BAB II

TINJAUAN PUSTAKA

2. 1 Indeks Harga Saham Gabungan

Indeks Harga Saham Gabungan (IHSG) merupakan angka indeks harga saham yang sudah disusun dan dihitung dengan menghasilkan *trend*, dimana angka indeks adalah angka yang diolah sedemikan rupa sehingga dapat digunakan untuk membandingkan kejadian yang berupa perubahan harga saham dari dari waktu ke waktu (Jogiyanto, 2013 :147). Indeks Harga Saham Gabungan (IHSG) yang dapat disebut juga *Indonesia Composite Index*, ICI, atau IDX *Composite* merupakan salah satu indeks pasar saham yang digunakan oleh Bursa Efek Indonesia (BEI). IHSG diperkenalkan pertama kali pada tanggal 1 april 1983, sebagai indikator pergerakan harga saham di Bursa Efek Indonesia (BEI). Indeks ini mencakup pergerakan harga seluruh saham biasa dan saham preferen yang tercatat di BEI. Hari Dasar untuk perhitungan IHSG adalah tanggal 10 Agustus 1982. Indeks ditetapkan dengan nilai dasar 100 dan saham tercatat pada saat itu berjumlah 13 saham. Perhitungan IHSG adalah sebagai berikut:

$$IHSG = \frac{\sum(p.x)}{d} x\ 100 \qquad (2.1)$$

dimana :

p = Harga Penutupan di Pasar Reguler,

x = Jumlah Saham, dan

d = Nilai Dasar.

2. 2 Peramalan

Peramalan atau *forecasting* merupakan suatu teknik untuk memperkirakan suatu nilai pada masa yang akan datang dengan memperhatikan data atau informasi masa lalu maupun saat ini. Menurut Render dan Heizer (2001) peramalan adalah sebuah ilmu peramalan peristiwa masa depan dengan menggunakan beberapa bentuk model matematis. Salah satu manfaat dari peramalan yaitu untuk melihat gambaran-gambaran tentang masa yang akan datang sehingga dapat mengantisipasi apa yang akan terjadi.

2. 3 Jaringan Syaraf Tiruan

Jaringan Syaraf Tiruan (JST) merupakan salah satu bagian dari *machine learning*. JST adalah teknik untuk memodelkan pemrosesan informasi berdasarkan kemampuan sistem syaraf pada otak manusia. JST menggunakan cara kerja dari syaraf pada otak manusia untuk pengolahan informasi dengan menyelesaikan sejumlah proses perhitungan pada komputer (Kusumadewi, 2003).

2.3.1 Komponen Jaringan Syaraf Tiruan

Menurut Puspitaningrum (2006) lapisan dalam jaringan syaraf tiruan dibagi menjadi 3 bagian antara lain:

1. Lapisan Masukan (*Input Layer*)

   Lapisan ini berisi neuron yang mempunyai nilai *input* masing-mansing. *Input* dari lapisan ini merupakan penggambaran suatu masalah.

2. Lapisan Tersembunyi (*Hidden Layer*)

   Hidden Layer merupakan lapisan yang menghubungkan antara *input layer* dan *output layer*. Lapisan ini merupakan lapisan yang menjalankan semua proses *training* dan proses *testing*.

3. Lapisan Keluaran (*Output Layer*)

   Nilai *output* dari hasil perhitungan keseluruhan yang merupakan output jaringan syaraf tiruan dari suatu permasalahan.

2.3.2 Arsitektur Jaringan Syaraf Tiruan

Menurut Kusumadewi (2003) arsitektur jaringan syaraf tiruan memiliki 3 jenis model yaitu :

1. Jaringan *Layer* Tunggal (*Single Layer Network*)

   Jaringan *layer* tunggal merupakan jaringan yang hanya memiliki satu lapisan. Dimana pada jaringan ini input yang diterima akan langsung diolah menjadi output tanpa harus melalui lapisan *hidden*.

