BAB 2
LANDASAN TEORI

Bab ini menjelaskan landasan teori dan tinjauan penelitian terdahulu yang menjadi dasar penelitian ini. Pembahasan dimulai dari penelitian terkait di bidang deteksi *fraud* kartu kredit, kemudian dilanjutkan dengan karakteristik dan tantangan deteksi *fraud*, *deep reinforcement learning* dan komponen utamanya, *supervised learning* dengan fokus pada *Random Forest*, kalibrasi probabilitas dan strategi penggabungan skor antar model, hingga metrik evaluasi yang digunakan. Seluruh konsep dan perumusan yang dibahas pada bab ini menjadi rujukan langsung pada metodologi Bab 3 dan hasil eksperimen Bab 4.

2.1 Penelitian Terkait

Subbab ini menelusuri fondasi metodologis yang menopang setiap komponen metode yang diusulkan, mulai dari penggabungan skor antar *classifier*, peran *reinforcement learning* sebagai sumber skor, kalibrasi probabilitas, hingga validitas *dataset*. Sebagai titik tolak, pendekatan *supervised ensemble* tanpa *reinforcement learning* sudah matang dan kompetitif pada metrik konvensional [4, 5, 6, 17]. Tabel 2.1 merangkum fondasi tersebut.

Tabel 2.1. Fondasi Metodologis bagi Metode yang Diusulkan

Peneliti	**Kontribusi Metodologis**	**Peran *RL* dalam *pipeline***	***Fusion* Skor Terkalibrasi**	**Dataset**
Fondasi 1 — Penggabungan skor antar classifier				
[14]	Rata-rata probabilitas berbobot antar *classifier*	—	Sebagian (tanpa kalibrasi)	Riil (Eropa)
[18]	Penurunan parameter operasional dari statistik aliran data	—	—	IEEE-*fraud*
Fondasi 2 — Reinforcement learning sebagai sumber skor				
[7]	DQN mandiri: menjanjikan, belum unggul	*Classifier* mandiri	Tidak	Kartu kredit publik
[9]	DQN komplementer terhadap model *tree* (*orthogonality*)	Sistem paralel terpisah	Tidak	Riil + UCI
[10]	*RL* sebagai penyetel *threshold*	Penyetel *threshold*	Tidak	Kartu kredit (1,6 jt)
[11]	*RL* penyetel *threshold* & bobot fitur	Penyetel *threshold*	Tidak	Riil skala besar
[12]	*Environment* agen dengan kapasitas aksi terbatas	Penyetel *threshold*	Tidak	Riil (ritel)
Fondasi 3 — Kalibrasi skor				
[13]	*Platt scaling*: skor skalar menjadi probabilitas	—	Alat kalibrasi	(survei)
Fondasi 4 — Validitas dataset Sparkov				
[15]	Uji *ensemble* Sparkov vs. data riil	—	—	Sparkov, Eropa
[16]	Uji *ensemble* pada Sparkov identik	—	—	Sparkov + 4 riil

Fondasi pertama menetapkan bahwa skor dari beberapa *classifier* dapat disatukan menjadi satu keputusan melalui mekanisme yang sederhana dan sudah teruji. Dal Pozzolo et al. [14], bekerja sama dengan mitra industri finansial,

menggabungkan skor dua *classifier* melalui rata-rata probabilitas berbobot, teknik yang sama dipakai penelitian ini. Perbedaannya terletak pada sifat kedua sumber skor: kedua *classifier* mereka sama-sama *supervised* dan sudah berskala probabilitas, sehingga dapat dirata-ratakan tanpa perlakuan tambahan. Hassanzadeh et al. [18] melengkapi fondasi ini dari sisi yang berbeda, yaitu dengan menunjukkan bahwa parameter operasional suatu sistem deteksi dapat diturunkan dari besaran statistik aliran data, bukan hanya ditetapkan secara sepihak. Pendekatan penurunan berbasis statistik inilah yang dipakai penelitian ini untuk menetapkan parameter *environment* pelatihan agen pada Bab 3. Keterbatasan fondasi pertama menghubungkan langsung ke pertanyaan berikutnya: apa yang terjadi bila salah satu sumber skor bukan *supervised*, melainkan *reinforcement learning*.

Di sinilah fondasi kedua berperan, dan literatur menempatkan *reinforcement learning* pada dua posisi yang keduanya belum sejajar dengan *supervised classifier*. Sebagai model mandiri, Ben Mekhlouf et al. [7] mendapati metrik DQN masih *modest* meski adaptif, sementara Papanastassiou et al. [9] menemukan hal yang lebih menentukan: meskipun model *tree* unggul pada metrik agregat, agen DQN konsisten menangkap sekitar 30% *fraud* unik yang tidak tertangkap model *tree*. Temuan ini menegaskan bahwa nilai *reinforcement learning* terletak pada komplementaritasnya, bukan keunggulan mandirinya, namun Papanastassiou et al. berhenti pada menjalankan kedua model sebagai sistem paralel yang terpisah tanpa pernah menyatukan skornya. Pada posisi kedua, sebagai penyetel ambang, Alkhozae et al. [10] dan Cui et al. [11] memakai *reinforcement learning* untuk menyetel *threshold* di belakang *classifier supervised*, sedangkan Shen & Kurshan [12] memakai DQN dengan cara serupa dan sekaligus menunjukkan bagaimana *environment* agen dapat dirancang dengan kapasitas aksi yang terbatas per periode agar agen belajar memprioritaskan, bukan menandai semuanya. Menyatukan temuan-temuan ini menghasilkan arah yang jelas: komplementaritas *reinforcement learning* nyata dan layak dimanfaatkan, tetapi memanfaatkannya sebagai sumber skor yang sejajar, bukan penyetel ambang atau sistem paralel, menuntut satu langkah teknis yang belum diambil, yaitu menyetarakan skalanya lalu menggabungkannya.

Langkah itulah yang disediakan fondasi ketiga. Skor *Random Forest* berskala $[0, 1]$ sedangkan *Q-value* DQN tak terbatas, sehingga keduanya perlu disetarakan sebelum digabung. Silva Filho et al. [13] menyediakan *Platt scaling*, yang memetakan skor skalar apa pun menjadi probabilitas terkalibrasi (dirinci pada Subbab 2.5). Dirangkai dengan rata-rata probabilitas berbobot dari Dal

Pozzolo et al. pada fondasi pertama, kedua alat ini melengkapi rangkaian *fusion*: kalibrasi menyetarakan skala, lalu *probability averaging* menyatukannya menjadi satu keputusan. Keduanya teruji sendiri-sendiri, tetapi belum pernah dirangkai untuk memasangkan *supervised classifier* dengan *reinforcement learning*.

Fondasi keempat memastikan rangkaian di atas diuji pada dasar pengujian yang kredibel. Jemai et al. [15] menguji *dataset* Sparkov terhadap data riil Eropa dan mendapati performa *ensemble supervised* konsisten lebih rendah pada Sparkov, sementara Al-Bulushi et al. [16] menguji rentang model yang lebih luas pada berkas Sparkov yang identik dengan penelitian ini (*bagging* terbaik: akurasi 0,99, *recall* 0,90, *precision* 0,77). Keduanya menegaskan Sparkov sudah teruji dan menyediakan angka pembanding pada metrik yang sama, tetapi sama-sama tidak melibatkan *reinforcement learning* sebagai komponen *ensemble*, dan pada kasus Al-Bulushi membagi data secara acak sehingga rentan *look-ahead bias*; penelitian ini memakai berkas yang sama dengan pemilahan kronologis.

