian.
5. Menghitung bobot pada daun (leaf weight) untuk menentukan arah percabangan pohon dalam meningkatkan akurasi prediksi.
6. Memperbarui hasil prediksi model dengan mempertimbangkan prediksi sebelumnya, learning rate, serta keluaran dari pohon baru.
7. Mengulangi langkah 1–6 secara iteratif hingga jumlah maksimum pohon keputusan yang ditentukan tercapai.
8. Menghasilkan prediksi akhir model, kemudian menghitung probabilitas tiap kelas untuk menentukan hasil klasifikasi.

Sebagai ilustrasi perhitungan, penelitian ini menggunakan contoh sample dataset yang ditampilkan pada Tabel 3.8, yang terdiri atas 12 fitur, dengan kolom *is malicious* digunakan sebagai label target.

Tabel 3.8 Contoh Sampel *Dataset*

Kolom	**Data 1**	**Data 2**	**Data 3**	**Data 4**
Weekday_Logon_After	0	0	0	0
Weekday_Logon_Normal	328	692	692	703
Weekend_Logon	0	0	0	126
device_insert	346	0	0	642
http_access_count	4878	49478	32870	48095
unique_domains	109	230	357	271
avg_req_per_day	18.33	82.73	55.80	104.55
email_sent_count	480	4711	3012	4239
unique_receivers	299	1529	1251	1926
file_copy_count	357	0	0	589
file_after_hours	0	0	0	0
decoy_file_access	0	0	0	0
is_malicious	1	0	0	1

Prediksi awal (F_0) pada XGBoost ditetapkan dengan menghitung *log odds* dari kelas target.

$$F_0 = ln(\frac{N_1}{N_0}) = \ln\left(\frac{2}{2}\right) = \ln(1) = 0$$

Setelah diperoleh hasil dari prediksi awal, langkah berikutnya adalah menghitung probabilitas awal (P_0) menggunakan persamaan berikut.

$$P_0 = \frac{1}{1 + e^{-F_0}} = \frac{1}{1 + e^{-0}} = \frac{1}{1 + 1} = 0.5$$

Langkah selanjutnya adalah perhitungan Gradien dan Hessian berdasarkan P_0 dan nilai aktual (y_i) seperti pada tabel 3.9.

Tabel 3.9 Perhitungan Gradien dan Hessian

No.	y_i	P_0	Gradien ($g_i = P_0 - y_i$)	Hessian ($h_i = P_0(1 - P_0)$
1	1	0.5	-0.5	0.25
2	0	0.5	0.5	0.25
3	0	0.5	0.5	0.25
4	1	0.5	-0.5	0.25

Langkah selanjutnya adalah membangun pohon keputusan pertama untuk memprediksi gradien dengan kriteria *split*. Dimisalkan fitur yang akan dihitung adalah *device_insert.*

$$Similarity\ Score = \frac{(\sum g_i)^2}{\sum h_i + \lambda} = \frac{(0)^2}{1 + 1} = 0$$

Pemisahan *(split)* akan dilakukan pada nilai 0.0 karena nilai dengan *device_insert* ≤ 0.0 pada kelompok kiri dan *device_insert* > 0.0 pada kelompok kanan.

Tabel 3.10 Hasil Pemisahan

Kelompok	Nomor Sampel	$\sum g_i$	$\sum h_i$
Kiri	2, 3	1	0.5
Kanan	1, 4	-1	0.5

Proses dilanjutkan dengan perhitungan *similarity score* pada daun kiri dan daun kanan.

$$Similarity\ Score\ Kiri = \frac{(\sum g_i)^2}{\sum h_i + \lambda} = \frac{(1)^2}{0.5 + 1} = \frac{1}{1.5} \approx 0.667$$

$$Similarity\ Score\ Kanan = \frac{(\sum g_i)^2}{\sum h_i + \lambda} = \frac{(-1)^2}{0.5 + 1} = \frac{1}{1.5} \approx 0.667$$

Selanjutnya, *gain* atau keuntungan untuk pemisahan dihitung dengan persamaan berikut ini. Dimisalkan nilai gamma (γ) adalah 0.

$$Gain = Similarity\ Score\ Kiri + Similarity\ Score\ Kanan - Similarity\ Score - \gamma$$

$$Gain = 0.667 + 0.667 - 0 - 0 = 1.334$$

Setelah nilai *gain* berhasil didapatkan, skor akhir (w_j) untuk setiap daun akan dihitung, yaitu kontribusi pohon ini pada prediksi log-odds.

$$w_L = \frac{-1}{0.5+1} = \frac{-1}{1.5} \approx -0.667$$

$$w_R = \frac{-(-1)}{0.5+1} = \frac{1}{1.5} \approx 0.667$$

Prediksi akhir dari XGBoost adalah jumlah dari prediksi awal (F_0) dan prediksi dari semua pohon yang ditambahkan, diskalakan oleh Learning Rate (η). Learning rate diasumsikan bernilai 0.3.

$$F_1 = 0 + 0.3 \times 0.667 \approx 0.2 \qquad P_1 = \frac{1}{1+e^{-0.2}} \approx 0.55$$

$$F_2 = 0 + 0.3 \times (-0.667) \approx -0.2 \qquad P_2 = \frac{1}{1+e^{-(-0.2)}} \approx 0.45$$

$$F_3 = 0 + 0.3 \times (-0.667) \approx -0.2 \qquad P_3 = \frac{1}{1+e^{-(-0.2)}} \approx 0.45$$

$$F_4 = 0 + 0.3 \times 0.667 \approx 0.2 \qquad P_4 = \frac{1}{1+e^{-0.2}} \approx 0.55$$

Dengan *threshold* standar yang digunakan untuk klasifikasi bernilai 0.5, maka hasil prediksi di atas menjadi seperti pada tabel 3.11.

Tabel 3.11 Hasil Klasifikasi

No.	label	P_i	Hasil Prediksi
1	1	0.55	1
2	0	0.45	0
3	0	0.45	0
4	1	0.55	1

Hasil prediksi pada tabel 3.11 menunjukkan bahwa model telah belajar dengan sempurna untuk dataset ini hanya dengan 1 iterasi. Apabila model belum sempurna, dibutuhkan iterasi berikutnya dengan menghitung gradien dan hessian baru berdasarkan hasil prediksi sebelumnya.

3.4.4 *Cross-Validation*

Pada tahap *cross-validation*, setiap *fold* yang dihasilkan melalui proses spliting digunakan sebagai data *validasi* secara bergantian, sementara *fold* lainnya berfungsi sebagai data pelatihan (*training set*). Tahap ini bertujuan untuk menilai sejauh mana model mampu melakukan generalisasi terhadap data yang belum pernah ditemui selama proses pelatihan.

Setelah model dilatih menggunakan dua *fold* sebagai data pelatihan, pengujian dilakukan pada satu *fold* yang ditetapkan sebagai data validasi. Hasil prediksi dari pengujian

tersebut kemudian dibandingkan dengan label sebenarnya untuk menghitung metrik evaluasi seperti presisi, *recall*, dan *F1-score*.

Proses ini diulangi hingga setiap *fold* berperan sebagai data validasi satu kali. Nilai performa dari masing-masing *fold* selanjutnya dirata-ratakan untuk memperoleh hasil evaluasi akhir model. Pendekatan ini menghasilkan proses validasi yang lebih andal karena mencerminkan performa model pada berbagai subset data serta meminimalkan potensi bias akibat pembagian data yang tidak seimbang.

3.4.5 *Grid Search*

Tahap *Grid Search* dilaksanakan setelah proses *cross validation* dengan tujuan menemukan kombinasi *parameter* terbaik yang mampu menghasilkan performa model paling optimal. Metode ini bekerja dengan menelusuri seluruh kemungkinan kombinasi nilai dari *parameter* yang telah ditentukan sebelumnya pada *parameter grid*.

Pada setiap kombinasi *parameter*, model dilatih dan divalidasi menggunakan prosedur *cross validation* yang sama seperti pada tahap sebelumnya. Hasil evaluasi dari setiap kombinasi dibandingkan berdasarkan metrik kinerja tertentu, seperti akurasi atau F1-score, sesuai dengan fokus evaluasi penelitian.

Kombinasi parameter yang menghasilkan nilai rata-rata validasi tertinggi dipilih sebagai konfigurasi akhir model. Dengan demikian, proses *Grid Search* berperan penting dalam mengoptimalkan model secara sistematis dan objektif, serta memastikan bahwa model beroperasi dengan *parameter* yang paling sesuai terhadap karakteristik dataset yang digunakan.

3.5 Pengujian Model

Proses pengujian model dilakukan dengan menguji hasil latih model dengan data baru yang belum pernah dijumpai sebelumnya. Hal ini berguna untuk mengevaluasi perubahan performa model dengan pemrosesan data tambahan dibandingkan dengan model tanpa pemrosesan data lebih lanjut. Metrik evaluasi yang digunakan adalah *precision*, *recall*, *F1-Score*, dan *AUC-ROC score.*

Tabel 3.12 Sampel Data Pengujian

No	Label	Prediksi Probabilitas	Hasi Prediksi (Threshold 0.5)	Hasil
1	1	0.999915	1	TP
2	1	0.995743	1	TP
3	0	0.962171	1	FP
4	1	0.941738	1	TP
5	0	0.010646	0	TN
6	0	0.000441	0	TN
7	0	0.000433	0	TN
8	0	0.000260	0	TN
9	0	0.000165	0	TN

Berikut ini adalah contoh perhitungan *precision* untuk sampel data pada tabel 3.12.

$$Precision = \frac{TP}{TP + FP} = \frac{3}{3+1} = \frac{3}{4} = 0.75$$

Berikut ini adalah contoh perhitungan *recall* untuk sampel data pada tabel 3.12.

$$Recall = \frac{TP}{TP + FN} = \frac{3}{3+0} = \frac{3}{3} = 1$$

Berikut ini adalah contoh perhitungan *F1-Score* untuk sampel data pada tabel 3.12.

$$\text{F1-score} = 2 \times \frac{0.75 \times 1}{0.75+1} = 2 \times \frac{0.75}{1.75} = 2 \times 0.42857 \approx 0.857$$

Berikut ini adalah contoh perhitungan AUC-ROC untuk sampel data pada tabel 3.12.

$$\text{AUC-ROC} = \int_0^1 TPR(FPR)d(FPR) = \sum_{i=1}^{n} \frac{(TPR_i + TPR_{i-1})}{2} \times (FPR_i - FPR_{i-1})$$

$$L_A = \frac{0.667 + 0.667}{2} \times (0.1667 - 0) = 0.667 \times 0.1667 = 0.1111$$

$$L_B = \frac{1+1}{2} \times (0.3333 - 0.1667) = 1 \times 0.1667 = 0.1667$$

$$L_C = \frac{1+1}{2} \times (0.5 - 0.3333) = 1 \times 0.1667 = 0.1667$$

$$L_D = \frac{1+1}{2} \times (0.6667 - 0.5) = 1 \times 0.1667 = 0.1667$$

$$L_E = \frac{1+1}{2} \times (0.8333 - 0.6667) = 1 \times 0.1667 = 0.1667$$

$$L_F = \frac{1+1}{2} \times (1 - 0.8333) = 1 \times 0.1667 = 0.1667$$

$$\text{AUC-ROC} = 0.1111 + 0.1667 + 0.1667 + 0.1667 + 0.1667 + 0.1667 \approx 0.9444$$

Pengujian model dilakukan untuk merepresentasikan performa model secara lebih menyeluruh melalui berbagai skenario evaluasi yang dirancang untuk mengukur konsistensi, akurasi, dan kemampuan generalisasi model. Oleh karena itu, pada penelitian ini disajikan empat tabel skenario pengujian model yang digunakan sebagai dasar evaluasi untuk menganalisis kinerja model pada berbagai kondisi pengujian.

