PREDIKSI VALUTA ASING MENGGUNAKAN LSTM YANG DIOPTIMALKAN DENGAN ALGORITMA GENETIK

SKRIPSI

Diajukan untuk memenuhi salah satu syarat
memperoleh gelar Sarjana Komputer
Program Studi Informatika

Diajukan oleh:
Daniel Budi Prasetyo
205314145

FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS SANATA DHARMA
YOGYAKARTA
2023

HALAMAN PERSETUJUAN PEMBIMBING

SKRIPSI

PREDIKSI VALUTA ASING MENGGUNAKAN LSTM YANG DIOPTIMALKAN DENGAN ALGORITMA GENETIK

Disusun oleh:

Daniel Budi Prasetyo

205314145

Dosen Pembimbing,

(Cyprianus Kuntoro Adi, S.J. M.A., M.Sc., Ph.D.) 16 Desember 2023

HALAMAN PENGESAHAN

SKRIPSI

PREDIKSI VALUTA ASING MENGGUNAKAN LSTM YANG DIOPTIMALKAN DENGAN ALGORITMA GENETIK

Dipersiapkan dan disusun oleh:

Daniel Budi Prasetyo

205314145

Telah dipertahankan di depan Dewan Penguji

Pada tanggal, ...

Dan dinyatakan memenuhi syarat

Susunan Panitia Penguji

Jabatan		Nama Lengkap	Tanda Tangan
Ketua	:		.....
Sekretaris	:		.....
Anggota	:		.....

Yogyakarta, ...

Fakultas Sains dan Teknologi

Universitas Sanata Dharma

Dekan,

Ir. Drs. Haris Sriwindono, M.Kom, Ph.D.

PERNYATAAN KEASLIAN KARYA

Saya menyatakan dengan sesungguhnya bahwa skripsi yang saya tulis ini tidak memuat karya atau bagian karya orang lain, kecuali yang telah disebutkan dalam kutipan dan daftar pustaka dengan mengikuti ketentuan sebagaimana layaknya karya ilmiah.

Apabila di kemudian hari ditemukan indikasi plagiarisme dalam naskah ini, saya bersedia menanggung segala sanksi sesuai peraturan perundang-undangan yang berlaku.

Yogyakarta, ...

Penulis,

Daniel Budi Prasetyo

LEMBAR PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH UNTUK KEPERLUAN AKADEMIS

Yang bertanda tangan di bawah ini, saya mahasiswa Universitas Sanata Dharma:

Nama : Daniel Budi Prasetyo

NIM : 205314145

Demi perkembangan ilmu pengetahuan, saya memberikan kepada Perpustakaan Universitas Sanata Dharma karya ilmiah saya yang berjudul:

PREDIKSI VALUTA ASING MENGGUNAKAN LSTM YANG DIOPTIMALKAN DENGAN ALGORITMA GENETIK

beserta perangkat yang diperlukan (bila ada). Dengan demikian saya memberikan hak kepada Perpustakaan Universitas Sanata Dharma baik untuk menyimpan, mengalihkan dalam bentuk media lain, mengolah dalam bentuk pangkalan data, mendistribusikan secara terbatas, dan mempublikasikannya di internet atau media lain untuk kepentingan akademis tanpa perlu meminta izin dari saya atau memberikan royalti kepada saya selama tetap mencantumkan nama saya sebagai penulis.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di Yogyakarta
Pada tanggal: .....
Yang menyatakan,

Daniel Budi Prasetyo

KATA PENGANTAR

Segala puja dan puji syukur kepada kehadirat Tuhan Yang Maha Esa, yang telah melimpahkan rahmat dan hidayah-Nya kepada kita semua, sehingga penulis dapat menyelesaikan tugas akhir dengan judul "Prediksi Valuta Asing Menggunakan LSTM yang Dioptimalkan dengan Algoritma Genetik" ini dengan sebaik – baiknya.

Penulis menyadari bahwa tugas akhir ini tidak dapat diselesaikan dengan baik tanpa adanya bantuan, dukungan, dan bimbingan dari berbagai pihak. Oleh karena itu, dalam kesempatan ini, penulis mengucapkan rasa terima kasih yang sebesar-besarnya kepada semua yang telah berperan serta dalam perjalanan penyusunan tugas akhir ini, di antaranya:

1. Romo Cyprianus Kuntoro Adi, S.J. M.A., M.Sc., Ph.D., selaku dosen pembimbing tugas akhir.
2. Ibu Ir. Agnes Maria Polina, S.Kom., M.Sc., selaku dosen pembimbing akademik.
3. Kedua orang tua dan kakak tercinta, yang selalu memberikan semangat dan doa dalam mengerjakan tugas akhir.
4. Seluruh dosen prodi Informatika Universitas Sanata Dharma, yang telah membimbing dan memberikan banyak ilmu pengetahuan kepada penulis.
5. Teman – teman prodi Informatika Universitas Sanata Dharma Angkatan 2020 yang selalu memberikan dukungan dan semangat dalam menyelesaikan tugas akhir.

Akhir kata, penulis berharap tugas akhir ini dapat bermanfaat bagi pembaca untuk menambah wawasan tentang LSTM dan Algoritma Genetik. Saya juga mengucapkan terima kasih kepada semua pihak yang tidak dapat saya sebutkan, semoga Tuhan Yang Maha Esa membalas semua kebaikan kalian semua.

Yogyakarta, ...

Daniel Budi Prasetyo

ABSTRAK

Foreign exchange (Forex) adalah salah satu pasar keuangan terbesar di dunia, dengan lebih dari $5,1 triliun diperdagangkan setiap hari. Pada penelitian ini, *Long Short-Term Memory* (LSTM) dan *Genetic Algorithm Long Short-Term Memory* (GA-LSTM) digunakan untuk memprediksi bagaimana pola harga dari USD, EUR, dan SGD. Data diambil dari *website* Google Finance dalam kurun waktu 5 tahun dengan total data sekitar 1977 data untuk USD dan EUR, dan 1956 data untuk SGD.

Penelitian ini melakukan normalisasi data forex menggunakan metode *min-max*. Hasil normalisasi akan di-*reshape* sesuai dengan *sliding window* dan data akan dibagi menjadi menggunakan 2 metode yaitu *split* dan *cross validation* untuk melatih dan menguji model. Setelah mendapatkan nilai evaluasi model, model dengan nilai *error* paling minimal pada kelompok *sliding window* akan dioptimasikan menggunakan Algoritma Genetik.

Dalam beberapa skenario, optimasi dengan Algoritma Genetik berhasil mengurangi nilai *error*, meskipun tidak selalu berlaku untuk semua kasus. Model LSTM telah terbukti mampu memprediksi pergerakan harga beli mata uang asing seperti USD, EUR, dan SGD terhadap IDR dengan tingkat *error* yang cukup memuaskan dalam jangka pendek.

Kata kunci: *Foreign exchange*, *Long Short-Term Memory*, Algoritma Genetik

ABSTRACT

Foreign exchange (Forex) is one of the largest financial markets in the world, with more than $5.1 trillion traded every day. In this study, Long Short-Term Memory (LSTM) and Genetic Algorithm Long Short-Term Memory (GA-LSTM) are used to predict the price patterns of USD, EUR, and SGD. The data is taken from the Google Finance website over a period of 5 years with a total of about 1977 data for USD and EUR, and 1956 data for SGD.

This study normalizes forex data using the min-max method. The normalization results will be reshaped according to the sliding window and the data will be divided using 2 methods, namely split and cross validation to train and test the model. After getting the model evaluation value, the model with the smallest error value in the sliding window group will be optimized using the Genetic Algorithm.

In several scenarios, optimization with the Genetic Algorithm managed to reduce the error value, although it does not always apply to all cases. The LSTM model has proven to be able to predict the movement of foreign currency purchase prices such as USD, EUR, and SGD against IDR with a satisfactory error rate in the short term.

Keywords: *Foreign exchange*, *Long Short-Term Memory*, Genetic Algorithm

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I

PENDAHULUAN

1.1. Latar Belakang

Foreign exchange (Forex) adalah salah satu pasar keuangan terbesar di dunia, dengan lebih dari $5,1 triliun diperdagangkan setiap hari. Karena kompleksitas dan volatilitasnya, prediksi harga menjadi sulit [1]. Terutama, di negara berkembang seperti Indonesia, yang sangat penting untuk mendukung pembangunan ekonomi yang berkelanjutan dan meningkatkan kesejahteraan rakyat. Ketidakstabilan nilai tukar dapat menyurutkan minat investor untuk berinvestasi, yang dapat menyebabkan kemunduran dalam pembangunan di Indonesia. Sebab, selama ini peran investor asing sangat besar dalam pertumbuhan ekonomi [2].

Deep Learning telah mencapai kesuksesan besar di bidang *image recognition*, *natural language processing*, *speech recognition*, *video processing*, dan lain – lain. Oleh karena itu, penerapan algoritma *Deep Learning* dalam prediksi nilai tukar juga mendapat perhatian luas [3, 4, 5]. Peneliti keuangan di seluruh dunia telah mempelajari dan menganalisis perubahan di pasar saham dan Forex. Penerapan kecerdasan buatan yang meluas telah menyebabkan peningkatan jumlah investor yang menggunakan model *Deep Learning* untuk memprediksi dan mempelajari harga saham dan Forex. Telah terbukti bahwa fluktuasi harga saham dan Forex dapat diprediksi [4].

Berdasarkan salah satu literatur yang peneliti baca, model LSTM lebih baik dibandingkan dengan model RNN. Dimana model LSTM memiliki Root Mean Square Error (RMSE) dan Mean Absolute Error (MAE) yang lebih kecil dibandingkan dengan model RNN [6]. Dengan literatur di atas sebagai dasar, peneliti ingin mengambil model LSTM tersebut sebagai bahan penelitian untuk memprediksi harga valuta asing dalam 5 tahun terakhir. Selain itu, peneliti juga akan menggunakan Algoritma Genetik untuk mengoptimasi model LSTM, yang diharapkan akan menurunkan *error* atau kesalahan dari model awal.

1.2. Rumusan Masalah

1. Bagaimana tingkat evaluasi matriks menggunakan LSTM untuk harga mata uang asing USD, EUR, dan SGD?
2. Apakah dengan optimasi parameter menggunakan Algoritma Genetik dapat menurunkan *error* pada sebuah model?

1.3. Batasan Masalah

1. Data yang digunakan adalah nilai tukar untuk USD/IDR, EUR/IDR, dan SGD/IDR dengan rentang waktu 5 tahun terakhir.
2. Arsitektur model *Deep Learning* yang digunakan adalah LSTM.
3. Algoritma optimasi yang digunakan adalah Algoritma Genetik.

1.4. Tujuan Penelitian

1. Untuk mengetahui perbandingan antara model awal dengan model yang telah dioptimasikan.
2. Untuk mengetahui apakah Algoritma Genetik berpengaruh terhadap penurunan *error* dari sebuah model.

1.5. Manfaat Penelitian

1. Meningkatkan pemahaman tentang prediksi nilai tukar mata uang asing.
2. Memperluas pengetahuan dalam bidang kecerdasan buatan dan keuangan.
3. Membantu pengambilan keputusan yang lebih baik di pasar forex.

BAB II

TINJAUAN PUSTAKA

2.1. Tinjauan Pustaka

Tabel 2.1 Review Literatur

Peneliti	Judul	Model	Hasil
Lina Ni, Yujie Li, Xiao Wang, Jinquan Zhang, Jiguo Yu, Chengming Qi	Forecasting of Forex Time Series Data Based on Deep Learning (2019)	C-RNN, LSTM, CNN	Hasil menggunakan algoritma C-RNN mendapatkan error yang lebih rendah dibandingkan dengan CNN dan LSTM, yaitu mulai dari 510 – 530.
M.S. Islam, E. Hossain	Foreign exchange currency rate prediction using a GRU-LSTM hybrid network (2021)	GRU-LSTM, LSTM, GRU, SMA	Hasil menggunakan algoritma GRU-LSTM mendapatkan error yang lebih rendah
Gunho Jung, Sun-Yong Choi	Forecasting Foreign Exchange Volatility Using Deep Learning (2021)	LSTM, Autoencoder-LSTM	Untuk memprediksi valuta asing algoritma Autoencoder-LSTM mendapatkan error yang lebih rendah dibandingkan dengan LSTM
Aghistina Kartikadewi, Lina Audina Abdul Rosyid, Anggraeni Eka Putri	Prediction of Foreign Currency Exchange (IDR and USD) Using Multiple Linear Regression (2020)	Multiple Linear Regression	Dengan menggunakan model yang diajukan peneliti mendapatkan hasil kurang lebih 165,38% pada MSE, 24,04% pada MAPE, dan 25,7% pada margin error

Muhammad Yasir, Mehr Yahya Durrani, Sitara Afzal, Muazzam Maqsood, Farhan Aadil, Irfan Mehmood, Seungmin Rho	An Intelligent Event-Sentiment-Based Daily Foreign Exchange Rate Forecasting System (2019)	Linear Regression, SVR, Deep Learning	Hasilnya menunjukkan bahwa metode berbasis deep learning memiliki kinerja yang lebih baik daripada metode lainnya. Selain itu, hasil prediksi membaik ketika sentimen dipertimbangkan dalam model, oleh karena itu Hong Kong, Pakistan, dan Inggris dikatakan lebih terpapar pada peristiwa besar yang terjadi lintas batas.
Mei-Li Shen, Cheng-Feng Lee, Hsiou-Hsiang Liu, Po-Yin Chang, Cheng-Hong Yang	An Effective Hybrid Approach for Forecasting Currency Exchange Rates (2021)	FSPSOSVR, PSOSVR, SVR, ANN, SARIMA, ARIMA, EST, RW	Secara khusus, di bawah skema FSPSOSVR, MAPE-nya adalah 2,296%, mengungguli 3,477%, 4,628%, 3,603%, 4,657%, 4,333%, 6,018%, dan 4,089% dari skema milik PSOSVR, SVR, ANN, SARIMA, ARIMA, EST, dan RW
Manav Kaushik, A K Giri	Forecasting Foreign Exchange Rate: A Multivariate Comparative Analysis between Traditional Econometric, Contemporary Machine Learning & Deep Learning Techniques	VAR, SVM, LSTM	Hasilnya dengan jelas menggambarkan bahwa teknik kontemporer SVM dan RNN (Long Short-Term Memory) mengungguli metode tradisional Auto Regression yang banyak digunakan. Model RNN dengan Long Short-Term Memory (LSTM) memberikan akurasi maksimum (97,83%) diikuti oleh Model SVM (97,17%) dan Model VAR (96,31%).
Yaxin Qu, Xue Zhao	Application of LSTM Neural Network in Forecasting Foreign Exchange Price (2019)	LSTM, RNN	Hasil percobaan menunjukkan bahwa model jaringan saraf LSTM memiliki root mean square error (RMSE) dan mean absolute error (MAE) yang lebih kecil daripada model jaringan RNN, dan harga prediksi lebih akurat.
Ruofan Liao, Petchaluck Boonyakunakorn, Napat Harnpornchai,	Forecasting the Exchange Rate for USD to RMB using RNN and SVM (2020)	RNN, LM, SCG, BR, SVM, ARIMA	Hasilnya menunjukkan bahwa MSE terendah dimiliki oleh model RNN dibandingkan dengan LM, SCG, BR, SVM, ARIMA.

