

BAB II

KAJIAN LITERATUR

2.1 Pasar Modal dan Saham

2.1.1 Definisi Pasar Modal dan Saham

Pasar modal didefinisikan sebagai wadah yang menghubungkan pihak yang memerlukan modal (emiten) dengan investor, dengan tujuan utama menciptakan alokasi sumber daya yang efisien demi menunjang pertumbuhan ekonomi. Efek tersebut mencakup berbagai instrumen seperti saham, obligasi, dan reksa dana [14]. Saham merupakan bukti kepemilikan (ekuitas) dalam suatu perusahaan. Penerbitan saham melalui mekanisme Penawaran Umum Perdana (*Initial Public Offering/IPO*) menjadi proses yang penting untuk menunjukkan bagaimana saham dapat diperjualbelikan di pasar modal [14].

Klasifikasi saham berdasarkan likuiditas dan kapitalisasi pasar di Bursa Efek Indonesia (BEI) tercermin dalam berbagai indeks selektif, seperti IDX30 dan IDX80. IDX30 terdiri dari 30 saham dengan likuiditas dan kapitalisasi pasar tertinggi, telah menjadi objek penelitian yang signifikan. Sundoro, Putlia, dan Lie (2024) menganalisis determinan kinerja IDX30 dan menemukan bahwa inflasi, nilai tukar, dan suku bunga berpengaruh terhadap pergerakan indeks ini [14]. Selanjutnya, Lestari dan Aslama (2024) menemukan bahwa penggunaan indikator teknikal seperti RSI, MACD, dan Bollinger Bands dalam pemilihan saham IDX30, memberikan wawasan bagi investor dalam memanfaatkan momentum pasar [15]. Selain itu, indeks IDX80 yang meliputi 80 saham unggulan juga dikaji oleh Meidiyustiani (2025); penelitian tersebut mengungkap bahwa profitabilitas dan *leverage* secara simultan mempengaruhi harga saham pada perusahaan indeks IDX80, dengan *leverage* berperan sebagai variabel moderasi yang memperkuat pengaruh profitabilitas terhadap harga saham [17]. Dengan demikian, klasifikasi berdasarkan likuiditas dan kapitalisasi pasar seperti IDX30 dan IDX80 menjadi acuan bagi investor untuk menyusun strategi dan menjadi bahan penelitian.

2.1.2 Karakteristik Harga Saham

Harga saham memiliki sifat non-stasioner yang cenderung bergerak secara *random walk* [18][19]. Pemodelan kuantitatif pasar saham sering menggunakan pendekatan *Multivariate Geometric Random Walk*, yang mengindikasikan bahwa pergerakan harga saham sulit diprediksi secara sempurna karena sifatnya yang acak [19]. Meskipun nilai eksponen Hurst (H) sering kali tidak sama dengan 0,5, yang mengindikasikan adanya memori jangka panjang, hal ini tetap kompatibel dengan model *random walk* dan tidak

melanggar hipotesis pasar efisien [19].

Selain itu, harga saham juga volatil dan dipengaruhi oleh sentimen pasar serta faktor eksternal [18][19]. Volatilitas yang tinggi menyebabkan harga dapat berfluktuasi secara cepat sebagai respons terhadap berita ekonomi, kebijakan moneter, atau peristiwa politik. Perubahan harga yang cepat dan tidak terduga ini mencerminkan proses penyerapan informasi baru ke dalam harga saham, sesuai dengan kerangka *random walk* [19]. Walaupun terdapat *long-range memory* dalam *return* saham, secara keseluruhan pasar tetap efisien dalam merespon informasi, sehingga volatilitas yang tinggi merupakan dampak dari sifat acak dan responsifnya harga saham [19].

Harga saham juga menunjukkan pola musiman tertentu serta memiliki korelasi yang kuat antar saham dalam satu sektor atau dengan indeks pasar [18][19]. Fenomena seperti *Weekend Effect* (*return* lebih tinggi di hari Jumat) menunjukkan adanya anomali jangka pendek yang menyimpang dari *random walk* murni. Pendekatan *Multivariate Geometric Random Walk* tetap relevan untuk memodelkan dinamika harga secara agregat [18]. Adanya *long-range memory* dalam *return* saham menjelaskan mengapa korelasi antar saham dapat bertahan dalam periode tertentu, meskipun dalam jangka panjang *return* tetap bersifat acak [19].

2.1.3 Saham BBKA dan AMRT

PT Bank Central Asia Tbk (BBKA) merupakan salah satu emiten perbankan terbesar di Indonesia yang termasuk kategori *blue chip* dengan kapitalisasi pasar dan likuiditas yang tinggi. Kinerja perusahaan didukung oleh pertumbuhan kredit yang stabil, kualitas aset yang baik, serta penerapan transformasi digital yang berkelanjutan. Selain faktor fundamental keuangan, aspek *Environmental, Social, and Governance* (ESG) juga menjadi perhatian investor dalam pengambilan keputusan investasi pada saham BBKA. Faktor tata kelola perusahaan (*governance*) memiliki pengaruh signifikan terhadap keputusan investasi investor ritel karena mencerminkan transparansi, akuntabilitas, dan kredibilitas perusahaan di pasar modal [20]. Karakteristik tersebut menjadikan BBKA sebagai salah satu saham yang sering digunakan sebagai representasi sektor perbankan dalam berbagai penelitian pasar modal.

PT Sumber Alfaria Trijaya Tbk (AMRT) merupakan perusahaan ritel modern yang bergerak pada sektor *consumer non-cyclicals* melalui jaringan minimarket Alfamart. Kinerja AMRT dipengaruhi oleh daya beli masyarakat, tingkat konsumsi rumah tangga, strategi ekspansi gerai, serta efisiensi operasional perusahaan. Sebagai emiten yang bergerak pada

sektor konsumsi, AMRT memiliki keterkaitan erat dengan kondisi ekonomi domestik karena produk yang dipasarkan merupakan kebutuhan sehari-hari masyarakat. Pertumbuhan jumlah gerai, peningkatan pendapatan, dan kemampuan perusahaan beradaptasi terhadap perubahan perilaku konsumen menjadi faktor yang memengaruhi persepsi investor terhadap saham AMRT. Oleh karena itu, saham AMRT sering digunakan sebagai representasi sektor konsumsi dalam penelitian terkait prediksi harga saham dan analisis risiko investasi [5].

Saham BBKA dan AMRT memiliki keterkaitan yang erat dengan Indeks Harga Saham Gabungan (IHSG) karena keduanya merupakan emiten berkapitalisasi besar pada sektornya masing-masing. BBKA mewakili sektor keuangan yang menjadi salah satu kontributor utama pembentukan IHSG, sedangkan AMRT mewakili sektor konsumsi yang mencerminkan aktivitas ekonomi masyarakat Indonesia. Menurut Trimono dkk. [5], kedua saham tersebut termasuk dalam kelompok saham *blue chip* yang mengalami peningkatan kapitalisasi pasar yang signifikan sehingga menarik untuk dianalisis dari aspek risiko investasi. Hasil penelitian menggunakan metode *Value-at-Risk* (VaR) dan *Adjusted Expected Shortfall* (Adj-ES) menunjukkan bahwa risiko kerugian saham AMRT berada pada kisaran 1,2%-3,4%, sedangkan BBKA berada pada kisaran 1,1%-3,2%, dengan nilai Violation Ratio (VR) kurang dari satu yang mengindikasikan tingkat akurasi prediksi risiko yang sangat baik. Temuan tersebut menunjukkan bahwa meskipun BBKA dan AMRT memiliki fundamental yang kuat, analisis risiko dan prediksi harga saham tetap diperlukan untuk mendukung pengambilan keputusan investasi yang lebih optimal [5].

2.2 Analisis Fundamental

2.2.1 Definisi Analisis Fundamental

Analisis fundamental merupakan salah satu pendekatan yang digunakan untuk menilai nilai intrinsik suatu instrumen investasi, khususnya saham, dengan mempertimbangkan berbagai faktor yang memengaruhi kinerja perusahaan. Pendekatan ini tidak hanya berfokus pada aspek keuangan, tetapi juga mencakup kondisi ekonomi, industri, serta dinamika pasar yang relevan. Melalui analisis ini, investor berupaya membandingkan nilai intrinsik saham dengan harga pasar yang berlaku guna menentukan apakah suatu saham berada dalam kondisi *Undervalued* atau *Overvalued* [21].

Dalam pelaksanaannya, analisis fundamental dilakukan dengan memanfaatkan berbagai informasi yang bersumber dari laporan keuangan perusahaan, seperti laporan laba rugi dan neraca, serta indikator rasio keuangan yang mencerminkan kinerja perusahaan.

Pendekatan ini dapat dilakukan melalui metode top-down yang diawali dari analisis kondisi makroekonomi hingga ke tingkat perusahaan, maupun metode bottom-up yang lebih menitikberatkan pada kondisi internal perusahaan, seperti profitabilitas, struktur modal, dan efisiensi operasional [21][22]. Dengan demikian, analisis fundamental mampu memberikan gambaran yang lebih menyeluruh mengenai kondisi dan prospek perusahaan.

Sejalan dengan hal tersebut, analisis fundamental memiliki peran penting dalam mendukung pengambilan keputusan investasi, terutama dalam jangka panjang. Pendekatan ini memungkinkan investor untuk mengidentifikasi peluang investasi yang muncul akibat adanya perbedaan antara harga pasar dan nilai sebenarnya dari suatu perusahaan. Selain itu, dengan menitikberatkan pada kekuatan dasar perusahaan, analisis fundamental membantu proses evaluasi investasi menjadi lebih sistematis dan rasional karena didasarkan pada kinerja riil perusahaan, bukan semata-mata pada pergerakan harga pasar [22].

2.2.2 Indikator Analisis Fundamental

Dalam analisis fundamental saham, rasio keuangan digunakan sebagai indikator untuk menilai kinerja dan kondisi keuangan perusahaan [23][24][25]. Rasio-rasio ini memberikan informasi penting bagi investor dalam pengambilan keputusan investasi, yang umumnya diukur melalui beberapa indikator utama:

1. *Return on Equity* (ROE)

Return on Equity (ROE) digunakan untuk mengukur kemampuan perusahaan dalam menghasilkan laba dari modal pemegang saham. ROE yang tinggi menunjukkan perusahaan lebih efisien dalam menghasilkan keuntungan [23]. Rumus ROE adalah sebagai berikut:

$$ROE = \frac{\text{Laba bersih}}{\text{Total Ekuitas}} \quad (1)$$

Dimana:

ROE	= <i>Return on Equity</i>
Laba Bersih	= Keuntungan setelah pajak
Total Ekuitas	= Total modal pemegang saham

2. *Debt to Equity Ratio* (DER)

Debt to Equity Ratio (DER) digunakan untuk mengukur tingkat penggunaan utang terhadap modal sendiri. DER yang tinggi menunjukkan risiko keuangan yang lebih besar [23][25]. Rumus DER adalah sebagai berikut:

$$DER = \frac{\text{Total Utang}}{\text{Total Ekuitas}} \quad (2)$$

Dimana:

DER = *Debt to Equity Ratio*
Total Utang = Total kewajiban perusahaan
Total Ekuitas = Modal pemegang saham

3. *Net Profit Margin* (NPM)

Net Profit Margin (NPM) digunakan untuk mengukur kemampuan perusahaan dalam menghasilkan laba bersih dari pendapatan. NPM yang tinggi menunjukkan efisiensi perusahaan dalam menghasilkan laba [25]. Rumus NPM adalah sebagai berikut:

$$NPM = \frac{\text{Laba bersih}}{\text{Pendapatan}} \quad (3)$$

Dimana:

NPM = Net Profit Margin
Laba Bersih = Keuntungan setelah pajak
Pendapatan = Total penjualan

4. *Earnings per Share* (EPS)

Earnings per Share (EPS) digunakan untuk mengukur laba yang diperoleh untuk setiap lembar saham. EPS yang tinggi menunjukkan kinerja perusahaan yang baik dari sisi profitabilitas [23]. Rumus EPS adalah sebagai berikut:

$$EPS = \frac{\text{Laba Bersih}}{\text{Jumlah Saham Beredar}} \quad (4)$$

Dimana:

EPS = *Earnings per Share*
Laba Bersih = Keuntungan setelah pajak
Jumlah Saham Beredar = Total saham yang beredar di pasar

5. *Price to Earnings Ratio* (PER)

Price to Earnings Ratio (PER) digunakan untuk mengukur seberapa mahal harga saham dibandingkan dengan laba yang dihasilkan perusahaan. PER yang rendah mengindikasikan saham dijual di bawah nilai bukunya (*Undervalued*), sedangkan PER yang tinggi mengindikasikan saham dijual di atas nilai bukunya (*Overvalued*) [24]. Rumus PER adalah sebagai berikut:

$$PER = \frac{\text{Harga Saham}}{\text{Laba per Lembar Saham (EPS)}} \quad (5)$$

Dimana:

PER = *Price to Earnings Ratio*

Harga Saham = Harga pasar per lembar saham

Earning per Share = Laba bersih per lembar saham

2.3 Analisis Teknikal

2.3.1 Definisi Analisis Teknikal

Analisis teknikal merupakan suatu pendekatan yang digunakan untuk memprediksi pergerakan harga saham dengan memanfaatkan data historis pasar, seperti harga dan volume perdagangan. Pendekatan ini didasarkan pada asumsi bahwa seluruh informasi yang relevan telah tercermin dalam harga, sehingga pola pergerakan di masa lalu dapat digunakan untuk memperkirakan arah tren di masa depan [29].

Lebih lanjut, dalam penerapannya, analisis teknikal dilakukan dengan menggunakan grafik harga serta berbagai indikator kuantitatif untuk mengidentifikasi pola dan tren pasar. Seiring perkembangannya, metode ini tidak hanya terbatas pada analisis visual, tetapi juga didukung oleh penggunaan algoritma dan teknologi seperti *machine learning* untuk meningkatkan akurasi prediksi [29].

Selanjutnya, analisis teknikal memiliki peran penting dalam mendukung pengambilan keputusan investasi, terutama dalam jangka pendek. Pendekatan ini membantu investor dalam menentukan waktu yang tepat untuk melakukan transaksi berdasarkan pergerakan harga dan volume, sehingga lebih responsif terhadap dinamika pasar yang cepat berubah [30].

2.3.2 Indikator Analisis Teknikal

Dalam analisis teknikal, indikator digunakan sebagai alat untuk membaca pola pergerakan harga dan membantu memprediksi arah tren pasar. Indikator-indikator ini umumnya berbasis pada data historis seperti harga dan volume perdagangan, serta digunakan untuk mendukung pengambilan keputusan investasi secara lebih objektif [31][32]. Beberapa indikator yang umum digunakan dalam analisis teknikal:

1. *Moving Averages* (MA)

Moving Averages (MA) digunakan untuk menghaluskan data harga sehingga memudahkan dalam mengidentifikasi tren pergerakan saham. MA digunakan untuk mengidentifikasi tren, dimana dalam penelitian ini digunakan SMA 20 dan SMA 50 dimana *Simple Moving Average* (SMA) merupakan Perhitungan rata-rata harga penutupan (closing price) [31]. Terdapat dua jenis MA yang umum digunakan, yaitu:

a. *Simple Moving Average* (SMA)

Rumus *Simple Moving Average* (SMA) adalah sebagai berikut:

$$SMA_n = \frac{P_1 + P_2 + \dots + P_n}{n} \quad (6)$$

Dimana:

SMA_n = Nilai SMA periode ke-n

$P_1, P_2, \dots, P_n$ = Harga penutupan pada tiap periode

n = Jumlah periode

b. *Exponential Moving Average* (EMA)

Rumus *Exponential Moving Average* (EMA) adalah sebagai berikut:

$$EMA_t = \alpha \cdot P_t + (1 - \alpha) \cdot EMA_{t-1} \quad (7)$$

Dimana:

EMA_t = EMA pada waktu ke-t

P_t = Harga pada waktu ke-t

EMA_{t-1} = EMA periode sebelumnya

α = Faktor smoothing

2. *Relative Strength Index* (RSI)

Relative Strength Index (RSI) digunakan untuk mengukur kekuatan relatif pergerakan harga saham dalam periode tertentu. Jika nilai RSI lebih kecil dari 30, maka saham tersebut dianggap *oversold*, sedangkan jika saham tersebut lebih besar dari 70, maka saham tersebut dianggap *overbought* [32]. Rumus RSI adalah sebagai berikut:

$$RSI = 100 - \left(\frac{100}{1 + RS} \right) \quad (8)$$

$$RS = \frac{\text{Rata - rata Gain}}{\text{Rata - rata Loss}} \quad (9)$$

Dimana:

RSI = *Relative Strength Index*

RS = *Relative Strength*

3. *Moving Average Convergence Divergence* (MACD)

Moving Average Convergence Divergence (MACD) digunakan untuk mengidentifikasi perubahan momentum dan tren harga saham. MACD digunakan untuk mendeteksi sinyal beli dan jual berdasarkan perpotongan garis [31]. Rumus MACD adalah sebagai berikut:

$$MACD = EMA_{12} - EMA_{26} \quad (10)$$

Signal Line dihitung sebagai:

$$\text{Signal Line} = \text{EMA}_9(\text{MACD}) \quad (11)$$

Dimana:

MACD = Selisih EMA 12 dan EMA 26

EMA₁₂ = EMA periode 12

EMA₂₆ = EMA periode 26

Signal Line = EMA 9 dari nilai MACD

4. Bollinger Bands (BB)

Bollinger Bands digunakan untuk mengukur volatilitas harga dan mengidentifikasi kondisi *overbought* atau *oversold*. Jika harga mendekati *upper band*, maka dapat dikatakan bahwa saham tersebut *overbought*. Namun jika harga mendekati *lower band*, maka dapat dikatakan bahwa saham tersebut *oversold* [32]. Rumus Bollinger Bands adalah sebagai berikut:

$$\text{Middle Band} = \text{SMA}_{20} \quad (12)$$

$$\text{Upper Band} = \text{SMA}_{20} + (2 \cdot \sigma) \quad (13)$$

$$\text{Lower Band} = \text{SMA}_{20} - (2 \cdot \sigma) \quad (14)$$

Dimana:

SMA₂₀ = *Simple Moving Average* periode 20

σ = Standar deviasi harga

5. On-Balance Volume (OBV)

On-Balance Volume (OBV) digunakan untuk mengkonfirmasi tren harga dengan melihat pergerakan volume perdagangan. OBV digunakan untuk mengkonfirmasi apakah tren harga didukung oleh volume [32]. Rumus OBV adalah sebagai berikut:

$$\text{OBV}_t = \text{OBV}_{t-1} + \text{Volume}_t (\text{jika } \text{Close}_t > \text{Close}_{t-1}) \quad (15)$$

$$\text{OBV}_t = \text{OBV}_{t-1} - \text{Volume}_t (\text{jika } \text{Close}_t < \text{Close}_{t-1}) \quad (16)$$

Dimana:

OBV_t = OBV pada waktu ke-t

OBV_{t-1} = OBV periode sebelumnya

Volume_t = Volume perdagangan pada waktu ke-t

Close_t = Harga penutupan saat ini

Close_{t-1} = Harga penutupan sebelumnya

2.4 Analisis Sentimen

Analisis sentimen merupakan pendekatan yang digunakan untuk mengidentifikasi dan mengevaluasi opini, emosi, maupun persepsi publik terhadap suatu objek berdasarkan data tekstual. Dalam pasar modal, analisis sentimen dimanfaatkan untuk memahami respons investor terhadap informasi yang beredar, seperti berita, media sosial, maupun forum diskusi, sehingga dapat digunakan untuk memperkirakan pergerakan pasar dan perilaku investor [36].

Dalam penerapannya, analisis sentimen dilakukan dengan memanfaatkan teknik pengolahan bahasa alami (*natural language processing*) dan metode *Machine Learning* untuk mengklasifikasikan sentimen ke dalam kategori positif, negatif, atau netral. Perkembangan teknologi memungkinkan proses analisis dilakukan secara *real-time* melalui integrasi data digital, seperti *web scraping* dan model *Deep Learning*, sehingga informasi yang diperoleh menjadi lebih cepat dan adaptif terhadap perubahan kondisi pasar [37].

Lebih lanjut, analisis sentimen memiliki peran penting dalam mendukung pengambilan keputusan investasi karena mampu memberikan gambaran mengenai kondisi psikologis pasar yang tidak selalu tercermin dalam data fundamental maupun teknikal. Informasi sentimen dapat membantu investor dalam memahami kecenderungan pasar secara lebih komprehensif, terutama pada kondisi pasar yang dipengaruhi oleh opini publik dan arus informasi yang bergerak cepat [36].

Penggunaan jendela waktu 7 hari dalam pengambilan data sentimen untuk analisis saham didasarkan pada bukti empiris bahwa pengaruh sentimen terhadap pergerakan harga memiliki periode efektif tertentu [38][39]. Penelitian menunjukkan bahwa sentimen dari pencarian Google dapat memprediksi return saham dalam rentang waktu 3 hingga 7 hari [38], sementara siklus sentimen investor terbukti memiliki periode signifikan selama 7 hari yang terbentuk dari kombinasi 5 hari perdagangan dan 2 hari libur akhir pekan [39]. Studi pada Bursa Efek Indonesia juga menggunakan jendela 7 hari ($t=-1$ hingga $t=5$) untuk mengamati reaksi pasar terhadap berita, di mana efek negatif terlihat pada saat pengumuman berita dan efek positif muncul pada hari ke-4 setelah berita [40]. Dengan demikian, periode 7 hari dipilih karena merepresentasikan siklus alami pengaruh sentimen di pasar modal serta menjadi horizon waktu yang optimal untuk prediksi jangka pendek [38][39][40].

2.5 Deep Learning

Deep Learning adalah cabang *machine learning* yang menggunakan jaringan saraf tiruan dengan banyak lapisan tersembunyi untuk mempelajari representasi data secara

hierarkis tanpa rekayasa fitur manual [41]. Berbeda dengan *Random Forest* yang merupakan ensemble pohon keputusan dangkal, *Deep Learning* mampu mengekstraksi pola dari level rendah hingga abstrak melalui propagasi balik dan aktivasi non-linear [42].

Contoh model *Deep Learning* meliputi CNN (*Convolutional Neural Network*), seperti ResNet (*Residual Network*) dan EfficientNet, untuk data gambar. Untuk data deret waktu dan teks, digunakan LSTM (*Long Short-Term Memory*) dan GRU (*Gated Recurrent Unit*) [43]. LSTM menggunakan tiga gerbang (input, forget, output), sementara GRU menyederhanakannya dengan dua gerbang (reset, update) sehingga lebih efisien [44].

Model *Deep Learning* modern lainnya adalah transformer dengan varian BERT (*Bidirectional Encoder Representations from Transformers*), GPT (*Generative Pre-trained Transformer*), dan ViT (*Vision Transformer*) yang menggunakan mekanisme *self-attention* multi-kepala. Selain itu, terdapat VAE (*Variational Autoencoder - Autoencoder Variasional*) serta GAN (*Generative Adversarial Network*) yang terdiri dari generator vs diskriminator untuk pembelajaran generatif [45]. Semua model ini memiliki lebih dari tiga lapisan non-linear, menjadikannya *Deep Learning* murni [42].

