LAPORAN TUGAS 2 LONG-SHORT TERM MEMORY

IF4074 PEMBELAJARAN MESIN LANJUT

Disusun oleh:

MAHESA LIZARDY 13520116

BRYAN AMIRUL HUSNA 13520146

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2023

DAFTAR ISI

BAB I
PENDAHULUAN

A. Long Short Term Memory (LSTM)

Long Short Term Memory (LSTM) adalah salah satu jenis arsitektur jaringan saraf tiruan yang digunakan dalam bidang pembelajaran mesin, khususnya dalam tugas-tugas yang melibatkan urutan data, seperti pengenalan ucapan, penerjemahan bahasa, dan prediksi urutan waktu. LSTM dirancang untuk mengatasi masalah vanishing gradient pada jaringan saraf rekurensi (RNNs), yang menyebabkan RNNs sulit melatih data urutan yang panjang. LSTM mempertahankan suatu sel memori internal yang dapat menyimpan informasi dalam jangka waktu yang lama. LSTM memiliki tiga gate dan satu *state* sebagai berikut.

a. Cell State

Cell state merupakan sel memori dari jaringan. bertindak sebagai perantara yang mentransfer informasi yang relevan sepanjang pemrosesan *sequence*.

b. Forget Gate

Forget Gate menentukan informasi mana yang akan dilupakan atau disimpan dari sel memori sebelumnya. Gerbang ini membantu LSTM untuk mempertahankan informasi yang relevan dan menghapus informasi yang tidak relevan.

c. Input Gate

Gerbang ini menentukan informasi baru apa yang akan disimpan dalam sel memori. Ini melibatkan dua langkah. Pertama, memutuskan informasi apa yang akan di update dari sel memori sebelumnya. kedua, memutuskan informasi baru apa yang akan disimpan dalam sel memori.

d. Output Gate

Output Gate mengontrol informasi apa yang akan dikirimkan ke lapisan berikutnya dari jaringan. Sel memori yang telah diproses oleh input gate dan forget gate akan menghasilkan keluaran dari LSTM.

Keberadaan *gate* ini memungkinkan LSTM untuk mempertahankan informasi dalam jangka waktu yang lama, mengatasi masalah vanishing gradient, dan menjaga aliran informasi dalam jaringan, sehingga cocok untuk tugas-tugas yang melibatkan urutan data yang panjang. LSTM telah membantu memecahkan berbagai masalah di berbagai bidang, dan terus menjadi area penelitian yang aktif dalam pembelajaran mesin dan kecerdasan buatan.

BAB II
IMPLEMENTASI

A. Long Short Term Memory

Pada bagian ini merupakan implementasi dari *Long Short Term Memory*. Kelas diberi nama LSTM dan terdapat pada berkas LSTM.py. Metode-metode yang terdapat pada kelas ini di antaranya:

a. __init__(self, input_shape, num_units, activation=None)
   Fungsi ini merupakan konstruktor kelas LSTM. Fungsi Ini digunakan untuk menginisialisasi berbagai parameter yang diperlukan untuk lapisan LSTM. Parameter yang akan diinisialisasi yaitu *weight* untuk input (U), *weight* untuk hidden (W), dan bias (b) untuk setiap *cell state, forget gate, input gate*, dan *output gate*. Selain itu akan juga diinisialisasi *activation function* yang akan digunakan dengan defaultnya yaitu relu.

b. sigmoid(self, x)
   Fungsi ini menerapkan fungsi aktivasi sigmoid pada input x, yang menghasilkan nilai antara 0 dan 1.

c. tanh(self, x)
   Fungsi ini akan menghitung nilai tanh dari input x.

d. relu(self, x)
   Fungsi ini menerapkan fungsi aktivasi ReLU (Rectified Linear Unit) pada input x. ReLU mengembalikan nilai maksimum antara 0 dan input.

e. forward(self, X)
   Fungsi ini akan melakukan operasi forward propagation. Pertama akan diinisialisasi nilai cell state dan hidden state dengan nilai 0. Setelah itu akan dilakukan pemrosesan setiap *sequence* yang ada pada input sesuai urutan. Akan dihitung nilai *forget gate*, *input gate*, *candidate cell state*, *cell state*, *output gate* dan *hidden state*. Nilai *cell state* dan *hidden state* akan di *update* untuk operasi *sequence* selanjutnya. Fungsi ini akan me-*return* nilai *hidden state sequence* terakhir yang telah dilakukan *activation function* sebelumnya.