Songsak Sriboonchitta			
Kwok Tai Chui, Brij B. Gupta, Pandian Vasant	A Genetic Algorithm Optimized RNN-LSTM Model for Remaining Useful Life Prediction of Turbofan Engine (2021)	RNN, LSTM, NSGA-II optimized RNN-LSTM	Weight untuk RNN-LSTM yang dirancang oleh Non-Dominated Sorting Genetic Algorithm II (NSGA-II) dapat mencapai RMSE rata-rata 17,2. Ini meningkatkan RMSE sebesar 6,07–14,72% dibandingkan dengan model dasar RNN dan LSTM.
Azar Niknam, Hasan Khademi Zare, Hassan Hosseininasab, Ali Mostafaeipour	Developing an LSTM model to forecast the monthly water consumption according to the effects of the climatic factors in Yazd, Iran (2023)	UV-LSTM, MV-LSTM	Ditemukan bahwa kesalahan forecasting error MV-LSTM seringkali lebih kecil daripada model UV-LSTM. Ini berarti model MV-LSTM mengungguli UV-LSTM. Sedangkan, jika model memperhitungkan faktor iklim, akurasi peramalannya akan meningkat.
Burak Gülmez	Stock price prediction with optimized deep LSTM network with artificial rabbits optimization algorithm (2023)	LSTM-ARO, LSTM-GA, ANN, LSTM1D, LSTM2D, LSTM3D	Ketika LSTM-ARO dibandingkan dengan model artificial neural network (ANN), tiga model LSTM yang berbeda, dan LSTM yang dioptimalkan oleh Genetic Algorithm (GA). Hasilnya menunjukkan bahwa LSTM-ARO mengungguli model lain berdasarkan kriteria evaluasi MSE, MAE, MAPE, dan R2.

2.2. Landasan Teori

2.2.1. Valuta Asing

Nilai tukar mata uang nasional merupakan harga relatif terhadap mata uang nasional lainnya, dan seperti harga pada umumnya, nilai tukar dapat mengalami kenaikan atau penurunan [7]. Ketika nilai tukar suatu mata uang, misalnya dolar, meningkat terhadap mata uang lain, seperti rupiah, hal ini menunjukkan bahwa satu unit mata uang tersebut dapat membeli lebih banyak mata uang lainnya. Dalam konteks ini, kita mengatakan bahwa mata uang tersebut menguat terhadap mata uang lainnya. Sebaliknya, ketika nilai tukar mata uang menurun terhadap mata uang lain, hal ini menunjukkan bahwa satu unit mata uang tersebut hanya dapat membeli jumlah mata uang lain yang lebih sedikit. Dalam hal ini, mata uang tersebut dianggap melemah terhadap mata uang lainnya.

2.2.2. Preprocessing

Data preprocessing atau *data preparation* adalah proses mengubah data mentah menjadi bentuk yang lebih sesuai untuk pemodelan [8]. Tahap ini sering dianggap sebagai aspek yang paling krusial, memakan waktu, dan sering terlupakan dalam sebuah proyek pembelajaran mesin yang berfokus pada pemodelan prediktif. Meskipun prinsip dasar *data preparation* relatif sederhana, terdapat beragam teknik lanjutan yang masing-masing terdiri dari algoritma yang berbeda. Teknik-teknik ini secara khusus dirancang untuk mengatasi berbagai situasi, dan masing-masing memiliki sekumpulan *hyperparameter*, tips, dan trik mereka sendiri untuk mencapai hasil optimal.

2.2.2.1. Normalisasi

Normalisasi adalah proses mengubah rentang nilai aktual yang dapat diambil oleh fitur numerik menjadi rentang nilai standar yang biasanya dalam interval [-1, 1] atau [0, 1] [9]. Terdapat beberapa metode normalisasi yang umum digunakan, salah satunya adalah normalisasi *min-max* yang biasanya mengubah data dalam interval [0, 1]. Dimana persamaan normalisasi tersebut adalah sebagai berikut:

$$\bar{\boldsymbol{x}}^{(j)} = \frac{\boldsymbol{x}^{(j)} - \boldsymbol{min}^{(j)}}{\boldsymbol{max}^{(j)} - \boldsymbol{min}^{(j)}} \tag{2.1}$$

Keterangan:

$\bar{x}^{(j)}$ = Nilai hasil normalisasi

$x^{(j)}$ = Nilai fitur j

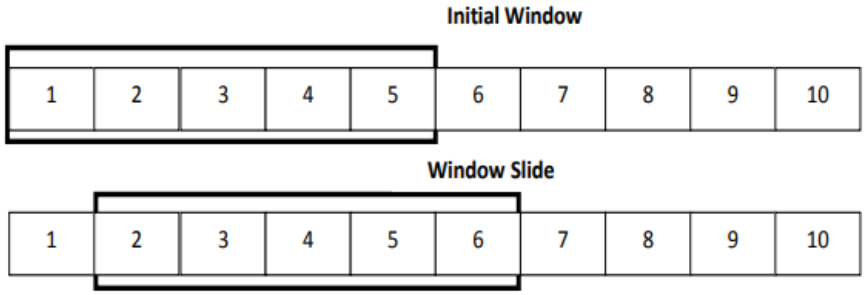
$min^{(j)}$ = Nilai minimal dari fitur j

$max^{(j)}$ = Nilai maksimal dari fitur j

2.2.2.2. Sliding Window

Sliding window merupakan salah satu metode yang dapat digunakan pada tahap *preprocessing* untuk merestrukturisasi data menurut kerangka waktu menjadi masalah klasifikasi [10]. Jumlah unit yang ditentukan dalam jendela disebut ukuran jendela. Setelah memilih segmen pertama, segmen berikutnya dipilih dari ujung segmen pertama.

Proses ini diulang sampai semua data deret waktu tersegmentasi. Proses *sliding window* ditunjukkan pada Gambar 2.1 dengan ukuran jendela 5.



Gambar 2.1 Proses Sliding Window (H.S. Hota dkk., 2017)

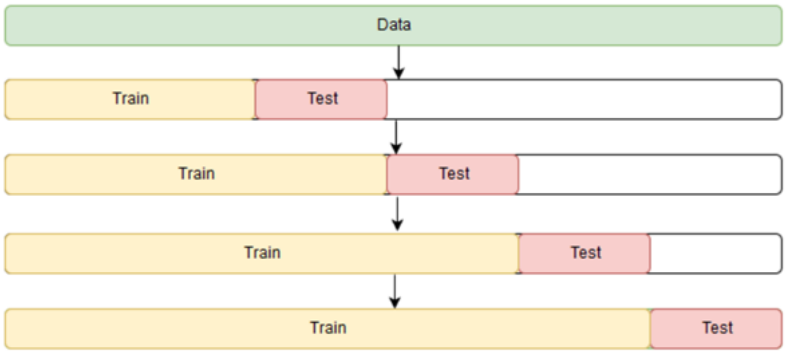
2.2.2.3. Split Data

Setelah memperoleh kumpulan data, langkah pertama yang dilakukan adalah melakukan pengacakan (*shuffle*) dan membagi data menjadi dua atau tiga bagian tergantung pada kebutuhan. Dalam era *Big Data* saat ini, umumnya data dibagi menjadi tiga bagian, yaitu: *training*, *validation*, dan *test*. Bagian *training* biasanya memiliki ukuran yang paling besar dan digunakan untuk melatih model. Sementara itu, bagian *validation* dan *test* memiliki ukuran yang relatif serupa dan jauh lebih kecil dibandingkan data *training*. Dimana *validation* digunakan untuk menyesuaikan *hyperparameter* model, dan *test* digunakan untuk mengevaluasi kinerja model pada data yang belum pernah dilihat sebelumnya [9].

2.2.2.4. Cross Validation

Cross Validation (CV) adalah teknik yang banyak digunakan untuk memilih model atau algoritma terbaik. Konsep intinya

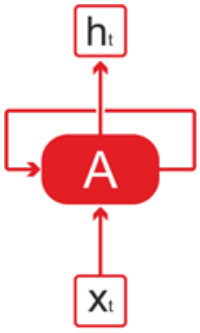
melibatkan pembagian data menjadi beberapa *subset* untuk menilai kinerja setiap algoritma [11]. Dalam proses ini, sebagian data digunakan untuk melatih setiap algoritme, sedangkan sisanya disisihkan untuk mengevaluasi seberapa baik kinerja algoritme.



Gambar 2.2 Time Series Cross-Validation (S. Shrivastava, 2020)

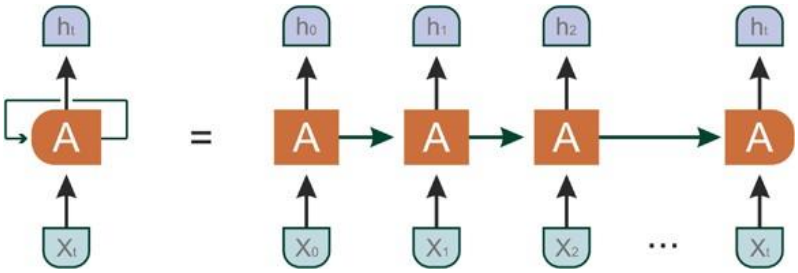
2.2.3. Recurrent Neural Network (RNN)

Recurrent Neural Networks (RNN) adalah salah satu jenis jaringan saraf yang dirancang untuk memproses *sequential data* dengan memperkenalkan *loop* yang memungkinkan informasi bertahan di dalam jaringan. Tidak seperti jaringan saraf tradisional, yang hanya mempertimbangkan *input* saat ini, RNN dapat memanfaatkan informasi masa lalu untuk membuat prediksi atau mengklasifikasikan *input* saat ini [12].



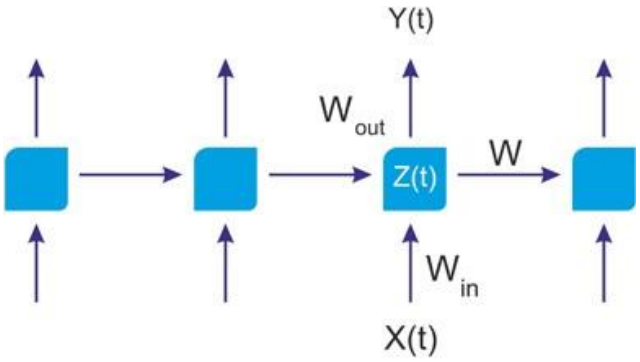
Gambar 2.3 RNN memiliki loop (G. Zaccone dkk., 2018)

Struktur dasar RNN terdiri dari modul berulang yang meneruskan pesan ke penggantinya. Saat dibuka, modul ini membuat struktur seperti rantai yang merepresentasikan aliran informasi sepanjang waktu. Setiap modul mengambil *input* pada langkah waktu tertentu dan menghasilkan *output*, sekaligus mempertahankan keadaan internal atau memori yang menangkap informasi tentang *input* sebelumnya.



Gambar 2.4 Representasi langkah dari RNN (G. Zaccone dkk., 2018)

Untuk mentransfer informasi antar langkah waktu, RNN menggunakan bobot transisi (W). Bobot ini memungkinkan jaringan untuk memperbarui status internalnya berdasarkan masukan saat ini dan status sebelumnya. Dengan demikian, RNN dapat menangkap dependensi dan pola dalam data berurutan.



Gambar 2.5 RNN menggunakan keadaan jaringan sebelumnya (G. Zaccone dkk., 2018)

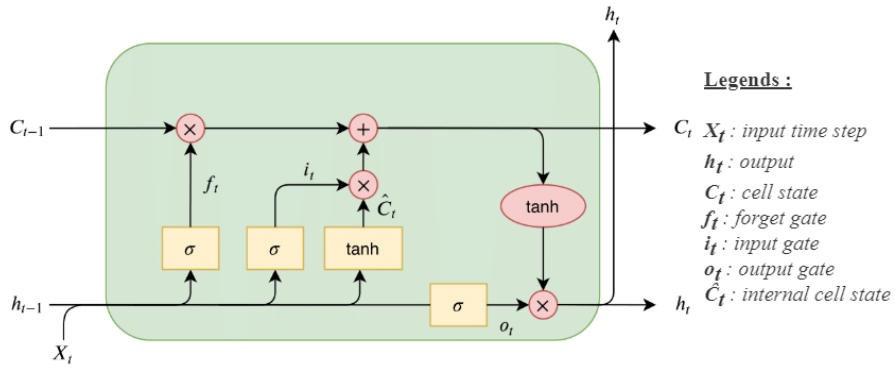
Namun, RNN klasik mengalami keterbatasan tertentu. Salah satu masalah utama adalah masalah gradien yang hilang, yang membuatnya sulit untuk menangkap ketergantungan jangka panjang. Selain itu, mereka kesulitan mempertahankan dan memanfaatkan informasi yang relevan dalam urutan yang panjang. Untuk mengatasi kelemahan ini, variasi RNN yang lebih baik yang disebut Long Short-Term Memory (LSTM) diperkenalkan.

2.2.4. Long Short Term Memory (LSTM)

Long Short Term Memory adalah jenis RNN khusus, yang mampu mempelajari dependensi jangka panjang. Layer tersebut diperkenalkan oleh Hochreiter & Schmidhuber pada tahun 1997 [12], yang bekerja sangat baik pada berbagai macam masalah dan sekarang digunakan secara luas terutama dalam tugas yang melibatkan prediksi dan klasifikasi *sequential data*.

Jaringan LSTM terdiri dari sel atau blok yang saling berhubungan. Setiap blok berisi tiga jenis gerbang: *input*, *output*, dan *forget gate*.

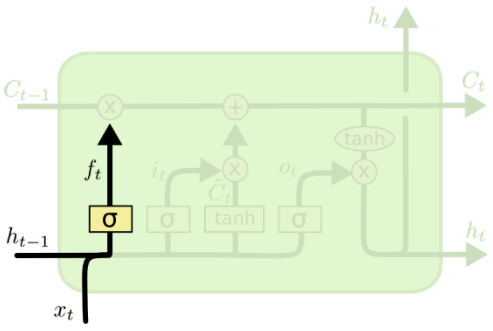
Gerbang ini mengontrol fungsi penulisan, pembacaan, dan pengaturan ulang pada sel memori.



Gambar 2.6 Arsitektur LSTM (Thorir, 2021)

2.2.4.1. Forget Gate

Forget gate menentukan berapa banyak data sebelumnya yang akan dilupakan dan berapa banyak data sebelumnya yang akan digunakan di langkah berikutnya. Hasil dari gerbang ini berada pada *range* 0-1. Nilai 0 melupakan data sebelumnya, 1 menggunakan data sebelumnya. *Forget gate* layer dapat dimodelkan seperti pada gambar 2.6. Dihitung dengan persamaan nomor 2.2.



