IMPLEMENTASI JARINGAN SARAF TIRUAN *BACKPROPAGATION* UNTUK MEMPREDIKSI JUMLAH PENDUDUK MISKIN DI INDONESIA DENGAN OPTIMASI ALGORITME GENETIKA

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh:
Arthur Julio Risa Ashshiddiqi
NIM : 165150209111009

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

IMPLEMENTASI JARINGAN SARAF TIRUAN *BACKPROPAGATION* UNTUK MEMPREDIKSI JUMLAH PENDUDUK MISKIN DI INDONESIA DENGAN OPTIMASI ALGORITME GENETIKA

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Arthur Julio Risa Ashshiddiqi
NIM: 165150209111009

Skripsi ini telah diuji dan dinyatakan lulus pada
20 April 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Indriati, S.T, M.Kom	Ir.Sutrisno, M.T
NIP: 19831013 201504 2 002	NIP: 19570325 198701 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T., M.T., Ph.D
NIP. 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 26 April 2018

Arthur Julio Risa Ashshiddiqi
NIM: 165150209111009

ABSTRAK

Kemiskinan merupakan masalah umum yang dihadapi setiap negara, dan Indonesia sebagai salah satunya. Peningkatan penduduk miskin terjadi hampir setiap tahunnya. Menurut Badan Pusat Statistik dengan indikator penduduk yang memiliki pengeluaran perbulan dibawah garis kemiskinan dikategorikan sebagai rakyat kurang mampu. Meningkatnya jumlah penduduk kurang mampu akan memicu terjadinya tindak kriminalitas, situasi tersebut terjadi karena individu tersebut akan melakukan apapun untuk memenuhi kebutuhannya. Dengan memprediksi jumlah penduduk miskin, diharapkan pemerintah ataupun lembaga – lembaga yang terkait dengan topik ini dapat membantu untuk mengurangi jumlah penduduk miskin dan tingkat pengangguran di Indonesia. Jaringan syaraf tiruan *backpropagation* adalah salah satu metode yang bisa dipakai untuk melakukan prediksi. Pelatihan bobot dan bias pada *backpropagation* dioptimasi menggunakan algoritme genetika untuk mendapatkan hasil yang lebih optimal. Pada penelitian ini metode jaringan syaraf tiruan *backpropagation* yang bobot pelatihannya dioptimasi menggunakan algoritme genetika menghasilkan nilai AFER sebesar 8.744579%.

Kata kunci: *Jaringan syaraf tiruan, backpropagation, algoritme genetika, time-series.*

ABSTRACT

Poverty is a common issues encountered by every country, and Indonesia is one of them. The escalation of the poor occurred almost every year. According to Indonesia Statistic Bureau (Badan Pusat Statistik) using population indicator based on their monthly expense below the line of poverty can be categorized as poor people. The increasing amount of the poor can trigger criminality, that is because those individuals will do anything to make ends meet. By predicting the amount of the poor, hopefully the government or any related institution can help decrease poverty and unemployment rate in Indonesia. Artificial neural network backpropagation is one of the method that can be used to make predictions. Weight and bias in backpropagation's training optimized using genetic algorithm to obtain more optimal results. In this artificial neural network backpropagation research method that the weight training optimized using genetic algorithm generate 8.744579% AFER points.

***Keywords**: Artificial neural network, backpropagation, genetic algorithm, time-series*

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Allah SWT yang telah memberikan rahmat, karunia dan hidayah-Nya sehingga penulis dapat menyelesaikan skripsi dengan judul "**IMPLEMENTASI JARINGAN SARAF TIRUAN *BACKPROPAGATION* UNTUK MEMPREDIKSI JUMLAH PENDUDUK MISKIN DI INDONESIA DENGAN OPTIMASI ALGORITME GENETIKA**". Skripsi ini diajukan sebagai ujian seminar skripsi dalam rangka untuk memperoleh gelar Sarjana Komputer di Fakultas Ilmu Komputer (FILKOM), Program Studi Teknik Informatika, Universitas Brawijaya Malang. Atas terselesaikannya skripsi ini, penulis mengucapkan rasa terima kasih kepada:

1. Indriati, S.T, M.Kom selaku dosen pembimbing pertama skripsi yang telah meluangkan waktu dan juga memberikan banyak pengarahan bagi penulis.
2. Ir.Sutrisno, M.T selaku dosen pembimbing kedua skripsi yang telah meluangkan waktu dan juga memberikan banyak pengarahan bagi penulis.
3. Wayan Firdaus Mahmudy, S.Si., M.T., Ph.D selaku dekan Fakultas Ilmu Komputer Universitas Brawijaya.
4. Segenap bapak dan ibu dosen yang telah mendidik dan mengajarkan ilmunya kepada penulis selama menempuh pendidikan di Fakultas Ilmu Komputer Universitas Brawijaya.
5. Orang tua yang telah yang telah memberikan semangat dan juga memberikan banyak bantuan moril maupun materiil selama perkuliahan hingga penyelesaian tugas akhir ini.
6. Seluruh teman – teman SAP 2016 Fakultas Ilmu Komputer Universitas Brawijaya yang telah memberi dorongan dan semangat.
7. Segenap staff dan karyawan di Fakultas Ilmu Komputer Universitas Brawijaya yang telah banyak membantu penulis dalam pelaksanaan penyusunan skripsi.

Penulis menyadari bahwa dalam menyelesaikan tugas akhir ini, tentunya masih jauh dari kesempernuaan. Maka dari itu kritik dan saran yang sifatnya membangun sangat diharapkan untuk kesempurnaan karya berikutnya.

Semoga penyusunan skripsi ini dapat berguna, bermanfaat bagi penulis maupun pembaca.

Malang, 28 Maret 2018

Penulis

arthurjuliorisa@gmail.com

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Kemiskinan adalah definisi dari kelompok atau perorangan dalam masyarakat dengan keadaan yang kurang sejahtera dan sulit untuk mencukupi seluruh kebutuhan dasar mereka. Menurut Kaplale (2012) negara berkembang atau terbelakang merupakan negara dengan penduduk yang pendapatan perkapitanya rendah. Banyak negara di dunia yang menjadikan kemiskinan sebagai masalah inti dari negara tersebut, Indonesia adalah salah satu dari negara berkembang dengan penduduk miskin yang cukup banyak. Menghapus kemiskinan merupakan misi utama dari suatu negara. Berbagai macam konsep kemiskinan telah diadaptasi dan dikaji oleh banyak negara, namun belum menghasilkan dampak yang diinginkan, seperti contoh Indonesia yang merupakan satu dari negara berkembang yang lain, yang masih memiliki persoalan tentang kemiskinan meskipun sudah berumur 71 tahun. Menurut Badan Pusat Statistik (2016) dengan indikator penduduk yang memiliki pengeluaran perbulan dibawah garis kemiskinan dikategorikan sebagai rakyat kurang mampu.

Dari banyaknya jumlah masyarakat miskin menandakan bahwa banyak pula jumlah pengangguran yang ada di Indonesia (Anindita, 2017), yang membuat turunnya daya saing dan beli di masyarakat. Hal tersebut menunjukkan bahwa lapangan pekerjaan yang dibutuhkan masih kurang untuk memenuhi kebutuhan mereka, lapangan pekerjaan yang kurang dapat memicu terjadinya tindak kekerasan, karena masayarakat yang ada pada situasi tersebut akan menggunakan apapun yang mereka bisa agar dapat memenuhi kebutuhannya meskipun dengan cara yang salah.

Terdapat berbagai macam metode yang bisa dipakai untuk memprediksi seperti algoritme genetika yang digunakan oleh (Witarayoga, 2016) dengan menggunakan variabel bebas yang mempengaruhi variabel lainnya untuk memprediksi tingkat kemiskinan di Indonesia. Kemudian *Backpropagation neural network* yang ditulis oleh (Yohannes, Mahmudy, & Rahmi, 2015) guna mengoptimalkan penentuan upah minimum di kota malang berdasarkan tingkat inflasi. Selanjutnya penelitian dari (Haviluddin & Alfred, 2015) yang menunjukkan *Backpropagation* yang dioptimasi mendapatkan hasil yang lebih optimal untuk memprediksi aktifitas lalu lintas. Dan penelitian (Adwandha, Ratnawati, & Adikara, 2017) mendapatkan hasil lebih optimal untuk *Backpropagation* yang dioptimasi menggunakan algoritma genetika dibandingkan dengan *Backpropagation* untuk memprediksi jumlah pengangguran terbuka.

Pada penelitian ini akan menggunakan Jaringan Syaraf Tiruan *Backpropagation* yang bobot pelatihannya akan dioptimasi menggunakan algoritme

genetika guna memprediksi jumlah penduduk miskin di Indonesia karena dari penelitian – penelitian sebelumnya yang mendapatkan hasil atau akurasi lebih optimal dibandingan menggunakan *Backpropagation* dengan data yang didapatkan dari laman https://www.bps.go.id/. Diharapkan dengan perkiraan jumlah penduduk miskin pada tahun – tahun berikutnya pemerintah ataupun lembaga – lembaga yang terkait dengan topik ini dapat membantu untuk mengurangi jumlah penduduk miskin dan tingkat pengangguran di Indonesia.

1.2 Rumusan Masalah

Dari latar belakang yang telah dijelaskan, penulis dapat mengambil rumusan masalah untuk rujukan melakukan penelitian ini, antara lain:

a. Bagaimana mengimplementasikan metode *Backpropagation* dan Algoritme Genetika dalam memprediksi jumlah penduduk miskin di Indonesia?
b. Menghitung tingkat akurasi penggunaan metode *Backpropagation* dan Algoritme Genetika untuk memprediksi jumlah penduduk miskin di Indonesia ?

1.3 Tujuan

Dari rumusan masalah yang telah ditentukan, berikut merupakan tujuan dari penelitian ini, antara lain:

a. Mengimplementasikan metode *Backpropagation* dan Algoritme Genetika dalam memprediksi jumlah penduduk miskin di Indonesia.
b. Menghitung tingkat akurasi penggunaan metode *Backpropagation* dan Algoritme Genetika untuk memprediksi data penduduk miskin di Indonesia.

1.4 Manfaat

Manfaat dari penelitian ini diantaranya, dapat berupa:

a. Mengetahui tingkat akurasi penggunaan metode *Bacpropagation* dengan Algoritme Genetika untuk memprediksi data penduduk miskin di Indonesia.
b. Pemerintah dapat memprediksi jumlah penduduk miskin di Indonesia agar dapat dilakukan pembinaan untuk dientaskan dari kemiskinan.
c. Lembaga atau Instansi lain yang bergerak di bidang kemanusiaan di Indonesia dapat memprediksi besar bantuan yang diperlukan pada penduduk miskin di Indonesia.

1.5 Batasan Masalah

Dari pokok masalah yang ditelti, maka pada penelitian ini, beberapa batasan masalahnya, antara lain:

a. Data yang digunakan adalah data jumlah penduduk miskin di Indonesia, dengan rentan waktu dari tahun 1998 sampai 2017.
b. Data yang digunakan bersumber pada laman resmi Badan Pusat Statistik Nasional (www. bps.go.id).

1.6 Sistematika Pembahasan

Sistematika pembahasan akan menunjukkan struktur kerangka penulisan dari penelitian ini yang tersusun dari beberapa bab, antara lain :

BAB 1 PENDAHULUAN

Pada bab ini terdapat latar belakang, rumusan masalah, tujuan, batasan masalah dan juga manfaat dari metode *Backpropagation* yang bobot pelatihannya dioptimasi menggunakan Algoritme Genetika untuk memprediksi jumlah penduduk miskin yang ada di Indonesia.

BAB 2 LANDASAN KEPUSTAKAAN

Pada Bab ini akan dideskripsikan beberapa dasar teori yang digunakan untuk mendukung penelitian yang akan dilakukan dalam memprediksi jumlah penduduk miskin yang ada di Indonesia.

BAB 3 METODOLOGI PENELITIAN

Pada bab ini akan diuraikan metode dan langkah – langkah yang digunakan untuk menyelesaikan masalah dengan menggunakan metode *Backpropagation* dengan Algoritme Genetika guna memprediksi jumlah penduduk miskin di Indonesia.

BAB 4 PERANCANGAN DAN IMPLEMENTASI

Pada bab ini akan dijelaskan seluruh perancangan yang dibuat antara lain diagram alir, perhitungan manual, perancangan antarmuka, dan perancangan ujicoba yang digunakan dalam penelitian ini. Kemudian Mengimplementasikan dan membahas metode *Backpropagation* dan Algoritme Genetika dalam memprediksi jumlah penduduk miskin di Indonesia.

BAB 5 PENGUJIAN DAN ANALISIS

Bab ini akan dideskripsikan skenario dari pengujian yang kemudian menganalisis hasil dari pengujian dalam memprediksi jumlah penduduk miskin di Indonesia, dengan menggunakan metode *Backpropagation* dan Algoritme Genetika.

BAB 6 PENUTUP

Bab ini memuat kesimpulan berdasarkan hasil penelitian dalam memprediksi jumlah penduduk miskin di Indonesia, menggunakan metode *Backpropagation* dengan Algoritme Genetika, kemudian akan dituliskan beberapa saran yang bertujuan untuk memberi ide atau masalah baru untuk diteliti pada penelitian berikutnya.

BAB 2 TINJAUAN PUSTAKA

2.1 Tinjauan Pustaka

Tinjauan pustaka yang digunakan pada penetilian ini sebanyak tiga tinjauan pustaka. Penelitian pertama dari Amina dan Irawan yang memprediksi jumlah penduduk miskin dengan metode Jaringan Saraf Tiruan *Backpropagation* yang berada di Kalimantan Selatan. Data dari tahun 2002 sampai tahun 2011 digunakan Amina dan Irawan dimana data tersebut berbentuk data bulanan. Kemudian struktur yang digunakan berupa lapisan input yang memiliki 4 masukan, lapisan hidden dengan 2 neuron, dan 1 lapisan out sebagai hasilnya, yang menghasilkan akurasi sebesar 99.98% (Amina & Irawan, 2014).

Selanjutnya penelitian kedua dari Lestari yang menggunakan algoritme *Backpropagation* guna memprediksi penjualan jamur, peneletian yang dilakukan Lestari menggunakan nilai MSE untuk mengetahui tingkat akurasinya dan didapatkan nilai MSE sebesar 0.00099976 (Lestari, 2017).

Kemudian penelitian ketiga dari Haviluddin & Alfred yang menggunakan metode *genetic-based backpropagation* untuk memprediksi aktifitas lalu lintas jaringan, penelitian ini membandingkan dua metode yaitu jaringan syaraf tiruan *backpropagation* dengan *genetic-based backpropagation*. Pada penelitian tersebut ditunjukkan bahwa metode *genetic-based backpropagation* lebih unggul dibandingkan metode jaringan syaraf tiruan *backpropagation* dengan tingkat akurasi yang dilihat dari nilai MSE sebesar 0.009636 dan 0.000565 (Haviluddin & Alfred, 2015). Tingkat akurasi yang tinggi dihasilkan dari nilai MSE yang semakin rendah. Seluruh tinjauan pustaka ditunjukkan pada Tabel 2.1.

Tabel 2.1 Tinjauan Pustaka

No	Penulis	Objek	Metode	Hasil
1	(Amina & Irawan, 2014)	Penduduk Miskin	*backpropagation*	Menghasilkan Akurasi sebesar 99.98%
2	(Lestari, 2017)	Penjualan Jamur	*backpropagation*	Menghasilkan nilai MSE sebesar 0.00099976

3	(Haviluddin & Alfred, 2015)	Aktifitas lalu lintas jaringan	*genetic-based backpropagation*	Menghasilkan nilai MSE sebesar 0.009636 dan 0.000565 dari metode *backpropagation* dan *genetic-based backpropagation*

2.2 Kemiskinan

Kemiskinan adalah keadaan yang mendefinisikan tentang kelompok yang kesulitan untuk memenuhi kebutuhan hidupnya dilihat dari aspek ekonomi. Terdapat beberapa pemikiran pada konsep kemiskinan. Kemiskinan *absolute* merupakan konsep kemiskinan yang melihat dari jumlah pendapatan bulanan terhitung kurang untuk mencukupi atau membeli beberapa kebutuhan utama (Noor, 2014). Kemudian konsep selanjutnya yaitu kemiskinan *relative* yang melihat kemiskinan dari membandingkan antara pendapatan yang diperoleh dengan kebutuhannya. Selanjutnya adalah kemiskinan struktural yaitu suatu kemiskinan yang diperoleh karena dilahirkan atau berada pada suatu tempat dengan keadaan tersebut. Kemudian yang terakhir adalah kemiskinan kultural, kemiskinan tersebut didapat karena adanya suatu budaya yang mengikat dan menjadikan kelompok masyarakat tersebut sulit untuk keluar dari keadaan tesebut.

Dalam sejarah Indonesia pada tahun 1970 sampai awal tahun 1990an, Indonesia berhasil menurunkan jumlah penduduk miskin. Sejumlah 28,6% *proverty head count* dapat diturunkan pada periode tersebut dan pada tahun 1997 saat Indonesia dilanda krisis ekonomi, kemiskinan naik kembali yang titik puncaknya terdapat di tahun 1999 dengan jumlah 23.3% (World Bank, 2006). Tetapi pada tahun 2005, kemiskinan kembali turun dengan jumlah 16%, meskipun begitu kemiskinan di Indonesia kembali meningkat pada tahun 2006 dengan jumlah 1.75% yang terjadi karena tidak diperbolehkannya impor beras yang menjadikan nilai jual beras naik, hal tersebut berhubungan karena kemiskinan umumnya bergantung dengan nilai jual suatu bahan pokok yang mayoritas penduduk Indonesia terbiasa memakan beras yang telah dimasak sebagai makanan pokoknya.

2.3 Prediksi Data *Time Series*

Prediksi adalah proses untuk menentukan suatu nilai untuk masa yang akan datang dan terdapat beberapa metode yang digunakan didalamnya. Prediksi biasanya digunakan untuk menentukan segi kualitas, ukuran, kuantitas, waktu dan tempat guna memenuhi permintaan suatu jasa atau barang. Terdapat dua macam konsep dalam prediksi, yaitu pendekatan secara kualitatif dan kuantitatif (Pakaja, Naba, & Purwanto, 2012).

Time Series adalah kelompok atau kumpulan data yang terbentuk dari rentan waktu tertentu dari suatu kejadian. Dari data *time series* biasanya dapat dianalisis dan digunakan untuk mengetahui perkembangan dari suatu kejadian serta melihat korelasi dan kejadian satu dengan yang lain (Nugroho K. , 2012).

Pada peramalan terdapat empat macam pola data (Pakaja, Naba, & Purwanto, 2012) antara lain:

1. *Trend*: Suatu pola yang membutuhkan waktu cukup lama untuk mengetahui peningkatan atau penurunan suatu kejadian.
2. *Seasonality* (musiman): Data berpola yang terpengaruh karena adanya terpengaruh musim tertentu.
3. *Cycles* (siklus): Jika terdapat data yang bergelombang dengan durasi 1 tahun yang terpengaruh oleh faktor politik atau ekonomi yang membentuk suatu pola.
4. *Horizontal*/*Stationary*/*Random Variation*: yaitu suatu pola yang tidak teratur karena tidak adanya konsistensi data dan tidak membentuk pola yang jelas.

2.4 Jaringan Syaraf Tiruan

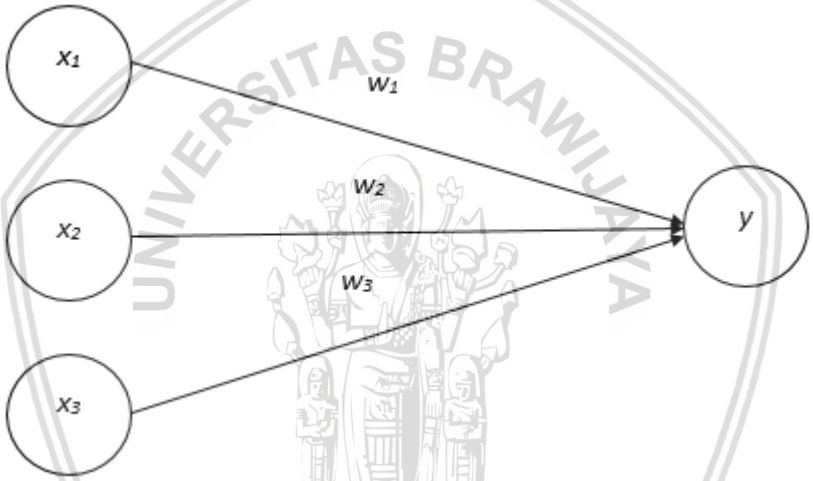
Jaringan syaraf tiruan umumnya biasa disebut dengan *neural network* adalah metode yang meniru cara kerja dan struktur syaraf dari otak manusia. Dimana otak dapat menyimpan, menirukan, dan mengenal suatu hal dengan cara mempelajari hal tersebut (Fausett, 1994). Hal tersebut yang menjadikan metode ini memiliki proses yang sama seperti otak manusia yang akan melakukan proses pembelajaran terlebih dahulu sebelum menggunakan sistem (Bustomi, Bisri, & Purwanti, 2014).

2.4.1 Struktur Jaringan Syaraf Tiruan

Jaringan saraf tiruan memiliki beberapa struktur yang terdapat beberapa elemen di dalamnya antara lain dinamakan bobot atau biasa disebut dengan *weight*, kemudian batas yang biasa disebut dengan *threshold*, dan yang terakhir fungsi aktivasi. Terdapat berbagai macam istilah pada jaringan syaraf tiruan, yaitu *input, output, dan hidden layer*. *Input* adalah proses untuk menghasilkan sinyal

output dengan memproses nilai masukan. *Output* adalah hasil yang akan dicapai dari nilai *input*, kemudian *hidden layer* adalah lapisan yang menghubungkan lapisan *input* dan *output* dengan tujuan memperluas kemampuan dari metode ini.

Terdapat penghubung dari neuron satu menuju neuron lainnya yang memiliki bobot tertentu pada setiap penghubung. Bobot menunjukkan seberapa besar pengaruh dari informasi tersebut untuk mendapatkan solusi. Jaringan syaraf tiruan dapat menangani beberapa masalah, antara lain prediksi, klasifikasi, pengenalan pola, dan optimasi. Pada Gambar 2.1 akan ditunjukkan arsitektur jaringan saraf tiruan sederhana.



Gambar 2.1 Arsitektur Jaringan Saraf Tiruan Sederhana

Gambar 2.1 menunjukkan bahwa terdapat 3 *neuron input* yaitu x_1, x_2, dan x_3 menuju *neuron output y* yang terhubung oleh bobot w_1, w_2, dan w_3. Pada persamaan 2.1 menunjukkan penjumlahan bobot sinyal dari *neuron* x_1, x_2, dan x_3 untuk menghasilkan *Net input* (*y_in*) (Fausett, 1994).

$$y\_in = w_1x_1 + w_2x_2 + w_3x_3 \quad \textbf{(2.1)}$$

Persamaan 2.2 merupakan permaan dari funsi aktifasi ne*uron y*.

$$f(x) = \frac{1}{1+\exp(-x)} \quad \textbf{(2.2)}$$

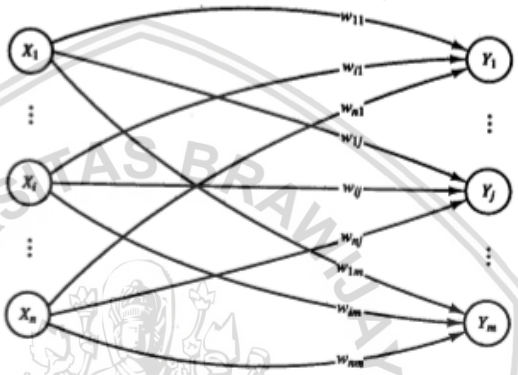
2.4.2 Macam – Macam Arsitektur Jaringan Saraf Tiruan

Pada Jaringan saraf tiruan memiliki beberapa macam arsitektur antara lain *single layer*, *multilayer,* dan *competitive layer*. Untuk menentukan jumlah lapisan pada suatu arsitektur, unit *input* tidak dihitung, karena pada unit

tersebut tidak terdapat suatu proses perhitungan atau hanya digunakan untuk lapisan masukan suatu nilai. Bobot yang terhubung dapat digunakan untuk mendefinisikan jumlah *layer* dari suatu jaringan.

a. Arsitektur *Single Layer*

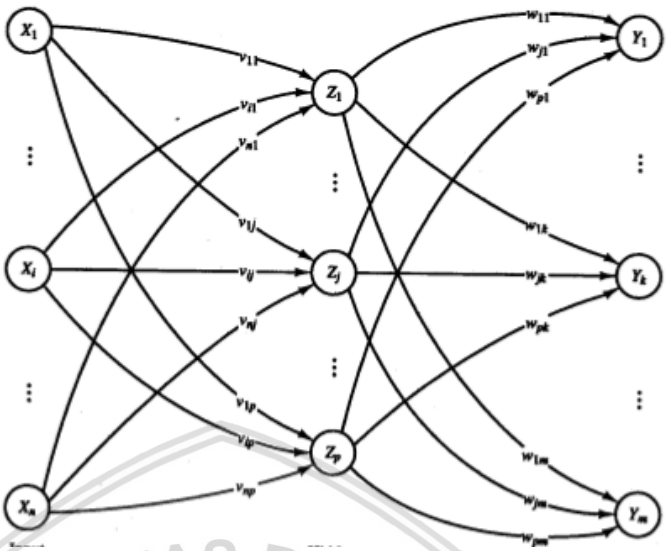
Jaringan syaraf tiruan dengan arsitektur *single layer* mempunyai 1 *layer* dari koneksi bobot. Pada unit input *single layer* akan langsung menuju *output* dan tidak terhubung ke unit *input* yang lain. Jaringan syaraf tiruan dengan arsitektur *single layer* dapat dilihat pada Gambar 2.2.



