BAB II

TINJAUAN PUSTAKA

2.1 Harga Batu Bara

Batu bara dapat didefinisikan sebagai batuan sedimen yang dapat terbakar dan berasal dari sisa tumbuhan yang telah mengalami proses geologi (diagenesis dan umur) (Akbar et al., 2018). Konsumsi akan batu bara dunia mengalami peningkatan tidak terlepas dari pesatnya permintaan energi dunia dimana batu bara menjadi pemasok energi kedua terbesar setelah minyak. Manfaat batu bara tidak terbatas sebagai pemasok energi listrik saja, namun juga digunakan di berbagai kegiatan industri seperti besi, baja, dan farmasi. Oleh karena itu, batu bara menjadi komoditas utama dalam perdagangan internasional. Harga batu bara di pasar internasional tidak konstan setiap harinya, hal ini menyebabkan beberapa negara produsen batu bara menentukan harga jual yang berbeda-beda. Indeks harga batu bara yang paling banyak digunakan sebagai acuan oleh seluruh negara di dunia adalah *Newcastle Export Index* (NEX). *Newcastle* merupakan fasilitas pemuatan batu bara terbesar di dunia juga sebagai tempat transaksi batu bara terbesar di Asia (World Energy Council, 2010). Penentuan harga batu bara dilakukan setiap hari dengan mata uang berupa Dollar Amerika Serikat per metrik ton. *Newcastle Export Index* memiliki kadar kalori dasar yang digunakan sebagai acuan dalam menentukan harga yang berlaku di pasar internasional.

2.2 Peramalan (*Forecasting*)

Definisi dari peramalan (*forecasting*) adalah seni dan ilmu untuk memperkirakan kejadian di masa depan. Hal tersebut dapat dilakukan dengan menggunakan data historis dan proses kalkulasi untuk memprediksikan sebuah proyeksi atas kejadian di masa datang. Peramalan merupakan suatu proses untuk memperkirakan berapa banyak kebutuhan di masa depan diantaranya kebutuhan akan ukuran kualitas, kuantitas, waktu, dan lokasi yang dibutuhkan dalam rangka memenuhi permintaan barang atau jasa berdasarkan data historis (masa lalu). Manfaat dari peramalan yaitu dapat mengetahui gambaran terkait suatu hal di masa yang akan datang. Selain itu, peramalan juga memberikan acuan penting bagi perusahaan dalam menentukan kebijakan atau perencanaan sehingga dapat mengantisipasi terjadinya hal-hal yang tidak diinginkan. Peramalan berdasarkan horizontal waktu diklasifikan menjadi 3 kategori, antara lain (Heizer dan Render, 2006):

1. Peramalan jangka pendek, peramalan ini mencakup jangka waktu hingga 12 bulan tetapi umumnya kurang dari 3 bulan.
2. Peramalan jangka menengah, peramalan ini pada umumnya mencakup waktu hingga 3 bulan hingga 3 tahun.
3. Peramalan jangka panjang umumnya peramalan ini untuk perencanaan 3 tahun atau lebih.

Selain itu, jika dilihat dari jangka waktu ramalan yang disusun, maka peramalan dibedakan menjadi 2, yaitu (Ginting, 2007):

1. Peramalan jangka panjang, yaitu peramalan yang dilakukan untuk penyusunan lebih dari satu setengah tahun atau tiga semester.
2. Peramalan jangka pendek, yaitu peramalan yang dilakukan untuk penyusunan hasil ramalan dengan jangka waktu yang kurang dari satu setengah tahun atau tiga semester.

Menurut Stevenson (2011) dalam buku *Operation Management*, terdapat 6 langkah dasar dalam proses peramalan yaitu:

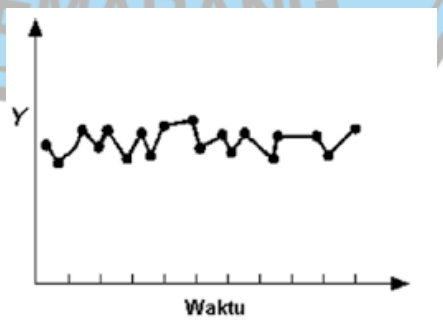
1. Menentukan tujuan dari peramalan. Bagaimana hasilnya dan kapan akan digunakan, langkah ini memberikan indikasi mengenai tingkat detail yang dibutuhkan dalam peramalan, banyaknya sumber daya yang dibutuhkan, dan tingkat akurasi.
2. Menentukan rentang waktu, semakin panjang rentang waktu maka semakin berkurang akurasi dari peramalan.
3. Memilih teknik atau metode *forecasting*.
4. Melakukan analisa data, karena data yang tidak akurat akan mengurangi validasi dari hasil peramalan.
5. Membuat persamaan.
6. Memantau hasil peramalan untuk mengetahui apakah performa yang dihasilkan memuaskan, jika tidak revisi lagi mengenai metode yang digunakan dan validitas dari data yang digunakan.