Menyatukan keempat fondasi memperlihatkan pola yang matang: mekanisme penggabungan skor sudah mapan, komplementaritas *reinforcement learning* sudah terbukti meski perannya masih terbatas, alat kalibrasi sudah tersedia, dan *dataset* sudah teruji. Setiap komponen berdiri kokoh sendiri-sendiri; yang belum pernah dilakukan adalah merangkainya menjadi satu kerangka, yaitu menempatkan *reinforcement learning* sebagai sumber skor sejajar yang dikalibrasi melalui *Platt scaling* dan digabungkan dengan *Random Forest* melalui rata-rata probabilitas berbobot menjadi satu keputusan. Celah inilah yang diisi penelitian ini.

2.2 *Credit Card Fraud Detection*

2.2.1 Karakteristik

Transaksi *fraud* kartu kredit memiliki tiga karakteristik utama. Pertama, *class imbalance* yang ekstrem, dengan rasio transaksi *fraud* terhadap transaksi normal pada *dataset* dunia nyata umumnya di bawah 1% [3]. Kedua, pelaku *fraud* terus mengubah pola serangan dari waktu ke waktu untuk menghindari deteksi [2]. Ketiga, transaksi bergantung pada konteks temporal seperti waktu transaksi, lokasi geografis, dan riwayat perilaku pengguna [19].

2.2.2 Tantangan

Sistem deteksi *fraud* kartu kredit menghadapi tiga tantangan teknis yang saling berkaitan.

Pertama, *class imbalance* ekstrem menyebabkan model *machine learning* cenderung bias terhadap kelas mayoritas, sehingga *recall* pada kelas minoritas menjadi rendah jika tidak ditangani secara eksplisit [3, 4]. Pendekatan penanganannya terbagi menjadi dua kelompok. Kelompok pertama mengubah distribusi data sebelum pelatihan, yaitu *oversampling* kelas minoritas (misalnya SMOTE atau VAE-GAN [17]) dan *undersampling* kelas mayoritas. Kelompok kedua tidak mengubah data, melainkan mengubah perlakuan algoritma terhadap kesalahan pada tiap kelas, yaitu pembobotan kelas (*class weighting*) yang bekerja langsung pada kriteria optimasi model, dan pengaturan komposisi *batch* pelatihan agar kelas minoritas tetap terwakili. Perbedaan keduanya penting: kelompok pertama menciptakan sampel yang sebenarnya tidak ada pada data asli, sedangkan kelompok kedua mempertahankan data apa adanya. Pada data yang memiliki urutan waktu, kelompok kedua lebih aman karena tidak menyisipkan sampel sintetis yang dapat merusak konsistensi kronologis.

Kedua, *concept drift*, yaitu perubahan distribusi data dari waktu ke waktu, menjadi tantangan signifikan karena pelaku *fraud* terus mengubah pola serangan. Hal ini menyebabkan model yang dilatih pada data masa lalu mengalami penurunan performa ketika diterapkan pada data terbaru [19]. Pendekatan adaptif berbasis *reinforcement learning* mulai dieksplorasi untuk mengatasi tantangan ini [7, 11].

Ketiga, keluaran model yang berbeda paradigma berada pada skala yang tidak setara, sehingga tidak dapat langsung disatukan. Tantangan ini muncul ketika sistem deteksi tidak lagi bertumpu pada satu model, melainkan menggabungkan beberapa model untuk memanfaatkan kelebihan masing-masing. Selama seluruh model yang digabungkan merupakan *classifier* probabilistik, persoalan ini tidak muncul karena keluarannya sudah sama-sama berada pada rentang $[0, 1]$ dan dapat diperbandingkan secara langsung. Persoalan baru timbul ketika salah satu sumber skor berasal dari paradigma yang berbeda. Model berbasis *reinforcement learning*, misalnya, menghasilkan estimasi nilai aksi yang besarannya ditentukan oleh skala *reward* yang dirancang perancang sistem, bukan oleh peluang suatu kejadian, sehingga nilai yang besar secara absolut belum tentu mencerminkan kemungkinan *fraud* yang lebih tinggi. Menggabungkan skor semacam itu dengan probabilitas tanpa penyetaraan akan membuat satu model mendominasi hasil akhir semata-

mata karena rentang keluarannya lebih lebar, bukan karena prediksinya lebih baik. Penyelesaian atas tantangan ini, yaitu kalibrasi skor menjadi probabilitas, dibahas pada Subbab 2.5.

2.3 *Deep Reinforcement Learning*

Deep Reinforcement Learning (DRL) adalah cabang *machine learning* yang menggabungkan *reinforcement learning* dengan *deep neural network* untuk memungkinkan agen belajar membuat keputusan optimal melalui interaksi dengan lingkungan [20]. Berbeda dengan *supervised learning* yang belajar dari pasangan data dan label statis, DRL belajar dari sinyal *reward* yang diterima sebagai konsekuensi dari tindakan yang diambil. Karakteristik pembelajaran berbasis interaksi ini memungkinkan DRL menangkap pola yang muncul dari urutan keputusan, bukan hanya pola statis antar fitur.

2.3.1 *Markov Decision Process*

Markov Decision Process (MDP) adalah kerangka matematis untuk memodelkan masalah pengambilan keputusan sekuensial di bawah ketidakpastian [20]. MDP didefinisikan oleh tupel $\langle \mathscr{S}, \mathscr{A}, P, R, \gamma \rangle$, dengan $\mathscr{S}$ merupakan himpunan *state*, $\mathscr{A}$ merupakan himpunan aksi, $P(s' \mid s, a)$ merupakan probabilitas transisi dari *state* s ke s' ketika aksi a dilakukan, $R(s,a)$ merupakan *reward* setelah melakukan aksi a pada *state* s, dan $\gamma \in [0,1]$ merupakan *discount factor* yang menentukan bobot relatif antara *reward* jangka pendek dan jangka panjang. Tujuan agen adalah belajar suatu *policy* $\pi : \mathscr{S} \to \mathscr{A}$ yang memaksimalkan *expected cumulative reward*:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2.1}$$

Karena *return* pada Persamaan 2.1 dapat mengikutsertakan *reward* jauh ke depan, fungsi *reward* dapat dirancang agar sinyal yang diterima agen sejalan dengan perilaku akhir yang diinginkan. Sifat ini memberi perancang sistem kontrol terhadap perilaku agen melalui desain *reward*, bukan hanya melalui pemilihan algoritma.

