

BAB II

KAJIAN LITERATUR

2.1 Prediksi Harga Saham

Saham atau surat berharga merupakan sekuritas yang menjadi bukti kepemilikan atau keterlibatan seseorang maupun suatu entitas dalam sebuah perusahaan [20]. Saham sebagai instrumen keuangan memiliki karakteristik risiko yang tinggi, potensi imbal hasil yang besar, serta fleksibilitas dalam perdagangannya, sehingga menjadi pilihan yang menarik bagi banyak investor [21]. Investor saham berupaya memilih perusahaan yang tepat untuk berinvestasi untuk memperoleh keuntungan dengan tingkat risiko yang relatif rendah [22]. Namun, harga saham bersifat fluktuatif sehingga pergerakannya yang tidak menentu dapat menimbulkan kerugian jika tidak diprediksi dengan tepat [23]. Oleh karena itu, Prediksi harga saham yang akurat dapat membantu investor dalam merancang strategi investasi yang optimal serta meminimalkan risiko [24].

Prediksi harga saham merupakan proses memperkirakan nilai saham di masa depan dengan memanfaatkan data historis [25]. Prediksi harga saham menjadi aspek penting dalam analisis sistem keuangan, mengingat pergerakan pasar saham dapat merepresentasikan kondisi ekonomi suatu negara hingga perekonomian dunia [26]. Namun, perubahan harga saham umumnya menunjukkan pola yang bersifat *non-linier*, sehingga proses peramalan menjadi lebih kompleks dan sulit dilakukan secara akurat [5]. Oleh karena itu, pendekatan *machine learning* dan *deep learning* semakin banyak digunakan karena kemampuannya dalam memprediksi harga secara lebih efektif dan memberikan prediksi yang lebih akurat [27].

2.2 Preprocessing Data

Preprocessing data merupakan tahapan penting untuk mengubah data mentah menjadi data yang berkualitas dan siap digunakan dalam proses pengembangan model, mulai dari pelatihan, validasi, hingga pengujian guna mengevaluasi kinerja model yang dibangun [20]. Proses pembersihan dan pemolesan data yang telah dikumpulkan dilakukan dengan menghilangkan kesalahan serta ketidakkonsistenan, menangani nilai yang hilang dan outlier secara tepat, serta menormalisasi data untuk menjamin keseragaman dan stabilitas selama pelatihan model [28].

2.3 Normalisasi Data

Normalisasi data merupakan salah satu tahapan *preprocessing* yang bertujuan menyesuaikan skala data sehingga proses pembelajaran model dapat berlangsung lebih efektif serta meningkatkan performa dan efisiensi pelatihan [29]. Tahap ini dilakukan untuk meminimalkan tingkat kesalahan serta mengurangi potensi terjadinya anomali pada data dengan menyelaraskan skala nilai tanpa mengubah karakteristik asli data [30]. Metode normalisasi yang digunakan dalam penelitian ini adalah *Min-Max Scaler*. Pada penelitian ini, metode normalisasi yang digunakan adalah *Min-Max Scaler*, yaitu teknik transformasi data yang mengubah nilai atribut ke dalam rentang interval [0,1] [3]. Adapun persamaan normalisasi *Min-Max Scaler* ditunjukkan sebagai berikut:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \dots\dots\dots (1)$$

Dengan X adalah nilai asli, X_{min} dan X_{max} adalah nilai minimum dan maksimum dari atribut tersebut [31].