Gambar 2.7 Forget Gate (Colah, 2015)

Persamaan Forget Gate

$$\boldsymbol{f_t = \sigma\left(W_f . [h_{t-1}, x_t] + b_f\right)} \quad (2.2)$$

Keterangan:

f_t = Forget gate

σ = Fungsi aktivasi sigmoid

W_f = Nilai weight forget gate

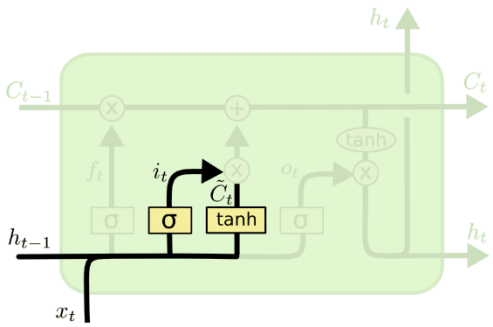
h_{t-1} = Nilai output sebelumnya

x_t = Nilai input saat ini

b_f = Nilai bias forget gate

2.2.4.2. Input Gate

Layer kedua adalah *input gate* yang terdiri dari *input gate* dan *tanh layer*. Data baru diperoleh pada lapisan ini. Bagian data *input* yang tidak diperlukan disaring dengan fungsi *sigmoid* dan kemudian data baru yang mungkin ditentukan dengan fungsi *tanh*. Perkalian hasil fungsi *sigmoid* dan hasil lapisan *tanh* ditambahkan ke keadaan sel untuk memperbaharui keadaan sel dan diperoleh keadaan sel yang baru. *Input gate* dapat dimodelkan seperti pada gambar 2.7 dan 2.8. Dihitung dengan persamaan 2.3, 2.4, dan 2.5.



Gambar 2.8 Input Gate (Colah, 2015)

Persamaan Input Gate

$$\boldsymbol{i_t = \sigma(W_i . [h_{t-1}, x_t] + b_i)} \quad (2.3)$$

Keterangan:

i_t = Input gate

σ = Fungsi aktivasi sigmoid

W_i = Nilai weight input gate

h_{t-1} = Nilai output sebelumnya

x_t = Nilai input saat ini

b_i = Nilai bias input gate

Persamaan Cell State baru

$$\boldsymbol{\hat{C}_t = tanh(W_C . [h_{t-1}, x_t] + b_C)} \quad (2.4)$$

Keterangan:

$\hat{C}_t$ = Cell state baru

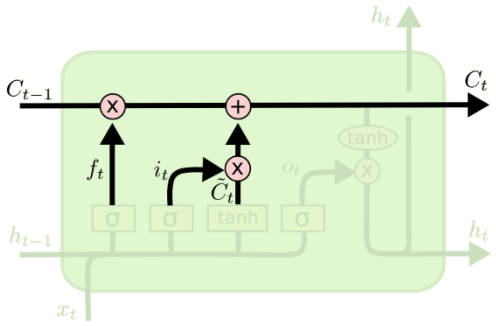
$tanh$ = Fungsi tanh

W_C = Nilai weight cell state

h_{t-1} = Nilai output sebelumnya

x_t = Nilai input saat ini

b_C = Nilai bias cell state



Gambar 2.9 Cell State (Colah, 2015)

Persamaan Memperbaharui Cell State

$$\boldsymbol{C_t = i_t . \widehat{C}_t + f_t . C_{t-1}} \tag{2.5}$$

Keterangan:

C_t = Cell state

i_t = Input gate

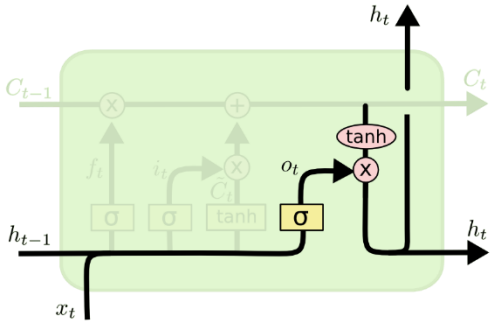
$\hat{C}_t$ = Cell state baru

f_t = Forget gate

C_{t-1} = Cell state sebelumnya

2.2.4.3. Output Gate

Pada *output gate*, status sel difilter dengan menggunakan fungsi *tanh* dan data masukan difilter dengan fungsi *sigmoid*. Perkalian hasil fungsi *sigmoid* dengan hasil *tanh* layer menjadi data keluaran. *Output gate* dapat dimodelkan seperti pada gambar 2.9. Dihitung dengan persamaan 2.6 dan 2.7.



Gambar 2.10 Output Gate (Colah, 2015)

Persamaan Output Gate

$$\boldsymbol{o_t = \sigma(W_o . [h_{t-1}, x_t] + b_o)} \tag{2.6}$$

Keterangan:

o_t = Output gate

σ = Fungsi aktivasi sigmoid

W_o = Nilai weight output gate

h_{t-1} = Nilai output sebelumnya

x_t = Nilai input saat ini

b_o = Nilai bias output gate

Persamaan Nilai Output

$$\boldsymbol{h_t = o_t \times tanh(C_t)} \tag{2.7}$$

Keterangan:

h_t = Nilai output

o_t = Output gate

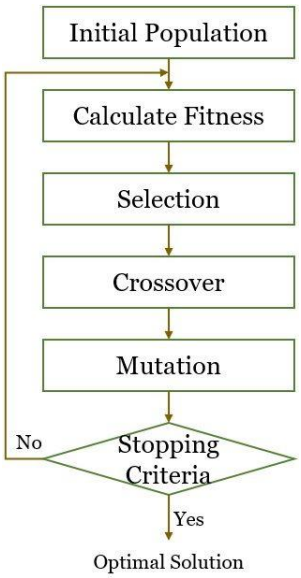
$tanh$ = Fungsi tanh

C_t = Cell state

2.2.5. Algoritma Genetik

Genetic Algorithm (GA) adalah pendekatan pencarian heuristik yang banyak digunakan untuk masalah optimasi. Mereka fleksibel dan dapat diterapkan pada berbagai skenario pengoptimalan, menjadikannya menarik dalam aplikasi praktis. GA didasarkan pada konsep evolusi, menarik inspirasi dari keberhasilan dan keragaman spesies di alam [13].

Kemampuan beradaptasi spesies terhadap lingkungannya dan perkembangan struktur kompleks telah menjadi faktor kunci dalam kelangsungan hidup mereka. Prinsip-prinsip perkawinan dan menghasilkan keturunan merupakan dasar bagi keberhasilan evolusi. Dengan mengadaptasi prinsip-prinsip ini, GA bertujuan untuk memecahkan masalah pengoptimalan dengan meniru proses evolusi.



Gambar 2.11 Langkah Algoritma Genetik (Neha, 2022)

2.2.5.1. Fitness

Dalam GA, *fitness* merujuk pada ukuran kualitas suatu solusi. *Fitness function* digunakan untuk mengevaluasi setiap solusi kandidat berdasarkan kemampuannya dalam memecahkan masalah optimasi. Desain *fitness function* merupakan bagian penting dari proses pemodelan pendekatan optimisasi, karena dapat membimbing pencarian. Sebagai contoh, dalam kasus masalah optimisasi yang terbatas, fungsi hukuman dapat digunakan untuk menurunkan *fitness* solusi yang tidak memenuhi syarat.

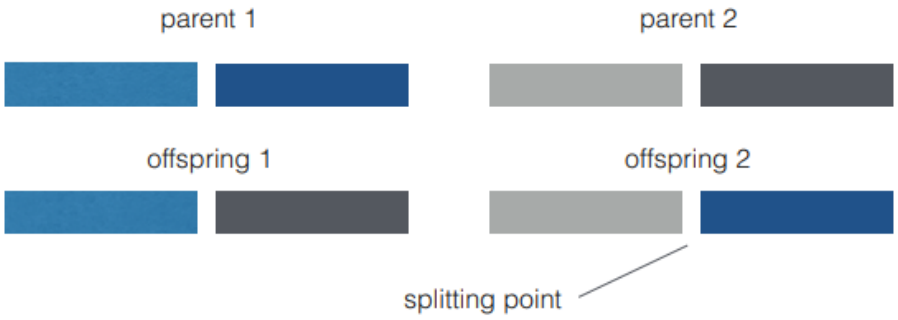
2.2.5.2. Seleksi

Seleksi adalah operator genetika dalam GA yang memilih solusi-solusi mana yang akan bertahan dan menjadi induk pada generasi baru. Proses seleksi didasarkan pada nilai kebugaran solusi-solusi dalam populasi, di mana solusi-solusi yang lebih baik memiliki peluang yang lebih tinggi untuk dipilih. Terdapat berbagai algoritma seleksi, salah satunya adalah *tournament selection*, di mana sekelompok solusi dipilih secara acak dan solusi-solusi terbaik dalam *subset* dipilih. Seleksi juga dapat digunakan menentukan induk – induk mana yang akan mengikuti proses *crossover*.

Gambar 2.12 Tournament Selection (A. Y. Ayoub dkk., 2020)

2.2.5.3. Crossover

Crossover adalah operator yang memungkinkan kombinasi materi genetik dari dua atau lebih solusi. Ini adalah operator genetik penting dalam GA, yang merupakan optimasi heuristik yang diilhami secara biologis. Operator *crossover* dapat dirancang untuk berbagai jenis representasi solusi, seperti *bit strings*, *continuous vectors*, dan permutasi simbol. Salah satu contoh operator *crossover* untuk representasi *bit strings* adalah *crossover n-point*, yang membagi dua solusi pada posisi n dan secara bergantian menyusunnya menjadi solusi baru.

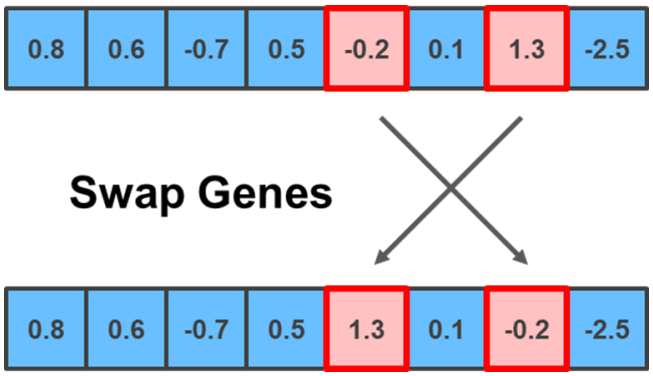


Gambar 2.13 Single-point Crossover (Kramer, 2017)

2.2.5.4. Mutasi

Mutasi adalah operator genetika penting lainnya dalam GA, yang mengubah sebuah solusi dengan memperkenalkan gangguan acak. Intensitas gangguan ini dikendalikan oleh tingkat mutasi. Operator mutasi harus memenuhi tiga persyaratan utama: keterjangkauan, ketidakberpihakan, dan skalabilitas. Berbagai operator mutasi dapat dirancang untuk berbagai jenis representasi solusi. Salah satunya adalah *swap mutation*, di mana setiap data akan ditukar dengan probabilitas

tertentu. Tingkat mutasi digunakan untuk mengatur intensitas dari *noise* yang ditambahkan.



Gambar 2.14 Swap Mutation (Silva, 2018)

2.2.6. Evaluasi Model

Evaluasi model adalah proses menggunakan matriks evaluasi yang berbeda untuk memahami kinerja model pembelajaran mesin, serta kekuatan dan kelemahannya. Evaluasi model penting untuk menilai kemanjuran model selama fase penelitian awal dan juga berperan dalam pemantauan model. Dalam pembuatan *regression* model, matriks evaluasi yang digunakan adalah matriks yang dapat menghitung *error*, antara lain MAE, MSE, RMSE, dan lain – lain.

2.2.6.1. Mean Absolute Error (MAE)

Mean Absolute Error (MAE) adalah rata-rata dari selisih absolut antara nilai yang diamati dan nilai yang diprediksi [14]. MAE juga dikenal sebagai Mean Absolute Deviation. Perbedaan antara MAE dan MSE adalah bahwa MAE mengambil selisih absolut antara nilai yang diprediksi dan nilai aktual, sedangkan MSE mengambil selisih kuadrat. Persamaan untuk MAE adalah sebagai berikut:

$$\boldsymbol{MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|} \quad (2.8)$$

Keterangan:

MAE = Mean Absolute Error

n = Jumlah data

$\hat{y}_i$ = Nilai prediksi

y_i = Nilai aktual

2.2.6.2. Mean Squared Error (MSE)

Mean Squared Error (MSE), juga dikenal sebagai Mean Squared Deviation, merupakan pengukuran dari perbedaan kuadrat antara nilai yang sebenarnya dan nilai yang telah diprediksi [14]. MSE digunakan untuk mengevaluasi sejauh mana garis atau model yang digunakan cocok dengan kumpulan data yang ada. MSE selalu memiliki nilai positif karena perbedaan kuadrat menghilangkan tanda negatif. Ketika nilai MSE mendekati nol, hal ini menunjukkan bahwa prediksi semakin mendekati nilai yang sebenarnya, yang berarti prediksi menjadi semakin akurat. Persamaan untuk MSE dapat dinyatakan sebagai berikut:

$$\boldsymbol{MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \quad (2.9)$$

Keterangan:

MSE = Mean Square Error

n = Jumlah data

$\hat{y}_i$ = Nilai prediksi

y_i = Nilai aktual

2.2.6.3. Root Mean Squared Error (RMSE)

Root Mean Squared Error (RMSE) adalah akar kuadrat dari rata-rata kuadrat dari semua *error* [14]. RMSE juga dikenal sebagai Root Mean Squared Deviation. Dengan kata lain, RMSE adalah standar deviasi dari *error*. RMSE juga mengindikasikan sejauh mana garis terbaik cocok dengan sekumpulan titik data. Persamaan untuk RMSE adalah sebagai berikut:

$$\boldsymbol{RMSE} = \sqrt{\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \frac{(\boldsymbol{\hat{y}_i - y_i})^{\boldsymbol{2}}}{\boldsymbol{n}}} = \sqrt{\boldsymbol{MSE}} \qquad (2.10)$$

Keterangan:

RMSE = Root Mean Squared Error

n = Jumlah data

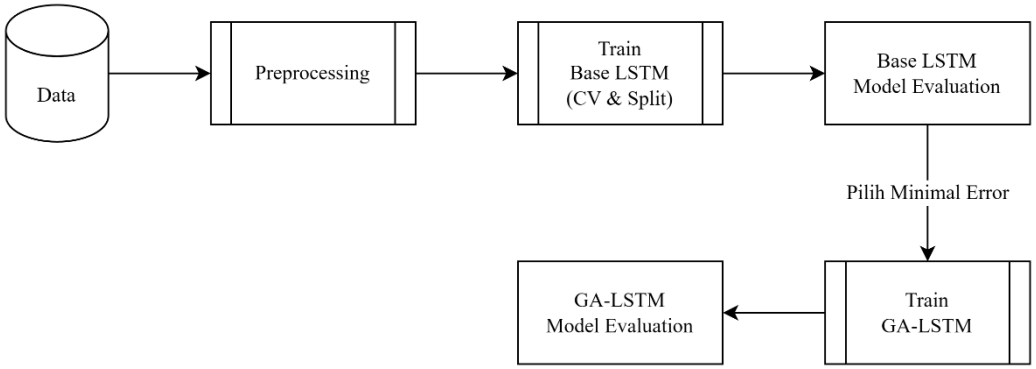
$\hat{y}_i$ = Nilai prediksi

y_i = Nilai aktual

MSE = Mean Squared Error

BAB III

METODE PENELITIAN



Gambar 3.1 Langkah Penelitian

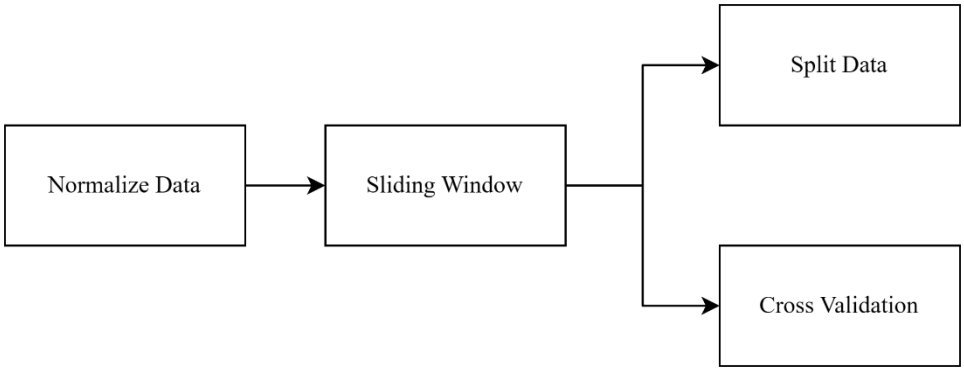
3.1. Deskripsi Data

Data yang akan peneliti gunakan untuk penelitian ini adalah data harga beli dari sebuah valuta asing setiap harinya. Data yang peneliti ambil memiliki rentang waktu kurang lebih 5 tahun sebelumnya, mulai dari 1 Januari 2018 – 31 Mei 2023. Valuta asing yang peneliti gunakan adalah USD/IDR, EUR/IDR, dan SGD/IDR. Dimana tiap – tiap data tersebut hanya memiliki 2 atribut, yaitu tanggal dan harga beli. Data – data tersebut diambil dari platform Google Finance dengan menggunakan fungsi yang sudah disediakan oleh Google Spreadsheet.