f. set_weight_hidden(self,state, weight)
   Fungsi ini akan melakukan setting *weight* hidden untuk *state* / *gate* tertentu dengan nilai input weight.

g. set_weight_input(self,state, weight)
   Fungsi ini akan melakukan setting *weight* input untuk *state* / *gate* tertentu dengan nilai input weight.

h. set_bias(self,state, bias)
   Fungsi ini akan melakukan setting *bias* input untuk *state* / *gate* tertentu dengan nilai input weight.

i. getModel(self)
   Fungsi ini akan dipanggil ketika melakukan save model. Fungsi ini akan mengembalikan json yang berisi parameter weight dan bias pada layer LSTM.

j. setModel(self, modelJson)
   Fungsi ini akan dipanggil ketika melakukan *load* model. Fungsi ini akan melakukan setting parameter layer LSTM dengan input modelJson

k. summary(self, lwidth, owidth, pwidth)
   Fungsi ini akan menampilkan ringkasan (summary) dari arsitektur LSTM, termasuk bentuk output (output shape) dan jumlah parameter (number of parameters) yang digunakan dalam jaringan.

B. Scaler

Pada bagian ini merupakan implementasi dari *scaler*, yaitu pemetaan nilai menjadi rentang 0 sampai 1. Kelas diberi nama Scaler dan terdapat pada berkas Scaler.py. Metode-metode yang terdapat pada kelas ini di antaranya:

a. __init__(self)
   Fungsi ini merupakan konstruktor kelas Scaler. Fungsi Ini digunakan untuk menginisialisasi data_min dan data_max menjadi bernilai None.

b. fit(self, X)
   Fungsi ini digunakan untuk “melatih” *scaler*, yaitu menyimpan nilai minimum dan maksimum yang akan digunakan untuk transformasi.

c. transform(self, X)
   Fungsi ini mentransformasikan data dalam rentang 0 sampai 1 berdasarkan nilai minimum dan maksimum yang tersimpan.

d. inverse_transform(self, X)
   Fungsi ini mengembalikan masukan dalam rentang 0 sampai 1 menjadi balikannya, yaitu nilai awal sebelum dilakukan transformasi.

BAB III
EKSPERIMEN

A. Rancangan Eksperimen

Eksperimen dilakukan pada dataset Stock market BMRA. Rincian eksperimen dapat diakses pada tautan Google Colaboratory yang terdapat di lampiran. Dataset terdiri atas dua berkas, yaitu berkas latih (Train_stock_market.csv) dan berkas pengujian (Test_stock_market.csv). Berkas latih memiliki 9.645 baris data, sedangkan berkas pengujian memiliki 39 baris data. Keduanya terdiri atas 7 kolom, yaitu Date, Low, Open, Volume, High, Close, dan Adjusted Close. Akan tetapi, terdapat beberapa perbedaan format data antara berkas data latih dan data uji, yaitu:

a. Perbedaan urutan penulisan kolom. Pada data latih, urutan kolomnya adalah Date-Low-Open-Volume-High-Close-Adjusted Close, sedangkan pada data uji, urutan kolomnya adalah Date-Open-High-Low-Close-Adj Close-Volume.
b. Perbedaan format Date. Format tanggal Date pada data latih adalah dd-mm-yyyy, sedangkan pada data uji adalah yyyy-mm-dd.
c. Perbedaan nama satu kolom. Pada data latih, terdapat kolom Adjusted Close, sedangkan kolom yang bersesuaian pada data uji diberi nama Adj Close.