Gambar 2.2 Arsitektur Jaringan Single Layer

b. Arsitektur *Multilayer*

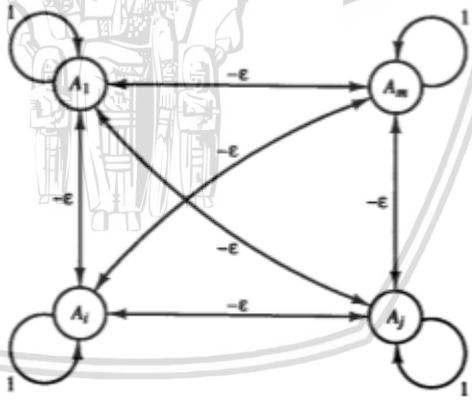
Arsitektur *multilayer* adalah arsitektur yang menggunakan satu atau banyak lapisan dari unit masukan atau *input* ke lapisan hasil atau *output*. Arsitektur *multilayer* memilih kelebihan dalam menyelesaikan perkara yang lebih rumit dari pada yang bisa dilakukan *single layer*, tetapi dalam melakukan *training* atau pelatihannya akan lebih rumit dibandingkan dengan arsitektur *single layer*. Jaringan Syaraf Tiruan dengan arsitektur *multilayer* ditunjukkan pada Gambar 2.3.



Gambar 2.3 Arsitektur Jaringan *Multilayer*

c. Arsitektur *Competitive Layer*

Pada arsitektur *competitive layer* tidak seperti arsitektur jaringan syaraf tiruan sebelumnya, pada arsitektur ini seluruh neuron saling terhubung. Arsitektur *competitive layer* dapat dilihat pada Gambar 2.4.



Gambar 2.4 Arsitektur Jaringan *Competitive Layer*

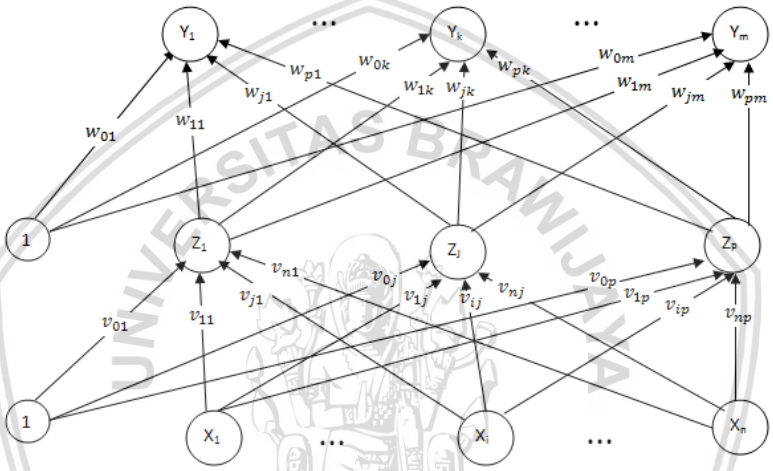
2.5 *Backpropagation*

Terdapat beberapa variasi algoritme dari metode Jaringan Syaraf Tiruan, salah satu macamnya adalah algoritme *Backpropagation*. Proses pembelajaran algoritme ini dilakukan dengan cara menyesuaikan bobot berdasarkan *error* yang dihasilkan. Dasar dari algoritma ini yaitu menghitung dampak setiap bobot terhadap errornya (Rumelhart, Hinton, & Williams, 1986). Di dalam proses pelatihannya akan dilakukan *training* pada jaringan agar jaringan dapat seimbang

dalam menggunakan banyak pola untuk menghasilkan keluaran optimal dari *input*-an (Hudiyawan, 2015).

2.5.1 Arsitektur *Backpropagation*

Terdapat beberapa istilah di dalam arsitektur algoritme *backpropagation* yaitu lapisan input yang biasa dilambangkan dengan *x*, kemudian bias yang dibiasa dilambangkan dengan 1, kemudian lapisan *hidden* yang dilambangkan dengan *z*, kemudian *output* atau *layer* terakhir dengan lambang *y*. akan ditunjukkan di Gambar 2.5.



Gambar 2.5 Arsitektur Algoritme *Backpropagation*

V_{ij} : melambangkan bobot dari lapisan *input* X_i menuju Z_j

V_{oj} : melambangkan bobot dari lapisan bias 1 menuju Z_j

W_{jk} : melambangkan bobot dari Z_j menuju Y_k

W_{0k} : melambangkan bobot dari lapisan bias di *hidden layer* menuju Y_j

2.5.2 Alur *Backpropagation*

Pada saat proses *training*, seluruh unit *output* (Y_k) akan melihat selisih hasil perhitungan dari target t_k untuk dapat ditentukan nilai error dari pola yang telah dibuat. Kemudian menghitung faktor δ_k (k = 1, …, m) yang diperoleh dari hasil *error*-nya. Menggunakan variabel δ_k guna pendistribusian selisih nilai

prediksi dan nilai asli dari lapisan *output* Y_k kembali menuju setiap unit pada lapisan sebelumnya. Menggunakan Δ_k guna perbaruan nilai bobot antara lapisan *hidden* dan lapisan keluaran. Kemudian menggunakan tahapan yang sama antara lapisan *hidden* dan *input layer*, dengan menghitung faktor δ_j (j = 1, …, p) di lapisan *hidden* (Fausett, 1994). Setelah seluruh faktor δ terdefinisi, maka seluruh bobot juga akan menyesuaikan pada seluruh lapisan secara bersamaan. Tabel 2.2 Akan dijelaskan lambang – lambang pada algoritme *backpropagation*.

Tabel 2.2 Istilah pada *backpropagation*

Lambang	Keterangan
x	input unit pada proses *training* x = (x_1, …, x_i, …, x_n).
t	target dari proses prediksi t = (t_1, …, t_k, …, x_m).
δ_k	Besaran koreksi *error* pada penyesuaian bobot pada w_{jk}.
Δ_j	Besaran koreksi *error* pada penyesuaian bobot pada v_{ij}.
α	*Learning rate.*
X_i	Input unit *i*. untuk menamai sebuah input unit, sinyal *input,* dan *output*.
v_{0j}	Bias yang berada di unit *hidden j.*
Lambang	Keterangan
Z_j	*hidden* unit *j*.
w_{0k}	Bias yang berada pada unit *output k.*
Y_k	*output* unit *k*.

Sumber: (Fausett, 1994)

Penjumlahan dari perkalian v_{ij} dan x_i digunakan untuk mendapatkan nilai z_in_j yang persamaannya ditunjukkan pada Persamaan 2.3.

$$\boldsymbol{z\_in_j = v_{0j} + \sum_i x_i v_{ij}} \qquad (2.3)$$

Hasil dari Persamaan 2.3 akan dimasukkan pada fungsi aktivasi untuk mendapatkan nilai z_j yang persamaannya ditunjukkan pada Persamaan 2.4.

$$\boldsymbol{z_j = f(z\_in_j)} \tag{2.4}$$

Nilai y_in_k didapatkan dengan menjumlahkan hasil dari perkalian fungsi aktivasi (z_j) dan bobot w_{ij} yang persamaannya ditunjukkan pada Persamaan 2.5.

$$\boldsymbol{y\_in_k = w_{0j} + \sum_i z_i w_{ij}} \tag{2.5}$$

Hasil dari persamaan 2.5 akan dimasukkan kedalam fungsi aktivasi yang bertujuan untuk mendapatkan y_k yang persamaannya ditunjukkan pada Persamaan 2.6.

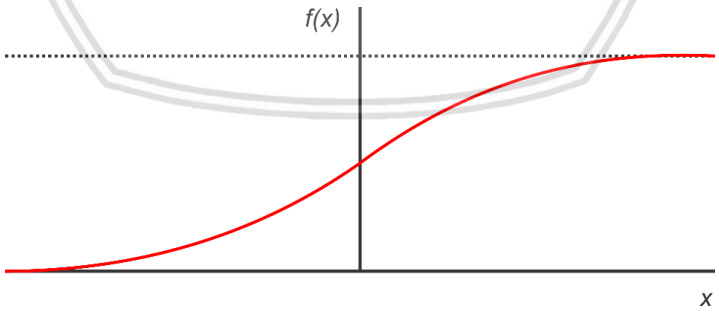
$$\boldsymbol{y_k = f(y\_in_k)} \tag{2.6}$$

2.5.3 Fungsi Aktivasi

Terdapat berbagai macam karakteristik pada fungsi aktivasi *Backpropagation* antara lain *differentiable, continuous, easy to compute,* dan *monotonically non-decreasing*. Fungsi *binary sigmoid* adalah salah satunya, fungsi ini mempunyai rentang (range) (0, 1), yang ditunjukkan pada Persamaan 2.7 dan digambarkan pada Gambar 2.6.

$$\boldsymbol{f_1(x) = \frac{1}{1+\exp^{(-x)}}} \tag{2.7}$$

$$\boldsymbol{f'_1(x) = f_1(x)[1 - f_1(x)]} \tag{2.8}$$

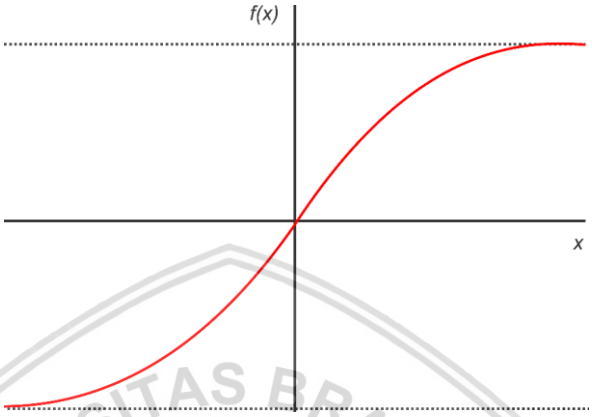


Gambar 2.6 *Binary Sigmoid*, dengan *Range* (0, 1)

Kemudian bipolar sigmoid adalah fungsi aktivasi selanjutnya. Fungsi aktivasi ini mempunyai rentang (range) (-1, 1), yang ditunjukkan diPersamaan 2.9, kemudian turunannya ditunjukkan pada Persamaan 2.10 dan dijelaskan di Gambar 2.7.

$$f_2(x) = \frac{2}{1+\exp^{(-x)}} - 1 \qquad (2.9)$$

$$f'_2(x) = \frac{1}{2}[1 + f_2(x)][1 - f_2(x)] \qquad (2.10)$$



Gambar 2.7 *Bipolar Sigmoid*, dengan *Range* (-1, 1)

2.5.4 Tahapan *Backpropagation*

Tahapan-tahapan di dalam proses pengolahan data dengan menggunakan algoritme *backpropagation*, diantaranya yaitu tahap pelatihan, atau bias disebut dengan proses *training*, Dimana dalam tahapan ini terdapat proses pengolahan yang diolah secara sistematis. Secara detail ditunjukkan dalam Tabel 2.3.

Tabel 2.3 Tahapan *Backpropagation*

Tahap	Keterangan
Tahap 0	Membangkitkan nilai bobot awal
Tahap 1	Melakukan tahap 2-9, jika pengelolaan belum mencapai titik berhenti
Tahap 2	Lakukan tahap 3-8 , untuk setiap pasangan *training*
Tahap 3	*Feedforward*: Setiap unit *input* (X_i, $i = 1, \ldots, n$) akan menyebarkan sinyal menuju ke *hidden unit* dengan membawa nilai (bobot) dari tiap unitnya.

Tahap 4	Seluruh unit pada lapisan *hidden* (Z_j, $j = 1, \ldots, p$) akan menghasilkan fungsi aktivasi bertujuan guna menentukan sinyal keluaran dengan persamaan 2.4 dari proses penjumlahan bobot masukkannya, dengan Persamaan 2.3. Selanjutnya sinyal yang dihasilkan akan meneruskannya menuju semua unit setelahnya (output).
Tahap 5	Semua unit *output* (Y_k, $k = 1, \ldots, m$) akan menghasilkan fungsi aktivasi sehingga dapat menghitung sinyal *output* dengan Persamaan 2.6 dengan cara menjumlahkan bobot sinyal *input* menggunakan Persamaan 2.5.
Tahap 6	*Backpropagation of error:* Dari semua unit pada lapisan *output* atau keluaran (Y_k, $k = 1, \ldots, m$) mendapatkan hasil pola perhitungan untuk tujuannya dan digunakan guna masukan pola pelatihan, selanjutnya mencari dengan melakukan perhitungan besaran kesalahannya dengan Persamaan 2.11, dan mencari dengan melakukan perhitungan nilai pembaruan bobot untuk memperbaiki nilai bobot sebelumnya dengan persamaan 2.12, juga memperbaiki nilai bobot biasnya dengan menggunakan persamaan 2.13, sehingga dapat mengirim sinyal δ_k ke unit-unit dibawahnya.
Tahap 7	Semua unit hidden (Z_j, $j = 1, \ldots, p$), dengan menggunakan persamaan 2.14 delta *input*-nya (dari unit layer dibawahnya) dijumlahkan, selanjutnya menghitung besaran kesalahan dari tiap unitnya menggunkan persamaan 2.15, dan digunakan sebagai acuan untuk melakukan perbaikan nilai bobot selanjutnya (v_{ij}) dengan persamaan 2.16, juga melakukan pembaruan bobot pada nilai bias (v_{0j}) dari Persamaan 2.17.

Tahap	Keterangan
Tahap 8	*Update* bobot dan bias: Semua unit keluaran (Y_k,$k=1,\ldots,m$) memperbaiki nilai bias-nya dan bobot-nya ($j = 0,\ldots,p$) dengan Persamaan 2.18. Di lapisan hidden semua unitnya (Z_j,$j=1,\ldots,p$) memperbaiki nilai bobot dan biasnya ($i=0,\ldots,n$) dari Persamaan 2.19.
Tahap 9	Menguji seluruh faktor penentu batas, yang telah ditetapkan.

Sumber: (Fausett, 1994)

Persamaan 2.11 digunakan guna melakukan proses perhitungan δk didalam proses *backpropagation* error, hal tersebut menggambarkan besaran kesalahan pada unit *output*.

$$\boldsymbol{\delta_k = (t_k - y_k) f'(y\_in_k)} \quad \textbf{(2.11)}$$

Persamaan 2.12 ditujukan untuk memperbarui nilai bobot dari w_{jk}.

$$\boldsymbol{\Delta w_{jk} = \alpha \delta_k z_j} \quad \textbf{(2.12)}$$

Persamaan 2.13 digunakan untuk melakukan proses perhitungan, sehingga didapat nilai baru bias sehingga dapat memperbarui besaran bias w_{0k}.

$$\boldsymbol{\Delta w_{0k} = \alpha \delta_k} \quad \textbf{(2.13)}$$

Persamaan 2.14 bertujuan guna melakukan perhitungan seberapa besar kesalahan dilapisan hidden.

$$\boldsymbol{\delta\_in_j = \sum_{k=1}^{m} \delta_k w_{jk}} \quad \textbf{(2.14)}$$

Hasil dari Persamaan 2.14 akan dimasukkan di Persamaan 2.15 yang merupakan fungsi aktivasi, bertujuan guna menentukan nilai kesalahan pada *hidden unit*.

$$\boldsymbol{\delta_j = \delta\_in_j f'(z\_in_j)} \quad \textbf{(2.15)}$$

Persamaan 2.16 digunakan sebagai faktor penentu perubahan (perbaikan) bobot yang berada pada *input unit* menuju ke *hidden unit*.

$$\boldsymbol{\Delta v_{ij} = \alpha \delta_j x_i} \quad \textbf{(2.16)}$$

Persamaan 2.17 digunakan sebagai faktor penentu perubahan (perbaikan) bobot yang berada pada nilai bias ke *hidden unit*.

$$\boldsymbol{\Delta v_{0j} = \alpha \delta_j} \quad \textbf{(2.17)}$$

Persamaan 2.18 digunakan untuk menghitung besaran nilai bonot yang baru berdasakan penjumlahan dari besaran nilai bobot *input unit* menuju ke *hidden unit* sebelumnya dengan faktor penentu perubahan nilai bobot.

$$\boldsymbol{w_{jk}(baru) = w_{jk}(lama) + \Delta w_{jk}} \quad \textbf{(2.18)}$$

Persamaan 2.19 digunakan untuk menghitung besaran nilai bonot yang baru berdasakan penjumlahan dari besaran nilai bobot *hidden unit* menuju ke *output unit* sebelumnya dengan faktor penentu perubahan nilai bobot.

$$\boldsymbol{v_{ij}(baru) = v_{ij}(lama) + \Delta v_{ij}} \quad \textbf{(2.19)}$$

2.6 Perhitungan MSE

Pada penelitian ini menggunakan *Mean Square Error* (MSE) untuk menghitung *error*-nya. Perhitungan *error* digunakan guna mengetahui tingkat kesalahan yang memiliki tujuan untuk mengetahui seberapa optimal metode dapat menghasilkan solusi dari suatu permasalahan atau dalam penelitian ini melakukan prediksi. MSE merupakan suatu fungsi untuk menghitung selisih antara nilai dari hasil prediksi dengan nilai sebenarnya. Pada Persamaan 2.20 akan ditunjukkan persamaan MSE (Deborah & Prathap, 2014). Tingkat kesalahan selaras dengan besar kecilnya nilai MSE, jika nilai MSE yang didapatkan semakin kecil, menunjukkan tingkat kesalahan yang dihasilkan semakin kecil pula, dan juga sebaliknya.

$$\boldsymbol{MSE = \frac{1}{n}\sum_{i=1}^{n}(Y'_i - Y_i)^2} \quad \textbf{(2.20)}$$

Keterangan :

MSE : Nilai *Mean Square Error*

n : jumlah setiap nilai yang diprediksi

Y' : hasil dari prediksi

Y : nilai asli atau sebenarnya

2.7 Perhitungan AFER

Digunakan AFER untuk menghitung tingkat kesalahannya selain fungsi MSE. Average Forecasting Error Rate atau biasa disebut dengan AFER adalah perhitungan yang menunjukkan selisih dari hasil prediksi dan data aktual dalam bentuk presentase (Syukriyawati, 2015). bertambah kecilnya nilai AFER yang dihasilkan berarti bertambah baik pula akurasi yang didapatkan. Persamaan fungsi AFER akan ditampilkan pada Persamaan dibawah ini.

$$AFER = \frac{\sum |\frac{Ai - Fi}{Ai}|}{n} * 100\% \quad \textbf{(2.21)}$$

Keterangan :

AFER : nilai Average Forecasting Error Rate

Ai : nilai asli

Fi : nilai keluaran proses prediksi

n : banyaknya data

2.8 Kecerdasan Buatan

Dibidang teknik dan sains, terdapat bidang baru yang dikenal dengan kecerdasan buatan umunya dikenal dengan *Artificial Intelligence*(*AI*) (Russell & Norvig, 1994). Membangun dan memahami entitas cerdas adalah tujuan utama dari bidang ini. Berdasarkan rasionalitas dan kinerjanya (*performance*) *AI* dapat didefinisikan.

Tabel 2.4 Matriks Definisi Kecerdasan Buatan

(Thinking humanly) Pemikiran layaknya manusia	*(Thinking rationally)* Pemikiran secara rasional
(Acting humanly) Melakukan perbuatan seperti manusia	*(Acting rationally)* Melakukan perbuatan secara rasional

Sumber: (Russell & Norvig, 1994)

2.9 Algoritme Genetika

Pada algoritme evolusi atau biasa disebut dengan EA terdapat beberapa macam algoritme, antara lain algoritme genetika atau biasa disebut GA, kemudian *Evolution Strategies* atau biasa disebut ES, selanjutnya g*enetic programming* (GP) dan *evolutionary programming* (EP). Namun seiring berjalannya jaman, algoritme genetika merupakan algoritme yang dapat berkembang mengikutinya. Algoritme ini dapat digunakan pada berbagai bidang seperti fisika ekonomi, biologi dan bidang lain karena algoritme ini mampu menangani suatu permasalahan yang kompleks terkait optimasi (Mahmudy, 2015).

2.9.1 Fitness

Fitness merupakan representasi dari suatu solusi dari masalah yang dicari. Fungsi fitness sendiri digunakan untuk mencari individu mana yang merupakan solusi terbaik dilihat dari semakin tinggi nilai fitness dari individu tersebut (Mahmudy, 2015). Pada penelitian ini menggunakan fungsi fitness yang ditunjukkan pada persamaan 2.3.

$$\boldsymbol{fitness = \frac{1}{MSE}} \quad \textbf{(2.22)}$$

2.9.2 Tahapan Algoritme Genetika

Terdapat beberapa tahapan di dalam Algoritme Genetika antara lain inisialisasi, reproduksi, evaluasi, dan seleksi. Tahap inisialisasi merupakan proses yang akan membangkitkan secara acak individu – individu dengan susunan gen tertentu. Selanjutnya tahap reproduksi yang akan dilakukan persilangan individu untuk menghasikan individu baru yang disebut dengan *offspring* pada setiap populasi. Kemudian tahap evaluasi yang merupakan proses untuk mengetahui seberapa baik solusi yang dihasilkan oleh setiap individu, solusi didapatkan dengan melakukan perhitungan fitness dimana semakin tinggi nilai yang dihasilkan berarti solusi yang didapat juga akan semakin baik. Yang terakhir merupakan tahap seleksi, pada tahap ini individu dengan solusi terbaik akan dipertahankan untuk lanjut pada generasi berikutnya dengan jumlah populasi yang telah ditentukan (Mahmudy, 2015).

a. Inisialisasi
   Proses ini secara acak akan dilakukan pembangkitan himpunan individu awal dan himpunan tersebut dikelompokkan menjadi suatu populasi atau *popSize*. setiap individu merupakan sebuah solusi dari masalah yang dipecahkan. (Mahmudy, 2015).
b. Reproduksi
   Pada proses ini akan dilakukan reproduksi antar individu pada populasi untuk mendapatkan keturunan baru yang disebut dengan *offspring*. Terdapat dua tahapan pada proses reproduksi, yaitu *crossover* dan mutasi. Pada tahapan ini *Crossover Rate*(Cr) dan Mutation Rate(Mr) harus ditentukan untuk menentukan banyaknya offspring yang akan dihasilkan sebesar (*cr x popsize*) untuk keturunan dari *crossover* dan (*mr x popsize*) untuk keturunan dari mutasi. (Mahmudy, 2015). Pada tahap crossover terdapat bermacam – macam metode, salah satunya yaitu one-cut-point. Metode ini akan memilih dua individu induk secara acak untuk ditukar susunan kromosomnya yang sebelumnya akan ditentukan titik pertukarannya. Kemudian pada tahap mutasi juga memiliki berbagai-macam metode dan reciprocal exchange adalah salah satunya. Metode ini

secara acak akan memilih satu individu induk dan dua titik untuk dilakukan pertukaran gen.

c. Evaluasi

Pada proses ini akan dicari tingkat kebugaran(*fitness*) dari masing – masing individu termasuk *offspring* yang diperoleh dari proses reproduksi. Pada penelitian ini, semakin tinggi tingkat kebugaran yang didapatkan maka semakin baik pula individu tersebut dijadikan sebagai sebuah solusi.

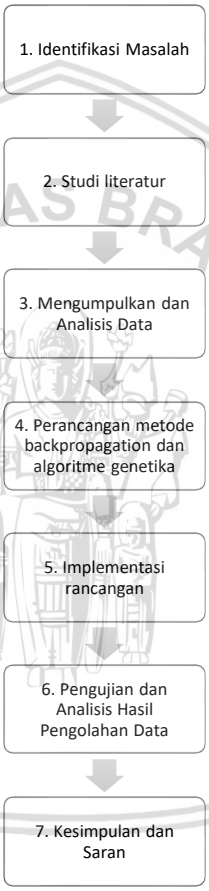
d. Seleksi

Pada proses ini akan mempertahankan individu termasuk *offspring* untuk lanjut ke generasi berikutnya. Pada proses ini terdapat berbagai macam metode, dan elitism selection adalah salah satunya. Metode ini akan mengurutkan seluruh individu dalam populasi termasuk *offspring* dari yang memiliki tingkat kebugaran paling tinggi hingga paling rendah, selanjutnya individu dengan tingkat kebugaran terbaik sebanyak *popsize* yang telah ditentukan sebelumnya akan diambil untuk lanjut ke generasi berikutnya (Mahmudy, 2015).

BAB 3 METODOLOGI PENELITIAN

3.1 Tahapan Penelitian

Pada penelitian ini terdapat beberapa tahapan, antara lain:



Gambar 3.1 Tahapan Penelitian

1. Identifikasi Masalah
   Masalah pada penelitian ini akan dideskripsikan dengan jelas, hal tersebut bertujuan untuk mendapatkan pokok dari permasalahannya.
2. Studi Literatur
   Beberapa literatur terkait akan dipelajari guna membantu dan mendukung penelitian ini.
3. Mengumpulkan dan Analisis Data

Tahapan ketiga akan dilakukan pengumpulan data, kemudian menganalisa data tersebut guna memperoleh informasi yang bermanfaat untuk mendukung penelitian ini.