2. Jaringan *Layer* Banyak (*Multi Layer Network*)

   Jaringan *layer* banyak merupakan jaringan yang memiliki lapisan lebih dari satu yang terletak di antara lapisan *input* dan *output* (memiliki 1 atau lebih lapisan tersembunyi). Kelebihan dari penggunaan Jaringan *layer* banyak dibanding jaringan *layer* tunggal yaitu dapat menyelesaikan masalah yang lebih sulit. Namun proses pelatihannya lebih rumit dan lama.

3. Jaringan *Reccurent*

   Umumnya, pada lapisan *reccurent* hubungan antar *neuron* tidak diperlihatkan pada diagram arsitektur.

2.3.3 Fungsi Aktivasi

Fungsi aktivasi dalam jaringan syaraf tiruan digunakan untuk menentukan *output* dari suatu *neuron* dengan argumen *net input*. *Net input* merupakan kombinasi linier *input* dan bobot (Siang, 2009). Selain itu fungsi aktivasi bertujuan untuk memodifikasi *output* ke dalam rentang nilai tertentu. Fungsi aktivasi yang sering digunakan dalam jaringan syaraf tiruan yaitu (Siang, 2009) :

1. Fungsi *Sigmoid Biner*

   Fungsi *sigmoid biner* memiliki interval output 0 sampai 1. Fungsi *sigmod biner* dirumuskan sebagai berikut :

$$f(x) = \frac{1}{1+e^{-x}} \quad (2.2)$$

   Keterangan :

   f(x) = Fungsi aktivasi *sigmoid biner*

   e = Eksponensial

   x = data ke- x

2. Fungsi *Sigmoid Bipolar*

   Fungsi *sigmoid bipolar* memiliki rentang nilai -1 sampai 1. Fungsi *sigmod bipolar* dirumuskan sebagai berikut :

$$f(x) = \frac{1-e^{-x}}{1+e^{-x}} \quad (2.3)$$

Keterangan :

f(x) = Fungsi aktivasi sigmoid bipolar

e = Eksponensial

x = data ke- x

2. 4 *Extreme Learning Machine*

Extreme Learning Machine (ELM) merupakan pengembangan dari metode jaringan syaraf tiruan yang pertama kali ditemukan oleh Huang, Zhu, dan Siew pada tahun 2004. Metode ELM sering disebut *Single Hidden Layer Feedforward Neural Networks* (SLNFs) karena hanya mempunyai satu *hidden layer* pada arsitektur jaringannya. Metode ini dibuat sebagai solusi untuk mengatasi permasalahan yang terdapat dalam JST *feedforward* terutama dalam hal *learning speed*. Parameter-parameter yang digunakan pada ELM seperti *input weight* dan bias ini dibangkitkan secara acak dalam suatu rentang tertentu. Namun dalam penerapannya untuk mendapatkan *input weight* dan bias yang optimal dapat dilakukan dengan optimasi. Kelebihan dari metode ini yaitu hanya menggunakan satu *hidden layer* dan penggunaan *input weight* dan *bias* mampu memberikan hasil prediksi yang lebih akurat dengan *learning speed* yang lebih cepat dibandingkan dengan jaringan *feedworward* biasa (Huang, et. al., 2004). Secara umum fungsi matematis metode ELM dengan jumlah *hidden nodes* sebanyak N dan *activation function* g(x) dapat dilihat pada persamaan (2.4).

$$\sum_{i=1}^{\bar{N}} \beta_i g_i(x_j) = \sum_{i=1}^{\bar{N}} \beta_i g_i(W_i X_j + b_i) \quad (2.4)$$

Keterangan :

j = 1, 2, … , N

W_i = $(W_{i1}, W_{i2}, \dots, W_{in})^T$, vektor dari *weight* yang menghubungkan *hidden nodes* ke-i dan *input nodes*

