BAB II
KAJIAN LITERATUR

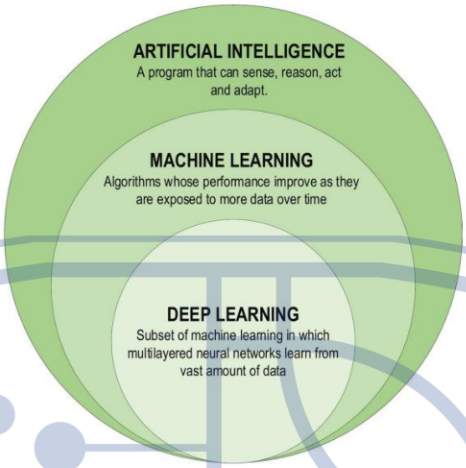
2.1 *Artificial Intelligence*

Artificial Intelligence (AI) merupakan bidang ilmu komputer yang berfokus pada pembuatan sistem dalam menyelesaikan tugas-tugas yang membutuhkan kecerdasan manusia. AI menggunakan model komputasi untuk meniru kemampuan kognitif manusia seperti belajar, menalar, memecahkan masalah, dan membuat keputusan dalam menyelesaikan tugasnya [16]. AI berperan dalam meningkatkan kualitas pengambilan keputusan di dalam sektor kehidupan, seperti di pemerintahan maupun industri, termasuk layanan kesehatan, pendidikan, transportasi, dan teknologi digital. Keunggulan AI terletak pada kemampuannya menganalisis data dalam jumlah besar secara cepat dan menghasilkan keputusan yang lebih efisien [17].

Artificial Intelligence dalam pengembangannya memiliki beberapa komponen utama yang memiliki fokus berbeda pada kategori kemampuan sistem untuk meniru kecerdasan manusia. Salah satu komponen yang paling berkembang adalah *Machine Learning* yang fokus pada pengembangan sistem untuk belajar dari data dan meningkatkan kinerjanya seiring waktu tanpa pemrograman yang eksplisit melalui pengenalan pola. Di dalam *Machine Learning* terdapat *Deep Learning*, yang memanfaatkan banyak lapisan jaringan saraf tiruan untuk memproses data kompleks, seperti pengenalan gambar dan suara. Selain itu, terdapat *Natural Language Processing (NLP)* yang memungkinkan sistem untuk memahami dan merespons bahasa manusia, sehingga banyak digunakan untuk sistem interaksi seperti pada *chatbot*, penerjemah otomatis, dan analisis sentimen. *Computer Vision* memungkinkan sistem menganalisis informasi visual seperti gambar dan video, yang banyak diterapkan pada sistem pengawasan, kendaraan otonom, dan analisis citra medis. Dalam bidang *Robotics*, teknologi AI diimplementasikan pada robot untuk melakukan tugas secara otomatis atau semi-otomatis dalam sektor seperti manufaktur, kesehatan, dan eksplorasi. Selain itu, *Expert Systems* memanfaatkan basis pengetahuan dan aturan tertentu untuk meniru cabang keahlian manusia dalam memberikan solusi pada bidang keahlian tertentu seperti diagnosis medis dan perencanaan keuangan. Berbagai cabang tersebut menunjukkan bahwa *Artificial Intelligence* berkembang menjadi bidang multidisiplin yang digunakan secara luas dalam berbagai sistem teknologi modern [18].

Perkembangan Artificial Intelligence (AI) mengalami kemajuan yang sangat pesat dalam beberapa tahun terakhir. AI menjadi salah satu teknologi kunci yang mendukung

transformasi digital dan berperan penting dalam mewujudkan Industry 4.0 melalui pengembangan sistem cerdas, peningkatan efisiensi, serta inovasi di berbagai sektor industry [19].



Gambar 2.1 Struktur Hierarki *Artificial Intelligence [20]*

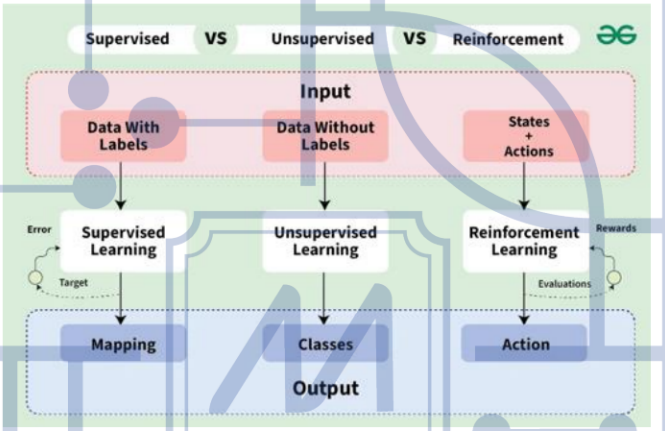
2.1.1 *Machine Learning*

Machine Learning adalah cabang dari *Artificial Intelligence (AI)* yang berfokus pada kemampuan sistem komputer untuk belajar dari data tanpa perlu diprogram di setiap tugas. *Machine Learning* memanfaatkan model matematis dalam mengenali pola data yang digunakan untuk membuat prediksi atau keputusan secara otomatis pada data baru [16].

Dalam proses pengembangan model *machine learning*, dataset dibagi menjadi *training set* dan *testing set*. *Training set* digunakan untuk melatih model agar mempelajari pola pada data, sedangkan *testing set* digunakan untuk mengevaluasi kinerja model terhadap data yang belum pernah dilihat sebelumnya [21]. Pada tahap pelatihan *(training)*, data dimasukkan ke dalam algoritma sehingga model dapat mempelajari hubungan antara *input* dan *output*. Dalam prosesnya, terdapat algoritma yang melakukan optimasi untuk menemukan parameter model terbaik dengan meminimalkan fungsi kerugian *(loss function)* untuk mengurangi kesalahan prediksi [22]. Setelah pelatihan selesai, model diuji menggunakan data pengujian *(testing set)* dan dievaluasi menggunakan metrik tertentu seperti *accuracy, precision, recall, dan F1-score* untuk mengukur performa model. Model dengan performa yang baik digunakan dalam melakukan prediksi terhadap data baru yang belum pernah dilihat sebelumnya [23].

Machine Learning dapat diklasifikasikan ke dalam beberapa kategori berdasarkan cara model mempelajari data:

1. Supervised Learning adalah metode yang menggunakan data berlabel, sehingga model dapat mempelajari hubungan input dan output. Metode ini umum digunakan pada tugas klasifikasi dan regresi [23].
2. *Unsupervised Learning* memanfaatkan data tanpa label untuk menemukan pola atau struktur tersembunyi dalam data, seperti pengelompokan data (clustering) atau reduksi dimensi [24].
3. *Reinforcement Learning* merupakan metode pembelajaran berbasis interaksi antara agen dan lingkungan, di mana model belajar menentukan tindakan terbaik berdasarkan umpan balik berupa *reward* atau *penalty* [25].



