ALGORITMA ARTIFICIAL NEURAL NETWORKS – GENETIKA ALGORITMA UNTUK MEMPREDIKSI HARGA VALAS

SKRIPSI

Diajukan Untuk Memenuhi Salah Satu Syarat

Memperoleh Gelar Sarjana Komputer

Program Studi Informatika

Disusun oleh:

FX Doni Wahyu Sb

175314022

Fakultas Sains dan Teknologi

Universitas Sanata Dharma

Yogyakarta

2022

ARTIFICIAL NEURAL NETWORKS ALGORITHM–GENETIC ALGORITHM TO PREDICT FOREX PRICES

THESIS

Present as Patrial Fullfillment of the Requirements

For The Degree of Sarjana Komputer

In Informatics Study Program

By:

FX Doni Wahyu Sb

175314022

Faculty of Science and Technology

Sanata Dharma University

Yogyakarta

2022

HALAMAN PERSETUJUAN

SKRIPSI

ALGORITMA ARTIFICIAL NEURAL NETWORKS – GENETIKA ALGORITMA UNTUK MEMPREDIKSI HARGA VALAS

Disusun Oleh

Fx Doni Wahyu Setyabudi

175314022

Dosen Pembimbing,

Drs. Hari Suparwito, S.J., M.App.IT

Tanggal, 30/01/2023

HALAMAN PENGESAHAN
SKRIPSI

ALGORITMA ARTIFICIAL NEURAL NETWORKS – GENETIKA ALGORITMA UNTUK MEMPREDIKSI HARGA VALAS

Dipersiapkan dan ditulis oleh :

FX Doni Wahyu Setyabudi
175314022

SUSUNAN DEWAN PENGUJI

JABATAN	NAMA LENGKAP	TANDA TANGAN
Ketua	:Eko Hari Parmadi, S.Si., M.Kom.	.....
Sekretaris	:Ir. Kartono Pinaryanto, S.T., M.Cs	.....
Anggota	:Drs. Hari Suparwito, S.J., M.App.IT	.....

Yogyakarta, 30 Januari 2023

Fakultas Sains dan Teknologi
Universitas Sanata Dharma
Dekan,

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

LEMBAR 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, 13 Januari 2023

Penulis,

FX Doni Wahyu Setyabudi

LEMBAR PERSEMBAHAN

"Sebab bukan kepada panahku aku percaya, dan pedangkupun tidak memberi aku kemenangan, tetapi Engkaulah yang memberi kami kemenangan terhadap para lawan kami, dan orang-orang yang membenci kami Kau beri malu. Karena Allah kami nyanyikan puji-pujian sepanjang hari, dan bagi nama-Mu kami mengucapkan syukur selama-lamanya."

Mazmur (44:6-8)

LEMBAR PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH UNTUK KEPENTINGAN AKADEMIS

Yang bertanda tangan dibawah ini, saya mahasiswa Universitas Sanata Dharma:

Nama : Fx Doni Wahyu Setyabudi

Nomor Induk Mahasiswa : 175314017

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

"ALGORITMA ARTIFICIAL NEURAL NETWORKS – GENETIKA ALGORITMA UNTUK MEMPREDIKSI HARGA VALAS"

Beserta perangkat yang diperlukan (bila ada). Dengan demikian saya memberikan hak kepada Perpustakaan Universitas Sanata Dharma baik untuk menyimpan, mengalihkan dalam bentuk media lain, mengolahnya dalam bentuk pangkalan data, mendistribusikannya secara terbatas, dan mempublikasikannya di internet atau media lain untuk kepentingan akademis tanpa meminta ijin dari saya maupun memberikan royalty kepada saya selama tetap mencantumkan nama saya sebagai penulis.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di Yogyakarta,

Pada tanggal: 23 Januari 2023

Yang menyatakan,

FX Doni Wahyu Setyabudi

ALGORITMA ARTIFICIAL NEURAL NETWORKS – GENETIKA ALGORITMA UNTUK MEMPREDIKSI HARGA VALAS

ABSTRAK

Harga valuta asing dapat diprediksi menggunakan beberapa indikator. Prediksi harga valuta asing biasanya dilakukan secara manual dimana pekerjaan ini akan memakan waktu dan tenaga lebih serta memiliki kesalahan yang besar. *Human error* merupakan masalah terbesar dalam prediksi harga valas. Tujuan dari penelitian ini adalah memprediksi harga penutupan valas pada hari yang sama dimana sistem dapat memprediksi harga secara otomatis. Dengan sistem ini, dapat mempersingkat waktu dan menghemat tenaga yang dibutuhkan dalam memprediksi harga valas. Terdapat 4 atribut utama dalam sistem ini yaitu harga pembukaan, harga tertinggi, harga terendah, dan harga terakhir. Atribut akan dilakukan pelatihan jaringan menggunakan *neural network* dengan penerapan algoritma genetika. Model *neural network* yang digunakan adalah 1 *hidden layer network*, 2 *hidden layer network*, dan 3 *hidden layer network*. Percobaan dilakukan dengan membandingkan *mean square error* (MSE) setiap *hidden layer*. Percobaan pada 3 model *neural network* menggunakan variasi 5 *neuron* di *hidden layer* pertama, 10 *neuron* di *hidden layer* kedua, dan 10 *neuron* di *hidden layer* ketiga. Lalu model *neural network* dengan *hidden layer* terbaik akan dilakukan percobaan fungsi aktivasi. Dari percobaan ini telah didapatkan hasil akurasi terbaik pada *neural network* – algoritma genetika dengan model 2 *hidden layer* dengan fungsi aktivasi ReLU sebesar 0.000061372. Dengan hasil ini maka neural network dengan penerapan algoritma genetika dapat digunakan untuk memprediksi harga valas dengan mengikuti trend yang ada.

Kata Kunci: Algoritma genetika, investasi, artificial neural network, valuta asing.

ARTIFICIAL NEURAL NETWORKS ALGORITHM–GENETIC ALGORITHM TO PREDICT FOREX PRICES

ABSTRACT

Foreign exchange prices can be predicted using several indicators. Prediction of foreign exchange prices is usually done manually where this work will take more time and effort and have big errors. Human error is the biggest problem in forex price prediction. The purpose of this study is to predict the closing price of foreign exchange on the same day where the system can predict prices automatically. With this system, it can shorten the time and save the energy required in predicting forex prices. There are 4 main attributes in this system, namely the opening price, the highest price, the lowest price, and the last price. Attributes will be carried out by network training using a neural network with the application of genetic algorithms. The neural network model used is 1 hidden layer network, 2 hidden layer networks, and 3 hidden layer networks. The experiment was carried out by comparing the mean square error (MSE) of each hidden layer. Experiments on 3 neural network models use variations of 5 neurons in the first hidden layer, 10 neurons in the second hidden layer, and 10 neurons in the third hidden layer. Then the neural network model with the best hidden layer will be tested on the activation function. From this experiment the best accuracy results have been obtained on the neural network - genetic algorithm with 2 hidden layer models with the ReLU activation function of 0.000061372. With these results, a neural network with the application of a genetic algorithm can be used to predict foreign exchange prices by following existing trends.

Keywords: Genetic algorithm, investment, artificial neural network, foreign exchange.

KATA PENGANTAR

Puji Syukur penulis panjatkan kepada Tuhan yang Maha Esa atas segala berkat dan karunia-Nya, sehingga penulis dapat menyelesaikan tugas akhir yang berjudul "ALGORITMA ARTIFICIAL NEURAL NETWORKS – GENETIKA ALGORITMA UNTUK MEMPREDIKSI HARGA VALAS". Tugas akhir ini merupakan salah satu mata kuliah wajib dan sebagai syarat untuk memperoleh gelar Sarjana Komputer Program Studi Informatika Universitas Sanata Dharma Yogyakarta.

Dalam proses penelitian, penulis banyak mendapat dukungan, bimbingan, dan motivasi dari berbagai pihak. Dengan segala kerendahan hati dan syukur yang mendalam, sudah sepantasnya penulis menyampaikan terima kasih kepada:

1. Tuhan yang Maha Esa, yang telah memberikan pertolongan dan kekuatan dalam proses pembuatan tugas akhir.
2. Romo Drs. Hari Suparwito, S.J., M.App.IT selaku dosen pembimbing tugas akhir yang telah bersedia meluangkan waktu, memberikan arahan dan masukan, serta motivasi penulis selama menyelesaikan skripsi.
3. Keluarga tercinta yang selalu memberikan dukungan dan doa sehingga memotivasi penulis dalam menyelesaikan tugas akhir.
4. Seluruh dosen Informatika Universitas Sanata Dharma yang telah mendidik dan memberikan ilmu pengetahuan kepada penulis yang digunakan sebagai bekal untuk menyelesaikan tugas akhir ini.
5. Teman-teman informatika 2017 yang telah memberikan kenangan indah tak terlupakan selama masa perkuliahan ini.
6. Sahabat-sahabat saya, Ani, Bagus, Zizi, Dwi Setya, Angger, Hasan, dan Dyah Eka yang telah menemani dan selalu ada ketika suka atau duka selama menyelesaikan tugas akhir ini.
7. Saudara Chrisstoper, Daniel, Koko, Joshept, yang telah suka rela menemani, membantu, memberi masukan dan saran selama pengerjaan tugas akhir ini.

Penulis menyadari bahwa masih banyak kekurangan dalam penyusunan tugas akhir ini, penulis sangat menerima apabila ada kritik dan saran yang membangun agar dapat digunakan untuk perbaikan dan kemajuan penulisan karya selanjutnya.

Yogyakarta, 20 Januari 2023

Penulis,

FX Doni Wahyu Setyabudi

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

1.1 Latar Belakang

Investasi di bidang *forex (foreign exchange)* atau lebih dikenal valuta asing (valas) merupakan suatu jenis pedagangan atau transaksi yang memperdagangkan mata uang suatu negara dengan mata uang negara lain. Investasi di bidang ini sendiri menjadi salah satu investasi yang mempunyai resiko yang tinggi, tetapi juga menghasilkan nilai kembali atau laba yang tinggi pula. Prediksi sendiri merupakan teknik yang penting dalam investasi di bidang valas. Dengan melakukan prediksi yang tepat pihak investor dan pedagang dapat memaksimalkan *profit*/untung dari setiap transaksi yang dibuat.

Untuk melakukan prediksi sendiri pihak investor biasanya akan melakukan analisa berdasarkan harga pasar di hari/waktu sebelumnya. Prediksi ini terkadang tidak selalu tepat dan terkadang merugikan pihak investor dan trader apabila prediksi yang dilakukannya salah/tidak tepat sasaran. Ketidaktepatan prediksi ini biasanya terjadi karena perubahan harga valuta asing sangat tidak menentu, maka dari itu pihak investor atau pedagang harus memiliki tingkat kejelian yang tinggi agar tidak mengalami kerugian yang berakibat bangkrut dalam jual-beli valuta asing ini. Masalah tersebut muncul karena banyak faktor, seperti pedagang kurang memahami pasar valuta asing, harga valuta asing yang cepat berubah karena perbedaan inflasi antar negara, harga valuta asing tersebut naik tiba-tiba karena berita politik, dll.

Berdasarkan penelitian dengan topik serupa yaitu prediksi menggunakan *neural network* oleh Primandani Arsi dan Joko Prayogi pada tahun 2020 didapat hasil berupa penurunan nilai RMSE dari 0,01 menjadi 0,008. Pada penelitian tersebut data yang digunakan hanya nilai kurs jual dan kurs beli dengan pengoptimalan nilai *learning rate* dan *momentum* (Arsi & Prayogi, 2020). Dan penelitian yang

dilakukan oleh Nadya Oktavia Rahardiani, Wayan Firdaus Mahmudy, dan Indriati yang menggunakan *neural network* dan algoritma genetika pada tahun 2018 juga didapatkan bahwa *neural network* dengan algoritma genetika menghasilkan nilai akurasi 60.60% dibandingkan dengan neural network dengan *stochastic gradient descendant* yang menghasilkan akurasi sebesar 88.40%. Pada penelitian tersebut data yang digunakan adalah data pemeriksaan darah pasien dengan 1 *hidden layer neural network* (Oktavia Rahardiani & Firdaus Mahmudy, 2018).

Berangkat dari penelitian ini penulis ingin melakukan penelitian serupa dengan menggunakan data set, model *neural network*, dan pengoptimalan yang sedikit berbeda. Model *neural network* yang digunakan oleh penulis adalah *multilayer perceptron* dengan 3 jenis *hidden layer* dan dengan proses optimalan nilai bobot pada *hidden layer*. Dari 3 jenis model ini akan dipilih 1 model yang memiliki nilai akurasi tertinggi, dimana data yang digunakan nanti akan dibagi menjadi data *training* untuk dilakukan pengoptimasian nilai bobot pada *hidden layer*, baru kemudian nilai bobot tersebut digunakan untuk melakukan prediksi pada data *testing*.

1.2 Rumusan Masalah

Dari latar belakang yang telah dijelaskan diatas, maka rumusan masalah di penelitian ini adalah memprediksi harga valas merupakan masalah yang cukup sulit lalu bagaimana cara memprediksi harga valas dengan pendekatan *machine learning*.

1.3 Tujuan Penelitian

Berdasarkan rumusan masalah diatas, maka tujuan penelitian ini adalah menerapkan ANN-GA untuk memprediksi harga penutupan pada hari tersebut.

1.4 Manfaat Penelitian

Berdasarkan rumusan masalah yang telah dibuat, maka manfaat penelitian ini adalah:

1. Manfaat penelitian ini bagi peneliti adalah mengimplementasikan ilmu jaringan syaraf tiruan dan algoritma genetika dalam memprediksi harga valas.
2. Manfaat penelitian ini bagi investor adalah memudahkan investor dalam mengambil keputusan dalam transaksi valas, serta meminimalisir kerugian investor, dan meningkatkan akurasi prediksi pedagang terhadap harga valas tersebut.

1.5 Batasan Masalah

Adapun batasan masalah yang dipakai dalam penelitian ini adalah sebagai berikut:

1. Data yang akan dipakai diambil dari situs https://www.finance.yahoo.com
2. Data yang digunakan hanya mata uang Euro Eropa dibanding Dollar Amerika.
3. Perangkat lunak dibangun menggunakan bahasa *python* 3.

1.6 Sistematika Penulisan

a. Bab I Pendahuluan

Bab I akan menjelaskan mengenai latar belakang penulis melakukan penelitian, rumusan masalah, tujuan masalah, dan manfaat penelitian bagi penulis dan pihak terkait yang dibatasi dengan batasan masalah penelitian. Bab I juga akan berisi sistematika penulisan yang menjelaskan isi dari tiap bab.

b. Bab II Landasan Teori

Untuk dapat memahami penilitian lebih mudah, penting untuk mengerti tentang dasar teori yang digunakan, tahapan – tahapan yang digunakan, serta metode yang digunakan. Bab ini juga akan menjelaskan tentang beberapa literatur yang digunakan sebagai referensi dalam penelitian ini.

c. Bab III Metodologi Penelitian

Metodologi penelitian berisi tentang diagram alur kerja yang digunakan dalam penelitian, data yang digunakan, dan model ANN-GA yang digunakan.

d. Bab IV Hasil dan Analisa

Hasil yang didapat dari percobaan yang penulis lakukan akan dijelaskan pada bab ini. Bab ini juga akan berisi tentang analisa dari hasil yang telah didapat.

e. Bab V Kesimpulan dan Saran

Bab ini menjelaskan tentang kesimpulan secara keseluruhan dari penelitian yang telah dilakukan dan saran penulis yang dapat digunakan untuk mengembangan penelitian serupa kedepannya.

BAB II

LANDASAN TEORI

2.1 Penelitian Terkait

Penulis menggunakan beberapa referensi sebagai acuan dalam mengerjakan penelitian ini. Berikut adalah daftar ringkasan dari referensi yang penulis gunakan.

