IMPLEMENTASI ALGORITMA GENETIKA UNTUK OPTIMASI LVQ PADA PENENTUAN KELAYAKAN KREDIT

(Studi Kasus : Bank X)

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Aghata Agung Dwi K
NIM: 135150201111044

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

PENGESAHAN

IMPLEMENTASI ALGORITMA GENETIKA UNTUK OPTIMASI LVQ PADA PENENTUAN KELAYAKAN KREDIT
(STUDI KASUS : BANK X)

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Aghata Agung Dwi K
NIM: 135150201111044

Skripsi ini telah diuji dan dinyatakan lulus pada
3 Agustus 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Candra Dewi, S.Kom, M.Sc	Indriati, S.T, M.Kom
NIP: 19771114 200312 2 001	NIP: 19831013 201504 2 002

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, 3 Agustus 2018

Aghata Agung Dwi K

NIM: 135150201111044

KATA PENGANTAR

Dengan mengucap syukur kepada Allah SWT, yang telah memberikan berkah, sehingga penulis dapat menyelesaikan pengerjaan skripsi dengan judul: Implementasi Algoritma Genetika untuk Optimasi LVQ pada Penentuan Kelayakan Kredit (Studi Kasus : Bank X). Dalam pengerjaan skripsi ini penulis telah mendapatkan berbagai bantuan, dukungan moral dan materil, sehingga penulis ingin menyampaikan rasa hormat dan terimakasih kepada:

1. Ibu Candra Dewi, S.Kom., M.Sc. selaku dosen pembimbing pertama yang telah memberi saran, semangat, dan motivasi kepada penulis sampai pengerjaan skripsi ini bisa terselesaikan.
2. Ibu Indriati, S.T, M.Kom. selaku dosen pembimbing dua yang telah memberikan dukungan serta saran dalam pengerjaan skripsi ini.
3. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D. selaku ketua jurusan Teknik Informatika dan Bapak Agus Wahyu Widodo, S.T, M.Cs selaku ketua prodi Informatika.
4. Bapak dan ibu dosen Fakultas Ilmu Komputer yang telah memberikan ilmu dan pengetahuan kepada penulis untuk menuju jenjang sarjana di Fakultas Ilmu Komputer.
5. Seluruh karyawan Fakultas Ilmu Komputer yang telah membantu penulis dalam memudahkan pelaksanaan pengerjaan skripsi ini.
6. Kedua orang tua, dan seluruh keluarga besar penulis yang selalu mendo'akan, memberikan kasih sayang, semangat, dukungan, dan motivasi kepada penulis.
7. Teman-teman Fakultas Ilmu Komputer yang terlibat secara langsung maupun tidak langsung yang tidak bisa penulis sebutkan satu persatu.
8. Ardianto Nugroho, Edi Juliyanto, Fariz R.A, M. Fawwaz Afiv, Risfan W.G, yang telah membantu penulis sejak awal perkuliahan.

Malang, 3 Agustus 2018

Penulis

ghatakyo@gmail.com

ABSTRAK

Dalam memberikan kredit kepada debitur, terkadang terdapat permasalahan yang timbul seperti kesalahan analisis terhadap debitur, dan menimbulkan kredit macet. Oleh sebab itu perlu adanya seleksi yang lebih akurat terhadap calon debitur. Namun untuk melakukan analisis yang teliti dan konsisten dibutuhkan waktu yang lama akibat banyaknya data yang harus dianalisis. Permasalahan tersebut dapat dituntaskan dengan membuat sistem analisa kredit menggunakan metode *Learning Vector Quantization* (LVQ) untuk mengklasifikasikan data calon debitur dan menentukan kelayakan kredit calon debitur. Tetapi penggunaan metode *Learning Vector Quantization* (LVQ) bergantung pada bobot yang diberikan dalam memberikan keakuratan hasil analis. Hal itu ditunjukkan dengan nilai akurasi yang didapatkan sebesar 79.37% dari pengujian menggunakan 63 data uji. Untuk mendapatkan nilai akurasi yang lebih optimal, bobot yang akan digunakan pada metode *Learning Vector Quantization* (LVQ) akan dioptimasi menggunakan algoritme genetika. Berdasarkan hasil pengujian sistem yang telah menggunakan bobot optimal, sistem mendapatkan nilai akurasi sebesar 93.65% dengan parameter *popsize* 20, Cr 0.9, Mr 0.1 dan jumlah generasi 10.

Kata kunci: *kredit, optimasi, algoritma genetika, jaringan saraf tiruan, Learning Vector Quantization* (LVQ).

ABSTRACT

In giving credit to the debitor, sometimes there are problems that arise such as error analysis to the debitor, and generate bad loans. Therefore, there needs to be a more accurate selection of prospective borrowers. But to do a thorough and consistent analysis takes a long time due to the amount of data that must be analyzed. This problem can be confirmed by making credit analysis system using Learning Vector Quantization (LVQ) method to classify the debitor data of the debitor and determine the creditworthiness of the debitor candidate. But the use of the Learning Vector Quantization (LVQ) method depends on the weight given in providing the accuracy of the analyst's results. It is shown with the obtained accuracy value of 79.37% of the test using 63 test data. To obtain a more optimal accuracy value, the weights that will be used in the Learning Vector Quantization (LVQ) method will be optimized using genetic algorithm. Based on the results of testing systems that have used the optimal weight, the system obtained an accuracy of 93.65% with parameter popsize 20, Cr 0.9, Mr. 0.1 and the number of generations 10.

Keywords: *credit, optimation, genetic algorithm, artificial neural network, Learning Vector Quantization (*LVQ*).*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR SOURCE CODE

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar belakang

Kredit adalah kemampuan untuk melaksanakan suatu pembelian atau mengadakan suatu pinjaman dengan suatu janji, pembayaran akan dilaksanakan pada jangka waktu yang telah disepakati. (Astiko, 1996). Kredit yang dalam bahasa latin disebut *credo* atau *credere* berarti percaya dapat dideskripsikan sebagai peminjaman berupa uang, barang atau jasa dari pihak pemberi pinjaman kepada peminjam yang didahului dengan kesepakatan pengembalian dalam jangka waktu yang disepakati kedua belah pihak. Pengembalian kredit dalam dunia perbankan disertai dengan pembayaran bunga atau hasil bagi keuntungan bersama, yang dibebankan pada pihak peminjam. Bunga yang dibayarkan oleh peminjam berdasarkan kesepakatan sesuai dengan jenis kredit yang diajukan pada pihak bank.

Dalam penentuan kelayakan kredit terlebih dahulu dilakukan analisis terhadap data–data dari sarat yang telah dikumpulkan oleh peminjam. Diperlukan ketelitian dalam menentukan kebutuhan dan kondisi calon peminjam agar menghasilkan keputusan kredit yang layak dengan efektif. Namun dalam hal ini terkadang masih banyak debitur yang diterima kreditnya tetapi mengalami pembayaran yang kurang lancar, bahkan macet. Oleh karena itu diperlukan penelitian lebih lanjut untuk menentukan pemberian kredit.

Terdapat penelitian sebelumnya yang membahas permasalahan penentuan pemberian kredit yang menggunakan metode TOPSIS (*Technique for Order Preference by Similiarity to Ideal Solution*) dan metode SAW (*Simple Additive Weighting*) oleh Harinda Bonita (2016) yang memiliki hasil akurasi pengujian pada penelitian ini didapatkan dengan mencocokkan data awal nasabah PT. Bank X dengan hasil perhitungan sistem. Terdapat pula penelitian oleh Wibowo (2017), tentang kredit motor menggunakan metode Jaringan Saraf Tiruan Backpropagation.

Berdasarkan latar belakang diatas, metode Jaringan Saraf Tiruan Backpropagation memiliki akurasi yang tinggi. Namun dalam sebuah penelitian oleh Wuryandari dan Afrianto (2012), menyatakan ada jenis jaringan saraf tiruan yang memliki akurasi yang lebih tinggi dan waktu komputasi yang lebih cepat dibandingkan jenis backpropagation, yaitu *Learning Vector Quantization*.