Gambar 2.2 *Supervised Learning, Unsupervised Learning & Reinforcement Learning* [26]

2.1.2 *Deep Learning*

Deep Learning (DL) merupakan cabang dari *Machine Learning* yang memanfaatkan jaringan saraf tiruan berlapis *(multi-layer neural networks)* untuk mempelajari representasi data secara hierarkis, dari tingkatan yang paling sederhana hingga yang paling abstrak. Pendekatan ini memungkinkan model mengenali pola dari data mentah pada berbagai tingkat abstraksi [27]. Melalui banyak lapisan pemrosesan, model ini mampu mengekstraksi fitur penting secara otomatis dari data tanpa perancangan fitur secara manual. Kemampuan tersebut menjadikan model ini unggul dalam memahami pola kompleks pada input seperti teks, gambar, dan suara [28].

Dalam *Deep Learning*, lapisan tersembunyi *(hidden layer)* berperan dalam membentuk representasi data yang semakin kompleks dari lapisan sebelumnya. Pada pengenalan gambar, lapisan awal dapat mendeteksi pola sederhana seperti tepi atau sudut. Informasi tersebut diproses pada lapisan berikut dengan menggabungkannya menjadi bentuk yang lebih kompleks, pada lapisan akhir, sistem mampu mengenali objek secara

keseluruhan. Pembelajaran berlapis ini membuat model mampu menangkap hubungan non-linear dalam data [20].

Beberapa arsitektur *Deep Learning* yang umum digunakan antara lain:

1. *Artificial Neural Network (ANN)* merupakan arsitektur dasar dari jaringan saraf tiruan yang terdiri dari lapisan masukan, lapisan tersembunyi, dan lapisan keluaran. Setiap *neuron* memproses input dengan bobot tertentu, dan menghasilkan output melalui fungsi aktivasi. ANN menjadi dasar bagi pengembangan arsitektur *Deep Learning* modern [28].
2. *Convolutional Neural Network (CNN)* digunakan untuk mengolah data berbentuk *grid* seperti gambar dan video. Arsitektur ini memanfaatkan operasi konvolusi untuk mengekstraksi fitur spasial secara bertahap, sehingga efektif untuk tugas pengenalan gambar, deteksi objek, dan segmentasi [29].
3. *Recurrent Neural Network (RNN)* digunakan untuk memproses data sekuensial dengan mempertahankan informasi dari langkah sebelumnya melalui koneksi berulang, arsitektur ini banyak digunakan dalam tugas pemrosesan bahasa alami dan prediksi deret waktu *(time series)* [20]
4. *Long Short-Term Memory (LSTM)* merupakan pengembangan dari RNN untuk mengatasi masalah vanishing gradient. Dengan menggunakan mekanisme *gate* seperti *input gate, forget gate, output gate* untuk mengontrol aliran informasi di dalam sel memori [20].
5. *Gated Recurrent Unit (GRU)* adalah varian *RNN* yang lebih sederhana dibandingkan LSTM karena hanya menggunakan dua gate yaitu reset gate dan update gate. *GRU* menjadi alternatif *LSTM* karena memiliki kompleksitas komputasi yang lebih rendah dengan performa sebanding [30].

2.1.3 *Natural Language Processing (NLP)*

Natural Language Processing (NLP) merupakan bidang dari *Artificial Intelligence* yang berfokus pada interaksi antara komputer dan bahasa manusia. Bidang ini mempelajari bagaimana komputer dapat memproses dan menganalisis data bahasa dalam jumlah besar. NLP mengembangkan sistem yang dapat memahami isi dokumen secara akurat, termasuk makna berdasarkan konteks bahasa. Oleh karena itu, sistem dapat mengekstraksi informasi dari teks secara otomatis [20].

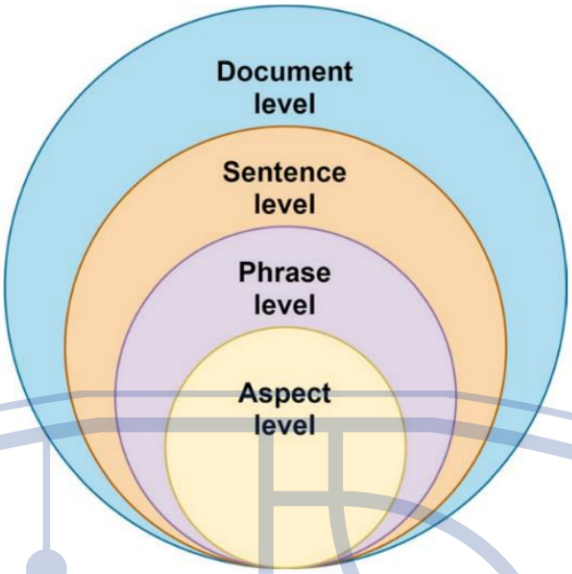
NLP telah diterapkan di berbagai bidang yang melibatkan pengolahan data teks. Dalam bidang kesehatan, NLP menganalisis literatur medis, catatan pasien, dan data klinis untuk mendukung pengambilan keputusan medis. Di bidang keuangan, NLP digunakan untuk menganalisis laporan keuangan, sentimen pasar, dan berita media dalam mendukung pengambilan keputusan investasi dan manajemen risiko. Selain itu, NLP juga diterapkan pada mesin pencari, sistem terjemahan otomatis, asisten *virtual*, dan analisis teks media sosial [20].

NLP menggabungkan konsep linguistik dan pembelajaran mesin. Dari sisi linguistik, NLP memanfaatkan pemahaman struktur bahasa seperti morfologi, sintaks, dan semantik untuk mengurai makna teks. Dari sisi pembelajaran mesin, NLP menggunakan pendekatan berbasis data di mana model dilatih menggunakan korpus teks berskala besar untuk mengenali pola bahasa secara otomatis [31]

Sistem NLP umumnya diproses melalui beberapa tahapan dalam *pipeline* pemrosesan teks. Tahap pertama adalah *preprocessing*, yaitu proses membersihkan dan menyeragamkan teks mentah agar lebih mudah diproses, misalnya melalui *lowercasing*, penghapusan tanda baca, dan *stopword removal*. Tahap kedua adalah *tokenization*, yaitu proses memecah teks menjadi unit-unit kecil yang disebut *token*, yang akan digunakan sebagai satuan dasar pemrosesan. Tahap ketiga adalah *embedding*, yaitu proses mengubah *token* menjadi vektor numerik agar dapat diproses oleh model pembelajaran mesin, sehingga hubungan antar kata dapat direpresentasikan secara matematis [31].

2.1.4 Analisis Sentimen

Analisis Sentimen, yang sering dikenal sebagai *Opinion Mining* merupakan salah satu bidang dalam *Natural Language Processing (NLP)* yang bertujuan untuk mengidentifikasi dan mengekstraksi opini dari teks. Analisis Sentimen digunakan untuk menganalisis opini, pemikiran, dan kesan terhadap berbagai topik, produk dan layanan yang terdapat dalam teks. Informasi hasil analisis sentimen dapat dimanfaatkan oleh berbagai pihak sebagai dasar dalam membuat keputusan berdasarkan opini atau teks [32].