2.4 Splitting Data

Splitting data merupakan proses pembagian dataset menjadi beberapa bagian yang digunakan dalam tahap pelatihan, validasi, dan pengujian model. Proses ini bertujuan untuk memastikan model dapat mempelajari pola data dengan baik serta mengevaluasi kemampuannya dalam melakukan prediksi pada data yang belum pernah digunakan sebelumnya. Proses *splitting* data sangat penting pada data deret waktu karena dapat mencegah terjadinya kebocoran informasi (*data leakage*) dari data masa depan ke dalam proses pelatihan model, sehingga hasil prediksi yang diperoleh tetap valid dan representatif [32]. Adapun jumlah data pada setiap subset dihitung menggunakan Persamaan (2), Persamaan (3), dan Persamaan (4).

$$nData_{train} = N \times P_{train} \dots\dots\dots (2)$$

$$nData_{val} = N \times P_{val} \dots\dots\dots (3)$$

$$nData_{test} = N - nData_{train} - nData_{val} \dots\dots\dots (4)$$

Dimana:

N : jumlah data keseluruhan

$nData_{train}$: jumlah data *training*

$nData_{val}$: jumlah data *validation*

$nData_{test}$: jumlah data *testing*

P_{train} : persentase data *training*

P_{val} : persentase data *validation*

2.5 Machine Learning

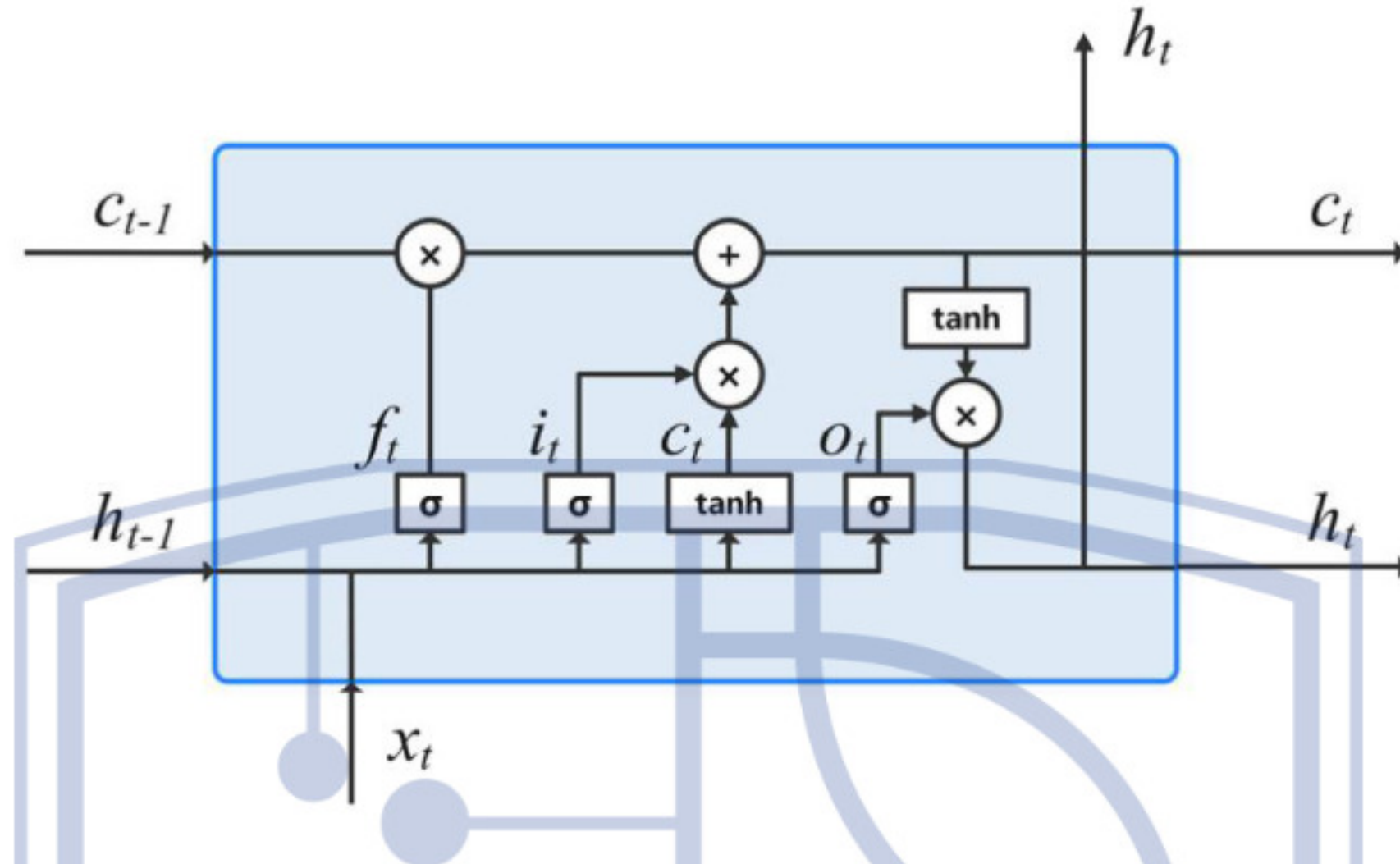
Machine learning merupakan salah satu bidang dalam *Artificial Intelligence* yang telah banyak diteliti dan diterapkan dalam memprediksi pergerakan harga saham [28]. Dalam proses peramalan, pemilihan model yang tepat menjadi faktor penting karena setiap algoritma memiliki karakteristik dan performa yang berbeda, sehingga penggunaan model yang kurang sesuai dapat menghasilkan prediksi yang tidak optimal [33]. *Machine learning* banyak digunakan karena kemampuannya menangani data nonlinier, non-stasioner, dan berdimensi tinggi secara efektif [13]. Berbagai teknik machine learning seperti *Linear Regression*, *Support Vector Machine* (SVM), *Random Forest*, *Nearest Neighbor*, dan *Decision Trees* telah digunakan untuk memprediksi harga pasar saham berdasarkan data historis [34].

