BAB II TINJAUAN PUSTAKA

2.1. Landasan Teori

2.1.1. Blockchain dan Ethereum

Blockchain merupakan teknologi buku besar terdistribusi (distributed ledger technology/DLT) yang memungkinkan penyimpanan dan pencatatan data secara terdesentralisasi, transparan, dan tidak dapat dimodifikasi (immutable) [1], [2]. Setiap transaksi yang terjadi dalam jaringan blockchain akan dikemas ke dalam sebuah blok, yang kemudian dihubungkan dengan blok sebelumnya menggunakan fungsi kriptografis hash, sehingga membentuk rantai blok yang resisten terhadap perubahan atau pemalsuan data.

Ethereum, yang diluncurkan oleh Vitalik Buterin pada tahun 2015, merupakan platform blockchain generasi kedua yang memperluas konsep blockchain melampaui sekadar pencatatan transaksi keuangan [12], [13]. Fitur utama Ethereum yang membedakannya dari Bitcoin adalah dukungan terhadap smart contract dan programmable blockchain, yang memungkinkan eksekusi kode program secara otomatis di atas jaringan terdesentralisasi. Ethereum menggunakan mekanisme konsensus Proof of Stake (PoS) sejak September 2022 (The Merge), yang secara signifikan mengurangi konsumsi energi dibandingkan Proof of Work (PoW).

Setiap entitas dalam jaringan Ethereum diidentifikasi oleh alamat wallet, yaitu string heksadesimal 40-karakter yang merepresentasikan kunci publik yang telah di-hash. Sifat pseudonymous dari alamat ini menyebabkan identitas pemilik wallet tidak dapat diketahui secara langsung, hanya pola transaksi yang dapat diamati secara publik melalui blockchain explorer seperti Etherscan [23]. Transaksi pada Ethereum bersifat irreversibel setelah dikonfirmasi, artinya tidak ada mekanisme pembatalan atau pengembalian aset yang hilang akibat penipuan.

Jaringan transaksi Ethereum dapat direpresentasikan sebagai sebuah directed multigraph

$$G=(V,) \tag{2.1}$$

di mana V adalah himpunan alamat wallet (*node*) dan EEE adalah himpunan transaksi yang terjadi antar wallet (*edge*). Representasi graph ini memungkinkan

analisis berbasis struktur jaringan untuk mendeteksi pola anomali, termasuk aktivitas phishing.

2.1.2. Phishing pada Jaringan Ethereum

Phishing merupakan salah satu bentuk serangan social engineering yang bertujuan untuk memperoleh informasi sensitif atau mengalihkan aset digital milik korban secara tidak sah melalui cara-cara yang menipu [3]. Dalam konteks blockchain Ethereum, aktivitas phishing biasanya termanifestasi dalam beberapa bentuk: (1) pembuatan alamat wallet palsu yang menyerupai alamat wallet yang sah; (2) pengelabuan pengguna melalui situs web atau smart contract palsu agar melakukan transfer ke alamat pelaku; serta (3) manipulasi approval token ERC-20 yang memungkinkan pelaku menguras saldo korban.

Wallet phishing pada Ethereum memiliki karakteristik pola transaksi yang dapat diidentifikasi melalui analisis graph. Penelitian Chen et al. [4] menunjukkan bahwa wallet phishing umumnya menunjukkan: (1) rasio in_degree terhadap out_degree yang sangat tinggi, mencerminkan penerimaan dana dari banyak korban; (2) interaksi dengan jumlah alamat unik yang sangat besar dalam periode waktu singkat; (3) pola temporal yang tidak wajar, seperti aktivitas tinggi yang terkonsentrasi dalam jendela waktu sempit; dan (4) nilai transaksi yang bervariasi luas, mencerminkan berbagai ukuran korban.

Deteksi wallet phishing melalui analisis on-chain data menjadi pendekatan yang semakin penting karena: pertama, data transaksi blockchain bersifat publik dan dapat diakses oleh siapa pun; kedua, analisis berbasis pola tidak memerlukan identitas asli pengguna; dan ketiga, hasil deteksi dapat langsung diintegrasikan ke dalam sistem keamanan berbasis blockchain itu sendiri. Pendekatan ini berbeda dari deteksi phishing konvensional yang bergantung pada analisis URL atau konten email.

2.1.3. Machine Learning

Machine Learning (ML) merupakan cabang dari kecerdasan buatan (Artificial Intelligence) yang memungkinkan sistem komputer untuk belajar dari data secara otomatis dan membuat prediksi atau keputusan tanpa diprogram secara eksplisit untuk setiap kasus [6]. Dalam konteks deteksi phising, ML digunakan untuk membangun

model klasifikasi yang dapat membedakan entitas yang mencurigakan dari entitas yang normal berdasarkan fitur-fitur yang diekstrak dari data historis.

Proses umum penerapan Machine Learning untuk klasifikasi mencakup tahapan: (1) pengumpulan dan preprocessing data; (2) feature engineering untuk mengubah data mentah menjadi representasi numerik yang bermakna; (3) pelatihan model menggunakan data berlabel; (4) validasi dan evaluasi performa model; serta (5) deployment model untuk inferensi pada data baru. Pada penelitian ini, keempat tahapan pertama dilakukan secara terstruktur untuk setiap skenario kombinasi fitur.

2.1.4. Feature Engineering

Feature engineering merupakan proses menggunakan pengetahuan domain untuk membentuk representasi fitur yang membuat pola pada data mentah lebih mudah dipelajari oleh algoritma Machine Learning. Proses ini mencakup feature extraction, feature construction, transformation, dan feature selection. Feature extraction mengubah data mentah menjadi atribut terukur; feature construction membentuk variabel baru dari kombinasi atribut yang ada; transformation menyesuaikan skala atau distribusi; sedangkan feature selection mempertahankan fitur yang relevan dan mengurangi fitur yang tidak informatif atau redundan [41].

Kualitas representasi fitur berpengaruh langsung terhadap kemampuan model menangkap hubungan antara data dan target. Fitur yang relevan dapat meningkatkan daya diskriminasi, mengurangi kompleksitas, mempercepat pelatihan, dan membantu menekan overfitting. Sebaliknya, fitur yang redundan, bocor terhadap label, atau dibentuk menggunakan informasi dari data uji dapat menghasilkan evaluasi yang terlalu optimistis. Oleh karena itu, seluruh parameter transformasi dan seleksi fitur harus dipelajari hanya dari data latih, kemudian diterapkan tanpa penyesuaian ulang pada data validasi atau data uji [41].

Pada data transaksi berbentuk graph, feature engineering diperlukan agar struktur relasional dapat digunakan oleh model tabular. Satu wallet dapat direpresentasikan sebagai satu baris, sedangkan incoming edge dan outgoing edge pada 1-hop neighborhood diagregasi menjadi fitur degree, jumlah mitra unik, statistik nilai transaksi, rasio aliran dana, dan rentang waktu aktivitas. Transformasi graph-to-tabular tersebut tidak menghilangkan sumber data asli, tetapi merangkum struktur dan

atribut edge menjadi representasi numerik per wallet yang dapat diproses oleh Logistic Regression, Random Forest, dan XGBoost.

Feature engineering juga mencakup pengayaan melalui sumber yang saling melengkapi. Fitur tambahan hanya bermanfaat apabila membawa informasi diskriminatif yang belum terwakili oleh fitur awal; penambahan fitur yang sangat berkorelasi dapat meningkatkan redundansi tanpa memperbaiki generalisasi. Karena itu, kontribusi suatu kelompok fitur perlu diuji melalui skenario perbandingan atau ablasi pada pembagian data yang identik. Dalam penelitian ini, prinsip tersebut diterapkan melalui perbandingan fitur MulDiGraph, Etherscan, dan BigQuery serta analisis feature importance untuk mengidentifikasi fitur yang paling berpengaruh.

2.1.5. Peningkatan Kinerja Machine Learning melalui Pendekatan Data-Centric dan Model-Centric

Peningkatan kinerja sistem Machine Learning dapat dilakukan melalui dua pendekatan yang saling melengkapi, yaitu data-centric dan model-centric. Pendekatan data-centric mempertahankan model atau protokol evaluasi secara relatif tetap, lalu meningkatkan mutu representasi data melalui pengumpulan, pembersihan, pelabelan, penanganan missing value, pengendalian duplikasi, penyeimbangan kelas, konstruksi fitur, dan pemilihan fitur. Data-centric AI menempatkan pengembangan data latih, pengembangan data inferensi, serta pemeliharaan data sebagai bagian inti dalam pembangunan sistem AI [42].

Pendekatan model-centric berfokus pada peningkatan algoritma dan proses pembelajaran setelah data tersedia. Kegiatannya meliputi pemilihan keluarga model, perancangan arsitektur, penentuan loss function, regularisasi, pembobotan kelas, hyperparameter tuning, threshold keputusan, serta pemilihan model berdasarkan validasi. Contohnya adalah pengaturan class_weight pada Logistic Regression dan Random Forest, scale_pos_weight serta kedalaman pohon pada XGBoost, dan jumlah layer, hidden dimension, dropout, serta neighbor sampling pada GraphSAGE. Perubahan model harus dievaluasi pada data validasi atau data uji yang sama agar peningkatannya tidak tercampur dengan perubahan komposisi data.

Kedua pendekatan tidak bersifat saling menggantikan. Perbaikan data dapat menaikkan batas informasi yang dapat dipelajari, sedangkan perbaikan model menentukan seberapa efektif informasi tersebut dimanfaatkan. Eksperimen yang

mengubah sumber fitur dengan model tetap mengukur kontribusi data-centric; eksperimen yang membandingkan beberapa algoritma pada fitur dan split yang sama mengukur kontribusi model-centric. Desain faktorial seperti ini membantu mengatribusikan peningkatan kinerja secara lebih jelas dan mengurangi risiko kesimpulan yang hanya didasarkan pada satu konfigurasi.