2.3 Analisis Runtun Waktu

Data runtun waktu (*time* series) merupakan jenis data yang dikumpulkan menurut urutan waktu dalam periode waktu tertentu. Adapun waktu adalah serangkaian observasi pada suatu peristiwa yang diambil dari waktu ke waktu. Dalam hal ini waktu dapat berupa hari, minggu, bulan, tahun, dan sebagainya. Dasar pikiran *time series* adalah pengamatan sekarang (Z_t) yang dipengaruhi oleh pengamatan yang dilakukan sebelumnya (Z_{t-k}). Analisis *time series* bertujuan untuk menemukan pola variasi masa lalu yang digunakan untuk memperkirakan nilai masa depan dan memberikan bantuan dalam manajemen operasi dalam membuat perencanaan (Winarno, 2007). Manfaat *time series* yaitu melakukan perencanaan sebagai suatu keputusan di masa depan berdasarkan pola data masa sebelumnya dengan hasil peramalan. Menurut Makridakis et al. (1999), pola data *time series* dapat dibedakan menjadi empat, yaitu:

1. Pola Horizontal

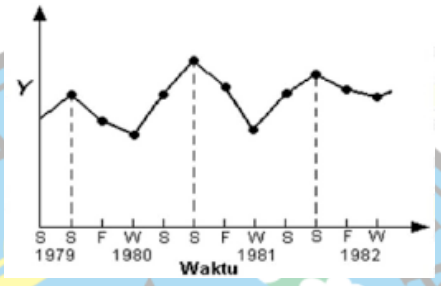
   Pola horizontal terjadi jika nilai dari data mengalami fluktuasi di daerah nilai rata-rata (*mean*) konstan. Dalam hal ini nilai rata-ratanya stasioner.



Gambar 2.1. Pola Data Horizontal

2. Pola Musiman

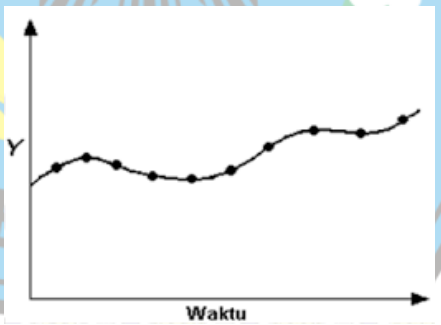
Pola musiman terjadi jika suatu deret dari data dipengaruhi oleh faktor musiman yang ditunjukkan oleh adanya pola yang teratur dan bersifat musiman.



Gambar 2.2. Pola Data Musiman

3. Pola Siklis

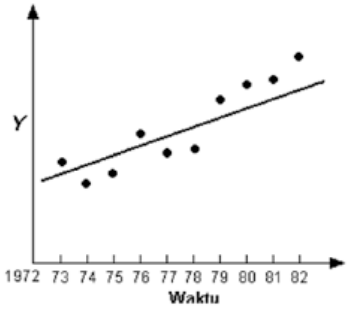
Pola siklis terjadi jika pola *time series* mengalami fluktuasi ekonomi jangka panjang yang berhubungan dengan siklus bisnis.



Gambar 2.3. Pola Data Siklis

4. Pola *Trend*

Pola *trend* terjadi jika pola data mengalami fluktuasi, pola data tersebut bervariasi dan tak beraturan.



Gambar 2.4. Pola Data *Trend*

2.4 Jaringan Syaraf Tiruan (JST)

Jaringan syaraf tiruan (JST) merupakan salah satu representasi buatan dari otak manusia yang selalu mencoba mensimulasikan proses pembelajaran pada otak manusia (Kusumadewi, 2003). Dalam otak manusia terdiri dari sel-sel yang disebut *neuron*. Jaringan syaraf tiruan (*neural network*) didesain dengan menirukan cara kerja otak manusia dalam menyelesaikan suatu permasalahan. Selain itu, jaringan ini digunakan untuk memodelkan cara kerja otak manusia dalam melaksanakan tugas tertentu. Pembuatan model tersebut didasari oleh kemampuan otak manusia dalam mengorganisasi *neuron*. Jaringan syaraf tiruan mampu melakukan kegiatan berdasarkan data masa lampau dengan mempelajarinya sehingga dapat memberikan keputusan terhadap data yang belum pernah dipelajari.

2.4.1 Komponen Jaringan Syaraf Tiruan

Secara umum, komponen jaringan syaraf tiruan memiliki 2 lapisan, yaitu *input layer* dan *output layer*. Seiring berkembangnya waktu, jaringan syaraf tiruan dikembangkan ada yang menambah 1 lapisan lagi yaitu *hidden*

layer yang letaknya antara *input* dan *output layer* (Hermawan, 2006). Berikut ini adalah 3 lapisan dalam komponen jaringan syaraf tiruan, yaitu:

a. *Input Layer* (Lapisan *Input*)

Input layer berisi *node* (*neuron*) yang masing-masing menyimpan sebuah nilai masukan yang tetap pada proses pelatihan (*training*) dan hanya berubah apabila diberi nilai *input* yang baru. *Input neuron* tersebut menerima pola masukan dari luar yang menggambarkan suatu masalah. Banyaknya *neuron* tergantung dari banyak *input* dalam model dan setiap *input* menentukan satu *neuron*.

b. *Hidden Layer* (Lapisan Tersembunyi)

Node (*neuron*) pada *hidden layer* disebut *node* tersembunyi. Dalam hal ini *output* tidak dapat diamati secara langsung. Semua proses dalam proses pelatihan (*training*) dan pengujian (*testing*) dijalankan pada lapisan ini. Banyaknya lapisan ini tergantung dari arsitektur yang dirancang, tetapi secara umum terdiri dari satu *hidden layer*.

c. *Output Layer* (Lapisan Keluaran)

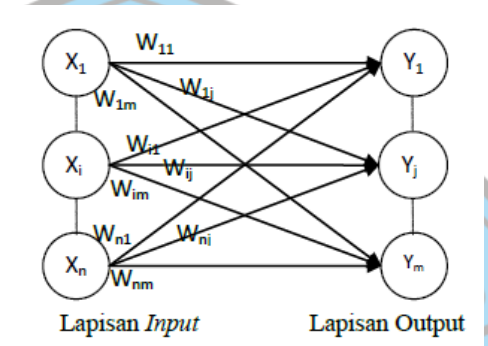
Output layer merupakan solusi dari suatu permasalahan. Pada lapisan ini menunjukkan nilai keluaran dari hasil perhitungan sistem oleh fungsi aktivasi pada *hidden layer* berdasarkan *input* yang diterima.