Tabel 3.13 Skenario Pengujian Model Pada *Subset Test Dataset* r4.2

Model	Train Set	TP	TN	FP	FN	Precision	Recall	F1	AUC-ROC
XGBoost	r4.2								
XGBoost + smote	r4.2								
XGBoost + smote + *tuning*	r4.2								
XGBoost	r5.2								
XGBoost + smote	r5.2								
XGBoost + smote + *tuning*	r5.2								
XGBoost	r6.2								
XGBoost + smote	r6.2								
XGBoost + smote + *tuning*	r6.2								

Tabel 3.13 menyajikan rancangan skenario pengujian performa model XGBoost dengan beberapa variasi pendekatan pelatihan. Setiap skenario menggunakan dataset pelatihan dari versi r4.2, r5.2, dan r6.2, dengan tiga konfigurasi utama, yaitu XGBoost tanpa penerapan teknik *oversampling*, XGBoost dengan penerapan SMOTE, dan XGBoost dengan kombinasi SMOTE serta proses *hyperparameter tuning*. Evaluasi dilakukan menggunakan beberapa metrik performa, yaitu *Precision*, *Recall*, *F1-Score*, dan *AUC-ROC* untuk menilai efektivitas setiap konfigurasi model dalam mendeteksi serangan pada versi dataset r4.2.

Tabel 3.14 Skenario Pengujian Model Pada *Subset Test Dataset* r5.2

Model	Train Set	TP	TN	FP	FN	Precision	Recall	F1	AUC-ROC
XGBoost	r4.2								
XGBoost + smote	r4.2								
XGBoost + smote + *tuning*	r4.2								
XGBoost	r5.2								
XGBoost + smote	r5.2								
XGBoost + smote + *tuning*	r5.2								
XGBoost	r6.2								
XGBoost + smote	r6.2								
XGBoost + smote + *tuning*	r6.2								

Tabel 3.14 menampilkan rancangan skenario pengujian performa model XGBoost dengan konfigurasi yang sama seperti pada Tabel 3.6, namun menggunakan data uji yang berbeda, yaitu versi r5.2. Tujuan pengujian ini adalah untuk mengevaluasi kemampuan model dalam melakukan generalisasi terhadap data yang belum pernah digunakan selama proses pelatihan. Hasil dari pengujian ini kemudian dibandingkan dengan skenario lainnya untuk menilai kinerja model terhadap data uji lain.

Tabel 3.15 Skenario Pengujian Model Pada *Subset Test Dataset* r6.2

Model	Train Set	TP	TN	FP	FN	Precision	Recall	F1	AUC-ROC
XGBoost	r4.2								
XGBoost + smote	r4.2								
XGBoost + smote + *tuning*	r4.2								
XGBoost	r5.2								
XGBoost + smote	r5.2								
XGBoost + smote + *tuning*	r5.2								
XGBoost	r6.2								
XGBoost + smote	r6.2								
XGBoost + smote + *tuning*	r6.2								

Tabel 3.15 menyajikan rancangan skenario pengujian performa model XGBoost dengan konfigurasi yang sama seperti pada tabel sebelumnya, tetapi menggunakan data uji berbeda, yaitu versi r6.2. Pengujian ini bertujuan untuk menilai konsistensi serta kemampuan model dalam mengenali pola serangan pada data yang memiliki karakteristik distribusi berbeda pada dataset r6.2.

Tabel 3.16 Skenario Pengujian Model Pada *Subset Test Dataset* gabungan

Model	Train Set	TP	TN	FP	FN	Precision	Recall	F1	AUC-ROC
XGBoost	r4.2								
XGBoost + smote	r4.2								
XGBoost + smote + *tuning*	r4.2								
XGBoost	r5.2								
XGBoost + smote	r5.2								
XGBoost + smote + *tuning*	r5.2								
XGBoost	r6.2								
XGBoost + smote	r6.2								
XGBoost + smote + *tuning*	r6.2								

Tabel 3.15 menampilkan rancangan skenario pengujian performa model XGBoost dengan konfigurasi yang sama seperti pada tabel-tabel sebelumnya, namun menggunakan data uji gabungan dari seluruh dataset uji yang telah digunakan pada skenario sebelumnya. Tujuan dari pengujian ini adalah untuk mengevaluasi performa model ketika dihadapkan pada data uji yang lebih beragam. Pengujian ini diharapkan dapat memberikan gambaran yang lebih menyeluruh mengenai kemampuan generalisasi model dalam mendeteksi serangan pada berbagai variasi data uji.

BAB IV
HASIL PENGUJIAN DAN PEMBAHASAN

4.1 Hasil Penelitian

Bagian ini menguraikan proses implementasi, hasil yang diperoleh, serta pembahasan berdasarkan perencanaan yang telah ditetapkan pada metodologi penelitian, meliputi tahap pengumpulan data, *preprocessing*, pelatihan model, pengujian, dan evaluasi. Pengujian dilakukan dengan menganalisis hasil evaluasi menggunakan beberapa parameter kinerja yang bertujuan untuk menilai kemampuan model dalam menurunkan tingkat *false positive* dan *false negative*.

4.2 Implementasi Pengumpulan Data

Data pada penelitian ini diperoleh dari *repository dataset Carnegie Mellon University* (CMU-CERT). Dataset yang dikumpulkan dibatasi pada data berformat CSV, karena format tersebut sesuai untuk tahap pemrosesan lanjutan dalam pemodelan machine learning. Gambaran hasil dari proses pengumpulan dataset disajikan pada Tabel 4.1.

Tabel 4.1 *File csv CERT Insider Threat Test Dataset r4.2*

Dataset	**Jumlah Baris**
decoy_file.csv	0
device.csv	405380
email.csv	2629979
file.csv	445581
http.csv	28434423
logon.csv	854859
psychometric.csv	1000
LDAP/2009-12.csv	1000
LDAP/2010-01.csv	1000
LDAP/2010-02.csv	989
LDAP/2010-03.csv	982
LDAP/2010-04.csv	980
LDAP/2010-05.csv	974
LDAP/2010-06.csv	968
LDAP/2010-07.csv	961
LDAP/2010-08.csv	947
LDAP/2010-09.csv	934
LDAP/2010-10.csv	918
LDAP/2010-11.csv	906
LDAP/2010-12.csv	890
LDAP/2011-01.csv	878
LDAP/2011-02.csv	868
LDAP/2011-03.csv	857
LDAP/2011-04.csv	846
LDAP/2011-05.csv	845

Tabel 4.1 menyajikan jumlah baris data pada setiap *file csv* dalam *CERT Insider Threat Dataset* r4.2 yang dikumpulkan selama periode simulasi 18 bulan. Setiap *file* merepresentasikan berbagai jenis aktivitas pengguna dalam suatu organisasi, dengan volume data yang bervariasi. Informasi ini memberikan gambaran awal mengenai skala dan karakteristik dataset yang digunakan, sebelum seluruh berkas digabungkan sebagai dasar pembentukan data analisis untuk proses pemodelan deteksi serangan orang dalam.

Tabel 4.2 *File csv CERT Insider Threat Test Dataset r5.2*

Dataset	**Jumlah Baris**
decoy_file.csv	11821
device.csv	836984
email.csv	17361575
file.csv	887621
http.csv	58960449
logon.csv	1810070
psychometric.csv	2000
LDAP/2009-12.csv	2000
LDAP/2010-01.csv	2000
LDAP/2010-02.csv	1987
LDAP/2010-03.csv	1975
LDAP/2010-04.csv	1964
LDAP/2010-05.csv	1958
LDAP/2010-06.csv	1940
LDAP/2010-07.csv	1928
LDAP/2010-08.csv	1911
LDAP/2010-09.csv	1889
LDAP/2010-10.csv	1876
LDAP/2010-11.csv	1858
LDAP/2010-12.csv	1837
LDAP/2011-01.csv	1824
LDAP/2011-02.csv	1807
LDAP/2011-03.csv	1793
LDAP/2011-04.csv	1778
LDAP/2011-05.csv	1762

Pada Tabel 4.2 ditampilkan distribusi jumlah baris data untuk setiap *file* pada *CERT Insider Threat Dataset* r5.2. Dibandingkan dengan versi sebelumnya, dataset ini memiliki jumlah pengguna dan aktivitas yang lebih banyak, sehingga volume data yang dihasilkan cenderung meningkat. Informasi ini memberikan gambaran awal mengenai kompleksitas data yang digunakan, sekaligus menjadi dasar dalam perancangan proses pengolahan dan pemodelan selanjutnya pada penelitian ini.