A Formulasi Deteksi *Fraud* sebagai MDP

Deteksi *fraud* dapat diformulasikan sebagai MDP dengan memandang proses pemeriksaan sebagai interaksi berurutan antara agen dan aliran transaksi, bukan sebagai klasifikasi tiap transaksi yang berdiri sendiri [9, 21]. Pada formulasi ini, *state* s_t merepresentasikan transaksi yang sedang dinilai beserta konteks keputusan sejauh langkah t, aksi $a_t \in \{0,1\}$ menyatakan keputusan melewati (*skip*) atau menandai (*flag*) transaksi, dan *reward* r_t mengukur ketepatan keputusan tersebut terhadap label sebenarnya [9]. Papanastassiou et al. [9] menekankan bahwa keunggulan formulasi ini terletak pada kemampuan *state* memuat konteks temporal, sehingga agen dapat menilai transaksi berdasarkan situasi keputusan yang sedang berlangsung, bukan hanya nilai fitur sesaat. Secara umum, *state* memuat representasi fitur transaksi beserta fitur konteks keputusan, dan aksi bersifat biner:

$$s_t = [\underbrace{x_t^{(1)}, \ldots, x_t^{(n)}}_{\text{fitur transaksi}}, \underbrace{c_t^{(1)}, \ldots, c_t^{(m)}}_{\text{fitur konteks}}] \tag{2.2}$$

dengan x_t merupakan fitur transaksi, c_t fitur konteks keputusan, serta $a_t = 0$ menyatakan *skip* dan $a_t = 1$ menyatakan *flag* [9].

Konteks keputusan yang dimuat dalam *state* memungkinkan agen menyesuaikan perilakunya terhadap kondisi yang sedang berlangsung, misalnya sisa langkah pada episode dan ketepatan keputusan sebelumnya. Kemampuan ini membedakan formulasi MDP dari *classifier* statis, yang menilai tiap transaksi dengan kriteria sama tanpa memperhitungkan keputusan lain yang telah diambil. Perlu dicatat bahwa fitur konteks dihitung dari keputusan agen sendiri, bukan dari label, sehingga tetap tersedia pada tahap inferensi ketika label tidak diketahui.

B Struktur Episode dan Pembatasan Aksi

Pembelajaran *reinforcement* memerlukan episode dengan awal dan akhir yang terdefinisi, sebab *return* pada Persamaan 2.1 dihitung sepanjang satu episode. Pada aliran transaksi yang bersifat kontinu, episode dibentuk dengan memotong aliran tersebut menjadi periode berukuran tetap.

Selain pembatasan panjang episode, *environment* juga dapat membatasi jumlah aksi tertentu yang boleh diambil agen dalam satu episode. Shen & Kurshan [12] merancang *environment* agen deteksi *fraud* dengan kapasitas aksi penandaan yang terbatas per periode, dan menunjukkan bahwa pembatasan

semacam ini mengubah persoalan yang dipelajari agen: agen tidak lagi cukup memutuskan apakah suatu transaksi mencurigakan, melainkan harus memutuskan apakah transaksi tersebut cukup meyakinkan untuk menghabiskan salah satu aksi yang tersedia. Tanpa pembatasan tersebut, kebijakan yang menandai hampir seluruh transaksi akan memperoleh *reward* tinggi pada komponen deteksi, sehingga pembelajaran berhenti pada solusi trivial.

Besaran kapasitas aksi tersebut dapat ditetapkan dari dua dasar yang berbeda. Dasar pertama bersumber dari kondisi operasional suatu institusi, sebagaimana pada Shen & Kurshan [12] yang menurunkannya dari kapasitas tim pemeriksa. Dasar kedua bersumber dari besaran statistik data itu sendiri: Hassanzadeh et al. [18] memformalkan kapasitas pemeriksaan sebagai fraksi dari laju kedatangan transaksi, yaitu besaran yang diturunkan langsung dari statistik aliran data. Ketika penelitian tidak melibatkan institusi yang dapat menetapkan kapasitas secara operasional, dasar kedua menjadi pilihan yang tersedia. Penerapan konkret kedua pembatasan ini pada penelitian ini dijabarkan pada Bab 3.

C Desain Fungsi *Reward*

Fungsi *reward* pada formulasi MDP untuk deteksi *fraud* umumnya disusun berdasarkan ketepatan keputusan agen terhadap label sebenarnya, dengan membedakan empat kemungkinan hasil sesuai kombinasi aksi a_t dan label y_t [9]:

$$r_t(a_t, y_t) = \begin{cases} +r_1 & \text{jika } a_t = 1 \text{ dan } y_t = 1 \text{ (TP)} \\ -r_1 & \text{jika } a_t = 0 \text{ dan } y_t = 1 \text{ (FN)} \\ +r_2 & \text{jika } a_t = 0 \text{ dan } y_t = 0 \text{ (TN)} \\ -r_2 & \text{jika } a_t = 1 \text{ dan } y_t = 0 \text{ (FP)} \end{cases} \tag{2.3}$$

Pada Persamaan 2.3, r_1 dan r_2 merupakan parameter besaran *reward* yang nilainya ditentukan sesuai kebutuhan desain, sehingga fungsi *reward* dapat disesuaikan dengan karakteristik domain dan strategi pembelajaran yang diinginkan [9]. Perlu ditegaskan bahwa label y_t hanya digunakan untuk menghitung *reward* pada tahap pelatihan; pada tahap inferensi, keputusan agen ditentukan sepenuhnya dari *state* pada Persamaan 2.2 tanpa memerlukan label, sehingga formulasi ini tetap merupakan pembelajaran dari interaksi dan bukan pemetaan langsung dari fitur ke label.

Selain *reward* per keputusan, struktur *reward* dapat dilengkapi sinyal di

akhir episode agar agen mengoptimasi perilakunya sepanjang episode, bukan hanya tiap transaksi secara terpisah. Karena *return* pada MDP dapat mengikutsertakan *reward* yang jauh ke depan (Persamaan 2.1), sinyal akhir episode ini ter-propagasi mundur ke keputusan-keputusan sebelumnya. Secara umum, sinyal akhir episode dapat dituliskan sebagai fungsi dari suatu metrik tingkat episode yang diskalakan dengan sebuah koefisien bobot:

$$r_{\text{EOW}} = \alpha \times \left(M_{\text{periode}} + \lambda \times b\right) \tag{2.4}$$

dengan M_{periode} merupakan metrik yang dihitung pada akhir episode, b sebuah bonus pemenuhan kendala, serta α dan λ merupakan koefisien bobot yang ditentukan sesuai kebutuhan desain [9]. Pemilihan M_{periode} merupakan keputusan desain pada tahap pelatihan dan tidak menentukan metrik yang digunakan untuk menilai performa akhir model; keduanya merupakan hal yang terpisah.

2.3.2 *Q-Learning*

Q-learning adalah algoritma *reinforcement learning* berbasis nilai (*value-based*) yang belajar fungsi nilai aksi $Q(s,a)$ untuk mengestimasi total *reward* yang dapat diperoleh jika mengambil aksi a pada *state* s kemudian mengikuti *policy* optimal [20]. Persamaan Bellman untuk fungsi Q optimal didefinisikan sebagai:

$$Q^*(s,a) = \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} Q^*(s',a') \mid s_t = s, a_t = a\right] \tag{2.5}$$

Pembaruan iteratif untuk *Q-learning* dilakukan dengan aturan:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right] \tag{2.6}$$

dengan α merupakan *learning rate*. Pada penerapan klasik, fungsi Q direpresentasikan sebagai tabel *lookup*. Namun, representasi tabular tidak dapat diterapkan pada *state space* yang besar atau kontinu sehingga diperlukan aproksimasi fungsi seperti *neural network*.