2.4.2 Arsitektur Jaringan Syaraf Tiruan

Arsitektur jaringan syaraf tiruan terbagi menjadi 3 model, yaitu (Puspitaningrum, 2006):

1. Jaringan Layar Tunggal (*Single Layer*)

Pada jaringan layar tunggal, *input neuron* yang menerima sinyal dari luar terhubung langsung ke *output neuron*. Jaringan ini hanya menerima *input* dan mengolahnya secara langsung menjadi *output* tanpa harus melalui *hidden layer*. Seperti terlihat pada gambar berikut:



Gambar 2.5. Jaringan Layar Tunggal

Keterangan:

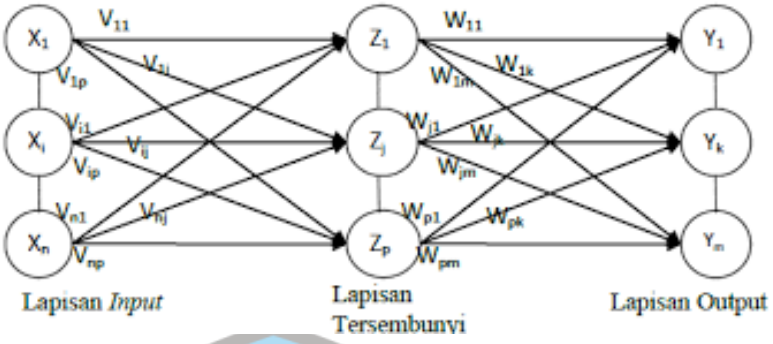
X_1, X_i, X_n : *Input*

Y_1, Y_j, Y_m : *Output*

$W_{11}..W_{mn}$: Matriks bobot

2. Jaringan Layar Jamak (*Multi Layer*)

Jaringan layar jamak memiliki satu atau lebih lapisan yang terletak di antara *input layer* dan *output layer*. Selain itu, jaringan ini dapat memecahkan masalah yang lebih rumit dibandingkan *single layer*. Pada jaringan ini tidak hanya terdiri atas *input layer*, *hidden layer*, dan *output layer* seperti pada gambar berikut:



Gambar 2.6. Jaringan Layar Jamak

Keterangan:

X_1, X_i, X_n : *Input*

Z_1, Z_j, Z_p : *Hidden layer*

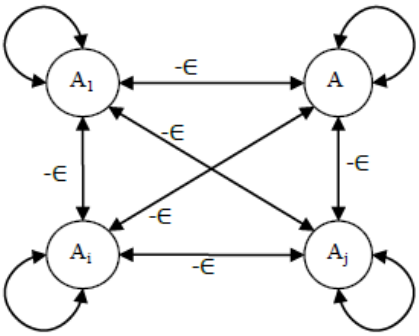
Y_1, Y_k, Y_m : *Output*

V_i : Matriks bobot pertama

W_i : Matriks bobot kedua

3. Jaringan *Reccurent*

Model jaringan *reccurent* mirip dengan jaringan layar tunggal maupun jaringan layar jamak. Namun, jaringan ini memiliki *output neuron* yang memberikan sinyal *input neuron* (*feedback loop*). Artinya sinyal tersebut mengalir dua arah, yaitu maju dan mundur. Jaringan ini memiliki minimal satu *feedback loop* yang terdiri dari masing-masing *neuron* untuk memberikan kembali *output* nya sebagai *input* pada *neuron* lain. Nilai bobot untuk tiap *neuron* adalah 1 dan bobot acak negatif dengan – ϵ untuk *neuron* lain dapat dilihat seperti pada gambar berikut.



Gambar 2.7. Jaringan *Reccurent*

Keterangan:

A_1, A_i, A_j : *Input*

– ϵ : Bobot acak negatif

2.4.3 Fungsi Aktivasi

Dalam jaringan syaraf tiruan terdapat fungsi aktivasi yang digunakan untuk menentukan *output* suatu *neuron* dengan berargumen *net input*. Argumen fungsi aktivasi adalah *net input* (kombinasi linier *output* dan bobotnya). Fungsi aktivasi bertujuan untuk memodifikasi *output* ke dalam rentang nilai tertentu. Berikut ini adalah fungsi aktivasi yang sering digunakan dalam jaringan syaraf tiruan (Kusumadewi, 2003) yaitu:

1. Fungsi Sigmoid Biner

   Fungsi sigmoid biner memiliki nilai *output* antara 0 sampai 1 dengan membentuk kurva S yang dapat memberikan hasil *output* lebih cepat. Rumus dari fungsi sigmoid biner dapat dilihat pada Persamaan 1.