Dalam beberapa penelitian sebelumnya banyak yang membahas cara meningkatkan performa dari metode LVQ, misalnya optimasi pada bobot metode jaringan syaraf tiruan menggunakan algoritma genetika oleh Hendry dkk. (2009). Terdapat juga penelitian yang menggunakan pembelajaran *Learning Vector Quantization* oleh Wijayanti, dkk. (2016).

Berdasarkan pengamatan penulis terhadap latar belakang diatas maka penulis mengambil judul skripsi “Implementasi Algoritma Genetika pada

Optimasi Bobot *Learning Vector Quantization* untuk Penentuan Kelayakan Kredit (Studi Kasus: Bank X)".

1.2 Rumusan masalah

1. Bagaimana mengoptimasi bobot optimasi *Learning Vector Quantization* (LVQ) dengan Algoritma Genetika untuk kelayakan kredit bagi calon nasabah?
2. Bagaimana tingkat akurasi hasil optimasi bobot optimal *Learning Vector Quantization* (LVQ) dengan Algoritma Genetika terhadap penentuan kelayakan kredit bagi calon nasabah ?

1.3 Tujuan

1. Menerapkan Algoritma Genetika pada optimasi bobot optimal *Learning Vector Quantization* (LVQ) untuk menentukan kelayakan kredit bagi nasabah BANK.
2. Menguji tingkat akurasi hasil optimasi bobot optimal *Learning Vector Quantization* (LVQ) dengan Algoritma Genetika.

1.4 Manfaat

Manfaat yang diharapkan penulis dari penelitian ini adalah dapat digunakan sebagai bahan pertimbangan dalam menentukan kelayakan nasabah dalam menerima kredit khususnya pada Bank Daerah Lamongan.

1.5 Batasan masalah

1. Reproduksi pada Algoritma Genetika menggunakan *crossover: two cut point,* dan mutasi: *random mutation.*
2. Data yang digunakan dalam penelitian ini berupa nilai Prinsip 5C kredit (*Character, Condition, Capital, Capacity, Collateral*), usia, dan tanggungan berasal dari Bank X tahun 2012.

1.6 Sistematika pembahasan

Bagian ini berisi struktur skripsi ini mulai Bab Pendahuluan sampai Bab Metodologi dan deskripsi singkat dari masing-masing bab. Diharapkan bagian ini dapat membantu pembaca dalam memahami sistematika pembahasan isi dalam skripsi ini.

Bab I PENDAHULUAN

Bab ini menjelaskan latar belakang masalah, rumusan masalah, batasan masalah, tujuan, dan manfaat serta sistematika penulisan yang terkait dengan Implementasi Algoritma Genetika untuk Optimasi LVQ pada Penentuan Kelayakan Kredit Studi Kasus (Bank X).

Bab II LANDASAN KEPUSTAKAAN

Bab ini memuat dan menjelaskan tentang dasar teori apa saja yang berhubungan dengan topik penulisan yang diangkat dan menjadi panduan dasar dalam pembuatan sistem pada penelitian ini. Hal yang dibahas meliputi tentang *Kredit, Algoritma Genetika, Crossover, Mutation, Jaringan Saraf tiruan* (JST)*, Learning Vector Quantization* (LVQ).

Bab III METODOLOGI

Bab ini menjelaskan tentang metode dan langkah-langkah yang digunakan dalam penelitian Implementasi Algoritma Gentika untuk Optimasi LVQ pada Penentuan Kelayakan Kredit Studi Kasus (Bank X). Metode penelitian terdiri dari studi literatur, analisis kebutuhan, pengumpulan data, perancangan sistem, implementasi, pengujian sistem dan analisis, penarikan kesimpulan dan saran.

Bab IV PERANCANGAN

Bab ini menjelaskan tentang perancangan pengujian populasi, pengujian nilai *cr* dan nilai *mr*, pengujian jumlah generasi, pengujian hasil akurasi optimasi bobot optimal, serta menguji hasil akurasi metode *Learning Vector Quantization* (LVQ) tanpa bobot optimal hasil optimasi algoritma genetika

Bab V IMPLEMENTASI

Bab ini menjelaskan tentang implementasi pengujian populasi, pengujian nilai *cr* dan nilai *mr*, pengujian jumlah generasi, pengujian hasil akurasi optimasi bobot optimal, serta menguji hasil akurasi metode *Learning Vector Quantization* (LVQ) tanpa bobot optimal hasil optimasi algoritma genetika.

Bab VI PENGUJIAN DAN ANALISIS

Bab ini menjelaskan analisis untuk menentukan nilai akurasi dari perbandingan data hasil pengolahan perhitungan manual dengan perhitungan yang dilakukan oleh sistem untuk Implementasi Algoritma Genetika dan LVQ pada Penentuan Kelayakan Kredit studi kasus Bank X.

Bab VII PENUTUP

Bab ini menjelaskan penarikan kesimpulan setelah analisis hasil dari pengujian dengan metode yang diterapkan dan saran untuk penelitian selanjutnya.

BAB 2 LANDASAN KEPUSTAKAAN

Bab ini memuat dan menjelaskan tentang dasar teori dan kajian pustaka apa saja yang berhubungan dengan topik penulisan yang diangkat dan menjadi panduan dasar dalam pembuatan sistem pada penelitian ini. Hal yang dibahas meliputi: kredit, algoritma genetika, *crossover, Mutation,* Jaringan Saraf Tiruan (JST), *Learning Vector Quantization* (LVQ).

2.1 Kajian pustaka

Penelitian ini dilakukan berdasar beberapa penelitian sebelumnya yang terkait dengan penelitian ini baik dari sisi metode maupun objek penelitian. Daftar penelitian yang dijadikan kajian pustaka dapat dilihat pada tabel berikut:

Tabel 2.1 Kajian Pustaka

No.	Judul	Obyek	Metode	Hasil
1.	Penggunaan Metode Topsis dan SAW untuk Penentuan Pemberian Kredit Pensiunan bagi Calon Nasabah (Studi Kasus: PT. Bank X)	Kredit Pensiunan	Topsis dan SAW	Pengujian ini memperoleh akurasi sebesar 63.33%. Hal ini dikarenakan nilai range dan bobot yang dipakai dari setiap kriteria pada perhitungan sistem, dan data yang digunakan kurang beragan dan seimbang, sehingga mempengaruhi hasil pengujian akurasi tersebut.
2.	Analisis dan Implementasi Optimasi Jaringan Saraf Tiruan dengan Menggunakan Algoritma Genetika untuk Pendiagnosaan Penyakit Stroke (Studi Kasus : RS. DR. M. Djamil Padang Sumbar	Penyakit Stroke	Jaringan Saraf Tiruan dan Algoritma Genetika	Pengujian sistem menggunakan JST-AG mendapatkan hasil akurasi sebesar 100%, sedangkan pengujian menggunakan JST hanya mencapai akurasi 96,67%.

3.	Identifikasi Diagnosis Perubahan Hasil Perawatan Kulit Menggunakan Metode *Learning Vector Quantization* (LVQ).	Kulit	*Learning Vector Quantization* (LVQ)	Pengujian ini mendapatkan hasil dari parameter terbaik dengan maksimum iterasi sebanyak 300 dengan akurasi 98.25%.
4.	Implementasi Pengolahan Citra dan *Learning Vector Quantization* (LVQ) dalam Penentuan Golongan Kendaraan.	Golongan Kendaraan	Pengolahan Citra dan *Learning Vector Quantization* (LVQ)	Semakin banyak citra latih maka semakin tinggi akurasi yang diperoleh, serta nilai parameter juga dapat mempengaruhi nilai akurasi hingga mendekati 100%.

2.2 Kredit

Kredit adalah kemampuan untuk melaksanakan suatu pembelian atau mengadakan suatu pinjaman dengan suatu janji, pembayaran akan dilaksanakan pada jangka waktu yang telah disepakati (Astiko, 1996). Untuk memudahkan pihak bank dalam memilih debitur maka bank akan menilai beberapa kriteria dari calon nasabahnya. Beberapa kriteria yang sering dipakai oleh bank adalah prinsip 5C (5C *Prinsiple*). Kriteria tersebut dapat dilihat sebbagai berikut:

1. *Character*

Merupakan data kepribadian atau watak dari calon debitur untuk mengetahui tingkat tanggung jawab dan kemauan dalam melaksanakan kewajibannya membayar sesuai kesepakatan dan jangka waktu yang telah ditentukan.