Tabel 1 Penelitian terkait

No	Judul	Metode	Kesamaan & Perbedaan	Tindakan Penelitian
1	Optimasi Bobot Multi-Layer Perceptron Menggunakan Algoritma Genetika Untuk Klasifikasi Tingkat Resiko Penyakit Stroke (Oktavia Rahardiani & Firdaus Mahmudy, 2018)	Algoritma genetika dan jaringan syaraf tiruan	Kesamaan dalam penelitian ini dengan penelitian penulis adalah pada penelitian ini menggunakan metode ANN dengan algoritma genetika dalam melakukan klasifikasi penyakit sedangkan untuk perbedaan penelitian ini adalah metode yang digunakan diterapkan untuk klasifikasi penyakit stroke	Pada penelitian ini metode ANN-GA digunakan untuk melakukan klasifikasi dan data yang digunakan adalah data hasil cek darah pasien sedangkan pada penelitian penulis metode tersebut akan digunakan untuk melakukan prediksi menggunakan data harga valas dengan pasangan euro dengan usd
2	Perbandingan Metode ANN-PSO dan ANN-GA Dalam Pemodelan Komposisi Pakan Kambing	*Artificial Neural Network multi-layer perceptron,* algoritma genetika, dan *Particle Swarm Optimization*	Kesamaan dalam penelitian ini dengan penelitian penulis adalah pada penelelitian ini metode yang	Metode yang digunakan pada penelitian ini yaitu ANN-GA akan dibandingkan antar hidden

	Peranakan Etawa (Pe) Untuk Optimasi Kandungan Gizi (Amerilyse Caesar dkk., 2016)		digunakan ANN dengan GA, sedangkan untuk perbedaan penelitian ini adalah metode ini digunakan untuk mengoptimisasi pakan kambing dan metode ANN-GA dibandingkan dengan metode ANN-PSO	layer sebanyak 1, 2, dan 3 hidden layer.
3	Optimasi Prediksi Nilai Tukar Rupiah Terhadap Dolar Menggunakan Neural Network Berbasiskan Algoritma Genetika (Arsi & Prayogi, 2020)	*Artificial neural network* dan algoritma genetika	Kesamaaan penelitian ini dengan penelitian penulis adalah pada penelitian ini digunakan metode ANN dan algoritma genetika untuk memprediksi nilai mata uang, sedangkan yang menjadi pembeda dengan penelitian penulis adalah pada penelitian ini algoritma genetika digunakan untuk memilih parameter terbaik pada jaringan ANN	Pada penelitian yang dilakukan penulis algoritma genetika digunakan untuk mencari nilai bobot terbaik untuk jaringan ANN.
4	Optimasi Parameter Artificial Neural Network Menggunakan Algoritma	*Artificial neural network* dan Algoritma genetika	Kesamaan penelitian ini dengan penelitian penulis adalah pada penelitian	Pada penelitian ini data yang digunakan adalah data administrasi mahasiswa yang

	Genetika Untuk Prediksi Kelulusan Mahasiswa (Ali dkk., 2019)		ini diterapkan ANN dengan bantuan algoritma genetika, serta metode ini juga digunakan untuk prediksi. Sedangkan perbedaan penelitian ini dengan penelitian penulis adalah pada penelitian ini algoritma genetika digunakan untuk menentukan nilai learning rate	berisi nilai, ipk, nim, dan nama sedangkan pada penelitian yang dilakukan penulis data yang digunakan adalah daftar historis harga pasangan valas euro dan usd. Serta algoritma genetika pada penelitian penulis akan digunakan untuk mengubah nilai bobot pada jaringan ANN.
5	Prediksi Pergerakan Harga Saham Pada Bank Terbesar Di Indonesia Dengan Metode Backpropagation Neural Network (Novita, 2016)	*Backpropagation Neural Network*	Kesamaan penelitian ini dengan penelitian penulis adalah digunakannya model neural network untuk memprediksi pergerakan sebuah harga, sedangkan perbedaannya adalah model neural network yang digunakan pada penelitian ini menggunakan backpropagation dan harga yang diprediksi adalah harga saham bank BRI dan BCA.	Metode *neural network* yang digunakan pada penelitian penulis akan digabungkan dengan metode lain dan data harga yang digunakan adalah harga valas.

Dari penelitian yang telah dilakukan pada tabel tersebut dapat disimpulkan bahwa *artificial neural network* meruapakan salah satu metode yang dapat digunakan untuk memprediksi pergerakan harga dan bisa digabungkan dengan algoritma genetika.

2.2 Valuta Asing

2.2.1 Pengertian & Fungsi Valuta Asing

Menurut Kamus Besar Bahasa Indonesia (KBBI), valuta asing adalah mata uang asing yang digunakan dalam perdagangan internasional. Sedangkan tempat memperjualbelikan valuta asing tersebut dinamakan pasar valuta asing atau sering disingkat pasar valas. Biasanya mata uang yang diperdagangkan adalah uang dari negara yang memiliki peran besar dalam perekonomian global (Nadira, 2021). Dalam pasar valas, selisih nilai kurslah yang akan dijadikan keuntungan bagi para pedagang. Sebagai contoh, pedagang membeli dollar AS saat harga sedang rendah dan akan menjualnya kembali saat harga sudah naik. Dari selisih harga tersebutlah pedagang valas akan mendapatkan keuntungan.

Dikutip dari kumparan.com secara umum valuta asing berfungsi untuk alat pembayaran dalam transaksi perdagangan internasional (Bisnis, 2021). Selain fungsi utama tersebut valuta asing juga memiliki fungsi lain sebagai berikut:

1. Sebagai alat tukar internasional

   Valuta asing memiliki fungsi sebagai alat tukar dalam berbagai transaksi internasional. Perdagangan internasional dapat mencakup pembelian dan penjualan barang atau jasa. Misalnya, jika Indonesia ingin membeli barang atau jasa dari negara lain maka transaksi biasanya harus menggunakan valas negara tersebut.

2. Sebagai alat pengendali kurs

   Valuta asing juga dapat berfungsi sebagai pengontrol kurs mata uang. Nilai kurs harus dikendalikan, karena kurs mata uang bisa naik atau turun. Untuk menghindari fluktuasi besar, pemerintah negara sering mengontrol kurs uang di negara mereka dengan menggunakan valuta asing.

3. Sebagai alat untuk memperlancar perdagangan internasional

Negara-negara yang melakukan transaksi internasional dapat menggunakan valuta asing sebagai sarana untuk memperlancar transaksi perdagangannya. Valuta asing sangat penting dalam pasar perdagangan internasional. Tanpa itu, transaksi harus dilakukan dalam mata uang negara tempat mereka diperdagangkan yang dapat membatasi perdagangan secara keseluruhan.

2.2.2 Kurs Valuta Asing

Secara umum kurs adalah nilai tukar mata uang sebuah negara yang diukur dengan mata uang lain, kurs sendiri dapat terbagi menjadi 3 jenis (Bisnis.com, 2021):

1. Kurs Jual

   Kurs jual adalah nilai atau harga valas disaat pedagang atau bank membeli valas tersebut. Sebagai contoh jika kita ingin menukar valas dari Dollar AS ke Rupiah Indonesia.

2. Kurs Beli

   Kurs beli adalah nilai valas disaat pedagang atau bank menjual valas. Misalkan kita ingin menukar Rupiah Indonesia ke Dollar AS.

3. Kurs Tengah

   Kurs tengah merupakan gabungan dari kurs jual dan kurs beli yang dibagi dua atau rata-rata.

Kurs mata uang tentu dipengaruhi oleh beberapa faktor dalam naik atau turunnya. Faktor yang mempengaruhi kurs valuta asing ini dibagi menjadi 4 faktor utama sebagai berikut (Karunia, 2022):

1. Permintaan dan penawaran valas

   Harga valas bisa jadi lebih mahal dari nilai nominal yang berlaku, jika permintaannya lebih banyak dari jumlah yang ditawarkan. Ini juga bisa terjadi ketika jumlah permintaannya tetap, sementara penawaran berkurang. Sebaliknya harga valuta asing akan lebih murah dari harga nominal, ketika permintaannya sedikit sementara penawaran banyak, atau permintaannya makin menurun meski jumlah penawarannya tetap.

2. Inflasi

   Tingginya tingkat inflasi menunjukkan bahwa harga barang dalam negeri meningkat tajam. Inflasi adalah kenaikan harga barang dan jasa yang cenderung menurunkan nilainya dari waktu ke waktu. Berkaitan dengan valuta asing, ini juga bisa mempengaruhi nilai tukar mata uang asing. Misalnya, ketika terjadi inflasi, nilai uang cenderung turun, sehingga nilai tukar mata uang asing juga mengalami penurunan.

3. Tingkat suku bunga

   Jika suatu negara menaikkan suku bunga, itu akan menciptakan permintaan yang lebih tinggi untuk mata uang mereka. Dengan peningkatan permintaan mata uang, uang akan mengalir lebih baik di dalam negeri, yang kemudian menghasilkan peningkatan investasi asing ke negara itu.

4. Tingkat pendapatan dan produksi

   Apabila suatu negara mengalami pertumbuhan ekonomi yang pesat, ini menunjukkan bahwa pendapatan masyarakatnya sedang tinggi. Peningkatan pendapatan masyarakat akan diiringi dengan daya beli yang juga akan meningkat. Saat ini terjadi, negara akan mengimport lebih banyak produk dari negara lain. Semakin besar nilai barang yang diimport maka akan diikuti dengan naiknya permintaan valuta asing.

2.3 Algoritma Genetika

2.3.1 Pengertian Algoritma Genetika

Algoritma Genetika merupakan teknik untuk menemukan solusi optimal dari permasalahan yang mempunyai banyak solusi. Teknik ini akan mempertimbangkan berbagai solusi hingga dapat menghasilkan solusi terbaik menggunakan kriteria yang telah ditentukan atau yang disebut fungsi fitness. Algoritma ini masuk dalam kelompok algoritma evolusioner dengan menggunakan pendekatan evolusi Darwin di bidang Biologi seperti pewarisan sifat, seleksi alam, mutasi gen dan kombinasi (Suharjito, 2021).

Terdapat 3 struktur utama dalam algoritma genetika yaitu sebagai berikut (Syah, t.t.):

1. Membangkitkan Populasi Awal

   Populasi awal ini dibangkitkan secara random sehingga didapatkan solusi awal. Populasi terdiri dari sejumlah kromosom-kromosom yang merepresentasikan solusi yang diinginkan.

2. Pembentukan Generasi Baru

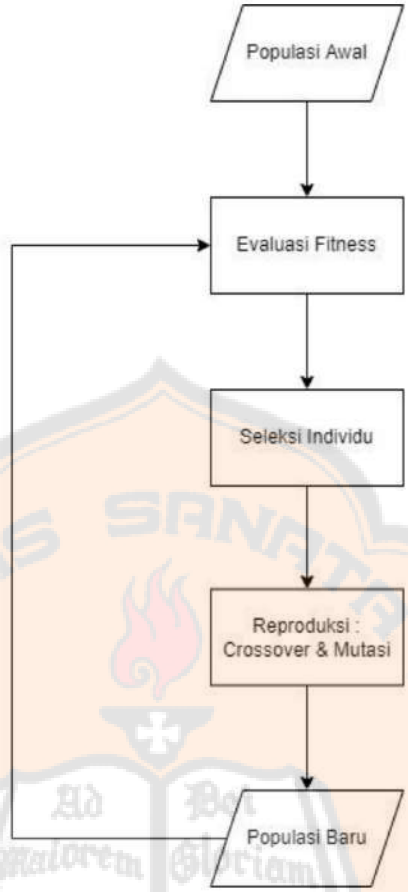
   Dalam pembentukan tersebut digunakan tiga operator, yaitu operator reproduksi/seleksi, crossover dan mutasi. Proses ini dilakukan berulang-ulang sehingga didapatkan jumlah kromosom yang cukup untuk membentuk generasi baru dimana generasi baru ini merupakan representasi dari solusi baru.

3. Evaluasi Solusi

   Pada proses ini akan dilakukan evaluasi setiap populasi dengan menghitung nilai fitness dari setiap kromosom dan akan terus melakukan evaluasinya hingga terpenuhi kriteria berhenti. Tetapi jika hingga akhir kriteria belum terpenuhi maka akan dibentuk generasi baru dengan mengulangi langkah 2. Beberapa kriteria berhenti yang sering digunakan antara lain:

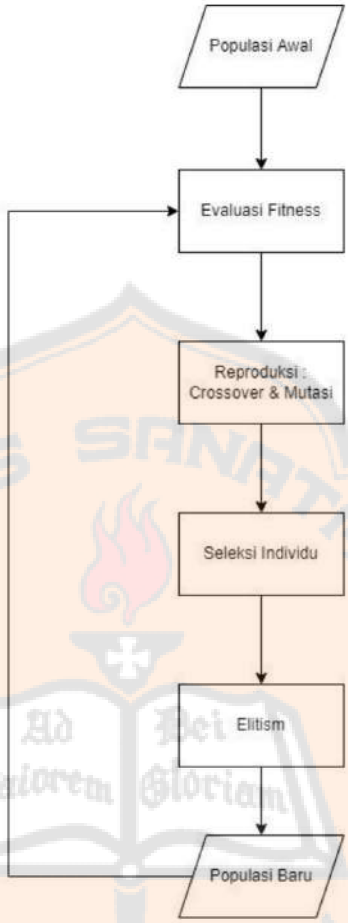
   a. Berhenti pada generasi tertentu.
   b. Berhenti setelah dalam beberapa generasi berturut-turut didapatkan nilai fitness tertinggi tidak berubah.
   c. Berhenti bila sampai n generasi tidak didapatkan nilai fitness yang lebih tinggi.

Siklus algoritma genetika pertama kali diperkenalkan oleh David Goldberg, dimana gambaran siklus tersebut dapat dilihat pada gambar 1.



Gambar 1 Siklus Algoritma Genetika oleh David Goldberg. Sumber: (Syah, t.t.)

Siklus ini kemudian diperbarui oleh beberapa ilmuwan yang mengembangkan algoritma genetika, yaitu Zbigniew Michalewicz dengan menambahkan operator elitism dan membalik proses seleksi setelah proses reproduksi. Siklus yang telah diperbarui ini dapat dilihat pada gambar 2.



Gambar 2 Siklus Algoritma Genetika yang diperbarui oleh Michalewicz. Sumber: (Syah, t.t.)

2.3.2 Kromosom, Populasi, Fitness, Crossover dan Mutasi

Dalam siklus algoritma genetika terdapat beberapa komponen yang akan terus digunakan selama siklus terjadi yaitu kromosom, populasi, fitness, dan reproduksi yang meliputi *crossover* dan mutasi (Kholik dkk., 2018).

a) Kromosom dan Populasi

Kromosom digunakan untuk merepresentasikan suatu solusi yang mungkin dari permasalahan yang akan diselesaikan menggunakan algoritma genetika. Kromosom bisa dinyatakan dalam banyak cara, seperti kromosom

biner, kromosom real, kromosom permutasi, dan lain sebagainya. Populasi dalam algoritma genetika adalah sekumpulan kromosom. Dalam satu populasi, akan terdapat N buah kromosom dengan nilai N adalah suatu parameter yang ditetapkan oleh pengguna.

b) Fitness

Fungsi fitness suatu kromosom adalah nilai seberapa cocok kromosom tersebut untuk menyelesaikan suatu masalah. Semakin tinggi nilai fitness, semakin optimal solusi kromosom tersebut.

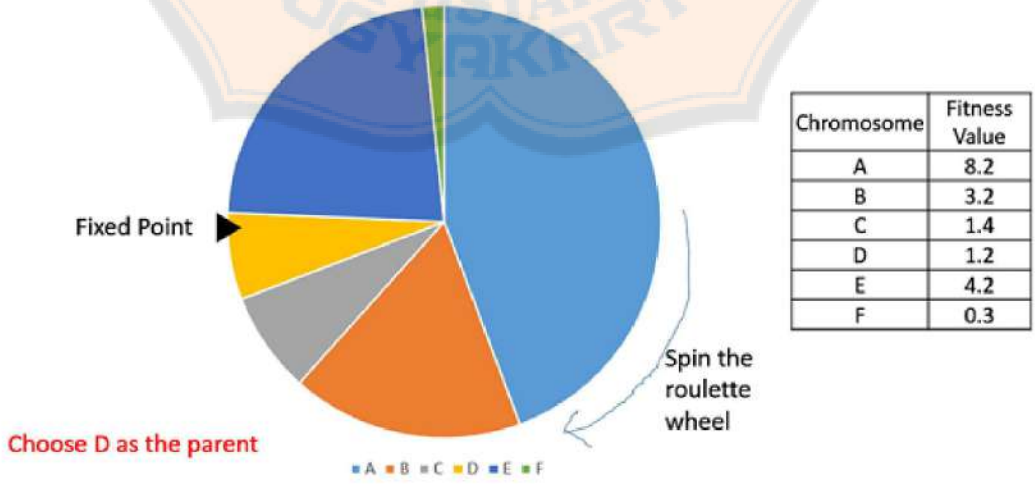
c) *Crossover* dan Mutasi

Proses *crossover* adalah proses pembentukan dua keturunan baru (kromosom anak) dari parent yang terpilih. Keturunan yang memenuhi persyaratan masalah akan ditambahkan ke populasi yang ada. Proses mutasi adalah mengganti gen dengan yang baru. Proses *crossover* dan mutasi biasanya berlangsung secara acak.