Tabel 4.3 *File csv CERT Insider Threat Test Dataset r6.2*

Dataset	Jumlah Baris
decoy_file.csv	31095
device.csv	1551828
email.csv	10994957
file.csv	2014883
http.csv	117025216
logon.csv	3530285
psychometric.csv	4000
LDAP/2009-12.csv	4000
LDAP/2010-01.csv	4000
LDAP/2010-02.csv	3977
LDAP/2010-03.csv	3952
LDAP/2010-04.csv	3930
LDAP/2010-05.csv	3911
LDAP/2010-06.csv	3894
LDAP/2010-07.csv	3870
LDAP/2010-08.csv	3853
LDAP/2010-09.csv	3833
LDAP/2010-10.csv	3803
LDAP/2010-11.csv	3773
LDAP/2010-12.csv	3744
LDAP/2011-01.csv	3719
LDAP/2011-02.csv	3697
LDAP/2011-03.csv	3674
LDAP/2011-04.csv	3654
LDAP/2011-05.csv	3639

Tabel 4.3 menggambarkan jumlah baris data pada setiap *file* dalam *CERT Insider Threat Test Dataset* r6.2. Versi *dataset* ini merepresentasikan skenario organisasi dengan skala yang lebih besar serta karakteristik aktivitas pengguna yang lebih beragam. Penyajian jumlah baris data pada setiap berkas bertujuan untuk menunjukkan besarnya data yang dianalisis dan menjadi pertimbangan dalam proses pengolahan data, khususnya terkait efisiensi pemrosesan dan kebutuhan sumber daya komputasi pada tahap pemodelan.

4.3 Implementasi *Pre-Processing*

Pemrosesan data pada penelitian ini dilakukan melalui beberapa tahapan, meliputi *feature extraction,* dan *data splitting*. Penjelasan lebih rinci mengenai masing-masing tahapan tersebut diuraikan sebagai berikut.

4.3.1 Implementasi *Feature Extraction*

Feature extraction dilakukan dalam beberapa tahap yaitu proses fitur logon, proses fitur device, proses fitur http, proses fitur email, dan proses fitur file, serta penggabungan label.

Modul Program 4.1 : Proses Logon

```
df['date'] = pd.to_datetime(df['date'])

df['hour'] = df['date'].dt.hour
df['weekday'] = df['date'].dt.dayofweek

df['is_weekend'] = df['weekday'] >= 5
df['is_after_hours'] = (df['hour'] < 7) | (df['hour'] > 18)
df['is_normal_hours'] = ~df['is_after_hours']

df['weekday_after'] = (~df['is_weekend']) & (df['is_after_hours'])
df['weekday_normal'] = (~df['is_weekend']) & (df['is_normal_hours'])

result = df.groupby('user_id').agg(
    Weekday_Logon_After=('weekday_after','sum'),
    Weekday_Logon_Normal=('weekday_normal','sum'),
    Weekend_Logon=('is_weekend','sum')
)
```

Modul Program 4.1 menunjukkan proses ekstraksi fitur logon, algoritma ini bertujuan untuk mengidentifikasi dan menghitung pola aktivitas logon pengguna berdasarkan waktu dan hari terjadinya logon. Proses diawali dengan inisialisasi tiga variabel penghitung, yaitu jumlah aktivitas pada hari kerja di luar jam kerja, jumlah aktivitas pada hari kerja di dalam jam kerja, dan jumlah aktivitas pada akhir pekan.

Selanjutnya, algoritma melakukan iterasi terhadap setiap kejadian yang dimiliki oleh seorang pengguna. Untuk setiap kejadian, sistem terlebih dahulu mengevaluasi apakah aktivitas tersebut terjadi pada hari kerja atau akhir pekan. Apabila aktivitas terjadi pada hari kerja, maka algoritma akan melakukan pemeriksaan lanjutan untuk menentukan apakah aktivitas tersebut dilakukan di luar jam kerja atau pada jam kerja normal. Aktivitas pada hari kerja di luar jam kerja akan menambah nilai penghitung aktivitas setelah jam kerja, sedangkan aktivitas pada jam kerja normal akan menambah nilai penghitung login normal.

Sebaliknya, apabila aktivitas dilakukan pada akhir pekan, maka kejadian tersebut akan langsung dikategorikan sebagai aktivitas akhir pekan dan nilai penghitung yang bersesuaian akan ditambahkan. Setelah seluruh kejadian login diproses, algoritma menghasilkan tiga nilai fitur yang merepresentasikan pola perilaku login pengguna, yaitu frekuensi aktivitas di luar jam kerja pada hari kerja, frekuensi aktivitas pada jam kerja normal, dan frekuensi aktivitas pada akhir pekan.

Fitur-fitur ini digunakan untuk menangkap indikasi perilaku tidak wajar, seperti aktivitas di luar jam kerja atau pada akhir pekan, yang dalam konteks deteksi *insider threat* dapat menjadi salah satu sinyal awal adanya aktivitas mencurigakan.

Modul Program 4.2 : Proses Fitur *Device*

```
df = df[df['activity'] == 'Connect']
result = df.groupby('user_id').size()
```

Pada Modul Program 4.2 ditunjukkan proses ekstraksi fitur perangkat (device), algoritma ini bertujuan untuk mengukur intensitas aktivitas pengguna dalam menghubungkan perangkat eksternal ke sistem. Proses diawali dengan inisialisasi sebuah variabel penghitung yang digunakan untuk menyimpan jumlah koneksi perangkat yang dilakukan oleh seorang pengguna.

Selanjutnya, algoritma melakukan iterasi terhadap seluruh kejadian aktivitas perangkat yang terkait dengan pengguna tersebut. Untuk setiap kejadian, sistem akan memeriksa jenis aksi yang dilakukan. Apabila aksi yang terdeteksi merupakan aktivitas c*onnect*, maka kejadian tersebut dianggap sebagai satu kali koneksi perangkat dan nilai penghitung akan ditambahkan satu.

Proses ini terus dilakukan hingga seluruh aktivitas perangkat pengguna selesai diproses. Setelah itu, algoritma menghasilkan satu nilai fitur yang merepresentasikan jumlah total koneksi perangkat yang dilakukan oleh pengguna selama periode pengamatan.

Fitur ini digunakan untuk menangkap pola penggunaan perangkat eksternal, di mana frekuensi koneksi perangkat yang tinggi dapat mengindikasikan potensi risiko keamanan, khususnya dalam konteks deteksi *insider attack* yang melibatkan pencurian atau pemindahan data melalui media eksternal.

Modul Program 4.3 : Proses Fitur *HTTP*

```
df['date'] = pd.to_datetime(df['date'])
df['date_only'] = df['date'].dt.date

access_count = df.groupby('user_id').size()
unique_urls = df.groupby('user_id')['url'].nunique()
days_active = df.groupby('user_id')['date_only'].nunique()

result = pd.DataFrame({
    'user_id': access_count.index,
    'http_access_count': access_count.values,
    'unique_domains': unique_urls.values,
    'avg_req_per_day': access_count.values / days_active.values
})
```

Modul Program 4.2 menunjukkan proses ekstraksi fitur aktivitas web (HTTP), algoritma ini bertujuan untuk menganalisis pola akses internet pengguna berdasarkan intensitas, keberagaman, dan frekuensi harian aktivitas web. Proses diawali dengan inisialisasi beberapa variabel yang digunakan untuk mencatat jumlah total akses web, daftar domain yang diakses, serta jumlah hari aktif pengguna dalam melakukan akses web.

Selanjutnya, algoritma melakukan iterasi terhadap setiap kejadian akses web yang dilakukan oleh pengguna. Pada setiap kejadian, sistem akan menambahkan satu nilai pada penghitung total akses web. Selain itu, alamat website yang diakses akan dicatat untuk membentuk kumpulan domain unik, sehingga sistem dapat mengidentifikasi seberapa beragam tujuan akses web pengguna. Tanggal kejadian akses web juga dicatat untuk menentukan hari-hari aktif pengguna dalam melakukan aktivitas web.

Setelah seluruh kejadian akses web selesai diproses, algoritma menghitung nilai rata-rata akses web per hari dengan membagi jumlah total akses web dengan jumlah hari aktif pengguna. Apabila tidak terdapat hari aktif, maka nilai rata-rata akses web per hari ditetapkan sebagai nol untuk menghindari kesalahan perhitungan.

Hasil dari proses ini berupa tiga fitur utama, yaitu jumlah total akses web, jumlah domain unik yang diakses, dan rata-rata akses web per hari. Ketiga fitur tersebut digunakan untuk menangkap perilaku *browsing* pengguna, di mana pola akses web yang tidak wajar, seperti jumlah akses yang sangat tinggi atau keberagaman domain yang berlebihan dapat menjadi indikator awal adanya aktivitas mencurigakan dalam konteks *insider threat*.

Modul Program 4.4 : Proses Fitur *E-Mail*

```
df['date'] = pd.to_datetime(df['date'])

sent_count = df.groupby('user_id').size()
unique_receivers = df.groupby('user_id')['to'].nunique()

result = pd.DataFrame({
    'user_id': sent_count.index,
    'email_sent_count': sent_count.values,
    'unique_receivers': unique_receivers.values
})
```

Pada Modul Program 4.4 proses ekstraksi fitur aktivitas *email*, algoritma ini bertujuan untuk mengidentifikasi pola komunikasi pengguna melalui *email* dengan melihat intensitas pengiriman dan keberagaman penerima *email*. Proses diawali dengan inisialisasi dua variabel utama, yaitu penghitung jumlah *email* yang dikirim oleh pengguna dan kumpulan penerima *email* yang unik.

Selanjutnya, algoritma melakukan iterasi terhadap setiap kejadian pengiriman *email* yang dilakukan oleh pengguna. Pada setiap kejadian, sistem akan menambahkan satu nilai pada penghitung jumlah *email* terkirim. Selain itu, alamat *email* penerima akan dicatat untuk membentuk kumpulan penerima yang unik, sehingga sistem dapat mengukur seberapa luas cakupan komunikasi pengguna melalui *email*.

Setelah seluruh kejadian pengiriman *email* selesai diproses, algoritma menghasilkan dua nilai fitur, yaitu jumlah total *email* yang dikirim oleh pengguna dan jumlah penerima *email* yang berbeda. Kedua fitur ini digunakan untuk menangkap pola komunikasi pengguna, di mana peningkatan jumlah *email* atau keberagaman penerima yang tidak wajar dapat mengindikasikan potensi aktivitas *insider*, seperti penyebaran informasi sensitif atau komunikasi intensif dengan pihak tertentu.