Gambar 2.3 Tingkatan Pada Analisis Sentimen [28]

Analisis Sentimen memiliki berbagai tingkatan yaitu:

1. *Document Level Sentiment Analysis* dilakukan pada keseluruhan dokumen dengan memberikan satu polaritas pada dokumen tersebut. Pada tingkat ini pendekatan *supervised learning* maupun *unsupervised learning* dapat digunakan untuk mengklasifikasikan dokumen [32].
2. *Sentence Level Sentiment Analysis* dilakukan dengan menganalisis setiap kalimat dan menentukan polaritasnya. Pendekatan ini sangat berguna untuk dokumen yang memiliki beragam polaritas di dalamnya. Berbeda dengan *Document Level Sentiment Analysis* yang memberikan satu polaritas pada satu dokumen, pendekatan ini menentukan polaritasnya secara independen. Metode yang digunakan umumnya sama, namun membutuhkan data pelatihan dan sumber daya yang lebih besar [32].
3. *Phrase Level Sentiment Analysis* dilakukan dengan mengekstraksi opini pada tingkat frasa yang akan diklasifikasikan. Setiap frasa dapat merepresentasikan satu atau beberapa aspek, sehingga efektif untuk meninjau ulasan produk yang terdiri dari beberapa baris kalimat [32].
4. *Aspect Level Sentiment Analysis* berfokus pada identifikasi aspek-aspek dalam suatu kalimat, kemudian menentukan polaritas untuk setiap aspek yang dilanjutkan dengan perhitungan agregasi sentimen untuk memperoleh sentimen keseluruhan dari kalimat [32].

2.2 *Artificial Neural Network (ANN)*

Artificial Neural Network (ANN) atau jaringan saraf tiruan merupakan model komputasi yang meniru cara kerja jaringan saraf biologis pada otak manusia. ANN tersusun dari unit komputasi yang disebut *neuron* buatan. Setiap *neuron* saling terhubung melalui bobot untuk memproses dan meneruskan informasi antar lapisan [33]. ANN merupakan dasar perkembangan *deep learning* modern dimana arsitektur *deep learning* merupakan pengembangan ANN dengan jumlah lapisan yang lebih dalam. Hal ini membuat model mampu memetakan *input* ke *output* yang diharapkan. Kemampuan belajar dari data menjadikan ANN sebagai salah satu komponen inti dalam berbagai sistem *Artificial Intelligence* saat ini [28].

ANN tersusun dari tiga komponen lapisan utama. Pertama, *input* layer yang menerima data masukan mentah dan meneruskannya ke lapisan berikutnya tanpa transformasi. Kedua, satu atau lebih *hidden layer* yang berfungsi sebagai lapisan perantara tempat proses pembelajaran terjadi. Pada lapisan ini, setiap *neuron* melakukan penjumlahan berbobot dari keluaran lapisan sebelumnya, kemudian menerapkan fungsi aktivasi untuk menghasilkan sinyal yang diteruskan ke lapisan selanjutnya. Ketiga, *output layer* yang menghasilkan prediksi atau klasifikasi akhir sesuai dengan tugas yang diselesaikan [27].

2.2.1 *Recurrent Neural Network (RNN)*

Recurrent Neural Network (RNN) merupakan jenis jaringan saraf tiruan yang digunakan untuk memproses data sekuensial, yaitu data yang memiliki urutan bermakna. RNN memiliki koneksi berulang *(recurrent connections)* yang memungkinkan keluaran dari satu *timestep* diteruskan kembali sebagai masukan pada langkah berikutnya. Mekanisme ini membuat RNN memiliki memori internal yang disebut *hidden state*, yang akan terus diperbarui di setiap *timestep* berdasarkan masukan saat ini dan kondisi sebelumnya [34].

RNN terdiri dari beberapa komponen, yaitu *hidden state* sebagai memori jaringan, lapisan input yang menerima data pada setiap *timestep*, dan lapisan output yang menghasilkan prediksi. RNN mampu menangkap ketergantungan temporal dan konteks dalam urutan data, sehingga model ini cocok digunakan untuk tugas pemrosesan bahasa alami, penerjemahan mesin, pengenalan suara, dan prediksi deret waktu *(time series)* [35].

Berdasarkan arsitekturnya, RNN memiliki beberapa varian. *Vanilla RNN* merupakan bentuk dasar satu lapisan berulang sederhana. Namun, model ini rentan terhadap masalah vanishing dan exploding gradient pada urutan data yang panjang. *Bidirectional RNN (BiRNN)* memproses data sekuensial dari dua arah, yaitu maju *(forward)* dan mundur *(backward)*, sehingga mampu memanfaatkan konteks masa lalu dan masa depan secara

bersamaan. *Deep RNN* menggunakan beberapa lapisan berulang yang disusun secara bertumpuk untuk mempelajari representasi data yang lebih kompleks [34].

RNN standar memiliki keterbatasan berupa masalah vanishing gradient. Kondisi ini terjadi ketika nilai gradien menjadi sangat kecil saat dipropagasikan mundur *(backpropagation)* melalui banyak *timestep*. Akibatnya, model kesulitan mempelajari ketergantungan jangka panjang pada data sekuensial. Keterbatasan ini mendorong pengembangan varian RNN yang lebih canggih, seperti *Long Short-Term Memory (LSTM)* dan *Gated Recurrent Unit (GRU)* [35].

2.2.2 *Gated Recurrent Unit (GRU)*

Gated Recurrent Unit (GRU) adalah pengembangan dari *Recurrent Neural Network (RNN)* sama seperti *Long Short-Term Memory (LSTM),* GRU dirancang untuk mengatasi permasalahan *vanishing gradient* dengan arsitektur yang lebih sederhana dibandingkan LSTM. GRU menyederhanakan arsitektur dengan cara menggabungkan dua gerbang dari LSTM yaitu *forget gate* dan *input gate* menjadi *update gate* [36].

Komponen yang berper an penting dalam operasional GRU adalah *hidden state*. *Hidden state* adalah representasi memori internal jaringan yang menyimpan informasi dari *timestep* sebelumnya untuk digunakan dalam memproses *input* saat ini. Dalam konteks RNN dan variannya, *hidden state* berfungsi sebagai akumulasi pengetahuan yang terus diperbarui seiring berjalannya urutan data [36].

Poin terpenting bagaimana GRU bekerja terletak pada pemanfaatan *hidden state* melalui dua gerbang utama, yaitu *update gate* dan *reset gate* [36]. *Update gate* mengontrol sejauh mana informasi pada *hidden state* sebelumnya dipertahankan dan dikombinasikan dengan informasi baru, sedangkan *reset gate* menentukan bagian informasi dari *hidden state* sebelumnya yang perlu dipertahankan atau diabaikan sebelum digunakan dalam pembentukan *hidden state* baru [37].