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

Keterangan:

$f(x)$: Fungsi aktivasi sigmoid biner

e : Eksponensial

x : Data ke-x

2. Fungsi Sigmoid Bipolar

Fungsi sigmoid bipolar hampir mirip dengan fungsi sigmoid biner, hanya saja fungsi sigmoid bipolar memiliki nilai antara -1 dan 1. Rumus dari fungsi sigmoid bipolar dapat dilihat pada Persamaan 2.

$$f(x) = \frac{1 - e^{-x}}{1 + e^{-x}} \quad (2)$$

Keterangan:

$f(x)$: Fungsi aktivasi sigmoid bipolar

e : Eksponensial

x : Data ke-x

2.5 *Extreme Learning Machine* (ELM)

Extreme Learning Machine (ELM) diperkenalkan oleh (Huang et al., 2006). ELM merupakan metode dari jaringan syaraf tiruan *feedforward* sederhana dengan menggunakan *single hidden layer* atau biasa disebut *Single Hidden Layer Feedforward Neural Networks* (SLFNs). Metode pembelajaran ELM digunakan untuk mengatasi kelemahan-kelemahan dari jaringan syaraf tiruan (JST) *feedforward* lainnya terutama dalam hal *learning speed*. Menurut Huang et al. (2006), terdapat dua alasan mengapa JST memiliki *learning speed* rendah yaitu sebagai berikut:

1. Menggunakan *slow gradient based learning algorithm* dalam melakukan pelatihan (*training*).
2. Semua parameter pada jaringan ditentukan secara iteratif menggunakan metode pembelajaran tersebut.

Parameter-parameter (bobot awal) yang digunakan pada ELM seperti bobot masukan dan *hidden* bias dipilih secara acak dalam suatu rentang tertentu sehingga ELM memiliki kinerja dari *learning speed* yang lebih cepat dan mampu menghasilkan *good generalization performance* tanpa ada masalah *overtraining*. Kemudian melakukan perhitungan secara analitis dengan menggunakan *Moore-Penrose Generalized Invers* untuk menghasilkan bobot keluaran. *Moore-Penrose Generalized Invers* atau sering juga dikenal *pseudoinvers* merupakan perluasan dari konsep invers matriks. Invers *Moore-Penrose* ada untuk setiap matriks baik matriks bujur sangkar maupun tidak bujur sangkar dan singular maupun non singular (Britz, 2007).

Definisi dari *Moore-Penrose Generalized Invers* adalah jika diberikan matriks A berukuran $(m\ x\ n)$ dan matriks A^+ merupakan invers *Moore-Penrose* dari matriks A jika memenuhi empat kondisi yaitu:

1. $AA^+A = A$
2. $A^+AA^+ = A^+$
3. $(AA^+)^* = AA^+$
4. $(A^+A)^* = A^+A$

Dengan A^* merupakan konjugat transpose dari matriks A.

Arsitektur jaringan *Extreme Learning Machine* (ELM) terdiri dari p unit pada *input layer*, m unit pada *single hidden layer*, satu bias, dan satu *output*. Selain itu, terdapat bobot penghubung antara *input neuron* dan *hidden neuron*, antara bias dan *hidden neuron*, serta antara *hidden neuron* dan *output*. Berikut ini adalah model runtun waktu ELM dengan m unit pada *single hidden layer*:

$$\hat{Y}_t = \sum_{i=1}^{m} \beta_i g\left(w_i \cdot x_j + b_i\right) \quad (3)$$

Dimana:

$\hat{Y}_t$: Data prediksi

β_i : Bobot penghubung antara *hidden neuron* dengan *output neuron*;

$\beta_i = (\beta_1, \beta_2, \dots, \beta_m)$

w_i : Bobot penghubung antara *input neuron* dengan *hidden neuron*;

$w_i = (w_{1i}, w_{2i}, \dots, w_{pi})$

x_j : Vektor data *input*; $x_j = (x_{j(t-1)}, x_{j(t-2)}, \dots, x_{j(t-p)})$

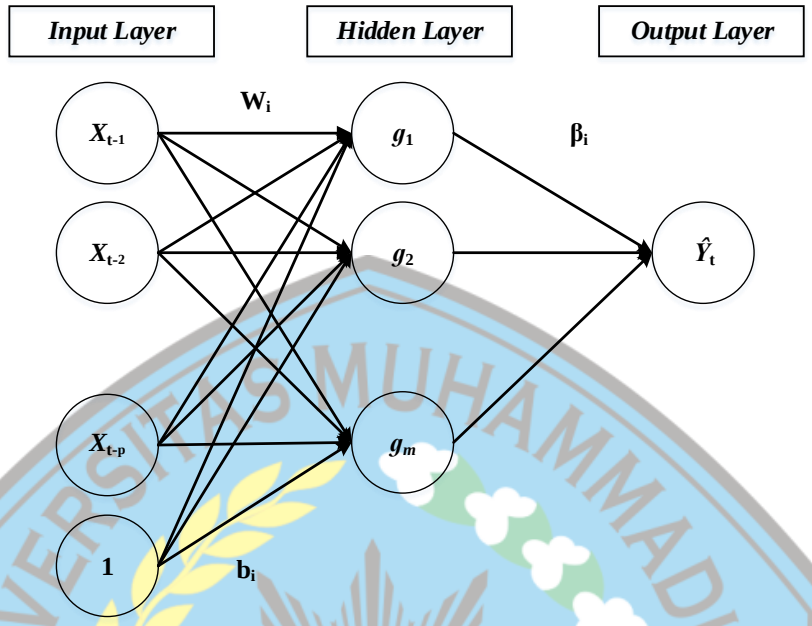
k : *Input neuron*; $k = 1,2, \dots, p$

i : *Hidden neuron*; $i = 1,2, \dots, m$

j : Sampel data; $j = 1,2, \dots, n$

b_i : Bobot bias ke *hidden neuron*; $b_i = (b_1, b_2, \dots, b_m)$

Berikut ini struktur ELM dapat dilihat pada Gambar 2.8 (Izati et al., 2019):



Gambar 2.8. Struktur *Extreme Learning Machine*

2.6 Algoritma *Extreme Learning Machine*

Menurut (Huang et al., 2006) mengemukakan bahwa ada tiga tahapan dalam metode *Extreme Learning Machine* (ELM) yaitu jika diberikan data latih, fungsi aktivasi *g(x)*, dan *m* unit *hidden*, maka:

1. Menentukan vektor bobot *input* W_j dan bias b_j faktor pengaruh *hidden* ke-*j*, b_j, $j = 1,...,m$.
2. Menghitung matriks keluaran pada *hidden layer* $H_{n \times m}$.
3. Menghitung bobot keluaran β.