Dalam penelitian ini, strategi data-centric diwujudkan melalui normalisasi alamat wallet, penghapusan duplikat, zero-filling yang dikendalikan, stratified split, undersampling kelas mayoritas, transformasi MultiDiGraph menjadi 18 fitur agregat, dan pengayaan fitur Etherscan serta BigQuery. Strategi model-centric diwujudkan melalui pembandingan Logistic Regression, Random Forest, XGBoost, dan GNN-GraphSAGE; pengaturan bobot kelas; konfigurasi hyperparameter; serta evaluasi F1-Score, precision, recall, ROC-AUC, PR-AUC, dan confusion matrix. Dengan demikian, peningkatan F1 tidak diposisikan hanya sebagai akibat pemilihan model, tetapi dianalisis sebagai hasil interaksi antara kualitas data, representasi fitur, dan kapasitas model.

2.1.6. Logistic Regression

Logistic Regression merupakan algoritma klasifikasi statistik yang memodelkan probabilitas suatu instance termasuk dalam kelas tertentu menggunakan fungsi sigmoid (logistic function) [6]. Meskipun namanya mengandung kata "regression", algoritma ini digunakan untuk klasifikasi biner maupun multi-kelas. Model Logistic Regression direpresentasikan sebagai berikut:

$$P(y=1 \mid x)=\sigma(w^{Tx}+b)=\frac{1}{1+e^{-(w^{Tx}+b)}} \tag{2.2}$$

di mana w adalah vektor bobot, x adalah vektor fitur, b adalah bias, dan σ adalah fungsi sigmoid. Model dilatih dengan meminimalkan fungsi binary cross-entropy loss menggunakan gradient descent atau metode optimasi lainnya.

Kelebihan Logistic Regression adalah kemudahan interpretasi, kecepatan pelatihan, dan performa yang baik ketika hubungan antarfitur mendekati linear. Keterbatasannya adalah kemampuan yang lebih terbatas dalam menangkap hubungan

non-linear dan interaksi fitur yang kompleks. Karena itu, Logistic Regression lazim digunakan sebagai baseline yang transparan dalam studi klasifikasi.

2.1.7. Random Forest

Random Forest merupakan algoritma ensemble learning yang membangun sejumlah besar pohon keputusan (decision tree) secara paralel menggunakan teknik bagging (Bootstrap Aggregating) dan random feature selection [7]. Setiap pohon dilatih pada subset data yang diambil dengan penggantian (bootstrap sample) dan hanya mempertimbangkan subset fitur secara acak pada setiap split. Prediksi akhir dihasilkan melalui voting mayoritas (untuk klasifikasi) dari seluruh pohon.

Secara matematis, prediksi Random Forest untuk klasifikasi dapat ditulis sebagai:

$$\hat{y} = \text{mode}\{h_t(x)\}_{t=1}^{T} \quad (2.3)$$

di mana $h_t(x)$ adalah prediksi pohon ke-t dan T adalah jumlah total pohon. Mekanisme bagging dan random subspace secara efektif mengurangi variance model dibandingkan satu pohon keputusan tunggal, sehingga menghasilkan model yang lebih robust terhadap overfitting.

Kelebihan Random Forest meliputi kemampuan menangani hubungan non-linear, ketahanan relatif terhadap outlier dan fitur yang kurang relevan, serta penyediaan feature importance berbasis penurunan impurity. Algoritma ini juga dapat menangani ketidakseimbangan kelas melalui pembobotan kelas. Karakteristik tersebut menjadikannya baseline ensemble yang kuat untuk klasifikasi data tabular.

Dalam kajian klasifikasi, Random Forest umum dipakai sebagai pembanding ensemble karena menggabungkan banyak pohon yang saling independen. Hasilnya dapat digunakan untuk membandingkan kapasitas model ensemble dengan model linear maupun boosting tanpa mengasumsikan bahwa satu metode selalu unggul pada semua dataset.

2.1.8. XGBoost (Extreme Gradient Boosting)

XGBoost (Extreme Gradient Boosting) merupakan implementasi yang dioptimalkan dari algoritma gradient boosting yang dikembangkan oleh Chen dan Guestrin (2016) [19]. Berbeda dengan Random Forest yang membangun pohon secara paralel dan independen, XGBoost membangun pohon secara sekuensial di mana setiap pohon baru berusaha memperbaiki kesalahan pohon-pohon sebelumnya.

Objektif yang dioptimalkan pada XGBoost terdiri dari dua komponen:

$$L(\theta)=\sum_i l(y_i,\hat{y}_i)+\sum_k \Omega(f_k) \tag{2.4}$$

di mana komponen pertama adalah fungsi loss (binary cross-entropy untuk klasifikasi biner) dan komponen kedua adalah regularisasi

$$\Omega(f_k)=\gamma T+\frac{1}{2}\lambda|w|^2 \tag{2.5}$$

yang menghukum kompleksitas pohon (T = jumlah daun, w = bobot daun). Regularisasi bawaan ini membuat XGBoost lebih resistant terhadap overfitting dibandingkan implementasi gradient boosting lainnya.

Untuk menangani ketidakseimbangan kelas, XGBoost menyediakan parameter scale_pos_weight yang dapat memberi bobot lebih tinggi pada kelas minoritas. Secara umum, nilai awal parameter ini dapat diturunkan dari rasio jumlah sampel negatif terhadap positif pada data latih, lalu divalidasi melalui proses pelatihan dan evaluasi yang terpisah dari data uji.

XGBoost banyak digunakan pada data tabular karena membangun pohon secara bertahap, mendukung regularisasi L1/L2, pembobotan kelas, serta feature importance. Kombinasi tersebut membuatnya relevan untuk masalah klasifikasi dengan hubungan fitur non-linear dan distribusi kelas yang tidak seimbang.

2.1.9. Graph Neural Network dan GraphSAGE

Graph Neural Network (GNN) adalah kelas model deep learning yang dirancang secara khusus untuk beroperasi langsung pada data yang memiliki struktur graph [20], [21], [22]. Berbeda dengan model tabular yang memerlukan fitur direpresentasikan sebagai vektor independen, GNN memanfaatkan informasi

struktural dari graph melalui mekanisme message passing antar node. Mekanisme ini memungkinkan setiap node untuk mengumpulkan dan mengagregasi informasi dari tetangga-tetangganya secara iteratif.

Secara umum, mekanisme message passing pada GNN dapat ditulis sebagai:

$$h_v^{(l+1)} = \mathrm{UPDATE}^{(l)}\left(h_v^{(l)}, \mathrm{AGGREGATE}^{(l)}\{h_u^{(l)} : u \in N(v)\}\right) \quad (2.6)$$

di mana $h^{(l)}_v$ adalah representasi (embedding) node v pada layer ke-l, N(v) adalah himpunan tetangga node v, AGGREGATE adalah fungsi agregasi (mean, max, atau LSTM), dan UPDATE adalah fungsi pembaruan (biasanya linear transformation diikuti non-linearity).

GraphSAGE (Graph Sample and AggreGatE) merupakan varian GNN yang dikembangkan oleh Hamilton et al. [20] yang mengatasi keterbatasan GCN (Graph Convolutional Network) konvensional dalam menggeneralisasi ke node baru yang tidak terlihat saat training (inductive learning). GraphSAGE menggunakan strategi neighbor sampling: pada setiap iterasi, hanya subset tetangga yang diambil secara acak, bukan semua tetangga. Ini membuat GraphSAGE lebih skalabel untuk graph besar.

Fungsi agregasi pada GraphSAGE V2 (SAGEConv) yang digunakan dalam penelitian ini:

$$h_v^{(l+1)} = \sigma\,¿ \quad (2.7)$$

Dalam penerapan GraphSAGE, jumlah layer, dimensi hidden, dropout, serta strategi sampling tetangga ditentukan pada tahap perancangan eksperimen. Pemilihan konfigurasi perlu mempertimbangkan ukuran graph, kepadatan koneksi, kompleksitas komputasi, dan risiko overfitting.

GraphSAGE relevan untuk deteksi fraud pada jaringan transaksi karena dapat menggabungkan atribut node dengan pola hubungan antarnode melalui message passing. Namun, manfaatnya bergantung pada kualitas dan kepadatan topologi: ketika

node memiliki sedikit hubungan, informasi tetangga yang dapat diagregasi juga terbatas.

2.1.9.1. Graph–Tabular Fusion dan Late Fusion

Graph–tabular fusion menggabungkan dua representasi yang saling melengkapi untuk wallet yang sama: representasi struktural dari graf transaksi dan atribut tabular yang tidak bergantung pada pesan antarnode. Penyatuan dilakukan melalui alamat wallet yang telah dinormalisasi, bukan dengan menambah populasi atau label baru. Pendekatan ini relevan ketika topologi graf menyediakan pola relasional, sedangkan atribut tabular seperti aktivitas alamat dan nilai transaksi menyimpan sinyal perilaku langsung. Secara konseptual, early fusion memasukkan seluruh fitur ke satu representasi sebelum klasifikasi. Sebaliknya, *late fusion* terlebih dahulu menghasilkan embedding graf z_i dari GraphSAGE, kemudian menggabungkannya dengan vektor tabular e_i pada tahap klasifikasi sebagai berikut:

$$h_i = [z_i \parallel e_i] \tag{2.8}$$

di mana h_i adalah representasi fusi wallet ke-i, z_i adalah embedding graf hasil GraphSAGE, dan e_i adalah vektor fitur tabular Etherscan untuk wallet yang sama.

Desain ini memungkinkan kontribusi setiap cabang diuji melalui ablasi. Dynamic multimodal fusion berbasis teks/BERT, seperti ETH-GBERT [40], merupakan contoh lain yang memakai modalitas berbeda dan tidak identik dengan late fusion graph–tabular. Dalam graph–tabular late fusion, cabang graf dan cabang tabular perlu dibuat non-redundan agar setiap sumber informasi hanya masuk satu kali. Embedding graf dan atribut tabular kemudian dapat dikonkatenasi atau digabung dengan mekanisme lain sebelum classifier akhir dilatih.