Secara teknis gerbang-gerbang ini adalah dua vektor berbeda yang menentukan informasi mana yang harus diteruskan dan diproses ke *output*. Gerbang-gerbang ini dapat dilatih untuk mempertahankan informasi penting dalam jangka waktu lama, tanpa membuangnya seiring waktu atau membuang informasi yang tidak relevan dan tidak signifikan untuk prediksi [38]. Langkah-langkah inti dalam satu set GRU berupa [39]

1. *Reset Gate*

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t]) \quad \text{.....(1)}$$

2. *Update Gate*

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t]) \quad (2)$$

3. Candidate *Hidden State*

$$\tilde{h}_t = \tanh(W_{\tilde{h}} \cdot [r_t \odot h_{t-1}, x_t]) \quad (3)$$

4. *Updated Hidden State*

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \quad (4)$$

Keterangan:

r_t = *Reset Gate*

z_t = *Update Gate*

h_t = *Hidden State* saat ini

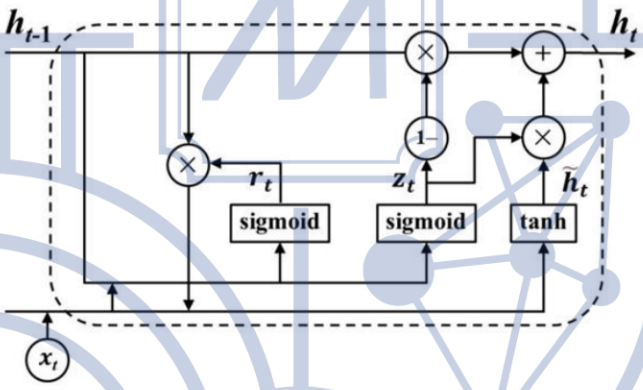
h_{t-1} = *Hidden State* sebelumnya

$\tilde{h}_t$= Kandidat *Hidden State*

W_r, W_z, W_h= Bobot

σ = fungsi *sigmoid* (0–1)

$\odot$ = perkalian elemen per elemen



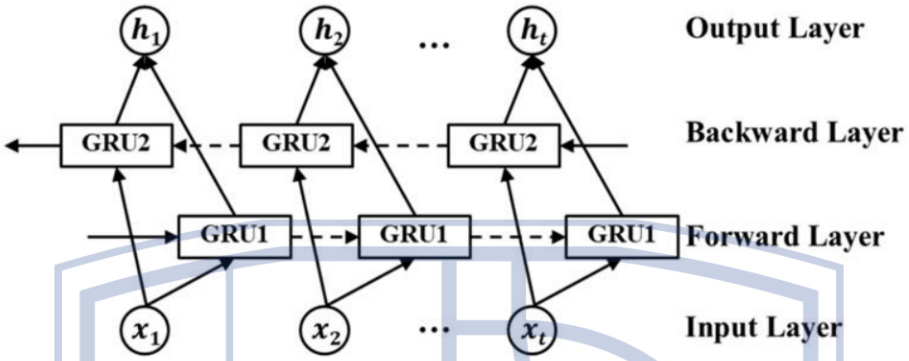
Gambar 2.4 Arsitektur GRU [39]

2.2.3 *Bidirectional Gated Recurrent Unit (Bi-GRU)*

Untuk mengatasi keterbatasan GRU dan menangkap pola temporal yang lebih komprehensif dari kedua arah, dikembangkan varian *Bidirectional Gated Recurrent Unit* (Bi-GRU). Bi-GRU memproses urutan data secara dua arah dengan menganalisis konteks maju (*past to future*) dan mundur (*future to past*) secara bersama-sama [40].

Model Bi-GRU menggunakan dua GRU *layer* yaitu *forward* GRU yang memproses data secara berurutan dari awal ke akhir (*timestep* $t = 1$ hingga $t = T$) *dan sebaliknya backward* GRU yang memproses data secara berurutan dari akhir ke awal (*timestep* $t = T$ hingga $t = 1$) untuk mengekstraksi informasi masa lalu dan masa depan. *Hidden State* yang dihasilkan dari *Forward* GRU (h_t^{fw}) dan *backward* GRU (h_t^{bw}) akan digabungkan untuk

menghasilkan informasi dari kedua arah [40]. Karakteristik tersebut memungkinkan struktur dua arah ini untuk membantu RNN dalam mengekstraksi lebih banyak informasi dan meningkatkan performa dari proses pembelajaran [41].



Gambar 2.5 Arsitektur Bi-GRU[39]

Proses ekstraksi informasi pada setiap *timestep* dilakukan dengan menghitung *hidden state* maju dan *hidden state* mundur secara independen sebagai berikut [39]:

1. *Forward Hidden State*

$$h_t^{fw} = GRU\left(x_t, h_{t-1}^{fw}\right) \quad (5)$$

2. *Backward Hidden State*

$$h_t^{bw} = GRU\left(x_t, h_{t-1}^{bw}\right) \quad (6)$$

3. *Final Output*

$$y_t = \left[h_t^{fw}; h_t^{bw}\right] \quad (7)$$

Keterangan :

h_t^{fw} = *Forward Hidden State*

h_t^{bw} = *Backward Hidden State*

h_{t-1}^{fw} = *Forward Hidden State* sebelumnya

h_t^{bw} = *Backward Hidden State* sebelumnya

x_t = *Input* sekarang (Token sekarang)

y_t = Gabungan *Forward Hidden State* dan *Backward Hidden State*

2.2.4 Glorot Uniform Initialization

Metode *Glorot Uniform Initialization* adalah metode inisialisasi bobot (*weight*) yang digunakan pada pelatihan jaringan saraf dalam (*deep neural network*) dalam mengatasi kesulitan pelatihan yang terjadi dimana nilai aktivasi dan gradien yang semakin mengecil

atau membesar ketika diteruskan melalui banyak lapisan yang dapat menyebabkan proses pembelajaran menjadi lambat dan menghambat konvergensi model [42]

Metode ini dirancang untuk menjaga kestabilan varians aktivasi dan gradien antar lapisan. Dengan distribusi bobot awal yang sesuai, informasi yang mengalir selama proses *forward propagation* maupun backpropagation dapat dipertahankan dengan lebih baik sehingga proses pelatihan menjadi lebih stabil dan efisien. Pengaruh pemilihan inisialisasi bobot yang tepat penting terhadap kemampuan model dalam mempelajari pola dari data. Sehingga, metode ini banyak digunakan sebagai teknik inisialisasi standar pada berbagai *framework deep learning* modern [42]

Pada Glorot Uniform Initialization, bobot diinisialisasi secara acak menggunakan distribusi uniform dengan batas nilai yang ditentukan berdasarkan jumlah neuron masukan n_{in} dan jumlah neuron keluaran n_{out}. Persamaan yang digunakan ditunjukkan pada Persamaan (8) [42]

$$W \sim U\left(-\sqrt{\frac{6}{n_{in}+n_{out}}}, \frac{6}{n_{in}+n_{out}}\right) \quad \text{.....(8)}$$

Keterangan:

W = bobot awal (*weight*)

n_{in} = jumlah neuron masukan (*input units*)

n_{out} = jumlah neuron keluaran (*output units*)