2.3.3 *Deep Q-Network*

Deep Q-Network (DQN) merupakan pengembangan *Q-learning* yang menggunakan *deep neural network* sebagai aproksimator fungsi $Q(s,a;\theta)$, dengan θ merupakan parameter *neural network* [20]. *Neural network* menerima representasi *state* sebagai *input* dan menghasilkan estimasi nilai Q untuk setiap aksi. Pelatihan dilakukan dengan meminimalkan *loss function*:

$$\mathscr{L}(\theta) = \mathbb{E}\left[\left(r + \gamma \max_{a'} Q(s',a';\theta) - Q(s,a;\theta)\right)^2\right] \tag{2.7}$$

Nilai Q yang dihasilkan Persamaan 2.7 tidak terbatas pada rentang tertentu, melainkan bergantung pada besaran *reward* yang dirancang dan pada *discount factor* yang digunakan. Sifat inilah yang menjadi sumber tantangan ketiga pada Subbab 2.2.2, dan yang mengharuskan kalibrasi sebelum *Q-value* dapat digabungkan dengan probabilitas model lain.

Penerapan DQN pada deteksi *fraud* kartu kredit telah dieksplorasi pada beberapa penelitian terkini [7, 8, 9], di mana DQN diposisikan sebagai agen yang membuat keputusan biner *flag* atau *skip* pada setiap transaksi dengan struktur *reward* yang mengutamakan kelas minoritas.

2.3.4 *Experience Replay*

Pelatihan DQN secara langsung dari urutan transisi yang dialami agen menghadapi dua masalah utama: korelasi antar sampel berurutan yang melanggar asumsi data *independent and identically distributed* (i.i.d.), serta ketidakstabilan karena distribusi data berubah seiring kebijakan agen [20]. *Experience replay* mengatasi masalah ini dengan menyimpan transisi (s_t, a_t, r_t, s_{t+1}) pada *replay buffer*, kemudian mengambil sampel *batch* secara acak dari *buffer* untuk pembaruan parameter. Pendekatan ini memutus korelasi temporal antar sampel dan memungkinkan setiap transisi digunakan untuk pembaruan beberapa kali.

Pada kasus *class imbalance* ekstrem, *experience replay* dapat dimodifikasi dengan strategi *stratified sampling* untuk memastikan setiap *batch* mengandung proporsi pengalaman *fraud* dan normal yang seimbang. Strategi ini menjamin agen tetap memperoleh sinyal pembelajaran dari kelas minoritas pada setiap pembaruan parameter, meskipun kelas tersebut sangat jarang muncul pada data asli. Tanpa modifikasi ini, sebagian besar *batch* hanya akan berisi transaksi normal sehingga

agen sulit mempelajari pola *fraud*. Perlu dicatat bahwa *stratified experience replay* merupakan penanganan *class imbalance* yang termasuk kelompok kedua pada Subbab 2.2.2, yaitu mengubah komposisi *batch* pelatihan tanpa menciptakan sampel sintetis maupun mengubah data asli.

2.3.5 *Double Deep Q-Network*

Double Deep Q-Network (Double DQN) merupakan perbaikan terhadap DQN standar yang ditujukan untuk mengatasi *overestimation bias* pada nilai Q [22]. Pada DQN standar, operator max yang sama digunakan untuk memilih aksi terbaik dan mengestimasi nilainya pada *target network*. Penggunaan ganda ini dapat menyebabkan estimasi nilai Q yang terlalu tinggi karena *noise* estimasi ikut diperkuat.

Double DQN memisahkan pemilihan aksi dan evaluasi nilai dengan menggunakan dua *neural network* terpisah: *policy network* θ yang memilih aksi terbaik, dan *target network* θ^- yang mengevaluasi nilai aksi tersebut. Target pembaruan pada Double DQN didefinisikan sebagai:

$$y = r + \gamma Q\left(s',\ \arg\max_{a'} Q(s',a';\theta);\ \theta^-\right) \tag{2.8}$$

Target network θ^- diperbarui secara berkala dengan menyalin parameter dari *policy network*. Pemisahan ini terbukti mengurangi *overestimation bias* dan meningkatkan kestabilan pelatihan [22].

2.3.6 *Dueling Deep Q-Network*

Dueling Deep Q-Network (*Dueling* DQN) memperkenalkan arsitektur *neural network* yang memisahkan estimasi nilai $Q(s,a)$ menjadi dua komponen: *state-value function* $V(s)$ yang merepresentasikan nilai dari berada pada *state* s, dan *advantage function* $A(s,a)$ yang merepresentasikan keuntungan relatif dari memilih aksi a dibandingkan aksi lain pada *state* yang sama [23]. Hubungan antara ketiga fungsi didefinisikan sebagai:

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + \left(A(s,a;\theta,\alpha) - \frac{1}{|\mathscr{A}|}\sum_{a'} A(s,a';\theta,\alpha)\right) \tag{2.9}$$

dengan θ merupakan parameter *shared feature extractor*, α merupakan parameter *advantage stream*, dan β merupakan parameter *value stream*. Pengurangan rata-rata *advantage* pada Persamaan 2.9 memastikan identifikasi unik antara $V(s)$ dan $A(s,a)$, sehingga estimasi nilai Q menjadi lebih stabil.

Arsitektur *Dueling* memberikan keuntungan pada lingkungan di mana sebagian besar *state* memiliki nilai aksi yang serupa. Pada kondisi tersebut, *neural network* dapat fokus mempelajari $V(s)$ tanpa harus mempelajari $A(s,a)$ untuk setiap aksi secara independen. Karakteristik ini relevan dengan deteksi *fraud* kartu kredit, di mana sebagian besar transaksi merupakan transaksi normal sehingga perbedaan antara aksi *flag* dan *skip* pada sebagian besar *state* sangat kecil.

2.4 *Supervised Learning* untuk *Credit Card Fraud Detection*

Supervised learning merupakan paradigma *machine learning* yang paling banyak digunakan untuk deteksi *fraud* kartu kredit karena ketersediaan label transaksi historis. Berbagai algoritma telah dievaluasi pada tugas ini, termasuk *Logistic Regression*, *Support Vector Machine*, *Decision Tree*, *Multi-Layer Perceptron*, *Random Forest*, dan *Gradient Boosting* [4, 5, 6]. Bagian ini secara khusus membahas *Random Forest* karena algoritma ini menjadi salah satu komponen utama *ensemble* yang diusulkan pada penelitian ini.

2.4.1 *Random Forest*

Random Forest adalah algoritma *ensemble* berbasis pohon keputusan (*decision trees*) yang dilatih melalui dua mekanisme utama: *bootstrap aggregating* (*bagging*) dan seleksi fitur acak pada setiap pemecahan *node* [4]. Setiap pohon dilatih pada *subset* data hasil *bootstrap sampling*, dan pada setiap pemecahan *node* hanya *subset* fitur secara acak yang dipertimbangkan untuk mencari pemecahan terbaik. Kombinasi kedua mekanisme ini menghasilkan *ensemble* pohon yang beragam (*diverse*), sehingga *variance* prediksi berkurang secara signifikan dibandingkan pohon keputusan tunggal.