2. *Condition*

Merupakan kondisi ekononmi yang dialami calon debitur saat ini serta prospek usaha yang dijalankan yang digunakan kreditur untuk mempertimbangkan apakah calon debitur tersebut layak untuk diberi kredit atau tidak.

3. *Capital*

Merupakan modal atau harta kekayaan yang dimiliki dari usaha yang sedang dikeloloanya. Modal ini bisa dilihat dari neraca, ratio keuntungan yang diperoleh seperti return on equity, return on investment.

4. *Capacity*

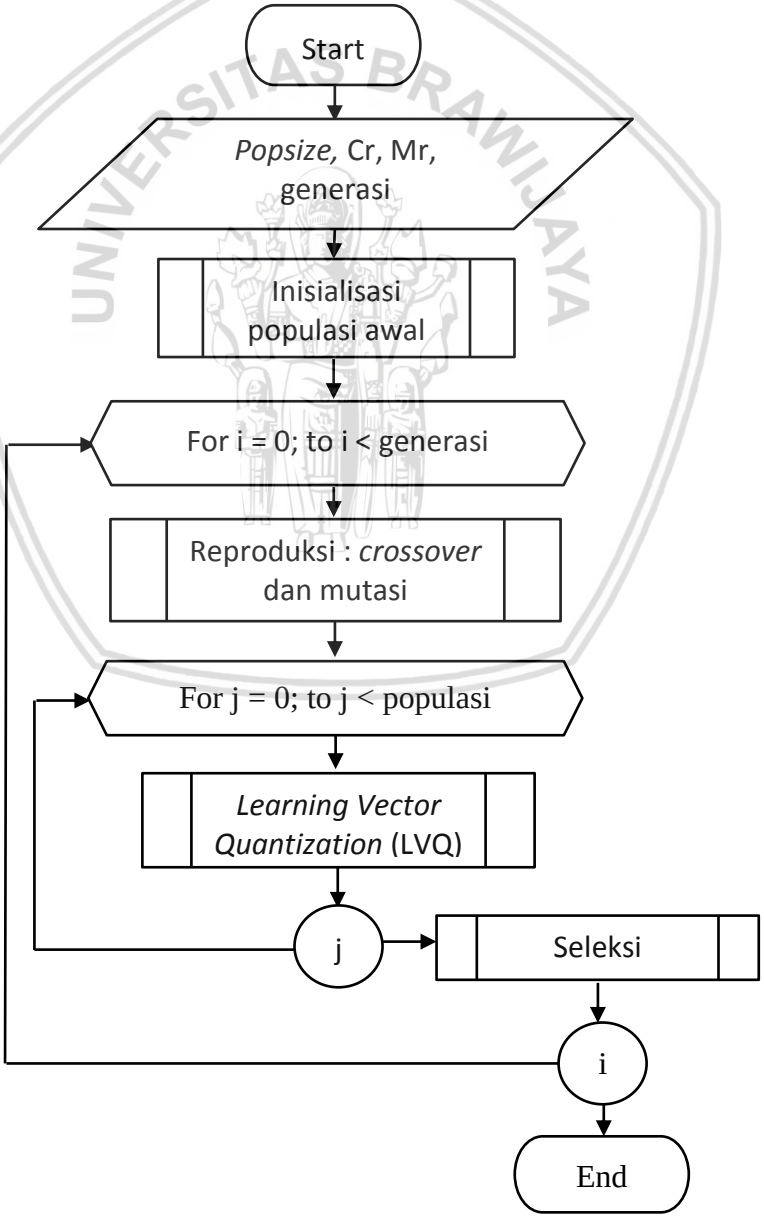
Merupakan penilaian terhadap calon debitur tentang kemampuannya dalam melunasi kewajiban-kewajibannya dari usaha yang dilakukan yang akan dibiayai dengan kredit dari bank.

5. *Collateral*

Merupakan jaminan yang dimiliki oleh calon debitur yang dapat disita apabila calon debitur tersebut tidak melakukan kewajibannya dalam pembayaran.

2.3 Algoritma genetika

Algoritma genetika adalah tipe Algoritma Evolusi yang paling populer dan berkembang pesat karena kemampuannya yang mampu menyelesaikan berbagai masalah yang kompleks. Tahapan penyelesaian masalah pada algoritma genetika dapat dilihat pada siklus berikut (Mahmudy, 2015).



Gambar 2.1 Siklus algoritma genetika

2.3.1 Inisialisasi populasi awal

Inisialisasi populasi awal merupakan proses pembangkitan nilai gen secara *random* yang direpresentasikan kedalam kromosom tiap individu. Inisialisasi dilakukan dengan menentukan jumlah populasi (*popsize*). Teknik pengkodean yang akan digunakan dalam pembangkitan inisialisasasi populasi awal yaitu dengan menggunakan pengkodean *real* atau nilai gen yang akan dibangkitkan berupa bilangan *real* dalam interval [-1; 1] (Sutojo, T dkk., 2011). Gambar 2.2 merupakan representasi pembangkitan populasi awal dengan popsize sejumlah 3 dengan menggunakan representasi kromosom bilangan *real* dengan panjang kromosom sebanyak 14.

Individu	K1	K2	K3	K4	K5	K6	K7	K8	K9	K10	K11	K12	K13	K14
P1	0.1	0.2	0.3	0.4	0.5	-0.6	-0.7	0.1	0.2	0.3	0.4	0.5	-0.6	-0.7
P2	0.2	0.3	0.4	0.5	0.6	-0.7	-0.1	0.2	0.3	0.4	0.5	0.6	-0.7	-0.1
P3	0.3	0.4	0.5	0.6	0.7	-0.1	-0.2	0.3	0.4	0.5	0.6	0.7	-0.1	-0.2

Gambar 2.2 Representasi Populasi Awal

Keterangan kolom:

Individu : Kolom untuk urutan individu
K1 : Kolom untuk nilai bobot atribut *Character* pada kelas diterima.
K2 : Kolom untuk nilai bobot atribut *Condition* pada kelas diterima.
K3 : Kolom untuk nilai bobot atribut *Capital* pada kelas diterima.
K4 : Kolom untuk nilai bobot atribut *Capacity* pada kelas diterima.
K5 : Kolom untuk nilai bobot atribut *Collateral* pada kelas diterima.
K6 : Kolom untuk nilai bobot atribut Usia pada kelas diterima.
K7 : Kolom untuk nilai bobot atribut Tanggungan pada kelas diterima.
K8 : Kolom untuk nilai bobot atribut *Character* pada kelas ditolak.
K9 : Kolom untuk nilai bobot atribut *Condition* pada kelas ditolak.
K10 : Kolom untuk nilai bobot atribut *Capital* pada kelas ditolak.
K11 : Kolom untuk nilai bobot atribut *Capacity* pada kelas ditolak.
K12 : Kolom untuk nilai bobot atribut *Collateral* pada kelas ditolak.
K13 : Kolom untuk nilai bobot atribut Usia pada kelas ditolak.
K14 : Kolom untuk nilai bobot atribut Tanggungan pada kelas ditolak.

2.3.2 Reproduksi

Reproduksi merupakan proses yang dilakukan untuk menghasilkan suatu keturunan dari setiap individu pada populasi (Kusumadewi, 2003). Proses reproduksi dilakukan bdengan cara dua tahap yaitu tahap *crossover* dan *mutasi*.

Proses reproduksi dari masing-masing tahap yang menghasilkan keturunan atau *child* akan ditempatkan dalam satu penampungan yang disebut *offspring*.