2.1.10. Imbalanced Dataset dan Teknik Penanganannya

Imbalanced dataset atau ketidakseimbangan kelas (class imbalance) adalah kondisi ketika distribusi label kelas tidak merata secara signifikan antara kelas mayoritas dan kelas minoritas [6]. Dalam deteksi fraud dan phishing, kondisi ini sering terjadi karena aktivitas fraud jauh lebih jarang daripada aktivitas normal.

Masalah utama yang ditimbulkan oleh class imbalance adalah bias model terhadap kelas mayoritas. Akibatnya, accuracy saja dapat memberi gambaran yang menyesatkan. Evaluasi perlu mempertimbangkan F1-Score, precision, recall, dan confusion matrix agar kemampuan mendeteksi kelas minoritas dapat dinilai secara lebih seimbang.

Terdapat tiga pendekatan utama untuk menangani class imbalance:

- Resampling: meliputi oversampling (memperbanyak sampel kelas minoritas) dan undersampling (mengurangi sampel kelas mayoritas). SMOTE (Synthetic Minority Over-sampling Technique) adalah teknik oversampling yang umum digunakan, yang membuat sampel sintetis interpolasi antara sampel minoritas yang ada.
- Cost-sensitive learning: memberikan bobot lebih tinggi pada misklasifikasi kelas minoritas melalui parameter seperti class_weight=balanced (Logistic Regression, Random Forest) atau scale_pos_weight (XGBoost).
- Threshold adjustment: mengubah threshold keputusan dari 0,5 default ke nilai yang lebih rendah untuk meningkatkan recall kelas minoritas dengan trade-off penurunan precision.

Pemilihan strategi resampling, cost-sensitive learning, maupun threshold adjustment perlu disesuaikan dengan ukuran data, tingkat ketidakseimbangan, tujuan deteksi, dan trade-off antara precision serta recall. Implementasi strategi yang dipilih untuk penelitian ini dijelaskan pada Bab III.

2.1.11. Metrik Evaluasi Model Klasifikasi

Evaluasi model klasifikasi pada kasus imbalanced dataset memerlukan pendekatan metrik yang komprehensif. Titik awal evaluasi adalah Confusion Matrix, yaitu tabel yang merangkum performa prediksi model dengan memisahkan empat jenis hasil klasifikasi:

Tabel II.1. Komponen Confusion Matrix

True Positive (TP)	Wallet phishing diprediksi sebagai phishing (benar)
True Negative (TN)	Wallet normal diprediksi sebagai normal (benar)

False Positive (FP)	Wallet normal diprediksi sebagai phishing (salah — False Alarm)
False Negative (FN)	Wallet phishing diprediksi sebagai normal (salah — Miss)

Tabel II. 1 Komponen Confusion Matrix

Dalam konteks deteksi phishing, False Negative (FN) lebih berbahaya daripada False Positive (FP), karena FN berarti wallet phishing tidak terdeteksi dan korban tidak mendapat peringatan. Metrik-metrik yang digunakan dalam penelitian ini:

Tabel II.2. Metrik Evaluasi Model yang Digunakan

Accuracy	Proporsi prediksi benar dari seluruh prediksi: $$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} (2.9)$$
Precision	Proporsi prediksi phishing yang benar: $$\text{Precision} = \frac{TP}{TP + FP} (2.10)$$ Tinggi = sedikit false alarm
Recall (Sensitivity)	Proporsi phishing yang berhasil terdeteksi: $$\text{Recall} = \frac{TP}{TP + FN} (2.11)$$ Tinggi = sedikit miss
F1-Score	Rata-rata harmonik Precision dan Recall: $$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} (2.12)$$ Metrik utama
FN Rate	Proporsi phishing yang tidak terdeteksi: $$\text{FNR} = \frac{FN}{TP + FN} (2.13)$$ Kritis untuk keamanan

Tabel II. 2 Metrik Evaluasi Model yang Digunakan

F1-Score dipilih sebagai metrik utama karena merupakan rata-rata harmonik antara Precision dan Recall, sehingga memberikan penilaian yang seimbang terhadap

performa deteksi phishing (recall tinggi) dan penghematan false alarm (precision tinggi). Model dengan F1 tinggi secara simultan mendeteksi banyak phishing dan tidak terlalu banyak memberikan false alarm, yang merupakan trade-off ideal untuk sistem peringatan dini keamanan blockchain.

2.1.12. Smart Contract dan Risk Registry

Smart contract merupakan program komputer yang tersimpan dan berjalan secara otomatis di dalam jaringan blockchain, dieksekusi ketika kondisi yang telah ditentukan sebelumnya terpenuhi [8]. Konsep ini pertama kali diperkenalkan oleh Nick Szabo pada tahun 1996 sebagai "computerized transaction protocol that executes the terms of a contract." Pada platform Ethereum, smart contract diimplementasikan menggunakan bahasa Solidity dan dikompilasi menjadi bytecode yang dieksekusi oleh Ethereum Virtual Machine (EVM).

Karakteristik utama smart contract yang membuatnya relevan untuk pencatatan risk score dalam penelitian ini adalah: (1) immutability — data yang tersimpan tidak dapat diubah atau dihapus setelah dikonfirmasi; (2) transparency — seluruh transaksi dapat dilihat oleh siapa pun melalui blockchain explorer; (3) timestamping — setiap pencatatan memiliki waktu yang terverifikasi secara kriptografis; serta (4) decentralization — tidak ada pihak tunggal yang dapat memanipulasi data.

Dalam penelitian ini, smart contract RiskRegistry dikembangkan menggunakan Solidity 0.8.20 dan di-deploy pada jaringan lokal Hardhat. Kontrak ini memiliki tiga fungsi utama: setRisk(address wallet, uint16 score, string model) untuk menyimpan risk score, getRisk(address wallet) untuk membaca data risk score, dan event RiskUpdated untuk mencatat audit trail. Gas cost untuk operasi setRisk sangat konsisten karena menggunakan operasi SSTORE pertama ke slot storage baru yang memiliki biaya tetap 20.000 gas unit.

Perlu dicatat bahwa implementasi pada jaringan Hardhat localhost dalam penelitian ini merupakan proof-of-concept untuk membuktikan feasibility integrasi ML-blockchain. Pada deployment nyata di Ethereum mainnet, latency per transaksi akan berkisar 12-15 detik (satu block time) dengan biaya gas yang bergantung pada kondisi pasar. Jaringan Layer-2 seperti Polygon atau Arbitrum dapat menjadi alternatif yang lebih efisien untuk deployment produksi.

Penerapan blockchain sebagai lapisan keterlacakan juga didukung oleh Falgenti et al., yang merancang jaringan Hyperledger Fabric dan smart contract untuk menelusuri asal komoditas serta mempertahankan jejak transaksi antarpihak [67]. Prinsip tersebut sejalan dengan penggunaan RiskRegistry dalam penelitian ini, meskipun objek dan platform blockchain yang digunakan berbeda.

2.1.13. Dataset EPTransNet dan Sumber Data Hybrid

Ethereum Phishing Transaction Network (EPTransNet) merupakan dataset graph transaksi Ethereum yang dikembangkan oleh kelompok riset XBlock dari Sun Yat-sen University [24]. Dataset ini merepresentasikan jaringan transaksi Ethereum dalam bentuk NetworkX MultiDiGraph (graph berarah dengan kemungkinan multiple edges antar node yang sama), di mana setiap node merepresentasikan satu alamat wallet Ethereum dan setiap edge merepresentasikan satu transaksi yang pernah terjadi antara dua alamat.

Label phishing pada EPTransNet (atribut isp=1) diperoleh dari daftar alamat wallet yang telah teridentifikasi sebagai phisher oleh komunitas keamanan blockchain dan laporan penipuan yang dikumpulkan oleh XBlock. Dataset ini berisi 2.973.489 node dengan 1.165 di antaranya berlabel phishing, menghasilkan rasio ketidakseimbangan 1:2.550. Periode data mencakup transaksi historis Ethereum hingga tahun 2019.

Untuk memperkaya informasi per node, penelitian ini menggabungkan EPTransNet dengan dua sumber data tambahan: (1) Etherscan API, yang menyediakan data on-chain seperti jumlah transaksi, saldo saat ini, usia akun, dan flag kontrak; serta (2) BigQuery Ethereum Public Dataset, yang menyediakan statistik transaksi historis yang lebih lengkap. Penggabungan ketiga sumber data ini menghasilkan dataset hybrid 8.595 alamat wallet dengan maksimal 36 fitur per node.

2.1.14. Kualitas Data dan Keandalan Label

Kualitas data merupakan prasyarat utama bagi model klasifikasi yang dapat digeneralisasi. Data yang berukuran besar tidak otomatis menghasilkan model yang baik apabila memuat duplikasi, label yang tidak konsisten, periode observasi yang tidak seragam, atau atribut yang tidak mewakili fenomena target. Dalam pendekatan data-centric, mutu dataset ditinjau melalui kelengkapan, konsistensi, validitas,

keunikan, ketepatan waktu, dan keterlacakan sumber. Dokumentasi dataset juga perlu menjelaskan tujuan pengumpulan, komposisi, proses pembersihan, batas penggunaan, dan potensi bias agar hasil eksperimen dapat ditafsirkan secara tepat [43].

Pada deteksi phishing Ethereum, keandalan label memiliki tantangan khusus karena alamat yang belum diberi label phishing tidak selalu identik dengan alamat yang benar-benar normal. Sebagian alamat berbahaya mungkin belum dilaporkan atau baru dikenali setelah periode dataset berakhir. Konsekuensinya, kelas negatif dapat mengandung label noise. Kondisi ini berbeda dari kesalahan pencatatan biasa karena label yang tampak sah pada waktu pengumpulan dapat berubah setelah tersedia bukti baru. Oleh sebab itu, istilah non-phishing atau tidak berlabel phishing perlu digunakan secara hati-hati ketika menjelaskan kelas negatif.