Tabel 3.1 Contoh Data Mentah

Date	Close
01/01/2020 23:58:00	13689.23
02/01/2020 23:58:00	13884.79
03/01/2020 23:58:00	13935.46

3.2. Preprocessing



Gambar 3.2 Langkah Preprocessing

3.2.1. Normalisasi Data

Langkah pertama *preprocessing* data valuta asing adalah normalisasi. Normalisasi yang akan digunakan adalah normalisasi *min-max*. Dimana tiap – tiap fitur valuta asing tersebut di normalisasi menggunakan persamaan 2.1 dan hasilnya seperti yang terdapat pada tabel 3.2.

Tabel 3.2 Contoh Data Normalisasi

Sebelum Normalisasi	Sesudah Normalisasi
13689.23	0
13884.79	0.794217
13935.46	1

3.2.2. Sliding Window

Langkah selanjutnya adalah menyegmentasi data menggunakan *sliding window*. Dimana data – data tersebut akan disegmentasikan berdasarkan ukuran jendela. Peneliti memilih untuk menggabungkan tiga ukuran jendela yang berbeda, yaitu 5, 10, dan 20.

3.2.3. Split Data

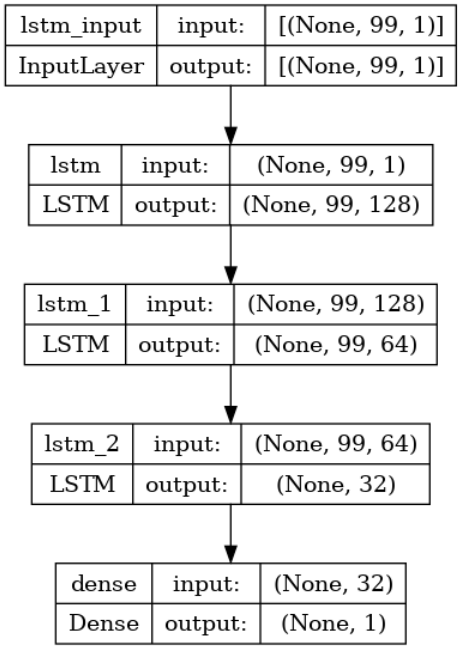
Langkah selanjutnya adalah membagi data menjadi 3 bagian, yaitu, *training*, *validation*, dan *test*. Pembagian data ini dilakukan dengan mengikuti dua komposisi yang berbeda, yaitu 90% untuk data *training* dan 10% untuk data *testing*, serta 80% untuk data *training* dan 20% untuk data *testing*. Tujuan dari langkah ini adalah untuk mengevaluasi apakah variasi dalam pembagian data dapat mempengaruhi nilai *error*. Sedangkan untuk bagian *validation*, data akan otomatis terbuat jika memasukkan parameter saat melatih model dan ukurannya kurang lebih adalah 10% dari total data *training*.

3.2.4. Cross Validation

Sama seperti pendekatan *Split Data* konvensional, dalam konteks ini peneliti akan menerapkan Metode *Cross Validation* (CV), khususnya *Time Series Cross Validation* (TSCV). Pemilihan metode ini bertujuan untuk mengevaluasi potensi perbedaan dalam nilai *error* yang dihasilkan oleh model ketika menggunakan pendekatan *Split Data* konvensional dan metode TSCV. Peneliti akan menggunakan nilai TSCV sebesar 5 dan 10 untuk pengujian ini.

3.3. Implementasi Model

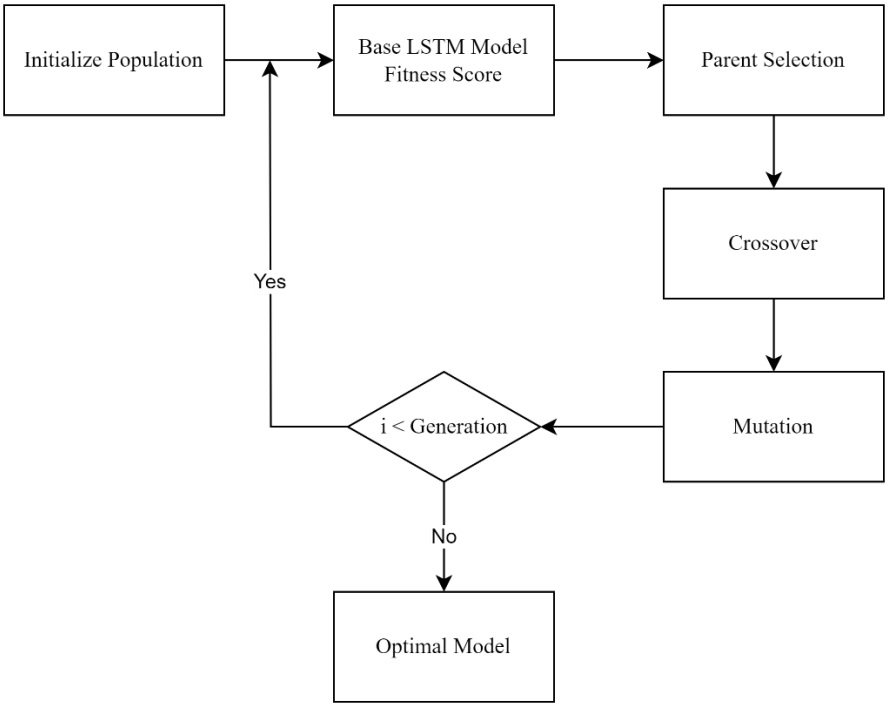
3.3.1. Base LSTM



Gambar 3.3 Arsitektur LSTM

Gambar 3.3 menunjukkan standar LSTM model arsitektur. Arsitektur model yang peneliti gunakan terdiri atas *Input layer* dengan ukuran sebesar *sliding window* dan jumlah atribut yang digunakan. Dilanjutkan dengan 3 LSTM *layer* yang memiliki ukuran 128, 64, dan 32. Dimana nanti 2 layer LSTM terakhir, 64 dan 32, akan dibuka secara bergantian saat percobaan. Terakhir, hasil dikeluarkan oleh *Dense output layer* dengan ukuran 1.

3.3.2. GA-LSTM



Gambar 3.4 Langkah GA-LSTM

Gambar 3.4 menunjukkan langkah bagaimana GA mengoptimalkan parameter yang ada di LSTM. Arsitektur model yang digunakan juga sama seperti gambar 3.3. Dimana nanti 2 layer LSTM terakhir, 64 dan 32, akan dibuka secara bergantian saat percobaan.

a. Inisiasi Populasi

Langkah pertama adalah menginisiasi populasi secara acak. Dimana populasi tersebut akan berbentuk *list* 2 dimensi, dengan panjang baris sepanjang apa yang dimasukkan oleh peneliti dan 3 kolom yang menentukan parameter dari *layer* LSTM.

b. Menghitung Fitness

Langkah kedua adalah menghitung nilai *fitness*. Nilai *fitness* didapatkan setiap kali model dilatih dengan kromosom atau setiap data yang ada pada populasi dan dievaluasi dengan data *test*.

c. Seleksi

Langkah ketiga adalah seleksi kromosom yang nanti akan di jadikan induk pada saat *crossover*. Tipe seleksi yang peneliti gunakan adalah *tournament selection*.

d. Crossover

Langkah keempat adalah melakukan *crossover* terhadap kromosom yang telah seleksi. Tipe *crossover* yang peneliti gunakan adalah *single-point crossover*.

e. Mutasi

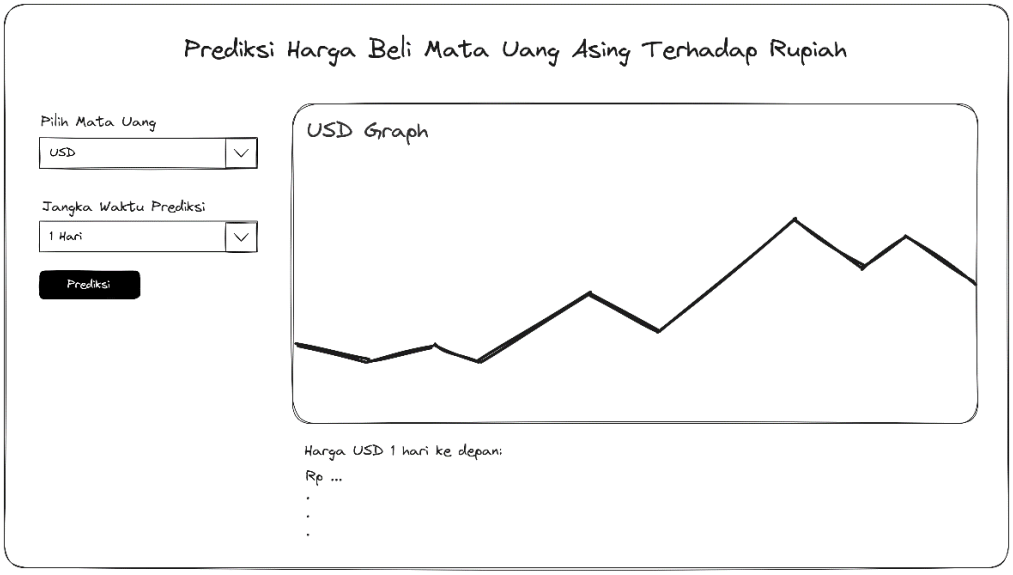
Langkah kelima adalah melakukan mutasi terhadap hasil *crossover*. Tipe mutasi yang peneliti gunakan adalah *swap mutation*. Dimana mutasi tersebut akan dijalankan dengan probabilitas 1%.

Terakhir, langkah b sampai dengan e akan diulang terus menerus sampai dengan generasi yang dimasukkan oleh peneliti. Setelah iterasi selesai, GA akan mengembalikan nilai yang paling optimal sebagai parameter model LSTM.

3.4. Evaluasi Model

Dalam penelitian ini kedua model tersebut akan dievaluasi dengan data yang belum pernah dilihat. Hasil prediksi sebuah model dan nilai aktual dapat mendapatkan nilai bagaimana performa model tersebut dalam memprediksi data. Nilai performa model tersebut dapat dihitung menggunakan dihitung dengan persamaan 2.8, 2.9, dan 2.10.

3.5. Desain User Interface



Gambar 3.5 Rancangan Desain GUI

Rancangan desain GUI yang peneliti ajukan kurang lebih seperti di atas. Untuk langkah penggunaannya:

1. Pilih mata uang yang akan di prediksi pada dropdown yang telah disediakan
2. Pilih jangka waktu mata uang yang ingin di prediksi
3. Tekan tombol prediksi
4. Tunggu program selesai menjalankan model dan hasilnya akan ditampilkan di bawah grafik

3.6. Kebutuhan Hardware dan Software

a. Spesifikasi Hardware
    1. Intel(R) Core(TM) i7-11800H @ 2.30GHz
    2. NVIDIA GeForce RTX 3050Ti
    3. RAM 16 GB
    4. SSD
b. Spesifikasi Software
    1. Windows 11
    2. Visual Studio Code (Base LSTM)
    3. Kaggle (GA-LSTM)
    4. Python 3.10
    5. Library Python:
        - TensorFlow dan Keras
        - Sklearn
        - Pandas
        - Numpy
        - Matplotlib

3.7. Rancangan Skenario Pengujian

Tabel 3.3 Skenario Pengujian Split

Skenario	Model	Parameter		
		LSTM Layer	Sliding Window	Train Split
1	Base LSTM (50 Epoch)	1	5	0.8
2				0.9
3			10	0.8
4				0.9
5			20	0.8
6				0.9
7		2	5	0.8
8				0.9
9			10	0.8
10				0.9
11			20	0.8
12				0.9
13		3	5	0.8
14				0.9
15			10	0.8
16				0.9
17			20	0.8
18				0.9

Tabel 3.4 Skenario Pengujian Cross Validation

Skenario	Model	Parameter		
		LSTM Layer	Sliding Window	K-Fold CV
1	Base LSTM (50 Epoch)	1	5	5
2				10
3			10	5
4				10
5			20	5
6				10
7		2	5	5
8				10
9			10	5
10				10
11			20	5
12				10

13		3	5	5
14				10
15			10	5
16				10
17			20	5
18				10

Setelah mendapatkan hasil pengujian pada masing – masing skenario, yaitu MAE, MSE, dan RMSE. Langkah selanjutnya adalah memilih skenario yang memiliki nilai minimal pada masing – masing kelompok *sliding window* dan menerapkan algoritma genetik untuk mengoptimasi unit LSTM.