2.6 Long Short-Term Memory (LSTM)

2.6.1 Definisi Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) merupakan salah satu jenis arsitektur *recurrent neural network* (RNN) yang dirancang untuk mengatasi masalah *vanishing gradient* sehingga mampu menyimpan informasi dalam jangka waktu panjang [46]. Dalam analisis teknikal saham, model LSTM digunakan untuk memprediksi indikator-indikator seperti MACD dan Stochastic Oscillator, sehingga dapat membantu investor menentukan waktu yang tepat untuk melakukan transaksi jual dan beli [46]. Pengaruh LSTM terhadap analisis teknikal juga ditunjukkan melalui kemampuannya dalam memprediksi harga saham sektor telekomunikasi dengan tingkat kesalahan prediksi (MAPE) di bawah 10% [47]. Penambahan indikator teknikal dalam model LSTM tidak secara signifikan meningkatkan akurasi prediksi, tetapi terbukti mempercepat efisiensi komputasi [47].

2.6.2 Arsitektur dan Cara Kerja LSTM:

LSTM terdapat *Pre-processing* berupa normalisasi data harga saham ke rentang [0,1] menggunakan metode Min-Max. Tujuan dari normalisasi adalah untuk menyamakan skala fitur data, yang merupakan langkah penting dalam model *Deep Learning* seperti LSTM

karena mencegah model menjadi bias terhadap fitur dengan nilai numerik yang lebih besar dan membantu stabilitas pelatihan [50]. Proses ini sangat penting karena LSTM menggunakan fungsi aktivasi seperti *sigmoid* dan *hyperbolic tangent* (tanh) yang sensitif terhadap skala input [51].

$$x' = (x - x_{min}) / (x_{max} - x_{min}) \quad (17)$$

adalah definisi standar dari Min-Max Normalisasi. Literatur menjelaskan bahwa transformasi ini secara linear memetakan nilai asli (x) ke dalam rentang baru (biasanya 0 hingga 1) berdasarkan nilai minimum dan maksimum dari fitur tersebut [51].

Pembentukan sequence data dengan parameter lookback dan horizon merupakan langkah fundamental dalam pemodelan deret waktu menggunakan LSTM. Proses ini mengubah data deret waktu menjadi format pembelajaran terawasi (supervised learning) dengan membuat pasangan input-target dari urutan data historis [43]. Lookback Window didefinisikan sebagai jumlah langkah waktu masa lalu (timesteps) yang digunakan sebagai input untuk memprediksi nilai di masa depan [44]. Pemilihan panjang lookback yang tepat sangat krusial karena mempengaruhi kemampuan model dalam menangkap ketergantungan jangka pendek dan panjang pada data deret waktu [45]. Lookback yang terlalu pendek dapat menyebabkan model kehilangan informasi pola penting, sedangkan lookback yang terlalu panjang dapat menyebabkan overfitting dan peningkatan kompleksitas komputasi [46]. Horizon menentukan seberapa jauh ke depan prediksi akan dibuat. Horizon = 1, yang dikenal sebagai next-step prediction atau one-step-ahead forecasting, merupakan pendekatan yang paling umum digunakan dalam peramalan deret waktu karena kesederhanaannya dan efektivitasnya untuk prediksi jangka pendek [43]. Pendekatan ini sangat sesuai untuk aplikasi prediksi harga saham di mana prediksi satu langkah ke depan sering kali menjadi dasar untuk pengambilan keputusan trading [47].

Sejak diperkenalkan secara resmi oleh Hochreiter dan Schmidhuber pada tahun 1997, arsitektur *Long Short-Term Memory* (LSTM) telah menjadi salah satu fondasi penting dalam pemrosesan data sekuensial [48]. LSTM dirancang sebagai solusi atas keterbatasan *vanishing gradient problem* yang dihadapi oleh *Recurrent Neural Network* (RNN) konvensional [49]. Arsitektur ini mengatasi masalah tersebut dengan memperkenalkan sebuah *memory cell* yang mampu menyimpan informasi dalam jangka waktu panjang, dikendalikan oleh tiga gerbang utama, yaitu: *forget gate*, *input gate*, dan *output gate* [50]. Secara teknis, mekanisme ini dapat dijelaskan melalui persamaan-persamaan berikut yang menggambarkan aliran data dalam setiap unit LSTM:

1. Forget Gate (f_t)

Forget Gate berfungsi untuk menentukan informasi mana yang akan dibuang dari *cell state*.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (18)$$

Dimana:

f_t = nilai aktivasi *forget gate* pada langkah waktu ke- t

σ = fungsi aktivasi sigmoid

W_f = matriks bobot untuk *forget gate*

h_{t-1} = *hidden state* dari langkah waktu sebelumnya

x_t = vektor masukan pada langkah waktu t

b_f = nilai bias untuk *forget gate*

2. Input Gate (i_t)

Input gate berfungsi untuk menentukan informasi baru yang akan disimpan ke dalam *cell state*.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (19)$$

Dimana:

i_t = *Input gate*

g_t = Kandidat nilai *cell state*

W_i, U_i, W_g, U_g = Bobot

$\tanh$ = Fungsi aktivasi *hyperbolic tangent*

3. Candidate Cell State

Candidate Cell State adalah status sementara pada sebuah cell dalam proses komputasi atau algoritma yang masih menjadi kandidat untuk diproses, dipilih, atau diperbarui berdasarkan kondisi tertentu.

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad (20)$$

Dimana:

$\tilde{C}_t$ = *candidate cell state* (nilai memori kandidat)

$\tanh$ = fungsi aktivasi tangen hiperbolik

W_C = matriks bobot untuk *candidate cell*

b_C = nilai bias untuk *candidate cell*

4. Cell State Update (C_t)

Cell state merupakan memori jangka panjang yang diperbarui berdasarkan informasi dari *forget gate* dan *input gate*.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \quad (21)$$

Dimana:

C_t = *cell state* (memori jangka panjang) pada langkah t

C_{t-1} = *cell state* sebelumnya

$*$ = perkalian element-wise (Hadamard product)

5. Output Gate

Output gate berfungsi untuk menentukan informasi yang akan dikeluarkan sebagai output.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (22)$$

Dimana:

o_t = nilai aktivasi *output gate*

W_o = matriks bobot untuk *output gate*

b_o = nilai bias untuk *output gate*

6. Hidden State Update (h_t)

Hidden state merupakan output dari LSTM pada waktu ke- t .

$$h_t = o_t * \tanh(C_t) \quad (23)$$

Dimana:

h_t = *Hidden state* saat ini

C_t = *Cell state*

o_t = *Output gate*

Berikut cara kerja LSTM yang dituangkan dalam gambar:

Cara kerja *Long Short-Term Memory* (LSTM) dimulai dengan menerima dua masukan utama pada setiap langkah waktu t , yaitu data baru x_t dan *hidden state* sebelumnya h_{t-1} , serta memori jangka panjang sebelumnya C_{t-1} [50]. Langkah pertama yang dilakukan adalah *forget gate* yang menggunakan fungsi sigmoid (σ) untuk menentukan informasi mana dari memori lama C_{t-1} yang akan dibuang. Nilai f_t yang dihasilkan berkisar antara 0 (lupakan semua) hingga 1 (pertahankan semua) [41]. Selanjutnya, *input gate* memutuskan nilai baru mana yang akan disimpan. Dua bagian bekerja di sini, yakni sigmoid (i_t) untuk memilih nilai yang akan diperbarui, dan

lapisan tanh ($\tilde{C}_t$) menciptakan nilai kandidat memori baru. Hasil perkalian keduanya menjadi informasi baru yang akan ditambahkan [49].

Kemudian, *cell state* diperbarui dengan persamaan $C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$. Proses ini menggabungkan informasi lama yang dipertahankan ($f_t * C_{t-1}$) dengan informasi baru yang diizinkan masuk ($i_t * \tilde{C}_t$) [51]. Memori jangka panjang C_t kini siap digunakan untuk langkah waktu berikutnya. Langkah terakhir adalah *output gate*, yang memutuskan bagian mana dari memori C_t yang akan menjadi keluaran (h_t). Nilai o_t dari sigmoid difilter dengan $\tanh(C_t)$ sehingga menghasilkan *hidden state* baru $h_t = o_t * \tanh(C_t)$. h_t ini berfungsi sebagai keluaran untuk langkah waktu t sekaligus menjadi masukan untuk langkah waktu berikutnya bersama dengan C_t [52]. Secara keseluruhan, tiga gerbang ini memungkinkan LSTM untuk secara selektif melupakan, mengingat, dan mengeluarkan informasi, sehingga mengatasi masalah *vanishing gradient* pada RNN standar [49].

2.6.3 Framework LSTM

Framework Long Short-Term Memory (LSTM) merupakan arsitektur jaringan saraf recurrent yang dirancang untuk mempelajari ketergantungan jangka panjang dalam data deret waktu, mengatasi masalah *vanishing gradient* yang terjadi pada RNN standar [50]. Dalam praktiknya, sebuah kerangka kerja LSTM tidak hanya terdiri dari sel LSTM tunggal, melainkan mencakup keseluruhan aliran pemrosesan mulai dari masukan hingga keluaran. Berbagai tinjauan pustaka sistematis menunjukkan bahwa pengembangan kerangka kerja LSTM modern sering menggabungkan LSTM dengan metode statistik (misalnya ARIMA-X [46]), varian arsitektur seperti *stacked LSTM* atau *bidirectional LSTM* [48][49], serta mekanisme koreksi data bawaan seperti yang diusulkan dalam *Corrector LSTM* [51]. Selain itu, untuk aplikasi finansial seperti prediksi harga saham sektor telekomunikasi [47] atau deteksi anomali pada deret panjang, arsitektur *extended LSTM* (xLSTM) dengan *exponential gating* dan memori revisi terbukti lebih unggul dibanding LSTM klasik [52].

Salah satu komponen kunci yang memperkaya *framework LSTM* adalah integrasi *Attention Mechanism*, yang memungkinkan model secara dinamis memberikan bobot berbeda pada setiap langkah waktu dalam urutan masukan. Sebagai contoh, *framework* untuk klasifikasi kelainan jantung dari sinyal ECG menggunakan lapisan *attention* di atas LSTM untuk memfokuskan pada segmen sinyal yang paling relevan secara diagnostik [53]. Sementara itu, pada analisis sentimen, *framework LSTM* dengan *multi-head attention* mampu menangkap berbagai aspek semantik secara paralel (seperti intensitas

sentimen, sarkasme, dan negasi) sehingga menghasilkan akurasi yang lebih tinggi dibandingkan LSTM tanpa *attention* [54]. Pendekatan ini sejalan dengan temuan bahwa kombinasi LSTM dengan *Attention Mechanism* dapat meningkatkan kinerja pada berbagai tugas deret waktu [48][50].