Modul Program 4.5 : Proses Fitur *File*

```
df['date'] = pd.to_datetime(df['date'], errors='coerce')

df['after_hours'] = (df['date'].dt.hour < 7) | (df['date'].dt.hour > 18)

df['pc'] = df['pc'].astype(str).str.strip()

copy_count = df.groupby('user_id').size()
after_hours_count = df.groupby('user_id')['after_hours'].sum()

decoy_df = pd.read_csv(decoy_path)

merged = df.merge(
    decoy_df[['decoy_filename','pc']],
    left_on=['filename','pc'],
    right_on=['decoy_filename','pc'],
    how='inner'
)

decoy_access = merged.groupby('user_id').size()

uids = set(copy_count.index) | set(after_hours_count.index) |
set(decoy_access.index)

result = pd.DataFrame({
    'user_id': list(uids),
    'file_copy_count': [copy_count.get(u,0) for u in uids],
    'file_after_hours': [after_hours_count.get(u,0) for u in uids],
    'decoy_file_access': [decoy_access.get(u,0) for u in uids],
})
```

Pada Modul Program 4.5 proses ekstraksi fitur aktivitas *file* dan akses *decoy*, algoritma ini bertujuan untuk menganalisis perilaku pengguna dalam mengakses *file*, khususnya untuk mendeteksi aktivitas yang berpotensi berbahaya. Proses diawali dengan inisialisasi tiga variabel penghitung, yaitu jumlah total aktivitas *file*, jumlah akses *file* yang dilakukan di luar jam kerja, dan jumlah akses terhadap *file decoy*.

Selanjutnya, algoritma melakukan iterasi terhadap setiap kejadian aktivitas *file* yang dilakukan oleh pengguna. Untuk setiap kejadian, sistem akan terlebih dahulu menambahkan satu nilai pada penghitung total aktivitas *file* sebagai representasi intensitas aktivitas pengguna terhadap *file*. Setelah itu, sistem akan mengevaluasi waktu kejadian untuk menentukan apakah aktivitas *file* tersebut dilakukan di luar jam kerja. Apabila aktivitas terjadi di luar jam kerja, maka nilai penghitung akses *file* di luar jam kerja akan ditambahkan.

Selain evaluasi waktu, algoritma juga melakukan pemeriksaan terhadap objek *file* yang diakses. Apabila file yang diakses oleh pengguna teridentifikasi sebagai *file decoy*, yaitu *file* jebakan yang disediakan untuk mendeteksi aktivitas mencurigakan, maka kejadian tersebut akan dicatat sebagai satu kali akses *decoy* dan nilai penghitung akses *file decoy* akan ditambahkan.

Setelah seluruh aktivitas *file* selesai diproses, algoritma menghasilkan tiga nilai fitur utama, yaitu jumlah total aktivitas *file*, jumlah akses *file* di luar jam kerja, dan jumlah akses *file decoy*. Fitur-fitur ini digunakan untuk menangkap indikasi perilaku berisiko, di mana akses *file* di luar jam kerja atau interaksi dengan *file decoy* dapat menjadi indikator kuat adanya upaya penyalahgunaan atau kebocoran data oleh *insider*.

Modul Program 4.6 : Proses Pelabelan

```
insider_df['user'] = insider_df['user'].astype(str).str.strip()

label_df = pd.DataFrame({
    'user_id': insider_df['user'].unique(),
    'is_malicious': 1
})
```

Pada Modul Program 4.6 proses pelabelan pengguna, algoritma ini bertujuan untuk menentukan kelas setiap pengguna berdasarkan informasi data pengguna *insider* yang disediakan dalam dataset CERT *Insider Threat*. Proses pelabelan dilakukan pada tingkat pengguna, sehingga setiap pengguna hanya memiliki satu label yang merepresentasikan status perilaku pengguna tersebut selama periode pengamatan.

Proses diawali dengan pembentukan daftar pengguna yang teridentifikasi sebagai pelaku serangan berdasarkan *file insiders.csv* pada dataset. Daftar ini berisi identitas pengguna yang terlibat dalam skenario serangan *insider*. Selanjutnya, algoritma melakukan evaluasi terhadap setiap pengguna yang terdapat pada dataset hasil ekstraksi fitur.

Untuk setiap pengguna, sistem akan memeriksa apakah identitas pengguna tersebut termasuk dalam daftar pengguna berbahaya. Apabila pengguna terdaftar sebagai pelaku serangan, maka pengguna tersebut akan diberi label 1 yang merepresentasikan kelas *malicious*. Sebaliknya, apabila pengguna tidak terdaftar dalam daftar tersebut, maka pengguna akan diberi label 0 yang merepresentasikan kelas *non-malicious*.

Pendekatan pelabelan pada tingkat pengguna ini dilakukan untuk memastikan konsistensi antara data fitur yang bersifat agregat dan target klasifikasi yang dihasilkan. Dengan demikian, model *machine learning* yang dilatih akan mempelajari pola perilaku pengguna secara keseluruhan, bukan berdasarkan kejadian. Pendekatan ini sesuai dengan tujuan dan batasan penelitian, yaitu deteksi *insider attack* berbasis perilaku pengguna.

Tabel 4.4 Kolom Hasil *Feature Extraction*

Dataset	**Keterangan**
user_id	Identitas pengguna yang menjadi unit analisis pada dataset hasil ekstraksi fitur.
Weekday_Logon_After	Jumlah *logon* yang dilakukan pengguna pada hari kerja di luar jam kerja.
Weekday_Logon_Normal	Jumlah *logon* yang dilakukan pengguna pada hari kerja dalam jam kerja.
Weekend_Logon	Jumlah aktivitas *logon* yang dilakukan pengguna pada akhir pekan.
Device_insert	Jumlah aktivitas pemasangan perangkat eksternal oleh pengguna.
http_access_count	Total jumlah permintaan akses *web* yang dilakukan pengguna.
unique_domains	Jumlah domain *web* unik yang diakses oleh pengguna.
avg_req_per_day	Rata-rata jumlah permintaan akses *web* yang dilakukan pengguna per hari.
email_sent_coun	Jumlah *email* yang dikirim oleh pengguna.
unique_receivers	Jumlah alamat email penerima unik yang menerima *email* dari pengguna.
file_copy_count	Jumlah aktivitas penyalinan *file* ke perangkat eksternal oleh pengguna.
file_after_hours	Jumlah aktivitas penyalinan *file* yang dilakukan di luar jam kerja.
decoy_file_access	Jumlah akses pengguna terhadap *file* umpan yang digunakan untuk mendeteksi aktivitas mencurigakan.
is_malicious	Label kelas untuk menunjukkan pengguna insider attack atau pengguna normal.

Kolom *Weekday_Logon_After* dan *Weekend_Logon* dapat menjadi salah satu indikasi adanya aktivitas kerja di luar jam normal, yang dalam konteks insider threat sering muncul ketika pelaku mencoba melakukan aktivitas tanpa pengawasan, seperti pencurian data di luar jam kerja. Kolom device_insert dapat menjadi indikasi adanya penggunaan perangkat external seperti *flash drive* atau *external hard drive*, yang sering dimanfaatkan untuk menyalin atau memindahkan data sensitif ke luar organisasi. Kolom *file_copy_count* dapat menjadi indikasi adanya aktivitas penyalinan file dalam jumlah besar, yang dapat berkaitan dengan upaya data *exfiltration* sebelum pengguna meninggalkan organisasi. Kolom *file_after_hours* dapat menjadi indikator adanya penyalinan *file* yang dilakukan di luar jam kerja, hal ini dapat memperkuat indikasi aktivitas pencurian data karena dilakukan pada waktu yang tidak biasa.

Kolom *http_access_count*, *unique_domains*, dan *avg_req_per_day* dapat menjadi indikasi adanya perbedaan pola aktivitas akses web, misalnya banyaknya akses ke situs unik dan banyaknya akses situs per hari yang dapat digunakan untuk memindahkan data organisasi.

Kolom *email_sent_count* dan *unique_receivers* dapat menjadi indikator aktivitas pengiriman *email*, yang berpotensi menunjukkan pengiriman informasi sensitif ke pihak eksternal atau pengiriman email massal yang tidak biasa.

Kolom *decoy_file_access* dapat menjadi indikasi adanya upaya pencarian atau akses terhadap file sensitif, karena *decoy file* dirancang untuk memicu *alarm* ketika diakses oleh pengguna yang tidak seharusnya.

Secara keseluruhan, pola yang terbentuk adalah penyimpangan perilaku pengguna dari kebiasaan normal, seperti jumlah *logon* di luar jam kerja, penggunaan perangkat eksternal yang tidak biasa, jumlah penyalinan *file* terutama diluar jam kerja, jumlah akses *web*, serta jumlah aktivitas *email*, yang secara bersama-sama dapat mengindikasikan potensi *insider attack*.

4.3.2 Implementasi *Data Splitting*

Data splitting berguna untuk membagi seluruh dataset menjadi 2 subset yang akan digunakan untuk *training* model dan untuk *testing* model. Dalam penelitian ini digunakan pembagian dengan perbandingan data *training* dan *testing* sebesar 80:20.

Modul Program 4.7 : Proses *Data Splitting*

```
X = df.drop('is_malicious')
y = df['is_malicious']

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=42
)
```

Modul Program 4.7 digunakan untuk membagi dataset ke dalam data latih dan data uji dengan menerapkan pendekatan *stratified sampling*. Pembagian dilakukan dengan terlebih dahulu mengelompokkan data berdasarkan kelas pada label y, sehingga setiap kelas diproses secara terpisah. Untuk setiap kelas, indeks data diacak guna menghindari bias urutan, kemudian sebagian data ditetapkan sebagai data uji berdasarkan rasio pengujian 20:80, sementara sisanya digunakan sebagai data latih. Dengan mekanisme ini, proporsi setiap kelas pada data latih dan data uji tetap konsisten dengan distribusi kelas pada dataset awal. Pendekatan stratifikasi ini penting untuk memastikan bahwa proses pelatihan dan evaluasi model dilakukan secara adil, khususnya pada permasalahan klasifikasi dengan distribusi kelas yang tidak seimbang, sehingga kinerja model yang dihasilkan lebih representatif terhadap kondisi data sebenarnya.

Tabel 4.5 Hasil *Data Splitting*

Versi Dataset	Total	Training	Testing
r4.2	1000	800	200
r5.2	2000	1600	400
r6.2	4000	3200	800

Tabel 4.5 menunjukkan hasil proses *data splitting* pada setiap versi dataset. Dataset r4.2 dengan total 1000 baris dibagi menjadi 800 baris untuk pelatihan dan 200 baris untuk pengujian. Pada dataset r5.2 yang berjumlah 2000 baris, sebanyak 1600 baris digunakan sebagai data latih dan 400 baris sebagai data uji. Sementara itu, dataset r6.2 yang memiliki total 4000 data terdiri dari 3200 baris sebagai data pelatihan dan 800 baris sebagai data uji.