2.3.3 Seleksi Parent

Seleksi parent digunakan untuk memilih individu yang akan digunakan untuk proses kawin silang dan mutasi nanti. Proses seleksi yang sering digunakan adalah sebagai berikut (Shyalika, 2019):

a. *Roulette Wheel*

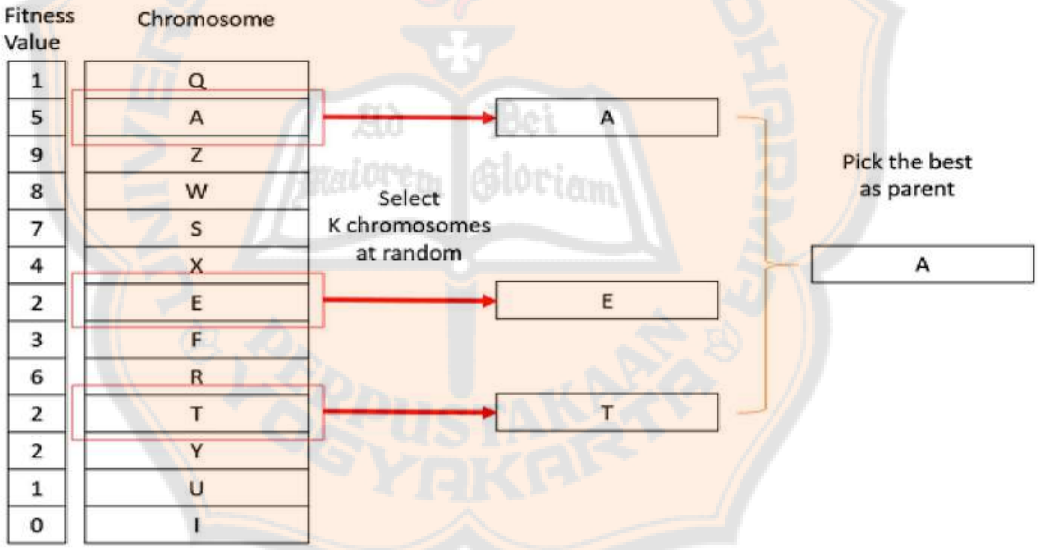


Chromosome	Fitness Value
A	8.2
B	3.2
C	1.4
D	1.2
E	4.2
F	0.3

Gambar 3 *Roulette Wheel*. Sumber: *(Genetic Algorithms - Parent Selection, t.t.)*

Pada *roulette wheel selection*, setiap kromosom dalam suatu populasi memiliki tempat yang sesuai dengan proporsinya terhadap keseluruhan nilai fitness pada suatu populasi, sehingga probabilitas terpilihnya kromosom tersebut akan semakin besar apabila nilai fitness juga besar atau dengan kata lain probabilitas terpilihnya parent sangat tergantung dari nilai fitnessnya. Lalu kromosom-kromosom ini nantinya akan diletakkan pada suatu bagian pada roulette wheel secara terurut dan besarnya bagian ini tergantung dari nilai fitness kromosom tersebut. Cara pemilihan ini akan dilakukan dengan sebuah pointer yang secara acak memilih bagian dari *roulette wheel* tersebut, dan bagian yang terpilih akan menjadi induk dari kromosom selanjutnya.

b. Turnamen *Selection*

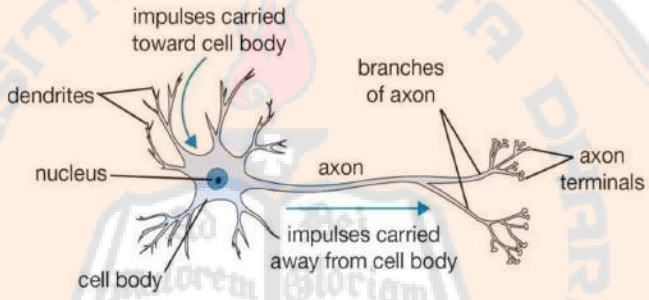


Gambar 4 *Turnamen Selection*. Sumber: *(Genetic Algorithms - Parent Selection, t.t.)*

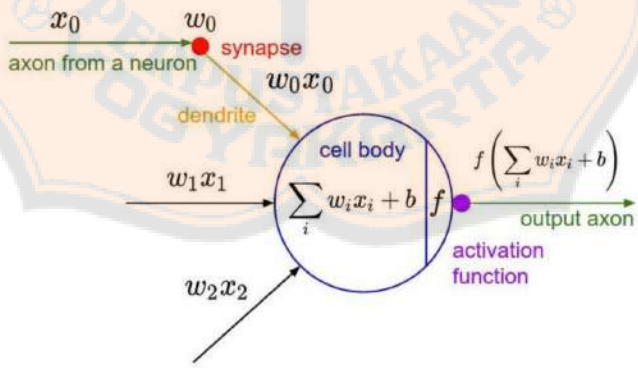
Pada metode seleksi turnamen akan dipilih beberapa individu secara random yang kemudian beberapa individu tersebut akan dipilih individu dengan nilai fitness terbaik. Individu yang mengikuti turnamen ini akan dipilih antara 2 sampai *n* (jumlah individu dalam populasi).

2.4 Artificial Neural Network

Artificial Neural Network (ANN) atau jaringan syaraf tiruan adalah sebuah metode atau teknik pendekatan informasi yang terinspirasi dari sistem syaraf otak manusia. Bagian utama dari ANN adalah struktur system informasi yang unik dan beragam untuk setiap aplikasi. ANN sendiri terdiri dari beberapa bagian syaraf (neuron) yang saling terhubung dan bekerja secara bersama-sama yang bertujuan untuk menyelesaikan suatu permasalahan, permasalahan yang umum diselesaikan dengan ANN adalah klasifikasi atau prediksi. Gambar dibawah ini adalah struktur ANN secara mendasar (Sena, 2017).



Gambar 5 Syaraf Manusia. Sumber: (Sena, 2017)



Gambar 6 Struktur *Artificial Neural Network*. Sumber: (Sena, 2017)

Proses pada ANN adalah sebagai berikut (Suhartono, 2012):

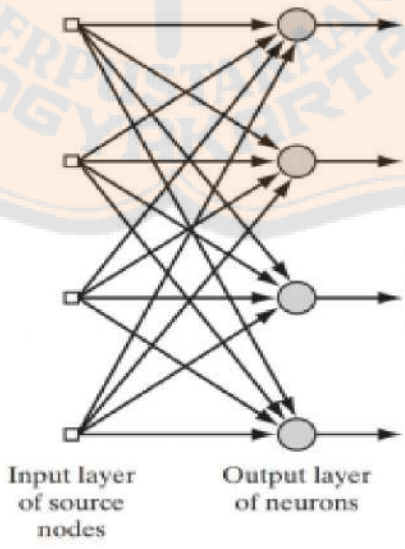
1. Setiap *neuron* akan menerima *Input* dan bobot (W) dari setiap *input*.

2. Setelah itu *Input* yang telah masuk kedalam *Neuron* akan dikali dengan bobot(W) dan dijumlahkan dengan bias.
3. Lalu hasil ini akan dijumlahkan untuk semua *neuron*, seperti yang dilambangkan pada gambar yaitu lambang sigma(∑).
4. Hasil penjumlahan ini selanjutnya akan diproses oleh fungsi aktivasi *neuron*, pada tahap ini akan dibandingkan hasil penjumlahan dengan nilai ambang (*threshold*) tertentu.
5. Jika hasilnya melebihi *threshold*, maka aktivasi *neuron* akan dibatalkan
6. Jika tidak, maka *neuron* akan diaktifkan.
7. Setelah aktif, maka *neuron* akan meneruskan nilai *output* melalui bobot *output* ke semua neuron yang terhubung
8. Proses ini akan terus berulang hingga pada *output neuron*.

2.4.1 Model Jaringan pada ANN

Beberapa model jaringan pada ANN yang sering digunakan secara umum adalah sebagai berikut (Yanuar, 2018):

a. Single Layer Network

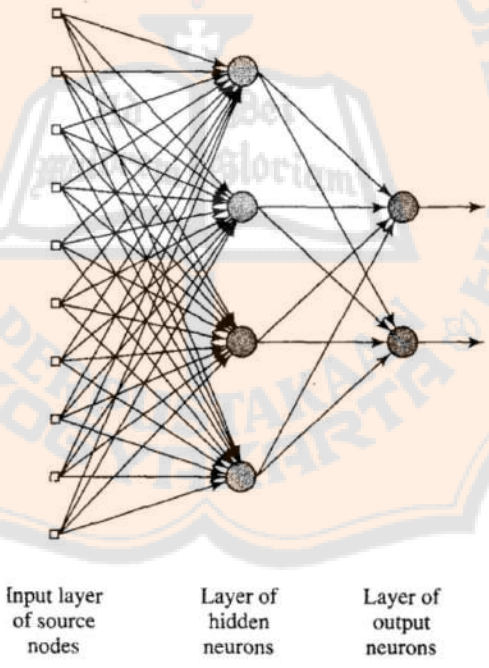


Gambar 7 *Single Layer Network*. Sumber: (Haykin, 1994)

Dalam model jaringan ini ANN terbentuk dari satu lapisan (*layer*) tanpa adanya *hidden layer*, yaitu hanya layer *input* dan *output* saja. Model jaringan ini adalah model jaringan paling sederhana dalam ANN. Cara kerja dari *single layer* adalah nilai dari *input* layer ini akan langsung dihubung dengan *output layer*, tetapi tidak berlaku untuk sebaliknya. Model ini adalah jenis jaringan *feed forward*.

Pada gambar 7 *input* dan *output* memiliki 4 *neuron*, namun yang dimaksud dengan *single layer* yaitu *output* dari jaringan, sedangkan outputnya tidak memiliki pengaruh karena pada saat melakukan *input* tidak terjadi proses komputasi

b. Multilayer Network

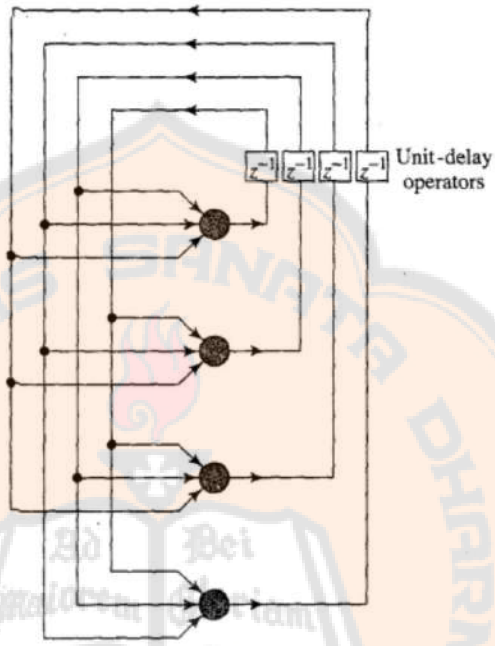


Gambar 8 *Multilayer Network*. Sumber: (Haykin, 1994)

Multilayer network adalah pengembangan dari *single layer network*, dimana akan terdapat satu atau dua *hidden layer* diantara *input* dan *output* layer. Cara kerja *multilayer network* adalah *input layer* akan melakukan proses komputasi dan hasilnya akan dimasukkan kedalam *hidden layer*, baru setelah

itu akan dilakukan komputasi lagi dan hasil dari *hidden layer* tersebut akan diteruskan ke *output* layer.

c. *Recurrent Network*



Gambar 9 *Recurrent Network*. Sumber: (Haykin, 1994)

Recurrent network terbentuk karena pada jaringan *single layer* dan *multilayer* harus memiliki *feedback* untuk dirinya sendiri pada setiap perulangan jaringannya. Pada *recurrent network* jaringan tidak memerlukan *feedback* untuk dirinya sendiri melainkan *feedback* dari *input* yang digunakan.

2.4.2 Fungsi Aktivasi dan Bias

Fungsi aktivasi berguna untuk menentukan *neuron* mana yang diizinkan untuk aktif pada *output* layer. Berikut ini adalah beberapa fungsi aktivasi yang umum digunakan (Baheti, 2023):

a. *Binary Step Function*

$$f(x) = \begin{cases} 1 & ,jika\ x \geq 0 \\ 0 & ,jika\ x < 0 \end{cases} \quad (1)$$

b. *Symmetric Hard Limit Function*

$$f(x) = \begin{cases} 1 & ,jika\ x \geq 0 \\ -1 & ,jika\ x < 0 \end{cases} \quad (2)$$

c. *Sigmoid Function*

$$f(x) = \frac{1}{1 + e^{-x}} \quad (3)$$

d. *Piecewise Linear Function*

$$f(x) = \begin{cases} 1 & ,\text{jika } x \geq 1 \\ x & ,\text{jika } 0 > x > 1 \\ 0 & ,\text{jika } x \leq 0 \end{cases} \quad (4)$$

e. *Linear Function*

$$f(x) = x \quad (5)$$

f. *Rectified Linear Unit Function* (ReLU)

$$f(x) = \max(0, x) \quad (6)$$

g. *Softmax*

$$softmax(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)} \quad (7)$$

Jika jumlah bobot sama dengan nol, maka bias ditambahkan untuk membuat output tidak nol atau sesuatu yang lain untuk meningkatkan respons sistem. Bias memiliki input yang sama, dan bobot sama dengan 1 (Trivusi, 2022), dengan melibatkan bias ini maka output ANN menjadi:

$$ANN = b + \sum_i^0 x_i w_i \quad (8)$$

BAB III

METODE PENELITIAN

3.1 ALUR KERJA PENELITIAN

Penelitian ini bertujuan untuk mengetahui perkiraan harga penutupan pada hari tersebut. Data yang digunakan dalam penelitian ini adalah daftar historis harga yang diambil dari API yahoo *finance*. Data yang telah diambil selanjutnya akan dilakukan *pre-processing* data terlebih dahulu untuk menghilangkan atribut data yang tidak digunakan dan normalisasi data. Selanjutnya data yang telah melalui *pre-processing* akan dilakukan proses pelatihan metode ANN-GA. Setelah pelatihan selesai, maka proses terakhir adalah pengujian model ANN-GA yang akan menghasilkan tingkat akurasi berupa MSE dan harga penutupan hari tersebut. Alur kerja penelitian dapat dilihat pada gambar berikut.

Gambar 10 Alur Kerja Penelitian

3.2 DATA PENELITIAN

3.2.1 Jenis Data

Data yang digunakan dalam penyelesaian tugas akhir ini adalah data historis Eropa Euro banding US Dollars.

3.2.2 Pengumpulan Data

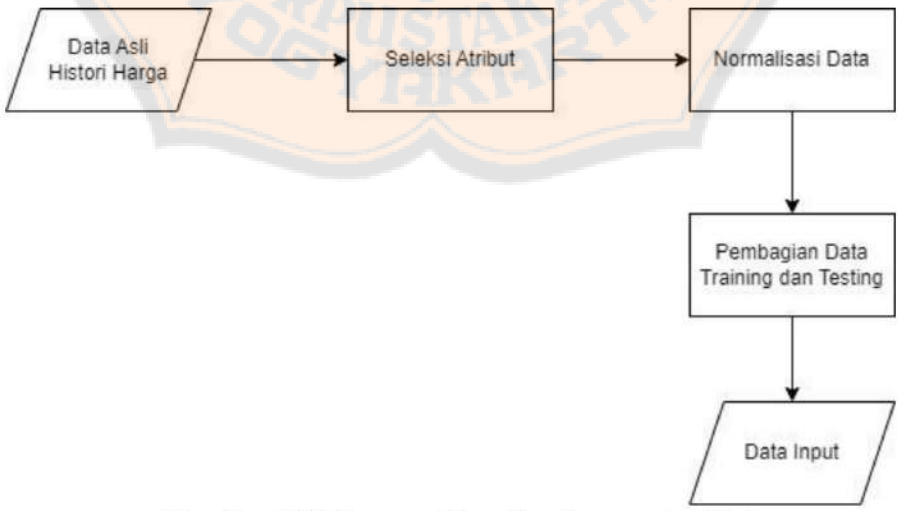
Data yang digunakan berasal dari API website finance.yahoo.com yang terdiri dari 2019 dataset. Dataset ini didapatkan dari harga per hari dari tanggal 1 Januari 2015 s/d 1 Oktober 2022, dan akan dibagi dengan metode *random sampling replacement* dengan *rasio* 70% *training* dan 30% *testing*. Dengan total data

sebanyak 2019 dataset, maka training data yang diperoleh adalah 1413 dan 606 testing data. Berikut adalah contoh dataset yang diambil.

Date	High	Low	Open	Close	Volume	Adj Close
2015-01-01	1.209863	1.209863	1.209863	1.209863	0.0	1.209863
2015-01-02	1.208956	1.201080	1.208868	1.208941	0.0	1.208941
2015-01-05	1.197590	1.188909	1.195500	1.194643	0.0	1.194643
2015-01-06	1.197000	1.188693	1.193830	1.193902	0.0	1.193902
2015-01-07	1.190000	1.180401	1.187479	1.187536	0.0	1.187536
2015-01-08	1.184806	1.175601	1.183894	1.183600	0.0	1.183600
2015-01-09	1.183830	1.176831	1.179426	1.179607	0.0	1.179607
2015-01-12	1.187200	1.178800	1.187014	1.187070	0.0	1.187070
2015-01-13	1.185902	1.175650	1.183236	1.183152	0.0	1.183152
2015-01-14	1.184357	1.173060	1.177676	1.177829	0.0	1.177829
2015-01-15	1.178856	1.159229	1.178384	1.178592	0.0	1.178592
2015-01-16	1.165000	1.146430	1.163900	1.163738	0.0	1.163738

Gambar 11 Daftar Historis Harga

3.3 *PRE-PROCESSING*



Gambar 12 Diagram Alur *Pre-Processing* Data

Tahap pertama pada *pre-processing* setelah data terkumpul adalah proses seleksi atribut. Proses ini dilakukan untuk menghilangkan variable yang kurang relevan dengan output yang diinginkan. Atribut ini bisa berupa tanggal atau atribut dengan nilai yang selalu bernilai 0 atau tidak bernilai. Lalu setelah proses seleksi atribut selesai, proses selanjutnya adalah normalisasi data. Pada tahap ini, data akan diubah nilai aslinya menjadi nilai tertentu. Untuk metode normalisasi yang digunakan adalah metode *min-max scaler,* dengan metode ini nilai data akan diubah menjadi nilai yang berada diantara 0 – 1. Proses ini akan menggunakan bantuan dari *library tensorflow* dari *google* dengan nama metode *min-max scaler*. Metode ini dilipih oleh penulis karena metode ini cukup sederhana dan tidak sulit untuk diimplementasikan kedalam data.

Setelah proses normalisasi data selesai akan dilanjutkan dengan proses pembagian data *training* dan data *testing*. Proses pembagian data akan menggunakan metode *random sampling without replacement* dengan bantuan dari library *sklearn* dengan nama metode *model selection*. Hasil dari proses pembagian data adalah data *training* dan data testing. Hasil akhir dari semua proses yang telah dilakukan adalah data dengan total variable 4, yaitu variable *close*, *open*, *high*, dan *low* dan nilai data antara 0 hingga 1 dengan pembagian data 70% *training* dan 30% *testing*. Data inilah yang akan menjadi input dari program ini.

3.4 PELATIHAN JARINGAN ANN-GA

3.4.1 Perancangan Jaringan ANN-GA

Setelah dilakukan *pre-processing*, maka tahap selanjutnya adalah *training* jaringan ANN dengan algoritma genetika. Tahap pertama dalam merancang jaringan ANN adalah memilih arsitektur jaringan yang akan digunakan. Pada penelitian ini, arsitektur yang akan digunakan adalah arsitektur jaringan dengan banyak lapisan (*multilayer network).* Dalam arsitektur ini akan terdapat beberapa parameter yang akan digunakan antara lain sebagai berikut:

1. Lapisan Tersembunyi (*Hidden Layer)* : Pada penelitian ini akan digunakan 1, 2, dan 3 lapisan *hidden layer*.

2. Unit Tersembunyi (*Neuron*) : Jumlah *neuron* yang digunakan dalam *hidden layer* pertama adalah 5, pada *hidden layer* kedua 10, dan *neuron* pada *hidden layer* ketiga 10.
3. Fungsi Aktivasi : Fungsi Aktivasi yang digunakan dalam penelitian ini adalah fungsi ReLU pada *hidden layer*.
4. Bobot Awal : Bobot awal yang akan digunakan adalah nilai random dari -10 hingga 10.