Selain pemilihan arsitektur dan *Attention Mechanism*, kerangka kerja LSTM yang tangguh juga harus memperhatikan aspek pra-pemrosesan data dan regularisasi untuk mencegah *overfitting*. Dalam data hidrologi, misalnya, tinjauan kritis terhadap varian RNN dan LSTM menekankan pentingnya penanganan data yang hilang dan normalisasi sebelum data dimasukkan ke dalam LSTM [49]. Demikian pula pada *Corrector LSTM*, mekanisme koreksi *outlier* secara *end-to-end* terintegrasi ke dalam aliran pelatihan, sehingga model tidak belajar dari kesalahan pencatatan yang dapat mengganggu prediksi [51]. Dengan menggabungkan lapisan koreksi data, arsitektur LSTM (termasuk varian seperti xLSTM [52]), dan *Attention Mechanism* [53][54], sebuah kerangka kerja LSTM dapat diadaptasi untuk berbagai domain seperti keuangan [46][47], hidrologi [49], analisis sentimen [54], dan klasifikasi sinyal biomedis [53], menjadikannya salah satu arsitektur paling fleksibel dalam *Deep Learning* untuk data deret waktu [50].

2.7 Random Forest

2.7.1 Definisi Random Forest

Random Forest adalah algoritma *Machine Learning* berbasis ensemble yang membangun banyak pohon keputusan dengan menerapkan mekanisme bagging dan pemilihan fitur secara acak, sehingga menghasilkan model yang lebih stabil dan akurat dibandingkan pohon tunggal [55]. Algoritma ini bekerja dengan membangun sejumlah pohon pada subset data hasil bootstrap-di mana setiap pemisahan (split) hanya mempertimbangkan subset fitur yang dipilih secara acak-sehingga menghasilkan keragaman antar pohon yang mampu mengurangi risiko *overfitting* secara signifikan [55]. Metode yang lebih baru seperti *Partitioned Random Forest* (PRF) bahkan mampu meningkatkan akurasi dengan membagi ruang fitur menjadi beberapa wilayah berbeda, sehingga algoritma dapat mempelajari batasan keputusan yang lebih kompleks [55]. Keunggulan utama *Random Forest* meliputi efisiensi komputasi, ketahanan terhadap noise dan data berdimensi tinggi, kemampuan menangani data yang tidak lengkap, serta menyediakan estimasi kepentingan fitur secara built-in [56]. Secara historis, *Random Forest* pertama kali didefinisikan oleh Leo Breiman pada tahun 2001, dan telah menjadi salah satu alat *machine learning* paling populer

karena kemudahan tuning serta interpretabilitasnya yang relatif baik [56].

Jumlah pohon keputusan ($n_{\text{estimators}}$) ditetapkan sebesar 100 berdasarkan rekomendasi perintis *Random Forest*, Leo Breiman, yang menyatakan bahwa 100 pohon cukup untuk mencapai kestabilan model dan konvergensi generalisasi error [64][56][55], dan nilai ini juga menjadi default pada pustaka standar seperti scikit-learn [7]. Parameter max_depth dibatasi hingga 10 untuk mencegah overfitting dan meningkatkan generalisasi, terutama pada data keuangan yang bersifat noisy [3][28][8]. Nilai min_samples_split sebesar 5 memastikan setiap pemisahan didasarkan pada subset data yang representatif sehingga mengurangi risiko *overfitting* [3][61][2], sementara min_samples_leaf sebesar 2 digunakan untuk menghasilkan pohon yang lebih smooth dan stabil [61][55], dengan nilai ini merupakan default yang umum digunakan dan terbukti efektif dalam berbagai studi prediksi saham [8][2]. Terakhir, random_state diatur ke 42 untuk menjamin reproduksibilitas dan konsistensi hasil eksperimen [64][5][10], sebagaimana direkomendasikan dalam berbagai penelitian untuk memastikan proses bootstrap sampling, pemilihan fitur acak, dan pembangunan pohon dapat direproduksi [15][5]. Keseluruhan hyperparameter ini bekerja secara sinergis untuk mengontrol kompleksitas model dan menghasilkan *Random Forest* yang optimal untuk prediksi data saham [3][61][28].

2.7.2 Arsitektur dan Cara Kerja *Random Forest*

Random Forest merupakan metode *ensemble learning* yang menggabungkan banyak pohon keputusan independen melalui dua mekanisme utama: *bagging* (*bootstrap aggregating*) dan pemilihan fitur secara acak [57]. Arsitektur sistem ini terdiri dari tiga lapisan fungsional. Pertama, pada lapisan pembangkitan data, algoritma mengambil T sampel *bootstrap* dari dataset asli berukuran N , sehingga setiap pohon dilatih pada subset data yang berbeda. Properti yang terkenal dalam bootstrap (terutama pada *bagging* di machine learning) adalah probabilitas bahwa sebuah sampel tertentu dari dataset asli akan muncul dalam satu sampel Bootstrap.

Probabilitas terpilih dalam satu pengambilan acak:

$$\frac{1}{n} \quad (24)$$

Probabilitas TIDAK terpilih dalam satu pengambilan acak:

$$1 - \frac{1}{n} \quad (25)$$

Probabilitas TIDAK PERNAH terpilih setelah n pengambilan independen (yaitu menjadi *out-of-bag*):

$$\left(1 - \frac{1}{n}\right)^n \quad (26)$$

Ketika ukuran sampel n membesar, probabilitas ini konvergen ke:

$$\lim_{n \rightarrow \infty} \left(1 - \frac{1}{n}\right)^n \quad (27)$$

Kedua, pada lapisan pembangunan pohon, hanya sebanyak *m* fitur yang dipilih secara acak pada setiap simpul untuk menjadi kandidat pemisah terbaik. Proses ini berulang hingga setiap pohon tumbuh maksimal tanpa pemangkasan (*pruning*) [58]. Ketiga, pada lapisan agregasi, seluruh pohon yang telah terbentuk memberikan suara (untuk klasifikasi) atau nilai rata-rata (untuk regresi) untuk menghasilkan prediksi final. Keberagaman antar pohon yang dihasilkan oleh dua sumber keacakan ini terbukti mampu menurunkan *variance* secara signifikan tanpa meningkatkan *bias* secara berarti [59].

1. Kriteria pemisahan simpul - Gini Impurity

Digunakan untuk mengukur tingkat ketidakmurnian suatu simpul s . Berdasarkan literatur [57], rumus *Gini Impurity* dinyatakan pada Persamaan (29):

$$G(s) = 1 - \sum_{k=1}^K p_k^2 \quad (28)$$

Dimana:/

$G(s)$ = nilai Gini Impurity pada simpul s
 K = jumlah kelas
 p_k = proporsi kelas ke- k

2. Penduga kesalahan generalisasi - Out-of-Bag (OOB) Error

Setiap sampel i diprediksi oleh pohon yang tidak menggunakan sampel tersebut dalam proses bootstrap [58]. Penduga OOB error dinyatakan pada Persamaan (30):

$$OOB_{error} = \frac{1}{N} \sum_{i=1}^N 1(\hat{y}_i^{OOB} \neq y_i) \quad (29)$$

Dimana:

N = jumlah total sampel
 $\hat{y}_i^{OOB}$ = prediksi OOB untuk sampel ke- i

y_i = label sebenarnya

$1(\cdot)$ = fungsi indikator

3. Kepentingan fitur - Mean Decrease Impurity (MDI)

Mengukur kontribusi fitur X_j dengan menghitung rata-rata penurunan *Gini Impurity* yang disebabkan oleh fitur tersebut pada seluruh pohon dalam *Random Forest* [59]. Rumus MDI dinyatakan pada persamaan (31):

$$MDI(X_j) = \frac{1}{T} \sum_{t=1}^T \Delta G_{t,j} \quad (30)$$

Dimana:

$MDI(X_j)$ = skor kepentingan fitur ke- j

T = jumlah pohon dalam *Random Forest*

$\Delta G_{t,j}$ = total penurunan Gini impurity pada pohon ke- t akibat penggunaan fitur X_j

Ketika suatu simpul dipecah menjadi dua anak simpul S_L (kiri) dan S_R (kanan) berdasarkan suatu kondisi pemisahan, pengurangan *impurity* atau *Information Gain* dihitung sebagai [59]:

$$\Delta G = G(S) - \left(\frac{N_L}{N} G(S_L) + \frac{N_R}{N} G(S_R) \right) \quad (31)$$

Di mana N , N_L , dan N_R adalah jumlah sampel pada simpul induk, anak kiri, dan anak kanan. Algoritma akan memilih pemisahan yang menghasilkan *Gain* maksimum.

Secara operasional, algoritma *Random Forest* bekerja melalui tiga tahap berurutan [57][58]. Tahap pertama yaitu pembentukan pohon, setiap pohon ke- t (dengan $t = 1, 2, \dots, T$), algoritma mengambil sampel *bootstrap* dari data latih. Selanjutnya, pohon dibangun secara rekursif: pada setiap simpul, algoritma secara acak memilih subset fitur sebanyak m , lalu menghitung nilai Gini Impurity (Rumus 17) untuk setiap kandidat pemisah. Pemisah terbaik adalah yang menghasilkan penurunan Gini terbesar. Proses ini berulang hingga simpul mencapai ukuran minimum (biasanya 1 untuk klasifikasi) atau tidak ada fitur yang tersisa. Tahap kedua yaitu prediksi individual, dimana setelah semua pohon terbentuk, setiap pohon menerima data uji baru x dan menghasilkan prediksi masing-masing [59]. Tahap ketiga yaitu agregasi akhir, dimana untuk tugas klasifikasi, *Random Forest* menggunakan majority voting, yaitu memilih kelas yang paling banyak muncul di antara seluruh pohon. Untuk tugas regresi, hasil akhir adalah rata-rata dari semua prediksi pohon.

Selain itu, akurasi generalisasi model dapat diestimasi secara internal menggunakan OOB error tanpa memerlukan data validasi terpisah, sementara skor kepentingan fitur menyediakan interpretabilitas model [58].

Setelah semua pohon dalam hutan dibangun, prediksi untuk data baru dilakukan dengan mengagregasi output dari setiap pohon. Untuk klasifikasi biner, setiap pohon memberikan suara untuk salah satu kelas, dan kelas dengan suara terbanyak menjadi prediksi final [60]:

$$\hat{y}(\mathbf{x}) = \arg \max_c |\{i \in \{1, \dots, T\} \mid h_i(\mathbf{x}) = c\}| \quad (32)$$

di mana T adalah jumlah pohon, $h_i(\mathbf{x})$ adalah prediksi pohon ke- i , dan c adalah kelas .

Selain prediksi kelas, *Random Forest* juga dapat memberikan estimasi probabilitas kelas. Probabilitas bahwa data $\mathbf{x}$ termasuk dalam kelas k dihitung sebagai proporsi pohon yang memprediksi kelas tersebut [60]:

$$p_k(\mathbf{x}) = \frac{1}{T} \sum_{i=1}^T \mathbb{1}(h_i(\mathbf{x}) = k) \quad (33)$$

Probabilitas ini secara alami mencerminkan tingkat keyakinan (*confidence*) model terhadap prediksi. Semakin tinggi proporsi suara untuk suatu kelas, semakin tinggi keyakinan model .

Margin atau *margin of confidence* untuk klasifikasi biner dapat didefinisikan sebagai selisih antara proporsi suara untuk kelas yang diprediksi dengan proporsi suara untuk kelas lainnya [60]:

$$\text{Margin} = p_{\text{predicted}} - p_{\text{oth}} \quad (34)$$

Nilai margin yang tinggi (mendekati 1) menunjukkan prediksi yang sangat meyakinkan, sedangkan margin yang rendah (mendekati 0) menunjukkan prediksi yang ambigu atau tidak meyakinkan .

2.7.3 Framework *Random Forest*

Framework Random Forest (RF) dibangun di atas prinsip *ensemble* dari banyak pohon keputusan yang dilatih pada subset data *bootstrap* dan subset fitur acak, kemudian digabungkan melalui voting atau rata-rata [56]. Penguatan teoretis terhadap kerangka ini terus berkembang. Liao dkk. mengusulkan *Invariant Random Forest* (IRF) yang menambahkan penalti pada pemisahan simpul untuk menjaga stabilitas model terhadap perubahan distribusi data [62]. Sementara itu, Wang dkk. merancang *Data-driven Multinomial Random Forest* (DMRF) yang mencapai konsistensi kuat dengan probabilitas

1, menjadikannya varian RF berkompleksitas rendah namun unggul dalam klasifikasi dan regresi [63].