4.4 Implementasi Pemodelan

Pada bagian ini dibahas proses implementasi pemodelan XGBoost yang terdiri dari *splitting data cross validation, oversampling data,* XGBoost *training, cross-validation,* dan *grid search*. XGBoost dipilih sebagai metode pembelajaran mesin karena kemampuannya dalam menangani permasalahan klasifikasi pada data dengan tingkat kompleksitas yang tinggi. Uraian lebih rinci mengenai tahapan pemodelan disajikan pada Modul Program 4.8 berikut ini.

Modul Program 4.8 : Pemodelan

```
smote = SMOTE(
    random_state=42,
    k_neighbors=1)

xgb = XGBClassifier(
    objective="binary:logistic",
    eval_metric="auc",
    use_label_encoder=False,
    n_jobs=-1,
    random_state=42,
    scale_pos_weight=1)

pipe = Pipeline([
    ("smote", smote),
    ("clf", xgb)
])

param_grid = {
    "clf__max_depth": [3, 5, 7, 9],
    "clf__min_child_weight": [1, 3, 5, 7],
    "clf__gamma": [0, 0.1, 0.3, 0.5],
    "clf__subsample": [0.6, 0.8, 1.0],
    "clf__colsample_bytree": [0.6, 0.8, 1.0],
    "clf__learning_rate": [0.01, 0.05, 0.1, 0.2],
    "clf__n_estimators": [100, 200, 500]
}

cv = StratifiedKFold(n_splits=3, shuffle=True, random_state=42)

grid = GridSearchCV(
    estimator=pipe,
    param_grid=param_grid,
    scoring="f1",
    cv=cv,
    verbose=2,
    n_jobs=-1,
    refit=True
)

grid.fit(X_train, y_train)
```

4.4.1 Implementasi *Splitting Data Cross Validation*

Pada tahap ini, data dibagi ke dalam tiga *fold* menggunakan metode *Stratified K-Fold* melalui library *StratifiedKFold()* dari *scikit-learn*, sebagaimana ditunjukkan pada Modul Program 4.8 baris ke-28. Pembagian ini dilakukan dengan mempertahankan proporsi kelas pada setiap *fold* agar distribusi data tetap sama. Parameter *shuffle* digunakan untuk mengacak urutan data sebelum proses pembagian, sehingga mengurangi potensi bias akibat urutan data awal. Sementara itu, parameter *random_state* digunakan untuk memastikan konsistensi hasil pembagian data, sehingga proses eksperimen dapat direplikasi dengan hasil pembagian data yang identik.

4.4.2 Implementasi *Oversampling Data*

Pada tahap ini, dilakukan *oversampling* data menggunakan metode SMOTE yang diimplementasikan melalui library *imblearn*, sebagaimana ditunjukkan pada Modul Program 4.8 baris pertama. SMOTE ditempatkan di dalam *pipeline* bersama dengan model XGBoost untuk memastikan bahwa proses *oversampling* hanya diterapkan pada data training di setiap fold selama proses *cross-validation*. Pendekatan ini bertujuan untuk mencegah terjadinya data kebocoran data antara data *training* dan data *validasi*.

Parameter *k_neighbors* pada SMOTE digunakan untuk menentukan jumlah tetangga terdekat dalam pembentukan sampel sintetis kelas minoritas. Parameter k_neighbors pada SMOTE ditetapkan sebesar 1 pada seluruh proses eksperimen. Pemilihan nilai ini didasarkan pada karakteristik data latih r6.2 yang pada beberapa *fold* dalam proses *cross-validation* hanya memiliki satu sampel pada kelas positif. Dalam kondisi tersebut, penggunaan nilai *k_neighbors* yang lebih besar tidak memungkinkan karena algoritma SMOTE memerlukan minimal k+1 sampel pada kelas minoritas untuk menentukan tetangga terdekat. Oleh karena itu, nilai *k_neighbors=1* dipilih agar proses oversampling tetap dapat dilakukan pada data latih yang terdiri dari gabungan 2 *fold* dengan jumlah sampel minoritas yang sangat terbatas. Nilai parameter ini dipertahankan secara konsisten pada seluruh dataset dalam penelitian untuk menjaga keseragaman konfigurasi eksperimen dan mengurangi variasi yang disebabkan oleh perubahan parameter antar dataset, sehingga hasil evaluasi model dapat dibandingkan secara lebih konsisten.

4.4.3 Implementasi *XGBoost Training*

Pada tahap pelatihan model, algoritma XGBoost diimplementasikan menggunakan fungsi *XGBClassifier()* sebagaimana ditunjukkan pada Modul Program 4.8 baris 5 dengan konfigurasi khusus untuk klasifikasi biner. Parameter *objective="binary:logistic"* digunakan untuk menghasilkan probabilitas kelas positif, sedangkan *eval_metric="auc"* dipilih sebagai metrik evaluasi selama proses pelatihan untuk mengukur kemampuan model dalam membedakan kelas *malicious* dan *non-malicious* secara keseluruhan.

Parameter *n_jobs=-1* digunakan untuk memanfaatkan seluruh inti prosesor yang tersedia guna mempercepat proses pelatihan, sedangkan *random_state* ditetapkan untuk menjaga konsistensi hasil eksperimen. Implementasi XGBoost dalam penelitian ini memanfaatkan karakteristik algoritma *gradient boosting*, di mana model dibangun secara iteratif melalui penambahan pohon keputusan yang berfokus pada kesalahan prediksi pada

iterasi sebelumnya. Pendekatan ini memungkinkan model untuk menangkap pola nonlinier yang kompleks pada data aktivitas pengguna.

Selain itu, parameter *scale_pos_weight=1* digunakan karena ketidakseimbangan kelas pada data telah ditangani pada tahap *oversampling* menggunakan teknik SMOTE. Dengan diterapkannya SMOTE, distribusi kelas pada data latih menjadi lebih seimbang, sehingga tidak diperlukan pembobotan tambahan pada fungsi objektif XGBoost.

4.4.4 Implementasi Cross-*Validation*

Pada tahap cross-validation, evaluasi performa model dilakukan menggunakan fungsi *Stratified K-Fold Cross Validation* yang diimplementasikan melalui library *scikit-learn*.

Dalam setiap iterasi *cross-validation*, model dilatih dan dievaluasi secara bergantian pada subset data yang berbeda, sehingga setiap fold berperan sebagai data validasi satu kali. Mekanisme ini memungkinkan pengukuran performa model yang lebih menyeluruh, karena model tidak hanya diuji pada satu subset data tertentu, tetapi pada seluruh bagian data secara bergantian.

Nilai performa yang diperoleh dari setiap iterasi kemudian digabungkan dan dirata-ratakan untuk menghasilkan estimasi kinerja model yang lebih stabil dan tidak bergantung pada satu skema validasi tertentu. Dengan demikian, cross-validation berperan penting dalam menilai kemampuan generalisasi model sebelum dilakukan tahap optimasi hyperparameter menggunakan grid search.

4.4.5 Implementasi *Grid Search*

Pada tahap *grid search*, optimasi *hyperparameter* dilakukan menggunakan fungsi *GridSearchCV* dari *library scikit-learn*. *Grid search* diterapkan untuk mengevaluasi seluruh kombinasi *hyperparameter* yang telah ditentukan pada *parameter grid* yang berada di dalam *pipeline*.

Setiap kombinasi parameter dievaluasi menggunakan skema *Stratified K-Fold Cross-Validation* yang telah ditetapkan sebelumnya, sehingga proses pelatihan dan validasi dilakukan secara konsisten pada setiap iterasi. Metrik evaluasi yang digunakan dalam proses grid search adalah *F1-score*, yang dipilih karena mampu merepresentasikan keseimbangan antara *precision* dan *recall*, khususnya pada kondisi dataset yang tidak seimbang.

Setelah seluruh kombinasi parameter dievaluasi, *GridSearchCV* akan memilih konfigurasi *hyperparameter* dengan nilai rata-rata *F1-score* tertinggi sebagai parameter optimal. Konfigurasi tersebut kemudian digunakan untuk melatih ulang model pada seluruh data training sehingga model akhir memiliki parameter yang telah dioptimalkan secara sistematis. Dengan pendekatan ini, proses pemilihan *hyperparameter* tidak dilakukan secara manual, melainkan melalui prosedur pencarian yang terstruktur dan terukur, sehingga hasil model yang diperoleh lebih objektif dan dapat direproduksi.

Pada penelitian ini diperoleh hasil *hyperparameter* terbaik seperti pada Tabel 4.6 berikut ini.

Tabel 4.6 Kombinasi Parameter Optimal XGBoost

Parameter	***Default***	***Value***	***Keterangan***
learning_rate	*0.3*	*0.2*	Nilai lebih kecil membuat proses pembelajaran lebih bertahap sehingga model cenderung lebih stabil dan risiko *overfitting* dapat berkurang.
max_depth	*6*	*9*	Kedalaman pohon lebih besar memungkinkan model menangkap pola dan interaksi fitur yang lebih kompleks.
n_estimators	*100*	*500*	Jumlah pohon yang lebih banyak meningkatkan kapasitas model dalam mempelajari pola data.
subsample	*1*	*0.8*	Hanya 80% data digunakan pada setiap pohon, sehingga membantu mengurangi varians dan meningkatkan kemampuan generalisasi.
min_child_weight	*1*	*1*	Sama dengan default, sensitivitas model terhadap pemisahan node tetap seperti konfigurasi standar.
gamma	*0*	*0*	Sama dengan default, pemisahan node dilakukan selama masih memberikan penurunan nilai loss.
colsample_bytree	*1*	*0.8*	Hanya 80% fitur digunakan pada setiap pohon, membantu mengurangi ketergantungan model pada fitur tertentu dan meningkatkan generalisasi.

Parameter pada Tabel 4.6 merupakan hasil proses *hyperparameter tuning* yang digunakan untuk meningkatkan kinerja model XGBoost dibandingkan dengan konfigurasi *default*. Parameter *learning_rate=0.2* lebih kecil dibandingkan nilai *default* sebesar 0.3, sehingga proses pembelajaran berlangsung lebih bertahap dan stabil, serta membantu mengurangi risiko *overfitting*. Parameter *max_depth=9* lebih besar dari nilai *default* yaitu 6, yang memungkinkan pohon keputusan menangkap hubungan fitur yang lebih kompleks. Selain itu, parameter *n_estimators=500* juga lebih tinggi dibandingkan nilai *default* sebesar 100, sehingga jumlah pohon dalam proses *boosting* menjadi lebih banyak dan memberikan kapasitas yang lebih besar bagi model untuk mempelajari pola pada data. Kombinasi parameter tersebut memungkinkan model melakukan pembelajaran yang lebih mendalam sekaligus tetap menjaga stabilitas proses pelatihan.