4. Perancangan metode *Backpropagation* dan algoritme genetika
   Setelah dilakukan analisa data, akan dilakukan perancangan arsitektur jaringan syaraf tiruan *backpropagation* yang kemudian pelatihan bobotnya akan dioptimasi menggunakan algoritma genetika.
5. Implementasi
   Perancangan yang dilakukan pada proses sebelumnya akan diimpementasikan menggunakan bantuan komputer berupa perangkat lunak NetBeans.
6. Pengujian Hasil Pengolahan Data
   Setelah implementasi dilakukan, akan dilakukan pengujian dari sistem yang telah dibuat, hal tersebut bertujuan mengetahui sistem yang dihasilkan sesuai dengan apa yang sudah dirancang.
7. Kesimpulan dan Saran
   Seluruh penelitian yang telah dilakukan akan disimpulkan untuk mengetahui hasil dari penelitian ini dan memberikan saran guna penelitian berikut dengan topik terkait.

3.2 Identifikasi Masalah

Masalah kemiskinan yang ada diberbagai negara di dunia, salah satunya Indonesia, akan berdampak kepada negara lain dalam melihat negara tersebut dan bagaimana ekonomi didalamnya yang dikelola. Karena pada umumnya angka kriminalitas yang tinggi biasanya terdapat pada negara dengan penghasilan perkapita yang relatif rendah, karena masyarakat tersebut akan melakukan cara apaun untuk membantu mencukupi kebutuhan hidup mereka.

3.3 Studi Literatur

Penelitian ini akan menggunakan referensi jurnal dalam negeri dan juga jurnal internasional dimana jurnal – jurnal tersebut merupakan penelitian – penelitian sebelumnya yang terkait dengan topik yang dibahas.

3.4 Mengumpulkan dan Analisa data

Penelitian ini menggunakan data jumlah penduduk miskin di Indonesiia dari tahun 1998 sampai dengan 2017A yang diperoleh dari Badan Pusat Statistik pada laman https://www.bps.go.id/linkTableDinamis/view/id/1119. Data tersebut merupakan jumlah penduduk miskin dari pedesaan dan perkotaan yang pada tahun 2011 sampai 2017 dilakukan survey sebanyak 2 kali di bulan Maret sebagai semester pertamanya dan di bulan September sebagai semester kedua. Selanjutnya data tersebut digunakan untuk memprediksi jumlah penduduk miskin

pada semester berikutnya berdasarkan tahun sebelumnya. Seluruh data penduduk miskin akan ditampilkan pada Tabel 3.1.

Tabel 3.1 Data Kemiskinan di Indonesia Tahun 1998 – 2017

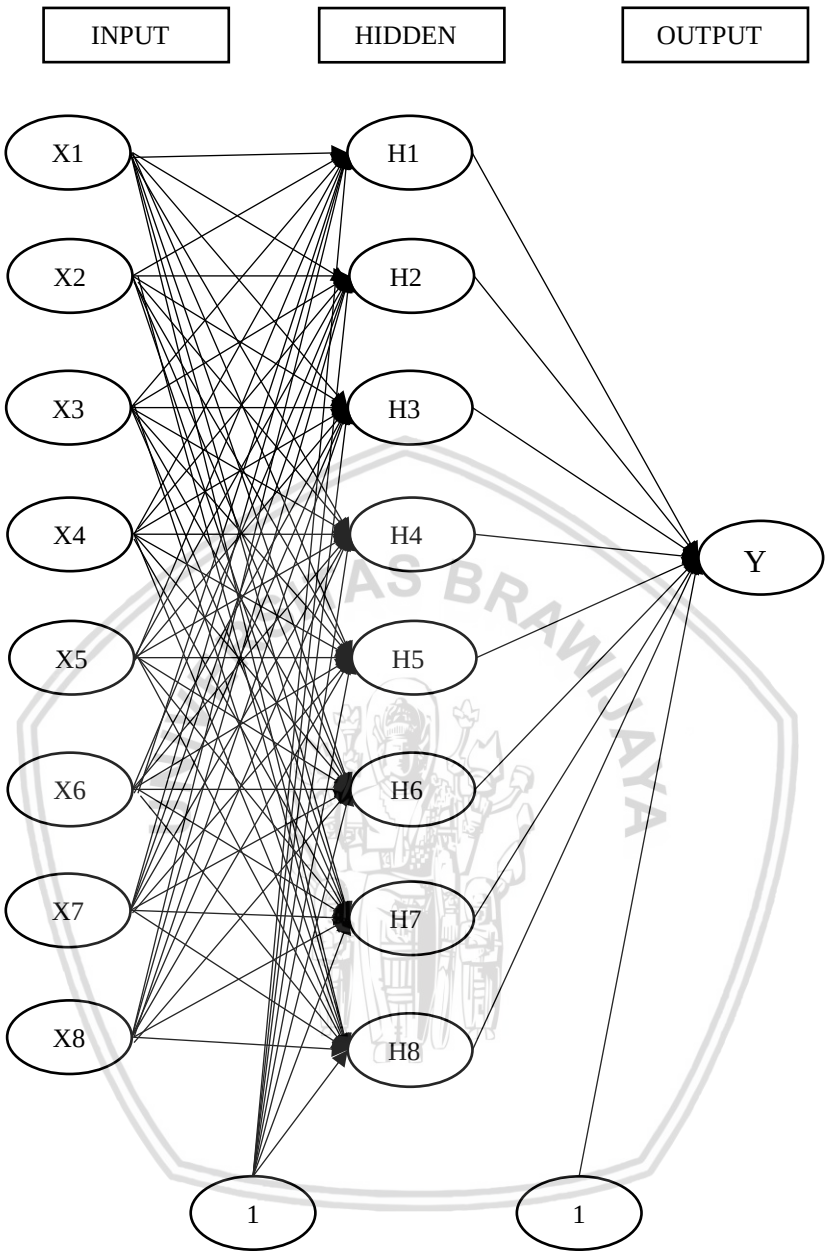
Tahun	Jumlah
1998	49500000
1999	47970000
2000	38740000
2001	37870000
2002	38390000
2003	37340000
2004	36150000
2005	35100000
2006	39300000
2007	37170000
2008	34960000
2009	32530000
2010	31020000
2011A	30020000
2011B	29890000
2012A	29130000
2012B	28590000
2013A	28070000
2013B	28550000
2014A	28280000
2014B	27730000
2015A	28590000
2015B	28510000
2016A	28010000
2016B	27760000
2017A	27770000

Sumber: (Badan Pusat Statistik 2017)

3.5 Perancangan Sistem

3.5.1 Perancangan Sistem

Pada penelitian ini akan menggunakan arsitektur jaringan syaraf tiruan yang ditampilkan oleh Gambar 3.2.

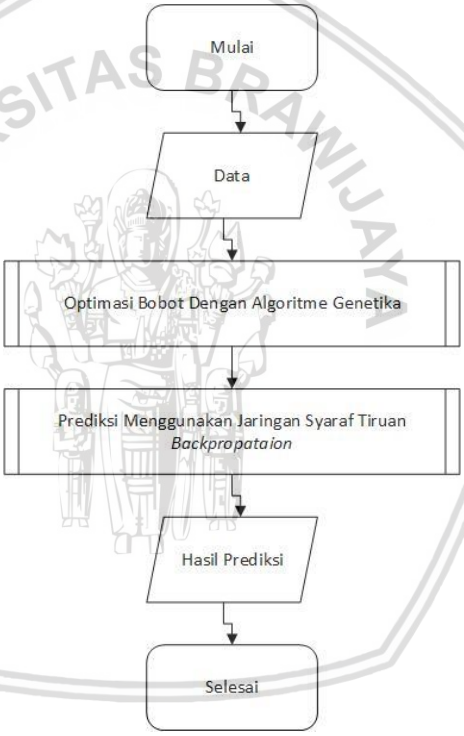


Gambar 3.2 Arsitektur Jaringan Syaraf Tiruan

Arsitektur jaringan syaraf tiruan yang digunakan mempunyai 3 layer yang terdiri dari lapisan *input*, lapisan *hidden* dan lapisan *output* atau solusi yang dihasilkan dari jaringan ini. Selanjutnya pada lapisan input terdapat neuron untuk masukan sistem sebanyak 8. Kemudian lapisan hidden yang merupakan lapisan penghubung lapisan input dan lapisan output terdapat neuron sebanyak 8. Kemudian pada lapisan output akan menghasilkan 1 buah output yang merupakan hasil prediksi jumlah penduduk miskin pada semester berikutnya.

Kemudian pada jaringan yang telah dibuat data akan dimasukkan dan dilakukan proses *feedforward*. Selanjutnya *output* atau hasil dari proses tersebut akan dibandingkan dengan nilai aslinya, lalu selisih dari hasil prediksi dan nilai aslinya digunakan untuk perbaikan bobot pada lapisan hidden dan input pada proses *backpropagation*. Kemudian data misklasifikasi akan diduplikasi dan ditambahkan yang bertujuan untuk dilakukan pelatihan kembali pada iterasi berikutnya.

Iterasi yang dilakukan akan berhenti di saat data misklasifikasi dianggap sudah optimal atau ditentukan batas berhentinya (*threshold*). Kemudian setelah bobot dinyatakan optimal, bobot tersebut lalu dipakai untuk memprediksi atau meramal jumlah penduduk miskin di Indonesia.



Gambar 3.3 Tahapan proses sistem secara umum

Berikut adalah penjelasan dari proses – proses yang akan dilakukan pada penenelitian ini menggunakan algoritme genetika untuk mengoptimasi prediksi menggunakan jaringan syaraf tiruan *backpropagation*:

1. Data yang diperoleh dimasukkan, data tersebut berupa data jumlah penduduk miskin di indonesia pada rentan atau jangka waktu tertentu yang telah ditentukan sebelumnya.

2. Menggunakan algoritme genetika untuk mencari bobot dan bias *w* yang memiliki fitness terbaik atau optimal.
3. Hasil bobot dan bias *w* dari algoritme genetika akan digunakan untuk parameter pada algoritme *backpropagation* untuk melakukan prediksi.
4. Sistem akan menampilkan hasil prediksi berupa jumlah penduduk miskin, MSE, dan iterasi setelah proses algoritme *backpropagation* selesai.

3.5.2 Normalisasi dan Denormalisasi Data

Pada penelitian ini akan dilakukan normalisasi data sebelum pelatihan dilakukan yang bertujuan untuk memperoleh data dengan format yang sama, persamaan untuk proses Normalisasi data akan ditunjukkan pada Persamaan 3.1.

$$x' = \frac{0.8(x-a)}{b-a} + 0.1 \quad \textbf{(3.1)}$$

Keterangan:

x': nilai hasil normalisasi proses normalisasi

x: nilai input

a: nilai minimum data yang digunakan

b: nilai maksimum data yang digunakan

Setelah data dinormalisasi dan dilakukan proses *backpropagation*, data akan dilakukan denormalisasi untuk mengetahui nilai asli dari hasil prediksi yang dinormalisasi. Yang ditunjukkan pada persamaan 3.2

$$y = \frac{(x'-0.1)(b-a)}{0.8} + a \quad \textbf{(3.2)}$$

Keterangan:

y: hasil proses denormalisasi

x': data hasil normalisasi

a: nilai maksimum data yang digunakan

b: nilai minimum data yang digunakan

3.6 Implementasi Sistem

Berdasarkan seluruh rancangan yang dibuat pada proses sebelumnya, implementasi akan dilakukan, antara lain implementasi lingkungan perangkat keras, perangkat lunak, implementasi metode yang digunakan pada aplikasi dan antar muka dari aplikasi yang sesuai dengan perancangan antar muka.

3.7 Pengujian dan Analisis

Pengujian sistem akan dilakukan pada tahap ini sesuai dengan rancangan pengujian yang dibuat dibab sebelumnya. Kemudian hasil yang didapatkan dari tahap pengujian akan dianalisis.

3.8 Kesimpulan dan Saran

Pada tahap kesimpulan dan saran, penarikan kesimpulan akan dibuat bersumber dari hasil yang diperoleh dari penelitian prediksi penduduk miskin dengan jaringan syaraf tiruan *backpropagation* yang bobot dan bias *w* pelatihannya dioptimasi dengan algoritme genetika. Selanjutnya memberikan saran yang ditujukan untuk penelitian selanjutnya dengan topik yang berkaitan.

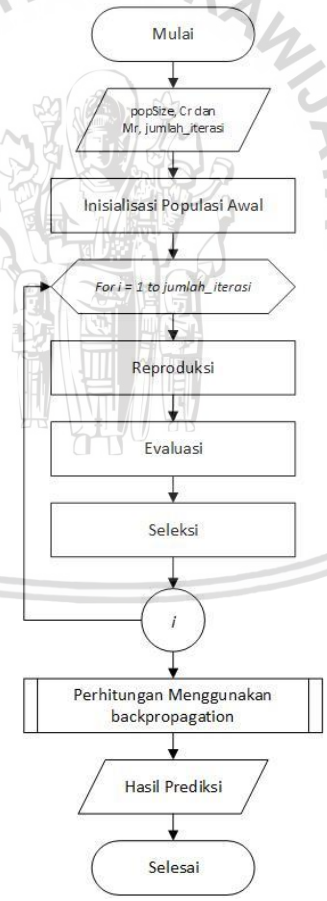
BAB 4 PERANCANGAN DAN IMPLEMENTASI

4.1 Perancangan Sistem

Pada sub bab perancangan sistem akan dijelaskan tentang seluruh rancangan sistem yang akan dibangun pada penelitian ini dengan menggunakan metode *backpropagation* untuk memprediksi, algoritme genetika sebagai metode untuk optimasi dan data – data yang digunakan.

4.1.1 Diagram Alir Sistem

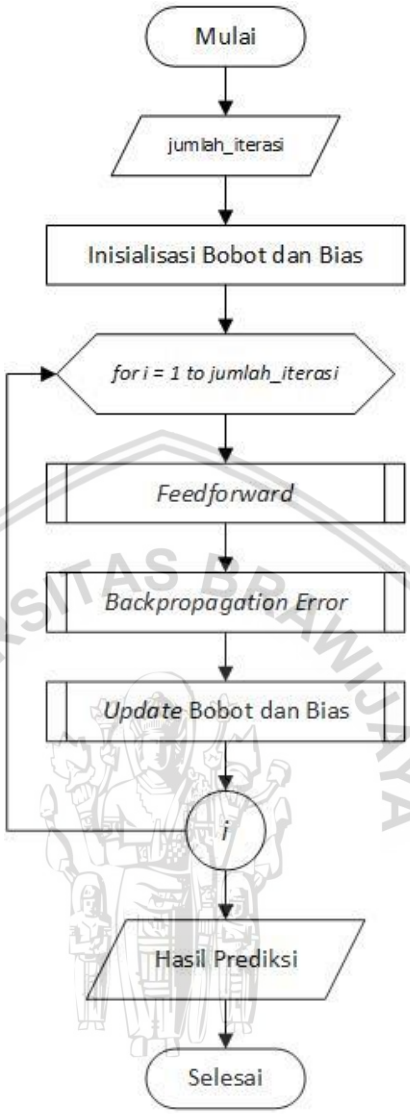
Diagram alir sistem digunakan untuk memperlihatkan tahapan – tahapan yang akan dilalui pada sistem. Pada Gambar 4.1 akan ditunjukkan tahapan yang ada pada algoritme genetika guna mengoptimasi bobot dan bias w.



Gambar 4.1 Diagram Alir Optimasi Bobot Menggunakan Algoritme Genetika

Gambar 4.1 merupakan beberapa tahap yang akan digunakan pada algoritme genetika. Yang dimana pada tahap ini algoritme genetika bertujuan untuk mengoptimasi bobot dan bias *w* pada proses prediksi yang menggunakan algoritme *backpropagation*. Tahapan – tahapan tersebut antara lain:

1. Memasukkan *input*-an pada sistem yaitu ukuran popSize, nilai Cr dan Mr dan ukuran generasi yang digunakan.
2. Populasi awal akan dibangkitkan secara acak untuk inputan pada proses training pada metode *backpropagation*.
3. Kemudian dilakukan proses reproduksi yang didalamnya terdapat proses *crossover* yang menggunakan metode *one-cut-point* dan proses mutasi yang menggunakan metode *reciprocal exchange*.
4. Selanjutnya dilakukan proses evaluasi yang bertujuan untuk menghitung fitness pada setiap individu.
5. Kemudian dilakukan proses seleksi yang menggunakan metode *elitism selectio*n.
6. Proses 3 sampai 5 akan dilakukan sebanyak jumlah generasi atau iterasi yang telah ditentukan sebelumnya.
7. Kemudian setelah proses sebelumnya selesai dilakukan sebanyak generasi yang ditentukan, hasil pada proses tersebut merupakan bobot dan bias *w* yang akan digunakan untuk melakukan proses training metode *backpropagation*.
8. Selanjutnya sistem akan mengeluarkan hasil prediksi dari proses perhitungan yang dilakukan ditahap sebelumnya.

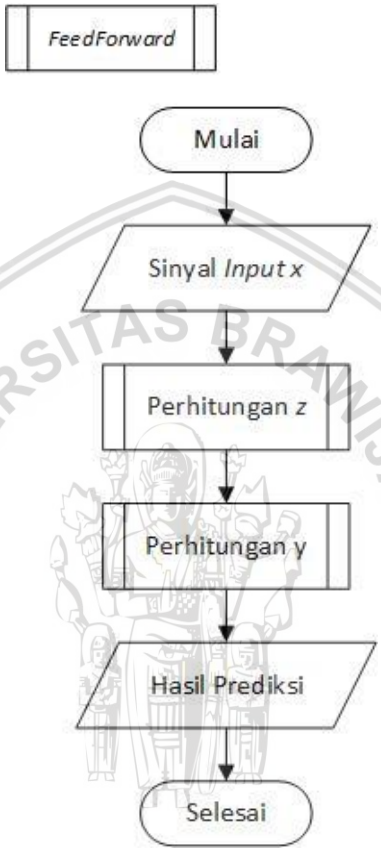


Gambar 4.2 Diagram Alir Algoritme *Backpropagation*

Gambar 4.2 merupakan tahapan yang akan dilakukan pada algoritme *backpropagation* untuk memprediksi jumlah penduduk miskin. Tahapan – tahapan tersebut antara lain:

1. Memberikan masukan pada sistem berupa jumlah iterasi.
2. Inisialisasi bobot dan bias dilakukan, dimana bobot dan bias *w* didapatkan dari hasil perhitungan pada algoritme genetika.
3. Setelah bobot dan bias telah diinisialisasi kemudian dilakukan proses *feedforward*.
4. Selisih hasil dari proses *feedforward* kemudian digunakan untuk melakukan tahap *backpropagation* error.

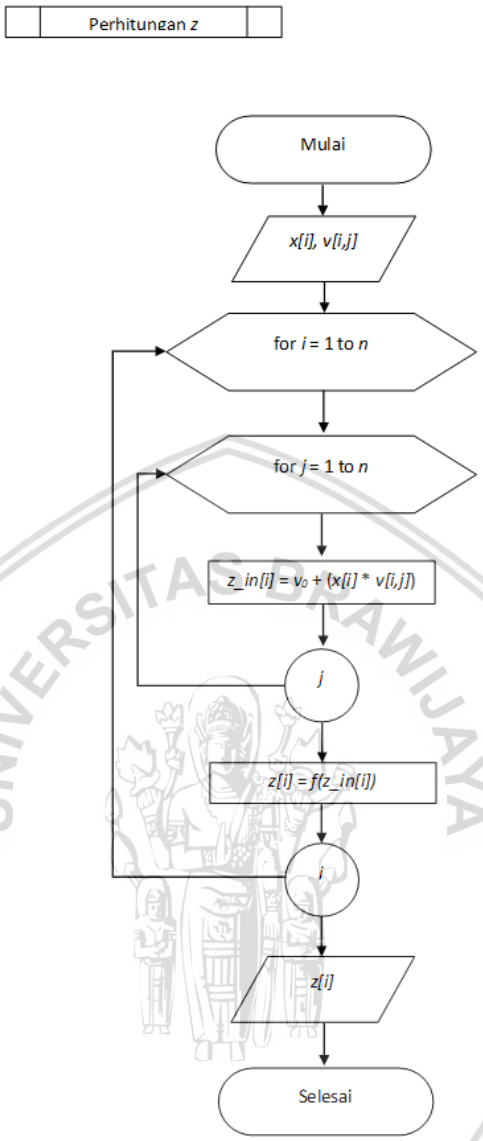
5. Selanjutnya dilakukan *update* pada bobot dan bias.
6. Tahapan 3 sampai 5 dilakukan terus menerus sebanyak jumlah iterasi yang ditentukan sebelumnya.
7. Sistem akan mengeluarkan hasil berupa hasil prediksi jumlah penduduk miskin pada semester berikutnya.



Gambar 4.3 Diagram Alir *Feedforward*

Gambar 4.3 merepresentasikan tahapan yang ada pada proses *feedforward* diantaranya sebagai berikut:

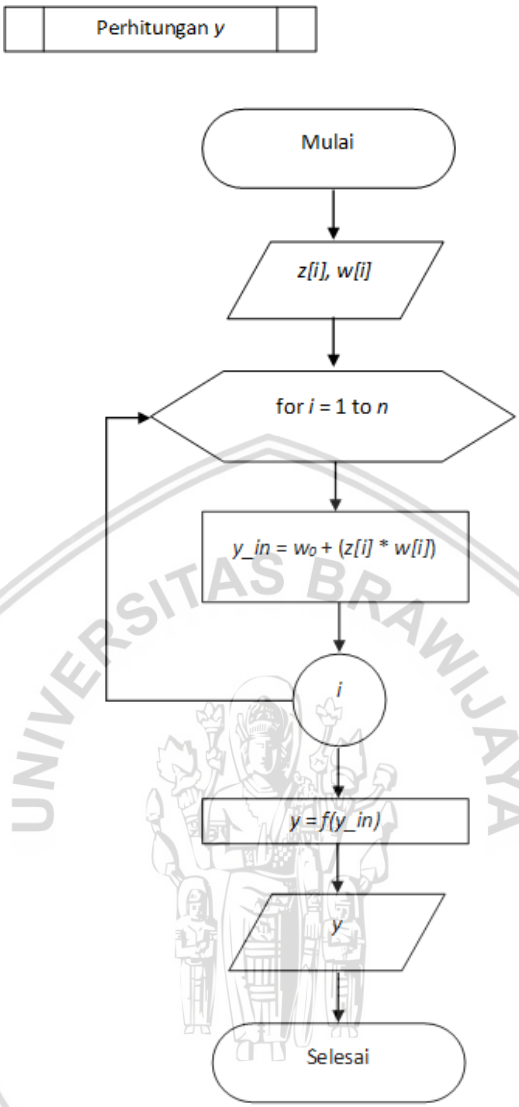
1. Sistem mendapatkan sinyal *input x* yang merupakan jumlah penduduk miskin pada tahun tertentu.
2. Mencari nilai output dengan melakukan perhitungan dari unit *hidden* (*z).*
3. Mencari nilai output dengan melakukan perhitungan dari unit *output (y).*
4. Sistem menghasilkan output berupa nilai *z* dan *y*.



Gambar 4.4 Diagram Alir Perhitungan *z*

Gambar 4.4 menjelaskan proses perhitungan nilai *output* dari unit *hidden* (*z)* diantaranya sebagai berikut:

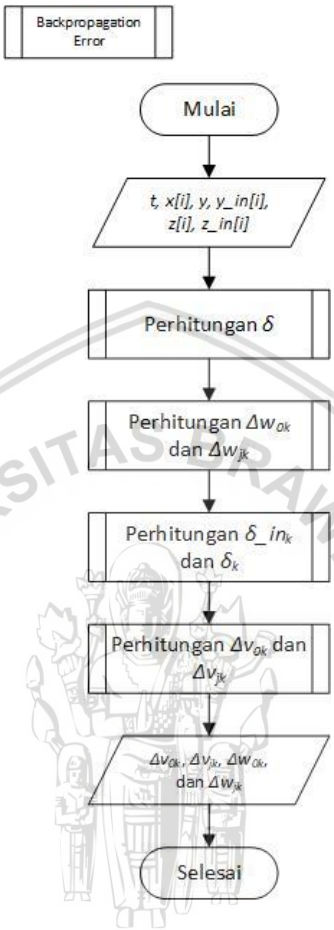
1. Sistem menerima *input*an sinyal *(x)* (*v)*.
2. Sistem menghitung nilai (*z_in*) dengan cara melakukan melakukan perulangan dan pada unit hidden menggunakan fungsi aktivasi *z = f(z_in)* untuk menghitung *output*nya.
3. Output berupa nilai *z* akan dihasilkan oleh sistem.