Berdasarkan persamaan (8), rentang nilai bobot akan menyesuaikan dengan ukuran layer yang digunakan. Semakin besar jumlah neuron pada suatu layer, maka rentang inisialisasi bobot akan semakin kecil. Pendekatan ini bertujuan untuk menjaga kestabilan distribusi aktivasi dan gradien selama proses pelatihan [42]

2.2.5 Fungsi Aktivasi

Fungsi aktivasi merupakan komponen penting dalam Artificial Neural Network (ANN) yang berperan dalam menentukan keluaran suatu neuron. Fungsi ini mentransformasikan hasil perhitungan linear yang terdiri dari kombinasi bobot, bias, dan masukan menjadi keluaran nonlinier. Kehadiran fungsi aktivasi memungkinkan jaringan saraf mempelajari pola yang kompleks dan memodelkan hubungan nonlinier dalam data. Tanpa fungsi aktivasi, jaringan saraf hanya akan menghasilkan transformasi linear meskipun memiliki banyak lapisan, sehingga kemampuan pembelajaran model menjadi terbatas [43].

Agar model mampu mengenali pola yang lebih kompleks, sejumlah *neuron* disusun ke dalam suatu lapisan. Lapisan-lapisan ini dihubungkan secara berurutan, dimulai dari

lapisan masukan *(input layer)*, dilanjutkan dengan satu atau lebih lapisan tersembunyi *(hidden layers)*, hingga lapisan keluaran *(output layer)*. Setiap lapisan menerima keluaran dari lapisan sebelumnya dan mengubahnya menjadi representasi baru menggunakan fungsi aktivasi nonlinier. Struktur berlapis tersebut memungkinkan jaringan saraf menangkap hubungan nonlinier dan mengekstraksi fitur secara bertahap [43].

Fungsi aktivasi memiliki peran penting dalam menentukan kemampuan jaringan dalam mempelajari pola data. Tanpa adanya fungsi aktivasi nonlinier, jaringan saraf hanya menghasilkan transformasi linear sehingga tidak mampu memodelkan hubungan kompleks dalam data [43]. Oleh karena itu, berbagai jenis fungsi aktivasi digunakan sesuai dengan kebutuhan dan karakteristik permasalahan. *ReLU (Rectified Linear Unit)* bernilai nol untuk input negatif dan mempertahankan nilai positif sehingga efisien dalam proses pelatihan. Sigmoid memetakan keluaran ke rentang 0 hingga 1 dan umum digunakan pada klasifikasi biner. Tanh menghasilkan keluaran dalam rentang −1 hingga 1 dan bersifat *zero-centered*. Sementara itu, *Softmax* mengubah sekumpulan nilai menjadi distribusi probabilitas yang berjumlah satu dan banyak digunakan pada klasifikasi multi-kelas [44].

Secara matematis, fungsi-fungsi aktivasi dasar tersebut dirumuskan sebagai berikut [44]:

1. *ReLU,* fungsi aktivasi yang mengubah semua nilai negatif menjadi nol.

$$f(x) = max(0, x) \quad \text{.....(9)}$$

2. *Sigmoid,* fungsi aktivasi yang memetakan output ke rentang antara 0 dan 1, sering digunakan untuk probabilitas.

$$f(x) = \frac{1}{1+e^{-x}} \quad \text{.....(10)}$$

3. *Tanh,* fungsi aktivasi yang memetakan output ke rentang antara −1 dan 1.

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \quad \text{.....(11)}$$

4. *Softmax,* fungsi aktivasi yang mengubah vektor nilai mentah menjadi distribusi probabilitas untuk klasifikasi multi-kelas.

$$\mathrm{S}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad \text{.....(12)}$$

Keterangan :

x = Nilai Input

$f(x)$ = Nilai Aktivasi

z_i = skor sebelum *Softmax*

e = Bilangan *Euler* (≈ 2,71828)

K = Jumlah kelas

i, j = Indeks kelas

$S(z_i)$ = Probabilitas hasil *Softmax*

Selain fungsi aktivasi, setiap neuron memiliki bias yang berfungsi menggeser ambang keputusan sehingga model menjadi lebih fleksibel dan tidak terbatas pada nilai nol. Ketika neuron pada suatu lapisan terhubung penuh dengan neuron pada lapisan berikutnya, terbentuklah fully connected layer atau dense layer. Pada lapisan ini, setiap neuron mengombinasikan seluruh keluaran dari lapisan sebelumnya dan menerapkan fungsi aktivasi untuk menghasilkan transformasi nonlinier yang lebih kompleks

$$z_l = Wl \times a_{l-1} + b_1 \quad (13)$$

Keterangan:

z_l = output linear pada layer ke-l (sebelum aktivasi)

Wl = matriks bobot pada layer ke-l

a_{l-1} = input dari layer sebelumnya (aktivasi layer ke-$l-1$)

b_1 = bias pada layer ke-l

2.2.6 Fungsi *Loss*

Dalam pelatihan model jaringan saraf, fungsi *loss* digunakan untuk mengukur perbedaan antara hasil prediksi model dan nilai target sebenarnya. Nilai loss merepresentasikan tingkat kesalahan model, di mana semakin kecil nilainya maka semakin baik performa model dalam melakukan prediksi. Selain sebagai indikator kesalahan, fungsi loss juga menjadi dasar dalam proses pembelajaran karena digunakan untuk menghitung gradien yang diperlukan dalam pembaruan *parameter* model [45].

Proses pembelajaran tersebut dilakukan melalui mekanisme *backpropagation*, yaitu proses perhitungan balik yang menyebarkan kesalahan dari lapisan output ke lapisan sebelumnya. Pada tahap ini, nilai loss digunakan untuk menghitung gradien terhadap setiap parameter, sehingga dapat diketahui kontribusi masing-masing bobot terhadap kesalahan yang terjadi. Gradien ini kemudian digunakan untuk memperbarui bobot dan bias secara iteratif agar model dapat meminimalkan kesalahan prediksi dan menghasilkan output yang lebih mendekati nilai aktual [29].

Pada kasus klasifikasi multi-kelas, fungsi loss yang umum digunakan adalah *Sparse Categorical Crossentropy (SCCE)*. Fungsi ini digunakan ketika label data berbentuk bilangan integer dan keluaran model direpresentasikan dalam bentuk distribusi probabilitas

melalui fungsi aktivasi *softmax*. *SCCE* bekerja dengan mengukur perbedaan antara probabilitas prediksi dan label sebenarnya dengan fokus pada probabilitas kelas yang benar. Secara matematis, fungsi *Sparse Categorical Crossentropy* dirumuskan sebagai berikut [45]:

$$L = -\frac{1}{N}\sum_{i=1}^{N} \log(pi, yi) \quad (14)$$

Keterangan:

L = nilai *loss* (tingkat kesalahan model)

N = jumlah data (sampel) yang digunakan

$\sum_{i=1}^{N}$ = penjumlahan dari data ke-1 sampai ke-N

i= indeks data

yi = label sebenarnya dari data ke-i

$p_{i,yi}$ = probabilitas prediksi pada kelas yang sesuai dengan label sebenarnya untuk data ke-i

log = fungsi logaritma natural

di mana N adalah jumlah data, yi merupakan label sebenarnya, dan $p_{i,yi}$ adalah probabilitas prediksi pada kelas yang sesuai dengan label tersebut. Nilai loss yang dihasilkan kemudian digunakan oleh algoritma optimisasi untuk memperbarui parameter model berdasarkan gradien yang dihitung melalui proses *backpropagation*, sehingga model dapat belajar secara bertahap dan meningkatkan performanya [46].