Seiring dengan meningkatnya kompleksitas data dan kebutuhan akan akurasi yang lebih tinggi, berkembanglah pendekatan *deep learning* sebagai bagian dari *machine learning* [35]. Teknologi deep learning dalam prediksi harga saham memungkinkan model mempelajari pola pada data deret waktu secara lebih efektif. Hal ini didukung oleh arsitektur jaringan saraf tiruan yang memiliki lebih banyak hidden layer dibandingkan neural network konvensional [31]. *Deep learning* juga banyak diterapkan dalam bidang keuangan, antara lain untuk prediksi harga saham dan indeks, optimasi portofolio, manajemen risiko, pengolahan informasi keuangan, serta pengembangan strategi pelaksanaan perdagangan [32]. Beberapa algoritma *deep learning* yang telah banyak diterapkan antara lain *Long Short-Term Memory* (LSTM), *Bidirectional Long Short-Term Memory* (BiLSTM), serta *Gated Recurrent Unit* (GRU) [14].

2.6 Long Short-Term Memory

Long Short-Term Memory (LSTM) adalah pengembangan dari arsitektur *Recurrent Neural Network* (RNN) yang pertama kali diperkenalkan pada tahun 1997 oleh Sepp Hochreiter dan Jürgen Schmidhuber [14]. Model LSTM memiliki kontribusi utama berupa mekanisme *self-loop* dengan pembaruan bobot secara iteratif yang memungkinkan aliran gradien bertahan dalam jangka panjang sehingga dapat mengatasi masalah *vanishing gradient* pada RNN [36]. Model ini juga memiliki sel memori untuk menyimpan informasi jangka panjang, di mana aliran informasi dalam sel tersebut dikontrol oleh tiga gerbang

khusus, yaitu *forget gate*, *input gate*, dan *output gate* [37]. Model LSTM dapat dilihat pada Gambar 2.1.



Gambar 1.1 Model Jaringan Long Short Term Memory [38]

Adapun tahapan dalam algoritma LSTM dapat dijelaskan sebagai berikut [39].