Gambar 4.5 Diagram Alir Perhitungan *y*

Gambar 4.5 menunjukkan proses perhitungan dari unit *output* (*y)* untuk nilai *output* diantaranya sebagai berikut:

1. Inputan sinyal *output* dari *hidden unit* (*z*) dan bobot (*w)* diterima sistem.
2. Pada sistem perulangan dilakukan guna mencari nilai sinyal (y_*in*) kemudian memasukkan fungsi aktivasi *y = f(y_in)* untuk menghitung nilai sinyal *output*nya.
3. Sistem menghasilkan output berupa nilai *y.*

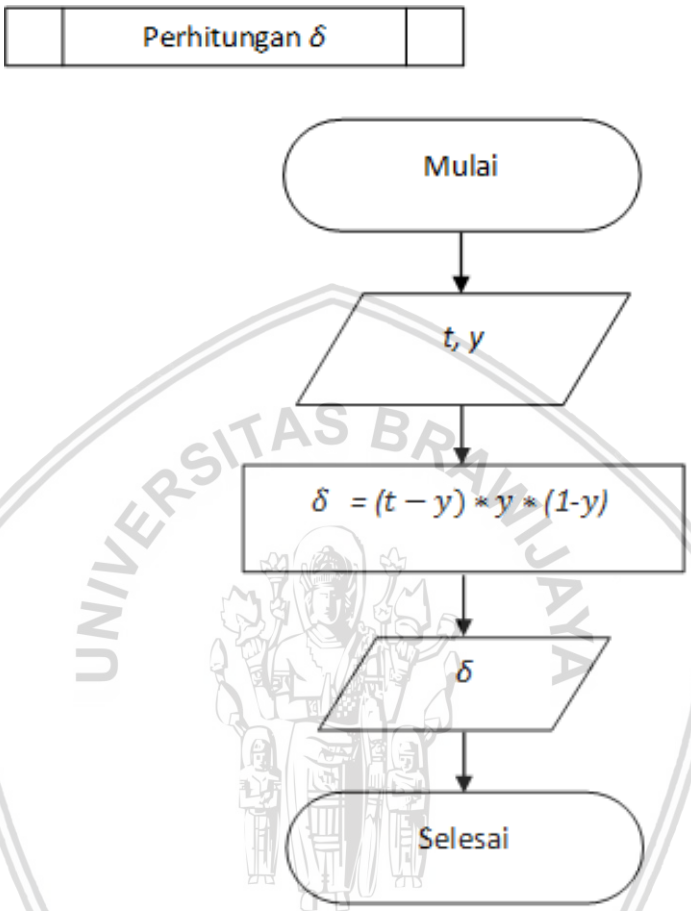


Gambar 4.6 Diagram Alir *Backpropagation Error*

Gambar 4.6 menunjukkan tahapan pada proses perhitungan *backpropagation error* diantaranya sebagai berikut:

1. *Inputan* berupa *t, x[i], y, y_in[i], z[i], z_in[i]* diterima sistem.
2. Sistem menghitung nilai δ guna mendapatkan tingkat *error* antara *output* dengan target.
3. Sistem mengitung nilai Δw_{0k} dan Δw_{jk} guna mendapatkan tingkat *error* bobot dan bias *w* yang berguna untuk *update* bobot *w*.
4. Sistem menghitung nilai δ_in_k dan δ_k.
5. Sistem menghitung nilai Δv_{0k} dan Δv_{jk} guna mendapatkan tingkat *error* bobot dan bias *v* yang berguna untuk *update* bobot *v*.

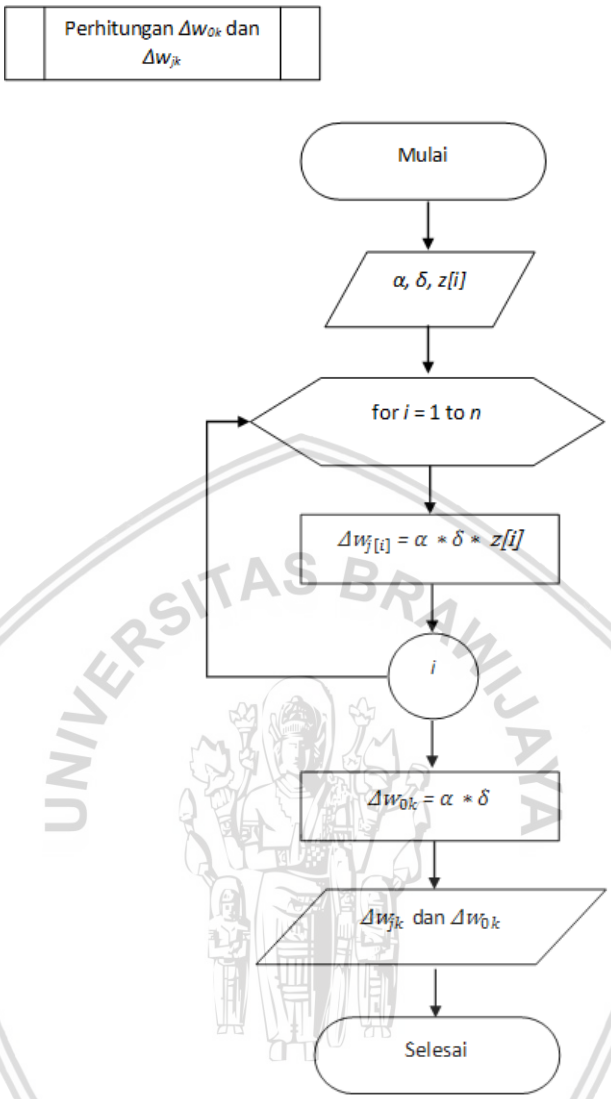
6. Akan dihasilkan *output* berupa nilai Δv_{0k}, Δv_{jk}, Δw_{0k}, dan Δw_{jk} oleh sistem.



Gambar 4.7 Diagram Alir Perhitungan *δ*

Gambar 4.7 menjelaskan proses perhitungan nilai δ diantaranya sebagai berikut:

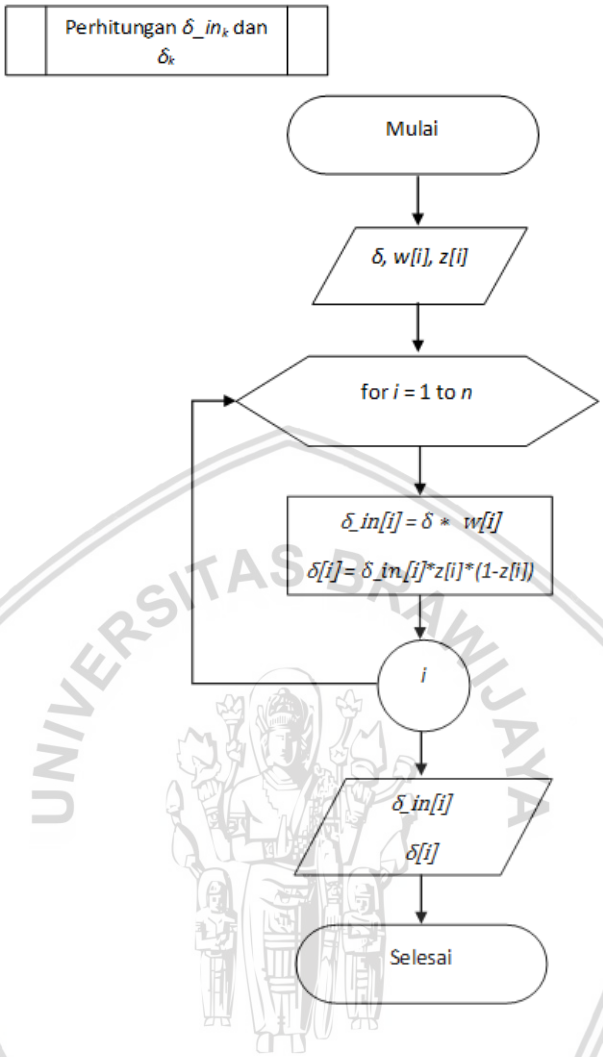
1. Sistem menerima *input*an berupa nilai *t* dan *y*.
2. Sistem menghitung nilai δ.
3. Sistem menghasilkan *output* berupa nilai δ.



Gambar 4.8 Diagram Alir Perhitungan Δw_{jk} dan Δw_{0k}

Gambar 4.8 menjelaskan seluruh proses perhitungan nilai Δw_{jk} dan Δw_{0k} diantaranya sebagai berikut:

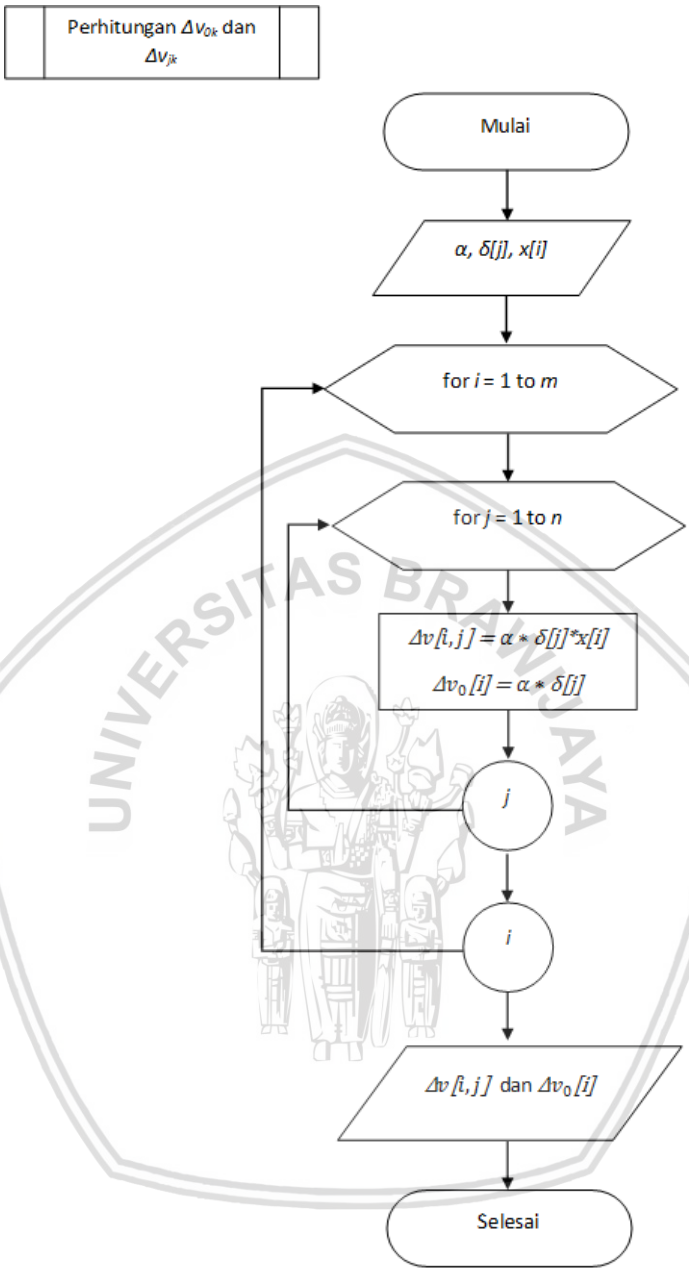
1. *Input*an berupa nilai α, δ, $z[i]$ diterima oleh sistem.
2. Perulangan dilakukan sistem guna menghitung nilai Δw_{jk}.
3. Sistem menghitung nilai Δw_{0k} setelah proses sebelumnya selesai.
4. Sistem menghasilkan *output* berupa nilai Δw_{jk} dan Δw_{0k}.



Gambar 4.9 Diagram Alir Perhitungan δ_in_k dan δ_k

Gambar 4.9 menjelaskan proses perhitungan nilai δ_in_k dan δ_k diantaranya sebagai berikut:

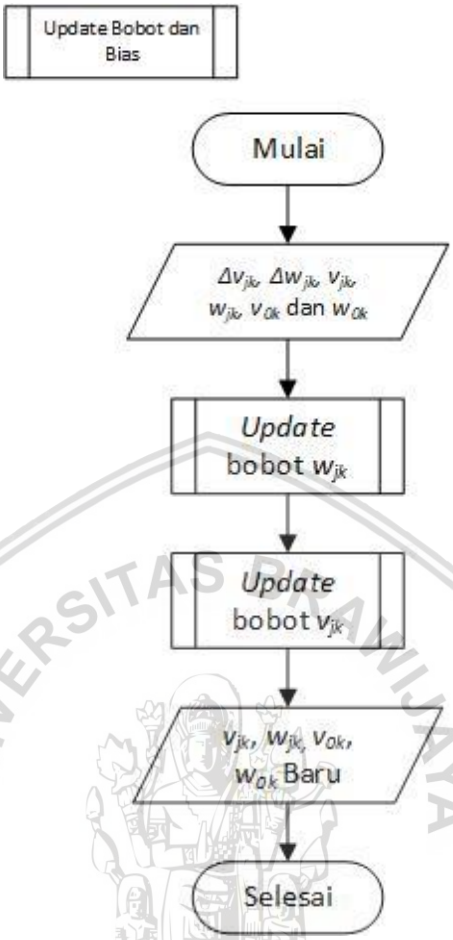
1. Sistem menerima *input*an berupa nilai *δ, w[i], z[i]*.
2. Sistem menghitung nilai δ_in_k dan δ_k.
3. Sistem menghasilkan *output* berupa nilai δ_in_k dan δ_k.
4. Sistem menghitung nilai δ_in_k dan δ_k.
5. Sistem menghitung nilai Δv_{0k} dan Δv_{jk} guna mendapatkan tingkat *error* bobot dan bias *v* yang berguna untuk *udpate* bobot *v*.
6. Akan dihasilkan *output* berupa nilai Δv_{0k}, Δv_{jk}, Δw_{0k}, dan Δw_{jk}.



Gambar 4.10 Diagram Alir Perhitungan Δv_{jk} dan Δv_{0k}

Gambar 4.10 Menunjukkan proses perhitungan nilai Δv_{jk} dan Δv_{0k} diantaranya sebagai berikut:

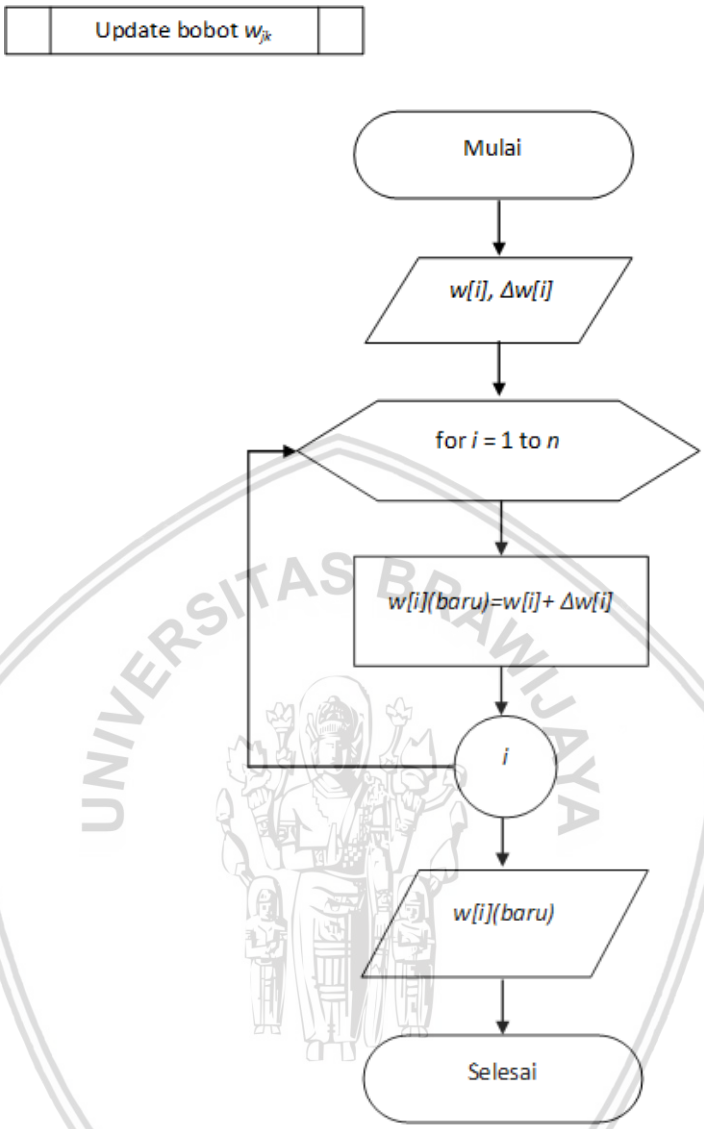
1. Sistem menerima *input*an nilai *α, δ[j], x[i]*.
2. Perulangan dilakukan sistem guna menghitung nilai Δv_{jk} dan Δv_{0k}.
3. Sistem menghasilkan keluaran nilai Δv_{jk} dan Δv_{0k}.



Gambar 4.11 Diagram Alir *Update* Bobot dan Bias

Gambar 4.11 menunjukkan seluruh proses untuk memperbarui bobot dan bias diantaranya sebagai berikut:

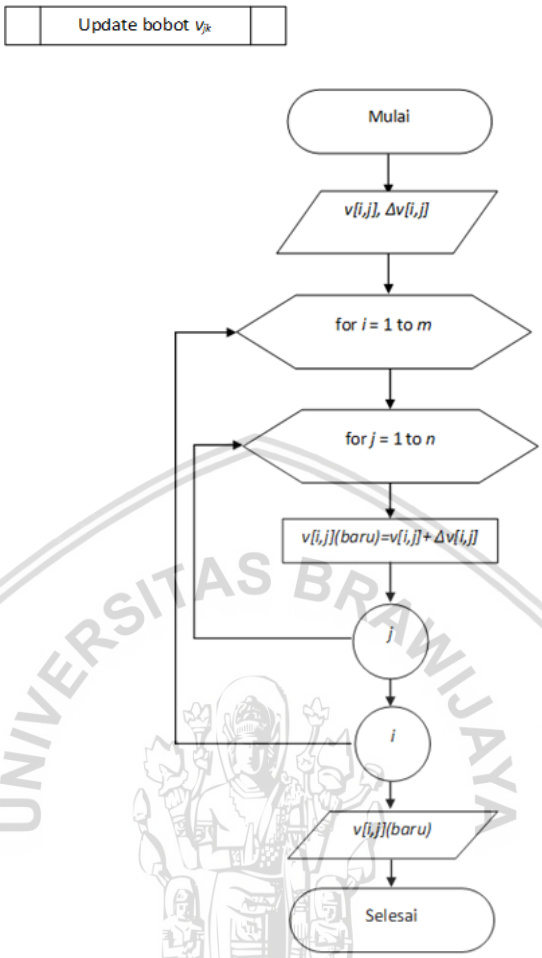
1. Sistem mendapatkan *input*an diantaranya nilai Δv_{jk}, Δw_{jk}, v_{jk}, w_{jk}, v_{0k} dan w_{0k}.
2. Sistem memproses perhitungan pembaruan bobot w_{jk}.
3. Sistem memproses perhitungan pembaruan bobot v_{jk}.
4. Sistem menghasilkan keluaran nilai bobot (v_{jk}, w_{jk}) dan bias (v_{0k}, w_{0k}) baru.



Gambar 4.12 Diagram Alir *Update* w_{jk} dan w_{0k}

Gambar 4.12 menunjukkan seluruh proses untuk memperbarui bobot dan bias (w_{jk}, w_{0k}) diantaranya sebagai berikut:

1. Sistem menerima *input*an berupa nilai Δw_{jk}, w_{jk}.
2. Memproses perhitungan untuk perbaruan bobot (w_{jk}) dan bias (w_{0k}) baru.
3. Sistem menghasilkan keluaran nilai bobot (w_{jk}) dan bias (w_{0k}) baru.



Gambar 4.13 Diagram Alir *Update* v_{jk} dan v_{0k}

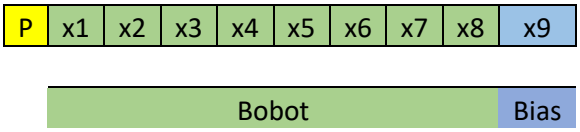
Gambar 4.13 menunjukkan seluruh proses untuk memperbarui bobot dan bias (v_{jk}, v_{0k}) diantaranya sebagai berikut:

1. Sistem menerima *input*an nilai Δv_{jk}, v_{jk}.
2. Memproses perhitungan untuk memperbarui bobot (v_{jk}) dan bias (v_{0k}) baru.
3. Sistem menghasilkan keluaran nilai bobot (v_{jk}) dan bias (v_{0k}) baru.

4.1.2 Perhitungan manual

Pada penelitian ini akan menggunakan representasi kromoson *real-code*. Kromosom merupakan susunan dari beberapa gen dari angka yang dibangkitkan secara acak dari range -0.1 sampai 0.9 karena nilai 0 atau 1 tidak hampir tidak pernah dicapai fungsi *sigmoid* (Jauhari, Himawan, & Dewi, 2016). Kromoson tersebut akan digunakan untuk menentukan nilai bobot dan bias pada Wij yang

optimal. Kromosom terbaik dapat dilihat dari nilai fitness yang dihasilkan. Nilai fitness didapatkan dari $\frac{1}{MSE}$. Bertambah besar nilai fitnessnya maka solusi yang dihasilkan akan bertambah baik. Representasi kromosom akan ditunjukkan pada Gambar 4.14.



Gambar 4.14 Representasi Kromosom

Gambar 4.14 merupakan bentuk dari representasi kromosom yang akan digunakan pada penelitian ini. Representasi kromosom memiliki panjang 9 gen dimana x1 sampai x8 merupakan bobot dan x9 merupakan bias. Pada perhitungan manual ini akan menggunakan *popsize* = 10, untuk tahap reproduksi akan menggunakan *crossover* dengan Cr = 0.2 dan mutasi sebesar Mr = 0.1.

Proses inisialisasi awal merupakan proses untuk menentukan populasi dan gen yang digunakan untuk pencarian solusi. Proses ini akan membentuk populasi sebanyak *popsize* yang telah ditentukan dan gen akan dimasukkan secara acak dari range 1 sampai banyaknya kode pada tiap gen. Tabel 4.1 akan menunjukkan populasi awal yang dibangkitkan secara acak.

Tabel 4.1 Populasi Awal

	Chromosom								
P	x1	x2	x3	x4	x5	x6	x7	x8	x9
p1	-0.7	0.6	0.8	-0.9	0.7	-0.9	-0.9	0.5	-0.7
p2	0.8	0.7	0.2	0.3	-0.5	0.3	0.1	0.4	-0.7
p3	-0.9	-0.6	-0.4	-0.5	0.4	0.7	-0.6	-0.4	-0.7
p4	0.6	-0.8	-0.2	0.3	-0.2	-0.7	0.2	0.5	0.1
p5	-0.2	-0.8	0.5	0.7	0.4	0.5	0.6	-0.2	-0.6
p6	0.5	0.5	-0.4	-0.9	0.6	0.6	-0.8	0.4	0.4
p7	-0.4	-0.2	0.8	0.5	-0.6	-0.3	-0.2	-0.1	0.7
p8	-0.6	0.7	0.4	0.2	0.7	0.8	0.2	0.4	0.1
p9	-0.5	0.7	0.1	-0.3	0.9	0.3	0.8	-0.1	0.9
p10	0.3	0.9	-0.9	0.3	0.6	-0.7	-0.2	-0.7	-0.1

Pada Tabel 4.1 ditunjukkan individu – individu yang dibangkitkan secara acak sejumlah *popsize* yang sebelumnya telah ditentukan. Setelah proses inisialisasi awal akan dilanjutkan pada tahap reproduksi.

Tahap reproduksi merupakan tahap dimana individu – individu yang telah dibangkitkan akan membuat keturunan yang hasil pada proses reproduksi akan ditempatkan pada penampungan *offspring*. Ada dua operator genetika yang akan digunakan pada tahap ini yaitu *crossover* dan *mutation*.

Proses *crossover* merupakan proses yang akan menggabungkan dua kromosom yang dipilih secara acak untuk dijadikan parent dalam menghasilkan kromosom baru (*offspring*). *Offspring* ditentukan dari parameter yang disebut dengan *crossover* rate (Cr). Metode *crossover* yang digunakan adalah One cut-point, yaitu pada metode ini akan ditentukan titik potong yang akan digunakan untuk titik acuan penukaran bagian kromosom yang satu dengan yang lain untuk menghasilkan *offspring*. Pada proses ini akan ditentukan 2 kromosom secara acak untuk dijadikan induk.

Dari pemilihan secara acak, induk yang terpilih adalah individu 2 dan 5. Kemudian dicari titik cutpoint secara acak yang menghasilkan cut point pada titik 6, yang akan ditunjukkan pada Gambar 4.15.

P2	0.8	0.7	0.2	0.3	-0.5	0.3	0.1	0.4	-0.7
P5	-0.2	-0.8	0.5	0.7	0.4	0.5	0.6	-0.2	-0.6
C1	0.8	0.7	0.2	0.3	-0.5	0.3	0.6	-0.2	-0.6
C2	-0.2	-0.8	0.5	0.7	0.4	0.5	0.1	0.4	-0.7

Gambar 4.15 Proses *Crossover*

Jumlah *offspring* ditentukan dengan cara *popsize* * *crossover* rate(Cr) yaitu 10 * 0.2 = 2 *offspring*. Jika pada proses *crossover* telah mendapatkan 2 anak, maka hanya akan dilakukan *crossover* sebanyak satu kali. Kemudian setelah proses *crossover* akan dilanjutkan tahap *mutation*.

Pada proses *mutation* ini metode yang digunakan adalah *reciprocal exchange mutation*. Cara kerja metode ini adalah dengan menentukan dua titik untuk dilakukan penukaran gen yang dipilih secara acak. Pada kasus ini proses yang akan dilakukan adalah menentukan induk yang kemudian dipilih dua titik yang akan digunakan untuk titik penukaran gen. Proses *mutation* memiliki parameter *mutation* rate(Mr) yaitu 0.1 dengan *popsize* sebesar 10. Untuk menentukan jumlah *offspring* yang dihasilkan yaitu Mr * *Offspring* yaitu 1 *offspring*.