2.2.7 *Optimizer*

Optimizer merupakan komponen dalam proses pelatihan jaringan saraf yang berfungsi memperbarui parameter model (bobot dan bias) berdasarkan gradien yang diperoleh dari *backpropagation*. Tujuan utamanya adalah meminimalkan nilai fungsi *loss* sehingga prediksi model semakin mendekati nilai target. Salah satu algoritma optimisasi yang banyak digunakan adalah *Adam (Adaptive Moment Estimation)*, yang menggabungkan konsep momentum dan adaptive learning rate dalam satu pendekatan. Momentum membantu mempercepat konvergensi dengan mempertimbangkan arah gradien sebelumnya, sedangkan adaptive learning rate memungkinkan setiap parameter diperbarui dengan laju yang berbeda secara otomatis [47], [48].

Pada setiap iterasi pelatihan, *Adam* memperbarui parameter melalui beberapa tahapan. Pertama, dihitung gradien dari fungsi loss terhadap parameter model [48]:

$$gt = \nabla_{\theta} J(\theta_{t-1}) \quad (15)$$

Kemudian dihitung estimasi momen pertama (rata-rata gradien):

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) gt \quad (16)$$

dan estimasi momen kedua (varians gradien):

$$vt = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \quad (17)$$

Untuk mengurangi bias pada awal iterasi, dilakukan *bias correction*:

$$\widehat{m_t} = \frac{m_t}{1-\beta_1^t}, \widehat{v_t} = \frac{v_t}{1-\beta_2^t} \quad (18)$$

Selanjutnya, parameter diperbarui dengan:

$$\theta_t = \theta_{t-1} - \alpha \cdot \frac{\widehat{m_t}}{\sqrt{\widehat{v_t}}+\epsilon} \quad (19)$$

Keterangan:

gt = gradien pada iterasi ke-*t*, yaitu turunan fungsi loss terhadap parameter model

∇_θ = operator turunan (gradien) terhadap parameter θ

$J(\theta_{t-1})$ = nilai fungsi loss pada parameter sebelumnya

(θ_{t-1}) = *parameter* (bobot dan bias) pada iterasi sebelumnya

m_t = estimasi momen pertama (rata-rata *gradien / momentum*)

m_{t-1} = nilai momentum pada iterasi sebelumnya

β_1 = koefisien momentum (umumnya 0.9)

vt = estimasi momen kedua (*varians* gradien)

v_{t-1} = *varians* pada iterasi sebelumnya

β_2 = koefisien *varians* (umumnya 0.999)

g_t^2 = kuadrat gradien

$\widehat{m_t}$ = momen pertama setelah koreksi bias

$\widehat{v_t}$ = momen kedua setelah koreksi bias

t = iterasi ke-t

θ_t = parameter model yang telah diperbarui

α = *learning rate* (laju pembelajaran)

ϵ = konstanta kecil untuk menghindari pembagian dengan nol

Melalui mekanisme tersebut, *Adam* mampu menyesuaikan besar langkah pembaruan secara adaptif untuk setiap parameter, sehingga menghasilkan proses pelatihan yang lebih stabil dan cepat konvergen, terutama pada data dengan distribusi gradien yang kompleks [48].

2.3 GloVe

GloVe (Global Vectors for Word Representation) merupakan model *log-bilinear* yang digunakan untuk merepresentasikan kata ke dalam ruang vektor berdimensi rendah dengan memanfaatkan statistik *co-occurrence* global dari korpus. Model ini menggabungkan pendekatan global berbasis *matrix factorization* dengan metode *local context window* untuk mempelajari representasi kata yang mampu menangkap hubungan semantik dalam korpus teks [49].

Dalam *GloVe*, hubungan statistik antar kata direpresentasikan melalui sebuah matriks *co-occurrence* yang mencatat frekuensi kemunculan kata dalam jendela konteks kata lainnya. Statistik global kemunculan bersama kata digunakan untuk mempelajari pola hubungan semantik [49].

GloVe memanfaatkan rasio probabilitas kemunculan bersama antar kata terhadap kata konteks tertentu yang mengandung informasi semantik. Relasi semantik ini dapat direpresentasikan sebagai hubungan linear di dalam ruang vektor. Model ini mempelajari representasi vektor kata sehingga perkalian titik *(dot product)* antar vektor mendekati logaritma jumlah [49].

Fungsi objektif *GloVe* dirumuskan sebagai berikut:

$$J = \sum_{i,j=1}^{V} f(X_{ij}) \cdot (w_i \cdot w_j + b_i + b_j - \log X_{ij})^2 \quad \text{.....(20)}$$

Keterangan:

V = kata-kata unik di dalam korpus

X_{ij} = Jumlah kemunculan kata j dalam konteks kata i

w_i, w_j = Bobot

b_i, b_j = Nilai bias

GloVe digunakan pada berbagai tugas *Natural Language Processing* karena mampu menangkap hubungan semantik antar kata berdasarkan statistik *co-occurrence* dalam korpus teks, terutama untuk memahami keterkaitan makna secara mendalam. Dalam analisis sentimen, representasi ini membantu model mengenali nuansa emosi atau opini, seperti ulasan produk dan media sosial. GloVe juga bermanfaat dalam *machine translation* karena mampu memetakan kata dari bahasa sumber ke representasi yang seimbang secara semantik dalam bahasa target [49].

Di samping kelebihannya, *GloVe* juga memiliki beberapa keterbatasan. *GloVe* sebagai *static embedding* merepresentasikan setiap kata dengan satu vektor tetap sehingga kurang mampu menangkap perubahan makna berdasarkan konteks. *GloVe* juga memiliki kendala pada kata *OOV (out-of-vocabulary)* yaitu kata yang tidak terdapat dalam data

pelatihan, sehingga kurang efektif dalam merepresentasikan kata baru atau jarang muncul [49].

2.4 *Cryptocurrency*

Cryptocurrency adalah mata uang digital yang memanfaatkan kriptografi untuk mengamankan transaksi serta mengatur penciptaan unit baru. Berbeda dengan mata uang konvensional yang dikendalikan bank sentral, *cryptocurrency* beroperasi secara terdesentralisasi melalui jaringan komputer yang tersebar. Sistem ini memungkinkan pengguna melakukan transaksi secara langsung tanpa perantara seperti bank atau lembaga keuangan. Karakteristik tersebut menjadikan *cryptocurrency* sebagai inovasi dalam sistem keuangan digital yang menawarkan transaksi lebih transparan dan efisien [50].