Untuk klasifikasi diskrit, prediksi akhir *Random Forest* dapat diperoleh melalui *majority voting* dari prediksi seluruh pohon. Dengan T pohon dalam *ensemble* dan $h_t(x)$ merupakan prediksi label oleh pohon ke-t:

$$\hat{y}(x) = \text{mode}\{h_1(x), h_2(x), \ldots, h_T(x)\} \tag{2.10}$$

Random Forest banyak digunakan dalam deteksi *fraud* kartu kredit karena tiga keunggulan praktis. Pertama, algoritma ini mampu menangani *class imbalance* secara internal melalui pembobotan kelas (*class weight*) atau *undersampling* yang terintegrasi pada *bootstrap sampling* [4, 6]. Kedua, algoritma ini memberikan estimasi pentingnya fitur melalui *Mean Decrease in Impurity* (MDI), yaitu total penurunan *Gini impurity* pada setiap pemecahan *node* yang melibatkan fitur tersebut, dirata-rata di seluruh pohon. Nilai ini dapat dianalisis untuk memahami kontribusi tiap fitur dan telah terbukti sebagai dasar seleksi fitur yang kompetitif pada deteksi *fraud* kartu kredit [24]. Ketiga, algoritma ini relatif tahan terhadap *overfitting* dibandingkan pohon keputusan tunggal karena rata-rata antar pohon. Ketiga keunggulan ini bersumber dari satu mekanisme yang sama, yaitu cara *Random Forest* memilih *node* pemecah, yang dijabarkan pada subbab berikut.

2.4.2 Pemilihan *Node* Pemecah pada *Random Forest*

Setiap pohon dalam *Random Forest* dibangun secara rekursif dengan memilih, pada tiap titik pemecahan (*node*), satu fitur beserta ambang pemisahnya. Pemilihan ini berlangsung dalam dua tahap yang bekerja bersama, dan keduanya menentukan sekaligus struktur pohon dan peringkat kepentingan fitur yang dihasilkan model.

Tahap pertama adalah pembatasan kandidat secara acak. Dari n fitur yang tersedia, hanya sebagian yang dipilih secara acak untuk dipertimbangkan pada *node* tersebut, dinotasikan sebagai himpunan kandidat $\mathscr{F}_m$ dengan $|\mathscr{F}_m| < n$. Pembatasan ini berlaku pada seluruh *node*, termasuk *node* teratas (*root*). Tanpa pembatasan acak tersebut, seluruh pohon akan memilih fitur pemecah yang sama pada posisi yang sama sehingga pohon-pohon menjadi hampir identik, dan pengurangan *variance* yang menjadi alasan utama penggunaan *ensemble* tidak tercapai [4].

Tahap kedua adalah pemilihan berdasarkan kriteria *impurity*. Di antara fitur kandidat tersebut, dipilih pasangan fitur j dan ambang τ yang paling memisahkan kelas positif dari kelas negatif. Ukuran ketercampuran kelas pada suatu *node* m dinyatakan melalui *Gini impurity*, dengan $p_{m,c}$ merupakan proporsi sampel berbobot kelas c pada *node* tersebut:

$$G(m) = 1 - \sum_{c \in \{0,1\}} p_{m,c}^2 \tag{2.11}$$

Nilai $G(m)$ bernilai nol ketika seluruh sampel pada *node* berasal dari satu kelas, dan mencapai maksimum ketika kedua kelas berimbang. Sebuah pemecahan dinilai baik apabila menghasilkan dua *node* anak yang masing-masing lebih murni dibanding *node* induknya. Penurunan *impurity* akibat pemecahan *node* m menjadi anak kiri m_L dan anak kanan m_R didefinisikan sebagai selisih antara *impurity* induk dan rata-rata berbobot *impurity* kedua anaknya:

$$\Delta G(m, j, \tau) = G(m) - \frac{N_{m_L}}{N_m} G(m_L) - \frac{N_{m_R}}{N_m} G(m_R) \tag{2.12}$$

dengan N_m, N_{m_L}, dan N_{m_R} merupakan jumlah sampel berbobot pada *node* induk dan kedua anaknya. Pembobotan $\frac{N_{m_L}}{N_m}$ dan $\frac{N_{m_R}}{N_m}$ mencegah pemecahan yang hanya memisahkan sedikit sampel dinilai unggul semata-mata karena *node* anak yang kecil cenderung murni.

Fitur pemecah yang akhirnya dipilih pada *node* m adalah pasangan yang memaksimalkan penurunan *impurity* di antara seluruh kandidat:

$$(j^*, \tau^*) = \arg \max_{(j,\tau) \in \mathscr{F}_m \times \mathscr{T}_j} \Delta G(m, j, \tau) \tag{2.13}$$

dengan $\mathscr{T}_j$ merupakan himpunan ambang kandidat untuk fitur j.

Ketiga persamaan di atas sekaligus menjelaskan dua hal yang saling terkait. Pertama, pada kondisi *class imbalance*, pembobotan kelas *cost-sensitive* bekerja tepat di dalam persamaan ini: bobot kelas minoritas masuk melalui $p_{m,c}$ pada Persamaan 2.11 dan melalui N_m pada Persamaan 2.12, sehingga satu sampel kelas minoritas dihitung setara sekian sampel kelas mayoritas dan pemecahan yang memisahkan kelas minoritas tetap terpilih meskipun jumlah sampelnya sedikit. Penanganan ketimpangan kelas dengan demikian tidak dilakukan di luar algoritma, melainkan melekat pada kriteria pembentukan pohon.

Kedua, akumulasi ΔG yang disumbangkan sebuah fitur di seluruh *node* tempat fitur tersebut terpilih, dirata-rata di seluruh pohon, merupakan definisi dari *Mean Decrease in Impurity* (MDI) yang telah disebutkan pada Subbab 2.4.1. Peringkat kepentingan fitur karena itu bukan perhitungan tambahan di luar model,

melainkan produk sampingan langsung dari proses pemilihan *node* pemecah, sehingga dapat digunakan sebagai seleksi fitur yang bersifat *embedded*. Pendekatan berbasis *importance* bawaan ini terbukti sah dan bahkan direkomendasikan untuk *dataset* berskala besar, karena lebih ringan secara komputasi namun tetap kompetitif dibandingkan metode seleksi yang lebih kompleks seperti SHAP [24].

2.4.3 Keluaran Probabilistik *Random Forest*

Selain prediksi label diskrit, *Random Forest* menghasilkan probabilitas prediksi yang merepresentasikan tingkat keyakinan model terhadap setiap kelas. Pada implementasi standar pustaka *scikit-learn*, probabilitas tersebut dihitung sebagai rata-rata probabilitas kelas yang diestimasi oleh setiap pohon, bukan sebagai *majority vote* dari label diskrit. Dengan $P_t(y = c \mid x)$ merupakan probabilitas kelas c yang diestimasi oleh pohon ke-t (umumnya berupa proporsi sampel kelas c pada *leaf node* yang diaktifkan oleh *input* x):

$$P(y = c \mid x) = \frac{1}{T} \sum_{t=1}^{T} P_t(y = c \mid x) \tag{2.14}$$

Pada konteks deteksi *fraud* kartu kredit, probabilitas $P(y = 1 \mid x)$ yaitu probabilitas transaksi diklasifikasikan sebagai *fraud* dapat digunakan sebagai skor risiko transaksi [25]. Keluaran probabilistik *Random Forest* berada pada rentang $[0, 1]$ secara alami sehingga dapat digabungkan dengan keluaran model lain yang juga menghasilkan probabilitas tanpa transformasi tambahan. Sifat ini berbeda dengan keluaran *Deep Q-Network* pada Persamaan 2.7, dan perbedaan inilah yang menuntut kalibrasi pada subbab berikut.