Sebelum dimasukkan ke model pembelajaran mesin, perlu dilakukan praproses pada data. Praproses tidak mencakup pemecahan data menjadi data latih dan validasi karena tidak terdapat tahap pelatihan pada tugas ini. Praproses yang dilakukan terdiri atas:

1. Penghapusan kolom yang tidak diperlukan, yaitu kolom Date dan Adjusted Close, sebagaimana spesifikasi tugas. Maka, kolom-kolom yang tersisa adalah Open, High, Low, Close, dan Volume.
2. Penskalaan tiap kolom sehingga memiliki nilai di antara 0 dan 1. Penskalaan diperlukan karena kolom Volume memiliki nilai yang sampai besar (hingga puluhan ribu) sehingga akan mendominasi model. Dengan penskalaan, nilai Volume akan memiliki rentang yang sama dengan nilai kolom-kolom lain.
3. Pembuatan sekuens dengan lebar sebesar *timestep* yang ditentukan. Sekuens ini adalah bentuk data yang diperlukan sebagai masukan untuk model RNN (LSTM).

Model yang digunakan dalam eksperimen terdiri atas 2 *layer*, yaitu sebuah *layer* LSTM dengan 64 unit dan sebuah *layer* keluaran Dense dengan 5 unit. Ringkasan model dapat ditemukan pada gambar III.1, yang merupakan hasil pemanggilan summary() pada model. Dimensi data masukan adalah (timestep, jumlah atribut) dan