Setelah parameter ANN ditetapkan selanjutnya untuk parameter algoritma genetika yang akan digunakan adalah sebagai berikut:

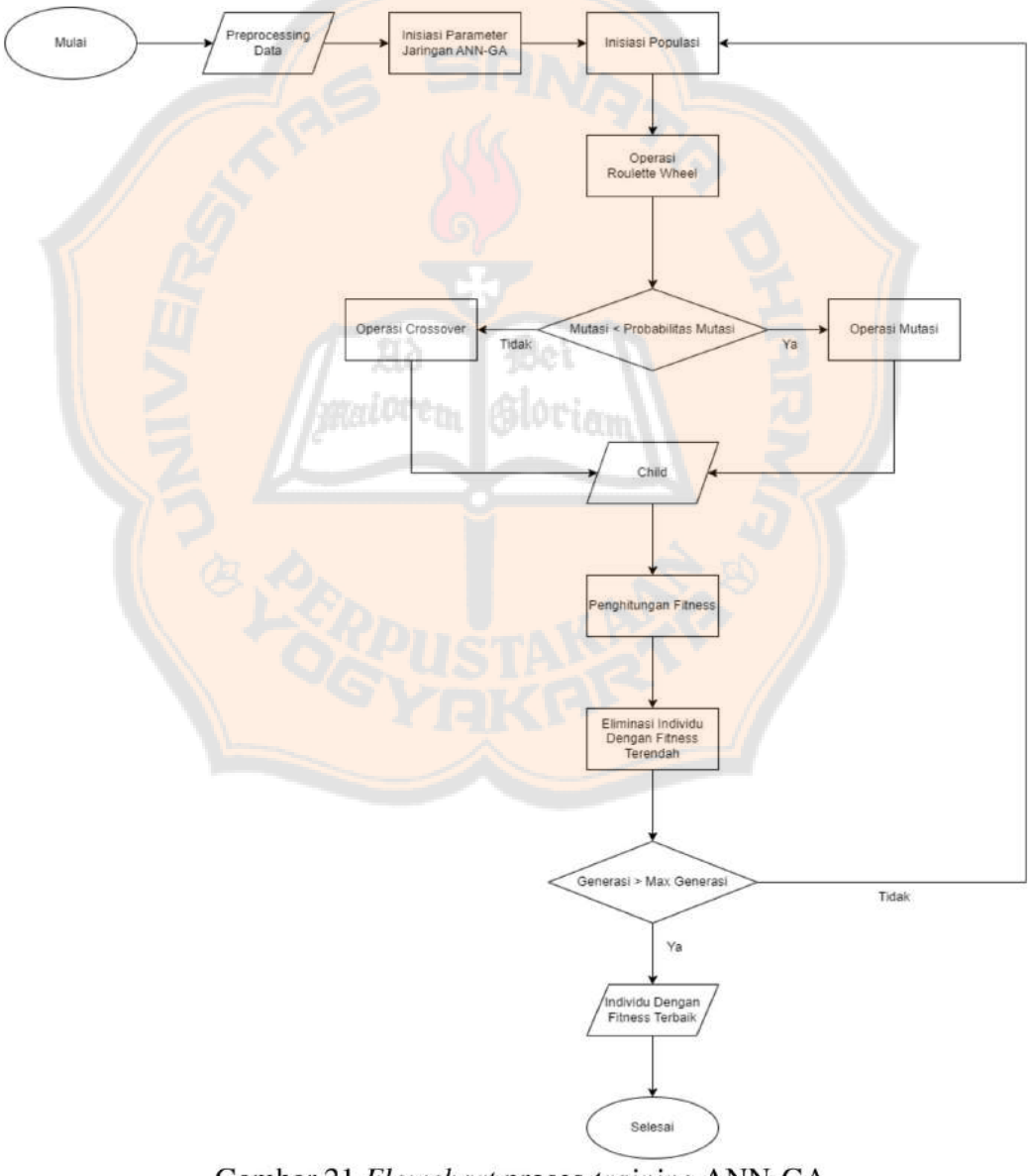
1. Generasi: Pada penilitian ini generasi yang akan digunakan adalah dari 10 generasi hingga 100 generasi.
2. Jumlah *Parent*: Jumlah *parent* yang dipilih adalah 4 parent.
3. Persentase Mutasi: Persentase mutasi adalah 50%.
4. Jumlah Solusi: Jumlah solusi yang digunakan adalah 10.
5. Jenis Pemilihan *Parent*: Pemilihan *parent* akan menggunakan jenis *roulette wheel selection*.
6. Tipe *Crossover*: *Crossover* menggunakan tipe *uniform*.
7. Tipe Mutasi: Tipe mutasi yang digunakan adalah *random mutation*.

Sebelum jaringan ANN dapat sepenuhnya melakukan prediksi, maka setelah arsitektur jaringan dan semua parameter ditetapkan, jaringan ANN akan melalui dua tahap terlebih dahulu yaitu tahap pelatihan (*training*) dan tahap pengujian (*testing*).

3.4.2 Pelatihan Jaringan

Tahap pelatihan berguna untuk memaksimalkan parameter yang ada sebelumnya dan untuk mencari kombinasi yang cocok antar parameter, sehingga didapatkan nilai akurasi yang paling tinggi diantara semua kombinasi parameter yang ada. Tahap pelatihan ini akan terbagi menjadi dua macam tahap, yaitu tahap pelatihan jaringan ANN dengan *forward propagation* dan tahap pengoptimalan bobot pada layer ANN dengan algoritma genetika.

Tahap pelatihan jaringan ANN dengan *forward propagation* akan didapatkan nilai tingkat keakurasian dari model ANN. Lalu setelah itu akan dilanjutkan dengan pengoptimalan bobot hidden layer dengan menggunakan algoritma genetika. Dari proses pengoptimalan ini akan didapatkan kombinasi nilai bobot *hidden layer* dengan nilai fitness tertinggi serta nilai *error* terendah. Data yang digunakan dalam proses pelatihan adalah data *training* sebanyak 1413 data. Langkah – langkah yang dilakukan selama pelatihan jaringan yaitu sebagai berikut:



Gambar 21 *Flowchart* proses *training* ANN-GA

Berikut penjelasan dari setiap langkah pada gambar 21:

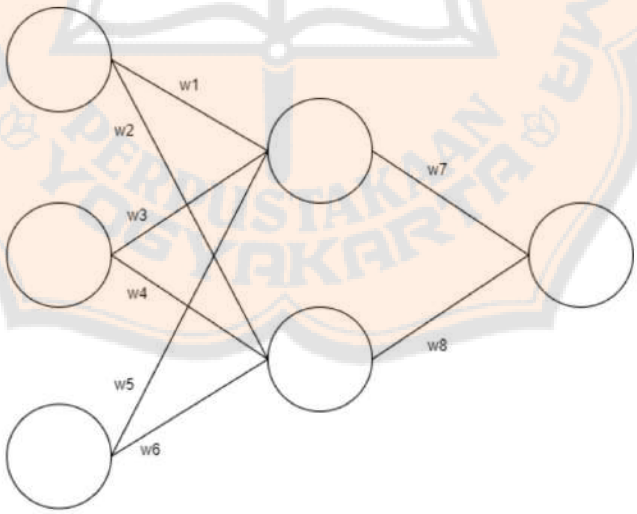
1. Inisiasi Parameter Jaringan ANN-GA

   Pada tahap ini setiap parameter pada ANN dan algoritma genetik akan diinisiasi dengan nilai yang telah disiapkan sebelumnya. Parameter ini meliputi parameter ANN dan algoritma genetika yaitu jumlah *hidden layer*, jumlah *neuron*, fungsi aktivasi, bobot awal yang digunakan, batas generasi, jumlah *parent,* persentase mutasi, jumlah solusi, jenis pemilihan *parent*, jenis *crossover,* dan tipe mutasi.

2. Inisiasi Populasi

   Selanjutnya setelah semua parameter terisi adalah tahap pembentukan populasi awal. Populasi berisi sekumpulan dari kromosom yang terbentuk. Setiap kromosom berisi nilai bobot jaringan ANN yang ada. Nilai ini akan diambil dari jaringan ANN yang telah melalui tahap *forward propagation*. Berikut adalah contoh dari sebuah kromosom yang terbentuk dari jaringan ANN.

Gambar 13 Contoh jaringan ANN



Tabel 2 Contoh kromosom

W1	W2	W3	W4	W5	W6	W7	W8

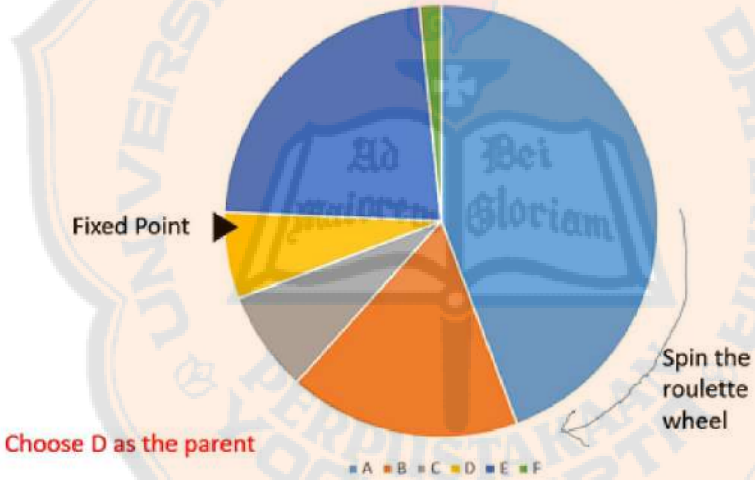
Tahap inisiasi populasi akan terus berjalan selama batas generasi belum tercapai. Saat inisiasi populasi pertama kali dijalankan, setiap kromosom akan berisi nilai *random* dengan nilai dari -10 hingga 10. Untuk inisiasi populasi selanjutnya, populasi akan berisi kromosom yang telah terpilih dari generasi sebelumnya. Inisiasi populasi akan menggunakan *library python* bernama *pygad* dengan nama kelas *initialize population*. Berikut contoh sebuah populasi dari contoh kromosom acak.

Populasi							
W1	W2	W3	W4	W5	W6	W7	W8
W2	W1	W3	W4	W5	W6	W7	W8
W3	W2	W1	W4	W5	W6	W7	W8
W4	W2	W3	W1	W5	W6	W7	W8
W5	W2	W3	W4	W1	W6	W7	W8
W6	W2	W3	W4	W5	W1	W7	W8
W7	W2	W3	W4	W5	W6	W7	W8
W8	W2	W3	W4	W5	W6	W7	W1
W1	W3	W2	W4	W5	W1	W7	W8
W1	W4	W3	W2	W5	W1	W7	W8

Gambar 14 Contoh populasi

3. Operasi *Roulette Wheel*

Pemilihan induk dalam penelitian ini akan dilakukan menggunakan metode *roulette wheel*. Metode ini menggunakan konsep seperti roda putar, dimana individu akan dipilih pada titik dimana roda tersebut berhenti berputar. Dalam metode ini bagian atau probabilitas dari setiap individu tidak sama. Besarnya probabilitas akan sebanding dengan nilai fitness individu tersebut, semakin tinggi nilai fitness akan semakin besar pula probabilitas yang didapat. Sebagai titik pemilih roda atau dalam hal ini pemilih individu akan menggunakan nilai acak dengan jarak nilai tertentu. Dalam beberapa contoh, titik pemilih individu dapat bernilai dari 0 hingga 1. Sebagai contoh berikut adalah gambaran umum dari *roulette wheel selection.*



Chromosome	Fitness Value
A	8.2
B	3.2
C	1.4
D	1.2
E	4.2
F	0.3

Gambar 15 Gambaran umum *roulette wheel selection.* Sumber: *(Genetic Algorithms - Parent Selection, t.t.)*

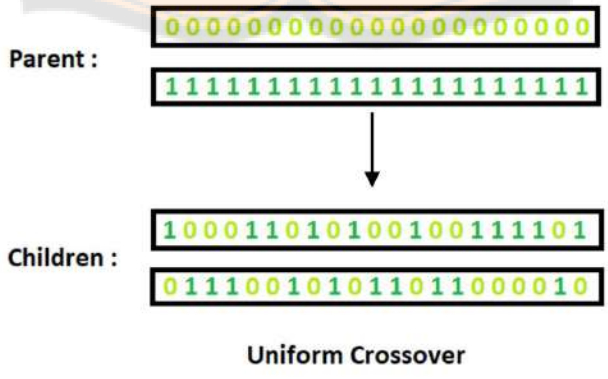
Pada penelitian ini *roulette wheel* akan memilih empat individu untuk dijadikan induk. Proses pemilihan induk menggunakan operasi *roulette wheel* akan dilakukan dengan bantuan *library python* bernama *pygad* dengan kelas *roulette wheel selection*. Jika proses selanjutnya berupa *crossover* maka keempat individu ini akan digunakan, tetapi jika operasi selanjutnya adalah mutasi maka salah satu dari keempat individu inilah yang akan digunakan dalam mutasi tersebut.

4. Syarat Terjadinya Mutasi

Mutasi dalam genetika tidak selalu terjadi, sehingga terjadinya mutasi akan ditentukan menggunakan probabiltas mutasi. Probabilitas ini akan menentukan seberapa sering mutasi terjadi dalam proses genetika. Probabilitas ini akan ditentukan oleh penulis yaitu sebesar 50%. Setelah probabilitas mutasi ditentukan maka dalam setiap iterasi akan dipilih nilai acak dari 1 hingga 100, proses mutasi akan dilakukan apabila nilai acak ini kurang atau sama dengan nilai probabilitas mutasi yang ditentukan.

5. Operasi *Crossover*

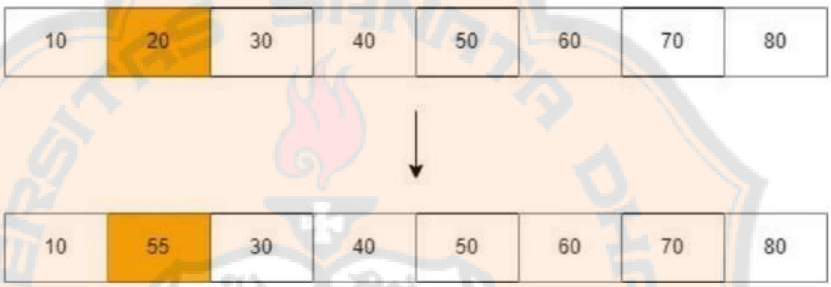
Operasi *crossover* berguna untuk mendapatkan kromosom baru dengan menggabungkan beberapa kromosom yang ada dengan harapan mendapatkan kromosom baru dengan nilai fitness lebih tinggi dari kromosom sebelumnya. Proses *crossover* pada penelitian ini akan menggunakan empat *parent* yang telah terpilih dari operasi *roulette wheel*. Ke-empat *parent* akan saling menukarkan isi kromosomnya untuk mendapatkan anak atau kromosom baru yang berbeda dengan induknya. Tipe *crossover* yang akan digunakan dalam penelitian ini adalah *uniform crossover*. Dalam tipe *crossover* ini tiap gen dalam kromosom induk pertama akan diundi, apakah gen tersebut ditukar dengan gen dari induk kedua atau tidak. Proses *crossover* akan dilakukan menggunakan bantuan dari *library python* bernama *pygad* dengan modul GA. Berikut (gambar 16) adalah contoh operasi *crossover*.



Gambar 16 Contoh operasi *crossover* dengan 2 individu

6. Operasi Mutasi

Operasi mutasi berguna untuk mendapatkan satu kromosom baru yang berbeda dari kromosom sebelumnya. Jenis mutasi bermacam macam tetapi dalam penelitian ini jenis mutasi yang digunakan adalah *random mutation*. Dalam mutasi jenis ini, salah satu gen dalam kromosom akan dipilih secara acak dan nilai gen yang terpilih tersebut akan diganti dengan nilai acak dengan rentang nilai tertentu. Dengan menggunakan cara ini diharapkan kromosom yang terbentuk akan berbeda dari kromosom sebelumnya. Berikut adalah ilustrasi dari *random mutation* dengan kromosom acak.



