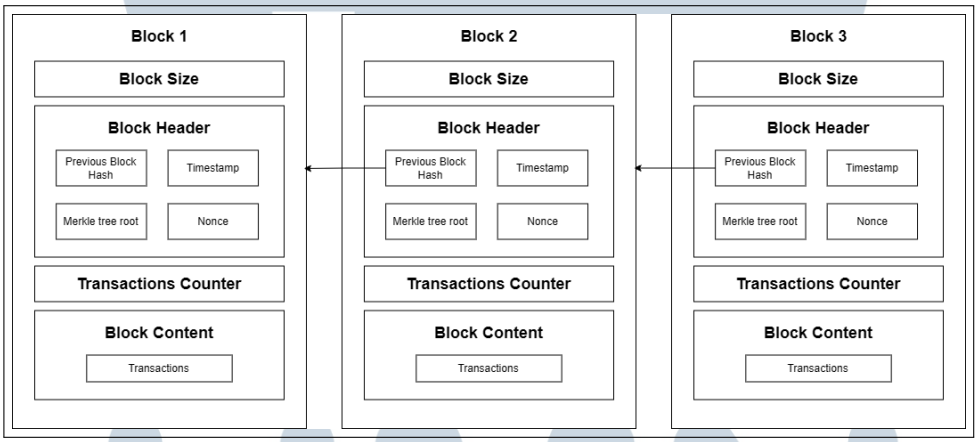
BAB 2
LANDASAN TEORI

2.1 Blockchain dan Ethereum

Blockchain adalah teknologi penyimpanan data terdistribusi yang memungkinkan pencatatan transaksi tersimpan secara aman, transparan, dan tidak dapat diubah [6]. Data transaksi tersebut disusun dalam bentuk blok yang saling berhubungan dengan blok sebelumnya menggunakan hash kriptografi, sehingga perubahan pada suatu blok akan merusak validitas seluruh rantai. Bentuk blok dapat dilihat pada Gambar 2.1.



Gambar 2.1. Bentuk dan isi block pada sebuah blockchain

Salah satu sifat terpenting dalam blockchain adalah desentralisasi, di mana tidak ada pihak tunggal yang mengontrol sistem. Validasi transaksi dilakukan melalui mekanisme konsensus yang disepakati oleh jaringan. Selain itu, sifat *immutability* yang memastikan bahwa data yang telah tercatat sangat sulit untuk dimanipulasi, sehingga meningkatkan keamanan dan kepercayaan antar pengguna [7].

Ethereum, salah satu platform *blockchain* yang memungkinkan pengembangan aplikasi terdesentralisasi melalui penggunaan *smart contract*. *Smart contract* adalah program yang berjalan otomatis di dalam *blockchain* ketika kondisi tertentu terpenuhi, sehingga menghilangkan kebutuhan pihak ketiga dalam proses transaksi.

2.2 Proof of Stake

Jaringan Ethereum saat ini bertumpu pada mekanisme konsensus *Proof of Stake* (PoS). Berbeda dengan *Proof of Work* (PoW) pada jaringan Bitcoin yang mengandalkan daya komputasi, PoS memilih validator berdasarkan jumlah aset yang didepositkan (*staking*). Untuk menjamin kejujuran pengguna jaringan, sistem ini menerapkan *slashing*, yaitu penalti berupa pemotongan aset bagi validator yang melakukan kecurangan atau ketidakadlian [8]. Mekanisme konsensus PoS meningkatkan *throughput* atau jumlah transaksi yang dapat diproses oleh jaringan. Dengan frekuensi blok yang stabil setiap 12 detik [9], aliran data transaksi menjadi sangat padat.

2.3 Malicious Transactions

Malicious transactions merupakan aktivitas transaksi pada jaringan *blockchain* yang dilakukan dengan tujuan merugikan pihak lain atau melanggar aturan yang berlaku. Meskipun *blockchain* dikenal aman dan transparan, sifat pseudonim pada alamat pengguna memungkinkan pelaku untuk menyembunyikan identitas asli mereka [10]. Hal ini menyebabkan ekosistem *blockchain* tetap rentan terhadap penyalahgunaan, terutama pada jaringan dengan volume transaksi tinggi seperti Ethereum yang mencakup aktivitas *phishing*, *scam*, *rug pull*, hingga pencucian uang (*money laundering*).