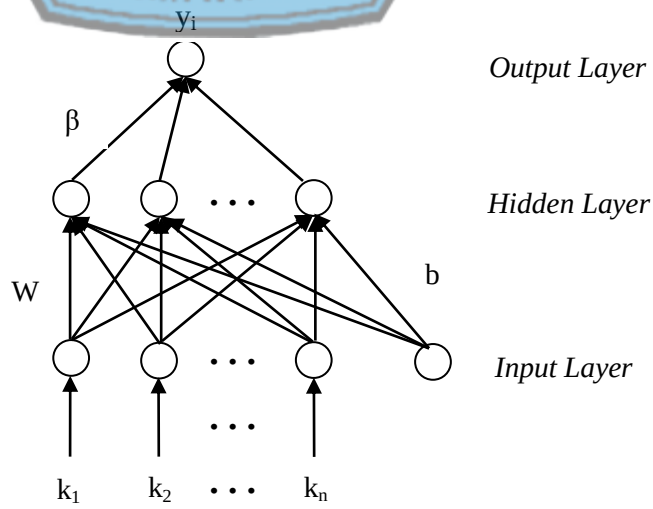
β_i = $(\beta_{i1}, \beta_{i2}, \dots, \beta_{im})^T$, vektor dari bobot yang menghubungkan *hidden nodes* ke-i dan *output nodes*

b_i = *threshold* dari *hidden nodes* ke-i

W_iX_i = *inner* produk dari W_i dan X_i

2.4.1 Struktur Jaringan Metode ELM

Struktur jaringan pada metode ELM terdiri dari 3 lapisan, yaitu *input layer*, *hidden layer*, dan *output layer* dimana setiap node pada masing–masing layer saling dihubungkan oleh bobot dengan nilai yang berbeda–beda namun saling terhubung menuju satu *output*. Struktur jaringan pada metode ELM dapat dilihat pada Gambar 2.1 (Huang, et. al., 2004).



Gambar 2.1 Struktur Jaringan Metode ELM

2.4.2 Normalisasi Data

Normalisasi data merupakan proses perubahan bentuk data menjadi nilai dalam batas 0-1. Proses ini ditujukan untuk menyesuaikan data *input* terhadap data *output*. Menurut Siang (2005) perhitungan normalisasi data berdasarkan pada persamaan (2.5).

$$Y'_t = \frac{0{,}8(Y_t - min(x))}{(max(x) - min(x)} + 0.1 \quad (2.5)$$

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*

$max(x)$ = Nilai maksimum data sebelum di *normalisasi*

2.4.3 Proses *Training*

Tujuan dari proses *training* yaitu untuk mendapatkan *output weight* dengan tingkat kesalahan yang rendah. Menurut Chandra, et. al. (2018) langkah-langkah proses *training* metode ELM yaitu :

1. Inisialisasi *input weight* dan bias dengan bilangan acak yang kecil secara *random*
2. Menghitung *output* di *hidden layer* menggunakan fungsi aktivasi. Perhitungan *output hidden layer* berdasarkan pada persamaan (2.6).

$$Hnet_j = \left(\sum_{k=1}^{n} x_{ik} w_{jk}\right) + b_j \quad (2.6)$$

Keterangan :

$Hnet_j$ = nilai *output hidden layer*

i = (1, 2, … , N), N adalah jumlah data

j = (1, 2, … , $\bar{N}$), $\bar{N}$ adalah jumlah *hidden neuron*

k = Jumlah *input neuron*

w = Bobot *input* dengan ukuran ordo matriks, *hidden neuron* x *input neuron*

x = data *input*

b = bias

3. Menghitung fungsi aktivasi *sigmoid biner* berdasarkan pada persamaan (2.7).

$$H_{ij} = \frac{1}{1+e^{-(Hnet_j)}} \quad (2.7)$$

Keterangan :

H_{ij} = Matriks dari fungsi aktivasi *output* pada *hidden layer*

e = Eksponensial

$Hnet_j$ = nilai *output hidden layer*

4. Menghitung Matriks *Moore-Penrose Generalized Inverse/ Moore-Penrose Pseudo Inverse* (H^+) berdasarkan pada persamaan (2.8).

$$H^+ = (H^T H)^{-1} H^T \quad (2.8)$$

Keterangan :

H^+ = Matriks *Moore-Penrose Generalized Inverse*.

H^T = Matriks H yang telah di*transpose*

H = Matriks H hasil keluaran *hidden layer* yang telah diaktivasi

$(H^T H)^{-1}$ = *Inverse* dari perkalian matriks H^Tdengan H

5. Menghitung *output weight* dari *hidden layer* ke *output layer* berdasarkan pada persamaan (2.9).

$$\beta = H^{+}T \tag{2.9}$$

Keterangan :

β = Matriks *output weight* dari hidden layer ke *output layer*

H^{+} = Matriks *Moore-Penrose Generalized Inverse*

T = Matriks Target.