Selain pengembangan teoretis, berbagai varian RF dirancang untuk tugas spesifik. Heidari dkk. memperkenalkan *Partitioned Random Forest* (PRF) yang membagi ruang fitur menjadi beberapa wilayah terpisah, mencapai akurasi 98,37% dalam pengenalan emosi wajah [55]. Liu dkk. mengembangkan *self-supervised Random Forest* (sForest) untuk deteksi anomali, yang menggabungkan transformasi distribusi data dengan mekanisme tanpa pengawasan dan terbukti mengungguli metode konvensional [57]. Kang dkk. menerapkan ensemble learning berbasis RF untuk mengidentifikasi indikator depresi pada petani, menghasilkan sensitivitas tertinggi (90,0%) dibandingkan model *tree-based* lainnya [59].

Dalam ranah analisis fundamental dan prediksi keuangan, *Random Forest* juga menunjukkan kinerja yang kompetitif. Kaur dkk. membandingkan RF dengan jaringan saraf tiruan untuk mengevaluasi kekuatan prediksi antara analisis fundamental dan teknikal, dan menemukan bahwa RF memberikan kinerja prediktif yang sangat baik [60]. Wang mendemonstrasikan bahwa model *machine learning* termasuk *Random Forest* mampu memprediksi penurunan harga saham yang signifikan hingga lima tahun ke depan hanya dengan mengandalkan data fundamental perusahaan [61]. Penelitian klasifikasi stunting menggunakan metode support vector regression juga mendefinisikan *Random Forest* sebagai salah satu algoritma *machine learning* yang termasuk dalam kategori *supervised learning* yang membangun banyak pohon keputusan selama proses training dan mengeluarkan kelas yang merupakan modus dari kelas (klasifikasi) atau rata-rata prediksi (regresi) dari masing-masing pohon individu [64].

2.8 IndoBERT

2.8.1 Definisi IndoBERT

IndoBERT (Indonesian Bidirectional Encoder Representations from Transformers) merupakan model bahasa pra-latih (*pre-trained language model*) berbasis arsitektur Transformer yang dioptimalkan khusus untuk pemrosesan Bahasa Indonesia. Model ini dilatih pada korpus teks berbahasa Indonesia berskala besar sehingga mampu menghasilkan representasi kata yang bersifat *contextualized*, yaitu memahami makna suatu kata berdasarkan konteks kalimat di sekitarnya. IndoBERT dapat di-*fine-tune* untuk berbagai tugas pemrosesan bahasa alami (NLP) seperti deteksi ujaran kebencian, analisis sentimen, klasifikasi berita, hingga pemrosesan dokumen akademik. Hasil penelitian pada berbagai skenario menunjukkan performa unggul model ini dibandingkan pendekatan tradisional

maupun model multilingual lainnya, dengan tingkat akurasi yang konsisten tinggi pada data berbahasa Indonesia [65][66].

Learning rate ditetapkan dalam rentang $2e-5$ hingga $3e-5$ berdasarkan temuan bahwa *learning rate* merupakan parameter terpenting dalam optimasi kinerja jaringan saraf, di mana nilai yang terlalu kecil menyebabkan konvergensi lambat sementara nilai yang terlalu besar dapat menyebabkan *loss* berfluktuasi dan gagal konvergen [6][7][8], dan rentang ini juga umum digunakan dalam *fine-tuning* model *transformer* seperti BERT dan variannya untuk tugas klasifikasi teks [73][68][69]. *Batch size* dipilih dalam rentang 8, 16, atau 32 karena penelitian menunjukkan bahwa ukuran *batch* yang lebih kecil menghasilkan konvergensi yang lebih cepat menuju solusi yang baik dan memungkinkan model untuk mulai belajar lebih awal sebelum melihat seluruh data [6][7], sementara ukuran 8 hingga 32 merupakan nilai umum yang sering digunakan dalam *fine-tuning* model bahasa dengan keterbatasan memori GPU [41][43][49]. Jumlah *epoch* ditetapkan antara 3 hingga 10 untuk menghindari *overfitting* sekaligus memastikan model memiliki iterasi yang cukup untuk mempelajari pola data [30][13], dengan rentang 3-10 *epoch* menjadi praktik standar untuk tugas klasifikasi teks dan *fine-tuning* model *transformer* [73][67][68]. *Optimizer* Adam dengan *weight decay* 0,01 dipilih karena kemampuannya dalam mengadaptasi *learning rate* secara per parameter dan telah terbukti efektif dalam berbagai tugas pemrosesan bahasa alami [43][44][49], dengan AdamW menjadi pilihan utama ketika menggunakan *weight decay* karena memisahkan *weight decay* dari pembaruan gradien sehingga regularisasi lebih terkontrol [6][8]. *Learning rate scheduler* linear diterapkan untuk menurunkan *learning rate* secara bertahap selama pelatihan, yang sesuai dengan strategi *decay learning rate* yang terbukti meningkatkan performa model [7][8][30]. *Warmup steps* sebesar 10% dari total langkah pelatihan diterapkan untuk menstabilkan proses pelatihan awal dan mencegah pembaruan yang tidak stabil pada tahap awal pelatihan [41][45][49], di mana *warmup ratio* 0,1 merupakan nilai yang umum digunakan dalam *fine-tuning* model *transformer* [6][7]. *Max sequence length* 128 atau 256 dipilih untuk menyeimbangkan kapasitas menangkap kalimat dengan efisiensi komputasi, sebagaimana panjang sekuens ini umum digunakan dalam *fine-tuning* model BERT dan variannya untuk tugas klasifikasi teks bahasa Indonesia [66][67][73].

2.8.2 Arsitektur dan Cara Kerja IndoBERT

Arsitektur Transformer merupakan fondasi utama dari model BERT dan IndoBERT

yang dikembangkan berdasarkan prinsip *self-Attention Mechanism* [45]. Berbeda dengan pendekatan *Recurrent Neural Network* (RNN) yang memproses data secara sekuensial, Transformer memungkinkan pemrosesan paralel dengan menghitung hubungan antar seluruh token dalam satu kalimat secara simultan [45]. Hal ini menjadikan Transformer lebih efisien dalam menangkap konteks global dari sebuah teks [45]. Pada tahap *pre-processing*, teks mentah dibersihkan (*case folding*, penghapusan angka & tanda baca) dan dipersiapkan agar dapat diproses oleh model.

WordPiece Tokenization [72]

IndoBERT menggunakan tokenizer berbasis *subword* (WordPiece) dengan ukuran kosakata sekitar 30.000-35.000 token. Tokenizer ini memecah kata menjadi unit-unit kecil untuk menangani kata yang jarang muncul atau berada di luar kosakata:

$$\text{Tokenize}(s) = [t_1, t_2, \dots, t_n] \quad (35)$$

di mana setiap t_i merupakan token dari kosakata V_{Indo} .

Representasi Input IndoBERT [72][90]

Setelah tokenisasi, format input untuk klasifikasi dibangun dengan menambahkan token khusus [CLS] (untuk agregasi representasi kalimat) dan [SEP] (pemisah antarkalimat):

$$S = (["CLS"], \text{"teks"}, ["SEP"]) \quad (36)$$

dimana :

S: Sekuens input final yang akan dimasukkan ke model.

[CLS]: Token khusus di awal sekuens yang berfungsi sebagai representasi agregat dari seluruh kalimat (digunakan untuk tugas klasifikasi).

"teks": Kalimat asli yang sudah di-tokenisasi.

[SEP]: Token pemisah (separator) yang menandakan akhir dari sebuah kalimat atau pemisah antar kalimat.

Pada tahap labeling, setiap teks diberi label aktual (positif, netral, negatif) yang nantinya akan digunakan sebagai target dalam pelatihan.

Fase Pra-pelatihan dan Fine-tuning

Pembagian data menggunakan rasio 80% data latih, 10% data validasi, dan 10% data uji dalam implementasi model IndoBERT untuk analisis sentimen. Model IndoBERT dijalankan dalam dua fase, yaitu *pre-training* (belajar representasi bahasa secara umum) dan *fine-tuning* (penyesuaian parameter pada tugas spesifik, seperti klasifikasi sentimen) [91].

Proses labeling menggunakan model yang telah di-*fine-tune* untuk menentukan kelas

sentimen [67][69].

Dalam *supervised learning* untuk tugas klasifikasi, label atau target merupakan variabel dependen kategorikal yang harus ditransformasikan ke dalam bentuk numerik agar dapat diproses oleh algoritma. Salah satu metode transformasi yang paling umum digunakan adalah *one-hot encoding* [91]. Secara konseptual, *one-hot encoding* merepresentasikan setiap kelas sebagai vektor biner ortogonal, di mana hanya satu posisi yang bernilai 1 dan sisanya 0. Hal ini memastikan bahwa jarak antar kelas adalah sama secara Euclidean sehingga model tidak keliru mengasumsikan adanya hubungan ordinal atau hierarki di antara kelas-kelas tersebut [91]. Representasi ini sepenuhnya bergantung pada ground truth atau label aktual yang telah ditetapkan pada tahap pelabelan data, di mana setiap label kategori (misalnya positif, netral, negatif) dipetakan ke vektor satu-panas yang spesifik untuk digunakan sebagai target output selama proses pelatihan model. Penggunaan *one-hot encoding* untuk merepresentasikan label target merupakan praktik standar dalam supervised learning, khususnya ketika menggunakan fungsi aktivasi softmax dan categorical cross-entropy loss pada neural network [91].

Pada tahap training, mencakup *forward pass* hingga *backpropagation*.
Embedding Layer [72][90]

Setiap token ID diubah menjadi vektor berdimensi 768, yang merupakan penjumlahan dari tiga jenis *embedding*:

$$E(t_l) = V(t_l) + P(\text{position}_l) + S(\text{segment}_l) \quad (37)$$

(Token Embedding + Position Embedding + Segment Embedding).

Arsitektur Transformer dan Self-Attention [45]

Vektor *embedding* diproyeksikan menjadi Query (Q), Key (K), dan Value (V). Skor perhatian dihitung menggunakan *scaled dot-product attention*:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (38)$$

Dimana:

- Q : Query (matriks pertanyaan), representasi vektor yang "bertanya" kepada token lain.
- K : Key (matriks kunci), representasi vektor yang "dijawab" oleh Query.
- V : Value (matriks nilai), representasi vektor yang berisi informasi aktual yang akan diambil.

QK^T : Perkalian matriks Query dengan transpos Key untuk menghitung skor kecocokan antar semua pasangan token.

$\sqrt{(d_k)}$: Akar kuadrat dari d_k (dimensi dari vektor Key). Berfungsi sebagai skala pembagi agar gradien tidak terlalu kecil (menghindari vanishing gradient).

Softmax : Fungsi aktivasi yang mengubah skor mentah menjadi probabilitas (total bobot = 1).

Proses ini diulang dalam 12 kepala (*multi-head attention*) secara parallel[45][72] , lalu hasilnya digabung:

$$\text{MultiHead}(X) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O \quad (39)$$

dimana :

X : Matriks input yang masuk ke lapisan multi-head (biasanya berupa vektor embedding + hasil normalisasi sebelumnya).

head_i : Hasil perhitungan *Attention*(Q_i, K_i, V_i) dari kepala perhatian ke- i .

h : Jumlah kepala perhatian (dalam IndoBERT/BERT base, nilai $h = 12$).