Label noise dapat menurunkan batas performa yang dapat dicapai model dan mengaburkan pola fitur yang sebenarnya. Confident learning menunjukkan bahwa ketidakpastian label dapat dianalisis dengan membandingkan probabilitas prediksi out-of-sample dan label observasi untuk menemukan contoh yang patut diaudit [44]. Namun, teknik tersebut tidak menggantikan verifikasi sumber. Pada data blockchain, pemeriksaan dapat dilakukan melalui pencocokan ulang dengan daftar label Etherscan, riwayat transaksi, hubungan dengan alamat yang telah dikonfirmasi, serta waktu pertama dan terakhir label diketahui.

Duplikasi wallet juga perlu ditangani sebelum pembagian data. Apabila alamat yang sama muncul pada data latih dan data uji, model dapat memperoleh keuntungan dari informasi yang secara praktis sudah pernah dilihat sehingga estimasi generalisasi menjadi terlalu optimistis. Normalisasi alamat ke format lowercase, penghapusan spasi, validasi panjang alamat, dan deduplikasi berbasis wallet_address merupakan kontrol minimum. Untuk data gabungan, proses join harus bersifat deterministik dan mencatat jumlah matched, unmatched, serta duplicated key dari setiap sumber.

Dalam penelitian ini, EPTransNet berfungsi sebagai populasi utama dan sumber label, sedangkan Etherscan serta BigQuery digunakan sebagai pengayaan fitur. Pemisahan peran ini penting agar penambahan sumber data tidak secara diam-diam mengubah definisi target. Audit kualitas perlu mencatat berapa wallet yang memiliki fitur tambahan, berapa yang memperoleh zero-filling, dan apakah proporsi missing

berbeda antara kelas phishing serta non-phishing. Jika missingness berkorelasi dengan kelas, indikator ketersediaan sumber dapat menjadi fitur, tetapi penggunaannya harus dijelaskan agar model tidak hanya mempelajari pola cakupan API.

2.1.15. Missing Value, Outlier, dan Robustness Data Transaksi

Missing value pada data transaksi dapat muncul karena beberapa mekanisme. Nilai dapat benar-benar tidak ada karena wallet belum melakukan jenis aktivitas tertentu; nilai dapat tidak terambil karena batas API, kegagalan kueri, atau perbedaan periode observasi; dan nilai dapat hilang akibat ketidakcocokan alamat ketika penggabungan. Ketiga kondisi tersebut memiliki makna yang berbeda. Nilai nol tepat digunakan ketika ketiadaan data memang berarti tidak ada aktivitas, tetapi berisiko menyesatkan apabila data tidak tersedia karena kegagalan ekstraksi.

Strategi imputasi harus ditentukan berdasarkan semantik fitur, bukan hanya kemudahan implementasi. Untuk tx_count atau total_value, zero-filling dapat bermakna tidak terdapat transaksi pada sumber tambahan. Untuk fitur statistik seperti rata-rata atau standar deviasi, nol perlu dibaca bersama jumlah transaksi karena rata-rata nol pada wallet tanpa transaksi berbeda dari rata-rata nol pada wallet aktif.

Indikator seperti etherscan_present, bigquery_present, atau graph_present membantu model dan pembaca membedakan nilai observasi dengan nilai hasil imputasi. Outlier pada blockchain tidak selalu merupakan kesalahan. Nilai transaksi sangat besar, degree tinggi, atau aktivitas sangat singkat dapat menjadi karakteristik penting dari exchange, smart contract, mixer, maupun phishing campaign. Karena itu, penghapusan outlier secara otomatis dapat membuang sinyal fraud yang justru ingin dideteksi. Analisis perlu membedakan data error, extreme legitimate behavior, dan extreme suspicious behavior. Pemeriksaan distribusi, kuantil, alamat sumber, serta konsistensi antarfitur membantu menentukan perlakuan yang tepat.

Metode Interquartile Range dapat digunakan untuk menandai nilai ekstrem melalui batas Q1 - 1,5 IQR dan Q3 + 1,5 IQR. Penandaan tersebut sebaiknya diposisikan sebagai alat audit, bukan aturan penghapusan universal. Winsorization dapat membatasi pengaruh nilai ekstrem pada persentil tertentu tanpa menghilangkan baris, sedangkan transformasi log1p dapat mengurangi skewness untuk fitur non-

negatif seperti nilai dan jumlah transaksi. Semua batas harus dihitung dari training set untuk mencegah informasi test set masuk ke preprocessing.

Robustness perlu diuji dengan membandingkan performa sebelum dan sesudah perlakuan outlier. Jika F1 meningkat tetapi recall phishing turun, perbaikan rata-rata mungkin mengorbankan kasus berisiko tinggi. Analisis juga dapat memisahkan kinerja berdasarkan kelompok degree rendah, sedang, dan tinggi untuk mengetahui apakah model hanya kuat pada wallet yang memiliki banyak transaksi. Dengan demikian, keputusan preprocessing didasarkan pada pengaruh terhadap generalisasi dan risiko false negative, bukan sekadar membuat distribusi terlihat lebih normal.

2.1.16. Data Leakage, Validasi, dan Reproducibility

Data leakage terjadi ketika informasi yang seharusnya tidak tersedia pada waktu prediksi digunakan selama pelatihan. Leakage dapat berasal dari penggunaan test set dalam preprocessing, duplikasi entitas antarsplit, atribut yang dibentuk setelah label diketahui, atau relasi graph yang menghubungkan node test dengan label training secara tidak terkendali. Dampaknya adalah performa evaluasi terlihat tinggi tetapi tidak bertahan ketika model diterapkan pada wallet baru. Pada graph learning, pemisahan data memerlukan perhatian tambahan. Walaupun label node test disembunyikan, struktur edge dan fitur tetangganya dapat tetap membawa informasi transduktif. Desain transductive dan inductive sama-sama valid, tetapi harus dinyatakan dengan jelas. Untuk klaim kemampuan mendeteksi wallet baru, evaluasi inductive lebih sesuai karena embedding test dibentuk tanpa menggunakan label test. GraphSAGE dirancang mendukung generalisasi inductive melalui fungsi agregasi yang dapat diterapkan pada node yang belum pernah dilatih [20].

Cross-validation membantu mengurangi ketergantungan hasil pada satu pembagian data. Namun, ketika hyperparameter dipilih berdasarkan fold yang sama dan performanya dilaporkan sebagai estimasi akhir, hasil tetap dapat bias. Nested cross-validation memisahkan inner loop untuk pemilihan konfigurasi dan outer loop untuk estimasi performa. Varma dan Simon menunjukkan bahwa pemilihan model dan estimasi error pada data yang sama dapat menghasilkan bias optimistis [45]. Untuk dataset besar, held-out test tetap dapat digunakan selama seluruh keputusan dibuat tanpa mengaksesnya.

Reproducibility membutuhkan dokumentasi lebih dari sekadar nama algoritma. Random seed, versi library, parameter sampling, definisi setiap fitur, periode data, aturan join, konfigurasi split, threshold, dan perangkat komputasi perlu dicatat. Sculley et al. menjelaskan bahwa dependensi data dan konfigurasi yang tersembunyi dapat membentuk technical debt dalam sistem Machine Learning [46]. Konfigurasi yang tidak terdokumentasi membuat hasil sulit diulang dan sulit dibedakan antara peningkatan metodologis dengan perubahan lingkungan.Cross-validation membantu mengurangi ketergantungan hasil pada satu pembagian data. Namun, ketika hyperparameter dipilih berdasarkan fold yang sama dan performanya dilaporkan sebagai estimasi akhir, hasil tetap dapat bias. Nested cross-validation memisahkan inner loop untuk pemilihan konfigurasi dan outer loop untuk estimasi performa. Varma dan Simon menunjukkan bahwa pemilihan model dan estimasi error pada data yang sama dapat menghasilkan bias optimistis [45]. Untuk dataset besar, held-out test tetap dapat digunakan selama seluruh keputusan dibuat tanpa mengaksesnya.

Penelitian ini menggunakan random_state=42 dan split yang identik untuk empat skenario agar perbandingan fitur bersifat adil. Pengendalian tersebut perlu dipertahankan pada Logistic Regression, Random Forest, XGBoost, dan GNN. Apabila model membutuhkan format data berbeda, indeks wallet pada train, validation, dan test tetap harus merujuk pada entitas yang sama. Dengan demikian, selisih performa lebih dapat diatribusikan pada representasi atau algoritma, bukan pada perbedaan tingkat kesulitan sampel.

2.1.17. Fitur Struktural Graph untuk Deteksi Anomali Transaksi

Graph-based anomaly detection memanfaatkan fakta bahwa perilaku tidak hanya tercermin pada atribut individu, tetapi juga pada pola hubungan antarpelaku. Survei Akoglu, Tong, dan Koutra mengelompokkan anomali graph ke tingkat node, edge, subgraph, dan perubahan dinamis [48]. Dalam jaringan Ethereum, wallet phishing dapat muncul sebagai node dengan pola aliran tidak seimbang, edge bernilai tidak lazim, ego-network berbentuk bintang, atau subgraph dengan pergerakan dana cepat menuju alamat tertentu.

Fitur degree menangkap jumlah transaksi masuk dan keluar, sedangkan unique_senders dan unique_receivers membedakan banyaknya transaksi dari keragaman counterpart. Dua wallet dapat memiliki in_degree yang sama tetapi pola yang berbeda: wallet pertama menerima berulang kali dari satu alamat, sedangkan

wallet kedua menerima satu kali dari banyak korban. Karena itu, kombinasi degree dan jumlah counterpart unik lebih informatif daripada salah satunya secara terpisah.
Fitur nilai transaksi menggambarkan volume dan variasi dana. total_received serta total_sent menangkap skala aktivitas, sedangkan mean, minimum, maksimum, dan standar deviasi membedakan transaksi homogen dari transaksi yang sangat bervariasi. Rasio sent_received mengukur kecenderungan wallet mempertahankan atau meneruskan dana. Penambahan konstanta kecil pada penyebut diperlukan untuk mencegah pembagian nol dan definisinya harus konsisten antara training serta inference.