2.5 Kalibrasi dan Penggabungan Skor

2.5.1 Motivasi Kalibrasi Probabilitas

Sebuah model klasifikasi probabilistik dikatakan terkalibrasi dengan baik (*well-calibrated*) apabila probabilitas yang dihasilkannya secara konsisten merepresentasikan tingkat keyakinan yang sebenarnya [13]. Sebagai contoh, jika model memprediksi probabilitas *fraud* sebesar 0,8 untuk sejumlah transaksi, maka idealnya sekitar 80% dari transaksi tersebut benar-benar merupakan transaksi *fraud*. Untuk klasifikasi biner, definisi formal kalibrasi yang disampaikan Silva Filho et al.

[13] adalah:

$$P(Y = 1 \mid f(X) = s) = s \tag{2.15}$$

dengan $f(X)$ merupakan skor yang dihasilkan model untuk *input* X, dan $s \in [0, 1]$ merupakan nilai prediksi. Persamaan 2.15 menyatakan bahwa pada subset *instance* yang menerima prediksi skor s, proporsi aktual yang berlabel positif harus mendekati s. Kalibrasi yang baik penting untuk pengambilan keputusan yang mempertimbangkan biaya kesalahan karena ambang keputusan optimal dapat ditentukan secara langsung dari probabilitas terkalibrasi.

Banyak algoritma *machine learning* cenderung menghasilkan probabilitas yang tidak terkalibrasi, baik karena terlalu yakin (probabilitas terlalu ekstrem mendekati 0 atau 1) maupun kurang yakin (probabilitas terlalu dekat dengan 0,5) [13]. Pada konteks penelitian ini, masalah kalibrasi menjadi krusial karena *Random Forest* menghasilkan probabilitas pada rentang $[0, 1]$ secara alami melalui Persamaan 2.14, sedangkan *Deep Q-Network* menghasilkan *Q-value* yang tidak terbatas pada rentang tertentu. Tanpa penyetaraan skala, kedua sinyal tidak dapat digabungkan secara valid karena *Q-value* yang besar secara absolut belum tentu mencerminkan kemungkinan *fraud* yang lebih tinggi.

Post-hoc calibration merupakan pendekatan kalibrasi yang dilakukan setelah model utama selesai dilatih. Pendekatan ini mempelajari fungsi pemetaan (*calibration map*) $g : \mathbb{S}_Y \to \mathbb{P}_Y$ dari ruang keluaran model terlatih $\mathbb{S}_Y$ ke ruang probabilitas terkalibrasi $\mathbb{P}_Y$, menggunakan *calibration set* berupa data validasi yang terpisah dari data pelatihan model utama [13]. Pendekatan ini tidak memengaruhi parameter model utama dan dapat diterapkan pada berbagai algoritma yang menghasilkan skor skalar.

Pemilihan data validasi sebagai *calibration set* memiliki dasar teknis. Jika kalibrasi dipelajari dari data pelatihan, parameter kalibrasi akan menyesuaikan diri terhadap distribusi keluaran yang sudah dihafalkan oleh model utama sehingga hasilnya cenderung terlalu yakin dan tidak mencerminkan ketidakpastian sebenarnya pada data yang belum pernah dilihat [13]. Sebaliknya, jika kalibrasi dipelajari dari data uji, akan terjadi kebocoran informasi (*data leakage*) yang membuat estimasi performa akhir bias optimistik. Data validasi adalah satu-satunya pilihan netral.

2.5.2 *Platt Scaling*

Platt scaling merupakan salah satu metode *post-hoc calibration* yang paling banyak digunakan dalam praktik. Metode ini awalnya diperkenalkan untuk *Support Vector Machine*, namun dapat diterapkan pada skor skalar apa pun. Pada metode ini, skor tidak terkalibrasi s ditransformasikan menjadi probabilitas terkalibrasi melalui fungsi sigmoid logistik [13]:

$$g(s; w, b) = \frac{1}{1 + \exp(-w \cdot s - b)} \tag{2.16}$$

Pada Persamaan 2.16, $w \in \mathbb{R}$ merupakan *shape parameter* yang menentukan kemiringan sigmoid, dan $b \in \mathbb{R}$ merupakan *location parameter* yang menentukan pergeseran titik tengah sigmoid. Kedua parameter diestimasi dengan memaksimalkan *log-likelihood* (atau secara ekuivalen meminimalkan *log-loss*) pada data validasi dengan s sebagai fitur tunggal dan label sebenarnya sebagai target, sehingga *Platt scaling* dapat dipahami sebagai *logistic regression* satu fitur [13].

Keunggulan *Platt scaling* adalah kesederhanaan dan jumlah parameter yang sedikit (hanya w dan b), sehingga membutuhkan jumlah data validasi yang relatif sedikit untuk pelatihan dan tidak rentan terhadap *overfitting*. Penerapan pada *Q-value* dari *Deep Q-Network* dimungkinkan karena keluaran tersebut juga berupa skor skalar [13]. Karena fungsi sigmoid bersifat monoton naik, kalibrasi ini tidak mengubah urutan relatif antar transaksi berdasarkan skor aslinya, melainkan hanya memetakan skor tersebut ke rentang $[0, 1]$ dengan interpretasi probabilistik. Sifat monoton inilah yang membuat kalibrasi aman diterapkan: informasi yang dipelajari agen tidak hilang, hanya diubah skalanya agar dapat digabungkan dengan keluaran model lain.

2.5.3 *Ensemble Fusion* melalui *Probability Averaging*

Setelah keluaran dari dua atau lebih model berada pada skala probabilitas yang seragam, skor-skor tersebut dapat digabungkan untuk menghasilkan prediksi *ensemble* yang memanfaatkan kelebihan masing-masing model. Salah satu strategi penggabungan yang paling sederhana dan efektif adalah rata-rata probabilitas berbobot (*weighted probability averaging*) [14, 25]:

$$p_{\text{ensemble}}(x) = w \cdot p_A(x) + (1 - w) \cdot p_B(x) \tag{2.17}$$

dengan $p_A(x)$ dan $p_B(x)$ merupakan probabilitas terkalibrasi dari dua model yang berbeda, dan $w \in [0, 1]$ merupakan bobot yang menentukan kontribusi relatif kedua model. Bobot w umumnya ditentukan secara empiris melalui pencarian (*grid search*) pada data validasi dengan optimasi metrik tertentu. Penentuan bobot pada data validasi, bukan pada data uji, diperlukan agar estimasi performa akhir tidak bias optimistik.

Strategi *probability averaging* memiliki beberapa keunggulan dibandingkan strategi penggabungan yang lebih kompleks, seperti *stacking*, *boosting*, atau *fusion* berbasis keandalan seperti *Bayesian reliability fusion* yang menimbang tiap komponen menurut riwayat performanya [26]. Pertama, pendekatan ini tidak memerlukan pelatihan *meta-learner* tambahan sehingga lebih sederhana dan efisien secara komputasi. Kedua, hasil penggabungan tetap berada pada rentang $[0, 1]$ dan dapat diinterpretasikan sebagai probabilitas. Ketiga, karena hanya memiliki satu parameter bebas, kontribusi masing-masing model terhadap hasil akhir dapat dibaca secara langsung dari nilai w, sesuatu yang tidak dimungkinkan oleh *meta-learner* yang lebih kompleks. Keempat, pendekatan ini terbukti efektif pada studi terkini untuk deteksi *fraud* kartu kredit [4].

