PREDIKSI VALUTA ASING MENGGUNAKAN *LONG SHORT-TERM MEMORY* 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

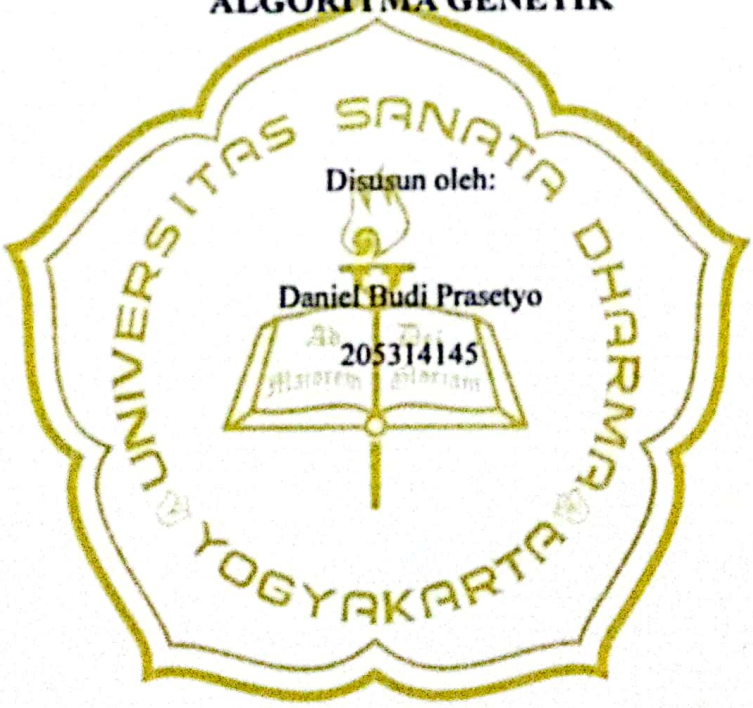
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS SANATA DHARMA
YOGYAKARTA

2024

HALAMAN PERSETUJUAN PEMBIMBING

SKRIPSI

PREDIKSI VALUTA ASING MENGGUNAKAN *LONG SHORT-TERM MEMORY* YANG DIOPTIMALKAN DENGAN ALGORITMA GENETIK



Disusun oleh:

Daniel Budi Prasetyo

205314145

Dosen Pembimbing,

(Cyprianus Kuntoro Adi, S.J. M.A., M.Sc., Ph.D.) 9 Februari 2024

HALAMAN PENGESAHAN

SKRIPSI

PREDIKSI VALUTA ASING MENGGUNAKAN *LONG SHORT-TERM MEMORY* YANG DIOPTIMALKAN DENGAN ALGORITMA GENETIK

Dipersiapkan dan disusun oleh:

Daniel Budi Prasetyo

205314145

Telah dipertahankan di depan Dewan Penguji

Pada tanggal, 19 Januari 2024

Dan dinyatakan memenuhi syarat

Susunan Panitia Penguji

Jabatan		Nama Lengkap	Tanda Tangan
Ketua	:	Ir. Drs. Haris Sriwindono M.Kom, Ph.D.	.....
Sekretaris	:	Eko Hari Parmadi, S.Si., M.Kom.	.....
Anggota	:	Cyprianus Kuntoro Adi, S.J., M.A., M.Sc., Ph.D.	.....

Yogyakarta, 12 - 2 - 2024

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, 12 Februari 2024

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 *LONG SHORT-TERM MEMORY* 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: 12 Februari 2024

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 *Long Short-Term Memory* 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, 12 Februari 2024

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.

Preprocessing pada terdiri atas deteksi *outlier*, normalisasi, *sliding window*, dan pembagian data menggunakan teknik *split* maupun *cross validation*. 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 yang paling optimal untuk memprediksi data USD, EUR, dan SGD terhadap IDR mendapatkan MAE berkisar 41.27, 60.89, dan 13.04 secara berurutan. Tetapi jika untuk memprediksi harga ke depannya model EUR perlu ditingkatkan lagi agar mendapatkan nilai error yang lebih kecil.

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

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.

Preprocessing consists of outlier detection, normalization, sliding window, and data sharing using split and cross validation techniques. After getting the model evaluation value, the model with the minimum error value in the sliding window group will be optimized using a Genetic Algorithm.

In some scenarios, optimization with Genetic Algorithms is successful in reducing error values, although this does not always apply to all cases. The most optimal LSTM model for predicting USD, EUR, and SGD data against IDR gets an MAE of around 41.27, 60.89, and 13.04 respectively. However, if we are to predict future prices, the EUR model needs to be improved further to get a smaller error value.

Keywords: Foreign exchange, Long Short-Term Memory, Genetic Algorithm, USD, EUR, SGD

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

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.
4. Parameter yang dioptimasi adalah jumlah *cell* pada setiap layer LSTM.

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. Deteksi Outlier

Outlier adalah data yang menonjol karena berbeda dari data lainnya [8, 9, 10]. Mereka tidak sering muncul, memiliki keunikan, atau ada beberapa aspek yang membedakannya. Metode statistik dapat digunakan untuk mengidentifikasi outlier, contohnya dengan menggunakan *boxplot*. *Boxplot* adalah metode untuk mendemonstrasikan secara grafis kelompok lokalitas, penyebaran, dan kemiringan data numerik melalui kuartilnya [11]. *Outlier* juga dapat diplot sebagai titik individual di dalam *boxplot*. Berikut merupakan persamaan untuk membentuk sebuah *boxplot*:

$$\boldsymbol{IQR = Q_3 - Q_1} \tag{2.1}$$

$$\boldsymbol{Q_1 = X_{\frac{1}{4}(n+1)}} \tag{2.2}$$

$$\boldsymbol{Q_3 = X_{\frac{3}{4}(n+1)}} \tag{2.3}$$

$$\boldsymbol{Batas\ Atas = Q_3 + 1.5 \times IQR} \tag{2.4}$$

$$\boldsymbol{Batas\ Bawah = Q_1 - 1.5 \times IQR} \tag{2.5}$$

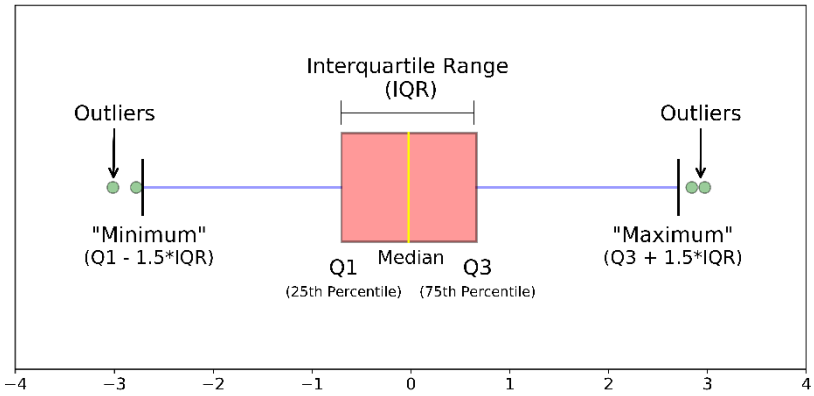
Keterangan:

IQR = Jarak antar kuartil

Q_3 = Kuartil ke-3

Q_1 = Kuartil ke-1

n = Jumlah data



Gambar 2.1 Boxplot (G. Michael, 2017)

2.2.2.2. 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:

$$\bar{\boldsymbol{x}}^{(j)} = \frac{\boldsymbol{x}^{(j)} - \boldsymbol{min}^{(j)}}{\boldsymbol{max}^{(j)} - \boldsymbol{min}^{(j)}} \tag{2.6}$$

Keterangan:

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

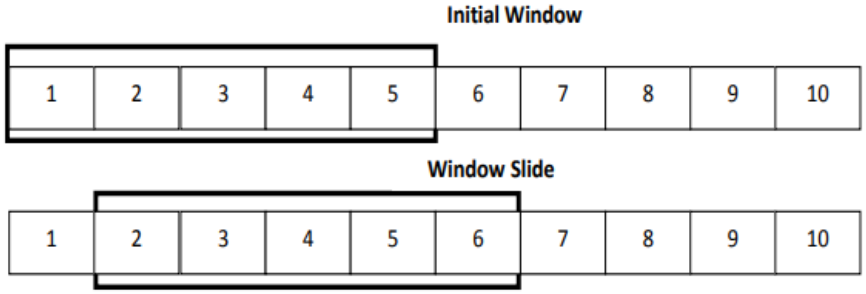
$x^{(j)}$ = Nilai fitur j

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

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

2.2.2.3. Sliding Window

Sliding window merupakan salah satu metode yang dapat digunakan pada tahap *preprocessing* untuk merestrukturisasi data menurut kerangka waktu menjadi masalah klasifikasi [12]. 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 2.2 Proses Sliding Window (H.S. Hota dkk., 2017)

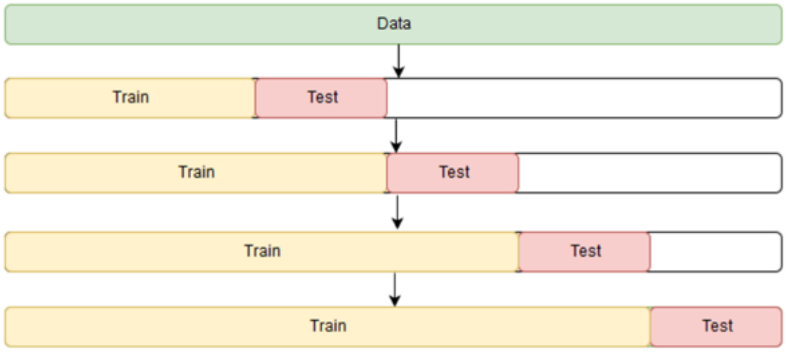
2.2.2.4. Split Data

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].

2.2.2.5. Cross Validation

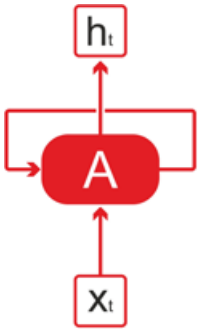
Cross Validation (CV) adalah teknik yang banyak digunakan untuk memilih model atau algoritma terbaik. Konsep intinya melibatkan pembagian data menjadi beberapa *subset* untuk menilai kinerja setiap algoritma [13]. Dalam proses ini, sebagian data digunakan untuk melatih setiap algoritme, sedangkan sisanya disisihkan untuk mengevaluasi seberapa baik kinerja algoritme.



Gambar 2.3 Time Series Cross-Validation (S. Shrivastava, 2020)

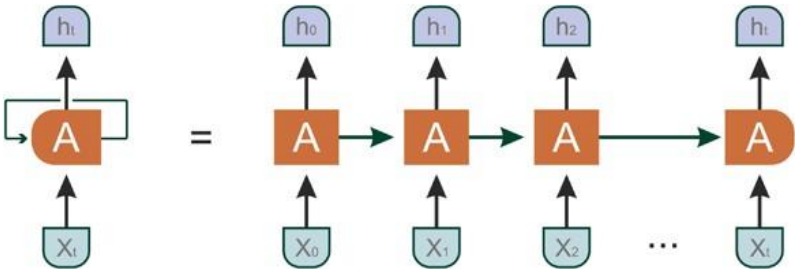
2.2.3. Recurrent Neural Network (RNN)

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 [14].



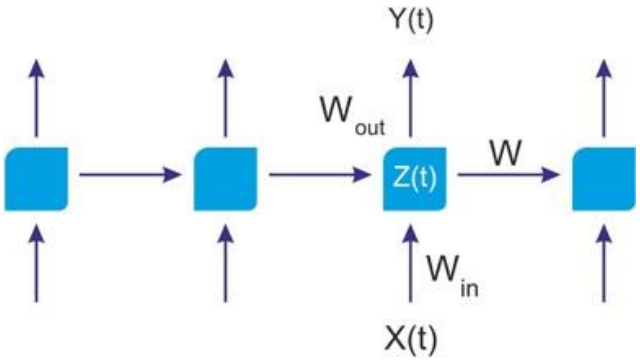
Gambar 2.4 RNN memiliki loop (G. Zaccone dkk., 2018)

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 2.5 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 2.6 RNN menggunakan keadaan jaringan sebelumnya (G. Zaccone dkk., 2018)

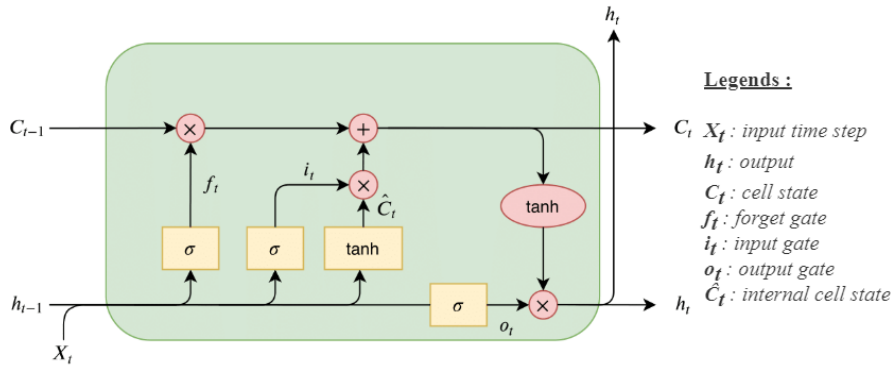
Namun, RNN klasik mengalami keterbatasan tertentu. Salah satu masalah utama adalah masalah gradien yang hilang, yang membuatnya sulit untuk menangkap ketergantungan jangka panjang. Selain itu, mereka kesulitan mempertahankan dan memanfaatkan informasi yang relevan dalam urutan yang panjang. Untuk mengatasi 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 [14], 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*.

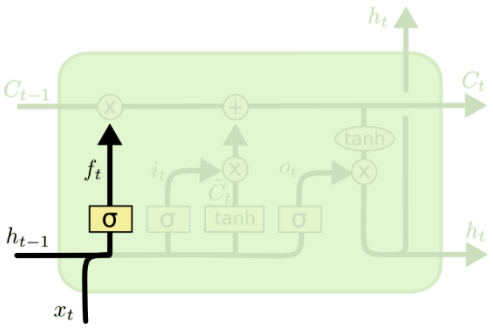
Gerbang ini mengontrol fungsi penulisan, pembacaan, dan pengaturan ulang pada sel memori.



Gambar 2.7 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 2.8 Forget Gate (Colah, 2015)

Persamaan Forget Gate

$$\boldsymbol{f_t = \sigma\left(W_f . [h_{t-1}, x_t] + b_f\right)} \quad (2.7)$$

Keterangan:

f_t = Forget gate

σ = Fungsi aktivasi sigmoid

W_f = Nilai weight forget gate

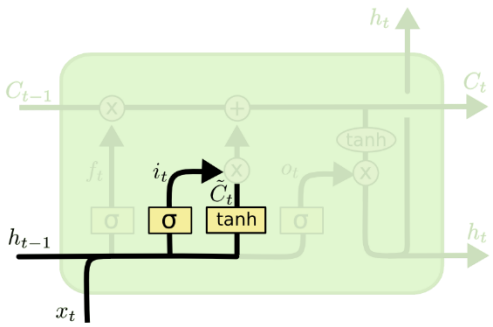
h_{t-1} = Nilai output sebelumnya

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 2.9 Input Gate (Colah, 2015)

Persamaan Input Gate

$$\boldsymbol{i_t = \sigma(W_i . [h_{t-1}, x_t] + b_i)} \quad (2.8)$$

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

$$\boldsymbol{\hat{C}_t = tanh(W_C . [h_{t-1}, x_t] + b_C)} \quad (2.9)$$

Keterangan:

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

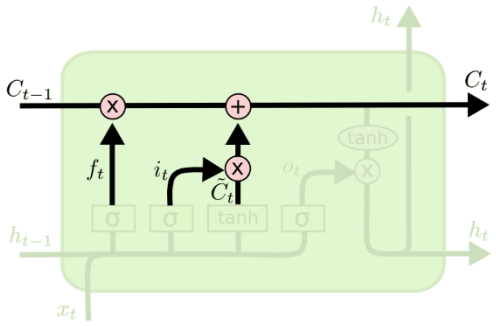
$tanh$ = Fungsi tanh

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 2.10 Cell State (Colah, 2015)

Persamaan Memperbaharui Cell State

$$\boldsymbol{C_t = i_t . \widehat{C}_t + f_t . C_{t-1}} \tag{2.10}$$

Keterangan:

C_t = Cell state

i_t = Input gate

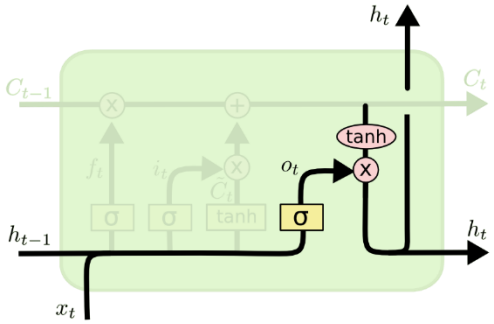
$\hat{C}_t$ = Cell state baru

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 2.11 Output Gate (Colah, 2015)

Persamaan Output Gate

$$\boldsymbol{o_t = \sigma(W_o . [h_{t-1}, x_t] + b_o)} \tag{2.11}$$

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

$$\boldsymbol{h_t = o_t \times tanh(C_t)} \tag{2.12}$$

Keterangan:

h_t = Nilai output

o_t = Output gate

$tanh$ = Fungsi tanh

C_t = Cell state

2.2.5. Fungsi Aktivasi

Untuk memungkinkan jaringan saraf mempelajari batasan keputusan yang kompleks, dibutuhkan fungsi aktivasi nonlinier ke beberapa lapisannya. Fungsi yang umum digunakan antara lain *tanh*, *ReLU*, *softmax*, dan variannya. Pada penelitian ini fungi aktivasi yang digunakan ada 2 jenis yaitu, *sigmoid* dan *tanh*.