BAB IV

HASIL PENELITIAN DAN PEMBAHASAN

4.1. Pengambilan Data

Dalam penelitian ini, harga nilai tukar mata uang asing yang digunakan diambil dari *website* Google Finance dengan bantuan Google Spreadsheet. Total 1977 data berhasil dikumpulkan untuk mata uang USD dan EUR, sementara untuk mata uang SGD hanya berhasil dikumpulkan sebanyak 1956 data. Berikut merupakan hasil pengambilan data menggunakan Google Spreadsheet:

Date	Close		Date	Close		Date	Close
1/1/2018 23:58	13550		1/1/2018 23:58	16274.77		1/1/2018 23:58	10127.81
2/1/2018 23:58	13496		2/1/2018 23:58	16280.09		2/1/2018 23:58	10153.04
3/1/2018 23:58	13468		3/1/2018 23:58	16174.8		3/1/2018 23:58	10118.64
4/1/2018 23:58	13415		4/1/2018 23:58	16189.36		4/1/2018 23:58	10103.24
5/1/2018 23:58	13411		5/1/2018 23:58	16131.02		5/1/2018 23:58	10103.98
6/1/2018 23:58	13250		6/1/2018 23:58	15937.1		7/1/2018 23:58	10117.38
7/1/2018 23:58	13427		7/1/2018 23:58	16161.81		8/1/2018 23:58	10075.76
8/1/2018 23:58	13427		8/1/2018 23:58	16071.85		9/1/2018 23:58	10044.16
9/1/2018 23:58	13430		9/1/2018 23:58	16020.11		10/1/2018 23:58	10073.49
10/1/2018 23:58	13441		10/1/2018 23:58	16069.52		11/1/2018 23:58	10073.01
....			....			....	
21/05/2023 23:58:00	14925		21/05/2023 23:58:00	16101		21/05/2023 23:58:00	11096.74
22/05/2023 23:58:00	14893		22/05/2023 23:58:00	16097		22/05/2023 23:58:00	11061.68
23/05/2023 23:58:00	14902		23/05/2023 23:58:00	16061		23/05/2023 23:58:00	11062.77
24/05/2023 23:58:00	14952.45		24/05/2023 23:58:00	16025		24/05/2023 23:58:00	11083.48
25/05/2023 23:58:00	14977		25/05/2023 23:58:00	16006		25/05/2023 23:58:00	11054.12
26/05/2023 23:58:00	15008.1		26/05/2023 23:58:00	16031		26/05/2023 23:58:00	11099.84
27/05/2023 23:58:00	15008.1		27/05/2023 23:58:00	16031		27/05/2023 23:58:00	11099.84
28/05/2023 23:58:00	14955		28/05/2023 23:58:00	16089		28/05/2023 23:58:00	11057.18
29/05/2023 23:58:00	14969.25		29/05/2023 23:58:00	16035		29/05/2023 23:58:00	11055.21
30/05/2023 23:58:00	14986		30/05/2023 23:58:00	16075		30/05/2023 23:58:00	11094.86
31/05/2023 23:58:00	14991		31/05/2023 23:58:00	16075		31/05/2023 23:58:00	11092.94

Gambar 4.1. Contoh Data Harga Beli Setiap Mata Uang

Untuk mempermudah pembacaan data, nilai tukar setiap mata uang telah diatur dalam *sheet* yang terpisah, dan agar dapat diakses melalui library pandas, *spreadsheet* harus dibuka terlebih dahulu untuk mendapatkan *link* yang diperlukan.

4.2. Preprocessing

4.2.1. Normalisasi

Tahapan pertama dalam *preprocessing* adalah normalisasi data. Atribut yang akan dinormalisasi adalah atribut *close*, dimana metode normalisasi yang digunakan adalah normalisasi *min-max*. Berikut merupakan implementasi kode untuk normalisasi:

```
scaler = MinMaxScaler()
close_price = df.Close.values.reshape(-1, 1)
scaled_close = scaler.fit_transform(close_price)
```

Gambar 4.2 Source Code Normalisasi

Pada proses ini, penulis menggunakan fungsi yang telah disediakan yaitu, MinMaxScaler, reshape, dan fit_transform. Fungsi tersebut berfungsi untuk memanggil fungsi normalisasi min-max, mengubah bentuk data menjadi 2 dimensi, dan mengubah data ke dalam bentuk normal. Berikut merupakan bentuk dan data hasil normalisasi:

```
---------- Normalize Data Shape ----------
(1959, 1)

---------- Normalize Data ----------
[[0.13503666]
 [0.14700219]
 [0.13068802]
 ...
 [0.57488343]
 [0.59368865]
 [0.59277804]]
```

Gambar 4.3 Contoh Data Hasil Normalisasi

4.2.2. Sliding Window

Selanjutnya adalah menerapkan *sliding window* pada data yang telah dinormalisasi. *Sliding window* ini akan menjadi salah satu variabel pengamatan untuk melihat performa model. Variabel ini memiliki berbagai macam nilai yaitu, 5, 10, dan 20. Berikut merupakan implementasi kode untuk *sliding window*:

```
def to_sequences(data, seq_len):
    d = []
    for index in range(len(data) - seq_len):
        d.append(data[index: index + seq_len])
    return np.array(d)
```

Gambar 4.4 Source Code Sliding Window

Fungsi ini menerima *input* berupa data dan panjang sekuens (*seq_len*). Pertama, fungsi ini membuat *list* kosong *d*. Kemudian, melakukan iterasi melalui data, mulai dari indeks 0 hingga panjang data dikurangi dengan *seq_len*. Pada setiap iterasi, fungsi ini mengambil

sekuens data dari indeks saat ini hingga indeks ditambah *seq_len* dan menambahkannya ke *list d*. Proses ini berlanjut hingga semua sekuens dengan panjang *seq_len* telah ditambahkan ke *list*. Akhirnya, fungsi ini mengembalikan *list d* sebagai *array NumPy*. Berikut merupakan data hasil penerapan *sliding window*:

```
----------- Sliding Window Data Shape -----------
(1938, 21, 1)
```

Gambar 4.5 Bentuk Data Hasil Sliding Window

4.2.3. Split Data

Tahapan terakhir sebelum masuk ke dalam model LSTM adalah menerapkan *split data* atau *cross validation*. Pada tahap *split data*, data akan dipisah menjadi 2 jenis, yaitu *data train* dan *data test*. Persentase *data train* juga akan dijadikan sebagai variabel pengamatan yang dimana memiliki nilai sebesar 80% dan 90%. Berikut merupakan implementasi kode untuk *split data*:

```
def preprocess(data_raw, seq_len, train_split):
    data = to_sequences(data_raw, seq_len)
    num_train = int(train_split * data.shape[0])
    X_train = data[:num_train, :-1, :]
    y_train = data[:num_train, -1, :]
    X_test = data[num_train:, :-1, :]
    y_test = data[num_train:, -1, :]
    return X_train, y_train, X_test, y_test
```

Gambar 4.6 Source Code Split Data

Setelah data berubah dalam bentuk sekuens, data akan dibagi menjadi data *train* dan data *test*. Dimana *X_train* dan *X_test* berisi semua data kecuali yang terakhir dari setiap sekuens, sementara *y_train* dan *y_test* berisi data terakhir dari setiap sekuens. Berikut merupakan bentuk data setelah penerapan *split data*:

```
---------- Train Data Shape ----------
(1744, 20, 1)
(1744, 1)
---------- Test Data Shape ----------
(194, 20, 1)
(194, 1)
```

Gambar 4.7 Bentuk Data Setelah Split

4.2.4. Cross Validation

Pada tahap *cross validation*, metode yang digunakan adalah Time Series Cross Validation. Sama seperti *split data*, hal ini juga akan dijadikan sebagai variabel pengamatan yang dimana memiliki nilai yaitu, 5 dan 10. Berikut merupakan implementasi kode untuk cross validation:

```python
def to_sequences(data, seq_len):
    d = []
    for index in range(len(data) - seq_len):
        d.append(data[index: index + seq_len])
    return np.array(d)

def preprocess(data_raw, seq_len):
    data = to_sequences(data_raw, seq_len)
    target = data[:, -1, :]
    input = data[:, :-1, :]
    return input, target

inputs, targets = preprocess(scaled_close, SEQ_LEN)

tscv = TimeSeriesSplit(n_splits=FOLD)
```

Gambar 4.8 Source Code TSCV

Code yang digunakan tidak jauh berbeda dengan yang ada pada *split data*. Setelah data, dipisah menjadi input (X) dan target (y). Langkah selanjutnya adalah memasukkan kedua data tersebut ke dalam fungsi TimeSeriesSplit, yang berfungsi membagi data *time series* menjadi beberapa *fold*, dengan setiap *fold* berisi lebih banyak dari *fold* sebelumnya. Berikut merupakan bentuk data setelah penerapan *cross validation*:

```
Fold No - 1
---------- Train Data Shape ----------
(323, 20, 1)
(323, 1)
---------- Test Data Shape ----------
(323, 20, 1)
(323, 1)

Fold No - 2
---------- Train Data Shape ----------
(646, 20, 1)
(646, 1)
---------- Test Data Shape ----------
(323, 20, 1)
(323, 1)
```

Gambar 4.9 Bentuk Data Setelah Cross Validation

4.3. Base LSTM

Setelah data melewati tahap *preprocessing*, tahap selanjutnya adalah melatih dan menguji model dasar (*Base LSTM*). Dalam hal ini, Base LSTM yang dimaksud adalah model dengan layer LSTM yang jumlah neutronnya sudah ditentukan di awal. Jumlah neuron atau unit yang digunakan adalah 128, 64, dan 32 untuk layer satu sampai dengan yang ketiga. Jumlah layer LSTM dalam sebuah model juga akan dijadikan variabel pengamatan. Berikut merupakan implementasi kode dalam membentuk model:

```
tf.keras.backend.clear_session()
model = Sequential()

for i, units in enumerate([128, 64, 32][:LSTM_Layer]):
    model.add(LSTM(units, return_sequences=(i < LSTM_Layer - 1), input_shape=(WINDOW_SIZE, 1)))
model.add(Dense(units=1))

model.summary()
```

Gambar 4.10 Source Code Pembuatan Model

Objek *Sequential* dibuat dan disimpan dalam variabel *model*, yang nantinya digunakan sebagai kerangka untuk menambahkan layer dalam model. Selanjutnya, kode melakukan iterasi melalui daftar unit hingga jumlah LSTM yang diinginkan dan menambahkan lapisan LSTM ke model untuk setiap unit dalam daftar. Parameter *return_sequences* diatur ke *True* untuk semua lapisan kecuali lapisan terakhir, dan *input_shape* diatur ke ukuran sliding window. Setelah semua lapisan LSTM ditambahkan, lapisan Dense dengan unit 1 ditambahkan ke model. Berikut merupakan salah satu bentuk model yang akan ditrain dan ditest:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 20, 128)           66560

 lstm_1 (LSTM)               (None, 20, 64)            49408

 lstm_2 (LSTM)               (None, 32)                12416

 dense (Dense)               (None, 1)                 33

=================================================================
Total params: 128,417
Trainable params: 128,417
Non-trainable params: 0
_________________________________________________________________
```

Gambar 4.11 Struktur Base LSTM

4.4. Optimasi Parameter LSTM

Setelah semua data diterapkan pada skenario dan mendapatkan hasil *error*. Tahap terakhir adalah mengoptimalkan jumlah unit yang terdapat pada layer LSTM. Dimana tidak semua skenario akan dioptimalkan parameternya, hanya skenario yang memiliki nilai *error*

paling rendah pada tiap – tiap kelompok *sliding window* yang akan dioptimalkan. Berikut merupakan implementasi kode algoritma genetik:

```
def genetic_algorithm(population_size, generations):
    population = np.random.randint(1, high=26, size=(population_size, 3))

    for generation in range(generations):
        print(f"Generation - {generation + 1}")
        fitness_scores = np.array([fitness_function(chromosome) for chromosome in population])
        best_chromosome = population[np.argmax(fitness_scores)]
        best_fitness = np.max(fitness_scores)

        new_population = []

        while len(new_population) < population_size:
            parent1 = selection(population, fitness_scores)
            parent2 = selection(population, fitness_scores)
            offspring_1, offspring_2 = crossover(parent1, parent2)
            mutate(offspring_1)
            mutate(offspring_2)
            new_population.append(offspring_1)
            new_population.append(offspring_2)

        population = np.array(new_population)

    return best_chromosome, best_fitness
```

Gambar 4.12 Source Code Algoritma Genetik

Fungsi ini menerima dua input, yaitu *population_size* yang menentukan jumlah kromosom dalam populasi, dan *generations* yang menentukan berapa banyak generasi yang harus dijalankan oleh algoritma. Populasi awal dibuat secara acak dengan menggunakan fungsi yang telah disediakan *NumPy*. Kemudian, untuk setiap generasi, skor *fitness* dihitung untuk setiap kromosom dalam populasi menggunakan fungsi *fitness_function*. Di dalam *fitness_function*, terdapat sebuah model LSTM yang akan dilatih dan dievaluasi untuk mendapatkan skor *fitness*. Kromosom dengan skor *fitness* tertinggi dan skor *fitness* disimpan dalam sebuah variabel. Berikut merupakan implementasi kode *fitness function*:

```python
def fitness_function(chromosome):
    lstm_units = [int(chromosome[i]*10) or default for i, default in enumerate([128, 64, 32])]

    # Build the LSTM model
    tf.keras.backend.clear_session()
    model = Sequential()
    for i, units in enumerate(lstm_units[:LSTM_Layer]):
        model.add(LSTM(units, return_sequences=(i < LSTM_Layer - 1), input_shape=(WINDOW_SIZE, 1)))
    model.add(Dense(1))

    # Compile and train the model
    model.compile(loss='mean_squared_error',
                  optimizer='adam')
    model.fit(X_train, y_train,
              epochs=10,
              batch_size=32,
              verbose=0,
              validation_split=0.1)

    # Evaluate the model
    loss = model.evaluate(X_test, y_test)

    # Return the negative value of the loss as the fitness score
    return -loss
```

Gambar 4.13 Source Code Fitness Function

Populasi baru kemudian dibuat dengan memilih pasangan induk dari populasi saat ini menggunakan fungsi *selection*, melakukan *crossover* pada mereka untuk menghasilkan dua keturunan, dan kemudian melakukan mutasi pada kedua keturunan yang dihasilkan. Fungsi *selection* akan memilih 3 kromosom secara acak, yang dimana yang lulus seleksi adalah kromosom yang memiliki skor *fitness* tertinggi. Berikut merupakan implementasi kode *tournament selection*:

```python
def selection(population, fitness_scores, tournament_size=3):
    indices = np.random.randint(len(population), size=tournament_size)
    tournament = population[indices]
    tournament_fitness = fitness_scores[indices]
    return tournament[np.argmax(tournament_fitness)]
```

Gambar 4.14 Source Code Tournament Selection

Fungsi *crossover* memilih secara acak satu titik antara 1 dan panjang kromosom, menghasilkan keturunan dengan menggabungkan bagian awal induk pertama dan bagian akhir induk kedua, serta sebaliknya.

Proses ini menciptakan kromosom baru sebagai hasil dari persilangan dua kromosom induk. Berikut merupakan implementasi kode *single-point crossover*:

```
def crossover(parent1, parent2):
    crossover_point = np.random.randint(1, len(parent1))
    offspring_1 = np.concatenate((parent1[:crossover_point], parent2[crossover_point:]))
    offspring_2 = np.concatenate((parent2[:crossover_point], parent1[crossover_point:]))
    return offspring_1, offspring_2
```

Gambar 4.15 Source Code Single-Point Crossover

Fungsi mutasi beroperasi dengan mengiterasi melalui setiap gen dalam kromosom dan, untuk setiap gen, menghasilkan angka acak antara 0 dan 1. Mutasi terjadi hanya jika angka acak tersebut berada di bawah 1%, yang mengakibatkan pertukaran posisi gen secara acak. Berikut merupakan implementasi kode *swap mutation*:

```
def mutate(chromosome, mutation_rate=0.01):
    for i in range(len(chromosome)):
        if np.random.rand() < mutation_rate:
            j = np.random.randint(len(chromosome))
            chromosome[i], chromosome[j] = chromosome[j], chromosome[i]
```

Gambar 4.16 Source Code Swap Mutation

Proses ini diulang sampai populasi baru mencapai ukuran populasi yang ditentukan. Akhirnya, populasi diperbarui menjadi populasi baru, dan proses ini diulang untuk jumlah generasi yang ditentukan. Fungsi kemudian mengembalikan kromosom terbaik dan *fitness* terbaik. Setelah mendapatkan kromosom terbaik, hal tersebut digunakan ke dalam pembuatan model LSTM yang optimal.

4.5. Hasil Pengujian

Pada penelitian ini, dilakukan beberapa percobaan dengan data yang berbeda. Percobaan ini menggunakan variasi data, jumlah layer LSTM, ukuran *sliding window*, dan teknik pembagian data apakah menggunakan *split* atau *cross validation*.

4.5.1. Pengujian Menggunakan Data USD/IDR

a. Pengujian Menggunakan Teknik Split

1. Base LSTM

Data mata uang USD akan dibagi menggunakan teknik *split* dan dengan variasi *split*, yaitu 80% dan 90% sebagai data *train*. Hal tersebut digunakan untuk mengetahui bagaimana jumlah data *train* mempengaruhi *error* sebuah model Base LSTM. Berikut merupakan tabel hasil pengujian menggunakan Base LSTM:

Tabel 4.1 Tabel Hasil Pengujian Base LSTM - USD - Split

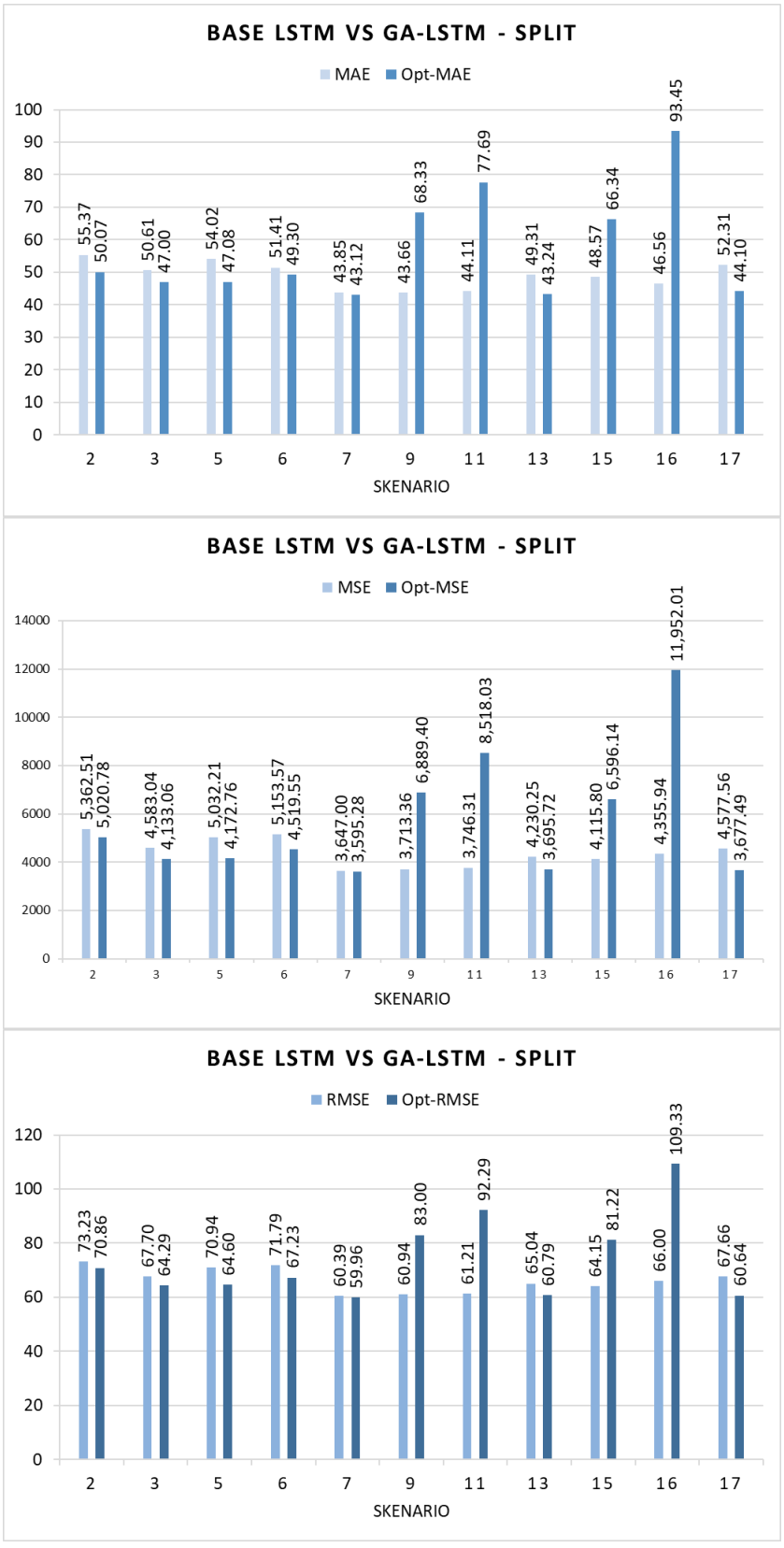
No.	LSTM Layer	Sliding Window	Train Split	MAE	MSE	RMSE
1	1	5	0.8	60.56185	6066.127	77.88534
2			0.9	55.36534	5362.506	73.22913
3		10	0.8	50.61418	4583.041	67.69816
4			0.9	53.95628	5328.28	72.99507
5		20	0.8	54.01649	5032.211	70.93808
6			0.9	51.40584	5153.569	71.78837
7	2	5	0.8	43.84822	3647.002	60.39042
8			0.9	46.60844	4351.589	65.96658
9		10	0.8	43.66192	3713.356	60.93731
10			0.9	70.3919	7314.721	85.52614
11		20	0.8	44.10654	3746.306	61.20707
12			0.9	67.88083	6834.217	82.66932
13	3	5	0.8	49.30926	4230.249	65.04036
14			0.9	90.6228	11532.68	107.3903
15		10	0.8	48.5728	4115.8	64.1545

16			0.9	46.56132	4355.938	65.99953
17		20	0.8	52.30634	4577.558	67.65765
18			0.9	65.92608	6896.733	83.04657

Background berwarna hijau merupakan nilai *error* minimal pada setiap kelompok *sliding window*, dan hasil tersebut akan menjadi fokus dalam penerapan GA-LSTM. Sementara itu, untuk *background* berwarna kuning menunjukkan bahwa ada nilai *error* yang lebih rendah dari yang lainnya di dalam kelompok tersebut, tetapi nilai *error* yang lain tetap tinggi. Oleh karena itu, peneliti memutuskan untuk mengoptimasi juga hal tersebut menggunakan GA-LSTM.

2. GA-LSTM

Setelah mendapatkan parameter yang menghasilkan nilai *error* paling minimal pada setiap kelompok. Langkah selanjutnya adalah mengoptimasi unit LSTM. Berikut merupakan grafik perbandingan nilai *error* antara Base LSTM dan GA-LSTM yang menggunakan teknik *split*:



Gambar 4.17 Grafik Perbandingan Base LSTM dan GA LSTM - USD - Split

Dari ketiga grafik di atas dapat dilihat bahwa mayoritas optimasi menggunakan GA-LSTM berhasil. Meskipun begitu ada beberapa optimasi yang tidak berhasil, contohnya ada pada skenario 9, 11, 15, dan 16. Di mana nilai *error* yang di hasilkan dari GA-LSTM lebih tinggi dari Base LSTM.

b. Pengujian Menggunakan Teknik Cross Validation

1. Base LSTM

Data mata uang USD akan dibagi menggunakan teknik *cross validation*, dan dengan variasi *fold*, yaitu 5 dan 10. Hal tersebut digunakan untuk mengetahui bagaimana jumlah *fold* mempengaruhi *error* sebuah model Base LSTM. Berikut merupakan tabel hasil pengujian menggunakan Base LSTM:

Tabel 4.2 Tabel Hasil Pengujian Base LSTM - USD - CV

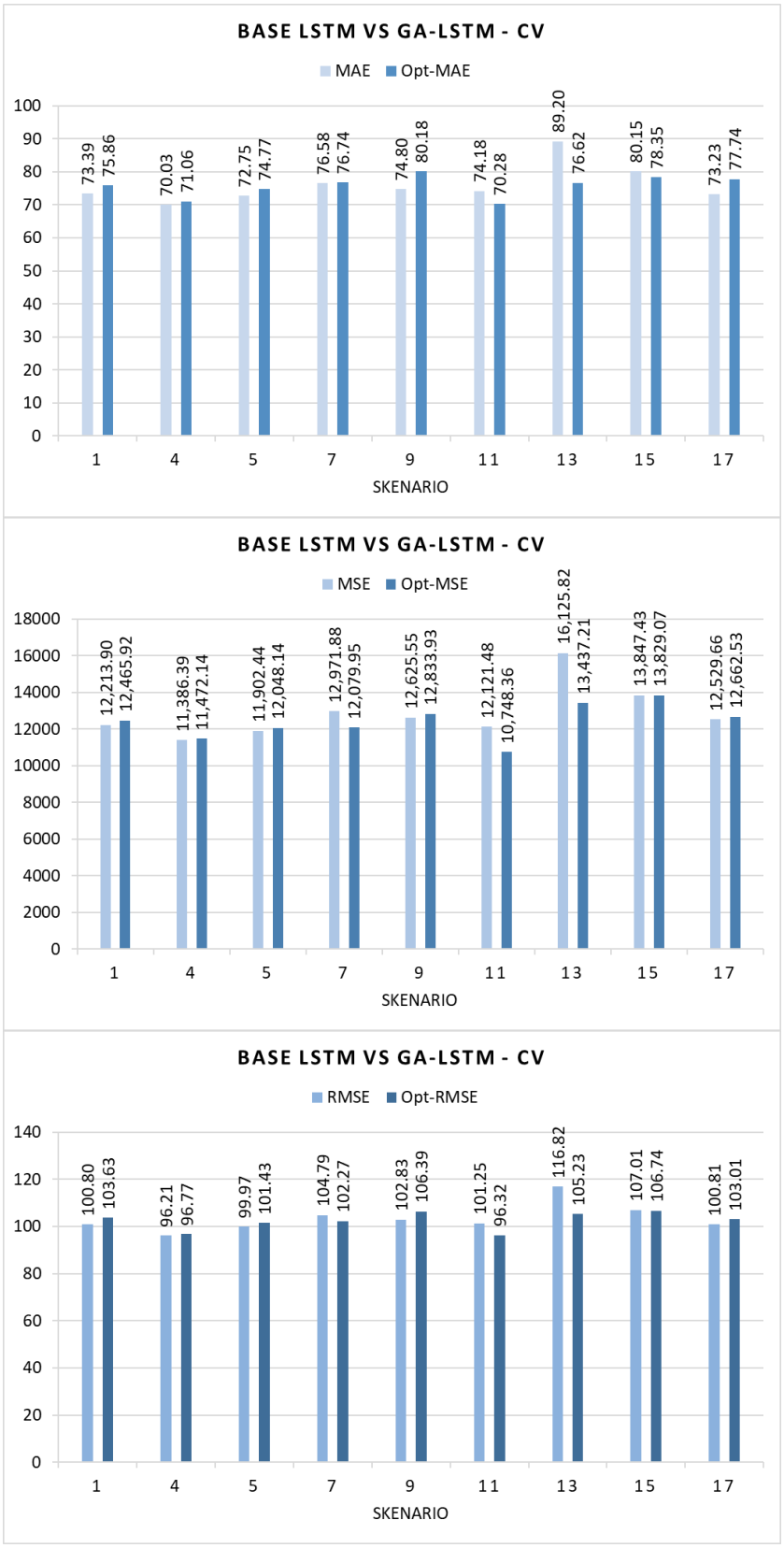
No.	LSTM Layer	Sliding Window	Fold	MAE	MSE	RMSE
1	1	5	5	73.38723	12213.9	100.7998
2			10	78.21488	13143.71	104.6313
3		10	5	73.13649	11626.64	99.09629
4			10	70.0252	11386.39	96.21182
5		20	5	72.74792	11902.44	99.97461
6			10	76.86195	12259.73	101.6533
7	2	5	5	76.57985	12971.88	104.7897
8			10	79.59907	13443.54	105.0076
9		10	5	74.80137	12625.55	102.8278
10			10	80.21139	13465.16	105.8685
11		20	5	74.18337	12121.48	101.2502
12			10	81.22573	12952.66	105.4363
13	3	5	5	89.19782	16125.82	116.8218
14			10	92.73233	18316.18	119.909
15		10	5	80.15152	13847.43	107.0129
16			10	88.41455	16825.52	114.3944
17		20	5	73.23473	12529.66	100.8125

18			10	86.12904	14803.58	110.5571

Background berwarna hijau merupakan nilai *error* minimal pada setiap kelompok *sliding window*, dan hasil tersebut akan menjadi fokus dalam penerapan GA-LSTM.

2. GA-LSTM

Setelah mendapatkan parameter yang menghasilkan nilai *error* paling minimal pada setiap kelompok. Langkah selanjutnya adalah mengoptimasi unit LSTM. Berikut merupakan grafik perbandingan nilai *error* antara Base LSTM dan GA-LSTM yang menggunakan teknik *cross validation*:



Gambar 4.18 Grafik Perbandingan Base LSTM dan GA LSTM - USD - CV

Dari ketiga grafik di atas dapat dilihat bahwa optimasi menggunakan GA-LSTM berhasil tetapi tidak sesignifikan sebelumnya, contohnya ada pada skenario 10, 11, 13, dan 15. Selain itu, meskipun nilai *error* dari skenario yang lain lebih tinggi, tetapi kenaikan tersebut tidak terlalu signifikan. Oleh karena itu, hal tersebut juga bisa disimpulkan bahwa optimasi berhasil.

4.5.2. Pengujian Menggunakan Data EUR/IDR

a. Pengujian Menggunakan Teknik Split

1. Base LSTM

Data mata uang EUR akan dibagi menggunakan teknik *split* dan dengan variasi *split*, yaitu 80% dan 90% sebagai data *train*. Hal tersebut digunakan untuk mengetahui bagaimana jumlah data *train* mempengaruhi *error* sebuah model Base LSTM. Berikut merupakan tabel hasil pengujian menggunakan Base LSTM:

Tabel 4.3 Tabel Hasil Pengujian Base LSTM - EUR - Split

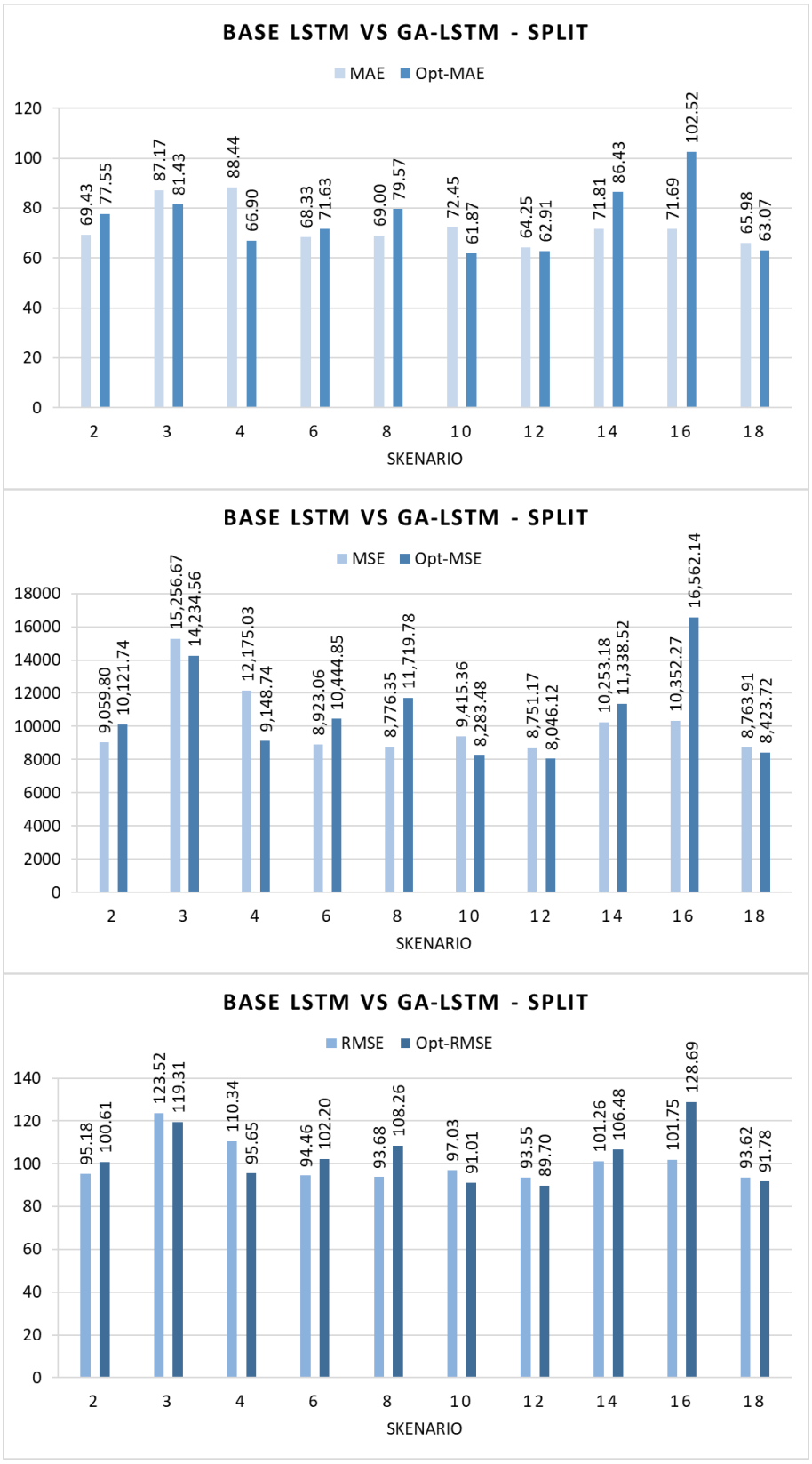
No.	LSTM Layer	Sliding Window	Train Split	MAE	MSE	RMSE
1	1	5	0.8	85.83965	14930.11	122.1888
2			0.9	69.42598	9059.795	95.18296
3		10	0.8	87.17181	15256.67	123.5179
4			0.9	88.4372	12175.03	110.3405
5		20	0.8	87.77099	15106.36	122.9079
6			0.9	68.32514	8923.064	94.46197
7	2	5	0.8	85.6312	14901.13	122.0702
8			0.9	69.00269	8776.348	93.68216
9		10	0.8	87.17326	15278.69	123.607
10			0.9	72.45287	9415.365	97.0328
11		20	0.8	85.14858	14689.49	121.2002
12			0.9	64.25243	8751.169	93.54768
13	3	5	0.8	89.85858	16216.62	127.3445

14			0.9	71.80528	10253.18	101.258
15		10	0.8	89.18191	15786.66	125.645
16			0.9	71.69277	10352.27	101.7461
17		20	0.8	87.07491	15263.29	123.5447