Gambar 17 Contoh *random mutation*

7. Penghitungan Fitness

Pada tahap ini akan dihitung semua akurasi dari child yang telah terbentuk. Proses penghitungan ini menggunakan prinsip dari *forward propagation* dimana output dari proses ini akan menghasilkan nilai prediksi yang akan dibandingkan dengan nilai asli dengan menghitung nilai *error* menggunakan *Mean Square Error* (MSE). Berikut rumus MSE yang digunakan.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2 \qquad (9)$$

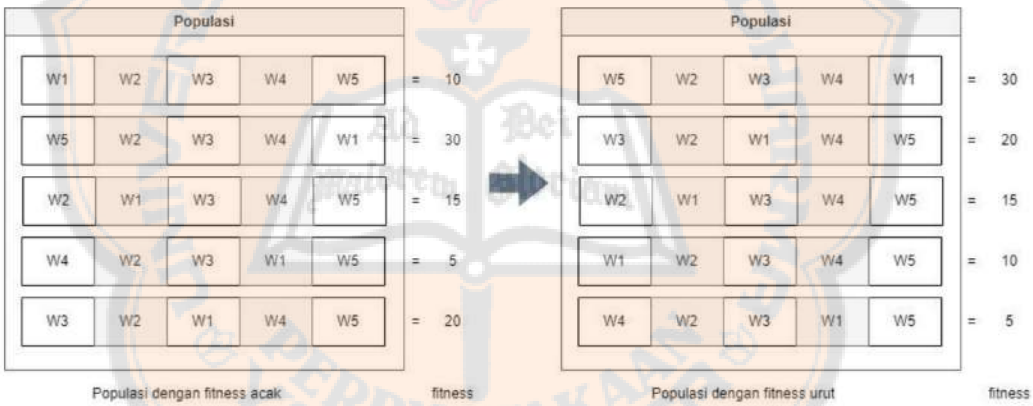
Dimana y_i adalah nilai asli output dan $\tilde{y_i}$ merupakan nilai prediksi. Dari kedua nilai ini akan dipangkat 2 dan dicari nilai rata – ratanya. Hasil MSE yang akan dijadikan nilai fitness dalam algoritma genetika. Untuk menghitung nilai fitness dari hasil MSE akan digunakan rumus sebagai berikut:

$$fitness = \frac{1}{MSE} \tag{10}$$

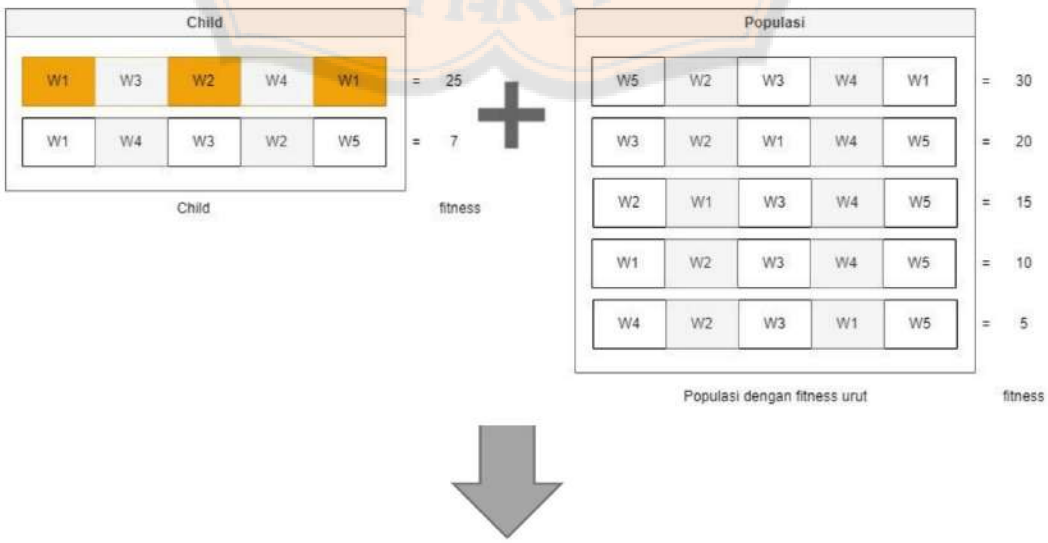
Dari rumus fitness tersebut dapat disimpulkan bahwa semakin tinggi nilai fitness yang dihasilkan maka akan semakin kecil nilai MSE yang terbentuk.

8. Eliminasi Kromosom dengan Fitness Terendah

Setelah setiap individu baru diketahui nilai fitnessnya maka selanjutnya adalah melakukan eliminasi individu yang mempunyai nilai fitness terendah. Dalam pengeliminasian ini suatu populasi akan diurutkan terlebih dulu dari fitness tertinggi hingga terendah. Dari pengurutan ini maka akan diketahui kromosom dengan fitness terendah dan akan digantikan dengan kromosom dari *child* dengan fitness lebih tinggi. Berikut adalah ilustrasi dari proses pengiliminasian kromosom dengan fitness terendah.



Gambar 18 Proses pengurutan kromosom



Gambar 19 Proses pengeliminasian dan penambahan *child* ke populasi

9. Syarat Penghentian Proses Genetika Algoritma

   Proses pengoptimalan dengan genetika algoritma akan terhenti apabila suatu generasi telah mencapai batas generasi yang ditentukan. Misalkan batas generasi adalah 100, maka saat generasi mencapai 100 proses pengoptimalan akan terhenti. Tetapi apabila belum, maka proses pengoptimalan akan terus berlanjut dan akan mengulangi dari langkah nomor 2.

10. Pemilihan Kromosom dengan Fitness Terbaik

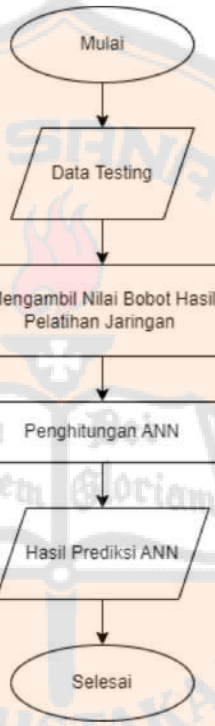
    Setelah proses pengoptimalan selesai, maka akan dilanjutkan dengan memilih kromosom dengan nilai fitness terbaik. Proses pencarian kromosom terbaik akan dilakukan dengan membandingkan setiap kromosom yang ada pada populasi. Hasil dari pencarian ini akan menyimpan kromosom dengan fitness tertinggi ke dalam sebagai kromosom terbaik.

11. Hasil Akhir

    Hasil dari proses pengoptimalan akan digunakan untuk proses pengujian nanti. Hasil ini berupa kromosom dengan nilai fitness terbaik diantara yang lain. Kromosom ini akan berisi nilai bobot layer dan nilai fitness. Nilai bobot ini yang akan digunakan sebagai bobot layer dalam jaringan ANN pada proses *testing*.

3.5 PENGUJIAN JARINGAN ANN-GA

Tahap pengujian ini berguna untuk mengetahui tingkat akurasi jaringan yang telah melalui proses pelatihan genetika algoritma terhadap data baru. Data yang digunakan pada tahap ini adalah data *testing* sebanyak 30% dari total data atau lebih tepatnya 606 data. Langkah langkah yang dilakukan dalam proses *testing* (gambar 20) adalah sebagai berikut:



Gambar 20 *Flowchart* proses *testing* ANN

Tahap pertama pada proses *testing* adalah mengambil nilai bobot hasil pelatihan jaringan ANN. Nilai bobot ini diambil dari kromosom terbaik yang telah dihasilkan dari proses pelatihan ANN. Selanjutnya adalah tahap penghitungan ANN. Tahap ini akan menghitung atau memprediksi nilai harga penutupan pada hari tersebut menggunakan perambatan maju (*feed forward*) dan mengubah nilai hasil prediksi menjadi nilai ril. Proses ini akan dilakukan sebanyak data *testing* yaitu 606 kali. Banyaknya data testing merepresentasikan banyaknya hari yang diprediksi sehingga hasil dari proses ini adalah harga penutupan sebanyak 606 hari. Hasil ini masih berupa skala dari 0 hingga 1 yang

dihasilkan dari proses normalisasi pada *pre-processing*, sehingga tahap akhir adalah mengubah kembali nilai skala tersebut menjadi nilai ril dan akan dihitung akurasinya menggunakan nilai *error* berupa MSE.

3.6 PERCOBAAN ANN-GA

Pada tahap ini hasil akurasi dari pengujian ANN-GA akan dibandingkan dengan mengubah parameter *hidden layer* dan generasi. Akan digunakan 1 *hidden layer*, 2 *hidden layer* dan 3 *hidden layer*. Selanjutnya percobaan akan dibagi dengan menggunakan 10 hingga 100 generasi dimana setiap generasi akan dilakukan percobaan sebanyak 5 kali untuk menentukan nilai rata-rata MSE. Akurasi yang didapat berupa besaran MSE untuk masing – masing metode. Semakin kecil nilai MSE, maka semakin baik metode tersebut.

Dari percobaan yang telah dilakukan akan dipilih salah satu hidden layer terbaik yang kemudian akan dilakukan percobaan dengan mengubah jenis fungsi aktivasi menggunakan *softmax*. Akhir dari percobaan ini akan didapat sebuah jaringan ANN-GA dengan hidden layer terbaik serta salah satu fungsi aktivasi terbaik dari jenis *softmax* atau relu.

BAB IV

HASIL DAN ANALISIS

4.1 HASIL

4.1.1 *Pre-processing*

Dalam penelitian yang penulis lakukan, data harga historis Euro/Usd yang diambil dari *API* yahoo *finance*. Data ini berisi data harga perhari dari 1 Januari 2015 s/d 1 Oktober 2022 dengan total data 2019 dan memiliki 7 atribut, tentu ke-6 atribut ini tidak akan digunakan semua. Atribut 1 dan 6 adalah tanggal dan harga penutupan yang telah disesuaikan yang nilainya sama dengan atribut harga penutupan akan dihilangkan, karena atribut ini tidak akan mempengaruhi harga terakhir. Maka atribut utama yang akan digunakan dalam penelitian ini adalah harga pembukaan, harga terendah, harga tertinggi, dan harga terakhir. Tabel data sebelum dan sesudah dihilangkan atribut 1 dan 6 dapat dilihat pada gambar berikut ini:

Date	High	Low	Open	Close	Volume	Adj Close
2015-01-01	1.209863	1.209863	1.209863	1.209863	0.0	1.209863
2015-01-02	1.208956	1.201080	1.208868	1.208941	0.0	1.208941
2015-01-05	1.197590	1.188909	1.195500	1.194643	0.0	1.194643
2015-01-06	1.197000	1.188693	1.193830	1.193902	0.0	1.193902
2015-01-07	1.190000	1.180401	1.187479	1.187536	0.0	1.187536
2015-01-08	1.184806	1.175601	1.183894	1.183600	0.0	1.183600
2015-01-09	1.183830	1.176831	1.179426	1.179607	0.0	1.179607
2015-01-12	1.187200	1.178800	1.187014	1.187070	0.0	1.187070
2015-01-13	1.185902	1.175650	1.183236	1.183152	0.0	1.183152
2015-01-14	1.184357	1.173060	1.177676	1.177829	0.0	1.177829
2015-01-15	1.178856	1.159229	1.178384	1.178592	0.0	1.178592
2015-01-16	1.165000	1.146430	1.163900	1.163738	0.0	1.163738

Gambar 21 Atribut 1 dan 6 sebelum dihilangkan

High	Low	Open	Close
1.209863	1.209863	1.209863	1.209863
1.208956	1.201080	1.208868	1.208941
1.197590	1.188909	1.195500	1.194643
1.197000	1.188693	1.193830	1.193902
1.190000	1.180401	1.187479	1.187536
1.184806	1.175601	1.183894	1.183600
1.183830	1.176831	1.179426	1.179607
1.187200	1.178800	1.187014	1.187070
1.185902	1.175650	1.183236	1.183152
1.184357	1.173060	1.177676	1.177829
1.178856	1.159229	1.178384	1.178592
1.165000	1.146430	1.163900	1.163738

Gambar 22 Atribut 1 dan 6 sesudah dihilangkan

Atribut harga terakhir akan menjadi atribut yang akan diprediksi atau *output program* dan atribut harga pembukaan, harga tertinggi, dan harga terendah akan menjadi *input* dari *program*.

Setelah menghilangkan atribut yang kurang relevan, selanjutnya akan dilakukan normalisasi data menggunakan metode *min-max*. Normalisasi data akan menghasilkan nilai dari skala 0 hingga 1. Dalam melakukan normalisai data penulis menggunakan *library python* dari *Scikit-Learn*. Penulis menggunakan metode *MinMaxScaler*() yang di *import* dari modul *preprocessing*.

Langkah *preprocessing* selanjutnya adalah pembagian data menjadi data *testing* dan data *training*. Proses ini menggunakan *library python* dari *Scikit-Learn* dengan *method* train_test_split() yang di *import* dari modul *model_selection*. *Method* ini menggunakan *metode random sampling without replacement* dengan jumlah test data sebesar 30%. *Source code* yang digunakan untuk normalisasi dan pembagian data dapat dilihat pada gambar berikut:

```
from sklearn.preprocessing import MinMaxScaler
PredictorScaler=MinMaxScaler()
TargetVarScaler=MinMaxScaler()

PredictorScalerFit=PredictorScaler.fit(X)
TargetVarScalerFit=TargetVarScaler.fit(y)

X=PredictorScalerFit.transform(X)
y=TargetVarScalerFit.transform(y)

X_train, X_test, y_train_sgd, y_test_sgd = train_test_split(X, y, test_size=0.3, shuffle=False)
```

Gambar 23 *Source code* normalisasi dan pembagian data

Hasil akhir dari proses *pre-processing* adalah data dengan 3 atribut sebagai data *input* dan 1 atribut *output*. Data ini juga telah diubah kedalam skala antara 1 hingga 0 dengan pembagian data *training* dan *testing*. Berikut adalah hasil akhir dari proses *pre-processing*.

```
input train :  [[0.8580338  0.84091095 0.87909311]
 [0.85462365 0.837771   0.84891503]
 [0.8087871  0.79841698 0.80709678]
 [0.80306141 0.79637211 0.80635376]
 [0.78128516 0.77213501 0.77786038]
 [0.76899301 0.75414974 0.76136762]
 [0.75367246 0.75076956 0.76559352]
 [0.77968984 0.76243943 0.77236225]
 [0.76673551 0.75794476 0.76153884]
 [0.74767169 0.75259483 0.75263934]
 [0.75009841 0.73354751 0.7051157 ]
 [0.70043846 0.68557069 0.66113679]
 [0.6746414  0.68165927 0.69196328]
 [0.68604495 0.67379516 0.69718042]
 [0.66878124 0.69041663 0.68843412]
 [0.6928816  0.68310934 0.6438539 ]
 [0.60014044 0.58999476 0.54345308]
 [0.52962107 0.55883796 0.54998834]
 [0.56739923 0.60592938 0.5792575 ]
 [0.60717166 0.59277394 0.60736462]]
```

```
Target train :  [[0.85881732]
 [0.85565484]
 [0.80658491]
 [0.80403938]
 [0.78219211]
 [0.76868551]
 [0.75498131]
 [0.78059532]
 [0.76714764]
 [0.7488781 ]
 [0.75149849]
 [0.70052003]
 [0.67603763]
 [0.68699461]
 [0.66948358]
 [0.69295831]
 [0.60077683]
 [0.52482994]
 [0.56678819]
 [0.60785906]]
```

Gambar 24 Data *training input* dan *ouput*

```
input test :  [[0.59696491 0.59344676 0.59975661]
 [0.58970644 0.57087106 0.59420646]
 [0.58518941 0.58632481 0.58532499]
 [0.59590708 0.59515274 0.61641403]
 [0.60774799 0.59906417 0.61363527]
 [0.5805518  0.57815978 0.58224312]
 [0.56848445 0.56162169 0.58020615]
 [0.59722896 0.58163573 0.58120272]
 [0.57217949 0.56330002 0.57384785]
 [0.56405979 0.5514056  0.57190591]
 [0.55239669 0.54810508 0.56136635]
 [0.54285947 0.55320693 0.56192383]
 [0.57636094 0.58134596 0.58211327]
 [0.5888726  0.5734901  0.59089316]
 [0.56752921 0.55052722 0.56738182]
 [0.55705351 0.54325296 0.56957812]
 [0.55817633 0.56065085 0.57739216]
 [0.56605405 0.55109808 0.56768329]
 [0.56146182 0.55558243 0.56660725]
 [0.56822367 0.56551082 0.57985962]]
```

```
target test :  [[0.59773014]
 [0.59143178]
 [0.58664469]
 [0.5957901 ]
 [0.60719425]
 [0.58064417]
 [0.56826469]
 [0.59768636]
 [0.57296299]
 [0.56513863]
 [0.55419024]
 [0.54271205]
 [0.5777603 ]
 [0.58897094]
 [0.56787399]
 [0.55799421]
 [0.55929152]
 [0.56661431]
 [0.56275429]
 [0.56804745]]
```

Gambar 25 Data *testing input* dan *ouput*

4.1.2 Pelatihan Jaringan ANN-GA

4.1.2.1 Membangun Model ANN

Dalam membentuk model jaringan ANN penulis menggunakan bantuan dari *library python* bernama *PyGAD*. Penulis menggunakan fungsi GANN() untuk membuat model ANN. Dalam membentuk model ANN fungsi ini memiliki parameter yang berisi *num_solutions*, *num_neurons_input*, *num_neurons_hidden_layers*, *num_neurons_output*, *hidden_activations*, dan *output_activations*. *Num_solutions* digunakan untuk menentukan jumlah solusi dalam populasi. *Num_neurons_input* digunakan untuk menentukan jumlah *neuron input* yang digunakan. *Num_neurons_hidden_layers* digunakan untuk menentukan jumlah *neuron* yang digunakan dalam *hidden layer*. *Num_neurons_output* digunakan untuk menentukan jumlah *neuron* pada *layer output*. *Hidden_activations* digunakan untuk menentukan jenis fungsi aktivasi yang digunakan dalam *hidden layer*. Lalu *output_activations* digunakan untuk menentukan fungsi aktivasi yang digunakan pada *layer output*.