Cryptocurrency memiliki beberapa karakteristik yang berbeda dari sistem keuangan tradisional. Pertama, bersifat desentralisasi karena tidak dikendalikan oleh satu otoritas pusat. Kedua, tingkat keamanan yang tinggi karena setiap transaksi dilindungi dengan kriptografi. Ketiga, transparan karena seluruh transaksi tercatat dan dapat diakses publik. Keempat, transaksi bersifat *irreversibel*, yaitu tidak dapat dibatalkan setelah dikonfirmasi. *Cryptocurrency* juga mendukung transaksi lintas negara dengan biaya yang lebih rendah dibandingkan sistem perbankan konvensional. Selain itu, cryptocurrency memiliki sifat *pseudonymous*, di mana identitas pengguna tidak ditampilkan langsung melainkan dalam bentuk alamat digital. Hal ini memberikan tingkat privasi, namun transaksi tetap dapat ditelusuri melalui sistem *blockchain* [51].

2.4.1 Perkembangan *Cryptocurrency*

Perkembangan cryptocurrency bergantung pada teknologi yang mendasarinya, yaitu *blockchain*. *Blockchain* adalah buku besar digital terdistribusi yang mencatat seluruh transaksi secara kronologis dalam blok-blok yang saling terhubung dan diamankan dengan kriptografi. Setiap transaksi diverifikasi oleh jaringan komputer dalam sistem *peer-to-peer* sebelum ditambahkan ke dalam blok baru. Sifat transparan, tidak dapat diubah *(immutable)*, dan terdesentralisasi membuat teknologi ini cocok untuk sistem keuangan digital tanpa bergantung pada otoritas pusat [52]. Blockchain menggunakan mekanisme konsensus seperti *Proof of Work (PoW)* atau *Proof of Stake (PoS)* untuk memverifikasi transaksi tanpa pihak ketiga sekaligus menjaga keamanan dan integritas jaringan [52].

Dalam perkembangannya, teknologi *blockchain* berkembang ke dalam bentuk aset digital yang dikenal sebagai *cryptocurrency* [53]. Salah satu *cryptocurrency* pertama yang

memanfaatkan teknologi ini adalah *Bitcoin*, yang menjadi titik awal berkembangnya ekosistem *cryptocurrency* secara global. Perkembangan *cryptocurrency* tidak hanya terbatas pada fungsi sebagai alat pembayaran, tetapi juga berkembang menjadi platform teknologi yang lebih kompleks [54]. Perkembangan *cryptocurrency* juga didorong kebutuhan akan sistem keuangan yang transparan, serta inovasi dalam bidang teknologi finansial *(fintech)* sehingga *cryptocurrency* mengalami evolusi baik dari segi teknologi [55].

2.4.2 *Bitcoin*

Bitcoin adalah mata uang digital yang termasuk dalam aset *cryptocurrency* yang memungkinkan transaksi *peer-to-peer* tanpa perantara melalui teknologi *blockchain*. *Bitcoin* merupakan mata uang digital yang paling dikenal dan berperan penting dalam perkembangan ekosistem *cryptocurrency* [54]. *Bitcoin* diperkenalkan pada tahun 2008 melalui *white paper* oleh Satoshi Nakamoto dan mulai digunakan secara publik pada tahun 2009 [56].

Dalam perkembangannya, *Bitcoin* tidak hanya digunakan sebagai alat pembayaran digital, tetapi juga menjadi instrumen investasi yang menarik perhatian investor di seluruh dunia. *Bitcoin* juga sering dianggap sebagai *store of value* atau penyimpan nilai seperti emas dalam sistem keuangan tradisional. Oleh karena itu, *Bitcoin* berperan sebagai indikator utama dalam pasar *cryptocurrency* yang sering dijadikan acuan karena pergerakan harganya memengaruhi aset kripto lainnya [55].

Seiring pertumbuhan ekosistem *cryptocurrency*, *Bitcoin* menjadi aset digital dengan kapitalisasi pasar terbesar. Namun, harga *Bitcoin* dikenal sangat volatil dibandingkan dengan aset keuangan tradisional. Fluktuasi harga dapat terjadi dalam waktu singkat dengan perubahan nilai yang cukup besar [57]. Perubahan harga *Bitcoin* dipengaruhi oleh berbagai faktor yang saling berkaitan. Faktor yang memengaruhi antara lain permintaan dan penawaran di pasar kripto, sentimen investor, perkembangan teknologi blockchain, regulasi pemerintah, serta kondisi ekonomi global. Perubahan volatilitas yang signifikan sering terjadi ketika terdapat pengumuman kebijakan ekonomi, khususnya kebijakan moneter dari negara-negara besar [58]. Berita dan informasi di media digital juga berpengaruh besar terhadap pergerakan harga *Bitcoin*. Sentimen positif maupun negatif di media berita dan media sosial dapat memicu perubahan harga secara cepat. Oleh karena itu, analisis sentimen menjadi pendekatan yang relevan untuk memahami kecenderungan sentimen pasar terhadap Bitcoin [59].

2.5 DSRM

Design Science Research Methodology (DSRM) merupakan paradigma penelitian dalam bidang *Information Systems* yang berfokus pada perancangan dan pengembangan artifact teknologi informasi sebagai solusi terhadap suatu permasalahan yang relevan. DSRM bertujuan memperluas batas kemampuan manusia dan organisasi melalui penciptaan artifact baru yang inovatif [60]. Artifact berupa model, metode, maupun sistem yang dirancang untuk memecahkan masalah tertentu. Pemahaman terhadap suatu permasalahan diperoleh melalui proses perancangan serta penerapan artifact tersebut secara langsung [60]

Model proses DSRM yang terdiri atas enam tahapan berurutan [60]. Tahapan tersebut adalah sebagai berikut:

1. *Problem Identification and Motivation*, mendefinisikan permasalahan penelitian secara spesifik sekaligus menetapkan urgensi penyelesaiannya sebagai landasan penelitian.
2. *Definition of the Objectives for a Solution*, menyimpulkan tujuan solusi yang akan dikembangkan berdasarkan rumusan masalah yang telah ditetapkan.
3. *Design and Development*, merancang dan membangun artifact sebagai wujud solusi dari permasalahan yang diidentifikasi.
4. *Demonstration*, menerapkan artifact pada satu atau lebih kasus nyata untuk membuktikan bahwa artifact tersebut mampu menyelesaikan masalah.
5. *Evaluation*, mengukur dan mengamati seberapa efektif artifact dalam menyelesaikan masalah menggunakan metrik yang relevan.
6. *Communication*, menyampaikan permasalahan, artifact yang dikembangkan, kegunaan, serta kontribusinya kepada khalayak ilmiah yang relevan [61].