Langkah-langkah melakukan peramalan menggunakan *Extreme Learning Machine* (ELM) sebagai berikut:

2.6.1 Normalisasi Data

Normalisasi data merupakan metode *pre-processing* dengan tujuan untuk standarisasi seluruh data yang akan digunakan agar berada pada rentang tertentu. Selain itu, normalisasi data dilakukan karena rentang nilai *input* yang tidak sama (*input* akan diproses ke nilai *output* yang kecil). Oleh karena itu, data harus disesuaikan agar dapat diproses untuk mendapatkan nilai normalisasi yang kecil. Fungsi aktivasi yang digunakan pada penelitian ini adalah fungsi sigmoid biner. Menurut Siang (2005), jika menggunakan fungsi aktivasi sigmoid biner maka data harus di transformasikan terlebih dahulu dengan interval [0,1]. Fungsi sigmoid biner akan lebih baik ditransformasikan dalam interval [0.1,0.9] karena fungsi ini merupakan fungsi asimtotik yang nilainya tidak pernah mencapai 0 ataupun 1. Berikut perhitungan normalisasi data dapat dilihat pada Persamaan 4 (Siang, 2005).

$$Y'_t = \frac{0.8(Y_t - \min(x))}{(\max(x) - \min(x))} + 0.1 \quad (4)$$

Keterangan:

Y'_t : Data ke-t dari hasil normalisasi

Y_t : Data ke-t sebelum di normalisasi

$\min(x)$: Nilai minimum data sebelum di normalisasi

$\min(x)$: Nilai maksimum data sebelum di normalisasi

2.6.2 Proses Pelatihan (*Training*)

Proses pelatihan harus dilakukan terlebih dahulu sebelum proses peramalan. Tujuannya untuk mendapatkan nilai bobot keluaran. Langkah-

langkah dalam melakukan proses pelatihan sebagai berikut (Huang et al., 2006):

1. Inisialisasi bobot masukan dan bias secara acak dengan rentang nilai antara -1 dan 1.
2. Langkah selanjutnya adalah menghitung keluaran *hidden layer* (H_{init}) (Fachrony et al., 2018). Berikut rumus untuk menghitung keluaran *hidden layer* dapat dilihat pada Persamaan 5.

$$H_{\text{init train}} = X_{\text{train}} \cdot W^{\text{T}} + b \quad (5)$$

Keterangan:

$H_{\text{init train}}$: Matriks hasil keluaran *hidden layer* untuk proses pelatihan

X_{train} : *Input* data menggunakan data latih

W^T : *Transpose* bobot masukan

b : Nilai bias

Setelah nilai H_{init} didapatkan dan dihitung menggunakan fungsi aktivasi sigmoid biner yang sangat cocok untuk menyelesaikan suatu permasalahan yang rumit dan bersifat non-linier. Rumus fungsi aktivasi sigmoid biner dapat dilihat pada Persamaan 6.

$$H = \begin{bmatrix} g(w_1 \cdot x_1 + b_1) & \dots & g(w_m \cdot x_1 + b_m) \\ g(w_1 \cdot x_2 + b_1) & \dots & g(w_m \cdot x_2 + b_m) \\ \vdots & \ddots & \vdots \\ g(w_1 \cdot x_n + b_1) & \dots & g(w_m \cdot x_n + b_m) \end{bmatrix}_{n \times m}$$

Keterangan:

$w_i \cdot x_j$: *Inner product* dari w_i dan x_j

w_i : Vektor bobot yang menghubungkan *input neuron* dengan

hidden neuron, $w_i = (w_{1i,}\, w_{2i}, \dots, w_{pi})$

x_j : Vektor data *input*, $x_j = (x_{j(t-1)}, x_{j(t-2)}, \dots, x_{j(t-p)})$

$$g(w_1 \cdot x_1 + b_1) = \frac{1}{1 + \exp^{-(w_1 \cdot x_1 + b_1)}}$$

atau

$$H = \frac{1}{1 + \exp^{-(H_{\text{init train}})}} \quad (6)$$

Keterangan:

H : Matriks dari fungsi aktivasi sigmoid biner

exp : Eksponensial

$H_{\text{init train}}$: Matriks keluaran *hidden layer* pada proses pelatihan

3. Menghitung bobot keluaran dengan melakukan *transpose* matriks hasil keluaran *hidden layer* menggunakan fungsi aktivasi. Selanjutnya, matriks hasil keluaran tersebut dikalikan matriks hasil keluaran *hidden layer* dengan fungsi aktivasi sigmoid biner (matriks H). Langkah berikutnya menghitung nilai *invers* dan matriks yang telah didapatkan sebelumnya. Lalu, menghitung matriks *Moore-Penrose Generalized Invers* dan hasil keluaran *hidden layer* dengan fungsi aktivasi. Berikut merupakan rumus untuk mencari matriks *Moore-Penrose Generalized Inverse*.

$$H^+ = (H^T H)^{-1} H^T \quad (7)$$

Keterangan:

H^+ : *Moore-Penrose Generalized Inverse*

H^T : Matriks H *transpose*

H : Matriks H keluaran *hidden layer*

$(H^T H)^{-1}$: *Inverse* dari perkalian matriks H^T dengan H

Berikut merupakan rumus untuk menghitung nilai bobot keluaran ($\hat{\beta}$).

$$\hat{\beta} = H^{+}.Y \quad (8)$$

Keterangan:

$\hat{\beta}$: Matriks bobot keluaran

H^{+} : Matriks *Moore-Penrose Generalized Invers* dari matriks H

Y : Matriks target

4. Menghitung hasil prediksi data latih ($\hat{Y}$).

$$\hat{Y} = H.\hat{\beta} \quad (9)$$

Keterangan:

$\hat{Y}$: Matriks hasil prediksi target data latih

H : Matriks H keluaran *hidden layer*

$\hat{\beta}$: Matriks bobot keluaran

2.6.3 *Root Mean Squared Error* (*RMSE*)

Root Mean Squared Error (RMSE) adalah aturan untuk mengevaluasi nilai kesalahan dengan penskalaan kuadrat dan mengukur rata-rata besaran kesalahan. RMSE merupakan akar kuadrat dari rata-rata perbedaan kuadrat antara prediksi dan observasi aktual. Berikut ini rumus untuk menghitung nilai RMSE ditunjukkan pada Persamaan 10.

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(Y_t - \hat{Y}_t)^2} \quad (10)$$

Keterangan:

Y_t : Nilai data target ke-t; $t = 1,2,3,4, \dots, N$

$\hat{Y}_t$: Nilai prediksi data target ke-t; $t = 1,2,3,4, \dots, N$

N : Jumlah data latih

2.6.4 Proses Pengujian (*Testing*)

Proses pengujian memiliki tujuan untuk mengevaluasi metode ELM berdasarkan hasil proses pelatihan yang telah dilakukan. Dalam proses ini menggunakan bobot masukan, bias, dan bobot keluaran yang didapatkan dari proses pelatihan. Langkah-langkah yang dilakukan dalam proses pengujian sebagai berikut:

1. Inisialisasi bobot masukan dan bias yang telah didapatkan dari proses pelatihan.
2. Menghitung keluaran *hidden layer* dengan menggunakan data uji menggunakan Persamaan 11.

$$H_{init\ test} = X_{\text{test}} \cdot W^{\text{T}} + b \quad (11)$$

Keterangan:

$H_{init\ test}$: Matriks hasil keluaran *hidden layer* untuk proses pengujian

X_{test} : *Input* data menggunakan data pengujian

W^{T} : *Transpose* bobot masukan

b : Nilai bias

3. Keluaran dari *hidden layer* dihitung menggunakan fungsi aktivasi dengan Persamaan 12.

$$H = \frac{1}{1 + \exp^{-(H_{\text{init test}})}} \quad (12)$$

Keterangan:

H : Fungsi aktivasi sigmoid biner

exp : Eksponensial

$H_{\text{init test}}$: Matriks keluaran *hidden layer* pada proses pengujian

4. Nilai bobot keluaran yang telah didapatkan pada proses pelatihan digunakan untuk menghitung keluaran *output layer* yang merupakan hasil prediksi. Berikut ini rumus untuk menghitung nilai *output layer* dapat dilihat pada Persamaan 13.

$$\hat{Y} = H \cdot \hat{\beta} \quad (13)$$

Keterangan:

$\hat{Y}$: *Output layer* yang merupakan hasil prediksi

$\hat{\beta}$: Nilai bobot keluaran dari proses pelatihan

H : Keluaran di *hidden layer* dihitung dengan fungsi aktivasi

2.6.5 Denormalisasi Data

Denormalisasi data merupakan kebalikan dari proses normalisasi data. Selain itu, denormalisasi data termasuk ke dalam metode *post-processing* yang dilakukan dengan mengubah skala data setelah dilakukan tahap pelatihan (*training*). Proses ini berfungsi untuk membangkitkan nilai yang telah dinormalisasi menjadi nilai asli (awal). Berikut ini adalah persamaan

untuk proses denormalisasi data dapat dilihat pada Persamaan 14.

$$\hat{Y}_t = \frac{(\hat{Y}'_t - 0.1)(\max(x) - \min(x))}{0.8} + \min(x) \quad (14)$$

Keterangan:

$\hat{Y}_t$: Data prediksi ke-t hasil denormalisasi

$\hat{Y}'_t$: Data prediksi ke-t hasil normalisasi

$\min(x)$: Nilai minimum data sebelum di normalisasi

$\min(x)$: Nilai maksimum data sebelum di normalisasi

2.7 *Artificial Bee Colony* (ABC)

Artificial Bee Colony (ABC) merupakan metode optimasi yang terinspirasi dari perilaku *bee colony* (kawanan lebah) dalam mencari dan mengeksplorasi sumber makanan secara efisien. Pada algoritma ABC, solusi atas permasalahan optimasi digambarkan sebagai sumber makanan (nektar) (Akay dan Karaboga, 2012). Algoritma ABC dikembangkan oleh Dervis Karaboga pada tahun 2005. Kinerja dari ABC memiliki kualitas yang lebih baik atau setara dengan algoritma *swarm* lain seperti *Genetic Algorithm*, *Particle Swarm Optimization*, *Differential Evolution*, dan *Evolution Strategies* dengan keuntungan penggunaan parameter kontrol yang lebih sedikit (Karaboga dan Akay, 2009). Pada algoritma ABC, *bee colony* terbagi menjadi 3 jenis yaitu *employee bee* (lebah pekerja), *onlooker bee* (lebah penjaga), dan *scout bee* (lebah pengintai). *Employee bee* bertugas mencari solusi (nektar) dan menghitung nilai *fitness* (kualitas dari solusi) serta memberikan informasi posisi solusi pada *onlooker bee* dengan melakukan *waggle dance*. *Bee*