2.3.2.1 *Crossover*

Crossover merupakan proses pindah silang yang dilakukan dengan cara memilih dua induk secara acak untuk menghasilkan keturunan. Cara yang dilakukan untuk mendapatkan jumlah keturunan yang diinginkan yaitu dengan menentukan nilai *crossover rate (cr)* x *popsize*. Metode yang dapat digunakan dalam mengimplementasikan operator *crossover* antara lain, *one-cut-point crossover*, *two-cut-point crossover*, *Extended intermediate crossover,* dan *order crossover.* Metode *crossover* yang diterapkan pada penelitian ini yaitu menggunakan *two-cut-point crossover* atau memilih dua titikl potong secara acak pada tiap dua induk yang terpilih. Gambar 2.3 merupakan contoh pengilustrasian proses *two-cut-point crossover*.

Individu	Gen													
P1	0.1	0.2	0.3	0.4	0.5	-0.6	-0.7	0.1	0.2	0.3	0.4	0.5	-0.6	-0.7
P2	0.2	0.3	0.4	0.5	0.6	-0.7	-0.1	0.2	0.3	0.4	0.5	0.6	-0.7	-0.1

Gambar 2.3 Ilustrasi Proses *two-cut-point crossover*

Individu	Gen													
C1	0.1	0.2	0.4	0.5	0.5	-0.6	-0.7	0.1	0.2	0.4	0.5	0.5	-0.6	-0.7
C2	0.2	0.3	0.3	0.4	0.6	-0.7	-0.1	0.2	0.3	0.3	0.4	0.6	-0.7	-0.1

Gambar 2.4 Ilustrasi hasil proses *two-cut-point crossover*

2.3.2.2 Mutasi

Mutasi merupakan operator dari proses reproduksi dengan memilih satu induk secara acak untuk dapat menghasilkan satu keturunan. Keturunan yang akan dihasilkan pada proses mutasi yaitu berdasarkan penentuan *mutation rate* (*mr*) x *popsize*. Metode mutasi yang dapat diterapkan dalam mengimplementasikan operator mutasi antara lain, *random mutation*, *reciprocal randommutation*, *shift mutation,* dan *insertion mutation*. Metode mutasi yang akan digunakan dalam penelitian ini yaitu menggunakan *random mutation*. *Random mutation* merupakan metode yang dilakukan dengan cara memilih titik potong secara acak yang kemudian dilakukan proses perhitungan menggunakan persamaan berikut:

$$x'_i = x'_i + r(max_i - min_j) \quad (2\text{-}1)$$

Keterangan:

x'_i = nilai gen terpilih

r = nilai *random* pada range misalkan [-0,1, 0,1]

Individu	Gen													
P1	0.1	0.2	0.3	0.4	0.5	-0.6	-0.7	0.1	0.2	0.3	0.4	0.5	-0.6	-0.7

Gambar 2.5 Ilustrasi proses *random* mutasi

Individu	Gen													
C1	0.1	0.2	0.36	0.4	0.5	-0.6	-0.7	0.1	0.2	0.33	0.4	0.5	-0.6	-0.7

Gambar 2.6 Ilustrasi hasil proses *random* mutasi

2.3.3 Evaluasi

Evaluasi merupakan untuk menghitung nilai kebugaran (*fitness*) pada setiap individu (Mahmudy, 2015). Tujuan yang ingin dicapai pada proses algoritma genetika yaitu dapat memaksimalkan nilai *fitness* (Sutojo, T dkk., 2011). Pada penelitian ini, persamaan yang digunakan untuk mendapatkan nilai *fitness* yaitu:

$$Fitness = \frac{Jumlah\ data\ sesuai}{Total\ semua\ data\ uji} \text{ x } 100 \qquad (2\text{-}2)$$

2.3.4 Seleksi

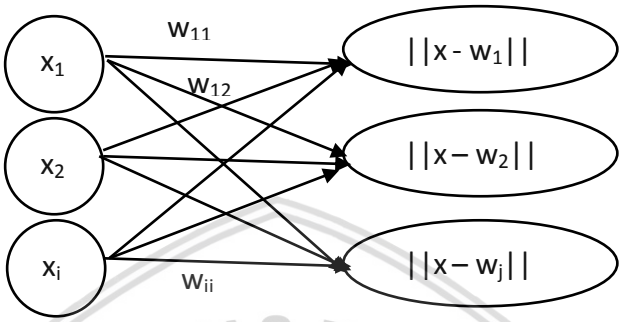
Seleksi merupakan proses pemilihan individu terbaik dari suatu generasi yang berisi *parent* dan *child* (hasil *crossover* dan mutasi) yang memiliki nilai *fitness* tinggi. Seleksi yang digunakan dalam penelitian ini adalah *Elitism Selection*, yaitu mengambil individu sejumlah populasi awal dengan *fitness* tertinggi. Hasil dari seleksi ini akan digunakan untuk menjadi *parent* pada generasi berikutnya.

2.4 Jaringan saraf tiruan

Jaringan saraf tiruan (JST) atau sering juga disebut sebagai *Artificial Neural Network* (ANN), adalah sebuah teknik dalam bidang kecerdasan buatan yang dimodelkan berdasarkan system saraf manusia. JST merupakan sistem adaptif yang dapat mengubah strukturnya untuk memecahkan masalah berdasarkan informasi eksternal maupun internal yang mengalir melalui jaringan tersebut. Oleh karena sifatnya yang adaptif, JST juga sering disebut dengan jaringan adaptif.

2.4.1 *Learning Vector Quantization*

Learning Vector Quantization **(LVQ)** adalah salah satu metode klasifikasi yang ada pada jaringan saraf tiruan (JST) dimana setiap unit outputnya mempresentasikan dari tiap-tiap kelas. Tujuan dari algoritma ini adalah untuk mendekati distribusi kelas vektor untuk meminimalkan kesalahan dalam pengklasifikasian (Dwi, 2011). Gambaran lebih lanjut mengenai LVQ dapa dilihat pada Gambar 2.7.



Gambar 2.7 Model algoritma LVQ

Langkah-langkah yang dilakukan dalam proses *Learning Vector Quantization* adalah sebagai berikut:

Masukkan bobot dari kromosom hasil algoritma genetika yang telah diseleksi dengan fitness terbaik untuk masukan awal.

1. Hitung jarak data terhadap masing-masing kelas.

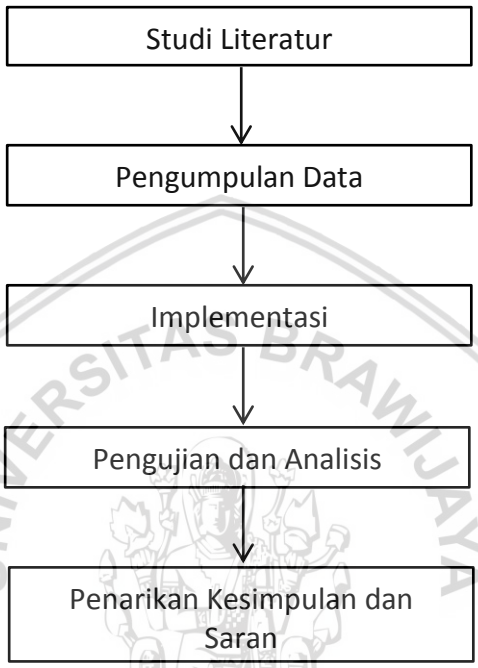
$$D(j) = \sqrt{\sum_{i=1}^{n} \left(x_i - w_{ij}\right)^2} \quad (2\text{-}3)$$

2. Menentukan jarak terdekat data terhadap kelas pada seluruh data di D(j).
3. Menghitung akurasi:

$$\text{Akurasi} \frac{jumlah\ kelas\ yang\ sesuai}{jumlah\ data} \quad (2\text{-}4)$$

BAB 3 METODOLOGI

Pada bab ini menjelaskan tentang metode yang digunakan serta langkah-langkah yang dilakukan untuk melakukan Implementasi Algoritma Genetika dan LVQ pada Penentuan Kelayakan Kredit Bank X. Alur metode penelitian dapat dilihat pada gambar 3.1 berikut:



Gambar 3.1 Alur metode penelitian

3.1 Studi literatur

Pada tahapan ini merupakan proses mencari informasi tambahan yang digunakan sebagai acuan dalam penelitian. Informasi yang dibutuhkan pada penelitian ini disusun berdasarkan referensi yang diperoleh dari artikel, buku, jurnal, serta dokumentasi projek baik nasional maupun internasional. Studi literatur digunakan sebagai pedoman pengetahuan dasar sebelum melakukan analisis, perancangan, implementasi dan pengujian dalam tahap-tahap penelitian. Berikut merupakan dasar teori yang dibutuhkan sebagai pendukung penelitian :

- Kredit
- Algoritma Genetika
- *Learning Vector Quantization* (LVQ)

3.2 Pengumpulan data

Data yang digunakan dalam penelitian ini adalah data debitur Bank X. Data tersebut berupa nilai 5C (*Character, Condition, Capital, Capacity, Collateral, Condition*), usia, tanggungan dari nasabah tahun 2012. Dari data latih tersebut dapat digunakan dalam Implementasi Algoritma Genetika untuk Optimasi LVQ pada Penentuan Kelayakan Kredit Studi Kasus Bank X.