2.2.5.1. Sigmoid

Fungsi *sigmoid* adalah fungsi real terdiferensiasi terbatas yang didefinisikan untuk semua nilai masukan nyata dan memiliki turunan non-negatif di setiap titik. Secara umum fungsi *sigmoid* bernilai nyata, monotonik, dan terdiferensiasi, mempunyai turunan pertama non negatif yang berbentuk lonceng. Domain fungsi ini, yang mencakup semua bilangan real dan kodomainnya, adalah (0, 1). Artinya, nilai apa pun yang diperoleh sebagai keluaran dari suatu neuron (sesuai perhitungan status aktivasinya), akan selalu berada di antara 0 dan 1 [14]. Persamaan untuk fungsi aktivasi *sigmoid* adalah sebagai berikut:

$$\boldsymbol{\sigma = \frac{1}{1 + e^{-x}}} \tag{2.13}$$

2.2.5.2. Tanh

Di sisi lain, tangen hiperbolik, atau *tanh*, adalah bentuk lain dari fungsi aktivasi. *Tanh* menekan angka bernilai nyata ke kisaran [-1, 1]. Seperti neuron *sigmoid*, aktivasinya jenuh, tetapi tidak seperti neuron *sigmoid*, keluarannya terpusat pada nol. Oleh karena itu, dalam

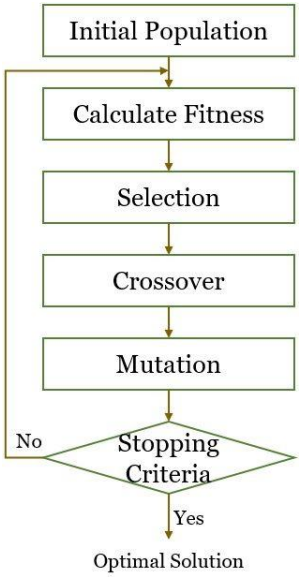
praktiknya, nonlinier *tanh* selalu lebih disukai daripada nonlinier *sigmoid* [14]. Persamaan untuk fungsi aktivasi *tanh* adalah sebagai berikut:

$$\boldsymbol{tanh}(\boldsymbol{x}) = \frac{\boldsymbol{e^{x}} - \boldsymbol{e^{-x}}}{\boldsymbol{e^{x}} + \boldsymbol{e^{-x}}} \tag{2.14}$$

2.2.6. 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 [15].

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 2.12 Langkah Algoritma Genetik (Neha, 2022)

2.2.6.1. 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.

2.2.6.2. Seleksi

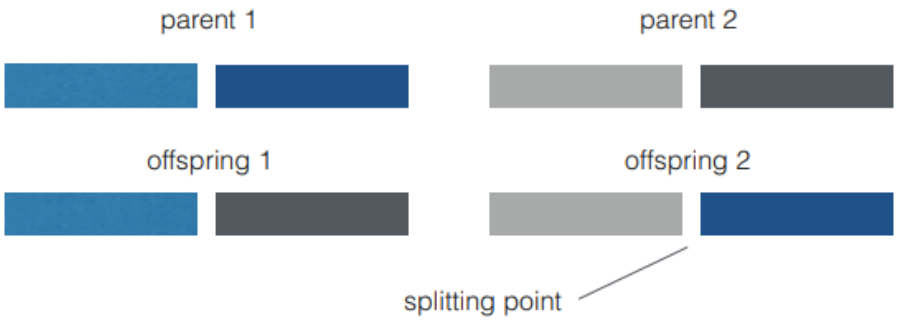
Seleksi adalah operator genetika dalam GA yang memilih solusi-solusi mana yang akan bertahan dan menjadi induk pada generasi baru. Proses seleksi didasarkan pada nilai kebugaran solusi-solusi dalam populasi, di mana solusi-solusi yang lebih baik memiliki peluang yang lebih tinggi untuk dipilih. Terdapat 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 2.13 Tournament Selection (A. Y. Ayoub dkk., 2020)

2.2.6.3. Crossover

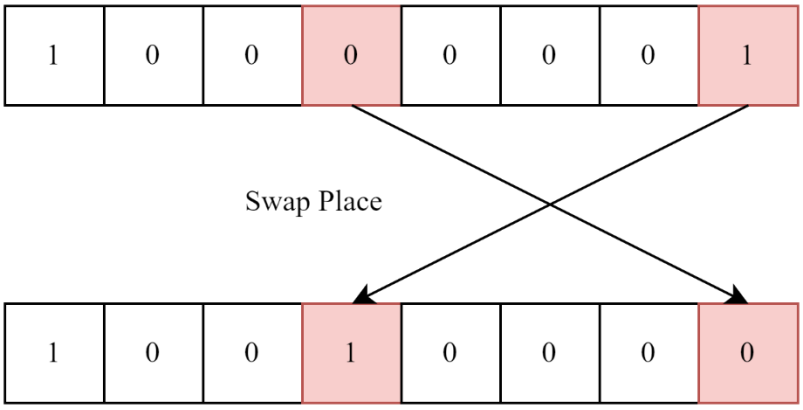
Crossover adalah operator yang memungkinkan kombinasi materi genetik dari dua atau lebih solusi. Ini adalah operator genetik penting dalam GA, yang merupakan optimasi heuristik yang diilhami secara biologis. Operator *crossover* dapat dirancang untuk berbagai jenis representasi solusi, seperti *bit strings*, *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 2.14 Single-point Crossover (Kramer, 2017)

2.2.6.4. 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 2.15 Swap Mutation

2.2.7. 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, dan RMSE.

2.2.7.1. Mean Absolute Error (MAE)

Mean Absolute Error (MAE) adalah rata-rata dari selisih absolut antara nilai yang diamati dan nilai yang diprediksi [16]. 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:

$$\boldsymbol{MAE} = \frac{\mathbf{1}}{\boldsymbol{n}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} |\boldsymbol{\hat{y}_i} - \boldsymbol{y_i}| \tag{2.15}$$

Keterangan:

MAE = Mean Absolute Error

n = Jumlah data

$\hat{y}_i$ = Nilai prediksi

y_i = Nilai aktual

2.2.7.2. 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 [16]. 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:

$$\boldsymbol{MSE} = \frac{\mathbf{1}}{\boldsymbol{n}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}}(\boldsymbol{\hat{y}_i} - \boldsymbol{y_i})^{\mathbf{2}} \tag{2.16}$$

Keterangan:

MSE = Mean Square Error

n = Jumlah data

$\hat{y}_i$ = Nilai prediksi

y_i = Nilai aktual

2.2.7.3. Root Mean Squared Error (RMSE)

Root Mean Squared Error (RMSE) adalah akar kuadrat dari rata-rata kuadrat dari semua *error* [16]. 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:

$$\boldsymbol{RMSE} = \sqrt{\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \frac{(\boldsymbol{\hat{y}_i - y_i})^{\boldsymbol{2}}}{\boldsymbol{n}}} = \sqrt{\boldsymbol{MSE}} \qquad (2.17)$$

Keterangan:

RMSE = Root Mean Squared Error

n = Jumlah data

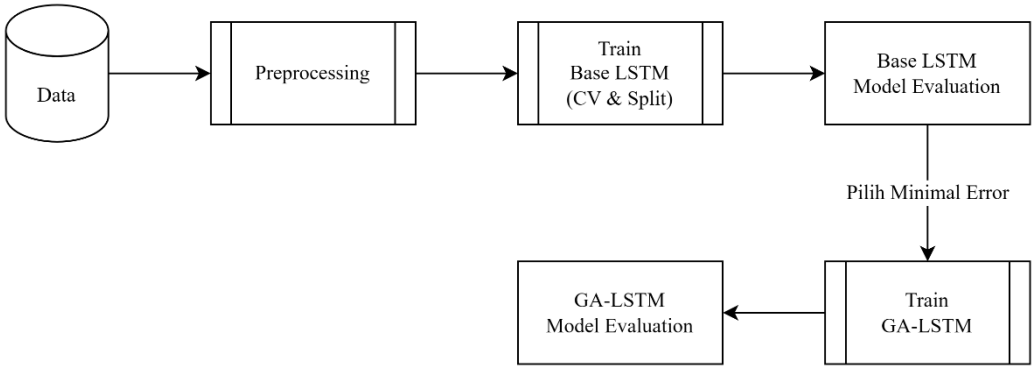
$\hat{y}_i$ = Nilai prediksi

y_i = Nilai aktual

MSE = Mean Squared Error

BAB III

METODE PENELITIAN



Gambar 3.1 Langkah Penelitian

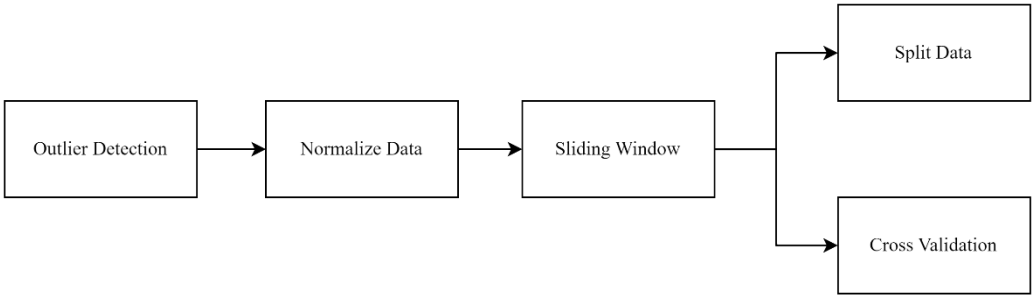
3.1. Deskripsi Data

Data yang akan peneliti gunakan untuk penelitian ini adalah data penutupan 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, EUR, dan SGD. Dimana tiap – tiap data tersebut hanya memiliki 2 atribut, yaitu tanggal dan harga beli setelah penutupan. 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. Deteksi Outlier

Langkah pertama dalam *preprocessing* data valuta asing adalah mendeteksi *outlier*. Hal ini digunakan untuk mengetahui apakah data yang diambil sudah benar – benar bersih. Untuk mempermudah mengetahui apakah data tersebut sudah bersih dari *outlier* maka digunakanlah *boxplot* seperti pada gambar 2.1. Setelah menemukan *outlier* maka nilai tersebut akan diubah menggunakan batas atas dan batas bawah dari sebuah *boxplot* yang didapatkan dari persamaan 2.4 dan 2.5.

3.2.2. Normalisasi Data

Langkah selanjutnya adalah normalisasi data. Normalisasi yang akan digunakan adalah normalisasi *min-max*. Dimana tiap – tiap fitur valuta asing tersebut di normalisasi menggunakan persamaan 2.6 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

3.2.3. 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.

3.2.4. 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*.

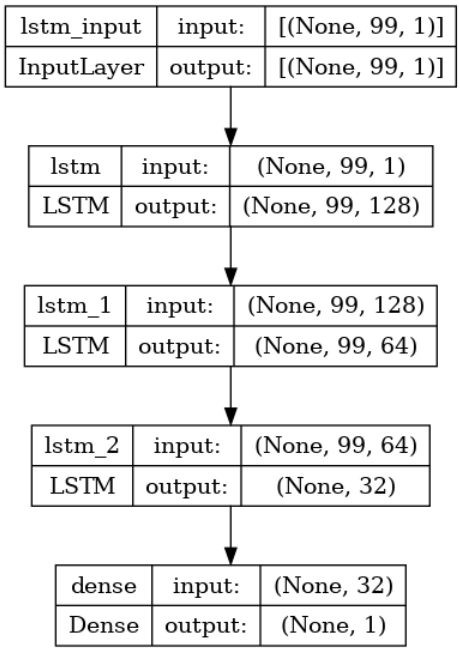
3.2.5. Cross Validation

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

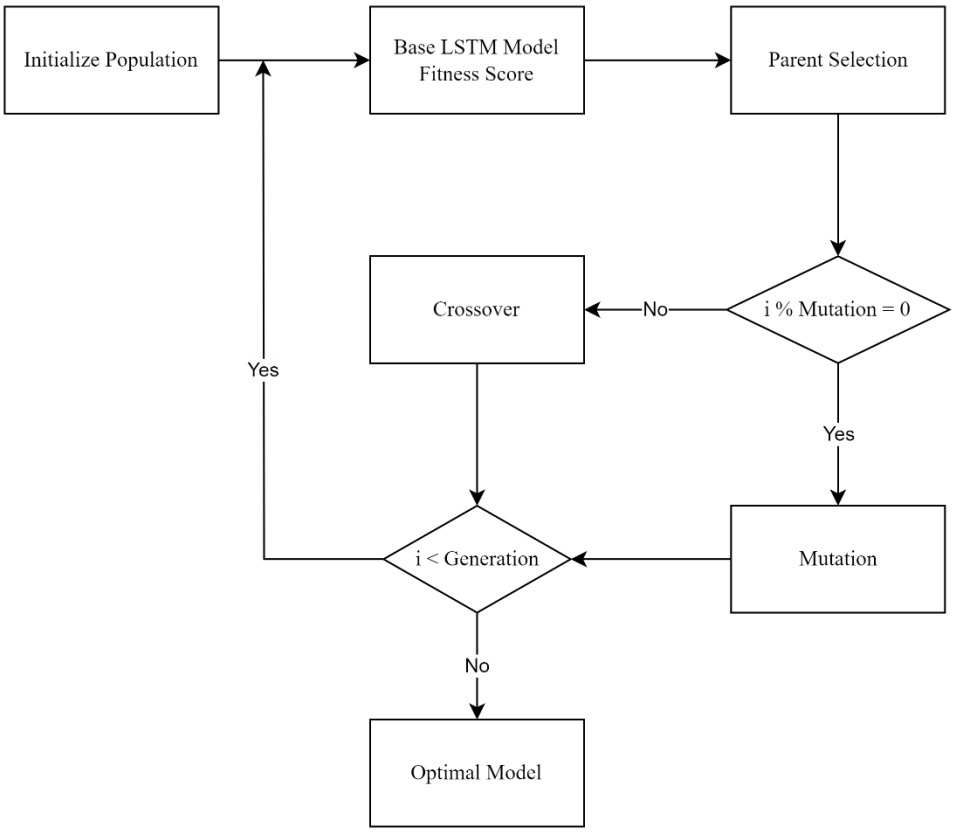
3.3.1. Base LSTM



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. Dimana parameter yang akan dioptimalkan adalah jumlah neuron atau *cell* untuk setiap *layer* LSTM. Arsitektur model yang digunakan juga sama seperti gambar 3.3, dimana nanti 2 layer LSTM terakhir akan dibuka secara bergantian saat percobaan.