Pada perhitungan manual ini individu yang terpilih secara acak untuk dilakukan *mutation* adalah individu 3 dan titik yang terpilih untuk penukaran gen adalah titik 4 dan 5. Pada kasus ini jika titik yang ditukar memiliki nilai yang lebih

tinggi dari nilai maksimum pada gen tersebut, maka nilai yang terpilih akan diganti dengan nilai maksimum pada gen tersebut. Proses *mutation* akan dijelaskan pada Gambar 4.16.

P3	-0.9	-0.6	-0.4	-0.5	0.4	0.7	-0.6	-0.4	-0.7
C3	-0.9	-0.6	-0.4	0.4	-0.5	0.7	-0.6	-0.4	-0.7

Gambar 4.16 Proses Mutasi

Pada 4.16 merupakan manualisasi dari proses mutasi yang kemudian akan dilanjutkan pada proses selanjutnya yaitu tahap Evaluasi dan Seleksi.

Pada tahap ini terdapat dua tahapan yaitu evaluasi dan seleksi. Perhitungan fitness pada seluruh individu dilakukan diproses evaluasi. Kemudian setelah selesai dilakukan perhitungan fitness akan dilanjutkan pada tahap seleksi. metode yang digunakan pada tahap seleksi adalah eltism, yaitu memilih individu sejumlah *popsize* yang ditentukan dengan melihat fitness yang paling tinggi.

Pada proses ini dari individu yang ada yaitu 10 dari *popsize*, 2 dari *crossover*, dan 1 dari *mutation* akan dipilih individu sejumlah *popsize* yang dapat lanjut ke generasi selanjutnya yaitu 10 individu. Hasil dari perhitungan fitness dapat dilihat pada Tabel 4.2.

Tabel 4.2 Proses Perhitungan Fitness

	Chromosom									
P	x1	x2	x3	x4	x5	x6	x7	x8	x9	Fitness
p1	-0.7	0.6	0.8	-0.9	0.7	-0.9	-0.9	0.5	-0.7	5.61E-14
p2	0.8	0.7	0.2	0.3	-0.5	0.3	0.1	0.4	-0.7	5.26E-15
p3	-0.9	-0.6	-0.4	-0.5	0.4	0.7	-0.6	-0.4	-0.7	4.4E-12
p4	0.6	-0.8	-0.2	0.3	-0.2	-0.7	0.2	0.5	0.1	1.15E-14
p5	-0.2	-0.8	0.5	0.7	0.4	0.5	0.6	-0.2	-0.6	8.1E-15
p6	0.5	0.5	-0.4	-0.9	0.6	0.6	-0.8	0.4	0.4	6.72E-15
p7	-0.4	-0.2	0.8	0.5	-0.6	-0.3	-0.2	-0.1	0.7	5.98E-15
p8	-0.6	0.7	0.4	0.2	0.7	0.8	0.2	0.4	0.1	2.39E-15
p9	-0.5	0.7	0.1	-0.3	0.9	0.3	0.8	-0.1	0.9	2.32E-15
p10	0.3	0.9	-0.9	0.3	0.6	-0.7	-0.2	-0.7	-0.1	3.59E-14
C1	0.8	0.7	0.2	0.3	-0.5	0.3	0.6	-0.2	-0.6	5.83E-15
C2	-0.2	-0.8	0.5	0.7	0.4	0.5	0.1	0.4	-0.7	7.15E-15
C3	-0.9	-0.6	-0.4	0.4	-0.5	0.7	-0.6	-0.4	-0.7	2.57E-12

Pada Tabel 4.2 merupakan hasil dari perhitungan fintess sebanyak individu yang ada, termasuk hasil reproduksi dari *crossover* dan *mutation*. Tahap selanjutnya adalah evaluasi dan seleksi dengan metode *elitsm*. Evaluasi akan mengurutkan setiap individu dari nilai fitness yang paling tinggi hingga yang paling rendah, selanjutnya akan dilakukan proses seleksi dengan metode *elitsm* berguna untuk mendapatkan individu dengan fitness terbaik sejumlah *popsize* dengan proses sorting dari hasil fitness. Individu yang memiliki fitness terbaik atau terpilih sejumlah *popsize* akan digunakan menjadi populasi baru. Hasil proses evaluasi akan ditunjukkan pada Tabel 4.3.

Tabel 4.3 Hasil Proses Evaluasi

	Chromosom									
P	x1	x2	x3	x4	x5	x6	x7	x8	x9	Fitness
p3	-0.9	-0.6	-0.4	-0.5	0.4	0.7	-0.6	-0.4	-0.7	4.4E-12
C3	-0.9	-0.6	-0.4	0.4	-0.5	0.7	-0.6	-0.4	-0.7	2.57E-12
p1	-0.7	0.6	0.8	-0.9	0.7	-0.9	-0.9	0.5	-0.7	5.61E-14
p10	0.3	0.9	-0.9	0.3	0.6	-0.7	-0.2	-0.7	-0.1	3.59E-14
p4	0.6	-0.8	-0.2	0.3	-0.2	-0.7	0.2	0.5	0.1	1.15E-14
p5	-0.2	-0.8	0.5	0.7	0.4	0.5	0.6	-0.2	-0.6	8.1E-15
C2	-0.2	-0.8	0.5	0.7	0.4	0.5	0.1	0.4	-0.7	7.15E-15
p6	0.5	0.5	-0.4	-0.9	0.6	0.6	-0.8	0.4	0.4	6.72E-15
p7	-0.4	-0.2	0.8	0.5	-0.6	-0.3	-0.2	-0.1	0.7	5.98E-15
C1	0.8	0.7	0.2	0.3	-0.5	0.3	0.6	-0.2	-0.6	5.83E-15
p2	0.8	0.7	0.2	0.3	-0.5	0.3	0.1	0.4	-0.7	5.26E-15
p8	-0.6	0.7	0.4	0.2	0.7	0.8	0.2	0.4	0.1	2.39E-15
p9	-0.5	0.7	0.1	-0.3	0.9	0.3	0.8	-0.1	0.9	2.32E-15

Setelah dilakukan proses evaluasi, akan terlihat individu yang sudah diurutkan dari nilai fitness yang paling tinggi hingga yang paling rendah, proses selanjutnya adalah seleksi yang akan ditunjukkan pada Tabel 4.4

Tabel 4.4 Hasil Proses Seleksi

	Chromosom										Fitness
P	P (asal)	x1	x2	x3	x4	x5	x6	x7	x8	x9	
p1	p3	-0.9	-0.6	-0.4	-0.5	0.4	0.7	-0.6	-0.4	-0.7	4.4E-12
p2	C3	-0.9	-0.6	-0.4	0.4	-0.5	0.7	-0.6	-0.4	-0.7	2.57E-12
p3	p1	-0.7	0.6	0.8	-0.9	0.7	-0.9	-0.9	0.5	-0.7	5.61E-14
p4	p10	0.3	0.9	-0.9	0.3	0.6	-0.7	-0.2	-0.7	-0.1	3.59E-14
p5	p4	0.6	-0.8	-0.2	0.3	-0.2	-0.7	0.2	0.5	0.1	1.15E-14
p6	p5	-0.2	-0.8	0.5	0.7	0.4	0.5	0.6	-0.2	-0.6	8.1E-15

p7	C2	-0.2	-0.8	0.5	0.7	0.4	0.5	0.1	0.4	-0.7	7.15E-15
p8	p6	0.5	0.5	-0.4	-0.9	0.6	0.6	-0.8	0.4	0.4	6.72E-15
p9	p7	-0.4	-0.2	0.8	0.5	-0.6	-0.3	-0.2	-0.1	0.7	5.98E-15
p10	C1	0.8	0.7	0.2	0.3	-0.5	0.3	0.6	-0.2	-0.6	5.83E-15

Tabel 4.4 merupakan hasil dari proses seleksi yang memiliki jumlah fitness paling besar, dari percobaan perhitungan manual didapatkan p3 individu dengan nilai fitness tertinggi sebesar 4.4E-12. Kemudian individu hasil dari algortima genetika dengan fitness tertinggi akan digunakan untuk bobot pada algoritme *backpropagation*.

Tahap selanjutnya pada penelitian ini adalah backpropation. Pada tahap ini proses pertama yang akan dilakukan adalah inisialisasi. Pada proses ini akan dibangkitkan bobot dan bias secara acak dengan rentang -0,1 sampai 0.9. Tetapi pada penelitian ini bobot dan bias w didapatkan dari hasil proses algoritme genetika. Pada inisialisasi ini terdapat 8 unit bobot w dan v ditambah dengan 1 unit bias yang ditampilkan pada Tabel 4.5 dan 4.6.

Tabel 4.5 Bobot dan Bias v

v	z_1	z_2	z_3	z_4	z_5	z_6	z_7	z_8
x_1	-0.9	0.1	0.4	0.6	0.8	0.6	0.8	0.5
x_2	-0.9	0.2	0.1	-0.3	0.3	-0.2	0.3	0.8
x_3	0.6	-0.9	-0.9	0.8	-0.5	-0.5	0.5	0.4
x_4	0.1	0.5	0.5	0.9	0.1	-0.1	-0.5	0.3
x_5	-0.2	0.7	-0.8	0.2	0.9	0.6	-0.9	-0.3
x_6	0.7	-0.6	0.6	0.4	0.5	0.2	0.7	0.6
x_7	-0.5	0.2	0.4	-0.8	-0.9	0.5	0.1	0.9
x_8	0.6	0.8	-0.6	-0.2	-0.8	0.2	0.6	0.3
x_9	-0.8	0.3	0.5	0.1	0.9	-0.5	-0.9	0.8
x_{10}	0.2	-0.9	0.5	0.7	0.6	0.2	0.8	0.3
1	-0.9	0.3	0.7	0.2	-0.2	0.2	0.5	0.8

Tabel 4.6 Bobot dan Bias w

w	bobot
z1	-0.9
z2	-0.6
z3	-0.4

z4	-0.5
z5	0.4
z6	0.7
z7	-0.6
z8	-0.4
bias	-0.7

Setelah tahap inisialisasi bobot dan bias w ditentukan, kemudian akan dilanjutkan pada tahap *feedforward*.

Tahap ini akan ditentukan pola – pola yang digunakan untuk proses training. Pada kasus ini untuk meramal tahun 2017A dibutuhkan data penduduk miskin dari tahun 2012A sampai 2016B (10 data) dan data pada tahun 2017A akan dijadikan target. Pada perhitungan manual akan digunakan 5 pola yaitu pola 1 sampai 5 berturut – turut untuk tahun 2015A, 2015B, 2016A, 2016B, dan 2017A yang ditunjukkan pada Tabel 4.7.

Tabel 4.7 Pola Training

	Pola 1	Pola 2	Pola 3	Pola 4	Pola 5
x1	0.27639	0.2209	0.184153	0.179375	0.151447
x2	0.2209	0.184153	0.179375	0.151447	0.131603
x3	0.184153	0.179375	0.151447	0.131603	0.112494
x4	0.179375	0.151447	0.131603	0.112494	0.130133
x5	0.151447	0.131603	0.112494	0.130133	0.120211
x6	0.131603	0.112494	0.130133	0.120211	0.1
x7	0.112494	0.130133	0.120211	0.1	0.131603
x8	0.130133	0.120211	0.1	0.131603	0.128663
x9	0.120211	0.1	0.131603	0.128663	0.110289
x10	0.1	0.131603	0.128663	0.110289	0.101102
Output	0.131603	0.128663	0.110289	0.101102	0.10147

Setelah ditentukan pola – pola untuk training, tahap selanjutnya pada proses *feedforward* adalah menghitung nilai z_in. Pada kasus ini terdapat 10 unit bobot, maka terdapat 8 buah z_in yaitu z_in_1 - z_in_8. Hasil dari perhitungan z_inj dari pola pertama dapat dilihat pada Tabel 4.8.

Tabel 4.8 Hasil perhitungan z_inj

z_in1	-1.2116
z_in2	0.3955

z_in3	0.7914
z_in4	0.6573
z_in5	0.1782
z_in6	0.3710
z_in7	0.8067
z_in8	1.5424

Setelah didapatkan nilai z_in yang ditunjukkan pada Tabel 4.8 langkah selanjutnya adalah memasukkan nilai z_inj ke dalam fungsi aktivasi untuk menghitung zj. Hasil perhitungan zj dapat dilihat pada Tabel 4.9.

Tabel 4.9 Hasil perhitungan zj

z1	0.2294
z2	0.5976
z3	0.6881
z4	0.6586
z5	0.5444
z6	0.5917
z7	0.6914
z8	0.8238

Langkah selanjutnya setelah didapatkan nilai zj adalah menghitung nilai y_inj. Pada kasus ini hanya ada 1 unit y_in karena hanya ada 1 output. Hasil perhitungan y_in yang didapatkan adalah y_in = -1.982017739.

Kemudian menghitung nilai yj. Hanya terdapat satu unit *y* karena hanya ada satu unit keluaran. Untuk menghitung nilai y, nilai y_in perlu dimasukkan kedalam fungsi aktivasi. Hasil perhitungan y yang didapatkan adalah y = 0.12110391. Setelah didapatkan nilai y, langkah selanjutnya adalah *backpropagation error*.

Pada tahap *backpropagation* langkah pertama yang dilakukan adalah mencari nilai δ_k. Nilai δ_k diperoleh dengan persamaan . Nilai y digunakan untuk output dan target dari pola pertama adalah targetnya. Nilai δ_k yang diperoleh adalah δ_k = 0.001117513.

Setelah diperoleh nilai δ_k tahap selanjutnya adalah menghitung nilai Δw_{ij}. Nilai Δw_{ij} digunakan untuk update atau pembaruan bobot dan bias w. Dalam menghitung Δw_{ij} digunakan nilai α sebesar 0.2. Maka diperoleh nilai Δw_{ij} yang ditunjukkan pada Tabel 4.10.

Tabel 4.10 Nilai Δw_{ij}

Δw11	5.13E-05

Δw12	0.000134
Δw13	0.000154
Δw14	0.000147
Δw15	0.000122
Δw16	0.000132
Δw17	0.000155
Δw18	0.000184
Δw10	0.000224

Langkah selanjutnya adalah mencari nilai δ_in_j. Nilai δ_in_j diperoleh dengan mengalikan δ_k dengan nilai w_{ij}. Hasil perhitungan δ_in_j diperlihatkan pada Tabel 4.11.

Tabel 4.11 Nilai δ_in_j

δ_in1	-0.0010058
δ_in2	-0.0006705
δ_in3	-0.000447
δ_in4	-0.0005588
δ_in5	0.00044701
δ_in6	0.00078226
δ_in7	-0.0006705
δ_in8	-0.000447

Setelah diperoleh nilai δ_in_j, langkah selanjutnya adalah mencari nilai δ_j dengan cara memasukkan nilai δ_in_j ke dalam fungsi aktivasi. Hasil perhitungan δ_j ditunjukkan pada Tabel 4.12.

Tabel 4.12 Nilai δ_j

δ1	-0.0001778
δ2	-0.00016124
δ3	-9.5929E-05
δ4	-0.00012563
δ5	0.000110869
δ6	0.000188987
δ7	-0.00014306
δ8	-6.4882E-05

Setelah didapatkan nilai δ_j, langkah selanjutnya adalah menghitung nilai Δv_{ij} dengan menggunakan persamaan. Hasil perhitungan Δv_{ij} nantinya akan digunakan untung pembaruan bobot dan bias v. Pada Tabel 4.13 akan diperlihatkan hasil perhitungan Δv_{ij}.

Tabel 4.13 Nilai Δv_{ij}

Δv	z_1	z_2	z_3	z_4	z_5	z_6	z_7	z_8
x_1	-9.83E-06	-8.913E-06	-5.3E-06	-6.944E-06	6.13E-06	1.04E-05	-7.9081E-06	-3.59E-06
x_2	-7.86E-06	-7.1236E-06	-4.24E-06	-5.55E-06	4.9E-06	8.35E-06	-6.3204E-06	-2.87E-06
x_3	-6.55E-06	-5.9385E-06	-3.53E-06	-4.627E-06	4.08E-06	6.96E-06	-5.269E-06	-2.39E-06
x_4	-6.38E-06	-5.7845E-06	-3.44E-06	-4.507E-06	3.98E-06	6.78E-06	-5.1323E-06	-2.33E-06
x_5	-5.39E-06	-4.8838E-06	-2.91E-06	-3.805E-06	3.36E-06	5.72E-06	-4.3332E-06	-1.97E-06
x_6	-4.68E-06	-4.2439E-06	-2.52E-06	-3.307E-06	2.92E-06	4.97E-06	-3.7654E-06	-1.71E-06
x_7	-4E-06	-3.6277E-06	-2.16E-06	-2.826E-06	2.49E-06	4.25E-06	-3.2187E-06	-1.46E-06
x_8	-4.63E-06	-4.1965E-06	-2.5E-06	-3.27E-06	2.89E-06	4.92E-06	-3.7234E-06	-1.69E-06
x_9	-4.27E-06	-3.8766E-06	-2.31E-06	-3.02E-06	2.67E-06	4.54E-06	-3.4395E-06	-1.56E-06
x_{10}	-3.56E-06	-3.2248E-06	-1.92E-06	-2.513E-06	2.22E-06	3.78E-06	-2.8612E-06	-1.3E-06
1	-3.56E-05	-3.2248E-05	-1.92E-05	-2.513E-05	2.22E-05	3.78E-05	-2.8612E-05	-1.3E-05

Setelah menghitung nilai Δv_{ij}, langkah selanjutnya adalah pembaruan bobot dan bias. Pembaruan pertama dilakukan pada bobot dan bias w. Perhitungan pembaruan bobot dan bias w menggunakan persamaan. Hasil bobot dan bias w baru ditunjukkan pada Tabel 4.14.

Tabel 4.14 bobot dan bias w baru

w Baru	bobot
z1	-0.899949
z2	-0.599866
z3	-0.399846
z4	-0.499853
z5	0.400122
z6	0.700132
z7	-0.599845
z8	-0.399816
bias	-0.699776

Kemudian setelah di-*update* bobot dari w, langkah selanjutnya adalah memperbarui bobot dan bias dari v. Perhitungannya dilakukan menggunakan persamaan. Hasil bobot dan bias baru dari v ditunjukkan pada Tabel 4.15.

Tabel 4.15 bobot dan bias v baru

v Baru	*z1*	*z2*	*z3*	*z4*	*z5*	*z6*	*z7*	*z8*
x1	-0.90001	0.099991087	0.399995	0.59999306	0.800006	0.60001	0.799992092	0.4999964

x2	-0.900008	0.199992876	0.099996	-0.3000056	0.300005	-0.199992	0.29999368	0.7999971
x3	0.599993	-0.90000594	-0.900004	0.79999537	-0.499996	-0.499993	0.499994731	0.3999976
x4	0.099994	0.499994216	0.499997	0.89999549	0.100004	-0.099993	-0.50000513	0.2999977
x5	-0.200005	0.699995116	-0.800003	0.19999619	0.900003	0.600006	-0.90000433	-0.300002
x6	0.699995	-0.60000424	0.599997	0.39999669	0.500003	0.200005	0.699996235	0.5999983
x7	-0.500004	0.199996372	0.399998	-0.8000028	-0.899998	0.500004	0.099996781	0.8999985
x8	0.599995	0.799995803	-0.600002	-0.2000033	-0.799997	0.200005	0.599996277	0.2999983
x9	-0.800004	0.299996123	0.499998	0.09999698	0.900003	-0.499995	-0.90000344	0.7999984
x10	0.199996	-0.90000322	0.499998	0.69999749	0.600002	0.200004	0.799997139	0.2999987
1	-0.900036	0.299967752	0.699981	0.19997487	-0.199978	0.200038	0.499971388	0.799987

Pada perhitungan manual, tahapan *feedforward* sampai update bobot dan bias akan diulangi sebanyak 5 kali. Kemudian hasil perhitungan bobot dan bias digunakan pada proses prediksi. Pada Tabel 4.16 ditunjukkan nilai bobot dan bias w setelah 5 iterasi.

Tabel 4.16 Bobot dan bias w

w Baru	bobot
z1	-0.900839
z2	-0.60192
z3	-0.402233
z4	-0.501973
z5	0.398358
z6	0.698142
z7	-0.602142
z8	-0.402566
bias	-0.703246

Kemudian pada Tabel 4.17 akan ditunjukkan nilai hasil perhitungan bobot dan bias v setelah 5 iterasi.

Tabel 4.17 Bobot dan bias v

v Baru	*z1*	*z2*	*z3*	*z4*	*z5*	*z6*	*z7*	*z8*
x1	-0.899933	0.100055059	0.400033	0.60004643	0.799962	0.599935	0.800052254	0.5000264
x2	-0.899935	0.200054058	0.100032	-0.2999548	0.299963	-0.200064	0.300050893	0.8000255
x3	0.600054	-0.89995551	-0.899974	0.8000373	-0.500031	-0.500053	0.500042022	0.4000211
x4	0.100055	0.500045154	0.500027	0.90003786	0.099969	-0.100053	-0.49995735	0.3000214
x5	-0.199938	0.700051529	-0.799969	0.20004281	0.899964	0.599939	-0.89995173	-0.299976
x6	0.700061	-0.59994873	0.60003	0.40004233	0.499965	0.19994	0.700047709	0.6000235

x7	-0.499936	0.200054032	0.400032	-0.7999553	-0.900037	0.499936	0.10005043	0.9000248
x8	0.600069	0.800057498	-0.599966	-0.1999524	-0.80004	0.199932	0.600053653	0.3000265
x9	-0.799928	0.300060697	0.500036	0.10004988	0.899958	-0.500071	-0.89994376	0.8000276
x10	0.200061	-0.89994873	0.50003	0.70004226	0.599965	0.19994	0.800047688	0.3000234
1	-0.899439	0.300471056	0.700279	0.20038875	-0.200324	0.199445	0.500438505	0.8002159

Bobot dan bias pada Tabel 4.16 dan 4.17 digunakan untuk proses prediksi dengan cara melakukan *feedforward*. Hasil perhitungan *feedforward* dihasilkan nilai y yang merupakan nilai prediksi, yang ditunjukkan pada Tabel 4.18.

Tabel 4.18 nilai y

Tahun	Nilai Y
2015A	0.11994001
2015B	0.11974472
2016A	0.12053125
2016B	0.12086966
2017A	0.12053628

Setelah nilai y didapatkan, langkah selanjutnya adalah denormalisasi untuk mengetahui nilai asli dari y. Pada Tabel 4.19 akan ditunjukkan nilai hasil prediksi perhitungan manual dibandingkan dengan data asli.

Tabel 4.19 Hasil prediksi

Tahun	Data Asli	Hasil Prediksi
2015A	28590000	28397623.43
2015B	28510000	28382729.55
2016A	28010000	28404602.08
2016B	27760000	27043403.17
2017A	27770000	28407871.94

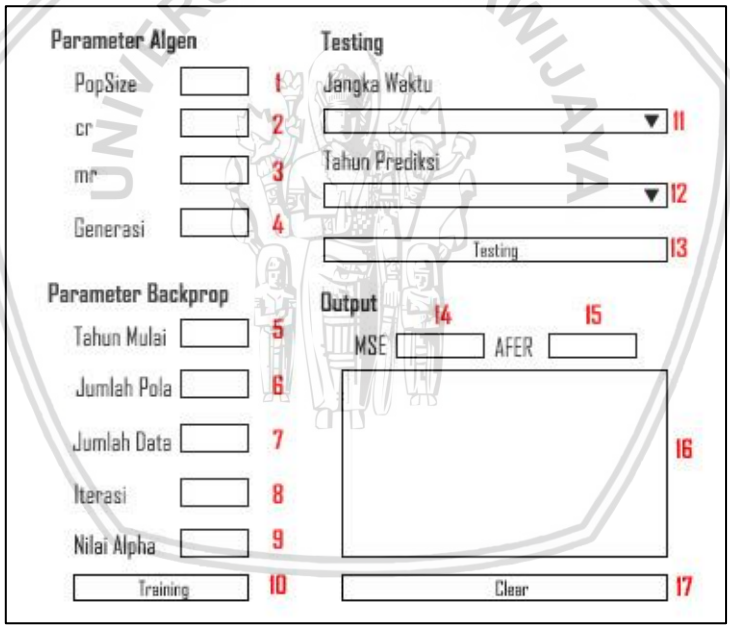
Dari hasil prediksi dan data asli akan dicari nilai MSE untuk mengetahui nilai error dari selisih data asli dan data prediksi. Hasil perhitungan MSE akan ditunjukkan pada Tabel 4.20.

Tabel 4.20 Perhitungan MSE

Tahun	Data Asli	Hasil Prediksi	Selisih Akar Kuadrat
2015A	28590000	28397623.43	0.000136028
2015B	28510000	28382729.55	7.95411E-05
2016A	28010000	28404602.08	0.000104896
2016B	27760000	27043403.17	0.000390743
2017A	27770000	28407871.94	0.000363526
MSE (Total/n)			0.000214947

4.1.3 Perancangan Antar Muka

Pada Gambar 4.17 akan dijelaskan perancangan antar muka yang digunakan pada penelitian ini.