Deteksi terhadap transaksi berbahaya ini secara teknis dilakukan melalui tiga pendekatan utama [11]. Pertama, analisis dimensi waktu dan volume digunakan untuk membedakan perilaku manusia dengan bot di mana pelaku sering menggunakan skrip yang berjalan secara otomatis untuk memindahkan dana secara massal dalam waktu yang singkat. Kedua, analisis topologi jaringan digunakan untuk mengidentifikasi teknik *address hopping* atau *peeling chain*, yaitu pola pengiriman dana berantai melalui banyak alamat perantara guna memutus jejak audit. Ketiga, analisis interaksi *smart contract* dilakukan dengan memeriksa profil kontrak, seperti status verifikasi kode sumber serta penggunaan fungsi berbahaya seperti *selfdestruct* yang sering menjadi indikasi eksploitasi.

Kompleksitas dari pola-pola tersebut menunjukkan bahwa deteksi manual atau sistem berbasis aturan statis (*rule-based*) tidak lagi memadai [12]. Pelaku *malicious transactions* terus mengubah strategi mereka untuk menghindari deteksi, sehingga diperlukan pendekatan yang mampu memproses berbagai variabel

kompleks. Oleh karena itu, penggunaan algoritma *machine learning* menjadi krusial untuk mengenali korelasi non linear yang menandakan aktivitas berbahaya.

2.4 Random Forest

Salah satu algoritma yang memiliki ketahanan tinggi dalam menangani klasifikasi data yang kompleks dan berdimensi besar adalah *random forest*. *Random forest* merupakan algoritma berbasis *ensemble learning* yang digunakan untuk melakukan klasifikasi maupun regresi. Algoritma ini bekerja dengan membangun sekumpulan *decision trees* secara independen dan kemudian menggabungkan hasil prediksi dari masing-masing pohon untuk menghasilkan keputusan akhir melalui voting mayoritas [13].

Pada algoritma *random forest*, setiap *decision tree* dibangun menggunakan *bootstrap sampling*, yaitu pengambilan sampel data secara acak dengan pengembalian (*sampling with replacement*) dari dataset pelatihan. Selain itu, pada setiap node hanya sebagian fitur yang dipertimbangkan untuk proses pemisahan. Pada implementasi *Random Forest Classifier*, jumlah fitur yang dievaluasi pada setiap node umumnya sebesar $\sqrt{p}$, dimana p merupakan jumlah total fitur. Dari subset fitur tersebut, algoritma memilih fitur yang menghasilkan nilai *gini impurity* atau *entropy* terbaik sebagai dasar pemisahan node [14]. Pemilihan subset fitur secara acak ini menghasilkan struktur pohon yang berbeda-beda sehingga mengurangi korelasi antar pohon dan meningkatkan kemampuan generalisasi model. Prediksi akhir diperoleh melalui mekanisme *majority voting*, yaitu kelas yang memperoleh suara terbanyak dari seluruh *decision tree* ditetapkan sebagai hasil klasifikasi.

2.4.1 Decision Trees

Decision tree merupakan algoritma *supervised learning* yang menjadi dasar dalam pembentukan model *random forest*. Setiap *decision tree* dibangun dengan membagi data secara rekursif berdasarkan atribut yang memberikan kualitas pemisahan terbaik. Pada setiap node, algoritma mengevaluasi kandidat atribut menggunakan fungsi *criterion*, seperti *Gini Impurity* atau *Entropy*, kemudian memilih atribut dan *threshold* yang menghasilkan penurunan impuritas terbesar [15]. Proses ini diulang hingga mencapai kondisi penghentian, seperti *max depth*, *min samples split*, atau seluruh data pada node berasal dari kelas yang sama.