a. Inisiasi Populasi

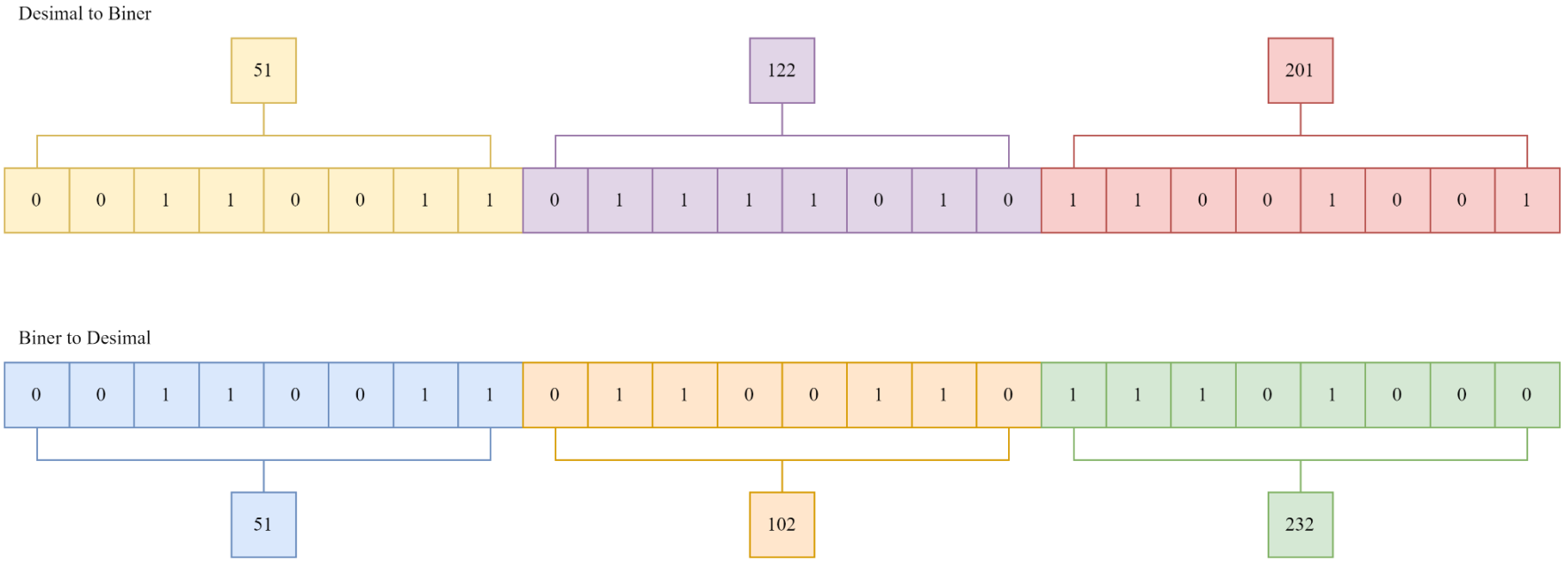
Langkah pertama dalam proses ini adalah menginisiasi populasi. Populasi ini dihasilkan secara acak dan memiliki struktur yang spesifik. Struktur ini berbentuk *list* 2 dimensi, yang memiliki karakteristik tertentu. Panjang baris dalam *list* ini adalah 10. Ini menunjukkan jumlah kromosom dalam populasi. Kromosom adalah unit dasar dari informasi genetik dan dalam konteks ini, mereka mewakili solusi potensial untuk masalah jumlah *cell* dalam layer LSTM. Dengan kata lain, setiap kromosom adalah satu kemungkinan solusi.

Selanjutnya, ada 3 kolom dalam *list* ini. Jumlah kolom ini menentukan jumlah *cell* dari layer LSTM. Dimana jumlah *cell* tersebut memiliki rentang 1 – 250. Jadi, setiap elemen dalam *list* 2 dimensi ini mewakili jumlah *cell* sebuah *layer* LSTM dalam satu kromosom. Dengan demikian, struktur *list* ini mencerminkan struktur dari solusi potensial yang kita cari, yaitu sebuah jaringan LSTM dengan jumlah *cell* tertentu. Berikut adalah contoh bagaimana populasi dan bentuk kromosom mungkin tampak pada inisiasi awal:

Kromosom 1	115	72	225
Kromosom 2	120	61	3
Kromosom 3	102	147	35
Kromosom 4	170	95	29
Kromosom 5	238	208	29
Kromosom 6	222	90	201
Kromosom 7	51	102	232
Kromosom 8	49	31	26
Kromosom 9	50	126	181
Kromosom 10	169	227	80

Gambar 3.5 Bentuk Populasi dan Kromosom GA-LSTM

Sebelum dan sesudah menjalankan *crossover* maupun mutasi, bentuk kromosom yang mulanya *list* angka desimal akan diubah menjadi biner 24 bit dan sebaliknya. Berikut merupakan contoh kromosom yang diubah dari desimal menjadi biner dan sebaliknya:



Gambar 3.6 Perubahan Data dari Desimal ke Biner dan Sebaliknya

b. Menghitung Fitness

Langkah kedua dalam proses ini adalah menghitung nilai *fitness*. Dalam konteks ini, nilai *fitness* dihitung setiap kali model dilatih dengan jumlah *cell* yang ada pada sebuah kromosom. Yang mulanya jumlah cell-nya adalah 128, 64, dan 32, akan diubah berdasarkan elemen yang ada di kromosom.

Selama pelatihan, model LSTM belajar mengenali pola dari waktu ke waktu, menyimpan informasi penting, dan melupakan informasi yang tidak relevan. Proses pelatihan melibatkan pemberian data masukan (x) dan data keluaran yang sebenarnya atau (y) kepada model. Model membuat prediksi ($\hat{y}$) berdasarkan data masukan dan prediksi tersebut dibandingkan dengan data keluaran sebenarnya. Perbedaan tersebut dihitung menggunakan metrik MSE dengan persamaan 2.16 dan hasilnya digunakan sebagai nilai *fitness* untuk kromosom.

Setelah nilai *fitness* dihitung untuk setiap kromosom dalam populasi, nilai-nilai ini dapat digunakan untuk memandu proses optimasi kita. Kromosom dengan nilai *fitness* yang lebih tinggi yaitu, mereka yang menghasilkan MSE yang lebih rendah saat model dilatih dan dievaluasi, akan dianggap sebagai solusi yang lebih baik untuk masalah ini. Berikut adalah contoh bagaimana populasi mungkin tampak setelah nilai *fitness* telah dihitung:

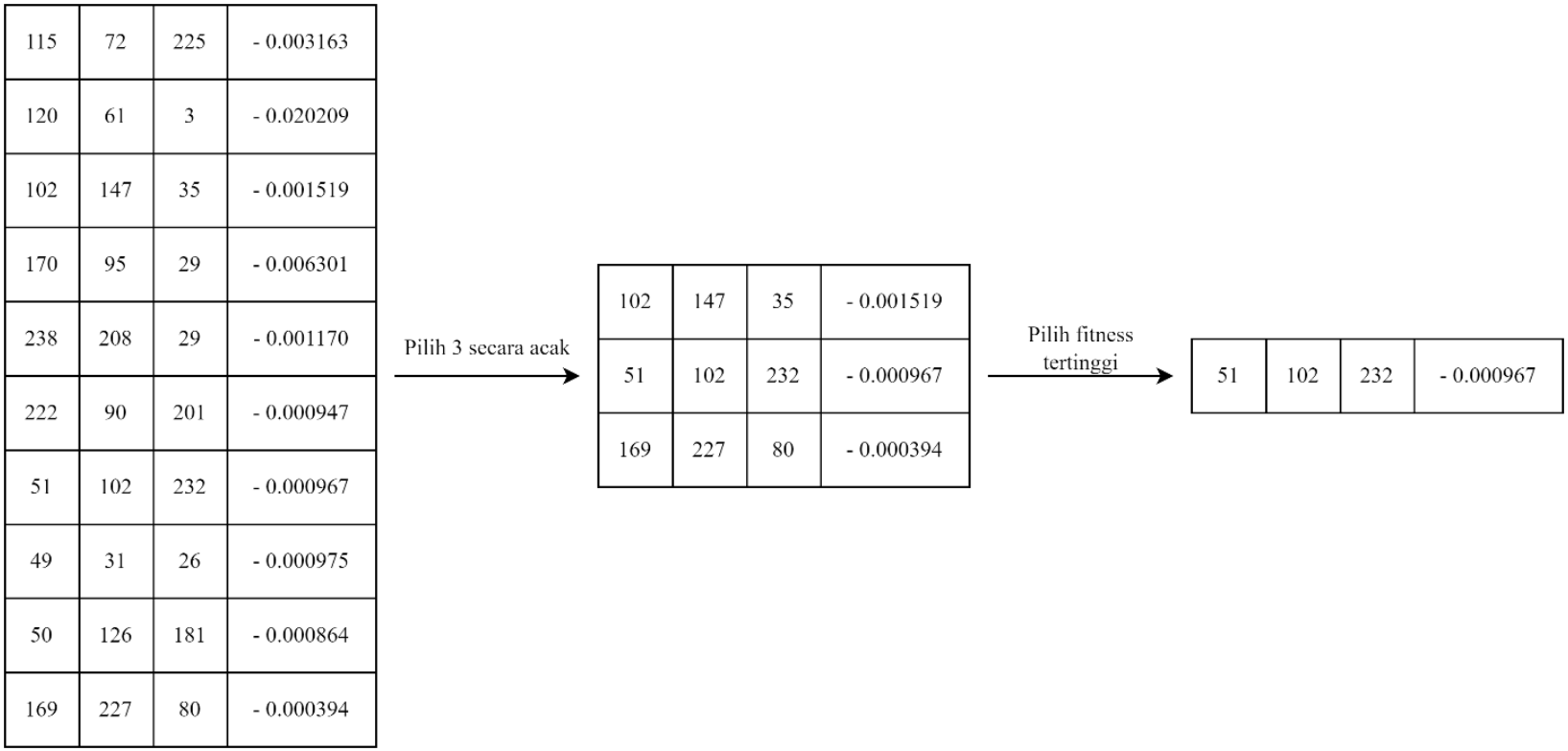
115	72	225	- 0.003163
120	61	3	- 0.020209
102	147	35	- 0.001519
170	95	29	- 0.006301
238	208	29	- 0.001170
222	90	201	- 0.000947
51	102	232	- 0.000967
49	31	26	- 0.000975
50	126	181	- 0.000864
169	227	80	- 0.000394

Gambar 3.7 Populasi dan Nilai Fitness

c. Seleksi

Langkah ketiga dalam proses ini adalah seleksi kromosom yang akan digunakan sebagai induk dalam operasi *crossover* atau mutasi. Metode seleksi yang digunakan dalam penelitian ini adalah *tournament selection*, yang merupakan metode populer dalam algoritma genetik karena efisiensinya dan kemampuannya untuk menjaga variasi dalam populasi. Berikut adalah langkah-langkah detail dalam proses seleksi menggunakan *tournament selection*:

- Pilih kromosom: Dari populasi yang terdiri dari 10 kromosom, pilih 3 kromosom secara acak. Tiga kromosom ini akan membentuk ‘turnamen’ mini, di mana mereka akan bersaing satu sama lain berdasarkan nilai *fitness* mereka.
- Pilih induk: Dari ketiga kromosom ini, pilih kromosom dengan nilai *fitness* tertinggi. Kromosom ini akan menjadi ‘pemenang’ turnamen dan akan dipilih sebagai induk untuk operasi *crossover* ataupun mutasi.



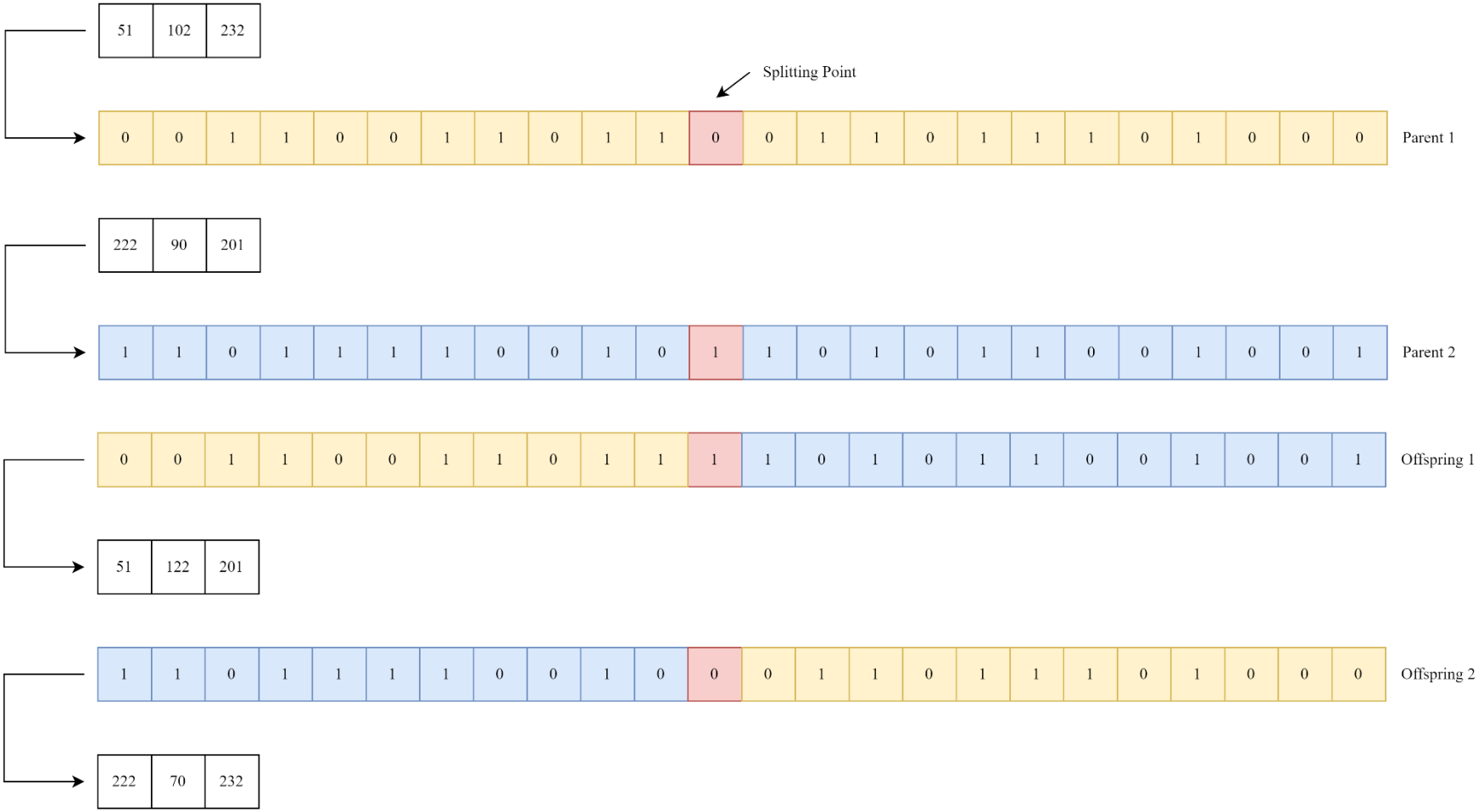
Gambar 3.8 Tournament Selection

Dengan melakukan seleksi dengan cara ini, memastikan bahwa kromosom dengan kinerja baik memiliki peluang lebih besar untuk dipilih sebagai induk dan mewariskan karakteristik mereka ke generasi berikutnya. Namun, karena kromosom dipilih secara acak untuk turnamen, ada juga peluang bagi kromosom dengan kinerja lebih rendah untuk dipilih sebagai induk, yang membantu menjaga variasi dalam populasi dan mencegah konvergensi prematur ke solusi sub-optimal.

d. *Crossover*

Langkah keempat dalam proses ini adalah melakukan operasi *crossover* pada dua kromosom induk yang telah dipilih melalui proses seleksi. Tipe *crossover* yang digunakan dalam penelitian ini adalah *single-point crossover.* Berikut adalah langkah-langkah detail untuk proses *crossover*:

- Konversi format: Ubah data kromosom dari format desimal ke format biner.
- Pilih titik pemisahan: Pilih indeks acak dari kedua kromosom sebagai titik pemisahan.
- Pertukaran informasi genetik: Tukar bagian kromosom setelah titik pemisahan antara kedua kromosom, untuk membentuk kromosom baru.
- Konversi kembali: Ubah kromosom baru yang telah dibentuk kembali ke format desimal.