Fitur temporal seperti timestamp_spread_in dan timestamp_spread_out merangkum panjang jendela aktivitas, tetapi belum menggambarkan urutan detail transaksi. Dua wallet dapat memiliki rentang waktu sama dengan distribusi aktivitas berbeda. Pengembangan selanjutnya dapat menambahkan burstiness, inter-arrival time, jumlah transaksi per jendela, serta perubahan pola dari waktu ke waktu. Namun, fitur temporal harus dibentuk menggunakan batas observasi yang konsisten agar tidak memanfaatkan transaksi setelah waktu prediksi.

Agregasi 1-hop menawarkan kompromi antara informasi lokal, biaya komputasi, dan kemudahan interpretasi. Neighborhood yang lebih jauh dapat menangkap jaringan pencucian dana, tetapi juga meningkatkan noise dan risiko memasukkan hub besar yang tidak relevan. GNN melakukan agregasi secara learned melalui message passing, sedangkan graph-to-tabular menggunakan statistik yang ditentukan sebelumnya. Perbandingan keduanya menilai apakah fleksibilitas representasi learned memberikan keuntungan nyata pada graph yang tersedia.

2.1.18. Generalisasi, Concept Drift, dan Monitoring Model

Model phishing yang baik pada data historis belum tentu mempertahankan kinerja ketika pola serangan berubah. Concept drift terjadi ketika hubungan antara fitur dan label berubah, sedangkan data drift terjadi ketika distribusi input berubah walaupun hubungan target belum tentu berubah. Pada Ethereum, drift dapat dipicu oleh jenis scam baru, perubahan biaya transaksi, migrasi aktivitas ke token atau layer-2, penggunaan smart contract baru, dan adaptasi pelaku terhadap sistem deteksi.

Monitoring perlu memeriksa distribusi fitur, proporsi missing, jumlah wallet baru, confidence prediction, tingkat positive prediction, dan metrik berbasis label ketika ground truth tersedia. Population Stability Index, Kolmogorov-Smirnov test, atau perbandingan kuantil dapat digunakan sebagai indikator perubahan distribusi. Alarm drift tidak langsung membuktikan penurunan akurasi, tetapi menandai kebutuhan audit atau pelabelan tambahan.

Label phishing sering tersedia terlambat sehingga evaluasi online memiliki delay. Sistem dapat menyimpan timestamp prediksi, versi model, snapshot fitur, risk score, dan status label ketika kemudian dikonfirmasi. Pencatatan tersebut memungkinkan backtesting: prediksi historis dibandingkan dengan label yang muncul setelahnya tanpa mengubah informasi yang tersedia pada waktu prediksi. Smart

contract risk registry dapat menyimpan bukti ringkas, sedangkan data detail tetap disimpan off-chain untuk efisiensi.

Retraining tidak harus dilakukan pada interval tetap apabila distribusi stabil. Pemicu dapat berbasis jumlah label baru, penurunan recall, peningkatan false positive, atau drift fitur yang melewati batas. Dataset baru harus menjalani pipeline kualitas yang sama dan dibandingkan dengan versi sebelumnya. Versioning model dan dataset penting agar perubahan risk score dapat ditelusuri serta rollback dapat dilakukan ketika versi baru menunjukkan masalah.

Dalam ruang lingkup tesis ini, deployment dilakukan pada lingkungan demo dan testnet sehingga evaluasi drift jangka panjang belum dilakukan. Namun, rancangan FastAPI, model_version, timestamp, dan risk registry menyediakan dasar untuk monitoring. Keterbatasan ini perlu dinyatakan agar hasil F1 91,02% dipahami sebagai performa held-out pada dataset penelitian, bukan jaminan performa permanen pada seluruh transaksi Ethereum di masa mendatang.

2.1.19. Sampling, Bias Populasi, dan Representativitas Dataset

Sampling menentukan populasi yang secara efektif dipelajari oleh model. Pada dataset sangat tidak seimbang, menggunakan seluruh kelas mayoritas dapat meningkatkan biaya komputasi dan membuat objective didominasi contoh non-phishing. Undersampling mengurangi masalah tersebut, tetapi juga mengubah prior kelas pada data latih. Model kemudian belajar pada distribusi yang berbeda dari kondisi operasional, sehingga probabilitas keluaran dan threshold perlu ditafsirkan dengan mempertimbangkan perubahan tersebut.

Rasio 1:5 dan 1:10 bukan hanya parameter teknis, melainkan asumsi tentang kompromi antara variasi kelas mayoritas dan kemampuan model memperhatikan kelas minoritas. Rasio yang terlalu kecil dapat membuang variasi wallet normal dan meningkatkan false positive. Rasio yang terlalu besar dapat menekan recall phishing. Pemilihan rasio perlu didasarkan pada validation set menggunakan metrik yang sesuai, bukan hanya pada accuracy yang sangat dipengaruhi kelas mayoritas [37].

Random undersampling mengasumsikan bahwa sampel mayoritas yang dipilih cukup mewakili variasi populasi. Asumsi ini dapat diperkuat dengan stratifikasi tambahan berdasarkan degree, periode aktivitas, jenis account, atau kuantil nilai transaksi. Dengan demikian, wallet normal beraktivitas rendah, exchange berdegree tinggi, smart contract, dan wallet aktif lainnya tetap terwakili. Tanpa kontrol tersebut, sampel mayoritas dapat secara kebetulan terlalu mudah dibedakan dari phishing.

Bias temporal juga perlu diperhatikan. Random split mencampurkan wallet dari periode berbeda ke training dan test sehingga mengukur generalisasi terhadap entitas yang belum dilihat pada distribusi waktu serupa. Temporal split melatih pada periode lama dan menguji pada periode lebih baru sehingga lebih mendekati skenario deployment. Nilainya biasanya lebih konservatif karena menghadapi drift, tetapi memberikan bukti tambahan tentang kemampuan model menghadapi pola serangan yang berkembang.

Bias sumber muncul ketika label phishing dan non-phishing dikumpulkan melalui mekanisme berbeda. Daftar phishing dapat berasal dari laporan komunitas atau investigasi, sedangkan wallet normal berasal dari populasi graph. Perbedaan proses tersebut dapat menciptakan shortcut, misalnya perbedaan umur wallet atau cakupan API, yang dipelajari model tanpa benar-benar memahami perilaku fraud. Audit fitur perlu menilai apakah importance tinggi berasal dari perilaku transaksi atau hanya dari mekanisme pengumpulan data.

Untuk menjaga validitas perbandingan, sampel wallet dan indeks split harus dibekukan sebelum membentuk empat skenario fitur. Jika setiap skenario memiliki subset wallet berbeda karena missing enrichment, selisih F1 dapat berasal dari perubahan tingkat kesulitan populasi. Left join dengan EPTransNet sebagai tabel utama, zero-filling terkontrol, dan indikator ketersediaan sumber memungkinkan seluruh skenario mempertahankan wallet yang sama.

Keterwakilan dataset tidak berarti dataset harus identik dengan seluruh jaringan Ethereum, tetapi batas populasi perlu dinyatakan. Hasil penelitian berlaku pada wallet dan periode yang tercakup oleh EPTransNet serta sumber pengayaan yang digunakan. Klaim terhadap jaringan yang lebih luas harus mempertimbangkan perbedaan waktu, aplikasi, jenis token, dan perubahan ekosistem. Pernyataan ruang lingkup seperti ini meningkatkan ketepatan ilmiah tanpa mengurangi kontribusi metode.

2.1.20. Arsitektur Sistem Machine Learning dan Risk Registry

Sistem Machine Learning end-to-end terdiri atas lebih banyak komponen daripada file model. Alur dimulai dari permintaan wallet, validasi alamat, pengambilan atau pembentukan fitur, penerapan preprocessing yang sama dengan training, inferensi model, konversi skor, penyimpanan hasil, dan penyajian respons. Setiap batas komponen perlu memiliki kontrak data yang jelas agar perubahan satu layanan tidak secara diam-diam mengubah distribusi input model.

Feature schema merupakan kontrak yang mencakup nama, urutan, tipe, satuan, nilai default, dan versi fitur. Model yang dilatih dengan 26 fitur tidak boleh menerima urutan kolom berbeda walaupun jumlahnya sama. Validasi schema sebelum inferensi mencegah silent failure ketika API mengubah nama atau format nilai. Hash schema dapat disimpan bersama metadata model untuk memastikan kompatibilitas pada saat deployment.

Model artifact perlu disertai preprocessing artifact, daftar fitur, threshold, versi library, tanggal training, metrik validasi, dan identitas dataset. Praktik ini membentuk model package yang dapat direproduksi. Hanya menyimpan file XGBoost tanpa StandardScaler atau aturan imputasi dapat menghasilkan perbedaan antara eksperimen dan produksi. Walaupun tree-based model tidak membutuhkan scaling, urutan fitur dan perlakuan missing tetap merupakan bagian dari artifact.

FastAPI berfungsi sebagai lapisan layanan yang menerima wallet_address dan model_version, menjalankan validasi, memanggil pipeline inferensi, lalu mengembalikan label, probability, risk score, risk level, dan metadata. Error perlu

dibedakan antara alamat tidak valid, sumber data tidak tersedia, fitur tidak lengkap, model tidak ditemukan, dan kegagalan blockchain. Pemisahan tersebut membantu observability serta mencegah kegagalan teknis ditampilkan sebagai prediksi non-phishing.