1. Perhitungan *input gate* dilakukan menggunakan persamaan:

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \dots \dots \dots (5)$$
2. Perhitungan *forget gate* dilakukan menggunakan persamaan:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \dots \dots \dots (6)$$
3. Hitung pembaruan *cell state* menggunakan persamaan:

$$c_t = f_t * c_{t-1} + i_t * \tanh(W_c[h_{t-1}, x_t] + b_c) \dots \dots \dots (7)$$
4. Perhitungan *output gate* dilakukan menggunakan persamaan:

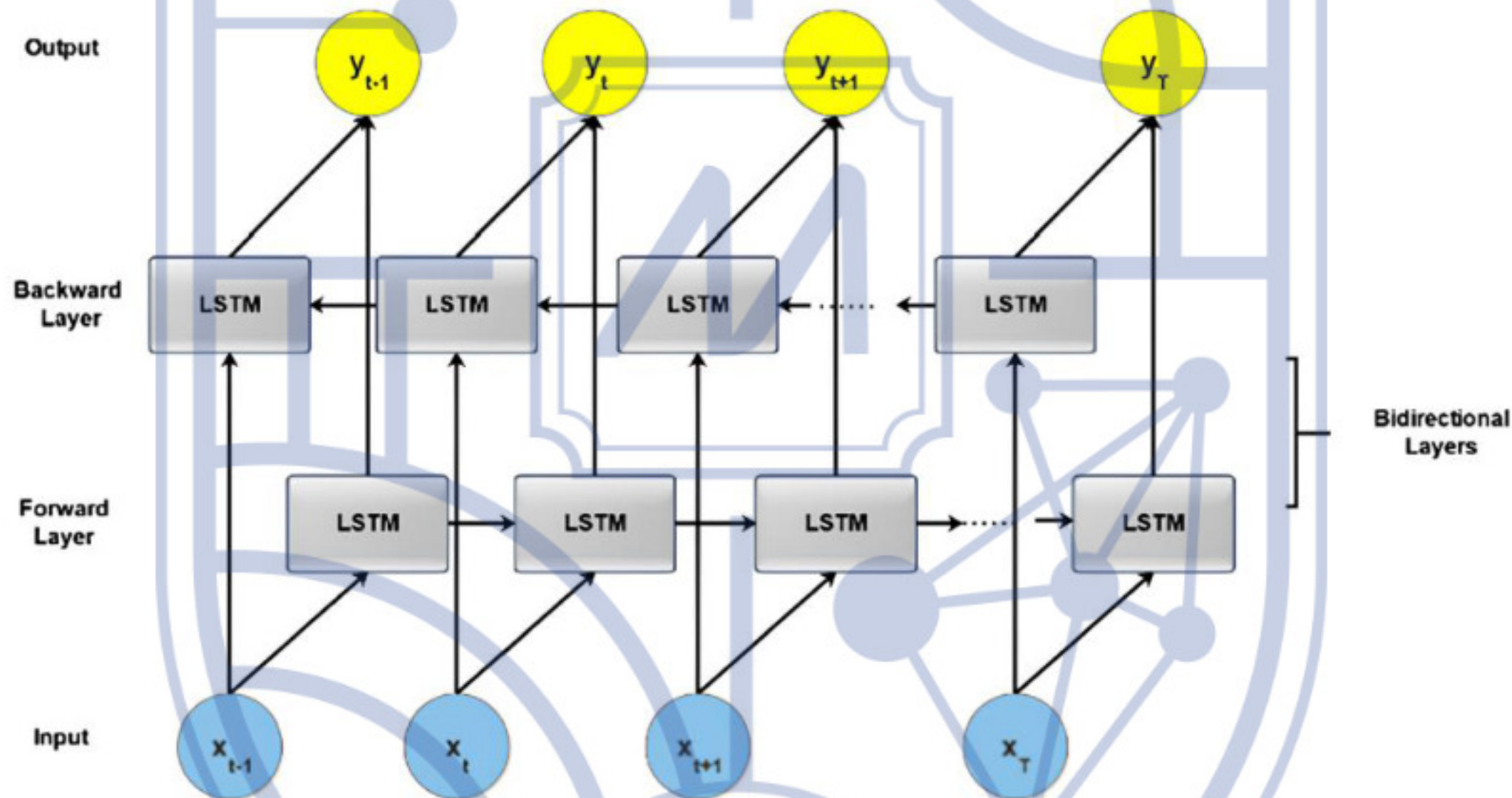
$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \dots \dots \dots (8)$$
5. Perhitungan *hidden state* dilakukan menggunakan persamaan:

$$h_t = o_t * \tanh(c_t) \dots \dots \dots (9)$$

Dimana, x_t merupakan input pada waktu ke- t , sedangkan h_{t-1} dan c_{t-1} masing-masing adalah *hidden state* dan *cell state* dari waktu sebelumnya. Nilai x_t dan h_{t-1} diproses oleh *input gate* (i_t), *forget gate* (f_t), dan *output gate* (o_t) dengan fungsi sigmoid. *forget gate* mengatur informasi dari c_{t-1} yang dipertahankan, sementara *input gate* menentukan informasi baru yang ditambahkan ke *cell state*. *Cell state* diperbarui menjadi c_t , dan *hidden state* h_t dihasilkan dari kombinasi c_t melalui fungsi tanh dengan keluaran *output gate*. Parameter W dan b masing-masing merepresentasikan matriks bobot dan vektor bias yang digunakan dalam proses perhitungan pada setiap gerbang [8][39].

2.7 Bidirectional Long Short-Term Memory

LSTM memiliki keterbatasan dalam memahami konteks secara menyeluruh karena hanya memproses data ke *forward* [10]. Untuk mengatasi hal tersebut, dikembangkan *Bidirectional Long Short Term Memory* (BiLSTM) yang terdiri dari dua lapisan tersembunyi LSTM, di mana LSTM pertama mempelajari data dengan memprosesnya ke arah maju (*forward*), sedangkan LSTM kedua memproses informasi ke arah sebaliknya (*backward*) [40]. Output dari LSTM yang memproses maju dan mundur digabungkan pada setiap langkah waktu, sehingga dengan 2 *layer* yang saling berlawanan arah, model dapat mempelajari baik informasi dari masa lalu maupun masa depan untuk setiap urutan input [14]. Pendekatan ini memungkinkan jaringan mempertahankan informasi sebelumnya dan informasi mendatang untuk digunakan sebagai dasar waktu saat ini dari data *time series* [13].



Gambar 1.2 Model Jaringan BiLSTM [41]

Adapun tahapan dalam algoritma BiLSTM dapat dijelaskan sebagai berikut [10].

1. Tentukan nilai input untuk lapisan tersembunyi di arah maju dengan persamaan berikut.

$$\vec{h}_t = f(w_1 x_t + w_2 h_{t-1}) \dots \dots \dots (10)$$

2. Tentukan nilai lapisan tersembunyi di arah mundur dengan persamaan berikut.

$$\overleftarrow{h}_t = f(w_3 x_t + w_5 h_{t+1}) \dots \dots \dots (11)$$

3. Tentukan nilai lapisan output dengan mengombinasikan hasil dari lapisan maju dan lapisan mundur sesuai persamaan berikut.

$$o_t = g(w_4 \vec{h}_t + w_6 \overleftarrow{h}_t) \dots \dots \dots (12)$$

Dimana, x_t merupakan *input* pada waktu ke- t , $\overrightarrow{h_t}$ adalah *hidden state* pada arah maju, dan $\overleftarrow{h_t}$ adalah *hidden state* pada arah mundur. Variabel h_{t-1} dan h_{t+1} masing-masing menyatakan *hidden state* pada waktu sebelumnya dan waktu berikutnya. Parameter w_1, w_2, w_3, w_4, w_5 , dan w_6 merupakan bobot yang digunakan dalam proses pembelajaran model. Fungsi $f(\cdot)$ digunakan pada lapisan tersembunyi, sedangkan $g(\cdot)$ digunakan pada lapisan keluaran. Nilai o_t merupakan output yang diperoleh dari penggabungan *hidden state* arah maju dan arah mundur [10][13].