2.5.4 Komplementaritas Antar Model

Efektivitas *ensemble* bergantung pada keberagaman antar model komponennya. Model yang menghasilkan prediksi sama persis tidak memberikan manfaat tambahan ketika digabungkan, sedangkan model yang menangkap pola berbeda dari data akan saling melengkapi sehingga *ensemble* mampu menangkap lebih banyak kasus positif dibandingkan model individual [4]. Sumber keberagaman dapat berasal dari perbedaan cara belajar antar algoritma, perbedaan paradigma pembelajaran (*supervised* versus *reinforcement learning*), atau perbedaan representasi fitur.

Pada penggabungan *Random Forest* dengan *Deep Q-Network*, perbedaan cara belajar bersifat mendasar. *Random Forest* mempelajari pemetaan langsung dari fitur transaksi ke label melalui pemecahan rekursif berbasis *impurity Gini* pada setiap *node* sebagaimana Persamaan 2.13, yang merupakan kriteria lokal tanpa mempertimbangkan rangkaian keputusan. Sebaliknya, *Deep Q-Network*

mempelajari *policy* pengambilan keputusan melalui maksimasi *cumulative reward* sebagaimana Persamaan 2.1, sehingga *policy*-nya menyerap struktur *reward* yang tidak tertangkap oleh kriteria *impurity*. Perbedaan sumber sinyal inilah yang menjadi dasar harapan bahwa kedua model akan salah pada kasus yang berbeda.

Kuantifikasi komplementaritas dapat dilakukan melalui analisis irisan dan gabungan dari himpunan transaksi yang berhasil dideteksi oleh masing-masing model (*detection set*). Salah satu metrik yang umum digunakan adalah *Jaccard similarity* yang mengukur tingkat tumpang tindih antara dua himpunan:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{2.18}$$

dengan A dan B merupakan himpunan transaksi *fraud* yang berhasil dideteksi oleh model A dan model B. Nilai J yang rendah mengindikasikan komplementaritas tinggi, di mana kedua model menangkap subset transaksi yang berbeda. Sebaliknya, nilai J yang tinggi mengindikasikan kedua model menangkap subset transaksi yang serupa, sehingga manfaat penggabungan menjadi terbatas.

2.6 *Evaluation Metrics*

Pemilihan metrik evaluasi yang tepat menjadi krusial pada tugas deteksi *fraud* kartu kredit karena *class imbalance* yang ekstrem menyebabkan metrik konvensional seperti *accuracy* menjadi tidak informatif [3]. Bagian ini menjelaskan metrik-metrik yang digunakan, disusun mulai dari *confusion matrix* sebagai dasar seluruh metrik berbasis ambang, dilanjutkan metrik lintas ambang, dan ditutup dengan uji statistik.

2.6.1 *Confusion Matrix*

Seluruh metrik berbasis ambang diturunkan dari empat komponen *confusion matrix*, yaitu *True Positive* (TP), *False Positive* (FP), *True Negative* (TN), dan *False Negative* (FN), sebagaimana dirangkum pada Tabel 2.2.

Tabel 2.2. *Confusion Matrix* pada Deteksi *Fraud*

	Prediksi *Fraud*	**Prediksi Normal**
Aktual *Fraud*	TP	FN
Aktual Normal	FP	TN

Pada konteks deteksi *fraud*, keempat komponen ini memiliki makna operasional yang berbeda. TP adalah *fraud* yang berhasil dideteksi, FN adalah *fraud* yang lolos dan berujung pada kerugian finansial langsung, FP adalah transaksi normal yang salah ditandai sehingga menimbulkan pemeriksaan yang sia-sia, dan TN adalah transaksi normal yang benar dilewatkan. Karena TN berjumlah sangat besar pada data yang timpang, metrik yang melibatkan TN pada penyebutnya cenderung memberikan gambaran yang terlalu optimistik, dan hal ini menjadi dasar pemilihan metrik pada subbab-subbab berikut.

2.6.2 *Accuracy*

Accuracy merupakan proporsi prediksi yang benar terhadap seluruh prediksi:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \tag{2.19}$$

Pada data dengan *class imbalance* ekstrem, metrik ini tidak informatif. Sebagai contoh, model yang memprediksi seluruh transaksi sebagai normal pada *dataset* dengan tingkat *fraud* 0,5% menghasilkan $TP = 0$ dan $FP = 0$, namun tetap mencapai *accuracy* 99,5% karena suku TN mendominasi Persamaan 2.19. *Accuracy* karena itu tetap dilaporkan pada penelitian ini, tetapi semata-mata agar hasil dapat dibandingkan dengan literatur yang melaporkannya, bukan sebagai dasar penilaian.

2.6.3 *Precision*

Precision merupakan rasio jumlah transaksi yang benar-benar *fraud* di antara seluruh transaksi yang diprediksi sebagai *fraud*:

$$\text{Precision} = \frac{TP}{TP+FP} \tag{2.20}$$

Semakin tinggi *precision*, semakin sedikit transaksi normal yang salah diselidiki. Pada konteks operasional, *precision* yang tinggi berarti sumber daya pemeriksaan digunakan secara efisien.

2.6.4 *Recall*

Recall (atau *sensitivity*) merupakan rasio jumlah transaksi *fraud* yang berhasil dideteksi terhadap seluruh transaksi *fraud* yang sebenarnya:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.21}$$

Semakin tinggi *recall*, semakin sedikit *fraud* yang lolos. Pada konteks deteksi *fraud* kartu kredit, *recall* yang tinggi sangat diinginkan karena kehilangan transaksi *fraud* mengakibatkan kerugian finansial langsung. Perlu dicatat bahwa baik *precision* maupun *recall* tidak melibatkan TN sama sekali, sehingga keduanya tidak terpengaruh oleh besarnya kelas mayoritas dan lebih sesuai untuk data yang timpang dibanding *accuracy*.

2.6.5 F1-*Score* dan Fβ-*Score*

Precision dan *recall* bergerak berlawanan arah: menurunkan ambang keputusan menaikkan *recall* tetapi menurunkan *precision*, dan sebaliknya. Karena itu keduanya perlu dirangkum menjadi satu ukuran tunggal agar model dapat diperbandingkan. F1-*score* merupakan rata-rata harmonik dari keduanya:

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.22}$$

Rata-rata harmonik dipilih, bukan rata-rata aritmetik, karena nilainya mendekati nilai terkecil di antara kedua komponen. Akibatnya, model yang unggul pada satu sisi tetapi sangat lemah pada sisi lain tidak akan memperoleh F1 yang tinggi. F1-*score* digunakan sebagai metrik berbasis ambang utama pada penelitian ini karena membobot *precision* dan *recall* secara setara, sehingga tidak mengandaikan prioritas operasional tertentu dan dapat dibandingkan langsung dengan literatur terkait yang melaporkannya [15, 16, 4].

Fβ-*score* merupakan generalisasi dari F1-*score* yang memungkinkan pembobotan asimetris antara *precision* dan *recall*:

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{Precision} \cdot \text{Recall}}{\beta^2 \cdot \text{Precision} + \text{Recall}} \tag{2.23}$$

Parameter β menentukan bobot relatif *recall* terhadap *precision*: nilai $\beta > 1$ memberi bobot lebih besar pada *recall*, sedangkan $\beta < 1$ sebaliknya. Untuk $\beta = 1$, Persamaan 2.23 tereduksi menjadi Persamaan 2.22.