Risk registry menyimpan wallet_address, risk_score, risk_status, timestamp, dan transaction hash sebagai bukti pencatatan. Blockchain memberikan sifat append-only dan keterlacakan, tetapi tidak menjamin kebenaran prediksi model. Karena itu, registry harus mencantumkan versi model atau referensi metadata agar auditor mengetahui konfigurasi yang menghasilkan skor. Pembaruan skor dapat disimpan sebagai record baru sehingga riwayat perubahan tetap tersedia.

Penyimpanan seluruh fitur dan penjelasan ke blockchain tidak efisien karena gas serta isu privasi dan ukuran data. Desain hybrid lebih sesuai: ringkasan serta hash disimpan on-chain, sedangkan fitur, log inferensi, dan penjelasan disimpan off-chain. Hash memungkinkan pemeriksaan integritas tanpa mengungkap semua data. Pada testnet atau localhost, pengukuran latency dan gas menunjukkan feasibility teknis, tetapi biaya mainnet tetap bergantung kondisi jaringan.

Keamanan sistem juga mencakup rate limiting, validasi input, perlindungan API key, timeout sumber eksternal, dan pencatatan audit. Wallet address merupakan data publik, tetapi pola kueri pengguna dan hasil analisis dapat bersifat sensitif dalam konteks operasional. Endpoint tidak boleh menerima path atau parameter yang memungkinkan injection. Smart contract perlu membatasi fungsi tulis pada account yang berwenang agar registry tidak dipenuhi skor palsu.

2.1.21. Strategi Lanjutan Penanganan Ketidakseimbangan Kelas

Ketidakseimbangan kelas dapat ditangani pada tingkat data, algoritma, dan keputusan. Tingkat data mengubah distribusi contoh melalui undersampling atau oversampling. Tingkat algoritma mengubah loss agar kesalahan pada kelas minoritas memperoleh bobot lebih besar. Tingkat keputusan mempertahankan model tetapi mengubah threshold. Pemisahan ini membantu eksperimen karena dua teknik pada tingkat berbeda dapat dikombinasikan tanpa dianggap sebagai prosedur yang sama.

SMOTE membuat sampel minoritas sintetis melalui interpolasi antarcontoh [26]. Pada fitur tabular kontinu, teknik ini dapat memperluas decision region kelas phishing. Namun, interpolasi degree, jumlah counterpart, dan indikator biner dapat menghasilkan kombinasi yang tidak merepresentasikan wallet nyata. Untuk data graph, membuat node sintetis juga membutuhkan definisi edge. Karena itu, random undersampling dan cost-sensitive learning lebih mudah dipertanggungjawabkan pada struktur penelitian ini.

Class weighting mempertahankan seluruh sampel tetapi meningkatkan kontribusi loss kelas phishing. Logistic Regression dan Random Forest menggunakan class_weight, sedangkan XGBoost menggunakan scale_pos_weight. Nilai rasio kelas merupakan titik awal, bukan aturan absolut. Bobot yang terlalu besar dapat meningkatkan recall namun menghasilkan banyak false positive, sehingga pemilihannya harus divalidasi berdasarkan tujuan sistem.

Balanced mini-batch pada GNN dapat memastikan node phishing muncul cukup sering selama optimasi. Akan tetapi, sampling node harus mempertahankan konteks tetangga yang diperlukan message passing. Neighbor sampling yang terlalu agresif dapat kehilangan edge penting, sedangkan penggunaan seluruh neighborhood dapat mahal pada hub besar. Konfigurasi fan-out perlu dilaporkan bersama jumlah layer dan ukuran batch.

Metrik macro average memberi bobot sama pada setiap kelas, sedangkan weighted average mengikuti proporsi kelas. Untuk fokus pada deteksi phishing, precision, recall, dan F1 kelas positif lebih mudah diinterpretasikan. Accuracy tetap dilaporkan sebagai informasi umum, tetapi tidak dijadikan dasar tunggal. PR-AUC membantu membandingkan ranking model ketika prevalensi positif rendah.

Evaluasi pada data uji dengan rasio 1:10 mengukur performa pada skenario eksperimen, bukan prevalensi seluruh jaringan. Ketika diterapkan pada populasi dengan phishing jauh lebih jarang, precision dapat menurun walaupun sensitivity dan specificity tetap. Oleh sebab itu, deployment memerlukan evaluasi ulang pada distribusi yang lebih mendekati kondisi operasional serta kalibrasi threshold berdasarkan biaya kesalahan.

2.1.22. Etika, Keterbatasan, dan Penggunaan Risk Score

Klasifikasi wallet berisiko memiliki konsekuensi terhadap pengguna sehingga hasil model tidak seharusnya diperlakukan sebagai vonis tunggal. False positive dapat memberi stigma pada wallet sah, exchange, donation address, atau smart contract yang memiliki pola mirip pengumpul dana. Risk score lebih tepat digunakan sebagai sinyal pendukung investigasi dan peringatan, terutama ketika bukti label serta penjelasan prediksi masih terbatas.

Transparansi memerlukan informasi tentang sumber data, periode, definisi label, metrik, threshold, dan versi model. Model card dapat merangkum tujuan, penggunaan yang dimaksud, penggunaan yang tidak dianjurkan, performa lintas kelompok data, dan keterbatasan [43]. Pada endpoint, metadata ringkas dapat dikembalikan agar pengguna mengetahui bahwa skor berasal dari model statistik dan bukan daftar hitam resmi.

Alamat blockchain bersifat publik, tetapi penggabungan alamat dengan hasil analisis, pola kueri, atau identitas off-chain dapat menimbulkan risiko privasi. Sistem perlu menyimpan hanya data yang diperlukan, membatasi akses log, dan menghindari pengungkapan informasi pengguna yang melakukan pemeriksaan. Penyimpanan on-chain perlu lebih hati-hati karena record sulit dihapus setelah dicatat.

Mekanisme koreksi diperlukan ketika label baru atau bukti investigasi mengubah status wallet. Registry append-only dapat menyimpan pembaruan sebagai record baru dengan timestamp dan model version, tanpa menghapus riwayat. Antarmuka perlu menampilkan status terbaru sekaligus menyediakan jejak perubahan. Prosedur koreksi mencegah skor lama terus digunakan tanpa konteks.

Keterbatasan dataset, sampling, label noise, sparse graph, dan lingkungan testnet harus dinyatakan secara eksplisit. Pernyataan tersebut tidak melemahkan

penelitian, tetapi membatasi interpretasi pada bukti yang benar-benar diuji. Klaim utama adalah efektivitas transformasi graph-to-tabular dan pengayaan Etherscan pada protokol penelitian, bukan kemampuan mendeteksi seluruh jenis fraud Ethereum tanpa kesalahan.

Penggunaan yang bertanggung jawab menempatkan human review pada keputusan berisiko tinggi. Model dapat memprioritaskan wallet untuk pemeriksaan, menampilkan alasan, dan memberi peringatan proporsional. Pemblokiran aset atau pelaporan publik memerlukan verifikasi tambahan dari sumber independen. Dengan desain ini, Machine Learning mempercepat penyaringan tanpa menggantikan penilaian keamanan dan proses banding.

2.2. Penelitian Terkait

Bagian ini mengulas penelitian-penelitian yang relevan dengan topik deteksi phishing pada jaringan Ethereum menggunakan pendekatan Machine Learning dan Graph Neural Network. Kajian ini dilakukan untuk mengidentifikasi state-of-the-art, kekurangan pendekatan yang ada, serta memposisikan kontribusi penelitian ini terhadap bidang ilmu yang ada.

Chen et al. [4] pada tahun 2020 menggunakan ledger Ethereum hingga blok 7.000.000 dan 1.683 alamat phishing dari Etherscan. Graph transaksi diubah menjadi cascade features, kemudian diklasifikasikan menggunakan LightGBM dalam kerangka Dual-sampling Ensemble untuk menangani ketidakseimbangan kelas. Dataset dikumpulkan sendiri oleh penulis dan tidak diberi nama khusus. Penelitian ini belum membandingkan model tabular dengan GNN dan belum menyediakan pencatatan risk score pada blockchain.

Wu et al. [5] membangun 50 subnetwork Ethereum dari 1.259 alamat phishing dan node tidak berlabel yang dipilih secara acak. Penelitian ini mengusulkan trans2vec untuk membentuk node embedding dengan mempertimbangkan nilai dan waktu transaksi, kemudian menggunakan one-class SVM untuk klasifikasi. Dataset merupakan hasil pengumpulan penulis dan tidak disebut sebagai dataset publik bernama. Fokus penelitian adalah network embedding, bukan perbandingan model tabular dan GNN maupun pencatatan hasil on-chain.

Chen et al. [14] memodelkan akun sebagai node dan transaksi sebagai edge, kemudian menggunakan GCN dan autoencoder untuk menggabungkan struktur spasial subgraph dengan fitur node. Paper tersebut menyatakan belum tersedia dataset ready-

to-use sehingga data graph dan label phishing dikumpulkan oleh penulis. Pendekatan ini menjadi acuan GNN, tetapi belum membandingkannya dengan XGBoost pada kondisi eksperimen identik dan belum memiliki risk registry.

Yuan et al. [15] membangun transaction subgraph untuk setiap target wallet dari data hasil web crawling. Model imgraph2vec digunakan untuk mempelajari representasi subgraph dengan mempertimbangkan block height, timestamp, dan nilai transaksi, sedangkan XGBoost digunakan sebagai classifier. Akses dataset tidak dinyatakan pada paper. Penelitian ini relevan sebagai pembanding XGBoost berbasis graph embedding, tetapi tidak membandingkan jalur tabular dan GNN pada dataset yang sama serta tidak mencatat hasil pada smart contract.

Kabla et al. [16] membangun dataset publik Eth-PSD dari Etherscan dan XBlock yang berisi 84.664 data awal, terdiri atas 79.216 alamat normal dan 5.448 alamat phishing. Setelah preprocessing dan feature selection, delapan classifier dibandingkan; K-Nearest Neighbors memberikan hasil terbaik dengan AUC sekitar 98% pada 10-fold cross-validation. Data direpresentasikan secara tabular sehingga relasi graph dan pencatatan risk score on-chain tidak dibahas.