colony menggunakan *waggle dance* untuk berkomunikasi mengenai kualitas dari solusi yang menggambarkan nilai *objective function* dan *onlooker bee* bertugas membandingkan nilai probabilitas tertinggi dari setiap solusi sehingga jumlah *onlooker bee* sama dengan jumlah solusi. Selanjutnya *employee bee* pada daerah solusi akan mencari solusi baru secara *acak* pada area sekitar (*neighborhood*). Pada saat itulah *employee bee* berubah menjadi *scout bee*. Hal ini menyebabkan *employee bee*, *scout bee*, dan solusi memiliki jumlah yang sama. Tahapan tersebut dilakukan hingga menemukan solusi optimal.

Langkah-langkah dari algoritma ABC dalam melakukan optimasi sebagai berikut (Fei dan He, 2015):

1. *Employee bee* melakukan inisialisasi posisi awal solusi secara acak sejumlah populasi *employee bee* menggunakan Persamaan 15 dan *set* nilai *trial* = 0 untuk setiap solusi.

$$x_{ij} = x_{minj} + \text{rand } [0,1] \times \left(x_{maxj} - x_{minj}\right) \quad (15)$$

Keterangan:

x_{ij} : Posisi solusi ke-i dengan parameter ke-j

x_{minj} : Nilai batas bawah dari dimensi solusi ke-j

x_{maxj} : Nilai batas atas dari dimensi solusi ke-j

rand [0,1] : Bilangan acak rentang [0,1]

i : Jumlah solusi

j : Jumlah dimensi solusi

2. *Employee bee* menghitung nilai *fitness* dari setiap solusi dengan Persamaan 16.

$$fitness_{\ i} = \frac{1}{1 + \ Obj\ Fun._{\ i}} \quad (16)$$

Keterangan:

$fitness_{\ i}$: Nilai *fitness* solusi ke-i

$Obj\ Fun._{\ i}$: Nilai RMSE

3. Selanjutnya *employee bee* melakukan *update* posisi pada setiap solusi (x_{ij}) agar mendapat solusi baru (x_{ijnew}) menggunakan Persamaan 17 dan menghitung nilai *fitness*-nya.

$$x_{ijnew} = x_{ij} + \phi[-1,1] \times (x_{ij} - x_{kj}) \quad (17)$$

Keterangan:

x_{ijnew} : Posisi solusi baru dari x_{ij}

$\phi[-1,1]$: Bilangan acak rentang [-1,1]

x_{ij} : Posisi solusi ke-i

x_{kj} : Posisi solusi ke-k tetangga dari solusi ke-i, dimana $k \neq i$

$i, j, k \in \{1,2,3, \dots, m\}$

m : Jumlah solusi

Kemudian dilakukan evaluasi nilai *fitness* dari x_{ij} dan x_{ijnew}. Jika $x_{ijnew} > x_{ij}$, maka x_{ijnew} akan menjadi solusi baru dan nilai *trial* di-*reset* menjadi 0. Jika sebaliknya, x_{ij} akan dipertahankan dan nilai *trial* ditambah 1.

4. *Onlooker bee* menghitung nilai probabilitas dari nilai *fitness* menggunakan Persamaan 18.

$$P_i = \frac{fitness_{\ i}}{\sum_{i=1}^{m} fitness_{\ i}} \quad (18)$$

Keterangan:

P_i : Probabilitas dari solusi ke-i

$fitness_{\ i}$: Nilai *fitness* dari solusi ke-i

5. Lalu *onlooker bee* akan memilih solusi baru x_{ijnew2} dan Jika x_{ijnew} menggunakan proses seleksi *Roulette Wheel* (RW). Proses seleksi ini dipilih karena menggunakan probabilitas nilai *fitness* sehingga nilai *fitness* yang besar memiliki kesempatan untuk terpilih lebih besar dibandingkan dengan nilai *fitness* yang memiliki nilai lebih kecil. Setelah terpilih, dilakukan *update* posisi dan evaluasi nilai *fitness* seperti langkah 3.
6. Kemudian *scout bee* menghitung nilai *trial* dari setiap solusi yang dipilih oleh *onlooker bee*. Jika nilai *maximum trial* ≥ *limit trial* yang sudah ditentukan dan tidak ada perbaikan nilai *fitness*, maka *scout bee* akan mengganti solusi baru menggunakan Persamaan 15, lalu *reset* nilai *trial* menjadi 0. Jika *maximum trial* ≥ *limit trial* dan terdapat perbaikan nilai *fitness*, maka tidak perlu diganti, lalu *reset* nilai *fitness* menjadi 0. Jika *maximum trial* ≤ *limit trial*, maka tidak perlu diganti dan nilai *trial* tidak perlu di-*reset*.
7. Selanjutnya *scout bee* menyimpan satu posisi solusi terbaik jika $(x_{globalBest})$ yang sudah ditemukan.
8. Ulangi langkah 3 sampai dengan 7, jika iterasi *maximum* yang sudah ditentukan belum tercapai.