dimensi data keluaran adalah (jumlah atribut). Jumlah atribut yang digunakan pada eksperimen ini adalah 5, yaitu Low, Open, Volume, High, dan Close. Karena tidak terdapat tahap pelatihan, bobot-bobot pada model didapatkan dari suatu berkas json dengan fungsi load_model() agar hasil prediksi tidak acak.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 64)                17920
dense (Dense)                (None, 5)                 325
=================================================================
Total params: 18245
_________________________________________________________________
```

Gambar III.1. Arsitektur model untuk eksperimen

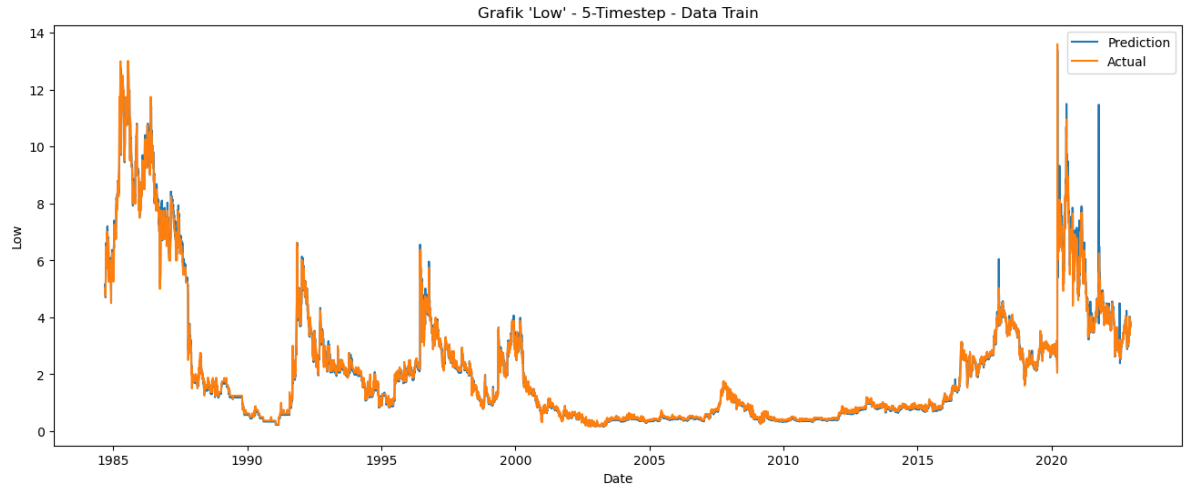
Eksperimen dilakukan dengan memvariasikan lebar *timestep*. Variasi timestep yang digunakan adalah 5, 15, dan 25 timestep. Metrik yang digunakan adalah *root mean square error* (RMSE) dengan prediksi yang baik akan memiliki RMSE yang rendah. RMSE berdasarkan nilai yang ternormalisasi, yaitu yang telah dipetakan menjadi rentang 0 sampai 1.

B. Hasil Eksperimen

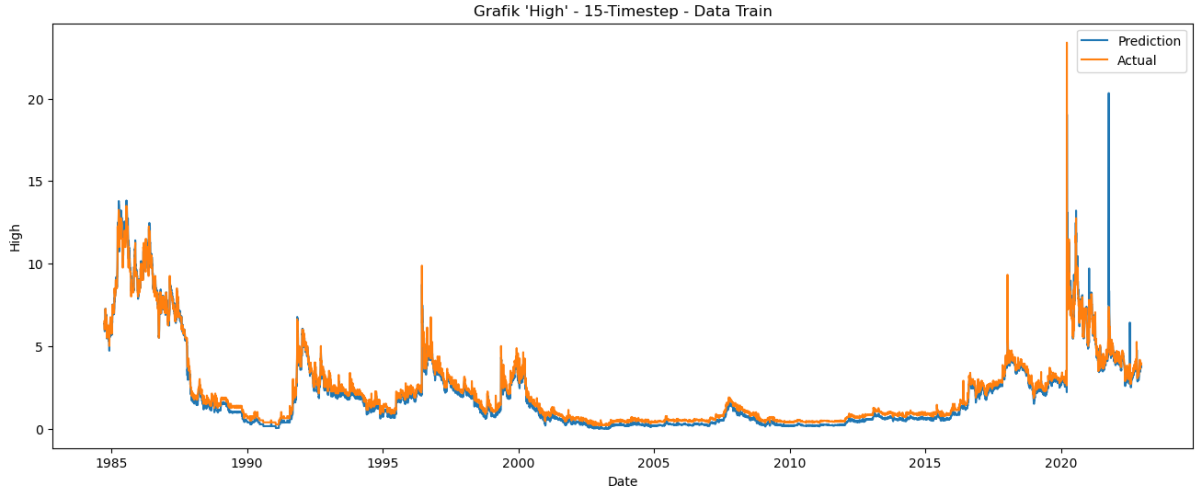