2.8 Firefly Algorithm

Firefly Algorithm (FA) merupakan algoritma metaheuristik yang kuat dan efisien, yang telah terbukti memiliki kinerja yang efektif dalam berbagai studi terbaru ketika digunakan untuk menyelesaikan masalah optimasi di bidang teknik [42]. Algoritma ini terinspirasi oleh perilaku berkedip kunang-kunang, di mana kinerja setiap solusi yang dihasilkan secara acak pada fungsi tujuan dinilai berdasarkan intensitas kecerahannya [43].

Dalam populasi FA, semua *firefly* bersifat uniseks, sehingga setiap individu dapat saling tertarik tanpa memperhatikan jenis kelamin [44]. *firefly* jantan dan betina menunjukkan ketertarikan pada frekuensi kilatan yang berbeda, yang dipengaruhi oleh kecepatan dan periode kilatan tersebut. Selain itu, hubungan antara intensitas cahaya (I) yang dipancarkan oleh tubuh kunang-kunang dengan jarak (r) dinyatakan oleh $I \propto 1/r^2$, yang menunjukkan bahwa intensitas cahaya berbanding terbalik dengan kuadrat jarak [42].

Parameter masukan untuk *firefly algorithm* meliputi N , yaitu jumlah kunang-kunang dalam populasi, parameter acak α , parameter daya tarik β ketika jarak r dari sumber sama dengan nol, koefisien reduksi cahaya γ , serta jumlah iterasi maksimum yang dijalankan oleh algoritma [19].

Intensitas cahaya dijelaskan pada persamaan (13), di mana, $I(r)$ menyatakan intensitas cahaya pada jarak r , γ merupakan koefisien penyerapan cahaya, dan I_0 adalah intensitas cahaya pada sumbernya [45].

$$I(r) = I_0 e^{(-\gamma r^2)} \dots\dots\dots(13)$$

Selain itu, setiap kunang-kunang memanfaatkan daya tarik β , yang sebanding dengan intensitas cahaya dari kunang-kunang tertentu dan juga dipengaruhi oleh jarak [46]. Sebagaimana ditunjukkan dalam persamaan (14) [47].

$$\beta(r) = \beta_0 e^{(-\gamma r^2)} \dots\dots\dots(14)$$

Persamaan (15) digunakan untuk menghitung jarak antara dua kunang-kunang, i dan j , yang berada pada posisi x_i dan x_j masing-masing, dan dapat didefinisikan sebagai *Cartesian distance* [46]. Di sini, x_{ik} merupakan elemen ke- k dari koordinat spasial kunang-kunang ke- i , sedangkan D menunjukkan dimensi dari permasalahan yang sedang dipertimbangkan [48].

$$r_{ij} = \|x_i - x_j\| = \sqrt{\sum_{k=1}^D (x_{i,k} - x_{j,k})^2} \dots\dots\dots (15)$$

Kunang-kunang cenderung tertarik pada individu yang lebih terang dan lebih menarik. Perilaku ini dapat dimodelkan melalui persamaan (15), di mana bagian kedua persamaan tersebut berasal dari persamaan daya tarik, yang pada dasarnya melibatkan pengaruh jarak terhadap tingkat daya tarik antara dua kunang-kunang [19][45]. Dimana, α merupakan parameter pengacakan, sedangkan angka acak yang diambil dari distribusi Gaussian atau distribusi seragam dilambangkan dengan κ . Simbol $r_{i,j}$ menunjukkan jarak antara dua kunang-kunang yang diamati, i dan j . Nilai tipikal yang umumnya memberikan hasil memuaskan untuk sebagian besar masalah adalah $\beta_0 = 1$ dan α berada dalam rentang $[0, 1]$ [46].

$$x_i^{t+1} = x_i^t + \beta_0 e^{-\gamma r_{i,j}^2} (x_j^t - x_i^t) + \alpha^t (\kappa - 0.5) \dots\dots\dots (16)$$

Parameter γ memiliki peran penting dalam menentukan laju konvergensi algoritma. Secara teoritis, nilai γ dapat berada dalam rentang $\gamma \in [0, \infty)$. Dalam praktiknya, pemilihan nilai γ biasanya disesuaikan dengan jenis masalah yang dihadapi, dan umumnya berada pada kisaran 0,1 hingga 10 [19].