3.3 Implementasi

Tahap ini menjelaskan tentang implementasi dari perancangan yang telah dibuat. Tahap implementasi meliputi:

1. Pembuatan User Interface.
2. Melakukan penentuan bobot optimal tiap atribut menggunakan Algoritma Genetika.
3. Melakukan perhitungan menentukan kelayakan penerimaan kredit menggunakan metode LVQ dengan atribut yang sudah terpilih.
4. Mendapatkan hasil berupa rekomendasi untuk kelayakan pemberian kredit

3.4 Pengujian dan analisis

3.4.1 Pengujian

Pengujian yang dilakukan meliputi pengujian populasi, pengujian nilai *cr* dan nilai *mr*, pengujian jumlah generasi, pengujian hasil akurasi optimasi bobot optimal, serta menguji hasil akurasi metode *Learning Vector Quantization* (LVQ) tanpa bobot optimal hasil optimasi algoritma genetika.

3.4.2 Analisis

Analisis dilakukan untuk menentukan nilai akurasi dari perbandingan data hasil pengolahan perhitungan manual dengan perhitungan yang dilakukan oleh sistem untuk Implementasi Algoritma Genetika dan LVQ pada Penentuan Kelayakan Kredit studi kasus Bank X.

3.5 Penarikan kesimpulan dan saran

Penarikan kesimpulan dapat dilakukan setelah analisis hasil dari pengujian dengan metode yang diterapkan. Tahap berikutnya yang merupakan tahap terakhir dari penulisan ini adalah pemberian saran yang ditujukan untuk memperbaiki kesalahan yang terjadi serta sebagai bâhan pertimbangan atas penelitian metode selanjutnya.

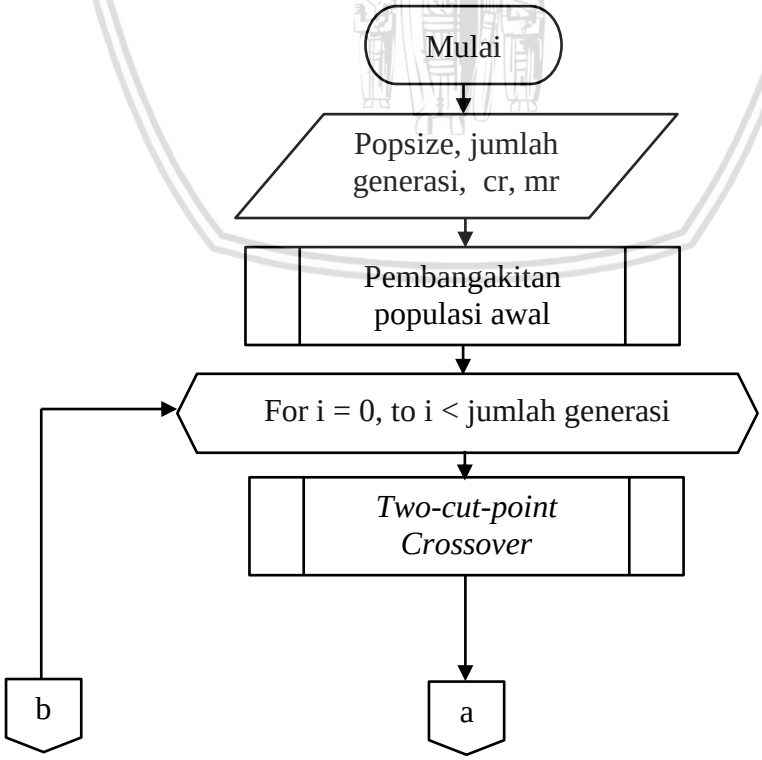
BAB 4 PERANCANGAN

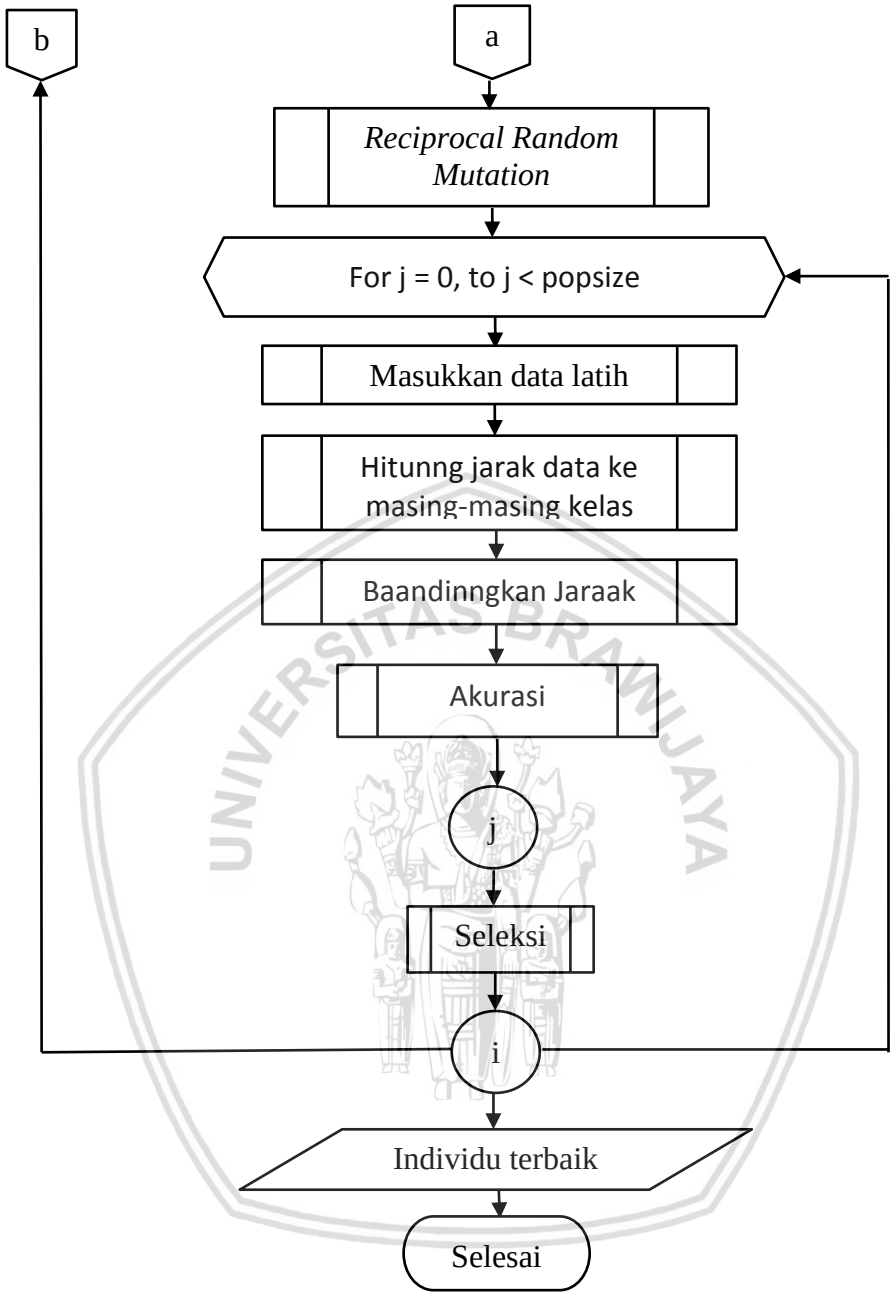
4.1 Formulasi permasalahan

Salah satu solusi yang dapat diselesaikan dalam permasalahan analisa kelayakan kredit yaitu dengan cara mengoptimasi nilai bobot optimal pada metode *Learning Vector Quantization* (LVQ) dengan menggunakan algoritma genetika. Algoritma genetika digunakan untuk mendapatkan nilai bobot optimal untuk metode *Learning Vector Quantization* (LVQ), sehingga dari poroses tersebut dapat memeberikan hasil yang optimal.