Kasus $\beta = 2$ relevan pada deteksi *fraud* kartu kredit karena kehilangan transaksi *fraud* (*false negative*) umumnya memiliki dampak finansial dan reputasional yang lebih besar dibandingkan pemeriksaan transaksi normal secara keliru (*false positive*) [27]. Substitusi $\beta = 2$ pada Persamaan 2.23 menghasilkan

$$F_2 = 5 \cdot \frac{\text{Precision} \cdot \text{Recall}}{4 \cdot \text{Precision} + \text{Recall}} \tag{2.24}$$

yang memberikan bobot *recall* sebesar empat kali bobot *precision* [27]. Pada penelitian ini, F2-*score* dilaporkan sebagai pelengkap F1-*score* untuk menunjukkan bagaimana perbandingan antar konfigurasi berubah ketika *recall* dibobot lebih besar, bukan sebagai metrik utama. Melaporkan keduanya sekaligus memungkinkan pembaca menilai apakah keunggulan suatu konfigurasi bertahan pada pembobotan yang berbeda atau justru bergantung pada pilihan β tertentu.

2.6.6 *Matthews Correlation Coefficient*

Matthews Correlation Coefficient (MCC) merupakan metrik yang memperhitungkan keempat komponen *confusion matrix* sekaligus, sehingga tidak dapat dinaikkan hanya dengan memprediksi kelas mayoritas [3]. Metrik ini didefinisikan sebagai:

$$\text{MCC} = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{2.25}$$

Nilai MCC berada pada rentang $[-1, 1]$, dengan 1 menyatakan prediksi yang sepenuhnya benar, 0 menyatakan performa setara tebakan acak, dan -1 menyatakan prediksi yang sepenuhnya terbalik. Keunggulan MCC pada data timpang terletak pada struktur pembilangnya: model yang memprediksi seluruh transaksi sebagai normal menghasilkan $TP = 0$ dan $FP = 0$, sehingga pembilang menjadi nol dan MCC bernilai nol. Model yang sama memperoleh *accuracy* di atas 99% melalui Persamaan 2.19. Perbandingan kedua metrik pada kasus yang sama inilah yang menjadikan MCC pelengkap penting bagi F1-*score*.

2.6.7 *Area Under Curve*: ROC-AUC dan PR-AUC

Seluruh metrik di atas menilai performa pada satu ambang keputusan tertentu, sehingga hasilnya bergantung pada pemilihan ambang tersebut. Untuk menilai kualitas skor yang dihasilkan model secara menyeluruh, tanpa terikat pada satu ambang, digunakan metrik berbasis luas daerah di bawah kurva.

Kurva *Receiver Operating Characteristic* (ROC) memplot *true positive rate* terhadap *false positive rate* pada seluruh nilai ambang, dengan

$$\text{TPR} = \frac{TP}{TP+FN}, \qquad \text{FPR} = \frac{FP}{FP+TN} \tag{2.26}$$

dan ROC-AUC merupakan luas daerah di bawah kurva tersebut. Pada kondisi *class imbalance* ekstrem, ROC-AUC memiliki keterbatasan yang penting: penyebut FPR pada Persamaan 2.26 didominasi oleh TN yang berjumlah sangat besar, sehingga penambahan *false positive* dalam jumlah besar sekalipun hanya menggeser FPR sedikit. Akibatnya, ROC-AUC cenderung memberikan nilai tinggi bahkan pada model yang secara praktis menghasilkan banyak alarm palsu [3].

Kurva *Precision-Recall* tidak memiliki keterbatasan tersebut karena tidak melibatkan TN sama sekali. Kurva ini memplot *precision* (Persamaan 2.20) terhadap *recall* (Persamaan 2.21) pada seluruh nilai ambang, dan PR-AUC merupakan luas daerah di bawahnya:

$$\text{PR-AUC} = \int_0^1 \text{Precision}(r)\,\mathrm{d}r \tag{2.27}$$

dengan r menyatakan *recall*. Sebagai pembanding, model tanpa kemampuan membedakan kelas menghasilkan PR-AUC yang setara dengan proporsi kelas positif pada data, sehingga pada tingkat *fraud* 0,521% nilai dasarnya adalah 0,00521, jauh di bawah nilai dasar ROC-AUC yang sebesar 0,5. Rentang nilai yang jauh lebih lebar inilah yang membuat PR-AUC lebih diskriminatif, sehingga metrik ini dijadikan acuan utama untuk membandingkan kualitas skor antar konfigurasi pada penelitian ini, sementara ROC-AUC tetap dilaporkan agar hasil dapat dibandingkan dengan literatur yang melaporkannya.

2.6.8 *Cumulative Gain*

Metrik berbasis luas kurva merangkum kualitas skor menjadi satu angka. Untuk memeriksa di bagian mana dari rentang skor keunggulan suatu model sebenarnya terletak, kualitas pengurutan dapat diperiksa secara langsung melalui kurva *cumulative gain*. Seluruh transaksi diurutkan menurun berdasarkan skor risiko, kemudian diplot proporsi *fraud* yang berhasil tertangkap terhadap proporsi transaksi yang ditinjau [26]. Untuk suatu proporsi transaksi teratas $\phi \in [0,1]$, nilai *gain* didefinisikan sebagai:

$$G(\phi) = \frac{TP_\phi}{N_{\text{fraud}}} \tag{2.28}$$

dengan TP_ϕ merupakan jumlah *fraud* yang berada dalam proporsi ϕ transaksi teratas dan N_{fraud} merupakan total *fraud* pada data. Kurva yang naik curam pada wilayah kiri, yaitu proporsi transaksi yang kecil, menandakan *fraud* terkonsentrasi pada skor tertinggi, sehingga model efektif memisahkan transaksi berisiko tinggi dari sisanya [26]. Sebagai pembanding, model tanpa kemampuan membedakan kelas menghasilkan garis diagonal, yaitu proporsi *fraud* yang tertangkap sebanding dengan proporsi transaksi yang ditinjau. Semakin jauh kurva berada di atas garis diagonal ini, semakin baik kemampuan model memberikan skor tinggi pada transaksi *fraud*.

2.6.9 Uji Statistik *Wilcoxon Signed-Rank*

Uji *Wilcoxon signed-rank* merupakan uji statistik non-parametrik untuk membandingkan dua himpunan hasil berpasangan ketika asumsi distribusi normal tidak terpenuhi atau ukuran sampel kecil [28]. Pada perbandingan dua metode *machine learning* yang dievaluasi pada beberapa *seed* pelatihan, uji ini sesuai karena tiga alasan. Pertama, pasangan nilai metrik pada *seed* yang sama bersifat berpasangan; kedua, selisih performa antar *seed* tidak dijamin berdistribusi normal; ketiga, jumlah *seed* umumnya kecil sehingga uji parametrik seperti *paired t-test* kurang andal. Demšar [28] merekomendasikan uji *Wilcoxon signed-rank* sebagai uji yang aman dan *robust* untuk membandingkan dua model *machine learning*.

Uji ini bekerja dengan menghitung peringkat dari selisih performa antar pasangan, kemudian membandingkan jumlah peringkat selisih positif dengan jumlah peringkat selisih negatif. Statistik uji diperoleh dari perbandingan tersebut,

dan *p-value* dihitung untuk menilai signifikansi statistik dari perbedaan performa antara kedua metode. *P-value* yang lebih kecil dari ambang α (umumnya 0,05) menunjukkan bahwa perbedaan tersebut signifikan secara statistik.