*Initialization of the parameters of FA (Population size, α , β_0 , γ and the number of iterations).
The Light intensity is defined by the cost function $f(x_i)$ where $x_i (i = 1, \dots, n)$.
While (iter < Max Generation).
 for $i = 1:n$ (all n fireflies)
 for $j = 1:n$ (all n fireflies)
 if ($f(x_i) < f(x_j)$), move firefly i towards j ,
 end if.
 Update attractiveness β with distance r .
 Evaluate new solution and update $f(x_i)$ in the same way as (4).
 end for j
end for i
rank the solutions and find the best global optimal.
end while.
Show the results.*

Gambar 1.3 Pseudocode Firefly Algorithm [48]

2.9 Improved Firefly Algorithm

Pada *Firefly Algorithm*, solusi dengan nilai optimum terbaik direpresentasikan oleh kunang-kunang dengan tingkat kecerahan yang sangat tinggi [49]. Namun, pendekatan klasik sering mengalami permasalahan terjebak pada minimum lokal akibat karakteristik perancangan yang bersifat nonlinier [50]. Untuk mengatasi keterbatasan tersebut, dikembangkan *Improved Firefly Algorithm* yang bertujuan meningkatkan kemampuan algoritma dalam menghindari jebakan minimum lokal. Pada metode ini, ditambahkan suatu komponen baru ke dalam algoritma FA untuk meningkatkan tingkat keacakan dan efisiensi pencarian. Selanjutnya, perbedaan antara posisi kunang-kunang acak dan posisi kunang-kunang ke-X dihitung, sehingga diperoleh mekanisme keacakan yang lebih efektif [49].

2.9.1 The Minimum Attraction

Dalam analisis parameter *Firefly Algorithm*, koefisien penyerapan cahaya γ yang terlalu besar atau tetap konstan dapat menyebabkan perluasan wilayah pencarian yang terlalu luas. Akibatnya, interaksi antar kunang-kunang menjadi sangat lemah, hampir mendekati nol [19]. Untuk mencegah situasi di mana kunang-kunang bergerak tanpa arah akibat jarak yang terlalu besar dan daya tarik yang menurun, diperkenalkan konsep daya tarik minimum sebagai langkah antisipatif, sebagaimana ditunjukkan pada Persamaan sebagai berikut.

$$\beta_{ij}(r_{ij}) = \beta_{min} + (\beta_0 - \beta_{min})e^{-\gamma r_{ij}^2} \dots\dots\dots(17)$$

Dimana, nilai β_0 ditetapkan sebesar 1, sedangkan β_k berada dalam rentang $[0, 1]$. Dengan demikian, ketika jarak antara kunang-kunang menjadi sangat besar dan mendekati nol daya tarik efektif, tingkat daya tarik antar individu dapat mencapai β_k .

2.9.2 Adaptive Step Length Factor

Dalam algoritma Firefly konvensional, faktor step length α ditetapkan sebagai nilai konstan. Pada fase awal iterasi, α berperan dalam mendukung kemampuan algoritma untuk mengeksplorasi solusi global secara efektif. Namun, seiring mendekatnya jumlah iterasi ke batas maksimum, sifat konstan dari α menyebabkan mayoritas kunang-kunang terkonsentrasi di sekitar solusi terbaik, sehingga interaksi dengan individu lainnya menjadi terbatas [17]. Untuk mengatasi ketidakseimbangan antara tahap eksplorasi awal dan tahap konvergensi akhir dari keseluruhan proses optimasi, penyesuaian dinamis α diperkenalkan, sebagaimana dalam Persamaan (18), dengan parameter p yang dihitung menurut Persamaan (19).

$$\alpha = \alpha_0 * P \dots\dots\dots(18)$$

$$P = \frac{Max-t}{Max} \dots\dots\dots(19)$$

Di mana Max menyatakan jumlah iterasi maksimum, sedangkan t menunjukkan iterasi saat ini.