Concat : Operasi penggabungan (menggabungkan semua hasil head secara horizontal).

W^O : Matriks bobot proyeksi akhir (Output Weight Matrix) yang digunakan untuk mencampur informasi dari semua kepala menjadi satu dimensi vektor akhir.

Feed-Forward Network (FFN) [88]

Setelah *self-attention*, output melewati lapisan FFN yang terdiri dari dua transformasi linear dengan aktivasi GELU[45][88]:

$$\text{FFN}(x) = \text{GELU}(xW_1 + b_1)W_2 + b_2 \quad (40)$$

dimana :

x : Vektor input yang keluar dari lapisan *multi-head attention* (setelah ditambahkan residu dan dinormalisasi).

W_1 : Matriks bobot linear pertama (memproyeksikan dimensi 768 menjadi lebih besar, misal 3072).

b_1 : Vektor bias (intercept) pertama.

GELU : *Gaussian Error Linear Unit*, fungsi aktivasi non-linear.

W_2 : Matriks bobot linear kedua (memproyeksikan kembali ke dimensi awal 768).

b_2 : Vektor bias kedua.

Residual Connection (Add) [42][45]

Untuk mengatasi *vanishing gradient*, mekanisme *skip connection* menambahkan input awal ke output sublapisan:

$$x_{out} = x_{in} + \text{SubLayer}(x_{in}) \quad (41)$$

dimana :

x_{in} : Vektor input asli sebelum masuk ke sublapisan (misal sebelum *Self-Attention* atau sebelum FFN).

$\text{SubLayer}(x_{in})$: Hasil olahan dari sublapisan (bisa berupa output *Self-Attention* atau output FFN).

x_{out} : Vektor output akhir yang merupakan penjumlahan langsung, berfungsi mencegah hilangnya gradien.

Layer Normalization [89]

Setelah operasi *Add*, aktivasi dinormalisasi secara independen per sampel untuk menstabilkan pelatihan:

$$\text{LN}(x) = \gamma \odot \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta \quad (42)$$

Dimana :

x : Vektor aktivasi yang akan dinormalisasi.

μ : Nilai rata-rata (mean) dari seluruh elemen dalam vektor x .

σ^2 : Nilai varians (ragam) dari seluruh elemen dalam vektor x .

ϵ (**epsilon**) : Konstanta kecil (10^{-5}) untuk mencegah pembagian dengan nol.

γ (**gamma**) : Parameter skala (scale) yang bisa dipelajari, untuk mengatur rentang output.

β (**beta**) : Parameter geseran (shift) yang bisa dipelajari, untuk mengatur nilai tengah output.

$\odot$: Operasi perkalian elemen-bijak (element-wise multiplication).

Klasifikasi dengan Softmax [91]

Vektor keluaran dari token [CLS] pada lapisan terakhir diubah menjadi logits melalui lapisan linear, lalu diubah menjadi probabilitas menggunakan fungsi *softmax* untuk 3 kelas sentimen:

$$P = \text{softmax}(h_{[\text{CLS}]} W^T) \quad (43)$$

dimana :

P : Vektor probabilitas untuk 3 kelas sentimen (positif, netral, negatif), totalnya = 1.

$h_{[CLS]}$: Vektor keluaran terakhir dari token [CLS] setelah melewati 12 lapisan Transformer.

W^T : Transpos dari matriks bobot lapisan linear klasifikasi. Perkalian $h * W^T$ menghasilkan 3 angka mentah (*logits*).

Cross-Entropy Loss Function [91]

Fungsi kerugian untuk mengukur selisih antara probabilitas prediksi dan label aktual (satunya kelas benar):

$$L = - \sum_{i=1}^K y_i \log(\hat{y}_i) \quad (44)$$

dimana :

L : Nilai *loss* (kerugian) yang akan diminimalkan. Semakin kecil, semakin baik.

K : Jumlah kelas sentimen, yaitu **3** (positif, netral, negatif).

i : Indeks kelas (1, 2, atau 3).

y_i : Label aktual (ground truth) dalam bentuk *one-hot encoding*. Hanya satu kelas yang bernilai 1, sisanya 0.

$\hat{y}_i$ (baca: y-topi) : Probabilitas prediksi model untuk kelas ke-i (nilai dari vektor P).

Gradient Descent dan Backpropagation [91]

Backpropagation menghitung gradien fungsi loss terhadap seluruh parameter menggunakan aturan rantai. Gradien untuk lapisan softmax terhadap logit adalah

$$\frac{\partial L}{\partial \text{logit}} = \hat{y} - y \quad (45)$$

dimana :

∂L : Turunan parsial dari fungsi *loss*.

∂logit : Turunan terhadap nilai logit mentah (sebelum softmax).

$\hat{y}$: Vektor probabilitas prediksi.

y : Vektor *one-hot* label aktual.

Bobot kemudian diperbarui menggunakan *Gradient Descent*:

$$W_{\text{baru}} = W_{\text{lama}} - \eta \cdot \frac{\partial L}{\partial W} \quad (46)$$

Dimana:

W_{baru} : Bobot/model parameter yang telah diperbarui.

W_{lama} : Bobot/model parameter sebelum diperbarui.

η (**eta**) : *Learning rate*, yaitu ukuran langkah pembaruan (seberapa besar lompatan yang diambil).

$\partial L / \partial W$: Gradien *loss* terhadap bobot W (menunjukkan arah perubahan yang harus dilakukan).

Proses Training dalam Batch (Mini-batch Gradient Descent) [91]

Dataset tidak dilatih sekaligus, tetapi dibagi menjadi batch kecil [91]. Untuk setiap batch, model melakukan forward pass, menghitung loss, backpropagation, lalu memperbarui bobot [91]. Proses ini diulang untuk beberapa epoch [91].

Setelah pelatihan selesai, model digunakan untuk memprediksi sentimen dari data baru.

Forward Pass (Inferensi)

Proses yang sama seperti forward pass pada saat training yang terdiri dari Tokenisasi, Embedding, 12 Lapisan Transformer, dan Lapisan Linear, dijalankan tanpa menghitung gradien [91].

Confidence Score dan Evaluasi [92]

Tingkat keyakinan model dihitung sebagai probabilitas maksimum dari keluaran *softmax*, dikonversi ke persentase:

$$\text{Confidence} = \max(P) \times 100\% \quad (47)$$

dimana :

Confidence : Persentase keyakinan model terhadap prediksinya.

$\max(P)$: Nilai probabilitas tertinggi di antara 3 kelas yang dihasilkan softmax

2.8.3 Framework IndoBERT

IndoBERT (*Indonesian Bidirectional Encoder Representations from Transformers*) merupakan model bahasa berbasis *transformer* yang dipra-latih (*pre-trained*) pada korpus berbahasa Indonesia berukuran besar, sehingga mampu memahami konteks dan relasi antarkata secara mendalam [67]. Kerangka kerja ini memungkinkan *fine-tuning* untuk berbagai tugas pemrosesan bahasa alami (*Natural Language Processing*, NLP). Audyna dkk. memanfaatkan IndoBERT untuk analisis sentimen berbasis aspek (*Aspect-Based Sentiment*

Analysis, ABSA) pada diskusi kendaraan listrik di media sosial X, menunjukkan keunggulannya dalam menangkap nuansa opini publik [65]. Sementara itu, Agam dkk. membandingkan *transformer* dengan RNN untuk klasifikasi dokumen berita berbahasa Indonesia, dan menemukan bahwa arsitektur seperti IndoBERT mencapai akurasi pelatihan hingga 98,5% serta generalisasi yang lebih baik dibandingkan RNN [66].

Optimalisasi kerangka IndoBERT dapat dilakukan melalui variasi arsitektur, *fine-tuning*, serta *hybrid labeling*. Shaw mengeksplorasi berbagai variasi IndoBERT (indobenchmark, indolem, cendol, distilindolem) untuk analisis sentimen bahasa Indonesia, dan menunjukkan bahwa model yang dilatih dengan data berukuran besar serta strategi *self-supervised* menghasilkan peningkatan akurasi yang signifikan [67]. Septian dkk. menerapkan IndoBERT untuk menganalisis sentimen publik terhadap putusan Mahkamah Konstitusi mengenai batas usia calon presiden dan wakil presiden, dan berhasil mengklasifikasikan sentimen dengan tingkat akurasi yang tinggi [68]. Aprilah dkk. mengonfirmasi bahwa *fine-tuning* IndoBERT menggunakan pustaka *Hugging Face* mampu mengatasi tantangan semantik kompleks dan ketidakseimbangan kelas dalam ABSA, mencapai akurasi global 97% [69]. Adhim dan Cahyono selanjutnya mengoptimalkan IndoBERT melalui *fine-tuning* dan *hybrid labeling* untuk analisis sentimen *Fear of Missing Out* (FOMO) di media sosial, dan memperoleh akurasi 94,50% serta F1-Score 94,52%, mengungguli model BiLSTM dan SVM [70].

Selain analisis sentimen umum, kerangka IndoBERT juga diadaptasi untuk topik-topik sensitif dan perbandingan dengan model klasik. Kono dkk. membandingkan IndoBERT yang telah di-*fine-tune* dengan *Logistic Regression* dan *Linear SVM* untuk analisis sentimen publik terhadap isu korupsi di Indonesia, dan menunjukkan bahwa IndoBERT menghasilkan performa terbaik dalam mengklasifikasikan sentimen negatif dan positif [71]. Dua inovasi tambahan memperkaya kerangka ini. Pertama, Ridho dan Yulianti mengusulkan deteksi berita palsu (*hoax detection*) menggunakan IndoBERT Large yang dikombinasikan dengan teknik *oversampling* SMOTE, mencapai akurasi 98% pada berita berbahasa Indonesia [72]. Kedua, Wajdi dkk. mengembangkan model hibrida IndoBERT yang diintegrasikan dengan *Gated Recurrent Unit* (GRU) untuk deteksi sarkasme dalam teks bahasa Indonesia, meningkatkan presisi secara signifikan dibandingkan model *transformer* tunggal [73]. Dengan demikian, kerangka IndoBERT sangat adaptif, mulai dari ABSA [65][69], klasifikasi dokumen [66], variasi model [67], analisis sentimen hukum dan sosial [68][70][71], hingga deteksi disinformasi [72] dan sarkasme [73].

2.9 Hybrid Method

2.9.1 Definisi Hybrid Method

Augmented Hybrid Method merupakan pendekatan prediksi keuangan yang menggabungkan model *Artificial Intelligence* dengan teknik augmentasi data atau penyisipan wawasan semantik untuk meningkatkan akurasi prediksi saham. Menurut Cohen dkk. [74], model hybrid yang mengintegrasikan Recurrent Neural Network (RNN) dan *Gated Recurrent Unit* (GRU) dengan data harga historis serta indikator teknikal (MA, SMA, EMA) mampu memprediksi saham di Bursa Efek Indonesia dengan nilai R^2 mencapai 0,98-0,99 pada saham bank utama. Sementara itu, Cohen dkk. [75] mendefinisikan *semantic-Augmented Hybrid* approach sebagai model yang menggabungkan kecerdasan semantik dari Large Language Models (LLM) seperti ChatGPT-4o dengan algoritma *machine learning* tradisional pada tiga kerangka prediktif (fundamental, teknikal, dan berbasis entropi), di mana pendekatan fundamental mencapai kinerja optimal jika didasarkan terutama pada wawasan semantik LLM, sedangkan pendekatan teknikal berkinerja terbaik hanya dengan ML. Kedua studi ini menegaskan bahwa augmentasi, baik melalui integrasi arsitektur ganda maupun penyisipan pemahaman semantic, mampu menangkap dinamika pasar yang lebih kompleks dibandingkan model tunggal [74][75].