Gambar 4.17 Rancangan Antar Muka

Pada perancangan antar muka yang ditunjukkan pada gambar 4.17 terdapat 4 bagian, yaitu Training algoritme genetika dan algoritme *backpropagation*, testing, dan kemudian output. Dari masing – masing bagian memiliki fungsi tertentu, yaitu:

1. *Text field* untuk menentukan jumlah *Popsize* pada proses algoritme genetika.
2. *Text field* untuk menentukan nilai Cr pada proses algoritme genetika.
3. *Text field* untuk menentukan nilai Mr pada proses algoritme genetika.
4. *Text field* untuk menentukan jumlah generasi pada proses algoritme genetika.

5. *Text field* untuk menentukan tahun mulai pada proses *backpropagation*.
6. *Text field* untuk menentukan jumlah pola pada proses *backpropagation*.
7. *Text field* untuk menentukan jumlah data pada proses *backpropagation*.
8. *Text field* untuk menentukan jumlah iterasi pada proses *backpropagation*.
9. *Text field* untuk menentukan nilai alpha pada proses *backpropagation*
10. *Button* untuk melakukan proses training.
11. *Combo box* untuk menentukan jangka waktu yang diprediksi pada proses testing.
12. *Combo box* untuk menentukan tahun yang akan diprediksi pada proses testing.
13. *Button* untuk melakukan proses testing.
14. *Text field* untuk menampilkan nilai MSE dari proses training.
15. *Text field* untuk menampilkan nilai AFER dari proses training.
16. *Text field* untuk menampilkan hasil prediksi.
17. *Button* untuk menghapus field output.

4.2 Perancangan Uji Coba dan Evaluasi

Untuk mengetahui kualitas dari aplikasi prediksi menggunakan jaringan syaraf tiruan *backpropagation* yang pelatihan bobotnya dioptimasi menggunakan algoritme genetika perlu dilakukan suatu pengujian pada sistem. Pada kasus ini akan dilakukan pengujian pada beberapa parameter guna mengetahui parameter untuk menghasilkan hasil prediksi paling optimal. Beberapa parameter yang akan diuji yaitu:

1. Uji coba ukuran populasi yang merupakan pengujian untuk mencari jumlah populasi (*Popsize*) yang paling optimal.
2. Uji coba generasi atau iterasi yang merupakan uji coba guna menentukan seberapa banyak generasi atau iterasi yang paling optimal.
3. Uji coba kombinasi antara *crossover*rate (Cr) dan *mutation*rate (Mr) untuk ditentukan kombinasi Cr dan Mr guna menghasilkan hasil yang paling optimal.
4. Uji coba nilai alpha terhadap jumlah iterasi untuk mengetahui nilai alpha dan jumlah iterasi yang menghasilkan hasil yang optimal.

4.2.1 Data Uji

Data uji yang digunakan merupakan data jumlah penduduk miskin di Indonesia dari tahun 1998 sampai tahun 2017A dengan data sebanyak 26 data.

4.2.2 Uji Coba Populasi

Pengujian ini bertujuan mencari besaran populasi(*Popsize*) teroptimum dalam mencari solusi. Pengujian populasi berpengaruh pada tingkat kebaikan suatu solusi, karena semakin besar ukuran populasi maka

daerah yang dicari akan semakin luas sehingga solusi yang dicari lebih beragam atau lebih bervariasi. Pada penelitian ini beberapa ukuran populasi yang akan diuji yaitu kelipatan 50 dari 50 hingga 700. Setiap parameter akan diuji sebanyak 10 kali. Rancangan Tabel hasil uji coba akan ditunjukkan pada Tabel 4.21.

Tabel 4.21 Rancangan Uji Coba Populasi

Uji Coba	Populasi	Fitness Terbaik
1	50	
2	100	
3	150	
4	200	
5	250	
6	300	
7	350	
8	400	
9	450	
10	500	
11	550	
12	600	
13	650	
14	700	

4.2.3 Uji Coba Generasi

Uji coba selanjutnya merupakan uji coba generasi. Pengujian ini bertujuan guna mencari jumlah iterasi yang dibutuhkan untuk menghasilkan solusi yang optimal. Hipotesa pada pengujian ini adalah jika ukuran generasi semakin besar, maka solusi yang diperoleh juga akan semakin besar. Tetapi semakin besar ukuran generasi yang ditentukan akan berpengaruh terhadap waktu komputasi dan juga tidak ada jaminan hasil yang diperoleh akan lebih baik. pengujian ini bertujuan untuk mengetahui nilai fitness yang dihasilkan terhadap ukuran generasi. Penelitian ini akan menguji ukuran generasi dari 1 generasi hingga 10 generasi. Setiap parameter generasi akan diuji sebanyak 10 kali. Rancangan Tabel hasil uji coba generasi akan ditunjukkan pada Tabel 4.22.

Tabel 4.22 Rancangan Hasil Uji Coba Generasi

Uji Coba	Generasi	Fitness Terbaik
1	1	

2	2	
3	3	
4	4	
5	5	
6	6	
7	7	
8	8	
9	9	
10	10	

4.2.4 Uji Coba Kombinasi *Crossover* rate(Cr) dan *Mutation* rate(Mr)

Pengujian ini bertujuan guna mendapatkan kombinasi yang optimal. *Cr* dan *Mr* sangat berpengaruh terhadap hasil yang diperoleh, karena keberagaman solusi salah satunya dihasilkan melalui parameter dari *crossover* dan *mutation*. Pengujian ini akan dilakukan dari Cr sebesar 0 dan Mr 1, yang kemudian nilai Cr akan naik sebesar 0.1 dan dan nilai Mr akan turun sebesar 0.1. Kombinasi tersebut akan berhenti sampai nilai Cr menjadi 1 dan nilai Mr menjadi 0. Setiap parameter akan diuji sebanyak 10 kali. Rancangan Tabel pengujian kombinasi Cr dan Mr akan ditunjukkan pada Tabel 4.23.

Tabel 4.23 Rancangan Uji Coba Kombinasi Cr dan Mr

Uji Coba	Cr	Mr	Fitness Terbaik
1	0.1	0.9	
2	0.2	0.8	
3	0.3	0.7	
4	0.4	0.6	
5	0.5	0.5	
6	0.6	0.4	
7	0.7	0.3	
8	0.8	0.2	
9	0.9	0.1	

4.2.5 Uji Coba Jumlah Pola Terhadap Nilai AFER

Uji coba ini memiliki tujuan untuk mengetahui nilai AFER yang dihasilkan dari banyaknya jumlah pola yang ditraining pada algoritme

backpropagation. Jumlah pola akan diuji dari 1 sampai 10 pola sebanyak 10 kali. Pada Tabel 4.24 akan ditunjukkan rancangan Tabel uji coba jumlah pola terhadap nilai AFER.

Tabel 4.24 Rancangan Uji Coba Jumlah Pola Terhadap Nilai AFER.

Jumlah Pola	Tahun Awal	Tahun Prediksi	Rata - rata
1	2012B	2017A	
2	2012A	2017A	
3	2011B	2017A	
4	2011A	2017A	
5	2010	2017A	
6	2009	2017A	
7	2008	2017A	
8	2007	2017A	
9	2006	2017A	
10	2005	2017A	

4.2.6 Uji Coba Jumlah Iterasi Terhadap Nilai AFER

Uji coba selanjutnya merupakan pengujian pada tahap *backpropagation* yaitu jumlah iterasi terhadap nilai AFER. Pada pengujian ini akan diuji beberapa iterasi. Jumlah iterasi yang diuji yaitu kelipatan 100 dari 100 sampai 1000 iterasi. Tujuan dari pengujian ini adalah mencari jumlah iterasi untuk menghasilkan nilai AFER tertinggi. Rancangan Tabel uji coba jumlah iterasi terhadap nilai AFER ditunjukkan pada Tabel 4.25.

Tabel 4.25 Rancangan Uji Coba Nilai Alpha Terhadap Jumlah Iterasi.

Uji Coba	Jumlah Iterasi	Rata - Rata
1	100	
2	200	
3	300	
4	400	
5	500	
6	600	
7	700	
8	800	
9	900	
10	1000	

4.2.7 Uji Coba Nilai Alpha (α)Terhadap Nilai AFER

Uji coba selanjutnya merupakan pengujian pada tahap *backpropagation* yaitu besaran angka *alpha* (α) terhadap nilai AFER untuk memperoleh nilai alpha yang paling optimal terhadap nilai AFER. Nilai alpha yang akan diuji yaitu dari 0.1 sampai 0.9. Rancangan Tabel uji coba nilai alpha terhadap nilai AFER ditunjukkan pada Tabel 4.26.

Tabel 4.26 Rancangan Uji Coba Nilai *Alpha* (α) Terhadap Nilai AFER

Uji Coba	Nilai Alpha	Rata - Rata
1	0.1	
2	0.2	
3	0.3	
4	0.4	
5	0.5	
6	0.6	
7	0.7	
8	0.8	
9	0.9	

4.2.8 Uji Coba Hasil Prediksi

Uji coba terakhir merupakan uji coba hasil prediksi. Pengujian ini menggunakan *backpropagation* dimana bobot pelatihannya dioptimasi dengan algoritme genetika untuk memprediksi jumlah penduduk miskin untuk jangka waktu 10 tahun. Pengujian akan membandingkan hasil prediksi dengan data asli dari tahun 2012B sampai 2017A. Racangan Tabel uji coba hasil prediksi akan ditunjukkan pada Tabel 4.27.

Tabel 4.27 Racangan Uji Coba Hasil Prediksi.

Tahun	Data Aktual	Hasil Prediksi	Selisih	*|(Ai - Fi) / Fi|*
2012B	28590000			
2013A	28070000			
2013B	28550000			
2014A	28280000			
2014B	27730000			
2015A	28590000			
2015B	28510000			
2016A	28010000			
2016B	27760000			
2017A	27770000			
AFER (%)				

4.3 Implementasi Lingkungan

Pada sub bab implementasi lingkungan akan dijelaskan seluruh spesifikasi dari perangkat keras dan perangkat lunak yang dipakai untuk penelitian ini.

4.3.1 Implementasi Lingkungan Perangkat Keras

Pada penelitian ini menggunakan perangkat keras berupa laptop Acer dengan seri Aspire 4750 dengan spesifikasi berikut :

- Prosesor: Intel Core i3 2310M (2.1 GHz, cache 3MB)
- VGA: Intel HD Graphics 3000
- Memori (RAM): 4 GB DDR3
- Ukuran Layar : 14”
- Resolusi Layar: 1366 x 768 piksel
- Harddisk: 500 GB

4.3.2 Implementasi Lingkungan Perangkat Lunak

Kemudian pada penelitian ini menggunakan perangkat lunak berupa :

- Sistem Operasi : Windows 10 Pro 1703
- Tipe Sistem : 64-bit
- Bahasa pemrograman : Java

4.4 Implementasi Algoritme

Pada penelitian ini sistem prediksi penduduk miskin di Indonesia mempunyai beberapa tahapan, yaitu pengambilan data menggunakan file csv, kemudian pada tahap optimasi algoritme genetika terdapat proses inisialisasi individu, reproduksi, evaluasi dan seleksi. selanjutnya pada tahap *backpropagation* terdapat proses normalisasi data, proses inisialisasi bobot secara random, proses *feedforward*, proses error pada *backpropagation*, proses perbaikan bobot dan bias pada hidden unit dan input unit.

4.4.1 Implementasi Algoritme Pengambilan Data

Pada penelitian ini menggunakan microsoft excel dengan data jumlah penduduk miskin di Indonesia dari tahun 1998 sampai 2017 menggunakan ‘.csv’ sebagai ekstensinya. Implementasi algoritme pengambilan data akan ditunjukkan pada Source Code 4.1.

```
1  public class Data {
```

```
2     String direktory = "D:/Kuliah/FILKOM/Skripsi/Aplikasi/Skripsi/data.csv";
3     BufferedReader br = null;
4     String line = "";
5     String splitBy = ",";
6     String[] data;
7     int countData = 0;
8
9     double[] jumlah;
10    String[] tahun;
11
12    public void getData(){
13        this.tahun = new String [26];
14        this.jumlah = new double [26];
15        try {
16            br = new BufferedReader(new FileReader(this.direktory));
17            while ((line = br.readLine()) != null) {
18                data = line.split(this.splitBy);
19                this.tahun[countData] = data[0];
20                this.jumlah[countData] = Double.parseDouble(data[1]);
21                this.countData++;
22            }
23        } catch(FileNotFoundException e){
24            e.printStackTrace();
25        } catch(IOException e){
26            e.printStackTrace();
27        } finally {
28            if (br != null) {
29                try {
30                    br.close();
31                } catch(IOException e){
32                    e.printStackTrace();
33                }
34            }
```

```
35          }
36      }
```

Source Code 4.1 Pengamambilan Data

4.4.2 Implementasi Algoritme Normalisasi Data

Pada penelitian ini normalisasi data berguna untuk memudahkan proses perhitungan dengan cara mengubah data dalam interval (0,1 – 0,9) yang ditunjukkan pada line 1 - 9. Seluruh implementasi algoritme untuk normalisasi data akan ditunjukkan pada Source Code 4.2.

```
1  public void normalisasi(double min, double max){
2          this.normJumlah = new double[this.jumlah.length];
3          this.max = max;
4          this.min = min;
5
6          for (int i = 0; i < this.normJumlah.length; i++) {
7              this.normJumlah[i] = Double.parseDouble(df4.format((this.max * (this.jumlah[i] - this.getDataMin()) / (this.getDataMax()-this.getDataMin())) + this.min));
8          }
9      }
```

Source Code 4.2 Normalisasi Data

4.4.3 Implementasi Algoritme Inisialisasi pada Algoritme *Backpropagation*

Pada penelitian ini tujuan dari inisialisasi pada *backpropagation* yaitu untuk membangkitkan bobot dan bias pada *v* dan *w* dengan batas (range) -0,1 sampai 0,9 secara acak, yang akan ditunjukkan pada line 1 – 33 Source Code 4.3.

```
1      public void setBobotV(int jumlah, int panjangData){
2          this.bobotV = new double[jumlah][panjangData + 1];
3
```

```
4           double max = 9;
5           double min = 1;
6           double range = ((max + 1) - min) / 10;
7
8           for (int i = 0; i < jumlah; i++) {
9               for (int j = 0; j < panjangData + 1; j++)
{
10                  this.bobotV[i][j]                    =
Double.parseDouble(df3.format(random.nextDouble()    *
range));
11                  System.out.print(this.bobotV[i][j]+"
| ");
12              }
13              System.out.println("");
14          }
15
16      }
17      public double[][] getBobotV(){
18          return this.bobotV;
19      }
20      //untuk inisialisasi bobot w
21      public void setBobotW(int jumlah){
22          this.bobotW = new double[jumlah + 1];
23
24          double max = 9;
25          double min = 1;
26          double range = ((max + 1) - min) / 10;
27
28          for (int i = 0; i < this.bobotW.length; i++) {
29              this.bobotW[i]                           =
Double.parseDouble(df3.format(random.nextDouble()    *
range));
30              System.out.println(this.bobotW[i]);
31          }
32
33      }
34      public double[] getBobotW(){
```

```
35          return this.bobotW;
36      }
```

Source Code 4.3 Inisialisasi Tahap *Backpropagation*

4.4.4 Implementasi Algoritme *Feedforward* pada Algoritme *Backpropagation*

Terdapat beberapa proses pada tahap *feedforward*, yaitu melakukan perhitungan *z_inj*, *zj*, *y_in* dan *y* yang sebelumnya telah dijelaskan pada bab manualisasi. Nilai *z_inj* dihasilkan dari perkalian dan penjumlahan data normalisasi dengan bobot v yang ditunjukkan pada line 1 - 37. Kemudian memasukkan *z_inj* ke dalam fungsi aktivasi untuk mendapatkan nilai *zj* yang ditunjukkan pada line 39 - 49. Kemudian nilai *y_in* dihasilkan dari perkalian dan penjumlahan bobot *wj* dan nilai *zj* yang ditunjukkan pada line 52 - 62. Kemudian memasukkan nilai *y_in* ke dalam fungsi aktivasi untuk mendapatkan nilai *y* yang ditunjukkan pada line 64 - 70. Seluruh proses tersebut akan dijelaskan pada Source Code 4.4.

```
1     public void hitungZ_in(int jumlah, int polaKe, int
panjangData){
2           this.z_in = new double[jumlah];
3
4           //inisialisasi z_in = 0
5           for (int i = 0; i < this.z_in.length; i++) {
6               this.z_in[i] = 0;
7           }
8
9           //proses hitung z_in
10          for (int i = 0; i < this.z_in.length; i++) {
11              for (int j = 0; j < panjangData; j++) {
12                  this.z_in[i]                                =
Double.parseDouble(df5.format(this.z_in[i]                      +
(this.pola[polaKe - 1][j] * this.bobotV[i][j])));
13              }
14              this.z_in[i]                                    =
Double.parseDouble(df5.format(this.z_in[i]                      +
this.bobotV[i][panjangData]));
```

```
15              System.out.println(this.z_in[i]);
16          }
17      }
18      //hitung nilai z_in
19      public void hitungZ_in_test(int jumlah, int polaKe,
int panjangData){
20          this.z_in = new double[jumlah];
21          //inisialisasi z_in = 0
22          for (int i = 0; i < this.z_in.length; i++) {
23              this.z_in[i] = 0;
24          }
25          //proses hitung z_in
26          for (int i = 0; i < this.z_in.length; i++) {
27              for (int j = 0; j < panjangData; j++) {
28                  this.z_in[i]                                =
Double.parseDouble(df5.format(this.z_in[i]                      +
(this.polaTest[polaKe - 1][j] * this.bobotV[i][j])));
29              }
30              this.z_in[i]                                    =
Double.parseDouble(df5.format(this.z_in[i]                      +
this.bobotV[i][panjangData]));
31              System.out.println("z_in"+(i+1)+"               =
"+this.z_in[i]);
32          }
33      }
34
35      public double[] getZ_in(){
36          return this.z_in;
37      }
38      //hitung nilai z
39      public void hitungZ(){
40          this.z = new double[this.z_in.length];
41          for (int i = 0; i < this.z.length; i++) {
42              this.z[i]  =  Double.parseDouble(df5.format(1
/ ( 1 + (Math.pow(this.e, -(this.z_in[i])))))));
43              System.out.println(this.z[i]);
44          }
45      }
```

```
46
47      public double[] getZ(){
48          return this.z;
49      }
50
51      //hitung nilai y in
52      public void hitungY_in(){
53          this.y_in = 0;
54          for (int i = 0; i < this.bobotW.length - 1; i++)
{
55              this.y_in                                    =
Double.parseDouble(df5.format(this.y_in  +  (this.z[i]  *
this.bobotW[i])));
56          }
57          this.y_in                                    =
Double.parseDouble(df5.format(this.y_in                  +
this.bobotW[this.bobotW.length-1]));
58      }
59
60      public double getY_in(){
61          return this.y_in;
62      }
63      //hitung nilai y
64      public void hitungY(){
65          this.y  =  Double.parseDouble(df4.format(1  /  (1
+(Math.pow(e, -(this.y_in))))));
66      }
67
68      public double getY(){
69          return this.y;
70      }
71
72
73          System.out.println("Bobot W :");
74          for (int i = 0; i < this.bobotW.length; i++) {
75              System.out.print(this.bobotW[i]+" | ");
76          }
```

```
77
78          System.out.println("z_in");
79          this.hitungZ_in(jumlah, polaKe, panjangData);
80          System.out.println("\n z");
81          this.hitungZ();
82          this.hitungY_in();
83          this.hitungY();
84          System.out.println("");
85          System.out.println("y_in = "+this.getY_in());
86          System.out.println("y = "+this.getY());
87      }
```

Source Code 4.4 *Feedforward*

4.4.5 Implementasi Algoritme *Backpropagation* Error

Terdapat beberapa proses pada tahap *backpropagation* error yaitu melakukan perhitungan nilai *δk, Δwjk, δ_inj, δj, Δvij*, dan *Δv0j*. Pada Source Code 4.5 baris ke 1 – 7 ditunjukkan proses perhitungan nilai *δk*. Pada baris ke 9 – 24 ditunjukkan proses perhitungan nilai *Δwjk*. Pada baris ke 26 – 37 ditunjukkan proses perhitungan nilai *δ_inj*. Pada baris ke 39 – 46 ditunjukkan proses perhitungan nilai *Δvij*. Kemudian terakhir baris ke 48 – 64 ditunjukkan proses perhitungan nilai *Δv0j*.

```
1       public void hitungDeltaK(int outputKe){
2           this.delta_k                                        =
    Double.parseDouble(df5.format((this.output[outputKe   -
    1] - this.y) * (this.y * (1 - this.y))));
3       }
4
5       public double getDeltaK(){
6           return this.delta_k;
7       }
8       //hitung delta w
9       public void hitungDelta_W(int jumlah){
10          this.delta_w = new double[jumlah + 1];
11
```

```
12          for (int i = 0; i < this.delta_w.length - 1; i++) {
13              this.delta_w[i] = Double.parseDouble(df5.format(this.delta_k * this.z[i] * this.alpha));
14          }
15          this.delta_w[this.delta_w.length - 1] = Double.parseDouble(df5.format(this.delta_k * this.alpha));
16
17          for (int i = 0; i < this.delta_w.length; i++) {
18              System.out.println(this.delta_w[i]);
19          }
20      }
21
22      public double[] getDelta_W(){
23          return this.delta_w;
24      }
25      //hitung delta in
26      public void hitungDelta_In(int jumlah){
27          this.delta_in = new double[jumlah];
28
29          for (int i = 0; i < this.delta_in.length; i++) {
30              this.delta_in[i] = Double.parseDouble(df5.format(this.delta_k * this.bobotW[i]));
31              System.out.println(this.delta_in[i]);
32          }
33      }
34
35      public double[] getDelta_In(){
36          return this.delta_in;
37      }
38      //hitung delta
39      public void hitungDelta(int jumlah){
40          this.delta = new double[jumlah];
41
```

```
42          for (int i = 0; i < this.delta.length; i++) {
43              this.delta[i]                                   =
Double.parseDouble(df6.format(this.delta_in[i]          *
this.z[i] * (1 - this.z[i])));
44              System.out.println(this.delta[i]);
45          }
46      }
47      //hitung delta v
48      public    void    hitungDelta_V(int    jumlah,    int
panjangData, int polaKe){
49          this.delta_v = new double[jumlah][panjangData +
1];
50
51          for (int i = 0; i < jumlah; i++) {
52              for (int j = 0; j < panjangData; j++) {
53                  this.delta_v[i][j]                      =
Double.parseDouble(df7.format(this.alpha                *
this.delta[i] * this.pola[polaKe - 1][j]));
54                  System.out.print(this.delta_v[i][j]+"
| ");
55              }
56              this.delta_v[i][panjangData]                =
Double.parseDouble(df7.format(this.alpha                *
this.delta[i]));
57
System.out.print(this.delta_v[i][panjangData]+" | ");
58              System.out.println("");
59          }
60      }
61
62      public double[][] getDelta_V(){
63          return this.delta_v;
64      }
65
66
67          this.hitungDeltaK(polaKe);
68          System.out.println("delta_k                     =
"+this.getDeltaK());
69          System.out.println("\n delta_w");
```

```
70          this.hitungDelta_W(jumlah);
71          System.out.println("\n delta_in");
72          this.hitungDelta_In(jumlah);
73          System.out.println("\n delta");
74          this.hitungDelta(jumlah);
75          System.out.println("\n delta_v");
76          this.hitungDelta_V(jumlah,        panjangData,
polaKe);
77      }
```

Source Code 4.5 *Backpropagation* Error

4.4.6 Implementasi Algoritme Update pada Algoritme *Backpropagation*

Terdapat 2 proses pada tahap update, ditunjukkan pada Source Code 4.6 merupakan update bobot dan bias *v* dan *w*. Pada Source Code 4.7 merupakan proses update bobot dan bias *w*. Dan pada Source Code 4.8 merupakan proses update bobot dan bias *v*.

```
1  public void update(int panjangData){
2          System.out.println("\n Update bobot W");
3          this.updateW();
4          System.out.println("\n Update bobot V");
5          this.updateV(panjangData);
6      }
```

Source Code 4.6 Update Bobot dan Bias *v* dan *w*

```
1   public void updateV(int panjangData){
2          for (int i = 0; i < this.bobotV.length; i++) {
3              for (int j = 0; j < panjangData + 1; j++) {
4                  this.bobotV[i][j]                    =
Double.parseDouble(df6.format(this.bobotV[i][j]       +
this.delta_v[i][j]));
5                  System.out.print(this.bobotV[i][j]+"  |
");
6              }
7              System.out.println("");
```

```
8           }
9       }
```

Source Code 4.7 Perhitungan Update Bobot dan bias *v*

```
1     public void updateW(){
2           for (int i = 0; i < this.bobotW.length; i++) {
3               this.bobotW[i] =
Double.parseDouble(df6.format(this.bobotW[i] +
this.delta_w[i]));
4               System.out.println(this.bobotW[i]);
5           }
6       }
```

Source Code 4.8 Perhitungan Update Bobot dan bias *w*

4.4.7 Implementasi Algoritme Inisialisasi pada Algoritme Genetika

Tahap inisialisasi individu akan memilih nilai secara acak pada batas tertentu sejumlah *popsize* yang sebelumnya ditentukan ditunjukkan pada baris 1 - 12. Parameter jumlah *chromosom* dan nilai *popsize* digunakan pada proses ini yang akan ditunjukkan pada Source 4.9.

```
1   public void inisialisasi(int popsize, int stringLen){
2           this.popsize = popsize;
3           this.stringLen = stringLen;
4           this.individu = new
double[this.popsize][this.stringLen];
5
6           double max = 9;
7           double min = 1;
8           double range = ((max + 1) - min) / 10;
9
10          for (int i = 0; i < popsize; i++) {
11              for (int j = 0; j < stringLen; j++) {
12                  this.individu[i][j] =
Double.parseDouble(df3.format(random.nextDouble() *
range));
```

```
13
   System.out.print(this.individu[i][j]+" | ");
14              }
15              System.out.println("");
16          }
17      }
```