2.8 *Modified Artificial Bee Colony*

Modified Artificial Bee Colony merupakan pengembangan atau bentuk modifikasi dari metode optimasi *Artificial Bee Colony* (ABC). Modifikasi algoritma ABC memberikan performa konvergensi yang lebih baik jika dibandingkan dengan algoritma ABC biasa (Shahrudin dan Mahmuddin, 2014). Berikut ini merupakan tahapan dari algoritma MABC dalam melakukan optimasi:

1. Inisialisasi populasi awal akan dibangkitkan menggunakan Persamaan 19.

$$x_{ij} = x_{minj} + \text{rand } [0,1] \times \left(x_{maxj} - x_{minj}\right) \quad (19)$$

Keterangan:

x_{ij} : Posisi solusi ke-i dengan parameter ke-j

x_{minj} : Nilai batas bawah dari dimensi solusi ke-j

x_{maxj} : Nilai batas atas dari dimensi solusi ke-j

rand [0,1] : Bilangan acak rentang [0,1]

i : Jumlah solusi

j : Jumlah dimensi solusi

2. Setelah populasi awal dibangkitkan, kemudian mencari nilai *fitness* untuk setiap solusi menggunakan Persamaan 20.

$$fitness_{\ i} = \frac{1}{1 + \ Obj\ Fun._{\ i}} \quad (20)$$

Keterangan:

$fitness_{\ i}$: Nilai *fitness* solusi ke-i

$Obj\ Fun._{\ i}$: Nilai RMSE

3. Selanjutnya *employee bee* mencari solusi baru (*update* posisi pada setiap solusi) menggunakan bentuk modifikasi yang dilakukan adalah pada formula algoritma ABC dapat dilihat pada Persamaan 21.

$$x_{ijnew} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) + \varphi_{ij}(y_j - x_{ij}) \quad (21)$$

Keterangan:

ϕ_{ij} : Angka acak berdistribusi *uniform* dalam rentang 0 hingga 1.5

y_j : Elemen ke-j dari solusi *global* terbaik

Kemudian melakukan evaluasi nilai *fitness* dari x_{ij} dan x_{ijnew}. Jika $x_{ijnew} > x_{ij}$, maka x_{ijnew} akan menjadi solusi baru dan nilai *trial* di-*reset* menjadi 0. Jika sebaliknya, x_{ij} akan dipertahankan dan nilai *trial* ditambah 1.

4. *Onlooker bee* akan memilih solusi berdasarkan probabilitas menggunakan Persamaan 22. Formula pada modifikasi algoritma ABC terinspirasi dari mekanisme pencarian algoritma Particle Swarm Optimization (PSO) dan digunakan untuk memperbaiki tingkat konvergensi dari algoritma PSO.

$$P_i = \exp\left(-\frac{1}{\rho} fitness_{\ i}\right) \quad (22)$$

Keterangan:

P_i : Probabilitas dari solusi ke-i

$fitness_{\ i}$: Nilai *fitness* dari solusi ke-i

ρ : 2.5

Lalu *onlooker bee* akan memilih solusi baru x_{ijnew2} dan Jika x_{ijnew} menggunakan proses seleksi *Roulette Wheel* (RW). Setelah terpilih, dilakukan *update* posisi dan evaluasi nilai *fitness* seperti langkah 3.

5. Kemudian *scout bee* menghitung nilai *trial* dari setiap solusi yang dipilih oleh *onlooker bee*. Jika ada solusi yang ditinggalkan, maka *scout bee* akan mengganti solusi baru secara acak menggunakan Persamaan 21.
6. Selanjutnya *scout bee* menyimpan satu posisi solusi terbaik jika ($x_{globalBest}$) yang sudah ditemukan.
7. Ulangi langkah 3 sampai dengan 5, sampai persyaratan terpenuhi atau iterasi maksimum.

Menurut Shahrudin dan Mahmuddin (2014), mengemukakan bahwa berapapun nilai yang digunakan pada simulasi penelitian yang telah dilakukan akan menghasilkan kesimpulan *Modified Artificial Bee Colony* (MABC) lebih baik daripada *Artificial Bee Colony* (ABC).

2.9 Ukuran Kesalahan Peramalan

Dalam suatu peramalan terdapat evaluasi terhadap hasil peramalan dengan menggunakan ukuran kesalahan peramalan. Hasil peramalan terbaik memiliki nilai kesalahan peramalan yang terkecil. Ukuran kesalahan yang digunakan dalam penelitian ini adalah nilai rata-rata kesalahan persentase absolut (*Mean Absolute Percentage Error*). Nilai *Mean Absolute Percentage Error* (MAPE) merupakan pengukuran besarnya kesalahan dengan menghitung ukuran persentase

penyimpangan antara data aktual dengan data peramalan yang diperoleh. Berikut rumus dalam menghitung nilai MAPE yaitu:

$$\text{MAPE} = \frac{1}{N}\sum_{t=1}^{N}\frac{\left|Y_t - \hat{Y}_t\right|}{Y_t} \times 100\% \quad (23)$$

Keterangan:

N : Jumlah data uji

Y_t : Nilai data target ke-t; $t = 1,2,3,4, \dots, N$

$\hat{Y}_t$: Nilai prediksi data target ke-t; $t = 1,2,3,4, \dots, N$

Menurut Moreno et al. (2013), kriteria dari nilai MAPE sebagai berikut:

Tabel 2.1. Kriteria Nilai MAPE

MAPE	Kriteria
<10	Kemampuan peramalan sangat baik
10-20	Kemampuan peramalan baik
20-50	Kemampuan peramalan layak/memadai
>50	Kemampuan peramalan buruk

Sedangkan untuk menghitung nilai akurasi digunakan hasil nilai MAPE yang ditunjukkan pada Persamaan 24 (Jauhari et al., 2016).

$$Akurasi = 100\% - \text{MAP}E \quad (24)$$