18			0.9	65.9785	8763.91	93.61576

Background berwarna hijau merupakan nilai *error* minimal pada setiap kelompok *sliding window*, dan hasil tersebut akan menjadi fokus dalam penerapan GA-LSTM. Sementara itu, untuk *background* berwarna kuning menunjukkan bahwa ada nilai *error* yang lebih rendah dari yang lainnya di dalam kelompok tersebut, tetapi nilai *error* yang lain tetap tinggi. Oleh karena itu, peneliti memutuskan untuk mengoptimasi juga hal tersebut menggunakan GA-LSTM.

2. GA-LSTM

Setelah mendapatkan parameter yang menghasilkan nilai *error* paling minimal pada setiap kelompok. Langkah selanjutnya adalah mengoptimasi unit LSTM. Berikut merupakan grafik perbandingan nilai *error* antara Base LSTM dan GA-LSTM yang menggunakan teknik *split*:



Gambar 4.19 Grafik Perbandingan Base LSTM dan GA LSTM - EUR - Split

Dari ketiga grafik di atas dapat dilihat bahwa hanya setengah optimasi menggunakan GA-LSTM berhasil secara signifikan, dan hal tersebut ditunjukkan pada skenario 3, 4, 10, 12, dan 18. Dimana nilai *error* yang di hasilkan dari GA-LSTM lebih rendah dari Base LSTM dan sisanya lebih tinggi.

b. Pengujian Menggunakan Teknik Cross Validation

1. Base LSTM

Data mata uang EUR akan dibagi menggunakan teknik *cross validation*, dan dengan variasi *fold*, yaitu 5 dan 10. Hal tersebut digunakan untuk mengetahui bagaimana jumlah *fold* mempengaruhi *error* sebuah model Base LSTM. Berikut merupakan tabel hasil pengujian menggunakan Base LSTM:

Tabel 4.4 Tabel Hasil Pengujian Base LSTM - EUR - CV

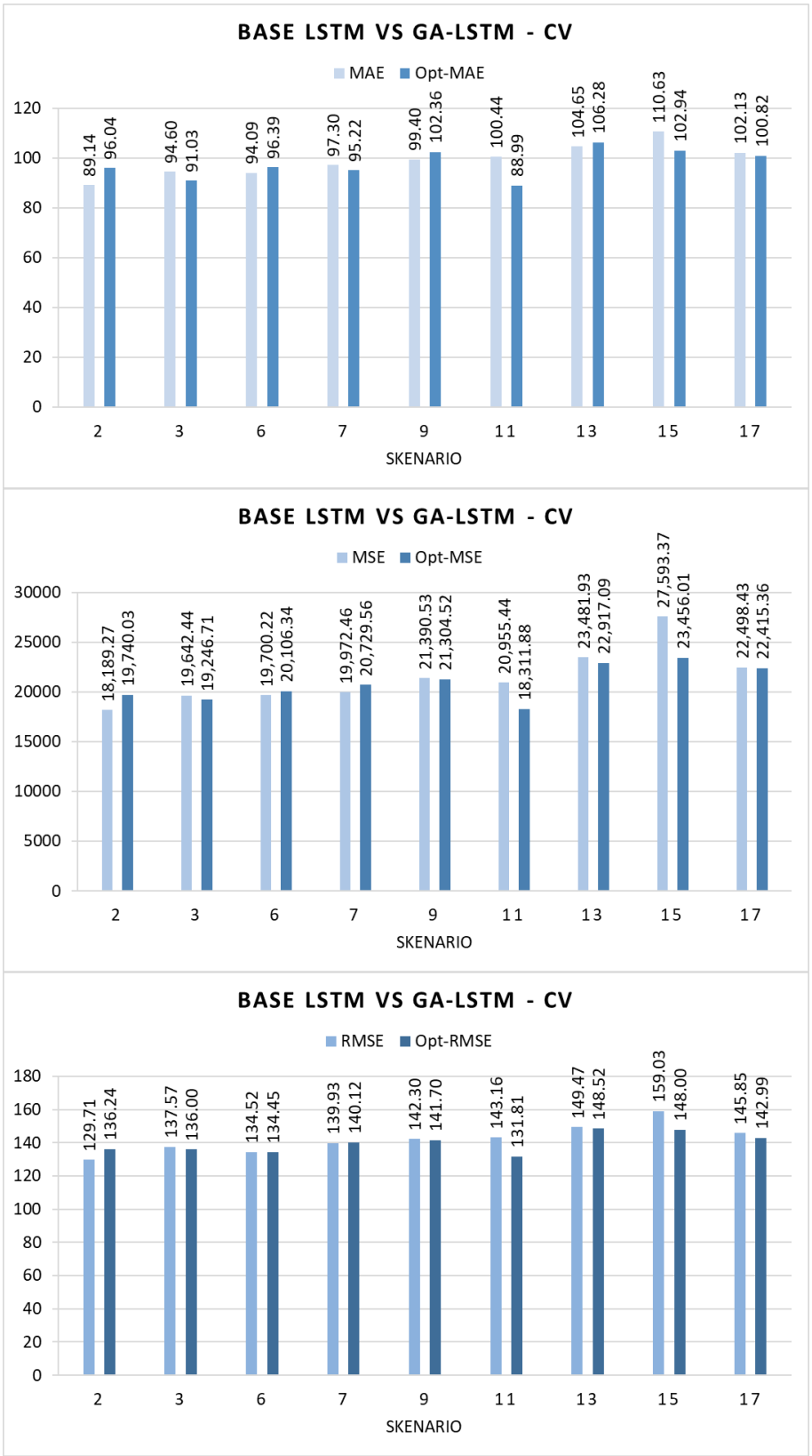
No.	LSTM Layer	Sliding Window	Fold	MAE	MSE	RMSE
1	1	5	5	96.20928	19993.68	138.1626
2			10	89.13667	18189.27	129.7138
3		10	5	94.59627	19642.44	137.565
4			10	98.79485	20793.56	138.2573
5		20	5	95.75965	19841.08	137.3672
6			10	94.09246	19700.22	134.5204
7	2	5	5	97.30276	19972.46	139.9317
8			10	107.1997	22258.56	145.6494
9		10	5	99.40198	21390.53	142.2954
10			10	104.7133	21969.74	143.2493
11		20	5	100.4403	20955.44	143.161
12			10	105.8104	23113.62	146.1236
13	3	5	5	104.6529	23481.93	149.4706
14			10	115.9938	25420.61	154.7026
15		10	5	110.6328	27593.37	159.0256
16			10	126.7253	30258.93	168.5489
17		20	5	102.1297	22498.43	145.8494

18			10	119.2702	27562.83	157.3196

Background berwarna hijau merupakan nilai *error* minimal pada setiap kelompok *sliding window*, dan hasil tersebut akan menjadi fokus dalam penerapan GA-LSTM.

2. GA-LSTM

Setelah mendapatkan parameter yang menghasilkan nilai *error* paling minimal pada setiap kelompok. Langkah selanjutnya adalah mengoptimasi unit LSTM. Berikut merupakan grafik perbandingan nilai *error* antara Base LSTM dan GA-LSTM yang menggunakan teknik *cross validation*:



Gambar 4.20 Grafik Perbandingan Base LSTM dan GA LSTM - EUR - CV

Dari ketiga grafik di atas dapat dilihat bahwa setengah optimasi menggunakan GA-LSTM berhasil. Meskipun begitu ada 1 optimasi yang tidak berhasil, contohnya ada pada skenario 2. Di mana nilai *error* yang di hasilkan dari GA-LSTM lebih tinggi dari Base LSTM. Sedangkan, untuk skenario 6, 7, dan 9, meskipun nilai *error* lebih tinggi, tetapi kenaikan tersebut tidak terlalu signifikan. Oleh karena itu, hal tersebut juga bisa disimpulkan bahwa optimasi berhasil.

4.5.3. Pengujian Menggunakan Data SGD/IDR

a. Pengujian Menggunakan Teknik Split

1. Base LSTM

Data mata uang SGD akan dibagi menggunakan teknik split dan dengan variasi *split*, yaitu 80% dan 90% sebagai data *train*. Hal tersebut digunakan untuk mengetahui bagaimana jumlah data *train* mempengaruhi *error* sebuah model Base LSTM. Berikut merupakan tabel hasil pengujian menggunakan Base LSTM:

Tabel 4.5 Tabel Hasil Pengujian Base LSTM - SGD - Split

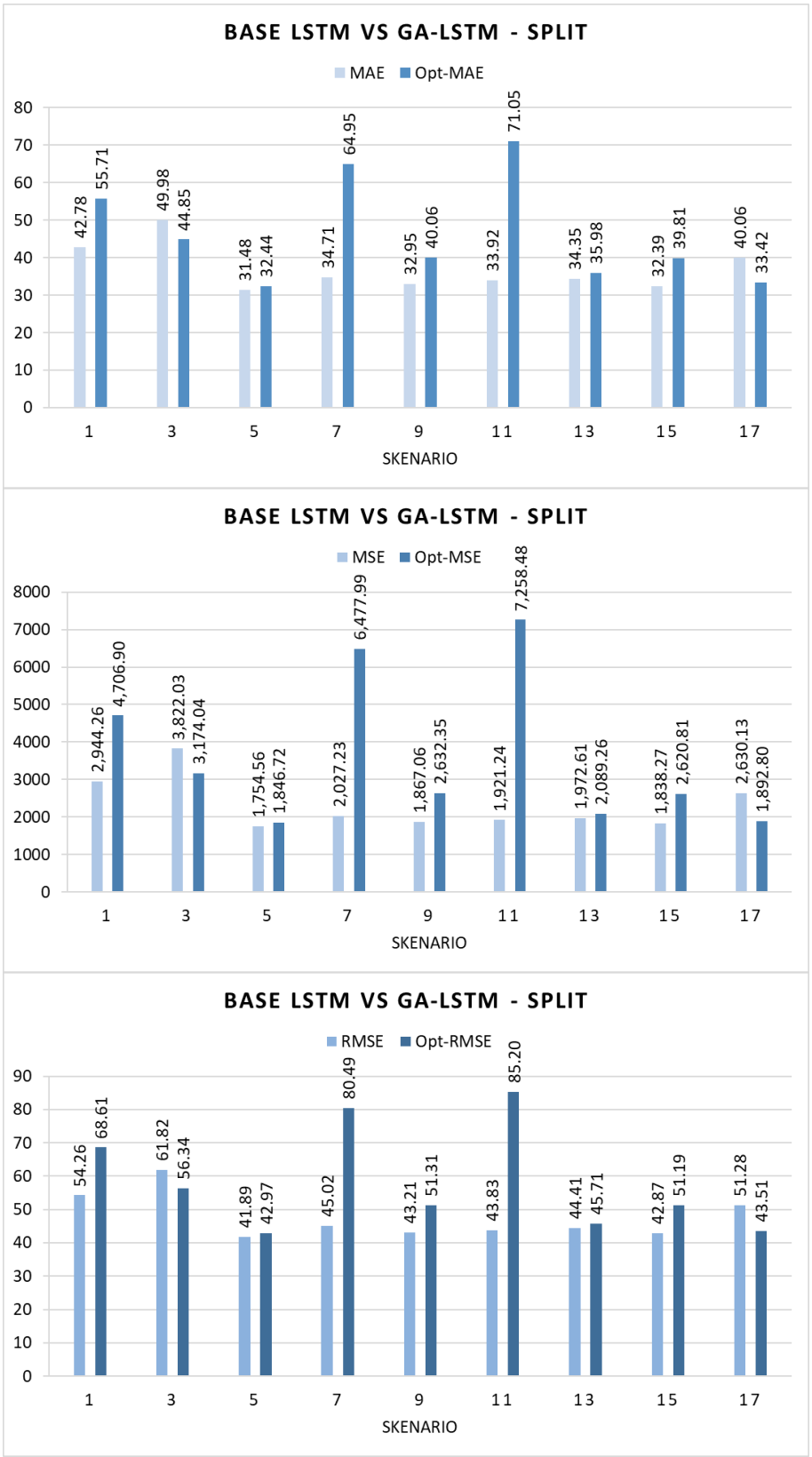
No.	LSTM Layer	Sliding Window	Train Split	MAE	MSE	RMSE
1	1	5	0.8	42.77984	2944.259	54.261
2			0.9	45.43815	3106.975	55.7403
3		10	0.8	49.9843	3822.025	61.8225
4			0.9	74.35366	6843.487	82.7254
5		20	0.8	31.48306	1754.561	41.8875
6			0.9	66.50409	5677.808	75.3512
7	2	5	0.8	34.70903	2027.232	45.0248
8			0.9	110.5612	14174.05	119.055
9		10	0.8	32.94833	1867.062	43.2095
10			0.9	80.53054	7933.33	89.0692
11		20	0.8	33.91728	1921.242	43.832

12			0.9	66.59807	5809.613	76.2208
13	3	5	0.8	34.35391	1972.607	44.414
14			0.9	52.80926	3977.744	63.0694
15		10	0.8	32.39082	1838.265	42.875
16			0.9	41.92661	2786.144	52.7839
17		20	0.8	40.05641	2630.129	51.2848
18			0.9	91.5707	10039.43	100.197

Background berwarna hijau merupakan nilai *error* minimal pada setiap kelompok *sliding window*, dan hasil tersebut akan menjadi fokus dalam penerapan GA-LSTM.

2. GA-LSTM

Setelah mendapatkan parameter yang menghasilkan nilai *error* paling minimal pada setiap kelompok. Langkah selanjutnya adalah mengoptimasi unit LSTM. Berikut merupakan grafik perbandingan nilai *error* antara Base LSTM dan GA-LSTM yang menggunakan teknik *split*:



Gambar 4.21 Grafik Perbandingan Base LSTM dan GA LSTM - SGD - Split

Dari ketiga grafik di atas dapat dilihat bahwa mayoritas optimasi menggunakan GA-LSTM tidak berhasil. Meskipun begitu ada beberapa optimasi yang berhasil, contohnya ada pada skenario nomor 3 dan 17. Di mana nilai *error* yang di hasilkan dari GA-LSTM lebih kecil dari Base LSTM. Sedangkan, untuk skenario 5 dan 13, meskipun nilai *error* lebih tinggi, tetapi kenaikan tersebut tidak terlalu signifikan. Oleh karena itu, hal tersebut juga bisa disimpulkan bahwa optimasi berhasil.

b. Pengujian Menggunakan Teknik Cross Validation

1. Base LSTM

Data mata uang SGD akan dibagi menggunakan teknik *cross validation*, dan dengan variasi *fold*, yaitu 5 dan 10. Hal tersebut digunakan untuk mengetahui bagaimana jumlah *fold* mempengaruhi *error* sebuah model Base LSTM. Berikut merupakan tabel hasil pengujian menggunakan Base LSTM:

Tabel 4.6 Tabel Hasil Pengujian Base LSTM - SGD - CV

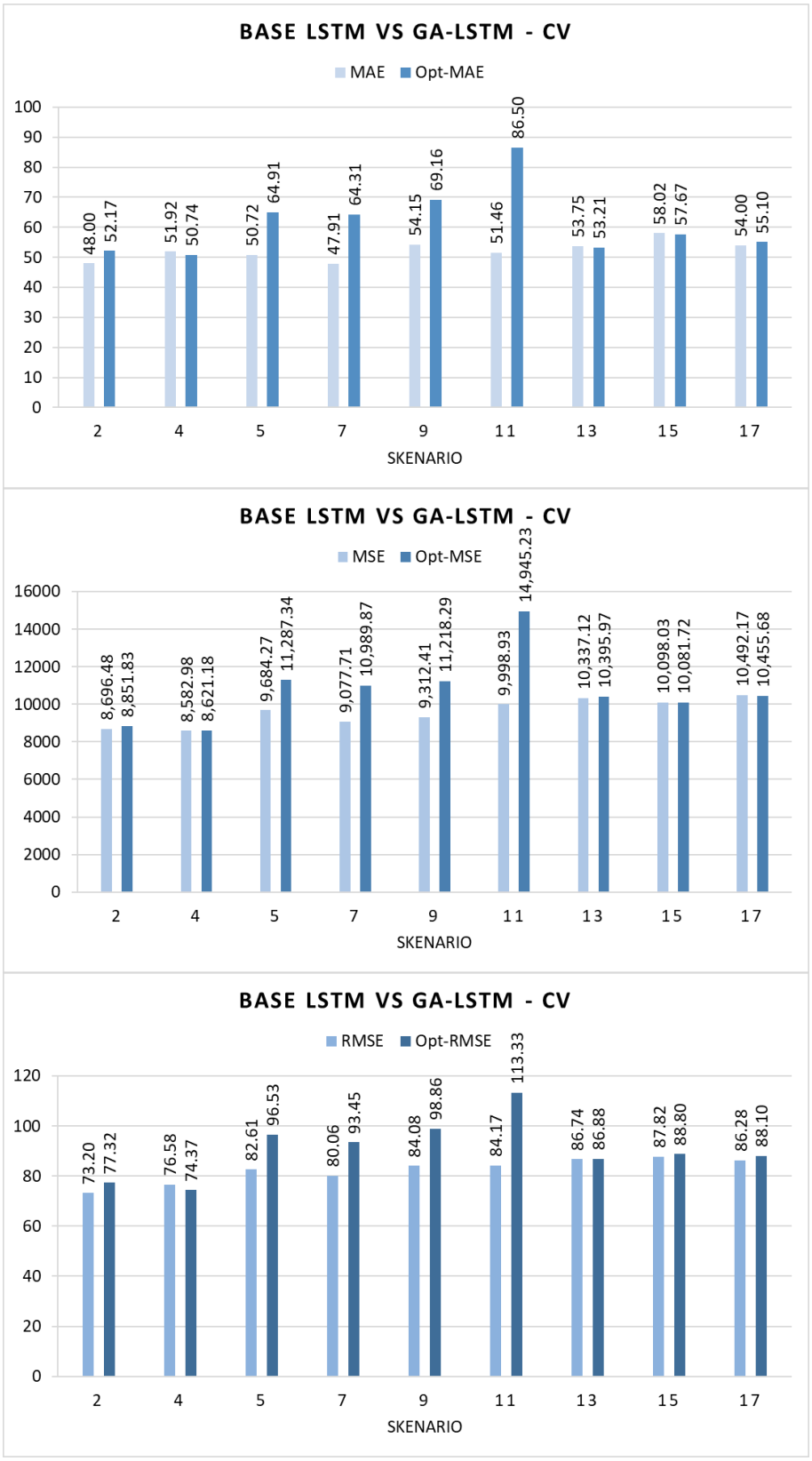
No.	LSTM Layer	Sliding Window	Fold	MAE	MSE	RMSE
1	1	5	5	52.03842	9606.051	84.4134
2			10	47.9954	8696.482	73.201
3		10	5	55.88893	9592.975	84.7546
4			10	51.9164	8582.981	76.5766
5		20	5	50.72133	9684.265	82.6104
6			10	59.96404	10316.8	83.7807
7	2	5	5	47.91017	9077.714	80.0606
8			10	56.45825	9710.383	81.3666
9		10	5	54.1467	9312.412	84.0823
10			10	59.4117	9682.536	84.7463
11		20	5	51.46321	9998.934	84.171
12			10	58.48993	10228.18	84.256
13	3	5	5	53.74724	10337.12	86.7422

14			10	60.99924	11261.53	87.7938
15		10	5	58.01666	10098.03	87.8238