Dalam percobaan ini akan terdapat 3 model ANN dengan 3 jenis *hidden layer* yang digunakan, yaitu model ANN dengan 1 *hidden layer*, 2 *hidden layer*,

dan 3 *hidden layer*. *Hidden layer* pertama akan berisi 5 *neuron*, *hidden layer* kedua berisi 10 *neuron*, dan *hidden layer* ketiga berisi 10 *neuron*. Lalu untuk jumlah solusi dalam populasi bernilai 10, jumlah *input* 3, *neuron output* bernilai 1, dan fungsi aktivasi dalam *hidden layer* adalah relu sedangkan untuk *output layer* tidak menggunakan fungsi aktivasi. *Source code* dari fungsi yang digunakan dapat dilihat dari gambar berikut:

```
GANN_model = pygad.gann.GANN(num_solutions=10,
                             num_neurons_input=3,
                             num_neurons_hidden_layers=[5],
                             num_neurons_output=1,
                             hidden_activations=["relu"],
                             output_activation="None")
```

Gambar 26 *Source code* model ANN 1 *hidden layer*

```
GANN_model = pygad.gann.GANN(num_solutions=10,
                             num_neurons_input=3,
                             num_neurons_hidden_layers=[5, 10],
                             num_neurons_output=1,
                             hidden_activations=["relu", "relu"],
                             output_activation="None")
```

Gambar 27 *Source code* model ANN 2 *hidden layer*

```
GANN_model = pygad.gann.GANN(num_solutions=10,
                             num_neurons_input=3,
                             num_neurons_hidden_layers=[5, 10, 10],
                             num_neurons_output=1,
                             hidden_activations=["relu", "relu", "relu"],
                             output_activation="None")
```

Gambar 28 *Source code* model ANN 3 hidden layer

4.1.2.2 Membangun Model GA

Dalam membangun model genetika algoritma penulis menggunakan *library* dari *pygad* dengan menggunakan fungsi GA. Fungsi ini memiliki parameter *num_generations*, *num_parents_mating*, *fitness_func*, *mutation_percent_genes*, *initial_population*, *init_range_low*, *init_range_high*, *parent_selection_type*, *crossover_type*, *mutation_type*, *keep_parent*, dan *on_generation*.

Num_generations berfungsi untuk menentukan jumlah generasi yang digunakan. *Num_parent_mating* adalah jumlah *parent* yang akan dipilih dalam setiap generasi. *Fitness_func* adalah metode yang digunakan dalam penghitungan fitness. *Mutation_percent_genes* adalah probabilitas terjadinya mutasi. *Initial_population* adalah populasi awal sebelum dilakukan optimisasi. *Init_range_low* adalah batas bawah nilai di tiap gen. *Init_range_high* adalah batas atas nilai di tiap gen. *Parent_selection_type* adalah tipe pemilihan *parent* yang digunakan. *Crossover_type* berupa tipe *crossover* yang digunakan. *Mutation_type* digunakan untuk memilih tipe mutasi yang digunakan. *Keep_parent* digunakan untuk menyimpan parent yang telah digunakan di generasi sebelumnya. *On_generation* digunakan untuk membangkitkan *method* dalam setiap generasi.

Dalam percobaan ini generasi maksimum pada genetika algoritma akan dilakukan mulai dari 10 hingga 100 generasi, jumlah *parent* yang dipilih 4, probabilitas mutasi 50%, populasi awal akan berisi nilai *random* antara -1 hingga 1, batas bawah nilai tiap gen adalah -10, batas atas nilai tiap gen adalah 10, tipe pemilihan *parent* adalah *roulette wheel selection*, tipe *crossover uniform*, tipe *mutasi random*, *keep_parent* berisi 1 yaitu akan menyimpan salah satu *parent* dari generasi sebelumnya, *method* yang digunakan untuk fungsi fitness adalah *fitness_func*, lalu untuk memperlihatkan nilai fitness tiap generasi dilakukan pada *method callback_generation*. *Source code* dari fungsi yang digunakan dapat dilihat dari gambar berikut:

```
ga_model = pygad.GA(num_generations=10,
                    num_parents_mating=4,
                    initial_population=initial_population,
                    fitness_func=fitness_func,
                    mutation_percent_genes=50,
                    init_range_low=-10,
                    init_range_high=10,
                    parent_selection_type="rws",
                    crossover_type="uniform",
                    mutation_type="random",
                    keep_parents=1,
                    on_generation=callback_generation)
```

Gambar 29 *Source code* model GA

Parameter *fitness_func* menggunakan fungsi bernama *fitness_func*. Dalam fungsi ini akan dilakukan prediksi menggunakan model dan data *training* yang ada. Prediksi dilakukan menggunakan nilai bobot dari kromosom yang telah terbentuk dari hasil *crossover* atau mutasi. Setelah dilakukan prediksi, selanjutnya akan dihitung nilai *error* menggunakan MSE. Nilai MSE ini yang akan dijadikan nilai fitness dengan rumus 1 dibagi MSE. Semakin besar nilai fitness yang didapat maka semakin baik nilai akurasi yang didapat. *Source code* dari *method fitness_func* ini dapat dilihat pada gambar berikut:

```
def fitness_func(solution, sol_idx):
    global GANN_model, X_train, y_train_ga

    predictions = pygad.nn.predict(last_layer=GANN_model.population_networks[sol_idx],
                                   data_inputs=X_train, problem_type="regression")

    solution_fitness = 1/np.mean(np.square(y_train_ga - predictions))
    return solution_fitness
```

Gambar 30 *Source code* fungsi *fitness_func*

Parameter *on_callback* berisi fungsi *callback_generation* yang akan dipanggil dalam setiap generasi yang telah berlangsung. Fungsi ini akan memperbarui populasi dari kromosom yang telah terbentuk dalam satu generasi. Setelah itu fungsi ini mencetak generasi, nilai fitness yang tercapai, dan perubahan fitness terbaik dalam setiap generasi. *Source code* dari *method* ini dapat dilihat pada gambar berikut:

```
def callback_generation(ga_model):
    global GANN_model, last_fitness

    population_matrices = pygad.gann.population_as_matrices(
                          population_networks=GANN_model.population_networks,
                          population_vectors=ga_model.population)

    GANN_model.update_population_trained_weights(population_trained_weights=population_matrices)

    print("Generation = {generation}".format(generation=ga_model.generations_completed))
    print("Fitness    = {fitness}".format(fitness=ga_model.best_solution()[1]))
    print("Change     = {change}".format(change=ga_model.best_solution()[1] - last_fitness))

    last_fitness = ga_model.best_solution()[1].copy()
```

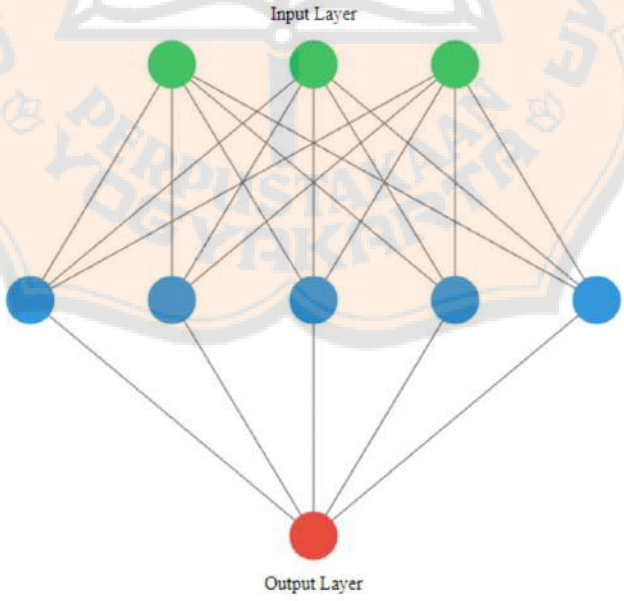
Gambar 31 *Source code* fungsi *callback_generation*

4.1.3 Percobaan ANN-GA

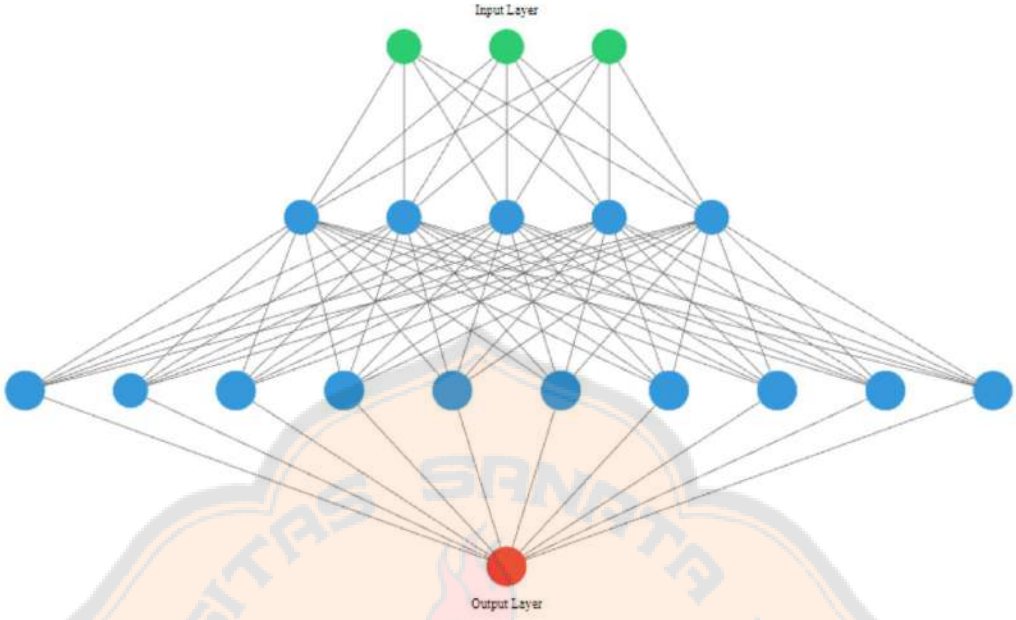
Percobaan akan terbagi menjadi 3 jenis, yaitu percobaan menggunakan 1 *hidden layer*, percobaan 2 *hidden layer*, dan percobaan dengan 3 *hidden layer*. Percobaan dengan 1 *hidden layer* akan menghasilkan kromosom dengan nilai fitness terbaik dari operasi genetika algoritma. Kemudian nilai kromosom ini yang akan menjadi nilai bobot dalam *testing* ANN. Begitu juga dengan percobaan dengan 2 *hidden layer* dan 3 *hidden layer*. Percobaan akan dibagi dari 10 hingga 100 generasi. Setiap generasi akan dilakukan 5 kali percobaan agar didapat nilai rata – rata MSE.

Dari percobaan tersebut akan didapatkan ANN-GA dengan *hidden layer* terbaik. ANN-GA dengan *hidden layer* terbaik kemudian akan dilakukan percobaan fungsi aktivasi *softmax*. Hasil akhir dari percobaan ini akan didapat ANN-GA dengan *hidden layer* terbaik dan fungsi aktivasi terbaik antara *relu* dan *softmax*.

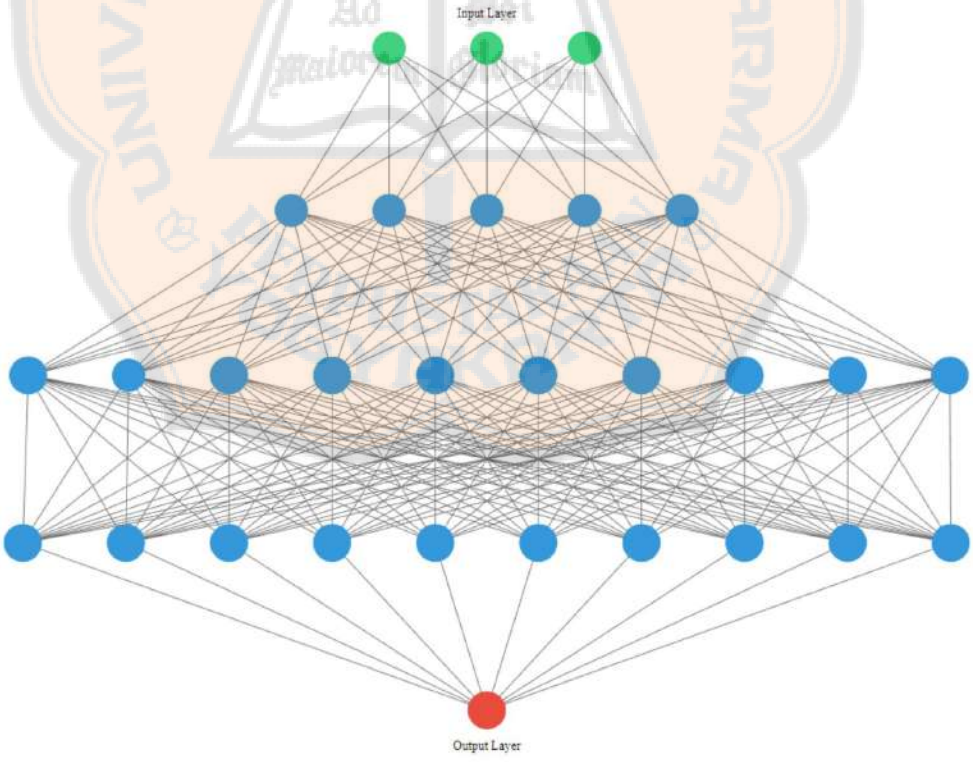
4.1.3.1 Percobaan Hidden Layer



Gambar 32 Model jaringan ANN 1 *hidden layer*



Gambar 33 Model jaringan ANN 2 *hidden layer*



Gambar 34 Model jaringan ANN 3 *hidden layer*

Dari model jaringan tersebut akan dilakukan beberapa percobaan ANN-GA. Percobaan dilakukan selama 5 kali pada tiap generasi yaitu antara 10 hingga 100 generasi. Kemudian akan dicari nilai rata – rata MSE untuk dibandingkan. Berikut tabel percobaan ANN-GA:

Tabel 3 Percobaan ANN-GA dengan 1 hidden layer

Generasi	Nilai MSE Percobaan Ke – i					Rata – Rata Nilai MSE
	1	2	3	4	5	
10	0.0001483	0.0001368	0.0000249993	0.000187799	0.000467632	0.000193131
20	0.00029098	0.00010637	0.000297502	0.0000183718	0.00012527	0.000167698
30	0.0001508	0.00006274	0.00040585	0.00045213	0.000237	0.0002617
40	0.00026029	0.00040495	0.000084014	0.00010094	0.00016398	0.000202834
50	0.00008436	0.00031473	0.000092703	0.000144	0.00040506	0.000208171
60	0.00009834	0.0002089	0.000194 9	0.0003563	0.00023566	0.00021882
70	0.00011664	0.0003557	0.0001731	0.000199	0.0002153	0.00021195
80	0.00017992	0.00006344	0.00049393	0.00013354	0.00010939	0.00019604
90	0.00018964	0.000167	0.00030046	0.00014038	0.0003517	0.00022983
100	0.00025831	0.00029181	0.000063415	0.0001913	0.00012988	0.00018694

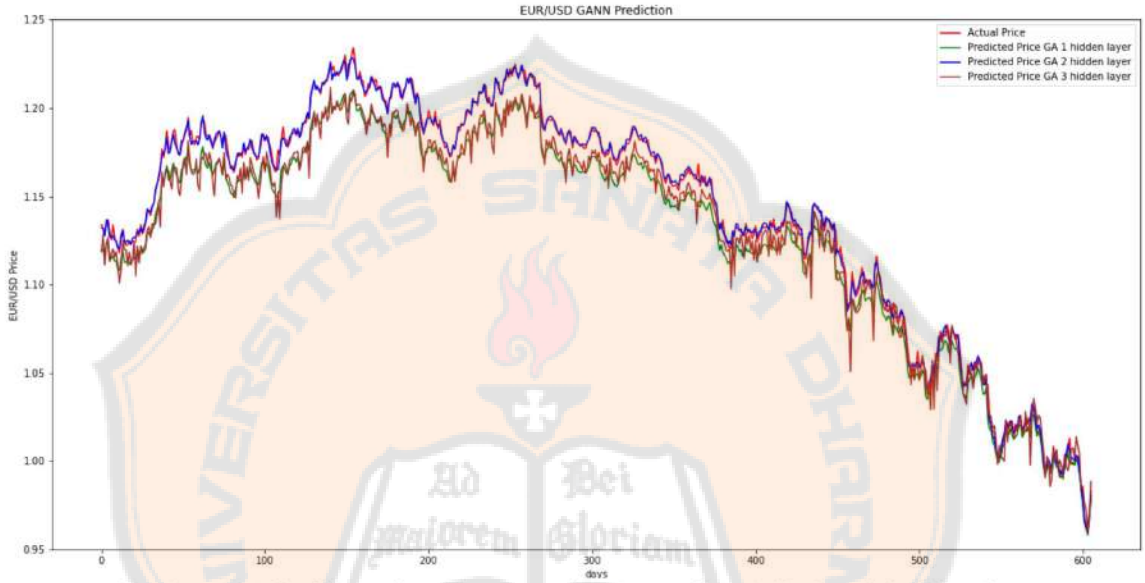
Tabel 4 Percobaan ANN-GA dengan 2 hidden layer

Generasi	Nilai MSE Percobaan Ke – i					Rata – Rata Nilai MSE
	1	2	3	4	5	
10	0.000175	0.0000217	0.00001276236	0.00008483385	0.0000125684	0.000061372
20	0.000259947	0.000159275	0.000058914	0.0002	0.000179591	0.00017154
30	0.000377	0.00011557	0.000085	0.000013	0.0001493	0.00014797
40	0.000030883	0.0002217	0.000321	0.000023	0.00041896	0.0002031
50	0.0001114	0.00002864	0.0003	0.0001971	0.00008938	0.0001453
60	0.000071273	0.000361656	0.0003216	0.00039929	0.00037	0.00030476
70	0.00016745	0.0004545	0.000095664	0.00018756	0.0002811	0.0002372
80	0.00015971	0.00015532	0.00005196	0.00039583	0.0004348	0.00023952
90	0.000044364	0.000298	0.0003224	0.00014666	0.0005342	0.0002689
100	0.0001	0.00025	0.000336	0.000234	0.00024	0.0002329

Tabel 5 Percobaan ANN-GA dengan 3 *hidden layer*

Generasi	Nilai MSE Percobaan Ke – i					Rata – Rata Nilai MSE
	1	2	3	4	5	
10	0.0004797	0.0012718	0.0004443	0.0006624	0.00019	0.0006096
20	0.0002945	0.0002277	0.0000769	0.00013	0.0001629	0.0001784
30	0.0003518	0.0001654	0.00043	0.0000578	0.00035	0.000271
40	0.00002235	0.000227	0.0002342	0.00003754	0.0001976	0.00014373
50	0.00006145	0.00006893	0.0004264	0.00049158	0.00039157	0.000289
60	0.00003463 9	0.00025512	0.0002366 3	0.00016535	0.0001436	0.0001671
70	0.00045519	0.00023711	0.0004367	0.00002698	0.00003245	0.00023788
80	0.00031	0.00039418	0.0001666	0.00011856	0.00009365	0.00023
90	0.00000087	0.00015	0.0001854	0.00000369	0.0000364	0.00008348
100	0.00005365	0.0002754	0.0004	0.00013555	0.0001	0.00019292

Dari percobaan tersebut didapat nilai rata – rata MSE terbaik dari 1 *hidden layer* dari ANN-GA adalah 0.000167698 dengan 20 generasi. MSE terbaik 2 *hidden layer* ANN-GA adalah 0.000061372 dengan 10 generasi. MSE terbaik untuk 3 *hidden layer* ANN-GA adalah 0.00008348 dengan 90 generasi. Berikut adalah grafik perbandingan ANN-GA dengan harga asli Eur/Usd dengan MSE terbaik.