2.4.2 Gini Impurity

Gini impurity adalah salah satu metode yang digunakan untuk menentukan kualitas pemisahan (*split*) pada setiap node dalam *decision tree*. Nilai *gini impurity* mengukur tingkat ketidakmurnian suatu node, yaitu seberapa besar kemungkinan sebuah data dipilih secara acak kemudian salah diklasifikasikan apabila mengikuti distribusi kelas pada node tersebut. Semakin rendah nilai *gini impurity*, semakin murni node tersebut, yang berarti sebagian besar data pada node berasal dari kelas yang sama. Rumus 2.1 digunakan untuk menghitung nilai *gini impurity* pada setiap node. Dalam proses pembentukan *decision tree*, algoritma menghitung nilai *gini impurity* untuk setiap kemungkinan *split* berdasarkan atribut yang tersedia, kemudian memilih atribut yang menghasilkan penurunan nilai *gini impurity* terbesar, sehingga node yang terbentuk memiliki tingkat kemurnian yang lebih tinggi dari *parent* node tersebut [16].

$$Gini = 1 - \sum_{i=1}^{n} p_i^2 \tag{2.1}$$

2.4.3 Entropy

Entropy adalah metode lain yang digunakan untuk mengukur tingkat ketidakpastian atau ketidakteraturan distribusi kelas pada suatu node dalam *decision tree*. Semakin tinggi nilai *entropy*, semakin besar ketidakpastian distribusi kelas pada node tersebut. Rumus 2.2 digunakan untuk menghitung nilai *entropy* pada setiap node. Dalam proses pembentukan *decision tree*, algoritma menghitung nilai *entropy* untuk setiap kemungkinan pemisahan berdasarkan atribut yang tersedia, kemudian memilih atribut yang menghasilkan *information gain* terbesar [17]. *Information gain* seberapa besar penurunan nilai *entropy* yang diperoleh setelah data dipisahkan menggunakan suatu atribut. Semakin besar nilai information gain, semakin baik atribut tersebut dalam memisahkan data ke dalam kelas yang lebih murni.

$$Entropy = -\sum_{i=1}^{n} p_i \log_2(p_i) \tag{2.2}$$

2.5 Evaluasi Kinerja Model

Evaluasi kinerja model *random forest* menggunakan *confusion matrix* yang terdiri dari *True Positive*, *True Negative*, *False Positive*, dan *False Negative*. Pada penelitian ini, kelas positif (y = 1) didefinisikan sebagai transaksi-transaksi berbahaya dan kelas negatif (y = 0) sebagai transaksi-transaksi normal atau tidak ada anomali. Dengan definisi tersebut: *true positive* (TP) adalah transaksi berbahaya yang diprediksi berbahaya, *true negative* (TN) adalah transaksi normal yang diprediksi normal, *false positive* (FP) adalah transaksi normal yang diprediksi berbahaya, dan *false negative* (FN) adalah transaksi berbahaya yang diprediksi normal.

2.5.1 Accuracy

Persentase total prediksi yang benar, baik positif maupun negatif, dibandingkan dengan keseluruhan jumlah data yang diuji. Rumus 2.3 menunjukkan cara perhintungan *Accuracy*.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.3}$$

2.5.2 Precision

Rasio prediksi positif yang benar-benar tepat dibandingkan dengan seluruh hasil yang diprediksi sebagai positif oleh model. Rumus 2.4 menunjukkan cara perhitungan *Precision*.

$$Precision = \frac{TP}{TP + FP} \tag{2.4}$$

2.5.3 Recall

Rasio kemampuan model dalam menemukan kembali seluruh data yang benar-benar positif dari total keseluruhan data positif yang ada. Rumus 2.5 menunjukkan cara perhitungan *Recall*.

$$Recall = \frac{TP}{TP + FN} \tag{2.5}$$

2.5.4 F1 Score

Nilai rata-rata harmonis antara precision dan recall yang digunakan untuk memberikan gambaran performa model secara lebih seimbang, terutama pada data yang tidak seimbang. Rumus 2.6 menunjukkan cara perhitungan *F1 Score*.