Hasil RMSE untuk data latih terdapat pada tabel III.1. Berdasarkan data tersebut, model cukup baik karena memiliki nilai RMSE yang cukup rendah. Contoh grafik perbandingan nilai prediksi dan sebenarnya pada data latih dapat dilihat pada gambar III.2 sampai dengan III.4. Lebar *timestep* terbaik berdasarkan eksperimen dan metrik RMSE adalah 5, diikuti 15 dan 25.

Tabel III.1. Nilai RMSE untuk data latih

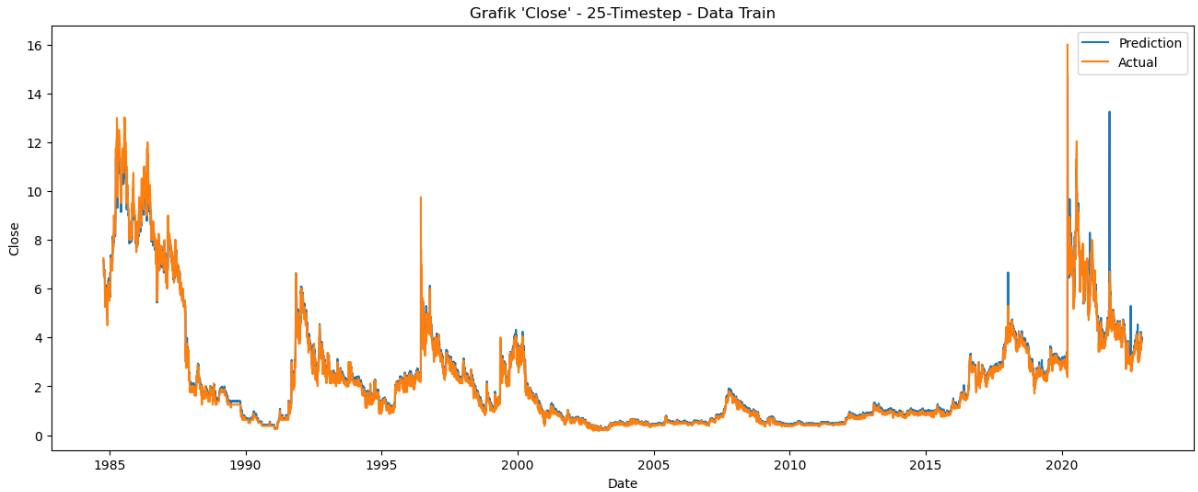
Timestep	RMSE
5	0,0129
15	0,0149
25	0,0153



Gambar III.2. Grafik prediksi vs sebenarnya untuk atribut Low, 5 timestep, dan data latih



Gambar III.3. Grafik prediksi vs sebenarnya untuk atribut High, 15 timestep, dan data latih

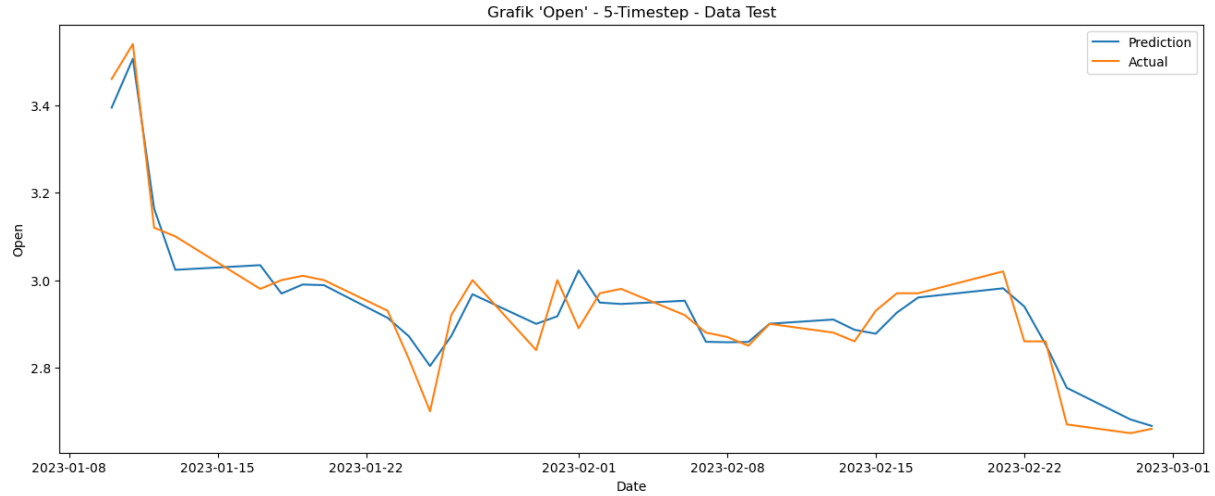


Gambar III.4. Grafik prediksi vs sebenarnya untuk atribut Close, 25 timestep, dan data latih

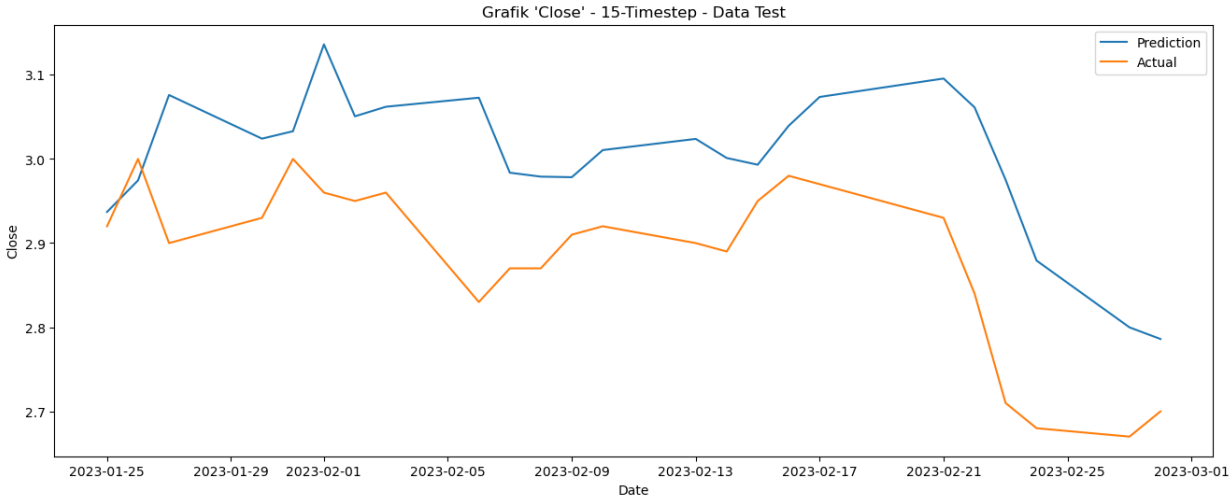
Hasil RMSE untuk data uji terdapat pada tabel III.1. Berdasarkan data tersebut, model cukup baik karena memiliki nilai RMSE yang cukup rendah. Contoh grafik perbandingan nilai prediksi dan sebenarnya pada data uji dapat dilihat pada gambar III.5 sampai dengan III.7. Lebar *timestep* terbaik berdasarkan eksperimen dan metrik RMSE adalah 5, diikuti 15 dan 25.

Tabel III.2. Nilai RMSE untuk data uji

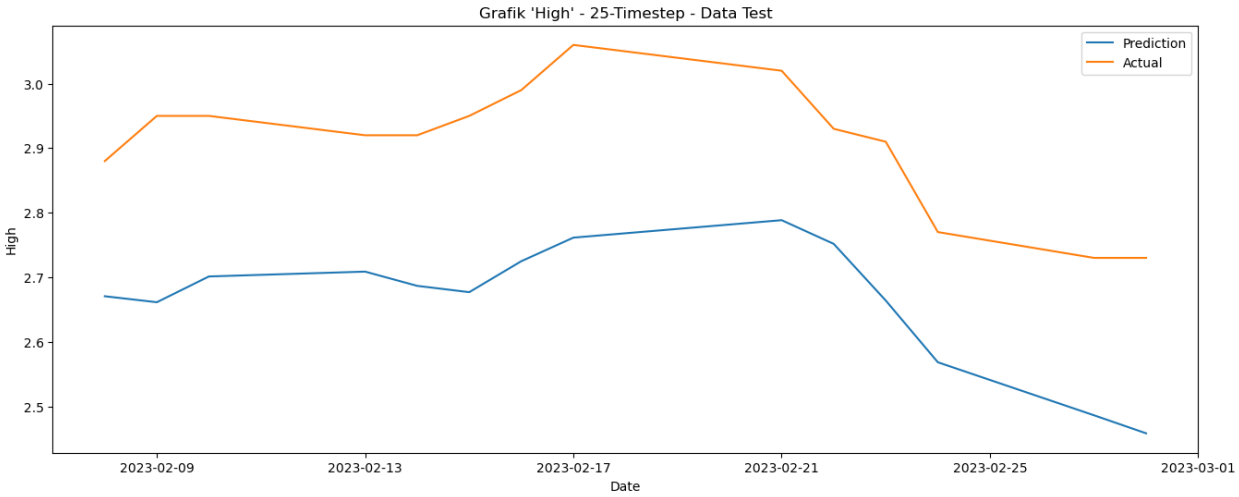
Timestep	RMSE
5	0,0044