Source Code 4.9 Inisialisasi Algoritme Genetika

4.4.8 Implementasi Algoritme Reproduksi pada Algoritme Genetika

Tahap reproduksi memiliki 2 proses, yaitu *crossover* dan mutasi. Pada tahap *crossover* akan dipilih dua individu secara acak untuk dijadikan induk yang ditunjukkan pada baris 1 - 51. Kemudian pada tahap mutasi akan dipilih satu individu secara acak yang ditunjukkan ada baris 59 - 114. Pada tahap ini parameter yang digunakan yaitu nilai Cr dan Mr yang kemudian akan dikalikan dengan nilai *popsize*, dan jumlah kromosom. Proses reproduksi ditunjukkan pada Source 4.10.

```
1   public void reproduksi(double cr, double mr){
2           int offspringC = (int)Math.round(this.popsize
    * cr);
3           int offspringM = (int)Math.round(this.popsize
    * mr);
4
5           System.out.println("Jumlah Child Crossover =
    "+offspringC);
6           System.out.println("Jumlah Child Mutasi =
    "+offspringM);
7           System.out.println("");
8
9           int max = this.popsize;
10          int min = 1;
11          int range = (max + 1) - min;
12          int randomParent;
13
14          //inisialisasi child
15          this.childC                =                new
    double[offspringC][this.stringLen];
```

```
16          //double[][]     tempOffspringC     =     new
double[offspringC][this.stringLen];
17
18          this.childM                =                new
double[offspringM][this.stringLen];
19          double[][]     tempOffspringM     =     new
double[offspringM][this.stringLen];
20
21          System.out.println("CROSS OVER");
22          for (int i = 0; i < offspringC; i++) {
23              randomParent  =  random.nextInt(range)  +
min;
24              System.out.println("Induk             dari
P"+randomParent);
25              for (int j = 0; j < this.stringLen; j++) {
26                  this.childC[i][j]                    =
this.individu[randomParent-1][j];
27                  System.out.print(this.childC[i][j]+"
| ");
28              }
29              System.out.println("");
30          }
31          System.out.println("");
32
33          int maxC = this.stringLen;
34          int minC = 1;
35          int rangeC = (maxC + 1) - minC;
36          int cutPoint;
37
38          //proses cross over
39          for (int i = 0; i < offspringC; i++) {
40              cutPoint = random.nextInt(rangeC) + min;
41              randomParent  =  random.nextInt(range)  +
min;
42              System.out.println("Cut      point      =
"+cutPoint);
43              System.out.println("Random    parent    =
"+randomParent);
```

```
44              for (int j = cutPoint-1; j < this.stringLen; j++) {
45                  this.childC[i][j] = this.individu[randomParent-1][j];
46              }
47          }
48
49          for (int i = 0; i < offspringC; i++) {
50              for (int j = 0; j < this.stringLen; j++) {
51                  System.out.print(this.childC[i][j]+" | ");
52              }
53              System.out.println("");
54          }
55
56          System.out.println("");
57          System.out.println("MUTASI");
58          for (int i = 0; i < offspringM; i++) {
59              randomParent = random.nextInt(range) + min;
60              System.out.println("Induk dari P"+randomParent);
61              for (int j = 0; j < this.stringLen; j++) {
62                  tempOffspringM[i][j] = this.individu[randomParent-1][j];
63
System.out.print(tempOffspringM[i][j]+" | ");
64              }
65              System.out.println("");
66          }
67
68          System.out.println("");
69
70          int exchangePoint1;
71          int exchangePoint2;
72          int maxM = this.stringLen;
73          int minM = 1;
74          int rangeM = (maxM + 1) - minM;
```

```
75
76          //proses mutasi (menukar posisi kromosom)
77          for (int i = 0; i < offspringM; i++) {
78              exchangePoint1 = random.nextInt(rangeM) +
minM;
79              exchangePoint2 = random.nextInt(rangeM) +
minM;
80              if (exchangePoint2 == exchangePoint1) {
81                  exchangePoint2                          =
random.nextInt(rangeM) + minM;
82              }
83              System.out.println("Child "+(i+1));
84              System.out.println("Exchange  point  1  =
"+exchangePoint1);
85              System.out.println("Exchange  point  2  =
"+exchangePoint2);
86              for (int j = 0; j < this.stringLen; j++) {
87                  if (j == (exchangePoint1-1)) {
88                      this.childM[i][j]                   =
tempOffspringM[i][exchangePoint2-1];
89                  } else if (j == (exchangePoint2-1)) {
90                      this.childM[i][j]                   =
tempOffspringM[i][exchangePoint1-1];
91                  } else{
92                      this.childM[i][j]                   =
tempOffspringM[i][j];
93                  }
94                  System.out.print(this.childM[i][j]+"
| ");
95              }
96              System.out.println("");
97          }
98
99          int popsizeBaru = this.popsize + offspringC +
offspringM;
100         this.himpunanIndividu          =          new
double[popsizeBaru][this.stringLen];
101         System.out.println("");
102
```

```
103         System.out.println("HIMPUNAN INDIVIDU");
104
105         //himpunan individu (gabungan induk + offspring)
106         for (int i = 0; i < popsizeBaru; i++) {
107             for (int j = 0; j < this.stringLen; j++) {
108                 if (i < this.popsize) {
109                     this.himpunanIndividu[i][j] = this.individu[i][j];
110                 } else if (i < (this.popsize + offspringC)) {
111                     this.himpunanIndividu[i][j] = this.childC[i - this.popsize][j];
112                 } else {
113                     this.himpunanIndividu[i][j] = this.childM[i - (this.popsize + offspringC)][j];
114                 }
115
System.out.print(this.himpunanIndividu[i][j]+" | ");
116             }
117             System.out.println("");
118         }
119
120     }
```

Source Code 4.10 Reproduksi

4.4.9 Implementasi Algoritme Evaluasi pada Algoritme Genetika

Pada tahap evaluasi akan dihitung fitness dari seluruh individu termasuk *offspring*, yang akan ditunjukkan pada baris 7 – 33 Source Code 4.11.

```
1   public void evaluasi(double min, double max, int jumlah, int mulai, int banyakPola, int panjangData, double alpha, int polaKe){
2           double MSE;
3
4           this.fitness = new double[this.himpunanIndividu.length];
```

```
5           this.inisialisasi(min, max, jumlah, mulai,
banyakPola, panjangData, alpha);
6
7           for (int i = 0; i < this.himpunanIndividu.length; i++) {
8               for (int j = 0; j < this.stringLen; j++) {
9                   this.bobotW[j] = this.himpunanIndividu[i][j];
10                  //System.out.print(this.bobotW[j]+"   | ");
11              }
12
13              this.MSE = 0;
14              for (int j = 1; j <= banyakPola; j++) {
15                  System.out.println("");
16                  this.feedforward(jumlah, panjangData,
j);
17                  this.setOutput(j);//////////
18                  this.denormalisasiT(this.getT());
19                  this.denormalisasiY(this.getY());
20                  this.error(this.getDenormY(),
this.getDenormT());
21              }
22              System.out.println("");
23              this.hitungMSE(banyakPola);
24              System.out.println("MSE = " + this.MSE);
25              this.fitness[i] = 1 / this.MSE;
26          }
27
28          //menghitung nilai fitness
29          System.out.println("FITNESS");
30          for (int i = 0; i < this.fitness.length; i++) {
31              System.out.println(this.fitness[i]);
32          }
33      }
```

Source Code 4.11 Evaluasi

4.4.10Implementasi Algoritme Seleksi pada Algoritme Genetika

Pada tahap seleksi bertujuan untuk mendapatkan individu – individu dengan solusi terbaik. Dengan cara seluruh individu termasuk *offspring* akan diurutkan dari fitness yang terbaik, kemudian akan dipilih individu sebanyak *popsize* yang ditunjukkan pada baris 4 - 64. Pada tahap ini parameter yang digunakan adalah *popsize* dan jumlah kromosom. Seluruhnya ditunjukkan pada Source Code 4.12.

```
1     public void seleksi(){
2         //mengurutkan fitness dari terbesar-terkecil
3         double temp;
4         int index;
5         int[] tempIndex = new int[this.fitness.length];
6         double[][] tempHimpunanIndividu = new
double[this.himpunanIndividu.length][this.stringLen];
7
8         for (int i = 0; i < tempIndex.length; i++) {
9             tempIndex[i] = i;
10        }
11
12        boolean selesai = true;
13        while(selesai){
14            selesai = false;
15            for (int i = 1; i < this.fitness.length;
i++) {
16                if (this.fitness[i-1] <
this.fitness[i]) {
17                    temp = this.fitness[i-1];
18                    index = tempIndex[i-1];
19                    this.fitness[i-1] =
this.fitness[i];
20                    tempIndex[i-1] = tempIndex[i];
21                    this.fitness[i] = temp;
22                    tempIndex[i] = index;
23                    selesai = true;
24                }
25            }
26        }
```

```
27
28          System.out.println("INDEX URUT");
29          for (int i = 0; i < tempIndex.length; i++) {
30              System.out.println(tempIndex[i]);
31          }
32
33          System.out.println("FITNESS URUT");
34          for (int i = 0; i < this.fitness.length; i++) {
35              System.out.println(this.fitness[i]);
36          }
37
38          System.out.println("INDIVIDU URUT");
39          for (int i = 0; i < this.himpunanIndividu.length; i++) {
40              for (int j = 0; j < this.stringLen; j++) {
41                  tempHimpunanIndividu[i][j] = this.himpunanIndividu[tempIndex[i]][j];
42
System.out.print(tempHimpunanIndividu[i][j]+" | ");
43              }
44              System.out.println("");
45          }
46
47          System.out.println("INDUK BARU");
48          for (int i = 0; i < this.individu.length; i++) {
49              for (int j = 0; j < this.stringLen; j++) {
50                  this.individu[i][j] = tempHimpunanIndividu[i][j];
51                  System.out.print(this.individu[i][j]+" | ");
52              }
53              System.out.println("");
54          }
55
56          this.individuTerbaik = new double[this.stringLen];
57          System.out.println("INDIVID TERBAIK");
```

```
58          for (int i = 0; i < this.stringLen; i++) {
59              this.individuTerbaik[i]                    =
this.individu[0][i];
60              System.out.print(this.individuTerbaik[i]+"
| ");
61          }
62
63          System.out.println("");
64          this.fitnessTerbaik = this.fitness[0];
65          System.out.println("FITNESS      TERBAIK      =
"+this.fitnessTerbaik);
66      }
67  }
```

Source Code 4.12 Seleksi

4.4.11Implementasi Algoritme Perhitungan MSE

Proses perhitungan MSE adalah melihat selisih dari hasil prediksi dengan data sebenarnya kemudian dibagi dengan banyaknya data input yang ditunjukkan pada baris 5 - 8. Selengkapnya ditunjukkan pada Source Code 4.13.

```
1  public void error(double output, double target){
2          this.MSE = this.MSE + Math.pow((output - target),
2);
3      }
4      //hitung MSE
5      public void hitungMSE(int banyakData){
6          double tempMSE = this.MSE;
7          this.MSE = tempMSE / banyakData;
8      }
```

Source Code 4.13 Perhitungan MSE

4.4.12Implementasi Algoritme Perhitungan AFER

Proses perhitungan AFER dilakukan dengan cara mencari nilai absolut dari data aktual dikurangi dengan data hasil prediksi, kemudian dibagi dengan data aktual. Hasil dari perhitungan tersebut akan dirata – rata dan dikali

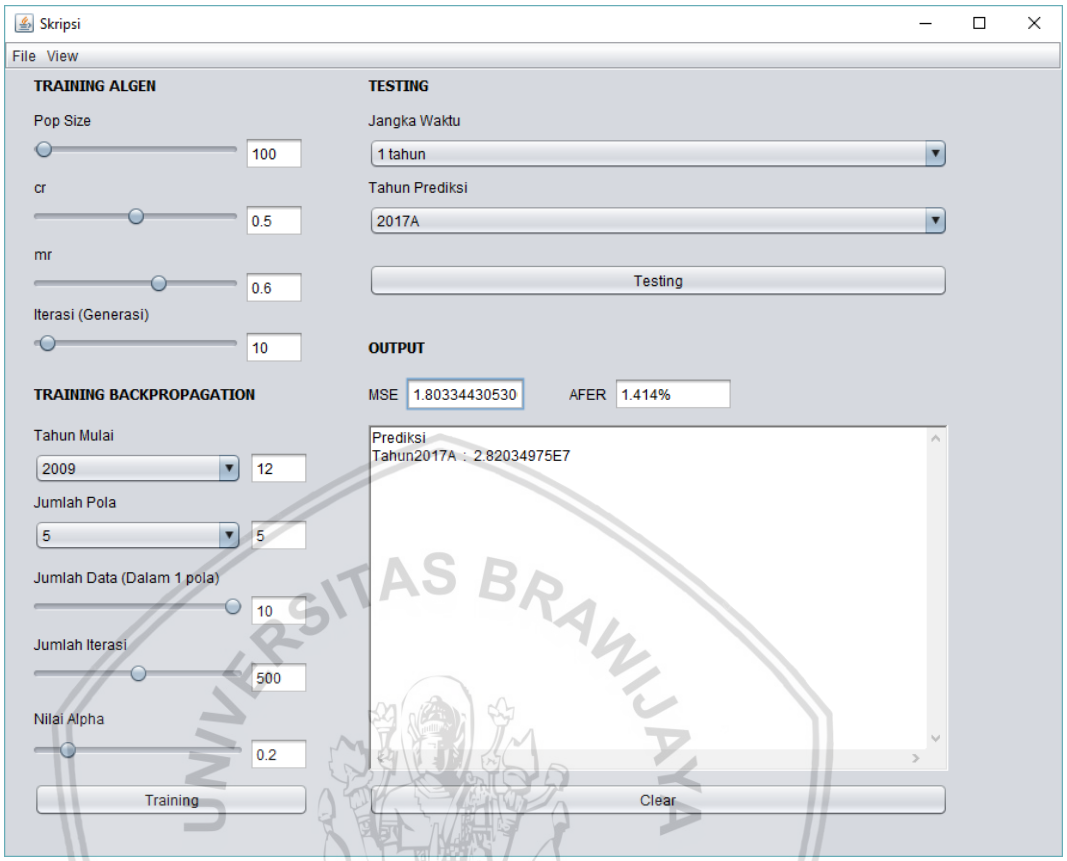
dengan 100% yang ditunjukkan pada baris 5 - 8. Selengkapnya akan ditunjukkan pada Source Code 4.14.

```
1     public void errorAFER(double output, double target){
2         this.AFER = this.AFER + Math.abs(((target - output) / target));
3     }
4
5     public void hitungAFER(int banyakData){
6         double tempAFER = this.AFER;
7         this.AFER = Double.parseDouble(df3.format((tempAFER / banyakData) * 100));
8     }
```

Source Code 4.14 Perhitungan AFER

4.5 Implementasi Antar Muka

Pada antar muka yang digunakan di penelitian ini terdapat beberapa elemen antara lain *Text field* yang digunakan untuk menampilkan hasil dari perhitungan MSE, AFER, dan komputasi, kemudian digunakan juga untuk menentukan beberapa parameter training seperti ukuran *popsize*, nilai Cr-Mr, jumlah generasi, tahun mulai, jumlah pola, jumlah data, jumlah iterasi dan nilai alpha. Selanjutnya combo box yang digunakan untuk memilih jangka waktu, dan tahun prediksi. Kemudian button yang digunakan untuk melakukan proses traning dan proses testing. Seluruh elemen tersebut akan ditunjukkan pada Gambar 4.18.



Gambar 4.18 Implementasi Antarmuka

BAB 5 PENGUJIAN DAN ANALISIS

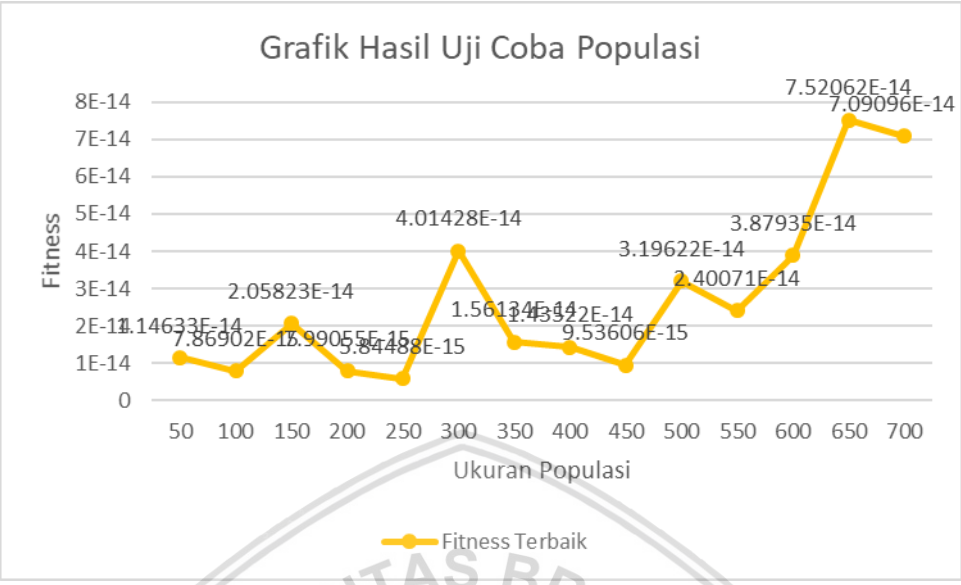
Bab ini akan menunjukkan hasil pengujian dan analisis dari prediksi jumlah penduduk miskin yang dihasilkan dari mengimplementasikan metode menggunakan *backpropagation* yang pelatihan bobotnya dioptimasi menggunakan algoritme genetika.

5.1 Uji Coba Populasi

Pada pengujian ini akan dicari nilai fitness yang paling optimal dari beberapa ukuran populasi pada algoritme genetika. Pengujian ini menggunakan data penduduk miskin di Indonesia pada tahun 2012B sebesar 6 generasi, Cr dan Mr dengan nilai 0.5 dan 0.5. Ukuran populasi yang diuji yaitu 50 sampai 700 dengan kelipatan 50. Dilakukan sebanyak 10 kali pengujian kemudian dihitung rata – rata fitness dari setiap uji coba. Hasil pengujian akan ditunjukkan pada Tabel 5.1 dan Gambar 5.1.

Tabel 5.1 Hasil Uji Coba Populasi

Uji Coba	Populasi	Fitness Terbaik
1	50	1.15E-14
2	100	7.87E-15
3	150	2.06E-14
4	200	7.99E-15
5	250	5.84E-15
6	300	4.01E-14
7	350	1.56E-14
8	400	1.44E-14
9	450	9.54E-15
10	500	3.2E-14
11	550	2.4E-14
12	600	3.88E-14
13	**650**	7.52E-14
14	700	7.09E-14



Gambar 5.1 Grafik Hasil Uji Coba Populasi

Dari Gambar 5.1 dan Tabel 5.1 hasil uji coba yang ditunjukkan, nilai fitness yang diperoleh dipengaruhi oleh ukuran populasi. Nilai *fitness* terkecil terdapat pada ukuran populasi sebesar 250, kemudian nilai *fitness* tertinggi didapatkan pada ukuran populasi sebesar 650 karena pada ukuran populasi berikutnya ukuran populasi cenderung menurun. Pada ukuran populasi 650 didapatkan nilai *fitness* tertinggi disebabkan karena daerah yang dieksplorasi lebih luas dibandingkan dengan ukuran populasi yang lebih kecil. Sehingga kemungkinan untuk didapatkannya solusi optimal lebih besar dibandingkan ukuran populasi yang lebih kecil.

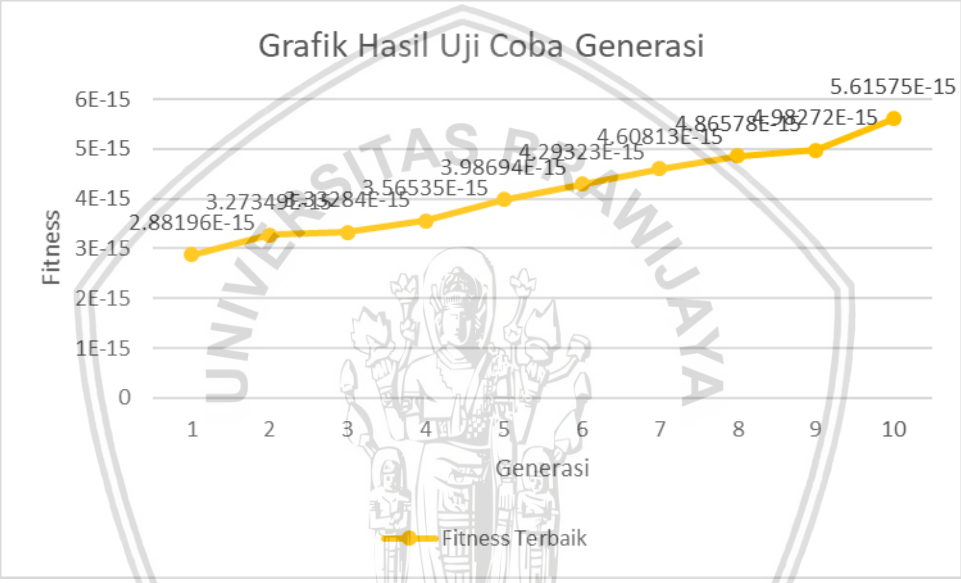
5.2 Uji Coba Generasi

Pada pengujian ini akan dicari nilai fitness yang paling optimal dari beberapa jumlah generasi pada algoritme genetika. Pengujian ini menggunakan data penduduk miskin di Indonesia pada tahun 2012B dengan ukuran populasi sebesar 650, kombinasi nilai Cr dan Mr sebesar 0.5 dan 0.5. Jumlah generasi yang diuji yaitu 1 sampai 10 generasi. Dilakukan 10 kali pengujian kemudian dicari rata-rata fitness dari tiap pengujian. Hasil pengujian akan ditunjukkan pada Tabel 5.2 dan Gambar 5.2.

Tabel 5.2 Hasil Uji Coba Generasi

Uji Coba	Generasi	Fitness Terbaik
1	1	7.47E-15
2	2	6.38E-15

3	3	9.29E-15
4	4	7.66E-15
5	5	8.7E-15
6	6	1.39E-14
7	7	5.62E-15
8	8	6.52E-15
9	9	6.38E-15
10	**10**	6.48E-15



Gambar 5.2 Grafik Hasil Uji Coba Generasi

Dari Gambar 5.2 dan Tabel 5.2 hasil uji coba yang ditunjukkan, nilai fitness yang diperoleh dipengaruhi oleh jumlah generasi. Terlihat pada grafik yang ditunjukkan pada Gambar 5.2 bahwa nilai fitness cenderung naik cenderung meningkat seiring dengan meningkatnya jumlah generasi. Nilai fitness tertinggi terdapat pada jumlah generasi 10 hal tersebut terjadi karena semakin banyak jumlah generasi maka daerah yang dieksplorasi juga akan semakin luas untuk mendapatkan solusi yang optimal. Namun semakin banyak jumlah generasi, waktu komputasi akan menjadi lebih lama.

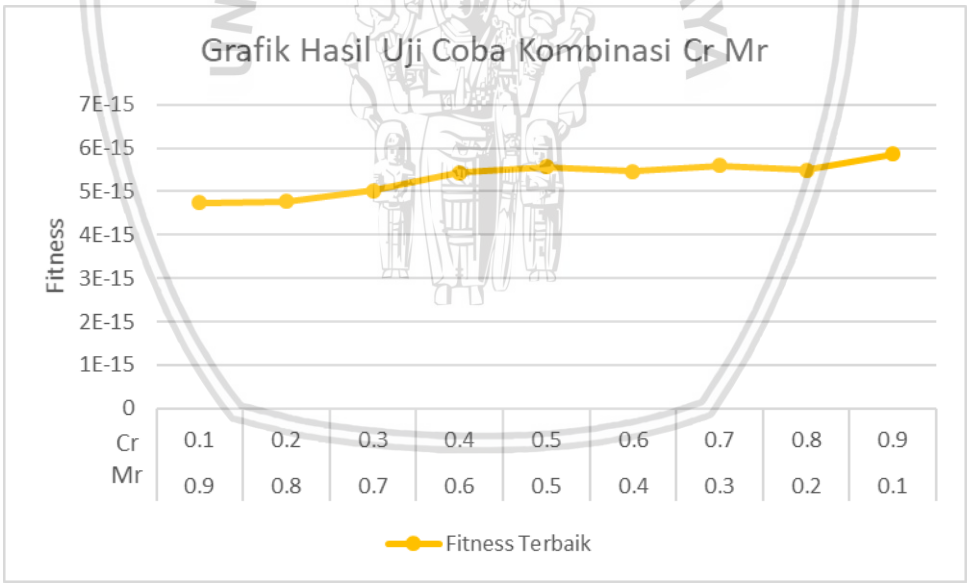
5.3 Uji Coba Kombinasi Cr dan Mr

Pada pengujian ini akan dicari nilai fitness yang paling optimal dari kombinasi nilai Cr dan Mr pada algoritme genetika. Pengujian ini menggunakan data penduduk miskin di Indonesia pada tahun 2012B dengan 650 untuk ukuran

populasi dan 10 generasi. Kombinasi nilai Cr dan Mr yang diuji yaitu nilai Cr dari 0.1 sampai 0.9 dan Nilai Mr dari 0.9 sampai 0.1. Dilakukan 10 kali pengujian kemudian dihitung rata-rata fitness dari tiap pengujian. Hasil pengujian akan ditunjukkan pada Tabel 5.3 dan Gambar 5.3.