Data yang digunakan dalam penelitian ini berjumlah 63 data calon debitur yang diperoleh dari Bank X Lamongan tahun 2012. Pada data analisa kelayakan kredit tersebut memiliki tujuh kriteria untuk menentukan layak tidaknya seorang calon debitur dalam menerima kredit diantaranya *character*, *condition*, *capacity*, *capital*, *collateral*, usia, dan jumlah tanggungan.

Pada penelitian ini, untuk mencari nilai bobot optimal pada *Learning Vector Quantization* (LVQ) menggunakan algoritma genetika, dibutuhkan nilai parameter algoritma genetika yang digunakan untuk mencari nilai *fitness* yang mendekati optimal. Parameter algoritma genetika yang digunakan antara lain adalah ukuran populasi (*popsize*), nilai *mutation rate* (mr), nilai *crossover rate* (cr), dan jumlah generasi. Individu yang memiliki nilai *fitness* yang mendekati optimal akan terpilih sebagai nilai bobot yang akan diproses pada tahap LVQ. Siklus penyelesaian masalah pada kasus optimasi nilai bobot optimal LVQ menggunakan algoritma genetika ditunjukkan oleh Gambar 4.1.





Gambar 4.1 Siklus penyelesaian masalah optimasi bobot optimal LVQ dengan menggunakan algoritma genetika pada penentuan kelayakan kredit

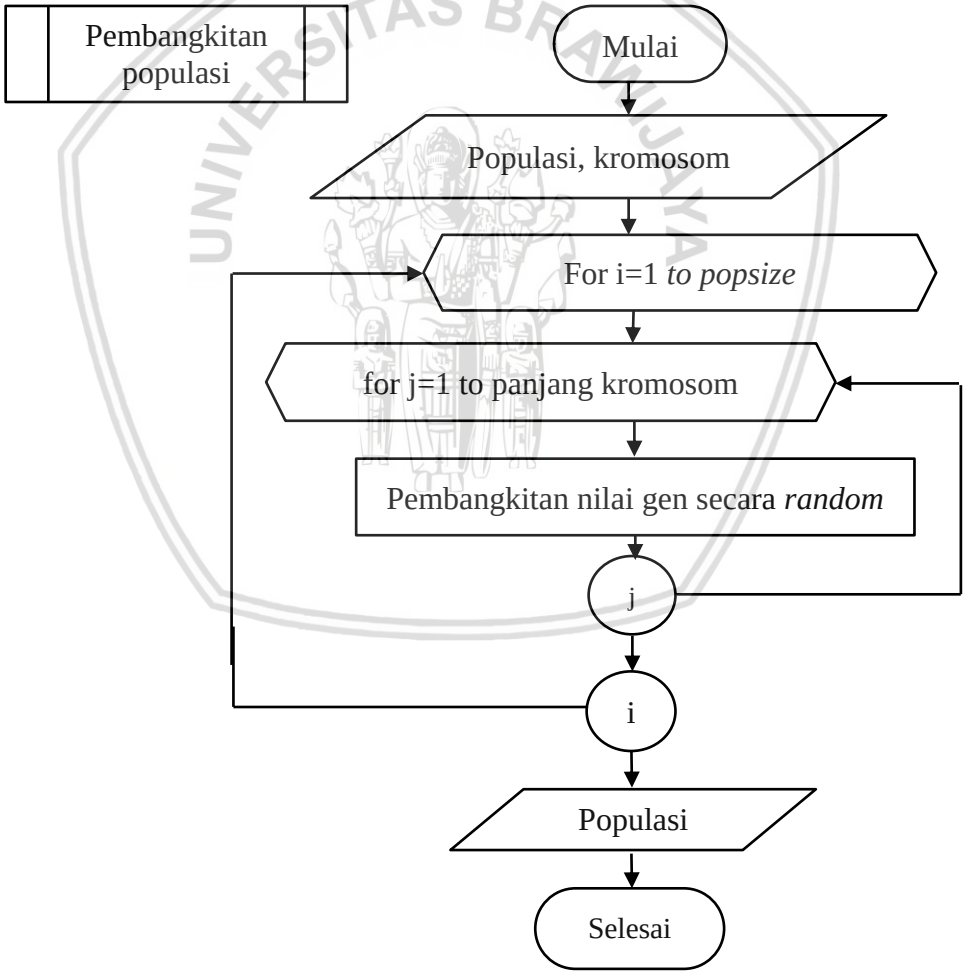
4.2 Penyelesaian malasah dengan algoritma genetika dan LVQ

Algoritma genetika banyak digunakan dalam optimasi berbagai permasalahan juga dalam penentuan bobot awal pada metode jaringan saraf tiruan. Alasan penggunaan algoritma genetika pada penelitian tersebut karena algoritma genetika tidak memerlukan perhitungan matematis yang rumit seperti yang diperlukan metode optimasi lain, *Differensial Evolution* misalnya.

Langkah awal yang digunakan untuk melakukan optimasi dengan algoritma genetika adalah dengan menentukan parameter-parameter yang terdapat pada algoritma genetika seperti *popsize*, nilai *cr*, nilai *mr*, dan jumlah generasi, jumlah kromosom. Langkah berikutnya adalah mengacak individu sejunlah *popsize,* atau sering disebut juga dengan inisialisasi populasi awal. Langkah berikutnya adalah melakukan reproduksi dengan menggunakan *two-cut-point crossover* dan *random mutation.* Setelah melakukan reproduksi maka masing-masing kromosom akan digunakan sebagai bobot optimal pada LVQ, kemudian mencari akurasi dari setiap individu untuk dijadikan *fitness.*

4.2.1 Pembangkitan populasi

Pembangkitan populasi digunakan untuk mendapatkan individu-indivdu yang nantinya akan dijadikan solusi bobot optimal pada metode *Learning Vector Quantization* (LVQ) sebanyak popsize dengan panjang kromosom sebanyak 7 gen. Langkah untuk membangkitkan populasi awal dapat dilihat pada Gambar 4.2.