Parameter *subsample=0.8* dan *colsample_bytree=0.8* lebih rendah dibandingkan nilai default yang masing-masing bernilai 1. Konfigurasi ini membatasi proporsi data dan fitur yang digunakan pada setiap pohon, sehingga proses pelatihan tidak selalu menggunakan seluruh sampel dan fitur yang tersedia. Pendekatan tersebut membantu mengurangi varians model serta meningkatkan kemampuan generalisasi dengan mengurangi ketergantungan pada pola tertentu pada data pelatihan.

Parameter *min_child_weight=1* pada konfigurasi ini sama dengan nilai *default* XGBoost, yang berarti tidak ada pembatasan tambahan pada jumlah minimum bobot sampel pada node daun. Dengan nilai ini, model tetap sensitif terhadap pola pada data dan dapat melakukan pemisahan *node* selama masih terdapat informasi yang relevan. Sementara itu, parameter *gamma=0* juga sama dengan nilai *default*, sehingga pemisahan *node* dapat dilakukan tanpa batasan tambahan selama pemisahan tersebut mampu menurunkan nilai *loss*.

Parameter *refit=True* pada Modul Program 4.8 baris 37 digunakan agar model dilatih ulang menggunakan kombinasi *hyperparameter* terbaik pada seluruh data training setelah proses *grid search* selesai. Dengan pendekatan ini, *grid search* tidak hanya berfungsi untuk memilih konfigurasi parameter yang optimal, tetapi juga menghasilkan model akhir yang siap digunakan pada tahap pengujian dan evaluasi performa.

4.5 Implementasi Pengujian Model

Proses pengujian model pada penelitian ini dilakukan untuk memperoleh hasil evaluasi yang sesuai dengan metodologi yang telah dijelaskan pada Bab 3. Pengujian bertujuan untuk membandingkan performa model tanpa perlakuan khusus, menggunakan teknik penanganan ketidakseimbangan data dengan SMOTE, serta model dengan optimasi dan penanganan ketidakseimbangan data menggunakan SMOTE. Hal ini digunakan untuk menilai kemampuan model dalam mendeteksi adanya aktivitas ancaman orang dalam.

Modul Program 4.9 : Pengujian

```
1  y_prob = model.predict_proba(X_test)[:, 1]
2
3  print(confusion_matrix(y, y_prob))
4  print(classification_report(y, y_prob, digits=4))
5  print("ROC-AUC:", roc_auc_score(y, y_prob))
```

Proses pengujian model sebagaimana ditunjukkan pada Modul Program 4.9 dilakukan dengan memprediksi probabilitas kelas positif pada data testing menggunakan fungsi *predict_proba()*. Probabilitas tersebut kemudian dievaluasi melalui *confusion matrix* dan *classification report* untuk memperoleh nilai *precision, recall,* dan *F1-score.* Selain itu, dihitung *ROC-AUC* untuk menilai kemampuan model dalam membedakan kelas secara keseluruhan berdasarkan distribusi probabilitas, sehingga evaluasi tidak hanya bergantung pada satu *threshold*.

Evaluasi dilakukan menggunakan *dataset test* yang terpisah dari data *training* guna mencegah terjadinya kebocoran data pada tahap pengujian. Kinerja model diukur menggunakan beberapa metrik evaluasi, yaitu precision, recall, F1-score, *Area Under the Receiver Operating Characteristic Curve* (AUC-ROC).

Penggunaan berbagai metrik bertujuan untuk memberikan gambaran menyeluruh mengenai akurasi prediksi model sekaligus efektivitasnya dalam melakukan klasifikasi. Selain pengujian pada data test, evaluasi juga dilakukan pada data *dataset test* versi lain untuk mengamati konsistensi performa model. Hasil pengujian pada *dataset training* versi r4.2 disajikan pada Tabel 4.7.

Tabel 4.7 Hasil Pengujian Model Pada Subset Test Dataset r4.2

Model	**Train Set**	**TP**	**TN**	**FP**	**FN**	**Precision**	**Recall**	**F1**	**AUC-ROC**
XGBoost	r4.2	13	186	0	1	1.0000	0.9286	0.9630	1.0000
XGBoost + SMOTE	**r4.2**	**14**	**185**	**1**	**0**	**0.9333**	**1.0000**	**0.9655**	**0.9992**
XGBoost + SMOTE + *tuning*	**r4.2**	**14**	**185**	**1**	**0**	**0.9333**	**1.0000**	**0.9655**	**0.9992**
XGBoost	r5.2	7	186	0	7	1.0000	0.5000	0.6667	0.9866
XGBoost + SMOTE	r5.2	12	183	3	2	0.8000	0.8571	0.8276	0.9816
XGBoost + SMOTE + *tuning*	r5.2	7	183	3	7	0.7000	0.5000	0.5833	0.9854
XGBoost	r6.2	0	186	0	14	0.0000	0.0000	0.0000	0.6502
XGBoost + SMOTE	r6.2	6	185	1	8	0.8571	0.4286	0.5714	0.7492
XGBoost + SMOTE + *tuning*	r6.2	6	185	1	8	0.8571	0.4286	0.5714	0.7120

Pada Tabel 4.7 ditampilkan hasil pengujian model pada *dataset testing* versi r4.2, dengan total 9 skenario yaitu model XGBoost saja, model XGBoost dengan kombinasi *oversampling* SMOTE, serta model XGBoost dengan kombinasi *oversampling* SMOTE serta *tuning* tambahan. Masing-masing model dilatih dengan 3 versi dataset, yaitu r4.2, r5.2, dan r6.2 guna mengamati konsistensi respon model terhadap versi dataset lain.

Tabel 4.8 Hasil Pengujian Model Pada Subset Test Dataset r5.2

Model	**Train Set**	**TP**	**TN**	**FP**	**FN**	**Precision**	**Recall**	**F1**	**AUC-ROC**
XGBoost	r4.2	9	379	1	11	0.9000	0.4500	0.6000	0.8572
XGBoost + SMOTE	r4.2	11	376	4	9	0.6875	0.5500	0.6111	0.8928
XGBoost + SMOTE + *tuning*	r4.2	10	377	3	10	0.7692	0.5000	0.6061	0.9105
XGBoost	r5.2	15	380	0	5	1.0000	0.7500	0.8571	0.9921
XGBoost + SMOTE	r5.2	18	377	3	2	0.8571	0.9000	0.8780	0.9966
XGBoost + SMOTE + *tuning*	**r5.2**	**18**	**378**	**2**	**2**	**0.9000**	**0.9000**	**0.9000**	**0.9964**
XGBoost	r6.2	0	380	0	20	0.0000	0.0000	0.0000	0.7028
XGBoost + SMOTE	r6.2	2	380	0	18	1.0000	0.1000	0.1818	0.7995
XGBoost + SMOTE + *tuning*	r6.2	2	380	0	18	1.0000	0.1000	0.1818	0.7887

Pada Tabel 4.8 disajikan hasil evaluasi model pada *dataset testing* versi r5.2 dengan skenario pengujian yang sama seperti sebelumnya. Pengujian dilakukan pada 9 konfigurasi, yang terdiri atas model XGBoost tanpa penanganan ketidakseimbangan data, model XGBoost dengan penerapan *oversampling* SMOTE, serta model XGBoost dengan kombinasi SMOTE dan proses *tuning* tambahan.

Setiap konfigurasi model dilatih menggunakan 3 versi dataset, yaitu r4.2, r5.2, dan r6.2, untuk menilai kemampuan generalisasi model ketika diuji pada distribusi data yang berbeda dari versi latihnya. Pendekatan ini memungkinkan analisis yang lebih menyeluruh terhadap stabilitas dan konsistensi performa model lintas versi dataset.

Tabel 4.9 Hasil Pengujian Model Pada Subset Test Dataset r6.2

Model	**Train Set**	**TP**	**TN**	**FP**	**FN**	**Precision**	**Recall**	**F1**	**AUC-ROC**
XGBoost	r4.2	0		0		0.0000	0.0000	0.0000	0.8048
XGBoost + SMOTE	r4.2	0		0		0.0000	0.0000	0.0000	0.6433
XGBoost + SMOTE + *tuning*	r4.2	0		0		0.0000	0.0000	0.0000	0.2196
XGBoost	r5.2	0		0		0.0000	0.0000	0.0000	0.2178
XGBoost + SMOTE	r5.2	0		0		0.0000	0.0000	0.0000	0.2347
XGBoost + SMOTE + *tuning*	r5.2	0		0		0.0000	0.0000	0.0000	0.1464
XGBoost	r6.2	0		0		0.0000	0.0000	0.0000	0.7610
XGBoost + SMOTE	**r6.2**	0		0		**0.0000**	**0.0000**	**0.0000**	**0.9725**
XGBoost + SMOTE + *tuning*	r6.2	0		0		0.0000	0.0000	0.0000	0.9625

Pada Tabel 4.9 ditampilkan hasil pengujian model menggunakan *dataset testing* versi r6.2 dengan skema eksperimen yang identik dengan tabel sebelumnya. Evaluasi dilakukan terhadap 9 konfigurasi, yang mencakup model XGBoost tanpa teknik penanganan ketidakseimbangan data, model XGBoost dengan penerapan oversampling SMOTE, serta model XGBoost yang dikombinasikan dengan SMOTE dan proses *tuning hyperparameter*.

Setiap konfigurasi tersebut dilatih menggunakan 3 variasi dataset, yaitu r4.2, r5.2, dan r6.2, untuk menguji konsistensi dan kemampuan generalisasi model terhadap perbedaan distribusi data antarversi. Dengan pendekatan ini, dapat dianalisis sejauh mana model mampu mempertahankan performa ketika diterapkan pada *dataset* dengan karakteristik jumlah pengguna dan proporsi kelas yang berbeda.