Gambar 35 Grafik perbandingan MSE ANN-GA 1, 2, dan 3 *hidden layer*

4.1.3.2 Percobaan Fungsi Aktivasi

Setelah di dapat *hidden layer* terbaik, percobaan akan dilanjutkan dengan percobaan fungsi aktivasi. Pada percobaan ini fungsi aktivasi ANN-GA dengan *hidden layer* terbaik akan diubah menjadi *softmax* dan akan dibandingkan dengan ANN-GA sebelumnya yang menggunakan fungsi aktivasi relu. Percobaan akan dilakukan menggunakan *hidden layer* dan generasi terbaik dari percobaan sebelumnya dan akan dilakukan sebanyak 5 kali untuk menentukan nilai rata-rata MSE. Nilai rata-rata MSE tersebut akan dibandingkan antara ANN-GA *hidden layer* relu dengan ANN-GA *hidden layer softmax*. Nilai MSE percobaan fungsi aktivasi dapat dilihat pada tabel berikut.

Tabel 6 Percobaan Fungsi Aktivasi

Percobaan ANN-GA ke-i	Fungsi aktivasi *hidden layer* 1, *hidden layer* 2		
	Softmax, ReLU	ReLU, *Softmax*	*Softmax*, *Softmax*
1	0.01118	0.004524	0.006
2	0.005354	0.004637	0.006242
3	0.004522	0.005117	0.01341
4	0.004554	0.004784	0.004459
5	0.011343	0.0044	0.0048
Rata-rata MSE	0.0073906	0.0046924	0.0069822

4.2 ANALISIS

Dari percobaan yang telah dilakukan didapatkan bahwa rata – rata MSE ANN-GA dari setiap *hidden layer* kurang baik tapi masih bisa mengikuti *trend* dari harga asli Eur/Usd. Hal ini dapat dilihat pada tabel berikut ini yang menunjukkan MSE setiap *hidden layer* dari setiap iterasi yang dilakukan.

Tabel 7 Hasil MSE 1 hidden layer

Generasi	Rata – rata MSE 1 *hidden layer* ANN-GA
10	0.000193131
20	0.000167698
30	0.0002617
40	0.000202834
50	0.000208171
60	0.00021882
70	0.00021195
80	0.00019604
90	0.00022983
100	0.00018694

Dari tabel tersebut dapat dilihat bahwa pada percobaan menggunakan 1 *hidden layer* didapatkan nilai MSE dari ANN-GA cenderung lebih stabil yaitu diantara 10^{-4} dengan MSE terendah adalah 0.000167698 dengan 20 generasi dan MSE tertinggi adalah 0.0002617 dengan 30 generasi.

Tabel 8 Hasil MSE 2 hidden layer

Generasi	Rata – rata MSE 2 *hidden layer* ANN-GA
10	0.000061372
20	0.00017154
30	0.00014797
40	0.0002031
50	0.0001453
60	0.00030476
70	0.0002372
80	0.00023952
90	0.0002689
100	0.0002329

Pada percobaan menggunakan 2 *hidden layer* didapatkan nilai MSE dari ANN-GA juga cenderung stabil di 10^{-4} dengan MSE terendah di 10^{-5} yaitu 0.000061372 dengan 10 generasi dan MSE tertinggi adalah 0.00030476 dengan 60 generasi.

Tabel 9 Hasil MSE 3 hidden layer

Generasi	Rata – rata MSE 3 *hidden layer* ANN-GA
10	0.0006096
20	0.0001784
30	0.000271
40	0.00014373
50	0.000289

60	0.0001671
70	0.00023788
80	0.00023
90	0.00008348
100	0.00019292

Percobaan menggunakan 3 *hidden layer* didapatkan nilai MSE dari ANN-GA hampir semuanya berada di 10^{-4} dengan MSE terendah di 10^{-5} yaitu 0.00008348 dengan 90 generasi dan MSE tertinggi adalah 0.0006096 dengan 10 generasi.

Tabel 10 Rata-rata MSE setiap layer ANN-GA

Generasi	Rata – rata MSE Jumlah *hidden layer* ANN-GA		
	1	2	3
10	0.000193131	0.000061372	0.0006096
20	0.000167698	0.00017154	0.0001784
30	0.0002617	0.00014797	0.000271
40	0.000202834	0.0002031	0.00014373
50	0.000208171	0.0001453	0.000289
60	0.00021882	0.00030476	0.0001671
70	0.00021195	0.0002372	0.00023788
80	0.00019604	0.00023952	0.00023
90	0.00022983	0.0002689	0.00008348
100	0.00018694	0.0002329	0.00019292

Dari ketiga percobaan yang dilakukan pada *hidden layer* didapatkan hasil terbaik pada ANN-GA adalah dengan menggunakan 2 *hidden layer* dengan generasi sebanyak 10 kali didapatkan hasil akurasi sebesar 0.000061372. ANN-GA menggunakan 2 *hidden layer* dengan generasi 10 kali kemudian akan dilakukan percobaan fungsi aktivasi menggunakan *softmax*.

Tabel 11 Perbandingan Rata-rata MSE Fungsi Aktivasi ANN-GA

Fungsi Aktivasi		Rata-rata MSE
Hidden Layer 1	*Hidden Layer* 2	
Softmax	ReLU	0. 0073906
ReLU	*Softmax*	0.0046924
Softmax	*Softmax*	0.0069822
ReLU	ReLU	0.000061372

Pada percobaan fungsi aktivasi didapatkan bahwa ANN-GA 2 *hidden layer* yang menggunakan fungsi aktivasi relu memiliki rata-rata MSE terendah dengan nilai 0.000061372 dibanding dengan ANN-GA 2 *hidden layer* yang menggunakan fungsi aktivasi *softmax*. Dari semua percobaan yang telah dilakukan dapat disimpulkan bahwa ANN-GA 2 *hidden layer* dengan fungsi aktivasi relu sudah cukup baik untuk dapat melakukan prediksi dan mengikuti *trend* harga dengan menghasilkan nilai akurasi terbaik sebesar 0.000061372.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Kesimpulan dari hasil penelitian berjudul algoritma *artificial neural networks* – genetika algoritma untuk memprediksi harga valas adalah:

1. *Neural network* dengan penerapan algoritma genetika dapat digunakan untuk memprediksi harga valas dengan mengikuti trend yang ada
2. Dari percobaan yang dilakukan dengan 1 *hidden layer*, 2 *hidden layer*, dan 3 *hidden layer* didapatkan hasil terbaik dari ANN-GA menggunakan model dengan 2 *hidden layer* dan jumlah generasi sebanyak 10 dengan MSE sebesar 0.000061372.
3. Pada percobaan fungsi aktivasi ANN-GA 2 *hidden layer* menggunakan generasi sebanyak 10 didapatkan hasil terbaik menggunakan fungsi aktivasi relu dengan nilai MSE sebesar 0.000061372.

5.2 Saran

Saran penulis untuk penelitian selanjutnya adalah:

1. Menggunakan jumlah *neuron* yang lebih banyak pada *hidden layer*.
2. Menggunakan jumlah generasi lebih banyak dengan rentang dari 10 hingga 200 atau 400, dan sebagainya.
3. Menggunakan tingkat mutasi gen yang lebih bervariasi seperti 10%, 20%, dan sebagainya.
4. Menggunakan jumlah populasi yang lebih banyak.
5. Menggunakan variasi *parent selection* selain *roulette wheel selection*, seperti turnamen *selection*.

6. Membuat tampilan antarmuka sehingga pengguna umum dapat mengoperasikan.

DAFTAR PUSTAKA

Ali, I., Sularto, L., & Kunci, K. (2019). Optimasi Parameter Artificial Neural Network Menggunakan Algoritma Genetika Untuk Prediksi Kelulusan Mahasiswa. *Jurnal ICT : Information Communication & Technology*, *18*(1), 54–59. https://ejournal.ikmi.ac.id/index.php/jict-ikmi

Amerilyse Caesar, C., Hanum, L., & Cholissodin, I. (2016). Studi Kasus pada UPT Pembibitan Ternak dan Hijauan Makanan Ternak Singosari-Malang. Dalam *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK)* (Vol. 3, Issue 3).

Arsi, P., & Prayogi, J. (2020). Optimasi Prediksi Nilai Tukar Rupiah Terhadap Dolar Menggunakan Neural Network Berbasiskan Algoritma Genetika. *JURNAL INFORMATIKA*, *7*(1). http://ejournal.bsi.ac.id/ejurnal/index.php/ji

Baheti, P. (2023, Januari 3). *Activation Functions in Neural Networks [12 Types & Use Cases]*. https://www.v7labs.com/blog/neural-networks-activation-functions

Bisnis, B. (2021, Agustus 14). *Mengapa Valuta Asing Sangat Penting Digunakan dalam Perdagangan Internasional | kumparan.com*. https://kumparan.com/berita-bisnis/mengapa-valuta-asing-sangat-penting-digunakan-dalam-perdagangan-internasional-1wK8KoTWQ1o/1

Bisnis.com. (2021, September 24). *Mengenal Kurs, Definisi, Jenis dan Faktor yang Mempengaruhinya*. https://ekonomi.bisnis.com/read/20210924/9/1446687/mengenal-kurs-definisi-jenis-dan-faktor-yang-mempengaruhinya

Genetic Algorithms - Parent Selection. (t.t.). Diambil 3 September 2022, dari https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_parent_selection.htm

Haykin, S. (1994). *Neural Networks - A Comprehensive Foundation - Simon Haykin*. Prentice Hall.

Karunia, V. (2022, Mei 21). *4 Faktor yang Memengaruhi Kurs Valuta Asing Halaman all - Kompas.com*. https://www.kompas.com/skola/read/2022/05/21/073000169/4-faktor-yang-memengaruhi-kurs-valuta-asing?page=all

Kholik, A., Eko Wahyudi, E., Devianto, K., Sholihah, N., Marjani Santosa, Y., Ilmu Komputer dan Elektronika, D., & Matematika dan Ilmu Pengetahuan Alam Universitas Gadjah Mada Yogyakarta, F. (2018). Sistem Rekomendasi Berbasis Genetic Algorithm: Studi Kasus Pembelian Komponen Komputer dan Aksesorisnya. Dalam *Seminar Nasional Aplikasi Teknologi Informasi (SNATi)*.

Nadira, A. (2021, Mei 3). *Apa Itu Valas? Berikut Penjelasan Lengkapnya!* https://www.oyindonesia.com/blog/apa-itu-valuta-asing

Novita, A. (2016). Prediksi Pergerakan Harga Saham Pada Bank Terbesar Di Indonesia Dengan Metode Back propagation Neural Network. *JITISI*, *5*(1), 877–1021.

Oktavia Rahardiani, N., & Firdaus Mahmudy, W. (2018). *Optimasi Bobot Multi-Layer Perceptron Menggunakan Algoritma Genetika Untuk Klasifikasi Tingkat Resiko Penyakit Stroke* (Vol. 2, Issue 8). http://j-ptiik.ub.ac.id

Sena, S. (2017, Oktober 28). *Pengenalan Deep Learning Part 1 : Neural Network | by Samuel Sena | Medium*. https://medium.com/@samuelsena/pengenalan-deep-learning-8fbb7d8028ac

Shyalika, C. (2019, Januari 29). *Genetic Algorithms -Selection. An Insight to Genetic Algorithms*. https://medium.datadriveninvestor.com/genetic-algorithms-selection-5634cfc45d78

Suharjito. (2021, Maret 2). *Algoritma Genetika dengan Python | Computer Science*. https://onlinelearning.binus.ac.id/computer-science/post/algoritma-genetika-dengan-python

Suhartono, D. (2012, Juli 26). *Dasar Pemahaman Neural Network*. https://socs.binus.ac.id/2012/07/26/konsep-neural-network/

Syah, P. (t.t.). *Pengertian, struktur dan prosedur Algoritma Genetika | Matkul.xyz*. Diambil 20 Januari 2023, dari https://matkul.xyz/pengertian-struktur-dan-prosedur-algoritma-genetika-2/

Trivusi. (2022, September 17). *Mengenal Jaringan Saraf Tiruan (JST): Arsitektur dan Jenis-jenisnya - Trivusi*. https://www.trivusi.web.id/2022/07/mengenal-jaringan-saraf-tiruan-jst.html

Yanuar, A. (2018, Mei 24). *Artificial Neural Network (ANN) – Universitas Gadjah Mada Menara Ilmu Machine Learning*. https://machinelearning.mipa.ugm.ac.id/2018/05/24/artificial-neural-network-ann/

LAMPIRAN

```
# Installing required libraries
!pip install tensorflow
!pip install keras
!pip install pygad
!pip install --upgrade pandas
!pip install --upgrade pandas-datareader
!pip install ann_visualizer
!pip install graphviz
!pip install yfinance
!sudo apt-get install graphviz && !pip install graphviz
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: tensorflow in /usr/local/lib/python3.7/dist-packages (2.9.2)
Requirement already satisfied: flatbuffers<2,>=1.12 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.12)
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.21.6)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (0.27.0)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.14.1)
Requirement already satisfied: packaging in /usr/local/lib/python3.7/dist-packages (from tensorflow) (21.3)
Requirement already satisfied: protobuf<3.20,>=3.9.2 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (3.19.6)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.50.0)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (3.3.0)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (0.4.0)
Requirement already satisfied: keras-preprocessing>=1.1.1 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.1.2)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (2.1.0)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.15.0)
Requirement already satisfied: tensorflow-estimator<2.10.0,>=2.9.0rc0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (2.9.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.3.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: keras<2.10.0,>=2.9.0rc0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (2.9.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-packages (from tensorflow) (57.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (4.1.1)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (14.0.6)
Requirement already satisfied: tensorboard<2.10,>=2.9 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (2.9.1)
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow) (3.1.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.7/dist-packages (from astunparse>=1.6.0->tensorflow) (0.38.4)
Requirement already satisfied: cached-property in /usr/local/lib/python3.7/dist-packages (from h5py>=2.9.0->tensorflow) (1.5.2)
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from tensorboard<2.10,>=2.9->tensorflow) (1.0.1)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/python3.7/dist-packages (from tensorboard<2.10,>=2.9->tensorflow) (1.8.1)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /usr/local/lib/python3.7/dist-packages (from tensorboard<2.10,>=2.9->tensorflow) (0.4.6)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.7/dist-packages (from tensorboard<2.10,>=2.9->tensorflow) (3.4.1)
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.7/dist-packages (from tensorboard<2.10,>=2.9->tensorflow) (2.23.0)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in /usr/local/lib/python3.7/dist-packages (from tensorboard<2.10,>=2.9->tensorflow) (0.6.1)
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.7/dist-packages (from tensorboard<2.10,>=2.9->tensorflow) (2.14.1)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.7/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.10,>=2.9->tensorflow)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.10,>=2.9->tensorflow
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.7/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.10,>=2.9->tensorflow) (4.9)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.7/dist-packages (from google-auth-oauthlib<0.5,>=0.4.1->tensorboard<2.10,>=2.
Requirement already satisfied: importlib-metadata>=4.4 in /usr/local/lib/python3.7/dist-packages (from markdown>=2.6.8->tensorboard<2.10,>=2.9->tensorflow) (4.
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from importlib-metadata>=4.4->markdown>=2.6.8->tensorboard<2.10,>=2.9->tens
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /usr/local/lib/python3.7/dist-packages (from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard<2
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0->tensorboard<2.10,>=2.9->tensorflow) (2.10)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0->tensorboard<2.10,>=
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0->tensorboard<2.10,>=2.9->tensorflow) (3.0.
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0->tensorboard<2.10,>=2.9->tensorflow) (202
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.7/dist-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<0.5,>=0.4.1->tens
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.7/dist-packages (from packaging->tensorflow) (3.0.9)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
```

```
Requirement already satisfied: keras in /usr/local/lib/python3.7/dist-packages (2.9.0)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting pygad
  Downloading pygad-2.18.1-py3-none-any.whl (56 kB)
     |████████████████████████████████| 56 kB 2.5 MB/s
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (from pygad) (3.2.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from pygad) (1.21.6)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib->pygad) (0.11.0)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->pygad) (2.8.2)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->pygad) (3.0.9)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->pygad) (1.4.4)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-packages (from kiwisolver>=1.0.1->matplotlib->pygad) (4.1.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.1->matplotlib->pygad) (1.15.0)
Installing collected packages: pygad
Successfully installed pygad-2.18.1
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (1.3.5)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (from pandas) (2022.6)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.7/dist-packages (from pandas) (1.21.6)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.7.3->pandas) (1.15.0)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: pandas-datareader in /usr/local/lib/python3.7/dist-packages (0.9.0)
Collecting pandas-datareader
  Downloading pandas_datareader-0.10.0-py3-none-any.whl (109 kB)
     |████████████████████████████████| 109 kB 5.3 MB/s
Requirement already satisfied: lxml in /usr/local/lib/python3.7/dist-packages (from pandas-datareader) (4.9.1)
Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/python3.7/dist-packages (from pandas-datareader) (2.23.0)
Requirement already satisfied: pandas>=0.23 in /usr/local/lib/python3.7/dist-packages (from pandas-datareader) (1.3.5)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.7/dist-packages (from pandas>=0.23->pandas-datareader) (1.21.6)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas>=0.23->pandas-datareader) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (from pandas>=0.23->pandas-datareader) (2022.6)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.7.3->pandas>=0.23->pandas-datareader) (1.15.0)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests>=2.19.0->pandas-datareader) (3.0.4)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests>=2.19.0->pandas-datareader)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests>=2.19.0->pandas-datareader) (2022.9.24)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests>=2.19.0->pandas-datareader) (2.10)
Installing collected packages: pandas-datareader
  Attempting uninstall: pandas-datareader
    Found existing installation: pandas-datareader 0.9.0
    Uninstalling pandas-datareader-0.9.0:
      Successfully uninstalled pandas-datareader-0.9.0
Successfully installed pandas-datareader-0.10.0
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting ann_visualizer
  Downloading ann_visualizer-2.5.tar.gz (4.7 kB)
Building wheels for collected packages: ann-visualizer
  Building wheel for ann-visualizer (setup.py) ... done
  Created wheel for ann-visualizer: filename=ann_visualizer-2.5-py3-none-any.whl size=4168 sha256=019827d2639c6fa1d265c0529a99a7c7da492b79486207c16e4151547
  Stored in directory: /root/.cache/pip/wheels/1b/fc/58/2ab1c3b30350105929308becddda4fb59b1358e54f985e1f4a
Successfully built ann-visualizer
Installing collected packages: ann-visualizer
Successfully installed ann-visualizer-2.5
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: graphviz in /usr/local/lib/python3.7/dist-packages (0.10.1)
Reading package lists... Done
Building dependency tree
Reading state information... Done
graphviz is already the newest version (2.40.1-2).
The following package was automatically installed and is no longer required:
  libnvidia-common-460
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 5 not upgraded.
/bin/bash: !pip3: command not found
```

```
# importing the libraries
import keras
import tensorflow
import pandas as pd
import numpy as np
import pygad
import pygad.nn
import pygad.gann
import random
import datetime as dt
import math
import plotly.graph_objects as go
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from ann_visualizer.visualize import ann_viz;
import matplotlib.pyplot as plt
import graphviz
import yfinance as yf
```

```
#Load data train
company = "EURUSD=X"

start = dt.datetime(2015, 1, 1)
end = dt.datetime(2022,10,1)

Data =  yf.download(tickers=[company], start=start, end=end)
Data
```