16			10	61.34418	11549.8	88.0731
17		20	5	54.00448	10492.17	86.2773
18			10	62.02136	12205.62	88.855

Background berwarna hijau merupakan nilai *error* minimal pada setiap kelompok *sliding window*, dan hasil tersebut akan menjadi fokus dalam penerapan GA-LSTM.

2. GA-LSTM

Setelah mendapatkan parameter yang menghasilkan nilai *error* paling minimal pada setiap kelompok. Langkah selanjutnya adalah mengoptimasi unit LSTM. Berikut merupakan grafik perbandingan nilai *error* antara Base LSTM dan GA-LSTM yang menggunakan teknik *cross validation*:



Gambar 4.22 Grafik Perbandingan Base LSTM dan GA LSTM - SGD - CV

Dari ketiga grafik di atas dapat dilihat bahwa mayoritas optimasi menggunakan GA-LSTM tidak berhasil, contohnya ada pada skenario 2, 5, 7, 9, dan 11. Selain itu, meskipun nilai *error* dari skenario yang lain lebih tinggi dari Base LSTM, seperti skenario 13, 15, dan 17, tetapi kenaikan tersebut juga tidak terlalu signifikan. Oleh karena itu, hal tersebut juga bisa disimpulkan bahwa optimasi berhasil. Sementara itu, hanya 1 optimasi yang berhasil dimana itu ditunjukkan oleh skenario 2.

4.6. Hasil Prediksi

Terakhir adalah menguji model untuk memprediksi harga beli mata uang dalam kurun waktu 1 – 20 hari, dengan bantuan GUI yang telah dikembangkan. Pada proses prediksi pengguna akan memilih mata uang apa yang akan di prediksi mulai dari, USD, EUR, dan SGD. Setelah itu, memilih jangka waktu yang akan dipilih dan untuk mengetahui hasil prediksi, pengguna harus menekan tombol ‘Prediksi’. Hasil prediksi akan muncul dalam beberapa saat bersamaan dengan grafik harga beli terdahulu.

Date	Close
01/06/2023 23:58:00	14914
02/06/2023 23:58:00	14901.9
03/06/2023 23:58:00	14901.9
04/06/2023 23:58:00	14990
05/06/2023 23:58:00	14855

Date	Close
01/06/2023 23:58:00	15988
02/06/2023 23:58:00	16127
03/06/2023 23:58:00	15947
04/06/2023 23:58:00	15947
05/06/2023 23:58:00	15950

Date	Close
01/06/2023 23:58:00	11070.24
02/06/2023 23:58:00	11038.11
03/06/2023 23:58:00	11038.11
04/06/2023 23:58:00	11097.09
05/06/2023 23:58:00	11008.96

Gambar 4.23 Hasil Prediksi dengan Data Asli

Gambar sebelah kanan merupakan harga asli dari mata uang USD, EUR, dan SGD 5 hari ke depan. Sedangkan gambar sebelah kiri, merupakan harga prediksi dari model yang paling optimal untuk setiap mata uang. Dari perbandingan gambar di atas menunjukkan bahwa nilai prediksi tidak berbeda jauh dari data yang asli. Sehingga, penulis menyimpulkan bahwa model LSTM dapat dengan baik memprediksi harga beli sebuah mata uang.

BAB V

PENUTUP

5.1. Kesimpulan

Berdasarkan serangkaian proses penelitian dan analisis hasil yang telah dijelaskan pada bab sebelumnya, dapat ditarik beberapa kesimpulan sebagai berikut:

1. Secara umum, teknik *cross validation* menghasilkan nilai *error* yang lebih tinggi dibandingkan teknik *split* data pada model LSTM dasar.
2. Peningkatan jumlah data latih pada teknik *split* data tidak selalu menurunkan nilai *error* model LSTM. Begitu pula dengan penambahan jumlah *fold* pada teknik *cross validation*.
3. Optimasi menggunakan Algoritma Genetika berhasil menurunkan nilai *error* pada beberapa skenario model LSTM, tetapi tidak selalu berhasil untuk semua kasus.
4. Model LSTM mampu memprediksi pergerakan harga beli mata uang asing USD, EUR, dan SGD terhadap IDR dengan cukup akurat untuk jangka pendek.

5.2. Saran

Berdasarkan penelitian yang telah dilakukan, ada beberapa saran yang dapat ditarik untuk pengembangan model secara lebih lanjut, yaitu:

1. Perlu dilakukan percobaan dengan variasi parameter model LSTM yang lebih luas, termasuk jumlah neuron dan aktivasi, untuk mendapatkan performa prediksi yang lebih optimal.
2. Menggunakan *hybrid* model yang menggabungkan LSTM dengan algoritma pembelajaran mesin lainnya juga berpotensi untuk meningkatkan performa prediksi.
3. Membandingkan dengan model prediksi *time series* lainnya seperti ARIMA dan Prophet dapat dilakukan sebagai *benchmark* performa model LSTM.
4. Mempertimbangkan untuk melakukan *feature engineering* terhadap data untuk meningkatkan kemampuan prediksi model.
5. Mempertimbangkan variabel eksternal seperti suku bunga, inflasi, pertumbuhan ekonomi dalam *feature input* model agar prediksi lebih akurat.

DAFTAR PUSTAKA

[1] M. S. Islam dan E. Hossain, "Foreign exchange currency rate prediction using a GRU-LSTM hybrid network," *ELSEVIER,* no. 3, 2021.

[2] A. Kartikadewi, L. A. A. Rosyid dan A. E. Putri, "Prediction of Foreign Currency Exchange (IDR and USD) Using Multiple Linear Regression," *International Journal of Engineering and Techniques,* vol. VI, no. 2, 2020.

[3] N. Lina, L. Yujie, W. Xiao, Z. Jinquan, Y. Jiguo dan Q. Chengming, "Forecasting of Forex Time Series Data Based on Deep Learning," *ELSEVIER,* no. 147, pp. 647-652, 2019.

[4] Z. Hu, Y. Zhao dan M. Khushi, "A Survey of Forex and Stock Price Prediction Using Deep Learning," *Appl. Syst. Innov.,* vol. IV, no. 9, 2021.

[5] M. Yasir, M. Y. Durrani, S. Afzal, M. Maqsood, F. Aadil, I. Mehmood dan S. Rho, "An Intelligent Event-Sentiment-Based Daily Foreign Exchange Rate Forecasting System," *Applied Science,* vol. IX, no. 15, p. 2980, 2019.

[6] Q. Yaxin dan Z. Xue, "Application of LSTM Neural Network in Forecasting Foreign Exchange Price," *Journal of Physics: Conference Series,* vol. 1237, no. 4, 2019.

[7] J. A. Frieden, D. A. Lake dan K. A. Schultz, World Politics: Interests, Interactions, Institutions 4th Edition, New York: W.W. Norton & Company, 2019.

[8] J. Brownlee, Data Preparation for Machine Learning: Data Cleaning, Feature Selection, and Data Transforms in Python, 2020.

[9] A. Burkov, The Hundred-Page Machine Learning Book, 2019.

[10] N. M. Norwawi, "Sliding window time series forecasting with multilayer perceptron and multiregression of COVID-19 outbreak in Malaysia," *ELSEVIER,* pp. 547-564, 2021.

[11] S. Arlot dan A. Celisse, "A survey of cross-Validation procedures for model selection," *Statistics Surveys,* no. 4, pp. 40-79, 2010.

[12] G. Zaccone dan M. R. Karim, Deep Learning with TensorFlow: Explore neural networks and build intelligent systems with Python, 2nd Edition, Birmingham: Packt Publishing, 2018.

[13] O. Kramer, Genetic Algorithm Essentials, Oldenburg: Springer Nature, 2017.

[14] A. V. Tatachar, “Comparative Assessment of Regression Models Based On Model,” *International Research Journal of Engineering and Technology (IRJET),* vol. 08, no. 09, 2021.