Tabel 4.10 Hasil Pengujian Model Pada Subset Test Dataset r6.2

Model	**Train Set**	**TP**	**TN**	**FP**	**FN**	**Precision**	**Recall**	**F1**	**AUC-ROC**
XGBoost	r4.2	22	1363	2	13	0.9167	0.6286	0.7458	0.9259
XGBoost + SMOTE	r4.2	25	1355	10	10	0.7143	0.7143	0.7143	0.9340
XGBoost + SMOTE + *tuning*	r4.2	24	1358	7	11	0.7742	0.6857	0.7273	0.9289
XGBoost	r5.2	22	882	483	13	0.0436	0.6286	0.0815	0.7610
XGBoost + SMOTE	**r5.2**	**30**	**885**	**480**	**5**	**0.0588**	**0.8571**	**0.1101**	**0.8263**
XGBoost + SMOTE + *tuning*	r5.2	25	906	459	10	0.0517	0.7143	0.0963	0.8133
XGBoost	r6.2	0	1365	0	35	0.0000	0.0000	0.0000	0.7388
XGBoost + SMOTE	**r6.2**	**8**	**1364**	**1**	**27**	**0.8889**	**0.2286**	**0.3636**	**0.8308**
XGBoost + SMOTE + *tuning*	r6.2	8	1364	1	27	0.8889	0.2286	0.3636	0.8025

Pada Tabel 4.10 disajikan hasil pengujian model menggunakan *dataset testing* gabungan yang terdiri dari *dataset testing* versi r4.2, r5.2, dan r6.2. Skenario pengujian yang diterapkan tetap konsisten dengan tabel-tabel sebelumnya, mencakup 9 konfigurasi yaitu model XGBoost, model XGBoost dengan penerapan *oversampling* SMOTE, serta model XGBoost yang dikombinasikan dengan SMOTE dan *tuning hyperparameter* tambahan.

Setiap konfigurasi model dilatih menggunakan 3 versi *dataset*, kemudian diuji pada *dataset* gabungan untuk mengevaluasi performa model pada distribusi data yang lebih heterogen. Pengujian ini bertujuan untuk mengukur stabilitas dan ketahanan model ketika dihadapkan pada variasi karakteristik data yang lebih kompleks, sehingga memberikan gambaran mengenai kemampuan generalisasi model dalam skenario yang lebih beragam.

4.6 Pembahasan

Pengujian dalam penelitian ini dilakukan untuk melihat dan mengevaluasi performa model XGBoost yang dibangun serta pengaruh *oversampling* SMOTE dan *tuning* tambahan terhadap model pada deteksi *insider threat* menggunakan CERT *Insider Threat Dataset*. Penelitian ini menggunakan beberapa parameter pengujian yaitu *precision*, *recall*, *F1-score*, serta *AUC-ROC.* Pengujian dilakukan menggunakan dataset *testing* dengan proporsi 20% dari 3 versi dataset yaitu r4.2, r5.2, dan r6.2 serta gabungan dataset *test* dari 3 versi.

Berdasarkan hasil pengujian pada seluruh subset test dataset, terlihat bahwa performa model sangat dipengaruhi oleh kesesuaian distribusi antara data latih dan data uji. Pada skenario data latih dan data uji pada versi yang sama, model menunjukkan performa yang sangat baik pada r4.2 dan r5.2. Pada versi r4.2, seluruh pendekatan menghasilkan F1-Score di atas 0.96 dengan AUC mendekati 1.0, yang menunjukkan bahwa pola pada dataset ini relatif mudah dipelajari oleh model. Penambahan SMOTE meningkatkan recall hingga 1.0 dengan sedikit penurunan precision, yang merupakan *trade-off* wajar pada kasus ketidakseimbangan kelas. Proses *tuning hyperparameter* belum memberikan peningkatan signifikan, yang mengindikasikan bahwa konfigurasi awal model sudah cukup memadai untuk distribusi ini.

Pada versi dataset r5.2, model XGBoost tanpa perlakuan khusus menunjukkan *precision* sempurna namun recall relatif rendah pada 0.7500, sehingga F1-Score hanya mencapai 0.8571 yang menandakan bahwa model masih melewatkan sebagian kasus positif akibat ketidakseimbangan kelas. Ketika diterapkan SMOTE, recall meningkat signifikan menjadi 0.9000 dengan sedikit penurunan precision menjadi 0.8571, menghasilkan F1-Score 0.8780 dan AUC-ROC 0.9966, yang menunjukkan bahwa *oversampling* SMOTE efektif memperbaiki sensitivitas tanpa mengorbankan kemampuan separasi probabilistik secara berarti. Penambahan tuning hyperparameter kemudian menghasilkan precision dan recall yang seimbang dengan F1-Score 0.9000, namun AUC-ROC mengalami sedikit penurunan menjadi 0.9964 yang mengindikasikan bahwa tuning memberikan peningkatan performa model pada *threshold* 0.5, tetapi faktor dominan yang memengaruhi performa pada r5.2 tetap berasal dari distribusi kelas yang tidak seimbang, bukan dari konfigurasi parameter model.

Sebaliknya, pada r6.2 ditemukan fenomena yang berbeda secara signifikan. Penerapan SMOTE meningkatkan nilai AUC-ROC secara signifikan dari 0.7610 menjadi 0.9725. Ketika model dilatih dan diuji pada r6.2, seluruh konfigurasi menghasilkan F1-Score sebesar 0.0000, meskipun nilai AUC-ROC pada beberapa konfigurasi sangat tinggi mencapai 0.9725 pada konfigurasi XGBoost dengan SMOTE. Kondisi ini menunjukkan bahwa model sebenarnya mampu melakukan klasifikasi dengan baik, tetapi gagal menghasilkan prediksi positif pada threshold default 0.5. Dengan kata lain, permasalahan utama pada r6.2 bukanlah ketidakmampuan model dalam mempelajari pola, melainkan isu kalibrasi *threshold* dan kemungkinan distribusi probabilitas yang sangat condong ke bawah 0.5 akibat ketidakseimbangan kelas atau karakteristik fitur yang berbeda secara signifikan.

Pada skenario data latih dan data uji pada versi yang berbeda, terlihat adanya indikasi kuat terjadinya perbedaan distribusi. Model yang dilatih pada r4.2 atau r5.2 gagal melakukan klasifikasi pada r6.2, ditunjukkan oleh *F1-Score* yang bernilai 0 pada sebagian besar kombinasi. Sebaliknya, model yang dilatih pada r6.2 juga tidak mampu memberikan performa baik pada r4.2 dan r5.2. Hal ini disebabkan karena ketidakseimbangan secara ekstrem pada dataset versi r6.2, sehingga model kesulitan untuk melakukan klasifikasi pada versi r6.2.

Pada pengujian menggunakan dataset gabungan, performa paling konsisten diperoleh dari model yang dilatih pada r4.2, dengan F1-Score berkisar di antara 0.72 hingga 0.75 dan AUC di atas 0.92. Model yang dilatih pada r5.2 menunjukkan *precision* yang rendah meskipun recall tinggi, sehingga menghasilkan F1-Score yang buruk. Hal ini mengindikasikan kecenderungan model untuk terlalu banyak memprediksi kelas positif ketika diuji pada distribusi campuran. Sementara itu, model yang dilatih pada r6.2 hanya menunjukkan perbaikan moderat ketika menggunakan SMOTE, tetapi tetap belum mencapai tingkat performa yang memadai.

Secara keseluruhan, hasil ini menunjukkan bahwa performa tinggi pada skenario data latih dan data uji pada versi yang sama tidak mencerminkan kemampuan generalisasi model. Dataset versi r6.2 terbukti menjadi versi dataset yang paling menantang karena memiliki ketidakseimbangan kelas yang sangan ekstrem. Selain itu, adanya perbedaan mencolok antara *AUC-ROC* yang tinggi dan *F1-Score* yang nol pada beberapa skenario menegaskan pentingnya optimasi *threshold* dan evaluasi berbasis *Precision-Recall* pada kasus ketidakseimbangan kelas. Dengan demikian, dapat disimpulkan bahwa model saat ini belum mampu menangani perubahan distribusi dan masih cukup sensitif terhadap versi dataset, sehingga belum sepenuhnya siap untuk skenario deteksi insider threat pada lingkungan yang dinamis.

BAB V
PENUTUP

Bab ini membahas kesimpulan dan saran dari hasil penelitian yang telah dilakukan sebelumnya.

5.1 Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan terkait penerapan algoritma XGBoost pada deteksi *insider attack* pada CERT *insider threat dataset* diperoleh kesimpulan berikut:

1. Penelitian ini berhasil mengimplementasikan model deteksi *insider attack* pada jaringan komputer menggunakan algoritma XGBoost melalui tahapan yang sistematis, meliputi *preprocessing* data, rekayasa fitur berbasis perilaku pengguna, penanganan ketidakseimbangan kelas menggunakan SMOTE, serta proses pelatihan dan evaluasi model. Implementasi tersebut menunjukkan bahwa XGBoost mampu memodelkan pola aktivitas *insider attack* secara efektif dengan nilai *F1-Score* mencapai 0.9655 pada dataset *training* dan *testing* r4.2 dengan bantuan SMOTE.
2. Hasil evaluasi menunjukkan bahwa optimasi *hyperparameter* menggunakan *grid search* mampu meningkatkan performa model pada beberapa skenario pengujian, meskipun peningkatan tersebut tidak terjadi secara konsisten pada seluruh skenario pengujian. Peningkatan *F1-Score* hanya terlihat pada kondisi tertentu, seperti pada dataset r5.2 dengan nilai 0.87 menjadi 0.9, sementara pada skenario lain peningkatan yang dihasilkan cenderung minimal atau bahkan menurun. Hal ini mengindikasikan bahwa model dengan penerapan SMOTE pada dasarnya sudah berada pada kondisi yang cukup optimal dalam menangani ketidakseimbangan kelas, sehingga ruang peningkatan melalui *tuning hyperparameter* menjadi terbatas. Dengan demikian, peran grid gearch pada penelitian ini tidak menjadi faktor dominan dalam meningkatkan kinerja model.
3. Kinerja model XGBoost menunjukkan performa yang baik pada versi dataset latih dan uji yang sama, dengan nilai *F1-Score* dan *AUC-ROC* yang tinggi, terutama pada versi dataset r4.2 dan r5.2. Penerapan SMOTE meningkatkan kemampuan identifikasi kelas minoritas, khususnya dalam meningkatkan *recall*. *Tuning hyperparameter* pada XGBoost dapat meningkatkan *F1-Score* pada data uji r5.2. meningkatkan Pada pengujian lintas versi, performa model mengalami penurunan signifikan akibat adanya perbedaan distribusi data, terutama pada r6.2. Hal ini menunjukkan bahwa model efektif dalam mengidentifikasi serangan pada distribusi data yang sesuai, tetapi belum mampu menangani distribusi data yang lebih ekstrem.