DSRM biasa digunakan dalam penelitian yang berorientasi pada pengembangan sistem berbasis deep learning dan kecerdasan buatan, metodologi ini mendukung siklus build-and-evaluate yang iteratif [60]. Dengan menggunakan pendekatan ini, penelitian membangun artifact yang fungsional dan memastikan artifact yang dihasilkan dapat divalidasi secara ilmiah dan relevan secara praktis [61]

2.6 *Confusion Matrix*

Confusion matrix adalah metode evaluasi pada sistem klasifikasi yang digunakan untuk mengukur kinerja model dalam memprediksi data baru. Metode ini membandingkan label aktual dan hasil prediksi untuk menunjukkan tingkat keberhasilan serta kesalahan klasifikasi. Melalui *confusion matrix*, performa dapat dianalisis secara kuantitatif untuk menilai kemampuan generalisasi terhadap data baru. Evaluasi ini umumnya dilakukan menggunakan data pengujian yang terpisah dari data pelatihan [62].

Confusion matrix terdiri dari empat komponen utama, yaitu *True Positive (TP), True Negative (TN), False Positive (FP), dan False Negative (FN). True Positive (TP)* menunjukkan data positif yang berhasil diprediksi dengan benar, sedangkan *True Negative (TN)* merepresentasikan data negatif yang juga diprediksi dengan benar. *False Positive (FP)* terjadi ketika data negatif diklasifikasikan sebagai positif, sedangkan *False Negative (FN)* terjadi ketika data positif justru diprediksi sebagai negatif. Keempat komponen ini menjadi dasar dalam mengevaluasi performa model klasifikasi [63]

Tabel 2.1 Tabel *Confusion Matrix* untuk *Binary Classification* [63]

		Predicted	
		Negative	**Positive**
Actual	**Negative**	**TN**	**FP**
	Positive	**FN**	**TP**

Berdasarkan *confusion matrix*, terdapat beberapa metrik evaluasi seperti *accuracy, precision, recall, dan F1-score* yang digunakan untuk mengukur performa model. *Accuracy* untuk mengukur tingkat ketepatan model secara keseluruhan, yaitu seberapa banyak prediksi yang benar dibandingkan dengan seluruh data yang dievaluasi. Namun, *accuracy* memiliki keterbatasan pada dataset yang tidak seimbang, karena nilai yang tinggi tidak selalu mencerminkan kemampuan model dalam mendeteksi kelas minoritas. Oleh karena itu, digunakan metrik lain seperti *precision dan recall*. *Precision* mengukur ketepatan prediksi pada kelas positif, yaitu proporsi prediksi positif yang benar. Sementara itu, *recall* mengukur kemampuan model dalam mendeteksi seluruh data positif yang ada. Namun, kedua metrik ini sering kali memiliki hubungan yang berlawanan, sehingga diperlukan metrik gabungan yaitu *F1-score*, yang memberikan ukuran evaluasi yang lebih seimbang karena mempertimbangkan kedua aspek tersebut secara bersamaan. Secara matematis, metrik evaluasi yang digunakan dalam penelitian ini dirumuskan sebagai berikut [63]:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (21)$$

$$Precision = \frac{TP}{TP+FP} \quad (22)$$

$$Recall = \frac{TP}{TP+FN} \quad (23)$$

$$F_{1-Score} = 2 \times \frac{Precision \times Recall}{Precision+Recall} \quad (24)$$

Keterangan:

TP = *True Positive*

TN = *True Negative*

FP = *False Positive*

FN = *False Negative*

2.7 Ketidakseimbangan Data (Class Imbalance)

Class imbalance merupakan kondisi dataset di mana distribusi jumlah data antar kelas tidak merata, sehingga satu atau lebih kelas memiliki jumlah sampel yang jauh lebih sedikit (minority class) dibandingkan kelas lainnya (majority class) [11]. Kondisi ini merupakan tantangan yang sering dijumpai dalam berbagai tugas klasifikasi pada machine learning. Sebagian besar algoritma pembelajaran standar bekerja optimal dengan distribusi kelas yang seimbang, sehingga ketika dihadapkan pada data yang tidak seimbang, model cenderung menghasilkan bias terhadap kelas mayoritas dan mengabaikan kelas minoritas [11] Akibatnya, akurasi pada kelas minoritas justru rendah, sebuah kondisi yang secara langsung menurunkan kualitas prediksi pada kelas yang pada dasarnya penting untuk dikenali secara tepat [11].

Salah satu pendekatan yang efektif untuk mengatasi class imbalance adalah cost-sensitive learning, yaitu metode yang menerapkan bobot kesalahan klasifikasi (misclassification cost) yang berbeda untuk setiap kelas tanpa mengubah distribusi data secara langsung [64]. Pendekatan ini diimplementasikan melalui class weight proporsional yang dihitung berdasarkan distribusi kelas menggunakan persamaan berikut [64].

$$w_c = \frac{N_{total}}{C \times N_c} \quad \text{.....(25)}$$

Keterangan:

w_c = bobot untuk kelas ke C

N_{total} = Jumlah total data training

C = jumlah kelas dataset

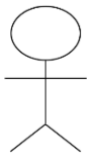
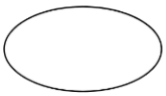
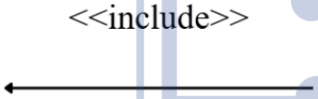
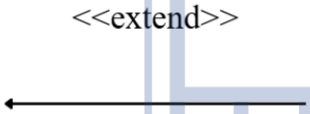
N_c = jumlah data yang termasuk dalam kelas ke C

2.8 *Use Case Diagram*

Use case diagram merupakan salah satu bentuk pemodelan yang digunakan dalam perancangan sistem informasi. Diagram ini menggambarkan hubungan atau interaksi antara satu atau lebih aktor dengan sistem yang dikembangkan. *Use case diagram* digunakan untuk mengidentifikasi fungsi-fungsi yang tersedia dalam suatu sistem serta menentukan pihak yang dapat mengakses dan menggunakan fungsi tersebut. Sehingga diagram ini merepresentasikan keterkaitan antara aktor dan sistem yang akan dibangun [65].

Tabel 2.2 Simbol-Simbol *Use Case Diagram* [66]

Simbol	Keterangan

	***Aktor***, mewakili peran orang, sistem yang lain, atau alat ketika berkomunikasi dengan Use Case.
	***Use Case***, abstraksi dari interaksi antara sistem dan aktor.
	Association, abstraksi dari penghubung antara aktor dengan Use Case.
	Generalisasi, menunjukkan spesialisasi aktor untuk dapat berpartisipasi dengan Use Case.
	***Include*****,** menunjukkan bahwa suatu Use Case seluruhnya merupakan fungsionalitas dari Use Case lainnya.
	***Extend*****,** Menunjukkan bahwa suatu Use Case merupakan tambahan fungsionalitas dari Use Case lainnya jika suatu kondisi terpenuhi.

Setiap *use case* biasanya dilengkapi dengan deskripsi dalam bentuk tabel. Deskripsi ini memuat informasi seperti nama use case, aktor yang terlibat, penjelasan singkat mengenai proses yang dilakukan, serta langkah-langkah yang menggambarkan alur penyelesaian secara normal [65].