2.9.3 Inertia Weight

Dengan mengacu pada pengembangan *Particle Swarm Optimization* yang menggunakan bobot inersia, algoritma dasar ditingkatkan melalui penerapan pengurangan bobot inersia secara linier [17][19]. Sebagaimana ditunjukkan pada Persamaan sebagai berikut.

$$w = w_{max} - (w_{max} - w_{min}) * \frac{t}{MaxT} \dots\dots\dots(20)$$

Pada persamaan tersebut, w_{max} dan w_{min} menunjukkan bobot maksimum dan minimum, sementara t serta t_{max} menyatakan iterasi ke-t saat ini dan jumlah iterasi maksimum [17][19]. Proses pembaruan posisi selanjutnya dilakukan berdasarkan Persamaan sebagai berikut.

$$x_i^{t+1} = wx(t) + \beta (x_j^t - x_i^t) + \alpha^t (k - 0.5) \dots\dots\dots(21)$$

2.10 Denormalisasi Data

Denormalisasi merupakan proses mengembalikan data hasil prediksi ke skala nilai aslinya dengan menggunakan transformasi kebalikan dari metode normalisasi *Min-Max Scaler* [51]. Tahap ini dilakukan setelah model menghasilkan nilai prediksi dalam bentuk data yang telah dinormalisasi. Dengan demikian, hasil prediksi dapat diinterpretasikan dalam satuan yang sama dengan data aktual. Adapun persamaan denormalisasi ditunjukkan pada Persamaan (22).

$$X' = X_{norm}(X_{max} - X_{min}) + X_{min} \dots\dots\dots(22)$$

dengan x' menyatakan nilai hasil denormalisasi, X_{norm} menyatakan nilai hasil prediksi yang masih berada pada rentang normalisasi, X_{max} merupakan nilai maksimum data, dan X_{min} merupakan nilai minimum data [30].

2.11 Metrik Evaluasi

Dalam model *deep learning*, *loss error* digunakan untuk menilai kinerja prediksi dengan mengukur selisih antara nilai aktual dan nilai prediksi menggunakan berbagai metode evaluasi [52]. Beberapa metrik yang digunakan untuk mengukur *loss error* antara

lain *Mean Absolute Error* (MAE), *Mean Absolute Percentage Error* (MAPE), dan *Root Mean Squared Error* (RMSE).

1. MAE

Mean Absolute Error merupakan metrik evaluasi yang digunakan untuk mengukur tingkat akurasi prediksi dengan menghitung rata-rata kesalahan absolut antara nilai prediksi dan nilai aktual, tanpa memperhatikan arah kesalahannya, di mana nilai MAE yang lebih kecil menunjukkan kinerja prediksi yang lebih baik [32]. MAE ditunjukkan pada Persamaan sebagai berikut.

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \dots \dots \dots (23)$$

2. RMSE

Root Mean Square Error adalah metrik umum untuk mengukur kesalahan regresi, berupa akar kuadrat dari MSE, yang menunjukkan seberapa baik data mengikuti garis regresi, di mana nilai RMSE yang lebih kecil menandakan kecocokan model yang lebih baik [8]. RMSE ditunjukkan pada Persamaan sebagai berikut.

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2} \dots \dots \dots (24)$$

3. MAPE

Mean Absolute Percentage Error adalah metrik evaluasi yang digunakan untuk menilai tingkat kesalahan relatif pada model prediksi, dengan menghitung rata-rata persentase selisih antara nilai prediksi dan nilai aktual dari data yang diamati [35]. MAPE ditunjukkan pada Persamaan sebagai berikut.

$$MAPE = \frac{\sum_{t=1}^n \left| \frac{y_t - \hat{y}_t}{y_t} \right|}{n} \times 100\% \dots \dots \dots (25)$$

Di mana, $\hat{y}_i$ merepresentasikan nilai yang diperkirakan untuk data ke-i, y_i merupakan nilai riil pada periode ke-i, dan n adalah banyaknya observasi.