Date	High	Low	Open	Close	Volume	Adj Close
2015-01-01	1.209863	1.209863	1.209863	1.209863	0.0	1.209863
2015-01-02	1.208956	1.201080	1.208868	1.208941	0.0	1.208941
2015-01-05	1.197590	1.188909	1.195500	1.194643	0.0	1.194643
2015-01-06	1.197000	1.188693	1.193830	1.193902	0.0	1.193902
2015-01-07	1.190000	1.180401	1.187479	1.187536	0.0	1.187536
...	...	...	...	...	...	...
2022-09-25	0.969989	0.958231	0.968992	0.968992	0.0	0.968992
2022-09-26	0.967006	0.959453	0.962371	0.962371	0.0	0.962371
2022-09-27	0.968626	0.954016	0.959619	0.959619	0.0	0.959619
2022-09-28	0.978732	0.963828	0.970817	0.970817	0.0	0.970817
2022-09-29	0.984999	0.973606	0.982956	0.982956	0.0	0.982956

2019 rows × 6 columns

```
# Separate Target Variable and Predictor Variables
Predictors=['Open', 'High', 'Low']
TargetVariable=['Close']

X=Data[Predictors].values
y=Data[TargetVariable].values

### Sandardization of data ###
from sklearn.preprocessing import MinMaxScaler
PredictorScaler=MinMaxScaler()
TargetVarScaler=MinMaxScaler()

# # Storing the fit object for later reference
PredictorScalerFit=PredictorScaler.fit(X)
TargetVarScalerFit=TargetVarScaler.fit(y)

# # Generating the standardized values of X and y
X=PredictorScalerFit.transform(X)
y=TargetVarScalerFit.transform(y)
```

```
# # Split the data into training and testing set
X_train, X_test, y_train_sgd, y_test_sgd = train_test_split(X, y, test_size=0.3, shuffle=False)

y_test_ga = np.reshape(y_test_sgd, (606,))
y_train_ga = np.reshape(y_train_sgd, (1413,))
# # Quick sanity check with the shapes of Training and testing datasets
print("input train shape : ", X_train.shape)
print("Target train shape : ", y_train_sgd.shape)
print("input test shape : ", X_test.shape)
print("target test shape : ", y_test_sgd.shape)
```

```
input train shape :  (1413, 3)
Target train shape :  (1413, 1)
input test shape :  (606, 3)
target test shape :  (606, 1)
```

```
def fitness_func(solution, sol_idx):
    global GANN_model, X_train, y_train_ga

    predictions = pygad.nn.predict(last_layer=GANN_model.population_networks[sol_idx],
                                   data_inputs=X_train, problem_type="regression")

    solution_fitness = 1/np.mean(np.square(y_train_ga - predictions))
    return solution_fitness
```

```
def callback_generation(ga_model):
    global GANN_model, last_fitness

    population_matrices = pygad.gann.population_as_matrices(
                          population_networks=GANN_model.population_networks,
                          population_vectors=ga_model.population)

    GANN_model.update_population_trained_weights(population_trained_weights=population_matrices)

    print("Generation = {generation}".format(generation=ga_model.generations_completed))
    print("Fitness    = {fitness}".format(fitness=ga_model.best_solution()[1]))
    print("Change     = {change}".format(change=ga_model.best_solution()[1] - last_fitness))

    last_fitness = ga_model.best_solution()[1].copy()
```

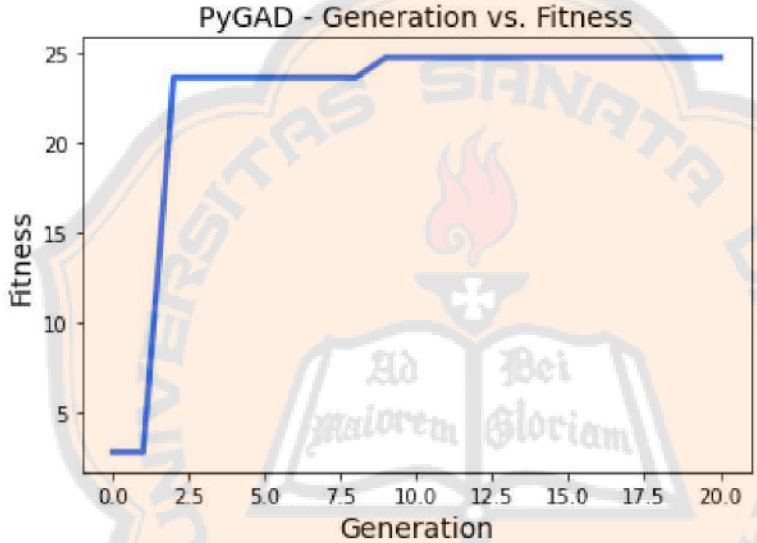
```
ga_model = pygad.GA(num_generations=20,
                    num_parents_mating=4,
                    initial_population=initial_population,
                    fitness_func=fitness_func,
                    mutation_percent_genes=50,
                    init_range_low=-10,
                    init_range_high=10,
                    parent_selection_type="rws",
                    crossover_type="uniform",
                    mutation_type="random",
                    keep_parents=1,
                    on_generation=callback_generation)

# train ANN GA
ga_model.run()

# plot the fitness
ga_model.plot_fitness()

# Returning the details of the best solution.
solution, solution_fitness, solution_idx = ga_model.best_solution()
print("Parameters of the best solution : {solution}".format(solution=solution))
print("Fitness value of the best solution = {solution_fitness}".format(solution_fitness=solution_fitness))
print("Index of the best solution : {solution_idx}".format(solution_idx=solution_idx))
```

```
last_fitness = 0
num_inputs = 3

# create ANN GA model
GANN_model = pygad.gann.GANN(num_solutions=10,
                             num_neurons_input=3,
                             num_neurons_hidden_layers=[5,10,10],
                             num_neurons_output=1,
                             hidden_activations=["relu","relu","relu"],
                             output_activation="None")

# create initial population
population_vectors = pygad.gann.population_as_vectors(population_networks=GANN_model.population_networks)
initial_population = population_vectors.copy()
```

```
if ga_model.best_solution_generation != -1:
    print("Best fitness value reached after {best_solution_generation} generations.".format(best_solution_generation=ga_model.best_solution_generation))
```

```
Generation = 1
Fitness    = 7.4220841578460846
Change     = 7.4220841578460846
Generation = 2
Fitness    = 23.642196735925808
Change     = 16.22011257807972
Generation = 3
Fitness    = 23.642196735925808
Change     = 0.0
Generation = 4
Fitness    = 23.642196735925808
Change     = 0.0
Generation = 5
Fitness    = 23.642196735925808
Change     = 0.0
Generation = 6
Fitness    = 23.642196735925808
Change     = 0.0
Generation = 7
Fitness    = 23.642196735925808
Change     = 0.0
Generation = 8
Fitness    = 23.642196735925808
Change     = 0.0
Generation = 9
Fitness    = 24.76207164828629
Change     = 1.1198749123604834
Generation = 10
Fitness    = 24.76207164828629
Change     = 0.0
Generation = 11
Fitness    = 24.76207164828629
Change     = 0.0
Generation = 12
Fitness    = 24.76207164828629
Change     = 0.0
Generation = 13
Fitness    = 24.76207164828629
Change     = 0.0
Generation = 14
Fitness    = 24.76207164828629
Change     = 0.0
Generation = 15
Fitness    = 24.76207164828629
Change     = 0.0
Generation = 16
Fitness    = 24.76207164828629
Change     = 0.0
```

```
Generation = 17
Fitness    = 24.76207164828629
Change     = 0.0
Generation = 18
Fitness    = 24.76207164828629
Change     = 0.0
Generation = 19
Fitness    = 24.76207164828629
Change     = 0.0
Generation = 20
Fitness    = 24.76207164828629
Change     = 0.0
```




```
Parameters of the best solution : [-1.00286148e+00  9.15321223e-01  1.82120648e-01  9.50143333e-02
  1.76723846e+00 -6.89066432e-02  4.99672182e-02 -1.06987404e+00
 -5.67618109e-02 -3.07923836e-01  4.55758832e-01 -4.73829863e-01
 -1.19279631e+00 -7.47292054e-01 -9.87893628e-01  1.22196277e+00
  5.78352905e-01 -1.69949458e-02 -6.50012061e-02  6.99050495e-01
  8.17294938e-02 -1.21707756e-01 -9.50973314e-02 -2.65540049e-03
  2.10424881e-02  3.46494960e-01 -1.11756441e+00 -4.85016137e-01
 -3.04390771e-01  6.34407458e-01  9.29656767e-01 -5.19165970e-01
 -3.39225076e-01 -3.12601252e-01  4.82460264e-01  8.28881960e-01
  8.65894559e-01 -6.27554486e-02 -8.09088888e-01 -8.08109051e-01
  6.69995644e-01 -2.67433154e-01 -8.47437082e-02  8.58239393e-01
  9.66579979e-03 -6.23536272e-02  6.52559957e-01 -1.45608130e-02
  9.12576813e-02 -9.30253029e-01 -7.48692408e-01  1.07145672e+00
 -1.38360795e-01  5.28514630e-01 -1.35808743e-01 -1.57651619e+00
 -1.85846513e+00  6.05875026e-01  1.78562802e+00 -9.07687872e-01
 -2.50419662e-01  3.05333591e-01 -9.66631041e-01 -4.98148555e-01
 -9.88438688e-01 -5.96719623e-01 -2.18867479e-01  1.66385033e-01
  9.08931089e-01 -2.17829708e-01 -3.13873846e-01  7.38806473e-01
 -1.02552781e-01 -7.48768734e-02  4.89244678e-02  8.69843497e-01
 -1.92019375e-01  9.85750634e-01 -4.22447687e-01  1.87817451e-01
  8.18546766e-01 -1.13754830e+00 -7.42298960e-02  5.93420834e-01
  1.40700984e-01 -2.62121633e-01 -1.16541435e+00 -1.45308841e+00
  1.23626097e+00 -3.30069900e-02  3.60956523e-01  2.97817880e-01
 -8.01767469e-02  6.95505577e-01 -2.99620219e-01  1.40227279e-01
  6.71252173e-01 -1.31820037e+00  8.81546431e-01 -1.65389000e-01
 -8.86980642e-01 -1.45057169e-01 -1.04000077e+00 -8.36293305e-01
 -1.05131885e+00  1.77495631e+00 -1.05823867e+00  2.74793284e-01
```

```
  2.05268198e-01  2.00845578e+00 -1.89122146e-01 -5.29011490e-01
  3.15575473e-01  5.87995758e-01  1.27435484e+00  1.44548746e-01
  4.18148117e-01  7.39290804e-02 -3.93671314e-02 -1.86026086e+00
 -6.92435742e-01  1.32180826e+00 -6.46035391e-01  4.43114419e-01
 -1.46446023e-01 -3.70592092e-01  1.38960383e+00  8.42594089e-01
 -3.30595604e-01  9.52680389e-02  5.64909806e-01  9.12355127e-02
 -5.61442049e-02  6.12593943e-02  1.12048699e+00 -9.91809210e-01
 -7.03888120e-02 -8.55363681e-01 -1.19510245e+00 -5.12718769e-02
  4.20406468e-02 -4.47111358e-01  8.92843587e-01  8.46903774e-01
 -6.79540609e-01 -5.40516285e-01 -2.57531177e-01  1.92946845e-02
 -9.21758658e-02 -2.56227075e-01  1.66232154e-02  6.89699905e-01
  3.57902656e-01 -6.05827248e-01  1.87254216e-01  4.54269193e-04
  6.75637082e-01 -1.57177528e+00 -3.79761070e-01  3.91992186e-01
  7.68219460e-01  3.94064664e-01 -6.49282753e-03 -1.00974121e-01
  1.53513100e-02 -8.18788125e-01 -1.32213124e-02  8.16999951e-02
  1.18040445e+00 -9.59430519e-01 -1.22936166e+00  6.17223321e-01
  4.84855516e-01  1.31874031e-02  4.82590096e-01]
Fitness value of the best solution = 24.76207164828629
Index of the best solution : 0
Best fitness value reached after 9 generations.
```

```
# # test the outputs of the data using the best solution.
ga_predictions = pygad.nn.predict(last_layer=GANN_model.population_networks[solution_idx],
                                  data_inputs=X_test,
                                  problem_type="regression")
ga_predictions=TargetVarScalerFit.inverse_transform(ga_predictions)

y_test = np.reshape(y_test_ga, (606,1))
y_test_orig=TargetVarScalerFit.inverse_transform(y_test)

# # Calculating MSE
MSE_ga = np.mean(np.square(y_test_orig - ga_predictions))

print("MSE of ANN-GA : {MSE}.".format(MSE=MSE_ga))

Test_Data=PredictorScalerFit.inverse_transform(X_test)

TestingData=pd.DataFrame(data=Test_Data)
TestingData['Actual_Price']=y_test_orig
TestingData['Predicted_Price_ANNGA']=ga_predictions
TestingData['MSE of ANNGA']=MSE_ga

# ga_prediction_1hLayer = ga_predictions
# ga_prediction_2hLayer_softmax_relu = ga_predictions
# ga_prediction_2hLayer_relu_softmax = ga_predictions
ga_prediction_2hLayer_softmax_softmax = ga_predictions
# ga_prediction_2hLayer = ga_predictions
# ga_prediction_3hLayer = ga_predictions
TestingData
```

MSE of ANN-GA : 0.00014517502745798577.

	0	1	2	Actual_Price	Predicted_Price_ANNGA	MSE of ANNGA
0	1.133723	1.138395	1.128566	1.133787	1.120064	0.000145
1	1.131606	1.131875	1.126951	1.131952	1.125240	0.000145
2	1.130288	1.136338	1.124366	1.130557	1.113563	0.000145
3	1.133414	1.138887	1.133414	1.133222	1.128970	0.000145
4	1.136868	1.140017	1.132605	1.136544	1.126761	0.000145

```
# plot prediction with real data
plt.plot(y_test_orig, color='red', label=f"Actual Price")

plt.plot(ga_prediction_1hLayer, color='green', label=f"Predicted Price GA 1 hidden layer")
plt.plot(ga_prediction_2hLayer, color='blue', label=f"Predicted Price GA 2 hidden layer")
plt.plot(ga_prediction_3hLayer, color='brown', label=f"Predicted Price GA 3 hidden layer")
# plt.plot(ga_prediction_2hLayer_softmax_relu, color='cyan', label=f"Predicted Price GA 2 hidden layer softmax")
# plt.plot(ga_prediction_2hLayer_relu_softmax, color='cyan', label=f"Predicted Price GA 2 hidden layer softmax")
# plt.plot(ga_prediction_2hLayer_softmax_softmax, color='cyan', label=f"Predicted Price GA 2 hidden layer softmax")
plt.title(f"EUR/USD GANN Prediction")
plt.xlabel('days')
plt.ylabel(f"EUR/USD Price")
plt.ylim(0.95,1.25)
plt.rcParams['figure.figsize'] = [20, 7]
# plt.rcParams['figure.figsize'] = [20, 10]
plt.legend()
# plt.savefig('img.png')
plt.show()
```