2.9.2 Attention Mechanism

Attention Mechanism adalah mekanisme dalam *Deep Learning* yang memungkinkan model memberikan bobot perhatian yang berbeda pada setiap bagian dari input berdasarkan tingkat relevansinya terhadap output yang ingin diprediksi [79]. Mekanisme ini terinspirasi dari sistem kognitif manusia di mana otak secara selektif memfokuskan perhatian pada informasi penting sambil mengabaikan informasi yang kurang relevan [80]. Dalam pemrosesan sekuensial seperti teks atau deret waktu, *Attention Mechanism* memungkinkan model untuk secara dinamis memilih informasi mana yang paling penting dari seluruh urutan input, mengatasi keterbatasan RNN dan LSTM yang cenderung melupakan informasi dari langkah yang lebih awal [79].

Berdasarkan kerangka umum yang diperkenalkan dalam literatur, *Attention Mechanism* secara matematis didefinisikan melalui fungsi yang memetakan query, key, dan value menjadi output berupa *weighted sum* dari *value* [79]. Formulasi umum *Attention Mechanism* dinyatakan sebagai berikut:

$$Attention(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (48)$$

Dimana:

$Q(query)$	= matriks yang merepresentasikan informasi yang ingin dicari
$K(key)$	= matriks yang tersedia
$V(value)$	= nilai data input
d_k	= dimensi dari <i>key</i> yang berfungsi sebagai faktor skala

Pembagian dengan $\sqrt{d_k}$ bertujuan untuk mencegah nilai hasil *dot product* menjadi terlalu besar, yang dapat menyebabkan nilai gradien menjadi sangat kecil setelah melewati fungsi *softmax*. Cara kerja *Attention Mechanism* dimulai dengan menghitung skor kompatibilitas antara setiap query dan key melalui operasi dot product (*Scaled Dot-Product Attention*) [79]. Skor ini mengindikasikan seberapa relevan suatu posisi input terhadap posisi output yang sedang diproses, di mana semakin tinggi skor dot product menunjukkan semakin besar relevansi antara kedua posisi tersebut [79]. Langkah kedua adalah melakukan penskalaan (*scaling*) dengan membagi skor dot product dengan akar kuadrat dari dimensi key untuk menjaga stabilitas numerik selama proses pelatihan [79]. Langkah ketiga adalah menerapkan fungsi softmax pada skor yang telah diskalakan untuk menghasilkan *attention weights* berupa probabilitas yang berjumlah satu [79]. Bobot ini menentukan seberapa besar perhatian model terhadap setiap posisi input, sehingga posisi dengan bobot lebih tinggi akan mendapatkan fokus yang lebih besar dalam menghasilkan output [80]. Langkah keempat adalah mengalikan setiap value dengan *attention weight* yang sesuai, kemudian menjumlahkan semuanya untuk menghasilkan output akhir berupa *weighted sum* [79].

Secara framework, *Attention Mechanism* terdiri dari tiga komponen utama: feature model, query model, dan *attention* model itu sendiri [79]. Feature model bertugas mengekstrak *feature vectors* dari input asli, yang kemudian akan diubah menjadi keys dan values melalui transformasi linear menggunakan matriks bobot yang dapat dipelajari selama pelatihan [79]. Query model berfungsi menghasilkan *query vector* yang biasanya berasal dari hidden state decoder atau representasi kontekstual lainnya, yang memberitahu *attention* model feature vectors mana yang harus diperhatikan [80].

Attention model merupakan inti dari *framework* yang menghitung *attention weights* dan menghasilkan *context vector* [80]. Framework ini memungkinkan model untuk secara fleksibel melihat kembali ke seluruh bagian input setiap kali menghasilkan output, sehingga sangat efektif untuk tugas-tugas seperti *machine translation*, *text summarization*, dan *time*

series forecasting [79][80]. *Framework* ini menjadi fondasi bagi model-model besar seperti BERT, GPT, dan berbagai turunannya termasuk IndoBERT untuk pemrosesan bahasa alami [79][80].

Dalam implementasi *Attention Mechanism*, setiap vektor input x_i diproyeksikan menjadi tiga representasi berbeda:

$$q_i = x_i W_Q, k_i = x_i W_K, v_i = x_i W_V \quad (49)$$

Matriks W_Q , W_K , dan W_V diinisialisasi secara acak dan dipelajari selama proses pelatihan menggunakan *backpropagation*. Proyeksi terpisah ini memungkinkan model untuk mempelajari representasi yang berbeda untuk peran *query*, *key*, dan *value*, menciptakan asimetri yang penting untuk fleksibilitas model [79]. *Query* (Q) mewakili informasi yang dibutuhkan oleh token/fitur saat ini, *Key* (K) mewakili informasi yang tersedia dari token/fitur lain, dan *Value* (V) mewakili konten aktual yang akan diintegrasikan.

Skor *attention* dihitung melalui perkalian titik (*dot product*) antara matriks *Query* dan transpose matriks *Key*:

$$S = QK^T \quad (50)$$

Skor ini mencerminkan kesesuaian atau kompatibilitas antara setiap pasangan fitur dalam sistem. Setelah diskalakan dengan $\sqrt{d_k}$ dimana $\sqrt{d_k}$ adalah jumlah dimensi dalam vektor, fungsi *softmax* diterapkan pada setiap baris untuk menghasilkan bobot probabilitas yang berjumlah 1:

$$\text{Softmax}(s_i) = \frac{e^{s_i}}{\sum_j e^{s_j}} \quad (51)$$

Pada praktiknya, untuk stabilitas numerik, perhitungan *softmax* sering dilakukan dengan mengurangi nilai maksimum terlebih dahulu:

$$\text{Softmax}(s_i) = \frac{e^{s_i - \max(s)}}{\sum_j e^{s_j - \max(s)}} \quad (52)$$

Output *attention* Z diperoleh melalui perkalian bobot *softmax* dengan matriks *Value*:

$$Z = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (53)$$

Proses ini menghasilkan representasi baru di mana setiap fitur merupakan agregasi tertimbang dari semua fitur lain, dengan bobot yang mencerminkan relevansi relatifnya.

Dengan kata lain, Z adalah rata-rata tertimbang dari nilai-nilai V , di mana bobotnya ditentukan oleh fungsi kompatibilitas antara *query* dan *key* yang bersangkutan [79].

Dalam arsitektur Transformer modern, *Attention Mechanism* diperluas menjadi *Multi-Head Attention*, di mana beberapa kepala *attention* beroperasi secara paralel [79].

$$\text{MultiHead}(X) = \text{Concat}(\text{head}_1, \dots, \text{head}_H)W^O \quad (54)$$

$$\text{head}_h = \text{Attention}(XW_Q^h, XW_K^h, XW_V^h) \quad (55)$$

Setiap head_h memiliki matriks proyeksi W_Q^h , W_K^h , W_V^h yang terpisah, memungkinkan model untuk fokus pada aspek informasi yang berbeda secara simultan. Hasil dari semua kepala digabung (*concatenate*) dan diproyeksikan melalui matriks W^O untuk menghasilkan output akhir [79].

Setelah *Attention Mechanism*, arsitektur Transformer menerapkan lapisan *Fully Connected* (Feed-Forward Network) yang bekerja secara independen pada setiap posisi [79]:

$$\text{FFN}(x) = \text{GELU}(xW_1 + b_1)W_2 + b_2 \quad (56)$$

Lapisan ini menambahkan kapasitas non-linearitas dan transformasi fitur yang dipelajari. Dalam prediksi harga saham, setelah melalui seluruh lapisan Transformer dan *attention*, representasi akhir dipetakan ke nilai prediksi harga menggunakan lapisan linear:

$$\hat{y} = Z_{\text{final}}W_{\text{out}} + b_{\text{out}} \quad (57)$$

Proses *backpropagation* pada seluruh lapisan ini memungkinkan model untuk mempelajari bobot-bobot secara *end-to-end*. Gradien dari fungsi *loss* dihitung menggunakan *chain rule* dan dipropagasikan mundur melalui semua lapisan, mulai dari lapisan output, melalui lapisan *feed-forward*, *Attention Mechanism*, hingga lapisan *embedding*. Dengan pendekatan *gradient descent*, matriks bobot W_Q , W_K , W_V , W^O , W_1 , W_2 , dan W_{out} diperbarui secara iteratif untuk meminimalkan kesalahan prediksi.

Penetapan ambang batas (*threshold*) dalam sistem klasifikasi mempengaruhi kualitas keputusan yang dihasilkan. Dalam klasifikasi biner, penggunaan *threshold* tunggal pada 0.5

seringkali dianggap kurang optimal karena tidak mempertimbangkan ketidakpastian prediksi dan asimetri biaya kesalahan antar kelas. pendekatan *thresholding adaptif* berbasis *cost-sensitive learning* dan *bayesian optimization* untuk mengoptimalkan ambang batas klasifikasi pada data dengan distribusi kelas yang tidak seimbang. Hasil penelitian menunjukkan bahwa penetapan zona *Hold* pada rentang 40-60% yang merupakan zona penolakan keputusan (*reject option*) adalah strategi yang sah secara metodologis untuk menghindari sinyal perdagangan yang ambigu, di mana skor di bawah 40% dianggap cukup yakin untuk diklasifikasikan sebagai *Sell* dan skor di atas 60% sebagai *Buy* [80].

2.10 Pengembangan Aplikasi

Metode *Waterfall* merupakan pendekatan pengembangan perangkat lunak yang bersifat sekuensial dan terstruktur, di mana setiap tahapan seperti analisis kebutuhan, desain sistem, implementasi, pengujian, hingga pemeliharaan harus diselesaikan secara bertahap sebelum melanjutkan ke fase berikutnya[81]. Pemanfaatan *framework* Laravel dalam studi tersebut membuktikan bahwa pendekatan sekuensial ini mampu menghasilkan aplikasi berbasis *website* yang andal sekaligus mendukung peningkatan kualitas secara lebih luas [81].

Selain implementasi pada platform *website*, metode *Waterfall* juga terbukti efektif untuk pengembangan aplikasi *mobile* dan sistem penjualan. Sebuah aplikasi catatan (*notes*) berbasis Android dikembangkan melalui lima tahap *Waterfall* analisis kebutuhan, desain, implementasi, pengujian, dan pemeliharaan yang berhasil menghadirkan fitur pengelolaan catatan serta pengingat dengan tingkat efisiensi dan efektivitas yang tinggi bagi pengguna [81]. Di sisi lain, penerapan metode yang sama pada aplikasi penjualan produk tradisional berbasis *website* menghasilkan sistem yang dilengkapi dengan fitur *login*, keranjang belanja, dan metode pembayaran yang terintegrasi, sehingga mampu mendukung digitalisasi usaha mikro, kecil, dan menengah (UMKM) [82].

Streamlit adalah *framework open-source* berbasis Python yang dirancang untuk membangun aplikasi *machine learning* dan *data science* secara cepat dan interaktif tanpa perlu menguasai teknologi *front-end* seperti HTML, CSS, atau JavaScript [83]. Framework ini mengubah *script* Python menjadi aplikasi web interaktif dengan pendekatan *reactive scripting*, di mana setiap interaksi pengguna menjalankan ulang seluruh *script* sehingga konsistensi data dan tampilan tetap terjaga [83]. Streamlit telah banyak diadopsi dalam berbagai implementasi, termasuk prediksi penyakit, deteksi *deepfake*, monitoring

lingkungan, dan sistem rekomendasi, yang membuktikan efektivitasnya dalam menjembatani kesenjangan antara *data scientist* dan kebutuhan penyajian hasil yang mudah diakses oleh pengguna non-teknis [81], [83].