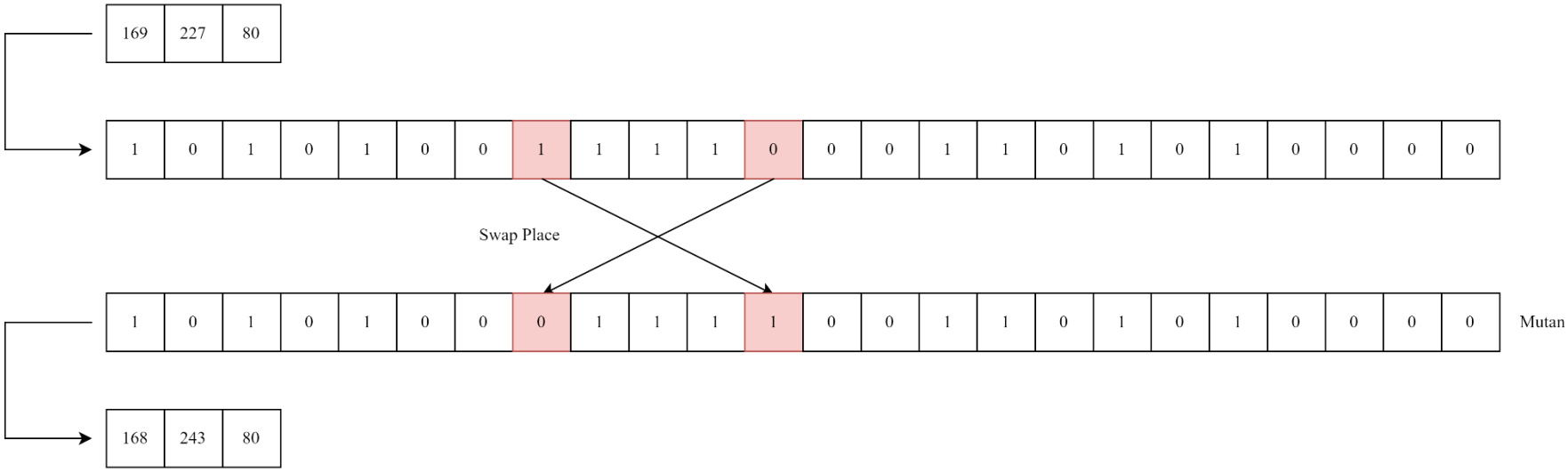
Gambar 3.9 Langkah Single-Point Crossover

Dengan demikian, proses *crossover* menghasilkan dua kromosom baru yang memiliki kombinasi informasi genetik dari kedua kromosom induk. Proses ini penting dalam Algoritma Genetika karena memungkinkan eksplorasi ruang pencarian yang lebih luas dan membantu dalam mencapai solusi optimal.

e. Mutasi

Langkah kelima dalam proses ini adalah melakukan operasi mutasi pada kromosom yang telah dihasilkan melalui proses seleksi. Tipe mutasi yang digunakan dalam penelitian ini adalah *swap mutation*. Berikut adalah langkah-langkah detail untuk proses mutasi:

- Konversi format: Ubah data kromosom dari format desimal ke format biner.
- Pilih dua indeks acak dan tukar nilainya: Pilih dua indeks acak dari kromosom dan tukar nilai pada indeks yang dipilih.
- Konversi kembali: Ubah kromosom baru yang telah dibentuk kembali ke format desimal.



Gambar 3.10 Langkah Swap Mutation

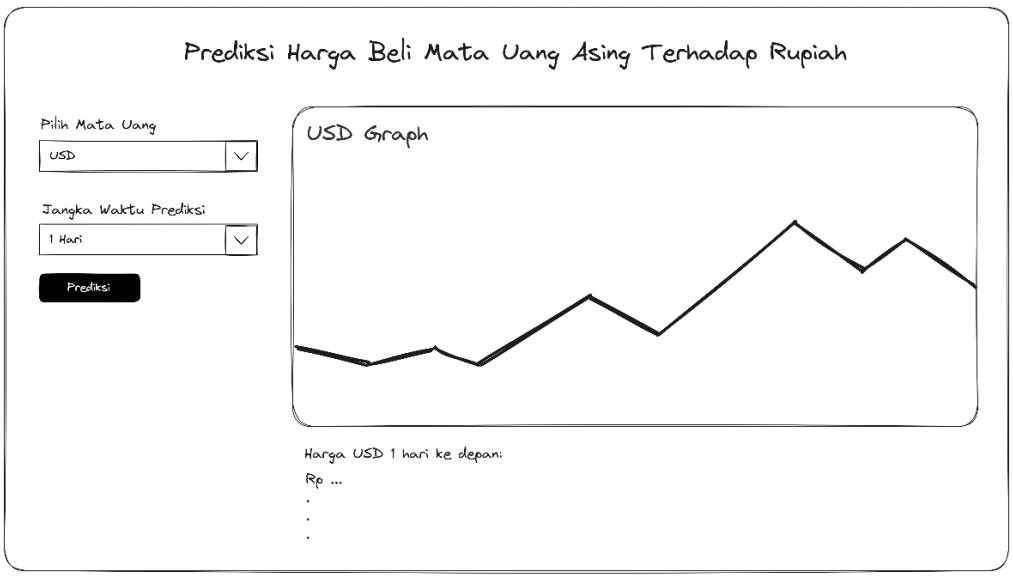
Mutasi ini tidak dilakukan setiap generasi, tetapi pada interval tertentu atau pada generasi tertentu. Ini untuk memastikan bahwa mutasi tidak terjadi terlalu sering, yang dapat menyebabkan populasi menjadi terlalu acak dan menghambat konvergensi algoritma. Dengan demikian, proses mutasi membantu menjaga keragaman dalam populasi dan memungkinkan Algoritma Genetika untuk menemukan solusi yang lebih baik dan lebih optimal.

Terakhir, langkah b hingga e dalam Algoritma Genetik akan diulangi secara berkelanjutan. Proses ini akan berlangsung selama jumlah generasi yang telah ditentukan oleh peneliti. Setelah semua generasi telah dijalankan, Algoritma Genetik akan menghasilkan solusi terbaik dari semua generasi. Solusi ini adalah jumlah sel yang paling optimal yang akan digunakan dalam *layer* LSTM pada model.

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 2.15, 2.16, dan 2.17.

3.5. Desain User Interface



Gambar 3.11 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 *error* paling 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

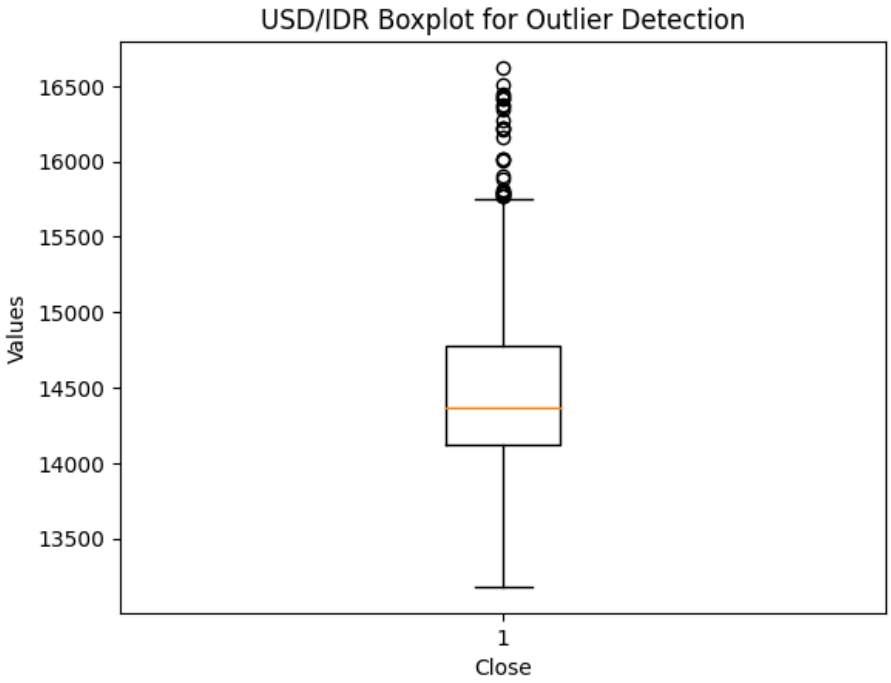
Gambar 4.1. Contoh Data Harga Beli Setiap Mata Uang

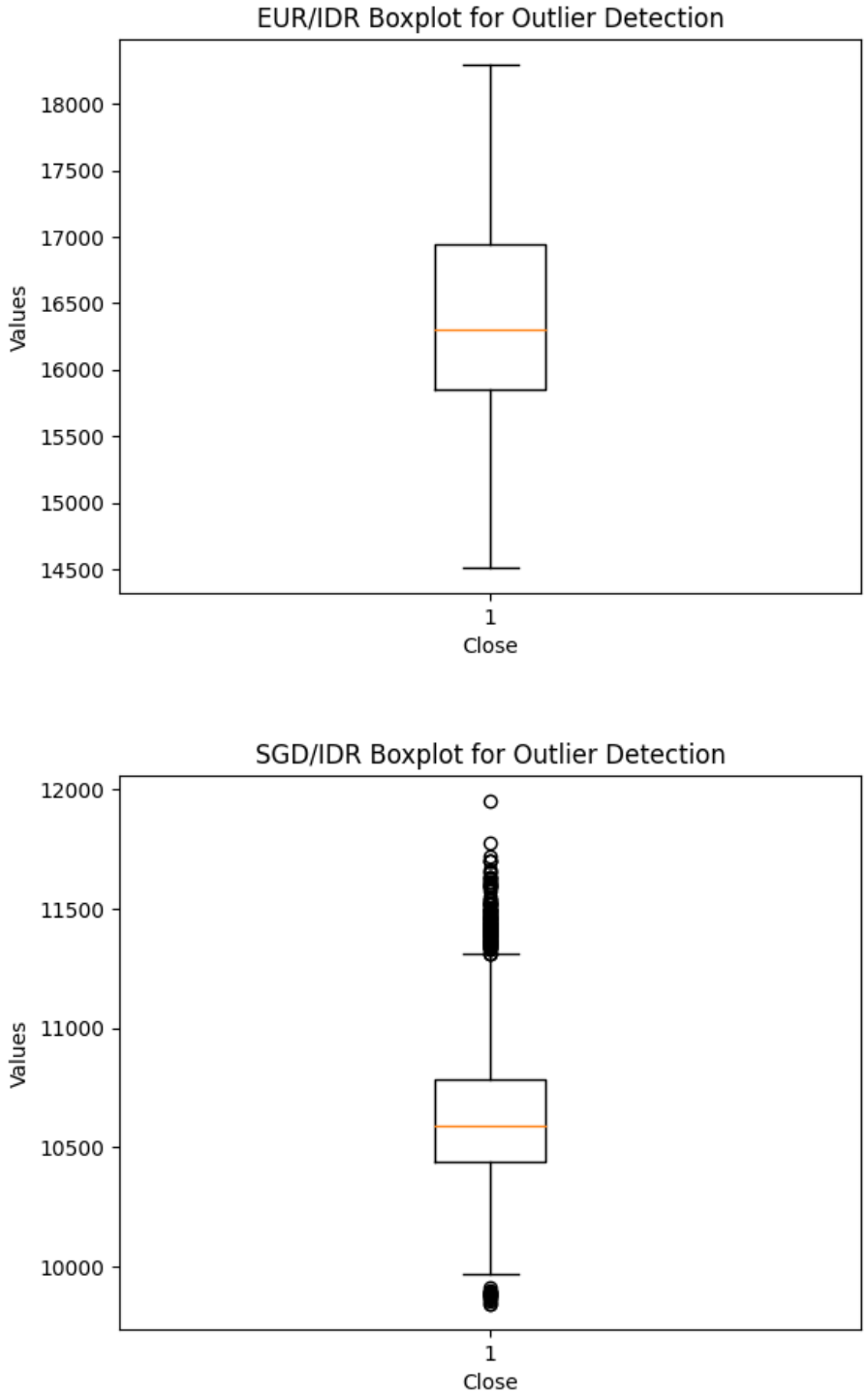
Untuk mempermudah pembacaan data, nilai tukar setiap mata uang telah diatur dalam *sheet* yang terpisah, dan agar dapat diakses melalui library pandas, *spreadsheet* harus dibuka terlebih dahulu untuk mendapatkan *link* yang diperlukan.

4.2. Preprocessing

4.2.1. Deteksi Outlier

Tahapan pertama dalam *preprocessing* adalah deteksi *outlier*. Pada tahapan ini data akan dilihat apakah memiliki data sebuah *outlier* atau tidak. Berikut merupakan *boxplot* untuk data sebelum *outlier* diganti:





Gambar 4.2 Boxplot untuk Mendeteksi Outlier

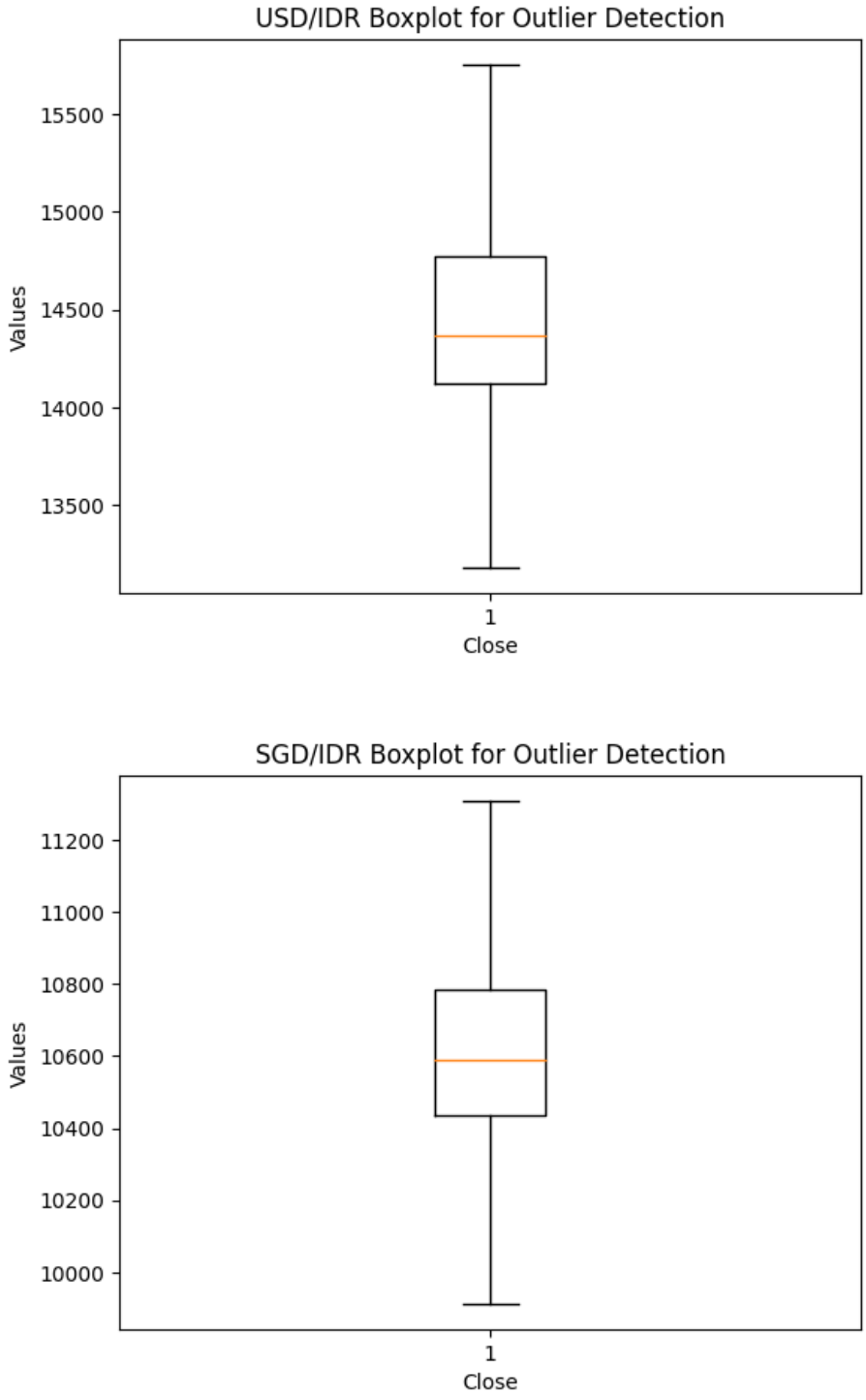
Dari gambar dapat diperhatikan bahwa data USD dan SGD memiliki *outlier*. Selanjutnya *outlier* – *outlier* tersebut akan diganti

menggunakan batas atas dan batas bawah dari data. Berikut merupakan implementasi kode untuk mengubah nilai *outlier*:

```
def replace_outliers(data):
    Q1 = np.percentile(data, 25)
    Q3 = np.percentile(data, 75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    data[data < lower_bound] = lower_bound
    data[data > upper_bound] = upper_bound
    return data
```