5.2 Saran

Berdasarkan kesimpulan yang telah diuraikan, penelitian ini memiliki beberapa potensi pengembangan, yaitu :

1. Penelitian selanjutnya dapat diarahkan pada peningkatan strategi *optimasi hyperparameter*, mengingat konfigurasi parameter saat ini belum menghasilkan peningkatan performa model secara signifikan. Upaya yang dapat dilakukan antara lain menerapkan metode optimasi *hyperparameter* yang lebih menyeluruh serta memperluas ruang pencarian pada Grid Search, sehingga diperoleh kombinasi *hyperparameter* yang lebih sesuai dengan karakteristik dataset dan mampu meningkatkan kinerja model.
2. Penelitian selanjutnya dapat difokuskan pada peningkatan kemampuan generalisasi model agar menghasilkan performa yang stabil ketika diuji pada versi dataset berbeda. Langkah yang dapat dilakukan adalah evaluasi lintas versi serta pemilihan fitur yang tidak terlalu bergantung pada karakteristik dataset tertentu. Dengan demikian, dapat bekerja baik pada data pelatihan dan mampu mempertahankan performa yang konsisten pada data yang berbeda.
3. Penelitian selanjutnya dapat dikembangkan dengan melakukan deteksi pada tingkat kejadian atau *event-level*, sehingga analisis tidak hanya dilakukan pada agregasi aktivitas pengguna, tetapi juga pada setiap peristiwa atau aktivitas yang terjadi dalam sistem. Pendekatan ini memungkinkan identifikasi pola perilaku yang lebih spesifik dan detail, sehingga potensi aktivitas mencurigakan dapat terdeteksi lebih dini. Dengan analisis pada tingkat kejadian, model diharapkan mampu memberikan informasi yang lebih rinci dan meningkatkan efektivitas dalam mendeteksi ancaman keamanan.

DAFTAR PUSTAKA

Ali, Anas, Mubashar Husain and Peter Hans. 2025. “Real-Time Detection of Insider Threats Using Behavioral Analytics and Deep Evidential Clustering”, May. <http://arxiv.org/abs/2505.15383>.

Al-shehari, Taher and Rakan A. Alsowail. 2021. “An Insider Data Leakage Detection Using One-hot Encoding, Synthetic Minority Oversampling and Machine Learning Techniques”. *Entropy* 23. <https://doi.org/10.3390/e23101258>.

Bentéjac, Candice, Anna Csörgő and Gonzalo Martínez-Muñoz. 2019. “A Comparative Analysis of XGBoost”, November. <https://doi.org/10.1007/s10462-020-09896-5>.

Bradley, Andrew E. 1997. “The Use of the Area under the {ROC} Curve in the Evaluation of Machine Learning Algorithms”. *Pattern Recognition*. Vol. 30.

Cahyawati, Relita Kurnia, Fadilla Fadwa, Kusuma Agustin and Kinanti Sekar Arum. 2023. “SEMINAR NASIONAL AMIKOM SURAKARTA (SEMNASA) 2023 Perancangan Keamanan Jaringan Menggunakan Metode Firewall Security Port”.

Chawla, Nitesh V, Kevin W Bowyer, Lawrence O Hall and W Philip Kegelmeyer. 2002. “SMOTE: Synthetic Minority Over-Sampling Technique”. *Journal of Artificial Intelligence Research*. Vol. 16.

Chen, Tianqi and Carlos Guestrin. 2016. “XGBoost: A Scalable Tree Boosting System”. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, June. <https://doi.org/10.1145/2939672.2939785>.

Chong, Kaisiang and Nathar Shah. n.d. “Comparison of Naive Bayes and SVM Classification in Grid-Search Hyperparameter Tuned and Non-Hyperparameter Tuned Healthcare Stock Market Sentiment Analysis”. *IJACSA) International Journal of Advanced Computer Science and Applications*. Vol. 13. <www.ijacsa.thesai.org>.

Gheyas, Iffat A. and Ali E. Abdallah. 2016. “Detection and Prediction of Insider Threats to Cyber Security: A Systematic Literature Review and Meta-Analysis”. *Big Data Analytics* 1. <https://doi.org/10.1186/s41044-016-0006-0>.

Glasser, Joshua and Brian Lindauer. 2013. “Bridging the Gap: A Pragmatic Approach to Generating Insider Threat Data”. In: . *Proceedings - IEEE CS Security and Privacy Workshops, SPW 2013*. 98–104. <https://doi.org/10.1109/SPW.2013.37>.

Gong, Yiru, Susu Cui, Song Liu, Bo Jiang, Cong Dong and Zhigang Lu. 2024. “Graph-Based Insider Threat Detection: A Survey”. *Computer Networks* 254: 110757. <https://doi.org/10.1016/J.COMNET.2024.110757> [accessed 26 May 2025].

He, Haibo and Edwardo A. Garcia. 2009. “Learning from Imbalanced Data”. *IEEE Transactions on Knowledge and Data Engineering* 21: 1263–1284. <https://doi.org/10.1109/TKDE.2008.239>.

Inayat, Usman, Mashaim Farzan, Sajid Mahmood, Muhammad Fahad Zia, Shahid Hussain and Fabiano Pallonetto. 2024. “Insider Threat Mitigation: Systematic Literature

Review". *Ain Shams Engineering Journal*, December. <https://doi.org/10.1016/j.asej.2024.103068>.

Kan, Xiu, Yixuan Fan, Jinjie Zheng, Chi hung Chi, Wanqing Song and Aleksey Kudreyko. 2023. "Data Adjusting Strategy and Optimized XGBoost Algorithm for Novel Insider Threat Detection Model". *Journal of the Franklin Institute* 360: 11414–11443. <https://doi.org/10.1016/j.jfranklin.2023.09.004>.

Ke, Guolin, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye and Tie-Yan Liu. 2017. "LightGBM: A Highly Efficient Gradient Boosting Decision Tree". <https://github.com/Microsoft/LightGBM.>.

Madan Kumar, C, Mohammed Rahimanbasha, Jamboju Ruchitha, Shaik Irfan, Burugula Nithin and U G Student. 2025. "International Journal of Communication Networks and Information Security Enhancing Insider Threat Detection in Cloud Environments with Ensemble Learning Techniques". <https://https://ijcnis.org/>.

Mamidanna, Sashi Kumar, C. R.K. Reddy and Akash Gujju. 2022. "Detecting an Insider Threat and Analysis of XGBoost Using Hyperparameter Tuning". In: . *Proceedings - IEEE International Conference on Advances in Computing, Communication and Applied Informatics, ACCAI 2022*. Institute of Electrical and Electronics Engineers Inc. <https://doi.org/10.1109/ACCAI53970.2022.9752509>.

Nugraha, Wahyu. 2021. "PREDIKSI PENYAKIT JANTUNG CARDIOVASCULAR MENGGUNAKAN MODEL ALGORITMA KLASIFIKASI". <https://www.kaggle.com/andrewmvd/heart->.

Nurse, Jason R.C., Oliver Buckley, Philip A. Legg, Michael Goldsmith, Sadie Creese, Gordon R.T. Wright and Monica Whitty. 2014. "Understanding Insider Threat: A Framework for Characterising Attacks". In: . *Proceedings - IEEE Symposium on Security and Privacy*. Institute of Electrical and Electronics Engineers Inc. 2014-January: 214–228. <https://doi.org/10.1109/SPW.2014.38>.

Saito, Takaya and Marc Rehmsmeier. 2015. "The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets". *PLoS ONE* 10. <https://doi.org/10.1371/journal.pone.0118432>.

Sarhan, Bushra Bin and Najwa Altwaijry. 2023. "Insider Threat Detection Using Machine Learning Approach". *Applied Sciences (Switzerland)* 13. <https://doi.org/10.3390/app13010259>.

Singh, Avinash, Vikas Pareek and Asish Sharma. n.d. "'Blockchain-Enabled Zero Trust Framework for Securing FinTech Ecosystems Against Insider Threats and Cyber Attacks'".

Soundrapandiyan, Rajkumar, Adhiyaman Manickam, Moulay Akhloufi, Yarlagadda Vishnu Srinivasa Murthy, Renuka Devi Meenakshi Sundaram and Sivasubramanian Thirugnanasambandam. 2023. "An Efficient COVID-19 Mortality Risk Prediction Model Using Deep Synthetic Minority Oversampling Technique and Convolution Neural Networks". *BioMedInformatics* 3: 339–368. <https://doi.org/10.3390/biomedinformatics3020023>.

Syahwaluddin, Risal and Debby Alita. 2024. "Penerapan Oversampling Pada Klasifikasi Ujaran Kebencian Menggunakan Bidirectional Encoder Representations from

Transformers". *The Indonesian Journal of Computer Science* 13. <https://doi.org/10.33022/ijcs.v13i4.4295>.

Tao, Xiaoling, Yuelin Yu, Lianyou Fu, Jianxiang Liu and Yunhao Zhang. 2023. "An Insider User Authentication Method Based on Improved Temporal Convolutional Network". *High-Confidence Computing* 3. <https://doi.org/10.1016/j.hcc.2023.100169>.

Uddin, Md Sihab, Erphan Bhuiyan, Subrata Sarker, Sajal Das, Niloy Sarker and Md Nazrul Islam Mondal. 2021. "An Intelligent Short-Circuit Fault Classification Scheme for Power Transmission Line". In: . *2021 International Conference on Automation, Control and Mechatronics for Industry 4.0, ACMI 2021*. Institute of Electrical and Electronics Engineers Inc. <https://doi.org/10.1109/ACMI53878.2021.9528200>.

Velarde, Gissel, Anindya Sudhir, Sanjay Deshmane, Anuj Deshmunkh, Khushboo Sharma and Vaibhav Joshi. 2023. "Evaluating XGBoost for Balanced and Imbalanced Data: Application to Fraud Detection", March. <http://arxiv.org/abs/2303.15218>.

Zou, Shihong, Huizhong Sun, Guosheng Xu and Ruijie Quan. 2020. "Ensemble Strategy for Insider Threat Detection from User Activity Logs". *Computers, Materials and Continua* 65: 1321–1334. <https://doi.org/10.32604/cmc.2020.09649>.