Draw.io merupakan perangkat lunak diagram serbaguna yang berperan penting dalam tahap awal perancangan *interface* pengguna (UI), khususnya untuk membuat flowchart, diagram UML, dan prototipe [84]. Dalam penggambaran UI, fungsi utama *draw.io* adalah menjembatani konsep abstrak menjadi representasi visual yang sistematis, di mana perancang dapat dengan mudah mengatur tata letak elemen, menentukan navigasi antarhalaman, serta mendokumentasikan alur proses bisnis sebelum diterjemahkan ke dalam desain visual yang lebih detail menggunakan alat bantu lain seperti *Figma* [84]. Dengan memanfaatkan fitur *drag-and-drop*, templat yang beragam, serta kemampuan kolaborasi *real-time*, *draw.io* memastikan bahwa fondasi interaksi sistem telah matang dan mudah dipahami oleh seluruh tim pengembang, sehingga pada akhirnya berkontribusi pada efisiensi dan efektivitas proses desain UI secara keseluruhan [84].

2.11 Metode Evaluasi

Metode evaluasi kecerdasan buatan (AI) terus berkembang, seperti yang ditunjukkan oleh dua studi terkini. Kerangka CDIO (*Conceive-Design-Implement-Operate*) untuk mengevaluasi naskah ilmiah yang dihasilkan AI secara sistematis, memastikan fondasi etis, akurasi ilmiah, serta komunikasi temuan yang efektif [85]. Sementara itu, Zhu dkk. (2026) mengeksplorasi penggunaan model AI generatif seperti GPT-4o dan DeepSeek-V3 untuk menilai kualitas makalah di bidang ilmu sosial [85]. Studi kedua ini menemukan bahwa AI memiliki korelasi terbatas dengan metrik sitasi, cenderung bias terhadap makalah dengan terminologi teknis yang eksplisit, serta meremehkan makalah yang memiliki abstraksi teoretis tinggi atau terikat kuat dengan sosial budaya [85]. Berikut rumus evaluasi [85]:

1. Root Mean Square Error (RMSE)

Rumus RMSE dinyatakan pada Persamaan (37):

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2} \quad (58)$$

Dimana:

- n = Jumlah Data
- $\hat{y}_i$ = Nilai Prediksi Ke- i

y_i = Nilai Aktual Ke- i

1. Normalized Root Mean Square Error (NRMSE)

Normalized Root Mean Square Error (NRMSE) merupakan salah satu metrik evaluasi yang digunakan untuk mengukur akurasi prediksi suatu model. Metrik ini pada dasarnya adalah bentuk normalisasi dari *Root Mean Square Error* (RMSE) yang bertujuan untuk mengatasi kelemahan RMSE yang bergantung pada skala data. Dengan normalisasi, NRMSE menjadi tidak memiliki satuan (*unit-independent*), sehingga memungkinkan perbandingan performa model secara lebih objektif antar variabel atau dataset yang berbeda.

Secara matematis, NRMSE didefinisikan sebagai rasio antara RMSE dengan nilai rata-rata dari data observasi:

$$NRMSE = \frac{RMSE}{\bar{y}} \quad (59)$$

Dimana:

$RMSE$ = *Root Mean Square Error*

$\bar{y}$ = nilai rata-rata dari data aktual

Nilai NRMSE yang ideal adalah 0 (nol), yang mengindikasikan bahwa model menghasilkan prediksi yang sempurna tanpa kesalahan. Semakin kecil nilai NRMSE, semakin baik performa model dalam melakukan prediksi [86].

2. Rumus Probabilitas Prediksi untuk Kelas Tertentu

Dinyatakan pada Persamaan:

$$\hat{P}(y = k) = \frac{\sum_{t=1}^N \mathbb{I}(h_t(x) = k)}{N} \quad (60)$$

Dimana:

$\hat{P}(y = k)$ = Probabilitas estimasi bahwa sampel x termasuk ke dalam kelas k

N = Jumlah total pohon dalam hutan (*Random Forest*)

t = Indeks pohon ke- t

$h_t(x)$ = Hasil prediksi (kelas) dari pohon ke- t untuk sampel x

$\mathbb{I}(\cdot)$ = Fungsi indikator (bernilai 1 jika kondisi benar, 0 jika salah)

k = Label kelas target (misal: 0 = *Overvalued*, 1 = *Undervalued*)

3. Margin Keyakinan (*Confidence*) untuk Multikelas

Rumus Margin Keyakinan untuk Multikelas dinyatakan pada Persamaan :

$$\text{Confidence} = \max_k(\hat{P}(k)) - \max_{j \neq k}(\hat{P}(j)) \quad (61)$$

Dimana:

Confidence = Tingkat keyakinan model terhadap prediksi yang dipilih

$\max_k(\hat{P}(k))$ = Probabilitas tertinggi di antara semua kelas (kelas pemenang)

$\max_{j \neq k}(\hat{P}(j))$ = Probabilitas tertinggi kedua di antara kelas lainnya (kelas pesaing terdekat)

2.12 Library Python

Data Processing yakni `pandas==2.0.3`, digunakan untuk manipulasi dan pemrosesan data keuangan, seperti membersihkan, menyusun, dan menganalisis data historis saham yang diambil dari pasar modal [4][14][15]. `numpy==1.24.3`, digunakan bersama `pandas` untuk melakukan komputasi numerik dan pra-pemrosesan data, seperti normalisasi dan transformasi data deret waktu [6][14]. `yfinance==0.2.28`, digunakan untuk mengunduh data pasar saham historis secara real-time dari Yahoo Finance sebagai sumber data utama dalam pengujian model prediksi [14][15].

Machine Learning & Deep Learning yakni `scikit-learn==1.3.0`, digunakan untuk berbagai tugas *Machine Learning* (misal: normalisasi data dengan `MinMaxScaler`, evaluasi metrik, dan implementasi algoritma seperti *Random Forest*) guna mendukung analisis teknis dan fundamental [28][30][59]. `tensorflow==2.13.0`, digunakan sebagai framework utama untuk membangun dan melatih model *Deep Learning*, seperti LSTM, dalam memprediksi pergerakan harga saham dan risiko investasi [7][13][41]. `xgboost==1.7.6`, digunakan sebagai algoritma *gradient boosting* yang efisien dan akurat untuk tugas klasifikasi serta prediksi pergerakan saham, terutama dalam pendekatan ensemble learning [28][58].

NLP & Sentiment Analysis yakni `transformers==4.31.0`, menyediakan arsitektur model transformer (seperti BERT) yang canggih untuk mengekstraksi sentimen dari data tekstual guna meningkatkan akurasi prediksi saham [45][66][73]. `torch==2.0.1`, digunakan sebagai backend *Deep Learning* untuk menjalankan model transformer dan arsitektur neural network lainnya dalam analisis sentimen serta prediksi deret waktu [41][45]. `torchvision==0.15.2`, meskipun umum untuk visi komputer, library ini dapat dimanfaatkan dalam ekosistem PyTorch untuk menangani data non-tabular atau sebagai bagian dari infrastruktur *Deep Learning* secara umum [42].

indobenchmark==0.2.0, digunakan untuk menguji dan mengevaluasi model NLP berbahasa Indonesia, seperti IndoBERT, dalam tugas analisis sentimen untuk pasar saham Indonesia [66][72][73].

Web Scraping & API yakni requests==2.31.0, digunakan untuk melakukan permintaan HTTP guna mengakses API keuangan atau mengunduh konten halaman web statis dalam proses pengumpulan data [36]. beautifulsoup4==4.12.2, digunakan untuk parsing HTML dan mengekstrak data terstruktur dari halaman web yang statis [36]. selenium==4.11.2, digunakan untuk mengotomatiskan browser dan mengekstrak data dari situs web yang kontennya di-render secara dinamis dengan JavaScript, terutama dalam pemantauan sentimen *real-time* [36].

Visualization yakni streamlit==1.25.0, digunakan untuk membangun aplikasi web interaktif yang memungkinkan pengguna menjelajahi data, memvisualisasikan tren, dan melihat hasil prediksi saham secara langsung [82]. plotly==5.15.0, digunakan untuk membuat visualisasi data interaktif dan dinamis, seperti grafik harga saham, guna mendukung analisis yang lebih mendalam [82][84]. matplotlib==3.7.2, digunakan untuk membuat plot dan grafik statis guna menganalisis tren data, memvisualisasikan kinerja model, serta menyajikan hasil evaluasi [82][84].

Utilities yakni python-dotenv==1.0.0, digunakan untuk mengelola variabel lingkungan (seperti API key) dengan aman dari file .env guna mendukung praktik pengembangan perangkat lunak yang aman [80][81]. pyyaml==6.0, digunakan untuk membaca dan menulis file konfigurasi dalam format YAML, yang umum untuk mengatur parameter model atau keseluruhan pipeline penelitian [34][80]. joblib==1.3.2, digunakan untuk menyimpan (*serialize*) dan memuat (*deserialize*) model *machine learning* yang telah dilatih, sehingga efisien untuk digunakan kembali dalam pengujian dan implementasi [81].

2.12 Penelitian Terdahulu

Untuk memperjelas posisi penelitian ini di antara hasil-hasil sebelumnya, berikut disajikan ringkasan penelitian terdahulu yang relevan dengan prediksi harga saham menggunakan pendekatan *Deep Learning* dan *Hybrid*.

Tabel 2.1 Keterbatasan Penelitian Terdahulu

No	Penulis (Tahun)	Metode	Hasil Penelitian	Keterbatasan
1	Dwiandiyanta dkk. (2025) [6], [74]	Hybrid RNN-GRU dengan indikator teknikal (MA, SMA, EMA)	RNN-GRU memberikan akurasi prediksi lebih tinggi	Hanya menggunakan data teknikal; belum mengintegrasikan analisis fundamental

			dibanding model tunggal.	dan sentimen
2	Cohen dkk. (2025) [75]	<i>Semantic-Augmented Hybrid: LLM (ChatGPT-4o) + machine learning tradisional</i>	Integrasi LLM meningkatkan performa prediksi dan strategi investasi.	Ketiga modalitas tidak digabungkan secara simultan dalam satu model adaptif; bobot tetap
3	Badar dkk. (2025) [14]	Autoencoder + CNN-LSTM dengan <i>Attention Mechanism</i>	Meningkatkan kemampuan menangkap pola pasar yang kompleks dan nonlinier.	Fokus pada cryptocurrency dan saham individual; belum mengintegrasikan fundamental dan sentimen
4	Li & Hu (2024) [13]	<i>Augmented Hybrid Deep Learning</i> dengan <i>Attention Mechanism</i>	Integrasi sentimen investor meningkatkan akurasi prediksi saham.	Belum mengintegrasikan analisis fundamental dan sentimen berbasis teks secara bersamaan
5	Peujio Fozap (2025) [12]	Optimasi <i>hyperparameter</i> pada model <i>hybrid Deep Learning</i> (CNN-LSTM, dll)	Optimasi hyperparameter meningkatkan akurasi dan menurunkan error prediksi.	Model masih cenderung <i>rigid</i> terhadap perubahan kondisi pasar karena tidak ada mekanisme adaptasi dinamis
6	Chen dkk. (2022) [10]	<i>Deep Learning</i> + analisis sentimen (NLP)	Sentimen meningkatkan performa prediksi dibanding data harga saja.	Sentimen hanya dari satu sumber (media sosial); mekanisme <i>fusion</i> bersifat tetap (rata-rata)