Gambar 4.3 Source Code Mengganti Nilai Outlier

Fungsi ini menerima *input* data. Pertama, fungsi akan mencari nilai kuartil 1 dan 3 dengan menggunakan fungsi persentil yang telah disediakan. Selanjutnya menghitung nilai IQR, batas bawah, dan batas atas dengan persamaan 2.1, 2.4, dan 2.5. Terakhir mengubah *outlier* dengan nilai batas atas atau batas bawah, dan akhirnya fungsi ini mengembalikan data sebagai nilai kembaliannya. Berikut gambar *boxplot* setelah pengubahan *outlier*:



Gambar 4.4 Boxplot Setelah Mengubah Nilai Outlier

4.2.2. Normalisasi

Tahapan selanjutnya 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 4.5 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 4.6 Contoh Data Hasil Normalisasi

4.2.3. 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 4.7 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 4.8 Bentuk Data Hasil Sliding Window

4.2.4. 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 4.9 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 4.10 Bentuk Data Setelah Split

4.2.5. 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:

```
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 4.11 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 4.12 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 4.13 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 4.14 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(generations, mutation_rate):
    print(f"Inizialized Population...")
    population = np.random.randint(1, high=251, size=(10, 3))
    fitness_scores = np.array([fitness_function(chromosome) for chromosome in population])
    max_fitness_each_gen = []

    for generation in range(generations):
        print(f"Generation - {generation + 1}")
        max_fitness_each_gen.append(np.max(fitness_scores))

        chromosome_1 = selection(population, fitness_scores)

        if generation % mutation_rate == 0:
            chromosome_1 = mutate(chromosome_1)
            fitness_score_1 = fitness_function(chromosome_1)
            fitness_score_2 = None
        else:
            chromosome_2 = selection(population, fitness_scores)

            chromosome_1, chromosome_2 = crossover(chromosome_1, chromosome_2)

            fitness_score_1 = fitness_function(chromosome_1)
            fitness_score_2 = fitness_function(chromosome_2)
```

```
        worst_index = np.argmin(fitness_scores)
        worst_fitness = fitness_scores[worst_index]

        if fitness_score_1 > worst_fitness:
            population[worst_index] = chromosome_1
            fitness_scores[worst_index] = fitness_score_1
            worst_index = np.argmin(fitness_scores)
            worst_fitness = fitness_scores[worst_index]

        if fitness_score_2 is not None and fitness_score_2 > worst_fitness:
            population[worst_index] = chromosome_2
            fitness_scores[worst_index] = fitness_score_2

        # Release some memory
        del chromosome_1, chromosome_2
        del fitness_score_1, fitness_score_2
        del worst_index, worst_fitness

    best_chromosome = population[np.argmax(fitness_scores)]
    best_fitness = np.max(fitness_scores)

    return best_chromosome, best_fitness, max_fitness_each_gen
```

Gambar 4.15 Source Code Algoritma Genetik

Fungsi ini menerima dua input, yaitu *generations* yang menentukan berapa banyak generasi yang harus dijalankan oleh algoritma dan *mutation_rate*, yang menentukan pada kelipatan berapa mutasi akan dijalankan. 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*. Skor *fitness* disimpan dalam sebuah *list* untuk masing – masing kromosom. Berikut merupakan implementasi kode *fitness function*:

```
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 4.16 Source Code Fitness Function

Populasi baru kemudian dibuat dengan cara memilih kromosom dari populasi saat ini menggunakan fungsi *selection*, melakukan *crossover* pada dua kromosom untuk menghasilkan dua keturunan baru, dan pada kelipatan ke-5 akan melakukan mutasi pada salah satu kromosom. Fungsi

selection akan memilih 3 kromosom secara acak, yang dimana yang akan lulus seleksi adalah kromosom yang memiliki skor *fitness* tertinggi. Berikut merupakan implementasi kode *tournament selection*:

```
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 4.17 Source Code Tournament Selection

Pertama kali menjalankan fungsi *crossover* adalah mengubah kedua kromosom induk menjadi *string* biner. Setelah itu, 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 dalam bentuk biner sebagai hasil dari persilangan dua kromosom induk. Terakhir, ubah hasil persilangan menjadi *list* 3 angka desimal. Berikut merupakan implementasi kode *single-point crossover*:

```
def crossover(parent_1: list, parent_2: list):
    parent_1 = decimal_to_binary(parent_1)
    parent_2 = decimal_to_binary(parent_2)

    crossover_point = np.random.randint(1, len(parent_1))
    offspring_1 = parent_1[:crossover_point] + parent_2[crossover_point:]
    offspring_2 = parent_2[:crossover_point] + parent_1[crossover_point:]

    offspring_1 = binary_to_decimal(offspring_1)
    offspring_2 = binary_to_decimal(offspring_2)
    return offspring_1, offspring_2
```

Gambar 4.18 Source Code Single-Point Crossover

Hampir sama dengan *crossover*, pertama ubah kromosom menjadi bentuk *string* biner, dan ubah lagi menjadi *list* biner. Selanjutnya, ambil 2 titik secara acak dan tukar posisi kedua angka biner. Terakhir, ubah hasil mutasi menjadi *list* 3 angka desimal. Berikut merupakan implementasi kode *swap mutation*:

```
def mutate(chromosome: list):
    chromosome = decimal_to_binary(chromosome)

    chromosome_list = list(chromosome)
    i = np.random.randint(len(chromosome_list), size=2)
    chromosome_list[i[0]], chromosome_list[i[1]] = chromosome_list[i[1]], chromosome_list[i[0]]

    chromosome_list = ''.join(chromosome_list)
    chromosome = binary_to_decimal(chromosome_list)
    return chromosome
```

Gambar 4.19 Source Code Swap Mutation

Akhirnya, kromosom lama akan diperbarui dengan kromosom baru yang memiliki nilai *fitness* lebih baik, dan proses ini diulang untuk 50 generasi. Fungsi kemudian mengembalikan kromosom terbaik, *fitness* terbaik, dan *fitness* terbaik tiap generasi. Setelah mendapatkan kromosom terbaik, kromosom tersebut digunakan ke dalam pembuatan model LSTM yang optimal.

```
def decimal_to_binary(chromosome: list):
    binary_string = ''.join([bin(x)[2:].zfill(8) for x in chromosome])
    return binary_string

def binary_to_decimal(binary_string: str):
    decimal_array = [int(binary, 2) for binary in [binary_string[i:i+8] for i in range(0, len(binary_string), 8)]]
    return decimal_array
```

Gambar 4.20 Source Code Desimal ke Biner dan Sebaliknya

Kedua kode di atas merupakan kode tambahan untuk mengubah *list* angka desimal menjadi *string* biner atau sebaliknya.

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:

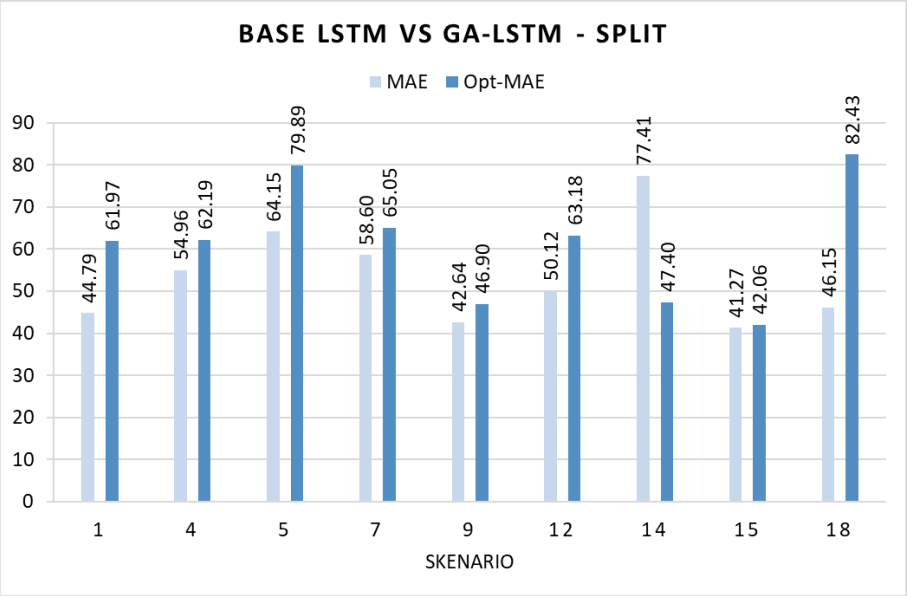
Tabel 4.1 Tabel Hasil Pengujian Base LSTM - USD - Split

No.	LSTM Layer	Sliding Window	Train Split	MAE
1	1	5	0.8	44.78792
2			0.9	50.12943
3		10	0.8	64.29817
4			0.9	54.95588
5		20	0.8	64.14593
6			0.9	80.0486
7	2	5	0.8	58.59934
8			0.9	89.17379
9		10	0.8	42.64078
10			0.9	55.12722
11		20	0.8	85.66703
12			0.9	50.12015
13	3	5	0.8	120.5928
14			0.9	77.41299
15		10	0.8	41.27147

16			0.9	69.60965
17		20	0.8	48.52497
18			0.9	46.15075

2. GA-LSTM

Setelah mendapatkan parameter yang menghasilkan nilai *error* paling minimal pada setiap kelompok. Langkah selanjutnya adalah mengoptimasi jumlah 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 - USD - Split

Dari grafik di atas dapat dilihat bahwa mayoritas optimasi menggunakan GA-LSTM tidak berhasil. Hanya 1 saja yang berhasil sepenuhnya yaitu pada skenario nomor 14. Dimana hasil yang didapatkan dari GA-LSTM lebih rendah daripada Base LSTM dibandingkan dengan yang lain. Selain itu, meskipun nilai *error* dari beberapa skenario yang lain

lebih tinggi, tetapi kenaikan tersebut tidak terlalu jauh. Oleh karena itu, hal tersebut juga bisa disimpulkan bahwa optimasi berhasil.

b. Pengujian Menggunakan Teknik Cross Validation

1. Base LSTM

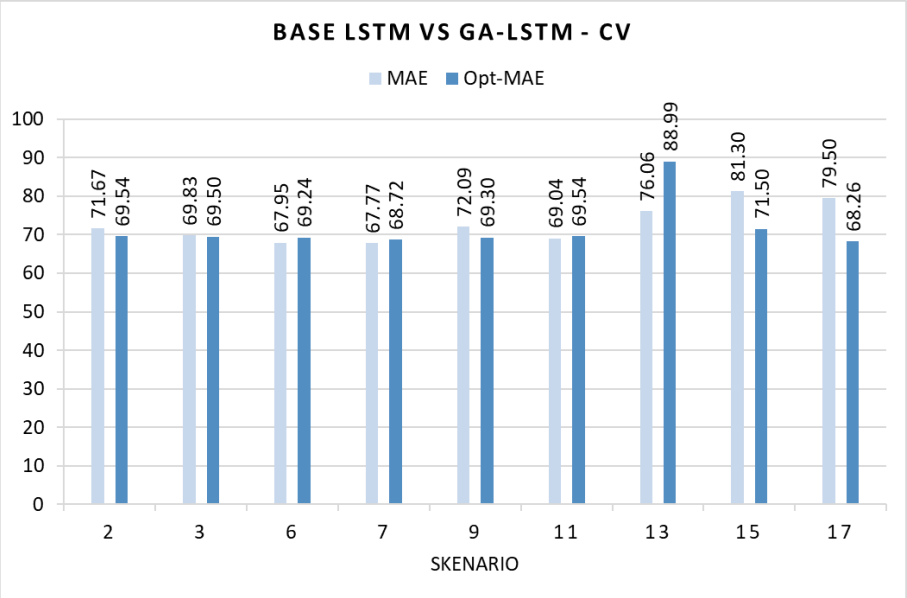
Data mata uang USD akan dibagi menggunakan teknik *cross validation*, dan dengan variasi *fold*, yaitu 5 dan 10. Hal tersebut digunakan untuk mengetahui bagaimana jumlah *fold* mempengaruhi *error* sebuah model Base LSTM. Berikut merupakan tabel hasil pengujian menggunakan Base LSTM:

Tabel 4.2 Tabel Hasil Pengujian Base LSTM - USD – CV

No.	LSTM Layer	Sliding Window	Fold	MAE
1	1	5	5	78.50908
2			10	71.66565
3		10	5	69.83034
4			10	71.49938
5		20	5	78.69399
6			10	67.95476
7	2	5	5	67.76532
8			10	78.14818
9		10	5	72.09135
10			10	75.55815
11		20	5	69.04162
12			10	71.74248
13	3	5	5	76.0621
14			10	83.46939
15		10	5	81.30345
16			10	91.08675
17		20	5	79.49773
18			10	81.53576

2. GA-LSTM

Setelah mendapatkan parameter yang menghasilkan nilai *error* paling minimal pada setiap kelompok. Langkah selanjutnya adalah mengoptimasi jumlah 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 - USD - CV

Dari grafik di atas dapat dilihat bahwa mayoritas optimasi menggunakan GA-LSTM berhasil. Dimana contohnya terdapat pada skenario 2, 3, 9, 15, dan 17. Selain itu, meskipun nilai *error* dari beberapa skenario yang lain lebih tinggi, tetapi kenaikan tersebut tidak terlalu jauh. 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:

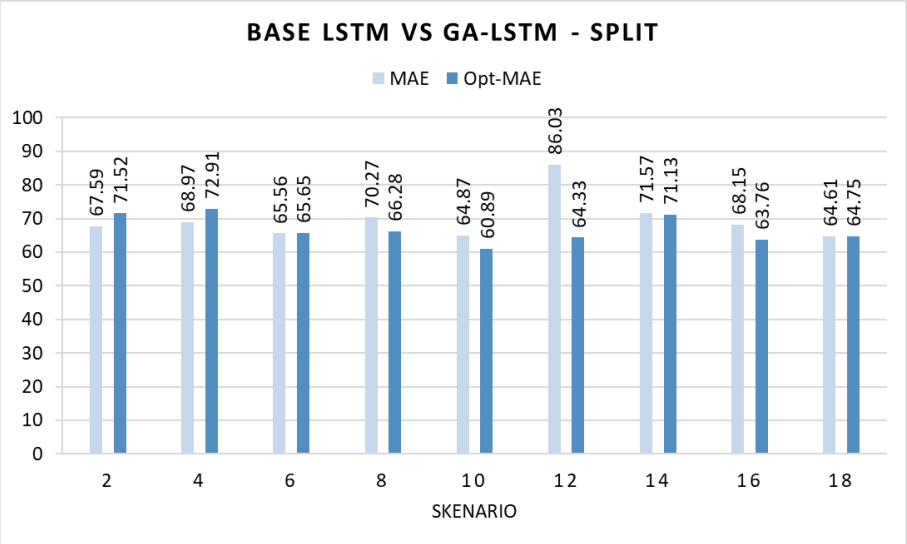
Tabel 4.3 Tabel Hasil Pengujian Base LSTM - EUR – Split

No.	LSTM Layer	Sliding Window	Train Split	MAE
1	1	5	0.8	86.12993
2			0.9	67.58582
3		10	0.8	85.60657
4			0.9	68.96591
5		20	0.8	96.69746
6			0.9	65.56289
7	2	5	0.8	90.87875
8			0.9	70.27362
9		10	0.8	83.74689
10			0.9	64.87418
11		20	0.8	87.50378
12			0.9	86.03064
13	3	5	0.8	92.43428
14			0.9	71.56865
15		10	0.8	97.27861
16			0.9	68.14946
17		20	0.8	94.44504
18			0.9	64.60814

2. GA-LSTM

Setelah mendapatkan parameter yang menghasilkan nilai *error* paling minimal pada setiap kelompok. Langkah selanjutnya adalah

mengoptimasi jumlah unit LSTM. Berikut merupakan grafik perbandingan nilai *error* antara Base LSTM dan GA-LSTM yang menggunakan teknik *split*:



Gambar 4.23 Grafik Perbandingan Base LSTM dan GA LSTM - EUR - Split

Dari grafik di atas dapat dilihat bahwa mayoritas optimasi menggunakan GA-LSTM berhasil, dan hal tersebut ditunjukkan pada skenario 8, 10, 12, 14, dan 16. Dimana hasil yang didapatkan dari GA-LSTM lebih rendah daripada Base LSTM dibandingkan dengan yang lain. Selain itu, meskipun nilai *error* dari beberapa skenario yang lain lebih tinggi, tetapi kenaikan tersebut tidak terlalu jauh. Oleh karena itu, hal tersebut juga bisa disimpulkan bahwa optimasi berhasil.