2.4.4 Proses *Testing*

Proses *testing* bertujuan untuk mengevaluasi kemampuan metode ELM dalam memprediksi, yang dilakukan berdasarkan *input weight*, bias, dan *output weight* dari perhitungan *training*. Menurut Chandra, et. al. (2018) langkah-langkah proses *testing* yaitu :

1. Inisialisasi *input weight* dan bias yang telah didapatkan dari proses *training*
2. Menghitung semua *output* di *hidden layer* menggunakan fungsi aktivasi menggunakan persamaan (2.6) dan (2.7).
3. Menghitung *output* pada *output layer* menggunakan hasil *output weight* dari *hidden layer* ke *output layer* dari proses *training* yang akan menghasilkan nilai prediksi, yang dapat dihitung berdasarkan pada Persamaan (2.10).

$$y = H\beta \tag{2.10}$$

Keterangan :

y = *Output* hasil prediksi

H = Matriks H hasil keluaran *hidden layer* yang telah diaktivasi

β = Matriks *output weight* dari *hidden layer* ke *output layer*

2.4.5 Denormalisasi Data

Denormalisasi data merupakan proses yang bertujuan untuk mengembalikan nilai yang telah dinormalisasi menjadi nilai asli. Menurut Siang (2005) perhitungan denormalisasi data sebagai berikut :,

$$\hat{Y}_t = \frac{(\widehat{Y'}_t - 0.1)(\max(x) - \min(x))}{0.8} + \min(x) \quad (2.11)$$

Keterangan :

$\widehat{Y'}_t$ = Nilai hasil normalisasi

$\hat{Y}_t$ = Hasil denormalisasi

$min(x)$ = Nilai minimum data sebelum di *normalisasi*

$max(x)$ = Nilai maksimum data sebelum di *normalisasi*

2. 5 *Bat Algorithm*

Bat Algorithm (BA) adalah salah satu algoritma *metaheuristik* yang diciptakan oleh Yang dan Xin-She pada tahun 2010 yang diadaptasi dari perilaku *echolocation* kelelawar dalam mencari makanan. Kemampuan *echolocation* ini membuat kelelawar dapat menemukan mangsa dan dapat membedakan rintangan dengan sumber makanan. Menurut Yang dan Xin-She (2010) aturan dari perilaku *bat* diasumsikan sebagai berikut :

1. Setiap *bat* menggunakan *echolocation* untuk mengetahui jarak dan membedakan antara makanan dan rintangan.

2. Setiap *bat* terbang secara acak pada kecepatan (V_i) dan posisi (X_i) dengan *frekuensi* (f), pang gelombang (λ), *loudness* (A) serta kecepatan *pulse rate* (r) ∈ [0,1] dalam mencari mangsa untuk menentukan target terdekat.
3. Tingkat *loudness* dapat bervariasi dalam berbagai hal, diasumsikan *loudness* bervariasi dari nilai positif yang maksimum (A_0) sampai ke nilai konstan minimum (A_{min}).

Menurut Yang dan Xin-She (2010) implementasi *Bat Algorithm* untuk optimasi sebagai berikut :

2.5.1 Posisi

Posisi *bat* merupakan representasi dari tiap solusi. Penyesuaian posisi *bat* dapat dihitung dengan persamaan (2.12).

$$x_i^t = x_i^{t-1} + v_i^t \tag{2.12}$$

Keterangan :

x_i^t = posisi *bat* ke – i pada iterasi ke- t

x_i^{t-1} = posisi *bat* ke – i pada iterasi t-1

v_i^t = kecepatan *bat* ke – i pada iterasi ke – t

2.5.2 *Frekuensi*

Frekuensi merupakan *elemen* bilangan *real* yang akan mempengaruhi nilai kecepatan. Pemilihan nilai minimum dan maksimum dapat dihitung dengan persamaan (2.13).