Li et al. [17] mengusulkan Temporal Transaction Aggregation Graph Network (TTAGN) pada graph yang dikumpulkan melalui Etherscan. TTAGN menggunakan LSTM untuk merepresentasikan urutan transaksi pada edge, attention untuk agregasi edge-to-node, GCN/autoencoder untuk fitur struktural, dan LightGBM sebagai classifier. Model memperoleh F1-score 81,6% dan AUC 92,8% pada skenario D3. Penelitian ini tidak mengevaluasi enrichment fitur multi-sumber atau risk registry.

Wang et al. [18] mengembangkan PDTGA berbasis Temporal Graph Attention untuk mendeteksi phishing pada dataset XBlock yang diperkaya melalui Etherscan. Dataset asal mencakup 1.660 target phishing dan 1.700 target non-phishing, kemudian dibentuk menjadi subgraph eksperimen D1, D2, dan D3. Model memanfaatkan 14 node features dan 3 edge features serta mencapai AUC 94,78% dan recall 88,76%. Penelitian tersebut tidak membandingkan pipeline XGBoost dan GraphSAGE pada fitur hybrid serta tidak mencatat hasil pada smart contract.

Kanezashi et al. [39] menggunakan XBlock Ethereum Phishing Transaction Network (EPTransNet/MulDiGraph), yaitu dataset yang juga menjadi sumber utama

penelitian ini, dengan 2.973.489 node, 13.551.303 edge, dan 1.165 node phishing. Penelitian tersebut membentuk subgraph seimbang dari seluruh node phishing dan node normal yang dipilih secara acak, kemudian membandingkan model homogeneous GNN (GCN, GAT, dan GraphSAGE) serta heterogeneous GNN (RGCN, HAN, dan HGT). RGCN menghasilkan performa terbaik secara keseluruhan. Meskipun menggunakan dataset yang sama dan mengevaluasi GraphSAGE, penelitian tersebut berfokus pada komparasi antar-GNN dan pengayaan tipe node dari Etherscan, tanpa membandingkan model tabular seperti Logistic Regression, Random Forest, dan XGBoost, serta tanpa mengintegrasikan hasil prediksi sebagai risk score pada smart contract.

Tabel II.3. Ringkasan Penelitian Terkait dan Posisi Penelitian Ini

Peneliti	Dataset/Sumber	Akses	Representasi	Algoritma	Hasil/Fokus	Keterbatasan terhadap Penelitian Ini
Chen et al. (2020) [4]	Ledger Ethereum + label Etherscan; dikumpulkan penulis	Dikumpulkan penulis	Cascade feature graph	LightGBM + dual-sampling ensemble	Keduanya: feature engineering, sampling, dan ensemble	Belum mengisolasi kontribusi perubahan data dan model pada split identik
Wu et al. (2020/2022) [5]	Ethereum client + dua sumber label; dikumpulkan penulis	Dikumpulkan penulis	Transaction-graph embedding	trans2vec + one-class SVM	Keduanya: embedding berbobot nilai/waktu dan classifier	Belum menguji enrichment multi-sumber serta model tabular modern dan GNN secara setara
Chen et al. (2020/2021) [14]	Graph transaksi Ethereum + label phishing; dikumpulkan penulis	Dikumpulkan penulis	Subgraph + node features	GCN + autoencoder (E-GCN)	Model-centric: arsitektur GNN dan autoencoder	Berfokus pada satu keluarga GNN; perubahan representasi data tidak diuji terpisah
Yuan et al. (2020) [15]	Transaksi hasil web crawling; akses dataset tidak dinyatakan	Publik/penerbit	Transaction-subgraph embedding	imgraph2vec + XGBoost	Keduanya: embedding subgraph dan classifier tabular	Tidak membandingkan pipeline tabular dan GNN pada populasi serta split yang sama
Kabla et al. (2022) [16]	Eth-PSD dari Etherscan + XBlock; dataset publik	Publik/penerbit	Fitur transaksi tabular	KNN, J48, Fast DT, C4.5, NBTree, PART, JRip, OneR	Model-centric: komparasi delapan classifier dan feature selection	Tidak memanfaatkan struktur graph atau enrichment fitur berbasis wallet
Li et al. (2022) [17]	Crawling Etherscan; akses dataset tidak dinyatakan	Publik/penerbit	Temporal MultiDiGraph	TTAGN + LightGBM	Keduanya: temporal edge aggregation dan classifier	Tidak menguji enrichment multi-sumber atau pembanding graph-to-tabular terkontrol
Wang et al. (2023) [18]	XBlock + Etherscan; dataset asal publik	Publik/penerbit	Temporal transaction graph	PDTGA/ TGAT + MLP	Model-centric: temporal graph attention	Kontribusi sumber fitur dan algoritma tidak diisolasi secara faktorial

Peneliti	Dataset/Sumber	Akses	Representasi	Algoritma	Hasil/Fokus	Keterbatasan (terhadap) (Penelitian) (Ini
Kanezashi et al. (2022) [39]	EPTransNet/ MultiDiGraph + tipe node Etherscan; publik	Publik/penerbit	Homogeneous dan heterogeneous graph	GCN, GAT, GraphSAGE, RGCN, HAN, HGT	Model-centric: komparasi enam GNN	Tidak menguji graph-to-tabular, enrichment tabular, dan XGBoost pada split sama
Yuan et al. (2020) [28]	Transaksi Ethereum + label dari dua situs otoritatif	Dikumpulkan penulis	Graph + node2vec embedding	node2vec + one-class SVM	Keduanya: network embedding dan classifier	Fokus pada satu representasi embedding; tanpa ablasi sumber fitur dan keluarga model
Xia et al. (2022) [38]	Transaksi on-chain + label situs otoritatif	Publik/penerbit	Attributed ego-graph embedding	Node relabeling + skip-gram + classifier	Data-centric: atribut nilai, jumlah, dan arah transaksi	Tidak membandingkan agregasi graph-to-tabular dengan GNN dan model tabular pada split identik
Xiong et al. (2024) [49]	Etherscan transaction network; dikumpulkan penulis	Dikumpulkan penulis	Directed multi-transaction graph	TransWalk + one-class SVM	Data-centric: biased walk multi-skala berbasis nilai dan waktu	Tidak menguji enrichment eksternal dan komparasi model tabular–GNN secara terkontrol
Li et al. (2022) [50]	Lima dataset transaksi Ethereum; dibangun penulis	Publik/penerbit	Lightweight transaction subgraph	PDGNN berbasis Chebyshev-GCN	Keduanya: network lightweighting dan end-to-end GNN	Berfokus pada graph classification; tidak mengisolasi dampak fitur tabular dan model
Li et al. (2023) [51]	Ethereum transaction graph bertahap; dibangun penulis	Dikumpulkan penulis	Incremental graph splits	SIEGE self-supervised GNN	Keduanya: spatial/temporal pretext tasks dan incremental learning	Tidak membandingkan feature enrichment dengan model tabular pada data uji yang sama
Li et al. (2023) [52]	Real-world Ethereum phishing datasets	Publik/penerbit	Transaction subgraphs + statistik	TGC graph contrastive learning	Keduanya: node/context contrast dan statistik transaksi	Tidak memisahkan kontribusi sumber data dan pilihan classifier melalui desain faktorial
Wang et al. (2025) [53]	Ethereum transaction subgraphs; data dikumpulkan penulis	Dikumpulkan penulis	Directed, temporal, dan multiple TSGN	TSGN + graph representation learning	Data-centric: pemetaan subgraph dan atribut transaksi	Tidak membandingkan graph-to-tabular, enrichment API, dan beberapa keluarga model pada split sama
Huang et al. (2024) [54]	Label + transaksi dari situs otoritatif	Dikumpulkan penulis	Augmented ego-graph	PEAE-GNN	Keduanya: augmentasi fitur struktur/transaksi /interaksi dan GNN	Peningkatan data dan arsitektur diterapkan bersamaan sehingga kontribusi relatifnya tidak terisolasi
Chen et al. (2024) [55]	Dua dataset transaksi Ethereum nyata	Publik/penerbit	Fitur node + temporal + structural	DA-HGNN: Conv1D, GRU-MHA, GAE	Keduanya: sliding-window augmentation dan hybrid GNN	Tidak menguji enrichment lintas sumber dan model klasik pada wallet/split identik
Zhang et al. (2024) [56]	Real-world Ethereum phishing datasets	Publik/penerbit	Temporally evolving transaction graph	GrabPhisher evolving GNN	Model-centric: continuous-time evolving topology	Fokus pada evolusi graph; bukan isolasi kontribusi representasi tabular dan pemilihan model
Liu et al. (2024) [57]	Dua dataset transaksi Ethereum nyata	Publik/penerbit	Basic, transaction, dan interaction features	FAAN-GBM + autoencoder /attention	Keduanya: feature enhancement dan gradient boosting	Tidak membandingkan jalur graph-to-tabular dan GNN pada protokol data yang identik
Fu et al. (2024) [58]	Ethereum phishing graph; tersedia atas permintaan	Dikumpulkan penulis	Temporal transaction graph	CT-GCN+: LSTM, attention, SMOTE, GCN	Keduanya: temporal features, graph balancing, dan GCN	Intervensi data dan model digabung; kontribusi enrichment multi-sumber tidak diuji
Lin et al. (2023) [59]	Ethereum transaction network + statistical features	Dikumpulkan penulis	Temporal/ heterogeneous network embedding	Account representation learning + classifier	Data-centric: topology dan fitur statistik	Tidak menguji graph-to-tabular, enrichment API, serta model tabular dan GNN pada split sama
Tang et al. (2024) [60]	Ethereum phishing transaction sequences	Dikumpulkan penulis	Explicit features + transaction sequence	BiLSTM4DPS + multi-head attention	Keduanya: feature sequence dan attention-BiLSTM	Tidak melakukan perbandingan faktorial sumber fitur dan keluarga model