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

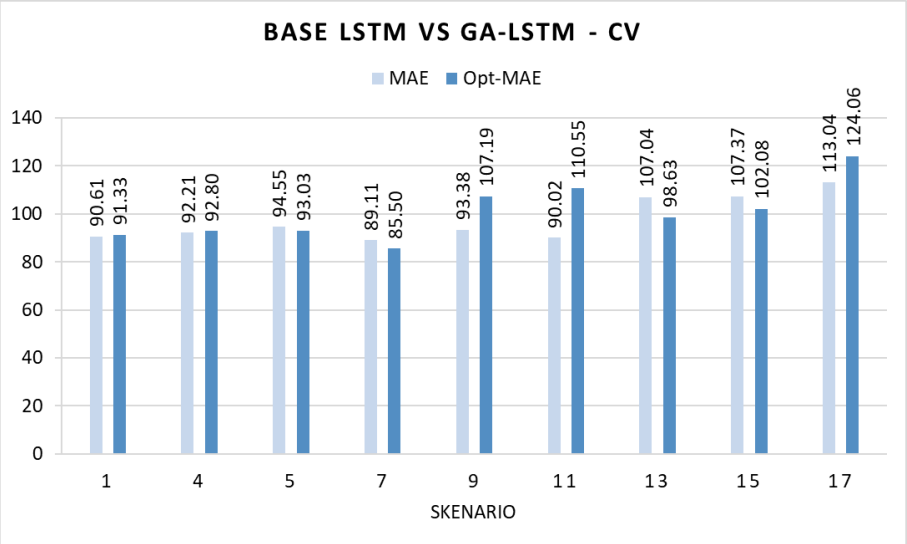
untuk mengetahui bagaimana jumlah *fold* mempengaruhi *error* sebuah model Base LSTM. Berikut merupakan tabel hasil pengujian menggunakan Base LSTM:

Tabel 4.4 Tabel Hasil Pengujian Base LSTM - EUR – CV

No.	LSTM Layer	Sliding Window	Fold	MAE
1	1	5	5	90.60559
2			10	94.95871
3		10	5	93.29106
4			10	92.20535
5		20	5	94.55227
6			10	96.01592
7	2	5	5	89.10508
8			10	93.54937
9		10	5	93.38299
10			10	95.39895
11		20	5	90.01693
12			10	95.02333
13	3	5	5	107.0407
14			10	108.3648
15		10	5	107.3652
16			10	113.0103
17		20	5	113.042
18			10	117.5863

2. GA-LSTM

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



Gambar 4.24 Grafik Perbandingan Base LSTM dan GA LSTM - EUR - CV

Dari grafik di atas dapat dilihat bahwa mayoritas optimasi menggunakan GA-LSTM berhasil. Dimana contohnya terdapat pada skenario 5, 7, 13, dan 15. Selain itu, meskipun nilai *error* dari beberapa skenario yang lain lebih tinggi, tetapi kenaikan tersebut tidak terlalu jauh. 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

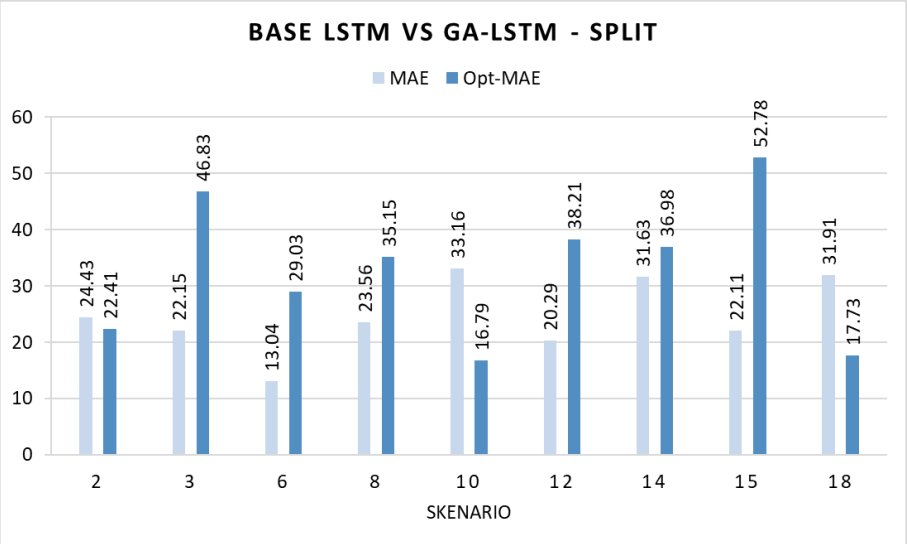
Data mata uang SGD akan dibagi menggunakan teknik split dan dengan variasi *split*, yaitu 80% dan 90% sebagai data *train*. Hal tersebut digunakan untuk mengetahui bagaimana jumlah data *train* mempengaruhi *error* sebuah model Base LSTM. Berikut merupakan tabel hasil pengujian menggunakan Base LSTM:

Tabel 4.5 Tabel Hasil Pengujian Base LSTM - SGD – Split

No.	LSTM Layer	Sliding Window	Train Split	MAE
1	1	5	0.8	24.69873
2			0.9	24.43479
3		10	0.8	22.1511
4			0.9	61.3383
5		20	0.8	22.21366
6			0.9	13.04035
7	2	5	0.8	26.18005
8			0.9	23.55826
9		10	0.8	49.05277
10			0.9	33.15783
11		20	0.8	30.09292
12			0.9	20.29498
13	3	5	0.8	52.75386
14			0.9	31.62649
15		10	0.8	22.1138
16			0.9	40.37385
17		20	0.8	49.04305
18			0.9	31.90867

2. GA-LSTM

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



Gambar 4.25 Grafik Perbandingan Base LSTM dan GA LSTM - SGD - Split

Dari grafik di atas dapat dilihat bahwa mayoritas optimasi menggunakan GA-LSTM tidak berhasil. Hanya beberapa saja yang berhasil yaitu pada skenario nomor 2, 10, dan 18. Dimana hasil yang didapatkan dari GA-LSTM lebih rendah daripada Base LSTM dibandingkan dengan yang lain.

b. Pengujian Menggunakan Teknik Cross Validation

1. Base LSTM

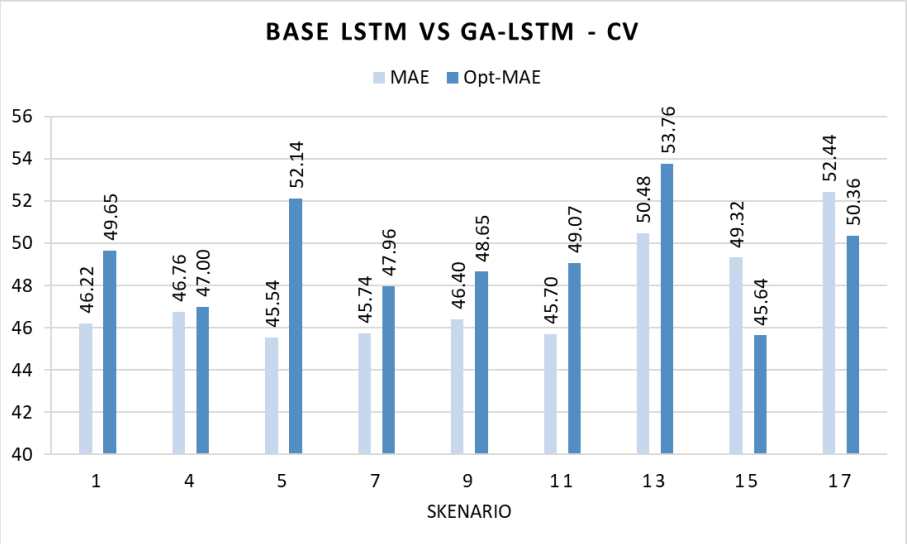
Data mata uang SGD akan dibagi menggunakan teknik *cross validation*, dan dengan variasi *fold*, yaitu 5 dan 10. Hal tersebut digunakan untuk mengetahui bagaimana jumlah *fold* mempengaruhi *error* sebuah model Base LSTM. Berikut merupakan tabel hasil pengujian menggunakan Base LSTM:

Tabel 4.6 Tabel Hasil Pengujian Base LSTM - SGD – CV

No.	LSTM Layer	Sliding Window	Fold	MAE
1	1	5	5	46.2215
2			10	47.25754
3		10	5	48.05351
4			10	46.7608
5		20	5	45.53695
6			10	50.79324
7	2	5	5	45.73907
8			10	45.87923
9		10	5	46.39697
10			10	48.32829
11		20	5	45.69571
12			10	49.09411
13	3	5	5	50.4788
14			10	58.2891
15		10	5	49.32427
16			10	52.69483
17		20	5	52.44273
18			10	60.04278

2. GA-LSTM

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



Gambar 4.26 Grafik Perbandingan Base LSTM dan GA LSTM - SGD - CV

Dari grafik di atas dapat dilihat bahwa mayoritas optimasi menggunakan GA-LSTM tidak berhasil. Hanya 2 saja yang berhasil yaitu pada skenario nomor 15 dan 17. Dimana hasil yang didapatkan dari GA-LSTM lebih rendah daripada Base LSTM dibandingkan dengan yang lain. Selain itu, ada skenario yang lain lebih tinggi, tetapi kenaikan tersebut tidak terlalu jauh, contohnya pada skenario nomor 4. Oleh karena itu, hal tersebut juga bisa disimpulkan bahwa optimasi berhasil.

4.6. Arsitektur Optimal

Dari seluruh percobaan yang dilakukan dari data USD, EUR, dan SGD, peneliti akhirnya mendapatkan arsitektur yang paling optimal dalam memprediksi harga mata uang tersebut. Berikut rincian arsitektur untuk setiap data:

- Untuk USD menggunakan model dengan 3 layer LSTM, dimana masing – masing layernya memiliki 128, 64, dan 32

cell, *sliding window* dengan ukuran 10, dan *train* data dengan jumlah 80% menggunakan teknik *split*. Hal ini dibuktikan pada Gambar 4.20 dan 4.21 dimana arsitektur tersebut mendapatkan MAE paling kecil yaitu sebesar 41,27.

- Sedangkan untuk EUR menggunakan model dengan 2 layer LSTM, dimana masing – masing layernya memiliki 237 dan 247 *cell*, *sliding window* dengan ukuran 10, dan *train* data dengan jumlah 90% menggunakan teknik *split*. Hal ini dibuktikan pada Gambar 4.22 dan 4.23 dimana arsitektur tersebut mendapatkan MAE paling kecil yaitu sebesar 60,89.
- Terakhir, untuk SGD menggunakan model dengan 1 layer LSTM yang memiliki jumlah 128 *cell*, *sliding window* dengan ukuran 20, dan *train* data dengan jumlah 90% menggunakan teknik *split*. Hal ini dibuktikan pada Gambar 4.24 dan 4.25 dimana arsitektur tersebut mendapatkan MAE paling kecil yaitu sebesar 13,04.

4.7. Hasil Prediksi

Terakhir adalah menguji model terbaik dari masing – masing data untuk memprediksi harga beli mata uang dalam kurun waktu 1 – 20 hari ke depan. Berikut merupakan hasil prediksi 5 hari ke depan dan nilai *error* jika dibandingkan dengan data asli:

```
USD/IDR Predicted Price for 5 Days
1 June 2023 : 14968.701
2 June 2023 : 14969.195
3 June 2023 : 14968.861
4 June 2023 : 14967.796
5 June 2023 : 14966.213
```

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

```
---------------- USD/IDR Actual Error ----------------
Mean Absolute Error for prediction : 64.47496093750014
Mean Squared Error for prediction : 4973.205184120197
Root Mean Squared Error for prediction : 70.52095563816614
```

Gambar 4.27 Hasil Prediksi dengan Data Asli - USD

```
EUR/IDR Predicted Price for 5 Days
1 June 2023 : 16059.675
2 June 2023 : 16063.440
3 June 2023 : 16068.009
4 June 2023 : 16072.641
5 June 2023 : 16077.258
```

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

```
---------------- EUR/IDR Actual Error ----------------
Mean Absolute Error for prediction : 101.8283203125
Mean Squared Error for prediction : 11160.06822566986
Root Mean Squared Error for prediction : 105.64122408259884
```

Gambar 4.28 Hasil Prediksi dengan Data Asli - EUR

```
SGD/IDR Predicted Price for 5 Days
1 June 2023 : 11059.618
2 June 2023 : 11062.005
3 June 2023 : 11063.063
4 June 2023 : 11063.227
5 June 2023 : 11062.908
```

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

```
---------------- SGD/IDR Actual Error ----------------
Mean Absolute Error for prediction : 29.45636718749993
Mean Squared Error for prediction : 1072.721167118844
Root Mean Squared Error for prediction : 32.75242230917958
```

Gambar 4.29 Hasil Prediksi dengan Data Asli - SGD

Berdasarkan ketiga gambar di atas, model USD dan SGD mampu memprediksi pergerakan harga beli dalam jangka pendek dengan cukup baik. Sedangkan model EUR perlu ditingkatkan lagi agar mampu memprediksi pergerakan harga beli. Hal tersebut dikarenakan selisih MAE antara arsitektur optimal dengan prediksi terlalu tinggi yaitu sekitar 40,94. Oleh karena itu, model EUR perlu pengoptimalan parameter kembali untuk dapat memprediksi harga mata uang dalam jangka pendek.

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 yang paling optimal untuk data USD adalah 3 layer LSTM, dimana masing – masing layernya memiliki 128, 64, dan 32 *cell*, *sliding window* dengan ukuran 10, dan *train* data dengan jumlah 80% menggunakan teknik *split* mendapatkan MAE sebesar 41.27.
5. Model yang paling optimal untuk data EUR adalah 2 layer LSTM, dimana masing – masing layernya memiliki 237 dan 247 *cell*, *sliding window* dengan ukuran 10, dan *train* data dengan jumlah 90% menggunakan teknik *split* mendapatkan MAE sebesar 60,89.

6. Model yang paling optimal untuk data SGD adalah 1 layer LSTM yang memiliki jumlah 128 *cell*, *sliding window* dengan ukuran 20, dan *train* data dengan jumlah 90% menggunakan teknik *split* mendapatkan MAE sebesar 13,04.
7. Model LSTM mampu memprediksi pergerakan harga beli mata uang asing USD dan SGD terhadap IDR dengan cukup akurat untuk jangka pendek, tetapi tidak untuk EUR.

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 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. Memperbanyak jumlah generasi yang ada pada Algoritma Genetik untuk mendapatkan model dengan hasil yang lebih optimal dibandingkan dengan yang sebelumnya.

5. Mempertimbangkan untuk menggunakan metode *crossover* dan mutasi yang lain untuk pengoptimalan parameter pada Algoritma Genetik.
6. Menggunakan algoritma optimasi yang lain seperti PSO dalam pengoptimalan parameter sebuah model.
7. Mempertimbangkan untuk melakukan *feature engineering* terhadap data untuk meningkatkan kemampuan prediksi model.
8. 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] S. García, J. Luengo dan F. Herrera, Data Preprocessing in Data Mining, Springer, 2015.

[11] S. Du Toit, A. Steyn dan R. Stumpf, Graphical Exploratory Data Analysis, New York: Springer-Verlag, 1986.

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

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

[14] 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.