$$f_i = f_{min} + (f_{max} - f_{min})\beta \tag{2.13}$$

Keterangan :

f_i = *frekuensi bat* ke- i

f_{min} = nilai minimum *frekuensi*

f_{max} = nilai maksimum *frekuensi*

β = variabel *konstan* diantara 0 dan 1

2.5.3 *Loudness*

Loudness (A) merupakan perubahan dalam suatu iterasi yang terjadi selama pencarian lokal di sekitar posisi *bat* terbaik global dan pencarian lokal di setiap *bat*.

$$x_{new} = x_{old} + \varepsilon A^t \quad (2.14)$$

Keterangan :

x_{new} = posisi *bat* baru

x_{old} = posisi *bat* sebelumnya

A^t = *loudness* rata-rata semua *bat* dalam satu iterasi

Ɛ = bilangan acak diantara -1 dan 1

Loudness berada di rentang nilai maksimum dan minimum yang telah ditentukan. *Loudness* akan berkurang saat *bat* mulai mendekati posisi terbaiknya. Nilai *loudness* dapat dihitung dengan persamaan (2.15).

$$A_i^{t+1} = a + A_i^t \quad (2.15)$$

Keterangan :

A_i^{t+1} = *loudness bat* ke – i untuk iterasi ke t + 1

A_i^t = *loudness bat* ke – i pada iterasi ke – t

α = variabel konstan diantara 0 dan 1

2.5.4 Kecepatan

Kecepatan masing-masing *bat* direpresentasikan dengan bilangan integer positif. *Bat* berkomunikasi satu dengan yang lainnya melalui posisi global terbaik dan bergerak menuju posisi terbaik global. Kecepatan pada *bat* dapat dihitung dengan persamaan (2.16).

$$v_i^t = v_i^{t-1} + (x^* - x_i^t) f_i \quad (2.16)$$

Keterangan :

v_i^t = kecepatan *bat* ke – i pada iterasi ke – t

v_i^{t-1} = kecepatan *bat* ke – i pada iterasi t–1

$(x^* - x_i^t)$ = perbedaan posisi *bat* global (*) dengan *bat* ke – i pada iterasi ke – t.

2.5.5 *Pulse Rate*

Pulse rate (r) berperan dalam penentuan waktu pencarian lokal *bat* terbaik global yang dilewati. Ketika *bat* mendekati posisi terbaik, nilai *pulse rate* akan perlahan – lahan berkurang. Nilai *pulse rate* dapat dihitung dengan persamaan (2.17).

$$r^{t+1} = r^0 + [1 - exp(-\gamma t)] \quad (2.17)$$

Keterangan :

r^{t+1} = *pulse rate bat* pada iterasi t+1

r^0 = nilai *pulse rate* awal pada *bat*

γ = variabel *konstan* diantara 0 dan 1

2. 6 Ukuran Kesalahan Peramalan

Ukuran kesalahan peramalan digunakan untuk mengevaluasi suatu hasil peramalan. Hasil peramalan yang baik memiliki nilai kesalahan peramalan yang kecil. Ukuran kesalahan yang digunakan dalam penelitian ini adalah nilai rata-rata kesalahan presentase absolut (*Mean Absolute Percentage Error*) karena MAPE termasuk ukuran standar statistik. MAPE merupakan pengukuran besarnya kesalahan dengan menghitung ukuran presentase penyimpangan antara data aktual dengan data peramalan yang diperoleh. Rumus perhitungannya adalah sebagai berikut:

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{y_t - \hat{y}_t}{y_t}\right| x\ 100\% \qquad (2.18)$$

Keterangan :

n = Jumlah data

$\hat{y}_t$ = Nilai prediksi pada periode ke-t

y_t = Nilai data aktual pada periode ke-t

Menurut Chang, Wang dan Liu (2007) kriteria nilai MAPE adalah sebagai berikut:

Tabel 2.1 Kriteria MAPE

MAPE	Pengertian
<10%	Kemampuan Peramalan Sangat Baik
10%-20%	Kemampuan Peramalan Baik
20%-50%	Kemampuan Peramalam Cukup
>50%	Kemampuan Peramalan Buruk