Tabel 5.3 Hasil Uji Coba Kombinasi Cr dan Mr

Uji Coba	Cr	Mr	Fitness Terbaik
1	0.1	0.9	4.75E-15
2	0.2	0.8	4.77E-15
3	0.3	0.7	5.03E-15
4	0.4	0.6	5.43E-15
5	0.5	0.5	5.56E-15
6	0.6	0.4	5.46E-15
7	0.7	0.3	5.6E-15
8	0.8	0.2	5.5E-15
9	**0.9**	**0.1**	5.86E-15



Gambar 5.3 Grafik Hasil Uji Coba Kombinasi Cr dan Mr

Dari Gambar 5.3 dan Tabel 5.3 hasil uji coba yang ditunjukkan, nilai fitness yang diperoleh dipengaruhi oleh kombinasi parameter *crossover* dan mutasi yaitu Cr dan Mr. Seperti yang ditunjukkan Gambar 5.3 kombinasi Cr dan Mr yang menghasilkan nilai fitness terbaik pada kombinasi Cr sebesar 0.9 dan Mr sebesar 0.1. Hal tersebut menunjukkan bahwa nilai Cr lebih berpengaruh untuk

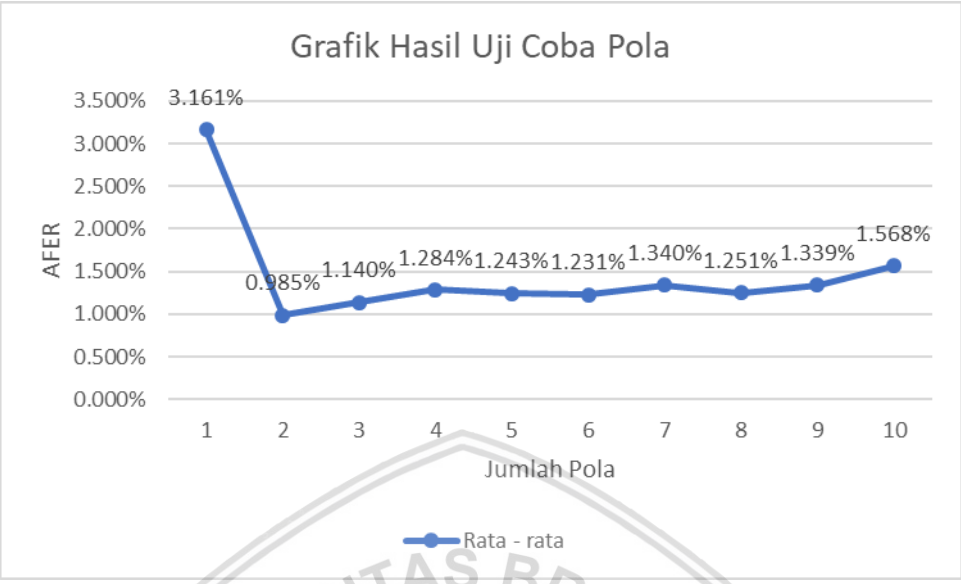
mendapatkan nilai fitness yang optimal dibandingkan dengan nilai Mr pada penelitian ini.

5.4 Uji Coba Jumlah Pola Terhadap AFER

Pada pengujian ini akan dicari nilai AFER yang paling optimal dari beberapa jumlah pola yang dilatih pada *backpropagation*. Pola yang dimaksud adalah data jumlah penduduk miskin pada tahun – tahun yang berurutan yang digunakan pada proses training. Pengujian ini menggunakan beberapa parameter yaitu nilai alpha sebesar 0.1 dan jumlah iterasi sebesar 1000. dilakukan 10 kali pengujian dengan pola yang berbeda untuk memprediksi jumlah penduduk miskin pada tahun 2017A. Hasil pengujian akan ditunjukkan pada Tabel 5.4 dan Gambar 5.4.

Tabel 5.4 Hasil Uji Coba Pola

Jumlah Pola	Tahun Awal	Tahun Prediksi	Rata - rata
1	2012B	2017A	3.16140%
2	**2012A**	**2017A**	0.98493%
3	2011B	2017A	1.13966%
4	2011A	2017A	1.28431%
5	2010	2017A	1.24305%
6	2009	2017A	1.23094%
7	2008	2017A	1.34040%
8	2007	2017A	1.25074%
9	2006	2017A	1.33931%
10	2005	2017A	1.56784%



Gambar 5.4 Grafik Hasil Uji Coba Pola Terhadap Nilai AFER

Dari Gambar 5.4 dan Tabel 5.4 hasil uji coba yang ditunjukkan, nilai AFER yang diperoleh dipengaruhi oleh pola yang diuji. Pada pengujian pola terhadap nilai AFER terlihat nilai AFER terbaik terdapat pada jumlah pola sebesar 2 pola dengan nilai AFER sebesar 0.98493% dan kemudian pada pola – pola selanjutnya nilai AFER cenderung menurun. Hal tersebut menunjukkan bahwa semakin banyak pola yang digunakan, maka akan menghasilkan nilai AFER yang semakin menurun.

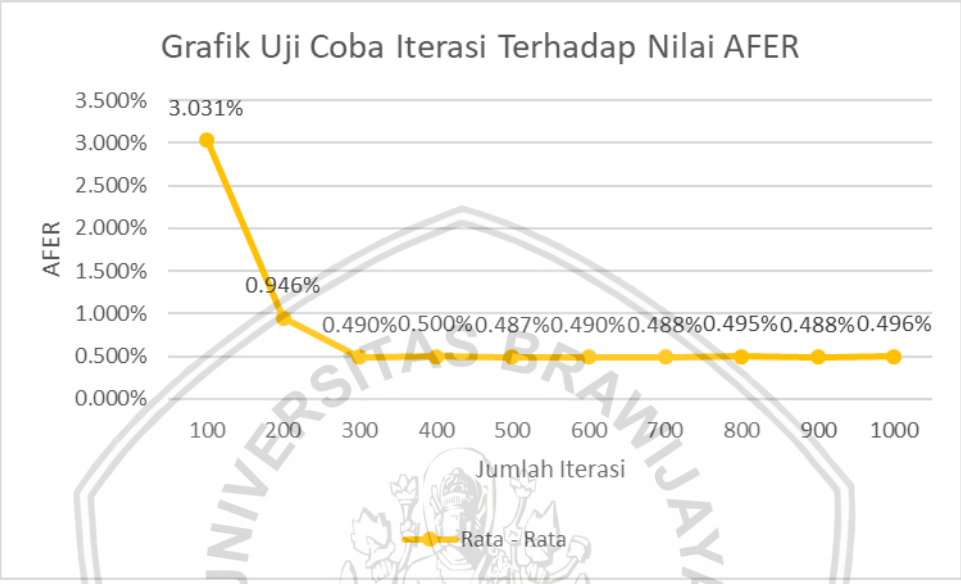
5.5 Uji Coba Jumlah Iterasi Terhadap Nilai AFER

Pada pengujian ini akan dicari nilai AFER yang paling optimal dari jumlah iterasi pada *backpropagation*. Pengujian ini menggunakan jumlah pola sebesar 2 yang digunakan untuk memprediksi tahun 2017A, pola tersebut diperoleh berdasarkan pengujian sebelumnya. Dilakukan 10 kali pengujian dengan nilai *alpha* (α) sebesar 0.1 dan jumlah iterasi yang diuji yaitu 100 hingga 1000 dengan kelipatan 100 iterasi per-uji coba. Pada tabel 5.5 dan Gambar 5.5 akan ditampilkan hasil pengujian jumlah iterasi.

Tabel 5.5 Hasil Uji Coba Jumlah Iterasi Terhadap Nilai AFER

Uji Coba	Jumlah Iterasi	Rata - Rata
1	100	3.03139%
2	200	0.94641%
3	**300**	0.48991%
4	400	0.50023%
5	500	0.48673%

6	600	0.49001%
7	700	0.48847%
8	800	0.49529%
9	900	0.48846%
10	1000	0.49579%



Gambar 5.5 Grafik Hasil Uji Coba Jumlah Iterasi Terhadap Nilai AFER

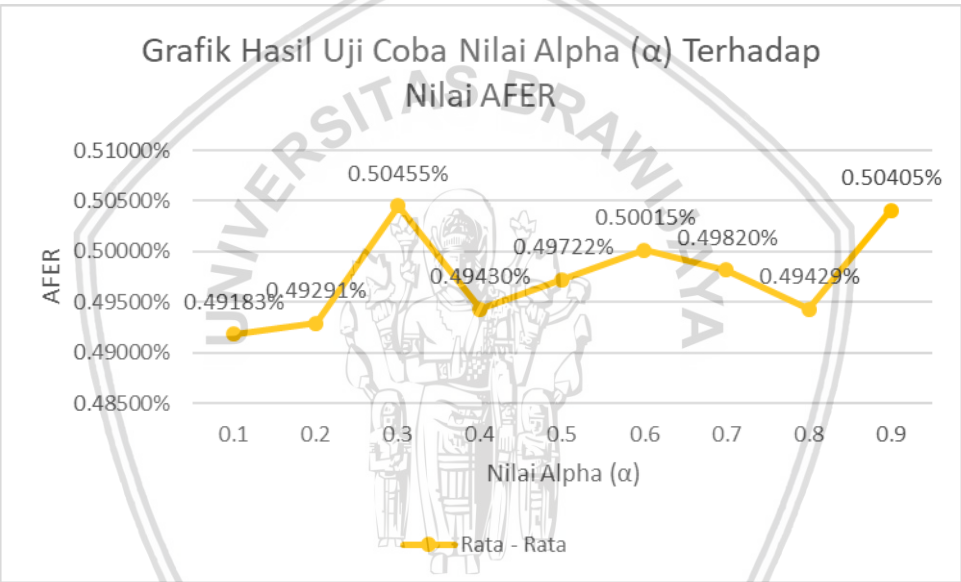
Dari Gambar 5.5 dan Tabel 5.5 hasil uji coba yang ditunjukkan, nilai AFER yang diperoleh dipengaruhi oleh jumlah iterasi. Pada pengujian jumlah iterasi terhadap nilai AFER terlihat penurunan nilai AFER yang signifikan sampai iterasi ke 300, kemudian pada iterasi selanjutnya nilai AFER yang dihasilkan bernilai konvergen, maka pada penelitian ini jumlah iterasi yang optimal sebesar 300 iterasi. Hal tersebut terjadi karena semakin banyaknya iterasi biasanya akan mendapatkan hasil yang lebih baik namun pada titik tertentu hasil yang didapatkan tidak memberikan perubahan yang signifikan dan cenderung memakan waktu komputasi yang lebih lama.

5.6 Uji Coba Nilai Alpha (α) Terhadap Nilai AFER

Pada pengujian ini akan dicari nilai AFER yang paling optimal dari nilai *alpha* (α) pada *backpropagation*. Pengujian ini menggunakan jumlah pola sebesar 2 yang digunakan untuk memprediksi tahun 2017A dan jumlah iterasi sebesar 300 yang didapatkan dari hasil pengujian sebelumnya. Pengujian dilakukan sebanyak 10 kali dengan nilai *alpha* (α) dengan *range* 0.1 sampai 0.9. pada Tabel 5.6 dan Gambar 5.6 akan ditampilkan hasil dari pengujiannya.

Tabel 5.6 Hasil Uji Coba Nilai *Alpha* (α) Terhadap Nilai AFER

Uji Coba	Nilai Alpha	Rata - Rata
1	0.1	0.49183%
2	0.2	0.49291%
3	0.3	0.50455%
4	0.4	0.49430%
5	0.5	0.49722%
6	0.6	0.50015%
7	0.7	0.49820%
8	0.8	0.49429%
9	0.9	0.50405%



Gambar 5.6 Grafik Hasil Uji Coba Nilai *Alpha* (α) Terhadap Nilai AFER

Dari Gambar 5.6 dan Tabel 5.6 hasil uji coba yang ditunjukkan, nilai AFER yang diperoleh dipengaruhi oleh nilai *alpha* (α). Pada pengujian nilai *alpha* (α) terhadap nilai AFER terlihat pada nilai AFER terbaik didapatkan pada nilai *alpha* (α) sebesar 0.1, kemudian pada nilai alpha selanjutnya menghasilkan nilai AFER yang cenderun menurun. Hal tersebut menunjukkan bahwa untuk mendapatkan hasil yang optimal, sistem membutuhkan ketelitian yang tinggi.

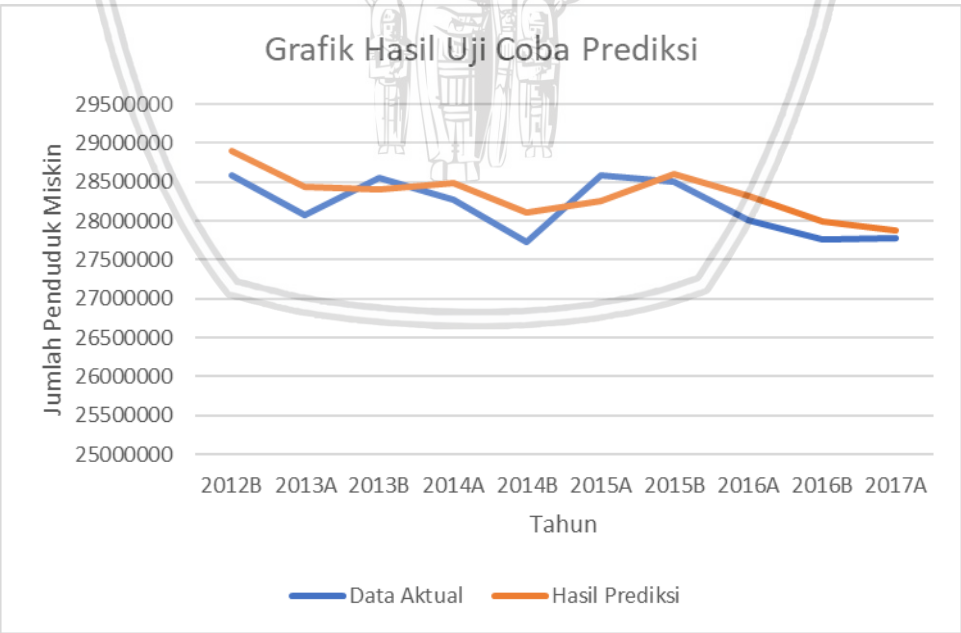
5.7 Uji Coba Hasil Prediksi

Hasil prediksi akan dicari pada pengujian ini menggunakan algoritme *backpropagation* dimana bobot pelatihannya dioptimasi dengan algoritme genetika dalam rentang waktu 10 tahun. Tahun yang diprediksi dimulai dari tahun

2012B sampai 2017A. Prediksi menggunakan parameter – parameter yang diperoleh dari pengujian sebelumnya. Untuk memprediksi tahun 2012B dibutuhkan data asli dari tahun 2005 sampai 2012A, kemudian untuk memprediksi tahun 2013A menggunakan data asli dari tahun 2006 sampai 2012B. Dengan cara yang sama akan dilakukan sampai tahun 2017A. Pada Tabel 5.7 dan Gambar 5.7 akan ditampilkan hasil uji coba prediksi.

Tabel 5.7 Hasil Uji Coba Prediksi

Tahun	Data Aktual	Hasil Prediksi	Selisih	*|(Ai - Fi) / Ai|*
2012B	28590000	28897416.25	307416	0.01075258
2013A	28070000	28429361.25	359361	0.012802325
2013B	28550000	28404870	145130	0.005083363
2014A	28280000	28491950	211950	0.007494696
2014B	27730000	28100090	370090	0.013346195
2015A	28590000	28252480	337520	0.011805526
2015B	28510000	28598078.75	88079	0.003089398
2016A	28010000	28317790	307790	0.010988576
2016B	27760000	27988518.75	228519	0.008231943
2017A	27770000	27876947.5	106948	0.003851188
AFER (%)				8.744579015



Gambar 5.7 Grafik Hasil Uji Coba Prediksi

Hasil Analisis

Dari Tabel 5.7 hasil uji coba prediksi yang dilakukan menggunakan *Backpropagation* yang bobot pelatihannya dioptimasi menggunakan algrotime genetika dan jumlah fitur sebanyak 8 unit, diperoleh parameter optimal dengan ukuran populasi sebesar 650 populasi, jumlah generasi sebesar 10 generasi, kombinasi Cr sebesar 0.9 dan Mr sebesar 0.1, jumlah pola sebanyak 2 pola, jumlah iterasi sebanyak 300 iterasi, nilai *alpha* sebesar 0.1, hasil prediksi dengan selisih terkecil terdapat pada prediksi tahun 2015B dengan selisih jumlah penduduk sebesar 88079, kemudian prediksi dengan selisih terbesar berada pada prediksi tahun 2014B dengan selisih jumlah penduduk sebesar 370090, hal tersebut terjadi karena pada tahun tersebut jumlah penduduk miskin terjadi penurunan yang pada tahun berikutnya kembali mengalami kenaikan yang membuat sistem sulit untuk melakukan prediksi. Berdasarkan hasil uji coba ini, menggunakan metode *Backpropagation* yang bobot pelatihannya dioptimasi menggunakan algrotime genetika bisa digunakan untuk memprediksi jumlah penduduk miskin dengan tingkat akurasi AFER sebesar 8.744579%.

BAB 6 PENUTUP

6.1 Kesimpulan

Dari penelitian yang telah dilakukan menggunakan metode jaringan syaraf tiruan *backpropagation* yang bobot pelatihannya dioptimasi menggunakan algoritme genetika untuk memprediksi jumlah penduduk miskin di Indonesia, dapat ditarik kesimpulan antara lain:

1. Metode Jaringan Syaraf Tiruan *backpropagation* yang bobot pelatihannya dioptimasi menggunakan algoritme genetika bisa dipakai untuk melakukan prediksi jumlah penduduk miskin di Indonesia dengan 2 tahapan. Tahap pertama menggunakan algoritme genetika untuk mengoptimasi bobot dan bias *w*, menggunakan representasi kromosom *real-code* dengan panjang kromosom 9 yang menggambarkan bobot dan bias *w*, pada proses reproduksi menggunakan metode *one-cut point* pada proses *crossover* dan metode *reciprocal exchange* pada proses mutasi, selanjutnya menggunakan metode *elitsm selection* pada proses seleksi dan fungsi fitness yang digunakan yaitu 1/MSE. Kemudian pada tahap kedua menggunakan *backpropagation* untuk melakukan prediksi.
2. Untuk menghasilkan hasil yang optimal, pada penelitian ini menggunakan nilai parameter yang diperoleh dari hasil pengujian, antara lain pada algoritme genetika dengan populasi sebesar 650, generasi sebesar 10, kombinasi nilai Cr sebesar 0.9 dan Mr sebesar 0.1. kemudian parameter *backpropagation* menggunakan 10 pola *training*, iterasi sebesar 300, nilai *alpha* (α) sebesar 0.1. Menggunakan parameter tersebut diperoleh nilai AFER sebesar 8.744579%.

6.2 Saran

Terdapat beberapa saran yang bisa digunakan pada penelitian selanjutnya dengan topik serupa yaitu bisa diberika variabel tambahan selain jumlah penduduk miskin, seperti banyaknya lapangan pekerjaan yang ada atau banyaknya pengangguran. Selanjutnya dilakukan pengujian untuk mengetahui jumlah fitur yang paling optimal, karena pada penelitian ini menggunakan fitur sebanyak 8 fitur. Kemudian menggunakan metode lain untuk mengoptimasi pelatihan bobot pada *backpropagation*, seperti *Naive Bayes* atau algoritme optimasi yang lain.

DAFTAR PUSTAKA

Adwandha, D. P., Ratnawati, D. E., & Adikara, P. P. (2017). Prediksi Jumlah Pengangguran Terbuka di Indonesia menggunakan Metode Genetic-Based Backpropagation. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, Vol.1*(No.1), 341-351.

Amina, F., & Irawan, M. I. (2014). Prediksi Jumlah Penduduk Miskin Di Provinsi Kalimantan Selatan Menggunakan Jaringan Saraf Tiruan Backpropagation. *Pasca Sarjana Jurusan Matematika FMIPA, Institut Teknologi Sepuluh*.

Anindita, A. (2017). Dampak Pertumbuhan Ekonomi, Jumlah Pengangguran Dan Kesejahteraan Masyarakat Terhadap Tingkat Kemiskinan Di Kabupaten Sidoarjo. *Peningkatan Ketahanan Ekonomi* (pp. 130-137). Malang: Seminar Nasional & Call For Paper, FEB Unikama.

Biro Pusat Statistik. (2016). Persentase Penduduk Miskin dan Garis Kemiskinan, 1970-2017.

Bustomi, A., Bisri, H., & Purwanti, E. (2014). Desain Perangkat Lunak Berbasis Jaringan Syaraf Tiruan Backpropagation untuk Klasifikasi Citra Rontgen Paru-paru. *JURNAL FISIKA DAN APLIKASINYA, Vol. 10*(No. 1), 19-23.

Deborah, M., & Prathap. (2014). Detection of Fake currency using Image Processing. *IJISET, Vol.1*(No.10), 151–157.

Fausett, L. (1994). *Fundamentals of Neural Networks (Prentice-H).* Upper Saddle River, NJ, USA: Association for Computing Machinery.

Haviluddin, & Alfred, R. (2015). A Genetic-Based Backpropagation Neural Network for Forecasting in Time-Series Data. (p. 6). Yogyakarta, Indonesia: IEEE. Retrieved from https://doi.org/10.1109/ICSITech.2015.7407796

Hudiyawan, A. R. (2015). Prediksi FOREX (Foreigen Exchange) Menggunakan Jaringan Syaraf Tiruan Al-Alaoi Backpropagation. *Laporan Tugas Akhir. Jurusan Teknik Informatika Universitas Brawijaya Malang*.

Jauhari, D., Himawan, A., & Dewi, C. (2016). *Prediksi Distribusi Air PDAM Menggunakan Metode Jaringan Syaraf Tiruan Backpropagation Di PDAM Kota Malang* (Vol. 3). Malang: Jurnal Teknologi Informasi Dan Ilmu Komputer.

Kaplale, R. (2012). Faktor-Faktor Yang Mempengaruhi Tingkat Kemiskinan Di Kota Ambon (Study Kasus Di Dusun Kranjang Desa Waiyame Kec. Teluk Ambon Dan Desa Waiheru Kec. Teluk Ambon Baguala Kota Ambon). *Jurnal Agribisnis Kepulauan (AGRILAN), Vol.1*(No.1), 110-115.

Lestari, Y. D. (2017). JARINGAN SYARAF TIRUAN UNTUK PREDIKSI. *Jurnal ISD, Vol.2*(No.1), 40-46.

Munawar, H. (2015). Prediksi Tingkat Kemiskinan di Provinsi Aceh dengan Model AR. *Jurnal Gradien, Vol.11*(No.1), 1061-1065.

Noor, M. (2014). PENANGGULANGAN KEMISKINAN DI INDONESIA (Studi Tentang Program Nasional Pemberdayaan Masyarakat Mandiri Perkotaan Di Kota Semarang). *Serat Acitya – Jurnal Ilmiah UNTAG Semarang, Vol.3*(No.1), 130-141.

Nugroho, K. (2012). Model Analisis Prediksi Menggunakan Metode Fuzzy Time Series. *INFOKAM, Vol.12*(No.1), 47-50.

Nugroho, Z. A., & Setyawan, Y. (2016). JARINGAN SYARAF TIRUAN DAN NAIVE BAYES UNTUK MENDETEKSI PENYAKIT. *Prosiding Seminar Nasional Aplikasi Sains & Teknologi (SNAST)*, 540-549.

Pakaja, F., Naba, A., & Purwanto. (2012). Peramalan Penjualan Mobil Menggunakan Jaringan Syaraf Tiruan dan Certainty Factor. *Jurnal EECCIS, Vol. 6*(No. 1), 23-28.

Rumelhart, D., Hinton, G., & Williams, R. (1986). *Learning Internal Representations by Error.* Cambridge: Parallel Distributed Processing, chapter 8, MIT Press.

Syukriyawati, G. (2015). *Implementasi Metode Average-Based Fuzzy Time Series Models pada Prediksi Jumlah Penduduk Provinsi DKI Jakarta.* Brawijaya University.

Witarayoga, R. (2016). *Prediksi Tingkat Kemiskinan Indonesia Menggunakan Algoritma Genetika.* Yogyakarta: Universitas Gajah Mada.

World Bank. (2006). *Making The New Indonesia Work For The Poor.* Jakarta: World Bank.

Yohannes, E., Mahmudy, W. F., & Rahmi, A. (2015). Penentuan Upah Minimum Kota Berdasarkan Tingkat Inflasi Menggunakan Backpropagation Neural Network (BPNN). *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK), Vol.2*(No.1), 34-40.