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

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

LAMPIRAN

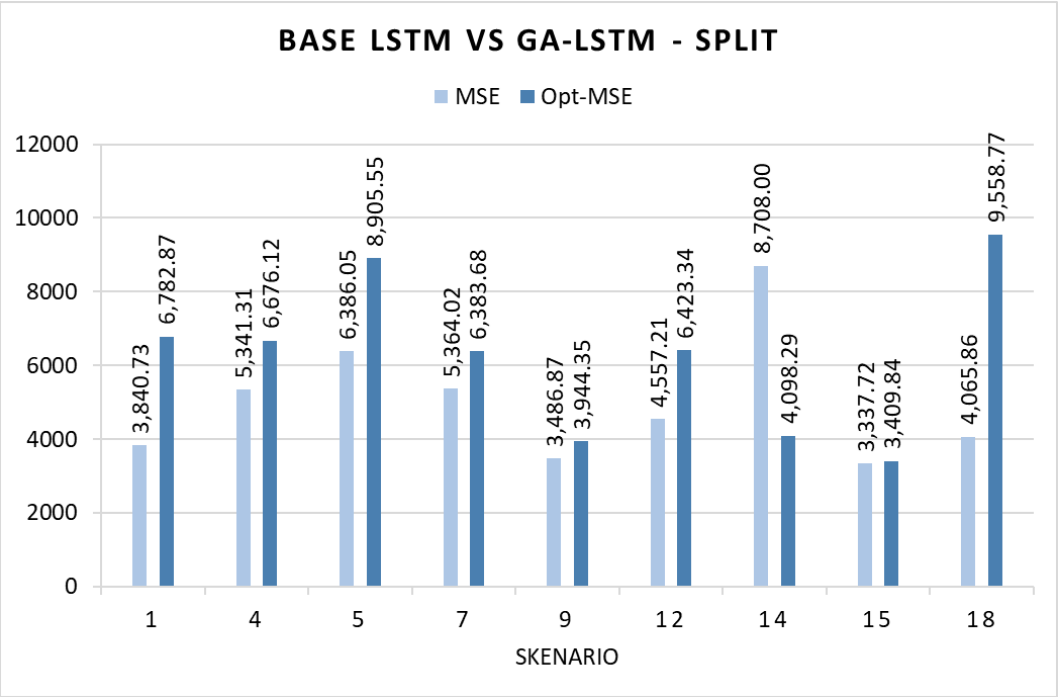
Lampiran 1 Tabel Hasil Base LSTM - USD - Split

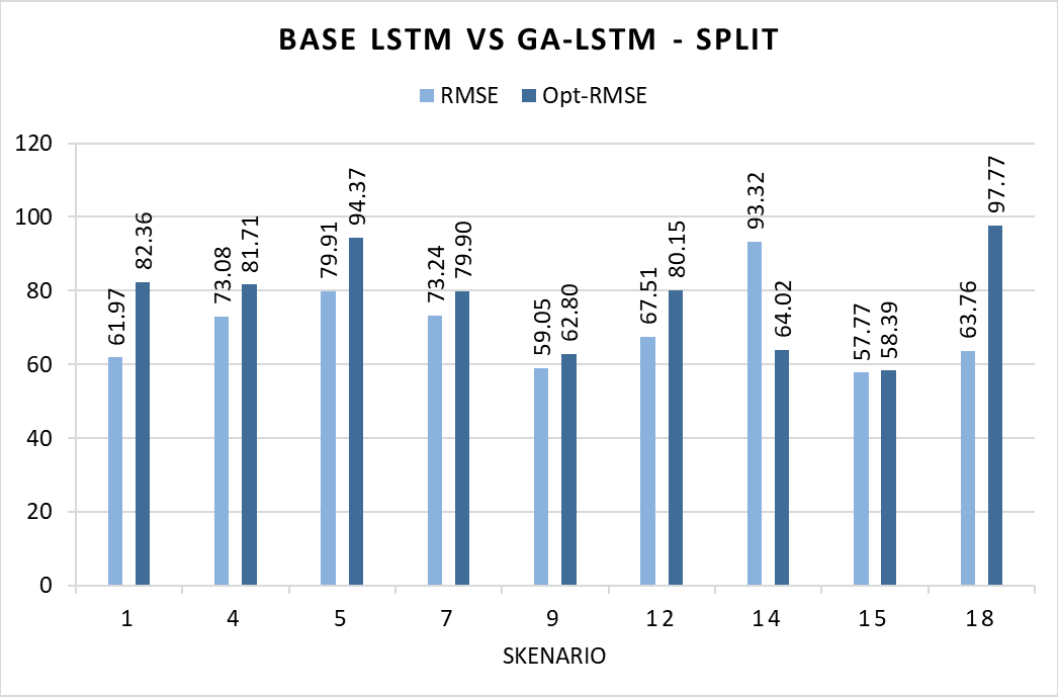
Skenario	Model	Parameter			Hasil	
		LSTM Layer	Sliding Window	Train Split	MSE	RMSE
1	Base LSTM (50 Epoch)	1	5	0.8	3840.727	61.9736
2				0.9	4896.378	69.97412
3			10	0.8	6413.274	80.08292
4				0.9	5341.315	73.0843
5			20	0.8	6386.053	79.91278
6				0.9	8926.742	94.48144
7		2	5	0.8	5364.023	73.23949
8				0.9	10533.77	102.6342
9			10	0.8	3486.874	59.04976
10				0.9	4974.861	70.5327
11			20	0.8	10017.81	100.089
12				0.9	4557.213	67.50713
13		3	5	0.8	17630.83	132.7811
14				0.9	8707.997	93.31665
15			10	0.8	3337.719	57.77299
16				0.9	6988.378	83.59652
17			20	0.8	4126.337	64.23657
18				0.9	4065.858	63.76408

Lampiran 2 Tabel Hasil GA-LSTM - USD - Split

Skenario	Kromosom Optimal	Fitness Score	Hasil Optimasi (GA-LSTM)	
			Opt-MSE	Opt-RMSE
1	[12, 226, 148]	-0.00088081	6782.874	82.358207
4	[13, 6, 86]	-0.00121468	6676.116	81.707505
5	[197, 52, 202]	-0.00097739	8905.547	94.369208
7	[195, 120, 217]	-0.00102451	6383.684	79.897959
9	[233, 219, 38]	-0.0008798	3944.351	62.804071
12	[177, 202, 209]	-0.00111239	6423.338	80.145728
14	[139, 237, 195]	-0.00154668	4098.293	64.017909
15	[247, 223, 220]	-0.0010239	3409.844	58.393867
18	[201, 229, 28]	-0.00131166	9558.769	97.768957

Lampiran 3 Grafik Perbandingan Base LSTM dan GA-LSTM - USD - Split





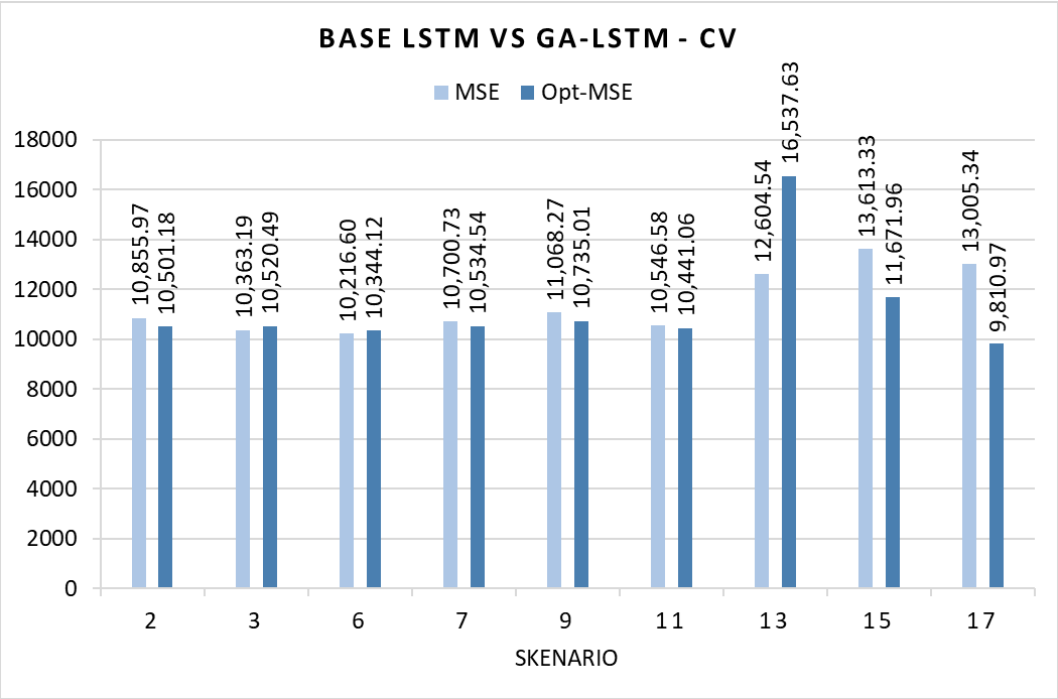
Lampiran 4 Tabel Hasil Base LSTM - USD – CV

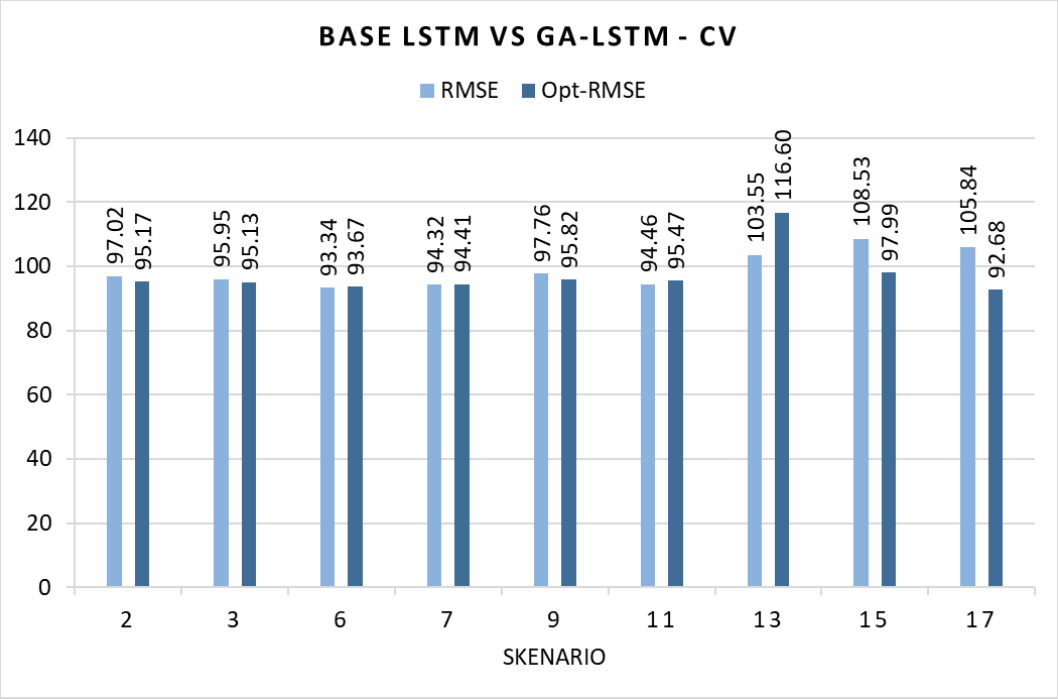
Skenario	Model	Parameter			Hasil	
		LSTM Layer	Sliding Window	Fold	MSE	RMSE
1	Base LSTM (50 Epoch)	1	5	5	12277.68	104.8977
2				10	10855.97	97.01966
3			10	5	10363.19	95.9502
4				10	11076.98	96.86242
5			20	5	11970.48	104.075
6				10	10216.6	93.33779
7		2	5	5	10700.73	94.31877
8				10	12745.6	103.2076
9			10	5	11068.27	97.7643
10				10	11758.34	100.4966
11			20	5	10546.58	94.46444
12				10	10713.91	96.06772
13		3	5	5	12604.54	103.5453
14				10	15098.33	109.8239
15			10	5	13613.33	108.5327
16				5	12277.68	104.8977
17			20	10	10855.97	97.01966
18				5	10363.19	95.9502

Lampiran 5 Tabel Hasil GA-LSTM - USD - CV

Skenario	Kromosom Optimal	Fitness Score	Hasil Optimasi (GA-LSTM)	
			Opt-MSE	Opt-RMSE
2	[221, 139, 183]	-0.001822	10501.18	95.170838
3	[158, 70, 157]	-0.00201337	10520.49	95.125791
6	[248, 41, 127]	-0.00197046	10344.12	93.668079
7	[202, 111, 131]	-0.00182678	10534.54	94.406994
9	[221, 120, 173]	-0.00209539	10735.01	95.816854
11	[210, 208, 121]	-0.00200196	10441.06	95.471993
13	[112, 219, 79]	-0.00211243	16537.63	116.59503
15	[115, 199, 177]	-0.00264894	11671.96	97.986146
17	[188, 237, 160]	-0.00262915	9810.965	92.683226

Lampiran 6 Grafik Perbandingan Base LSTM dan GA-LSTM - USD - CV





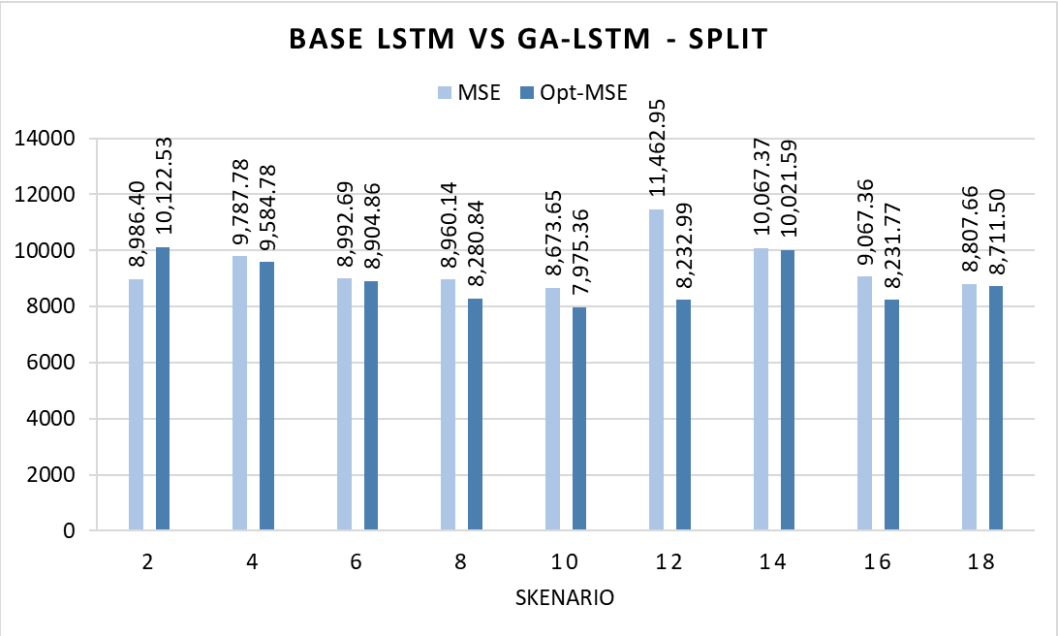
Lampiran 7 Tabel Hasil Base LSTM - EUR – Split

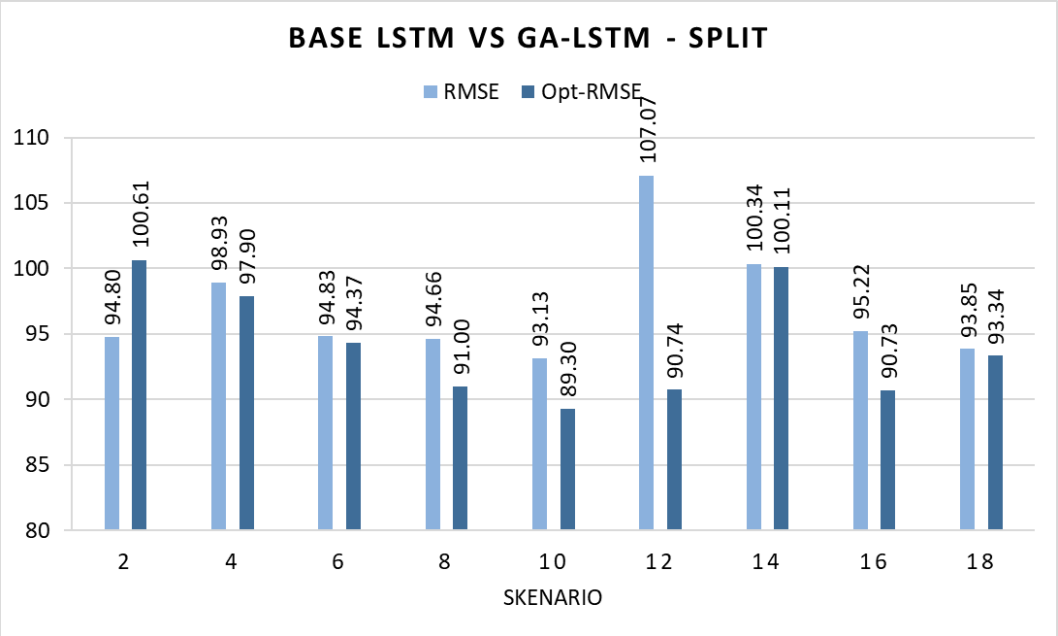
Skenario	Model	Parameter			Hasil	
		LSTM Layer	Sliding Window	Train Split	MSE	RMSE
1	Base LSTM (50 Epoch)	1	5	0.8	15007.75	122.5061
2				0.9	8986.404	94.79665
3			10	0.8	15017.86	122.5474
4				0.9	9787.784	98.93323
5			20	0.8	17463.68	132.1502
6				0.9	8992.689	94.82979
7		2	5	0.8	15839.49	125.855
8				0.9	8960.142	94.65803
9			10	0.8	14415.45	120.0644
10				0.9	8673.645	93.13241
11			20	0.8	14983.49	122.4071
12				0.9	11462.95	107.0652
13		3	5	0.8	17010.87	130.4257
14				0.9	10067.37	100.3363
15			10	0.8	17751.13	133.2334
16				0.9	9067.358	95.22267
17			20	0.8	17201.74	131.1554
18				0.9	8807.661	93.84914