15	0,0087
25	0,0093



Gambar III.5. Grafik prediksi vs sebenarnya untuk atribut Open, 5 timestep, dan data uji



Gambar III.6. Grafik prediksi vs sebenarnya untuk atribut Close, 15 timestep, dan data uji



Gambar III.7. Grafik prediksi vs sebenarnya untuk atribut High, 25 timestep, dan data latih

C. Kesimpulan

Berdasarkan eksperimen yang telah dilakukan, model berhasil melakukan prediksi dengan cukup baik meskipun tidak terdapat tahap pelatihan. Prediksi yang baik dapat terjadi karena model memuat bobot yang mungkin telah dilatih menggunakan implementasi yang lain. Nilai lebar timestep terbaik berdasarkan eksperimen dengan metrik RMSE adalah 5, baik untuk data latih maupun data uji, diikuti *timestep* 15 dan 25.

LAMPIRAN

A. Pembagian Tugas

NIM	Nama	Pembagian Tugas
13520116	Mahesa Lizardy	Struktur LSTM, *forward propagation* LSTM, load Model, laporan
13520146	Bryan Amirul Husna	*Summary* Model, save Model, *activation function* LSTM, eksperimen, laporan

B. Tautan

a. Tautan Google Colaboratory
https://drive.google.com/file/d/1r74KqvZFI6pJyNRYpSMQqx0OppiPvkMt/view?usp=sharing

b. Tautan Repository GitHub
https://github.com/lizardyy/Tubes_CNN

c. Tautan Video
https://youtu.be/SBRvwX6P0Zw