$$F1Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{2.6}$$

2.6 SEMMA model

Metodologi SEMMA merupakan sebuah kerangka kerja sistematis dalam proses *data mining*. Alur metodologi SEMMA ditunjukkan pada Gambar 2.2 SEMMA merujuk pada lima tahapan utama, yaitu *Sample*, *Explore*, *Modify*, *Model*, dan *Assess* yang bersifat iteratif, di mana hasil dari satu tahap dapat digunakan untuk menyempurnakan tahapan sebelumnya untuk mencapai model yang optimal [18].

Secara teoretis, kelima tahapan dalam metodologi SEMMA dapat dijelaskan sebagai berikut:

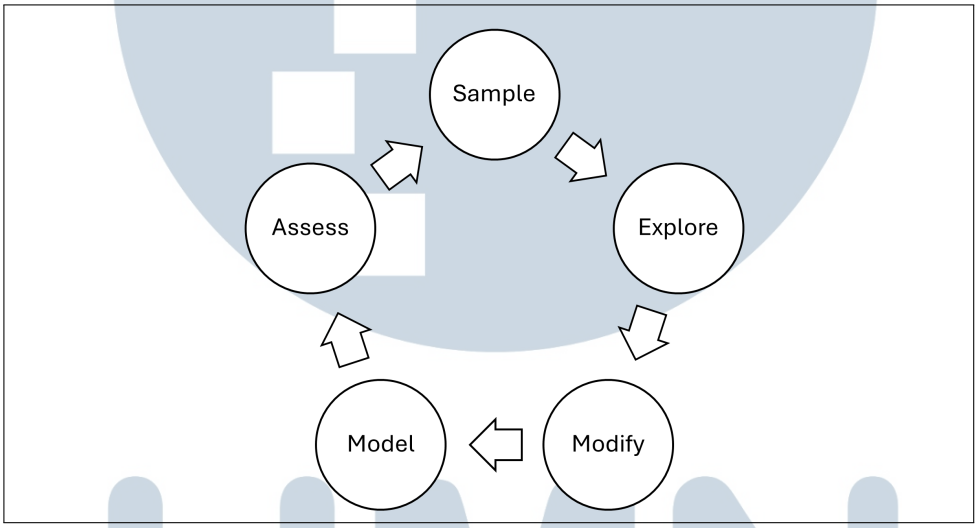
1. *Sample* (Sampel): Tahap ini melibatkan pemilihan dataset yang representatif dari populasi data yang luas. Tahap ini memastikan bahwa volume dan jenis data cukup signifikan untuk mencerminkan pola yang ingin dipelajari.

2. *Explore* (Eksplorasi): Tahap ini melibatkan pemeriksaan mendalam terhadap data untuk memahami tren, distribusi, serta hubungan antar variabel. Eksplorasi dilakukan melalui visualisasi dan analisis statistik deskriptif untuk mengidentifikasi anomali, *outliers*, ataupun *missing values*.

3. *Modify* (Modifikasi): Tahap ini mencakup manipulasi data agar data siap digunakan dalam pelatihan model. Tahap ini mencakup rekayasa fitur, pemilihan variabel yang relevan, serta transformasi data seperti normalisasi atau pengkodean ulang untuk meningkatkan kualitas input.

4. *Model* (Pemodelan): Tahap ini merupakan proses penerapan algoritma *machine learning* untuk menemukan pola-pola tersembunyi di dalam data. Berbagai optimasi juga dapat dilakukan agar model menghasilkan prediksi atau klasifikasi yang lebih akurat.

5. *Assess* (Penilaian): Tahap akhir ini berfungsi untuk mengevaluasi kegunaan dan keandalan model yang telah dibuat dan digunakan. Evaluasi dilakukan dengan menguji model pada data yang belum pernah dilihat sebelumnya menggunakan berbagai metrik performa untuk mengukur efektifitas model tersebut.



Gambar 2.2. Diagram alir model SEMMA