Lampiran 8 Tabel Hasil GA-LSTM - EUR - Split

Skenario	Kromosom Optimal	Fitness Score	Hasil Optimasi (GA-LSTM)	
			Opt-MSE	Opt-RMSE
2	[155, 212, 228]	-0.00076037	10122.53	100.6108
4	[231, 129, 245]	-0.00072149	9584.778	97.901878
6	[144, 190, 245]	-0.00073979	8904.855	94.36554
8	[164, 213, 117]	-0.00078099	8280.843	90.999137
10	[237, 247, 70]	-0.00072139	7975.356	89.304846
12	[156, 115, 209]	-0.0007601	8232.994	90.735846
14	[237, 231, 219]	-0.0008933	10021.59	100.10788
16	[148, 197, 222]	-0.00084863	8231.773	90.729117
18	[224, 247, 153]	-0.000826	8711.495	93.33539

Lampiran 9 Grafik Perbandingan Base LSTM dan GA-LSTM - EUR - Split





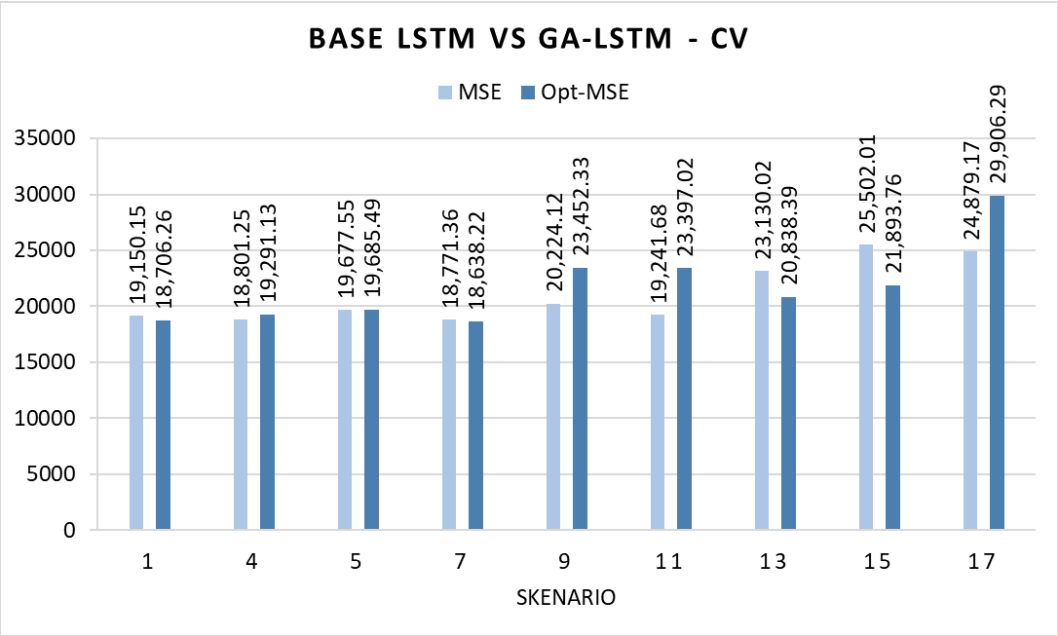
Lampiran 10 Tabel Hasil Base LSTM - EUR – CV

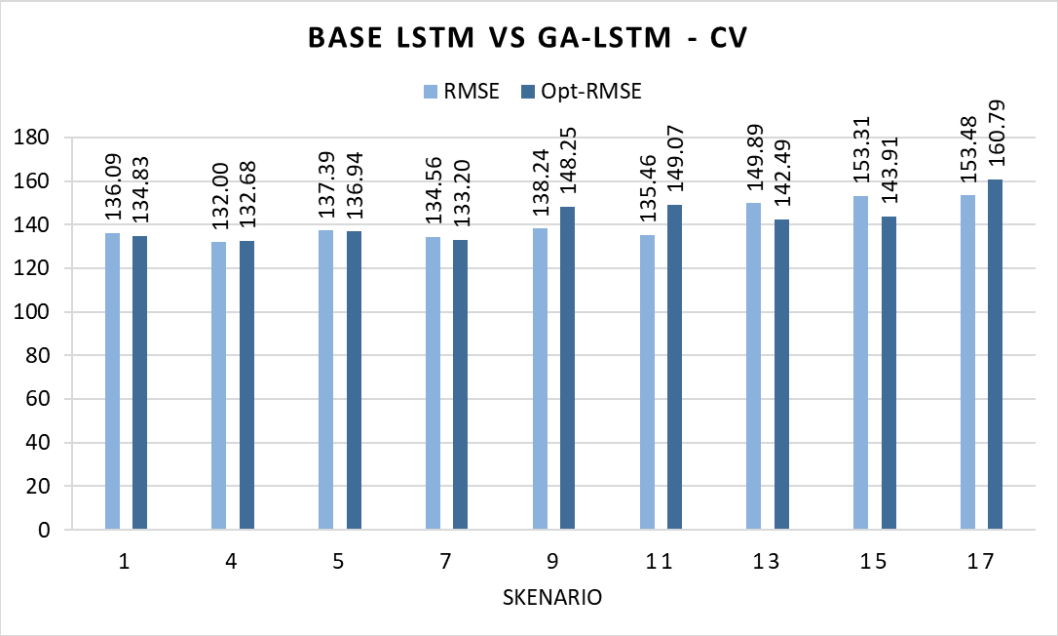
Skenario	Model	Parameter			Hasil	
		LSTM Layer	Sliding Window	Fold	MSE	RMSE
1	Base LSTM (50 Epoch)	1	5	5	19150.15	136.0859
2				10	19398.22	134.3296
3			10	5	19826.48	137.5794
4				10	18801.25	132.0046
5			20	5	19677.55	137.3852
6				10	19864.39	135.9006
7		2	5	5	18771.36	134.5575
8				10	19091.19	133.2921
9			10	5	20224.12	138.2365
10				10	19703.96	134.9555
11			20	5	19241.68	135.4552
12				10	20536.58	136.6424
13		3	5	5	23130.02	149.8932
14				10	23352.78	145.4553
15			10	5	25502.01	153.3072
16				5	26398.37	153.6982
17			20	10	24879.17	153.4804
18				5	27251.48	158.1592

Lampiran 11 Tabel Hasil GA-LSTM - EUR – CV

Skenario	Kromosom Optimal	Fitness Score	Hasil Optimasi (GA-LSTM)	
			Opt-MSE	Opt-RMSE
1	[248, 63, 74]	-0.00161558	18706.26	134.82933
4	[233, 26, 113]	-0.00167081	19291.13	132.68014
5	[240, 35, 18]	-0.00178445	19685.49	136.93883
7	[249, 174, 200]	-0.00168628	18638.22	133.20026
9	[247, 248, 6]	-0.00186473	23452.33	148.24843
11	[189, 185, 58]	-0.00198084	23397.02	149.06524
13	[134, 216, 13]	-0.00237228	20838.39	142.48683
15	[214, 241, 242]	-0.00247784	21893.76	143.91166
17	[213, 240, 113]	-0.00266251	29906.29	160.79208

Lampiran 12 Grafik Perbandingan Base LSTM dan GA-LSTM - EUR - CV





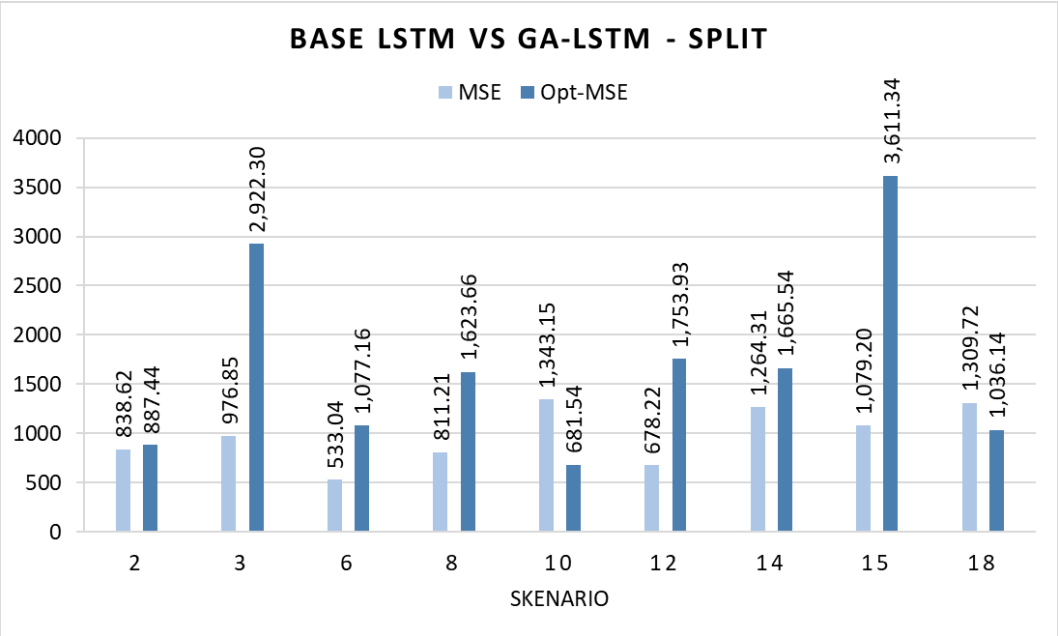
Lampiran 13 Tabel Hasil Base LSTM - SGD – Split

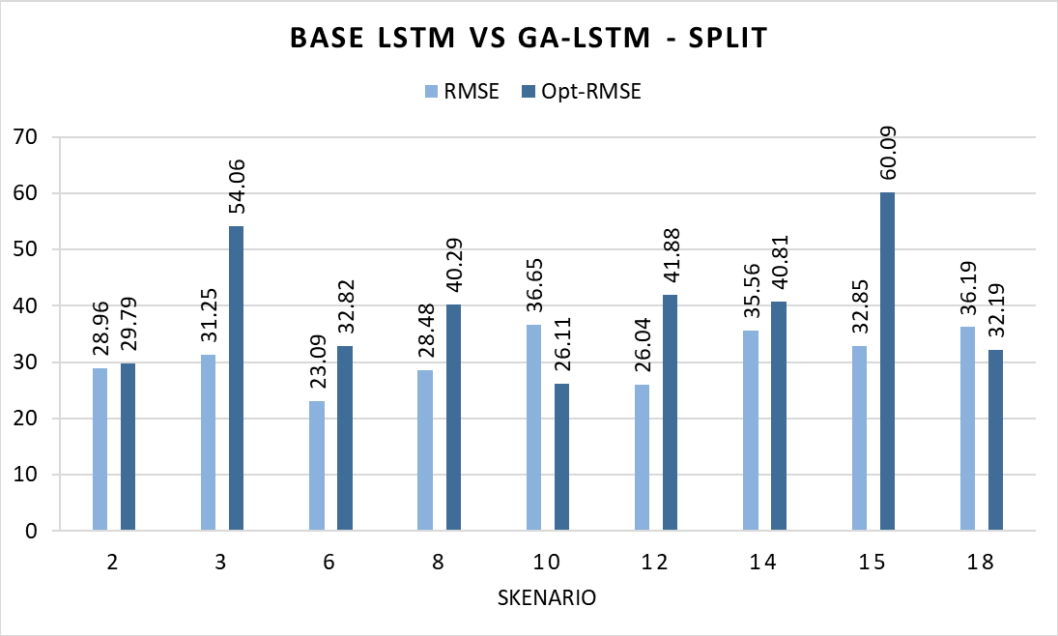
Skenario	Model	Parameter			Hasil	
		LSTM Layer	Sliding Window	Train Split	MSE	RMSE
1	Base LSTM (50 Epoch)	1	5	0.8	1029.553	32.0866
2				0.9	838.6234	28.959
3			10	0.8	976.8504	31.2546
4				0.9	4152.341	64.4387
5			20	0.8	974.3835	31.2151
6				0.9	533.0351	23.0876
7		2	5	0.8	1092.854	33.0583
8				0.9	811.21	28.4817
9			10	0.8	3215.791	56.7079
10				0.9	1343.148	36.649
11			20	0.8	1312.013	36.2217
12				0.9	678.2172	26.0426
13		3	5	0.8	3714.153	60.9439
14				0.9	1264.312	35.5572
15			10	0.8	1079.203	32.8512
16				0.9	1928.53	43.915
17			20	0.8	3283.483	57.3017
18				0.9	1309.715	36.19

Lampiran 14 Tabel Hasil GA-LSTM - SGD – Split

Skenario	Kromosom Optimal	Fitness Score	Hasil Optimasi (GA-LSTM)	
			Opt-MSE	Opt-RMSE
2	[91, 135, 39]	-0.00041335	887.4353	29.789852
3	[171, 88, 8]	-0.00076985	2922.296	54.058264
6	[250, 108, 235]	-0.00039799	1077.159	32.820104
8	[188, 208, 82]	-0.00044493	1623.663	40.294704
10	[228, 78, 50]	-0.00040971	681.5374	26.106271
12	[199, 81, 184]	-0.0003973	1753.931	41.879955
14	[216, 224, 175]	-0.00050605	1665.539	40.811017
15	[248, 156, 77]	-0.00084786	3611.339	60.094414
18	[231, 54, 138]	-0.00039825	1036.14	32.189126

Lampiran 15 Grafik Perbandingan Base LSTM dan GA-LSTM - SGD - Split





Lampiran 16 Tabel Hasil Base LSTM - SGD – CV

Skenario	Model	Parameter			Hasil	
		LSTM Layer	Sliding Window	Fold	MSE	RMSE
1	Base LSTM (50 Epoch)	1	5	5	7330.369	72.9435
2				10	7269.025	68.6544
3			10	5	7468.282	73.148
4				10	7190.853	68.8998
5			20	5	7749.6	72.8148
6				10	8110.381	73.0449
7		2	5	5	7374.418	72.3589
8				10	7501.779	69.4739
9			10	5	7355.406	72.3544
10				10	7407.342	69.0028
11			20	5	7491.846	73.0355
12				10	8036.92	71.8924
13		3	5	5	8509.408	78.7531
14				10	9812.988	81.8097
15			10	5	7929.646	75.1491
16				5	8270.676	74.542
17			20	10	8621.78	79.1231
18				5	10382.93	83.0949

Lampiran 17 Tabel Hasil GA-LSTM - SGD – CV

Skenario	Kromosom Optimal	Fitness Score	Hasil Optimasi (GA-LSTM)	
			Opt-MSE	Opt-RMSE
1	[143, 112, 195]	-0.00433205	8035.198	76.906982
4	[200, 68, 138]	-0.00425452	7117.088	68.876985
5	[155, 23, 218]	-0.0045194	8370.505	78.741646
7	[168, 214, 144]	-0.00435935	7648.556	74.75587
9	[155, 165, 133]	-0.00448879	7266.178	74.252768
11	[240, 67, 64]	-0.00472837	8016.891	75.27994
13	[34, 227, 210]	-0.00496336	8391.721	80.361992
15	[34, 198, 160]	-0.00551843	7918.155	73.553571
17	[95, 207, 126]	-0.00558219	8601.603	78.650835

Lampiran 18 Grafik Perbandingan Base LSTM dan GA-LSTM - SGD - CV