Peneliti	Dataset/Sumber	Akses	Representasi	Algoritma	Hasil/Fokus	Keterbatasan (terhadap) (Penelitian) (Ini
Hou et al. (2024) [61]	Ethereum transaction graph	Dikumpulkan penulis	Local structural features	Curriculum learning phishing detector	Model-centric: pengurutan kesulitan sampel	Tidak menilai enrichment data eksternal atau graph-to-tabular terhadap model klasik
Xu et al. (2023) [62]	Ethereum phishing transaction graph	Dikumpulkan penulis	Spatial + temporal representation	STFN spatio-temporal fusion	Keduanya: spatial-temporal feature fusion dan deep model	Kontribusi fitur dan model tidak dipisahkan pada eksperimen faktorial terkontrol
Yang et al. (2023) [63]	Ethereum transactions + labeled phishing accounts	Dikumpulkan penulis	Explainable transaction patterns	Translets	Keduanya: interpretable representation dan detector	Fokus explainability; tidak membandingkan enrichment multi-sumber dan keluarga model pada split sama
Zhang et al. (2023) [64]	Ethereum transaction network	Dikumpulkan penulis	Transaction features + graph structure	BM-GCN	Keduanya: behavior mining dan GCN	Tidak mengisolasi dampak transformasi data dari dampak arsitektur GCN
Wen et al. (2023) [65]	EPS-Dataset yang dibangun penulis	Publik/penerbit	Manual features + temporal transaction sequence	LBPS: BP-NN + LSTM-FCN	Keduanya: hybrid feature fusion dan deep neural network	Tidak menguji enrichment sumber publik dan komparasi tabular–GNN pada data sama
Yu et al. (2023) [66]	Streaming Ethereum transactions; dibangun penulis	Dikumpulkan penulis	Dynamic edge features	Broadcast/ storage streaming GNN	Keduanya: incremental representation dan streaming model	Berfokus pada deteksi streaming; belum mengukur kontribusi sumber fitur dan pilihan model secara terpisah
Penelitian Ini	EPTransNet + Etherscan API + BigQuery	Publik/API	Graph-to-tabular + graph	LR, RF, XGBoost, GraphSAGE	Evaluasi 4 skenario fitur × 4 model	Memisahkan kontribusi data-centric dan model-centric; RiskRegistry sebagai implementasi

Tabel II. 3 Ringkasan Penelitian Terkait dan Posisi Penelitian Ini

Sintesis 28 penelitian menunjukkan empat arus utama. Studi awal menggunakan fitur transaksi, sampling, network embedding, dan classifier klasik [4], [5], [15], [16], [28], [38]. Studi berikutnya memperdalam representasi melalui GNN, subgraph, dan graph contrastive learning [14], [39], [49]–[54]. Literatur terbaru menambahkan data augmentation, temporal/evolving graph, feature enhancement, attention, curriculum learning, explainability, dan streaming detection [55]–[66]. Walaupun cakupan metode semakin luas, sebagian besar studi mengubah representasi data dan arsitektur model secara bersamaan atau mengevaluasi satu keluarga model saja. Karena itu, research gap yang dipertahankan adalah belum jelasnya kontribusi relatif perbaikan data dan pemilihan model ketika populasi wallet, split, preprocessing, dan metrik dikendalikan secara identik.

2.2.1. Sintesis Literatur dan Posisi Penelitian

Literatur deteksi phishing Ethereum memperlihatkan tiga arah utama. Arah pertama menggunakan fitur transaksi tabular yang dibentuk dari degree, nilai, waktu, dan counterpart. Arah kedua menggunakan embedding atau GNN untuk mempelajari struktur relasi. Arah ketiga menggabungkan beberapa modalitas atau sumber data.

Ketiganya menunjukkan bahwa representasi wallet merupakan inti masalah; perbedaan performa tidak hanya berasal dari algoritma, tetapi juga dari cara transaksi mentah diubah menjadi unit analisis.

Studi berbasis tabular memiliki keunggulan interpretabilitas dan biaya komputasi yang lebih rendah, tetapi sangat bergantung pada kualitas feature engineering. Studi berbasis graph dapat menangkap konteks relasional yang sulit diringkas, tetapi membutuhkan topologi yang cukup padat dan desain split yang mengendalikan leakage. Studi fusion berpotensi menggabungkan kedua kekuatan tersebut, namun harus membuktikan bahwa setiap cabang membawa informasi non-redundan melalui eksperimen ablasi.

Posisi penelitian ini berada pada irisan pendekatan data-centric dan model-centric. MultiDiGraph EPTransNet tidak hanya digunakan sebagai input GNN, tetapi ditransformasikan menjadi representasi tabular per wallet melalui agregasi 1-hop. Representasi tersebut diperkaya dengan fitur Etherscan dan BigQuery sebagai intervensi data-centric. Selanjutnya, Logistic Regression, Random Forest, XGBoost, dan GraphSAGE dibandingkan sebagai intervensi model-centric pada populasi serta split yang sama.

Kebaruan yang diuji adalah kerangka evaluasi terkontrol yang memisahkan dan menggabungkan kontribusi data-centric dan model-centric. Empat skenario fitur mengukur dampak transformasi graph-to-tabular dan enrichment eksternal, sedangkan empat keluarga model mengukur dampak pemilihan algoritma. Analisis feature importance, confusion matrix, false negative rate, dan late-fusion ablation digunakan untuk memastikan bahwa peningkatan tidak hanya disimpulkan dari satu konfigurasi terbaik.

Model terpilih selanjutnya dihubungkan dengan FastAPI dan risk registry berbasis smart contract. Integrasi tersebut tidak diklaim meningkatkan akurasi atau menjadi kebaruan ilmiah utama, tetapi berfungsi sebagai kontribusi implementatif untuk menunjukkan alur penggunaan model dan pencatatan hasil klasifikasi yang dapat ditelusuri.

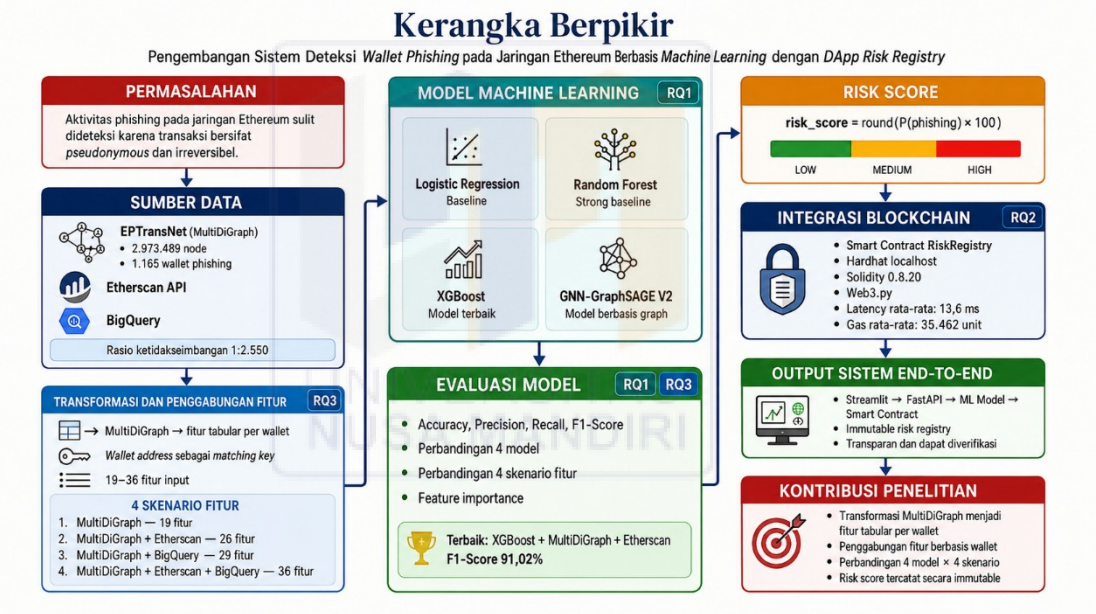
2.3. Kerangka Berpikir

Berdasarkan landasan teori yang telah diuraikan dan research gap yang telah diidentifikasi, penelitian ini dibangun di atas kerangka berpikir yang menghubungkan permasalahan nyata dengan solusi teknis yang dapat diukur. Gambar II.1 menunjukkan alur kerangka berpikir penelitian ini secara visual.

Titik awal kerangka berpikir adalah permasalahan nyata: meningkatnya aktivitas phishing pada jaringan Ethereum yang sulit dideteksi karena sifat pseudonymous blockchain dan kompleksitas pola transaksi. Solusi yang diusulkan adalah sistem end-to-end yang menggabungkan tiga komponen utama: (1) dataset

hybrid dari EPTransNet, Etherscan API, dan BigQuery yang memperkaya representasi setiap wallet; (2) perbandingan sistematis empat algoritma Machine Learning pada empat skenario kombinasi fitur; serta (3) pencatatan hasil prediksi sebagai risk score ke dalam smart contract sebagai mekanisme audit trail.

Kerangka ini secara eksplisit menjawab ketiga research question: RQ3 dijawab melalui eksplorasi empat skenario fitur dan analisis feature importance; RQ1 dijawab melalui perbandingan performa keempat model dengan metrik F1-Score sebagai ukuran utama; dan RQ2 dijawab melalui pengukuran overhead latency dan gas cost dari 20 transaksi ke smart contract Hardhat. Seluruh hipotesis yang diajukan pada BAB I diuji secara empiris melalui eksperimen yang dirancang dalam BAB III dan hasilnya dianalisis pada BAB IV.



Gambar II. 1

Gambar II.1. Kerangka Berpikir Penelitian