Gambar 4.2 Diagram alur pembangkitan populasi awal

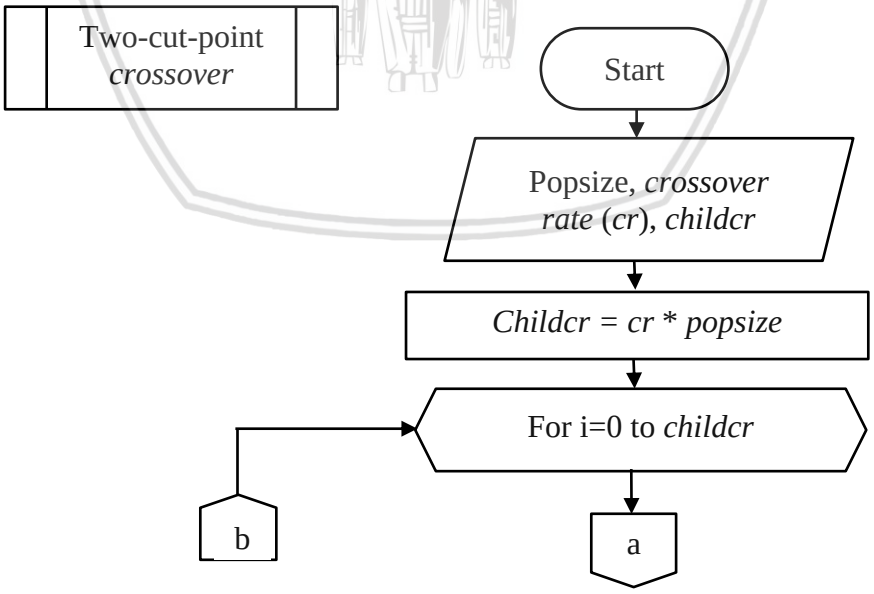
Penjelasan proses pembangkitan populasi awal pada gambar 4.3 dapat dijabarkan sebagai berikut:

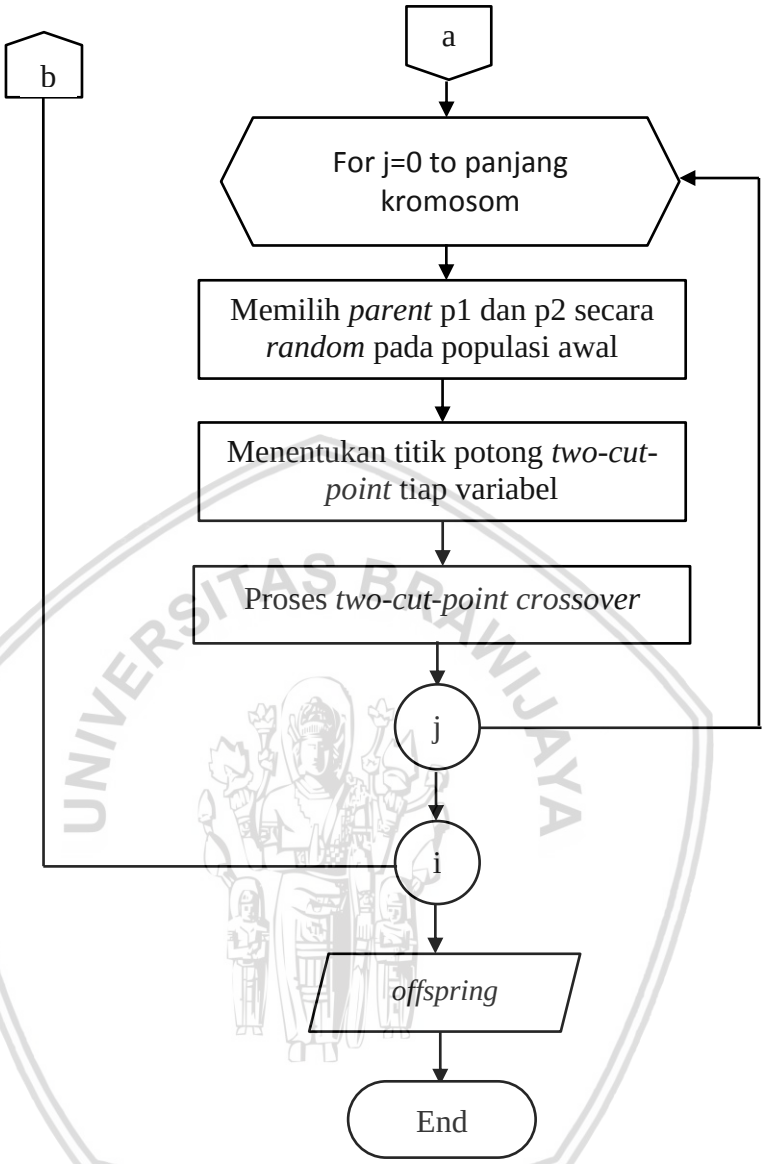
1. Masukkan *popsize* dan panjang kromosom.
2. Melakukan perulangan terhadap fungsi panjang kromosom, jika fungsi yang dijalankan belum mencapai panjang kromosom yang telah ditentukan maka, proses akan terus berulang sampai memenuhi syarat.
3. Melakukan perulangan sebanyak *popsize*, jika perulangan yang dilakukan belum mencapai *popsize* yang telah ditentukan, maka proses akan terus berulang sampai memenuhi syarat.
4. Membentuk kromosom dengan panjang 14 gen secara *random* berdasarkan *popsize* yang telah ditentukan.

Pengkodean yang akan digunakan menentukan kromosom pada popiulasi awal adalah pengkodean *real*. Pengkodean tersebut dipilih karena akan digunakan dalam menentukan bobot LVQ yang merupakan bilangan desimal dengan rentang nilai antara [-1, 1].

4.2.2 Reproduksi

Reproduksi adalah proses yang terdapat pada algoritma genetika yang digunakan untuk mendapatkan *child* atau *offspring* dari proses *crossover* dan *mutasi*. Metode crossover yang akan digunakan adalah *two-cut-pint crossover* dengan rumus *offspring* yaitu *cr x popsize,* variabel *cr* (*crossover rate*) yang akan digunakan memiliki rentang nilai antara [0 9]. Proses *crossover* ditunjukkan pada gambar 4.3.



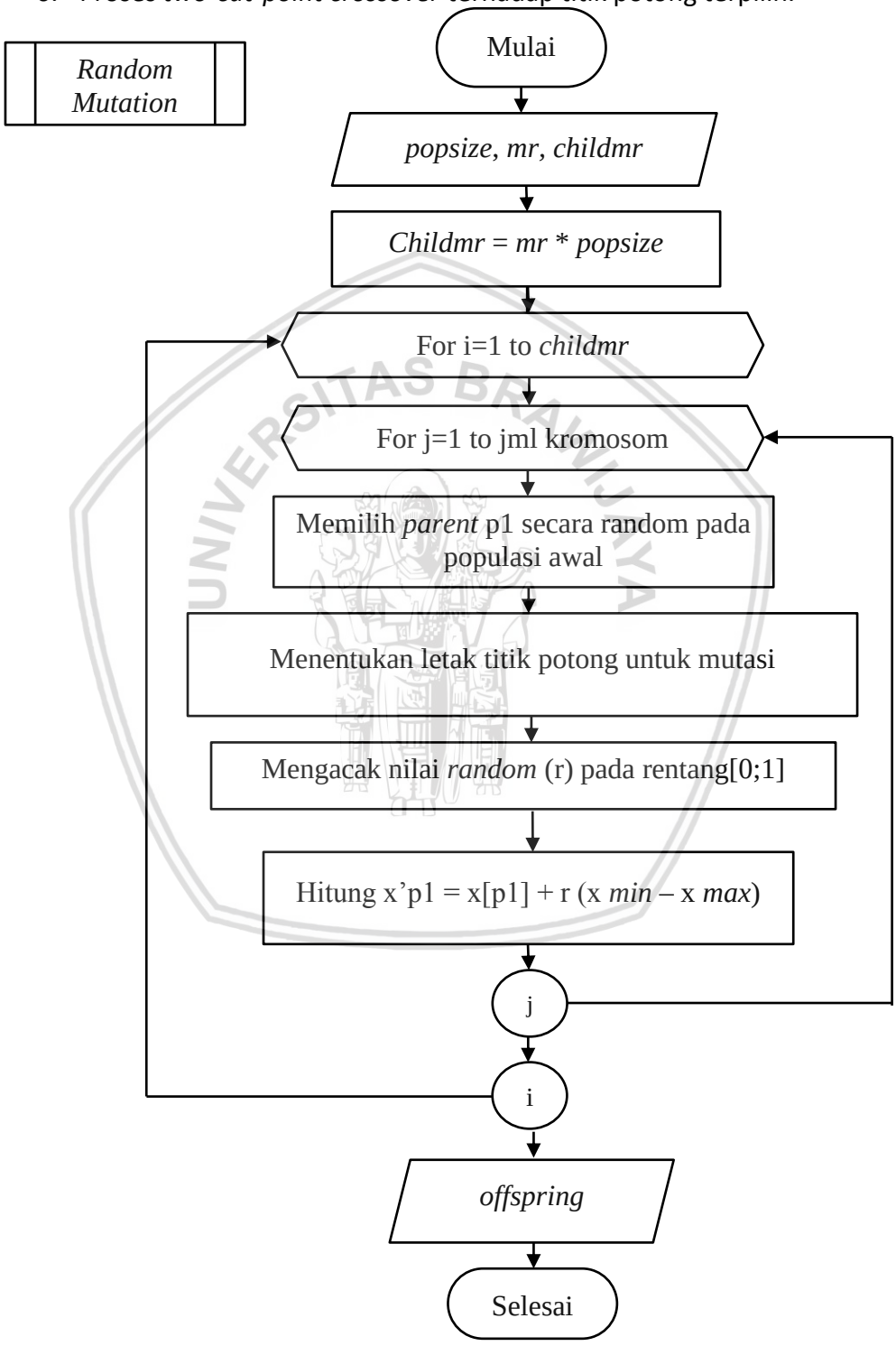


Gambar 4.3 Diagram alur *two-cut-point crossover*

Penjelasan secara rinci proses *one-cut-point crossover* dijelaskan sebagai berikut:

1. Masukkan *popsize*, *crossover rate* (*cr*), dan deklarasi *childcr*.
2. Menentukan nilai *childcr* dengan mengalikan nilai *cr* dan *popsize*.
3. Melakukan proses perulangan terhadap jumlah *childcr* yang telah ditentukan, jika fungsi yang dijalankan belum mencapai *childcr* yang ditentukan maka, proses akan terus berulang sampai memenuhi syarat.

4. Melakukan proses perulangan terhadap jumlah panjang kromosom tiap variabel yang digunakan untuk memilih titik potong yang akan dilakukan proses *one-cut-point*.
5. Memilih 2 *parent* secara acak p1 dan p2 pada populasi awal.
6. Proses *two-cut-point crossover* terhadap titik potong terpilih.



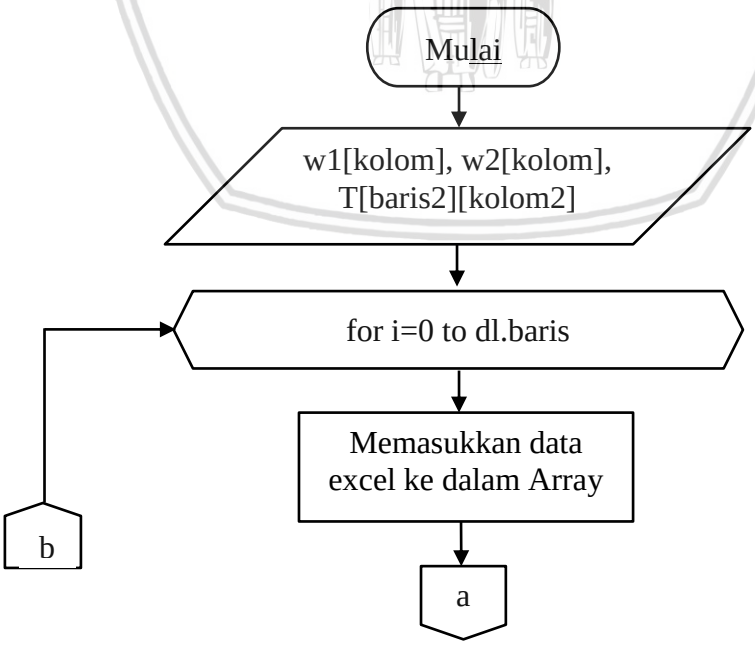
Gambar 4.4 Diagram alur proses *mutation*

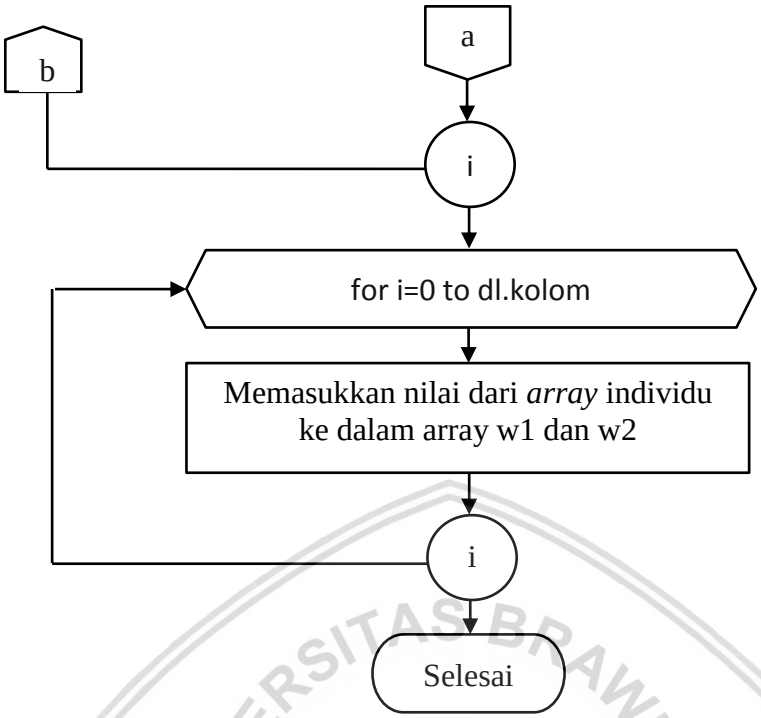
Keterangan dari alur flowchart *mutation* pada gambar 4.4 dijelaskan sebagai berikut:

1. Masukkan nilai *popsize*, mr, dan deklarasi *childmr*.
2. Hitung nilai *childmr* dengan mengalikan nilai *mr* dan *popsize*.
3. Melakukan proses perulangan terhadap fungsi jumlah *childmr* yang akan dihasilkan pada proses mutasi, jika fungsi yang dijalankan belum mencapai jumlah *childmr* yang ditentukan maka, proses akan terus berulang sampai memenuhi syarat.
4. Melakukan proses perulangan terhadap fungsi jumlah *kromosom* yang akan dilakukan proses pemilihan letak titik potong kemudian dilakukan proses mutasi, jika fungsi yang dijalankan belum mencapai jumlah *offspring* yang ditentukan maka, proses akan terus berulang sampai memenuhi syarat.
5. Memilih 1 *parent* secara acak untuk melakukan reproduksi menggunakan metode *random mutation*.
6. Mengacak nilai *random* (*r*) pada rentang nilai [0;1].
7. Melakukan proses *random mutation* pada gen terpilih dengan rumus x'p1 = x'[p1] + r (x min – x max).

4.2.3 Memasukkan bobot

Memasukkan bobot dilakukan pada setiap generasi. Proses ini dilakukan untuk memisah bobot kelas pertama dan kelas kedua kedalam dua *array* yang berbeda yang nantinya akan diproses lebih lanjut menggunakan metode *Learning Vector Quantization*(LVQ).



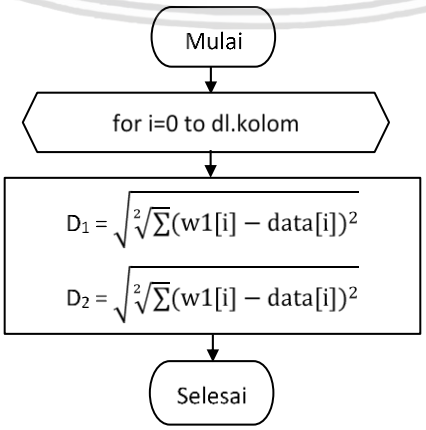


Gambar 4.5 Diagram alur memasukkan bobot

Langkah-langkah memasukkan bobot ditunjukkan pada Gambar 4.5:

1. Inisialisasi dan deklarasi variabel T[ ] untuk menyimpan target kelas dari pakar, w1[ ], dan w2[ ] untuk menyimpan bobot dari masing-masing kelas.
2. Perulangan sebanyak baris(jumlah data) untuk menyimpan target tiap data ke dalam *array.*
3. Perulangan sebanyak jumlah kolom(kriteria) untuk memisahkan bobot sesuai kelasnya.

4.2.4 Hitung jarak data



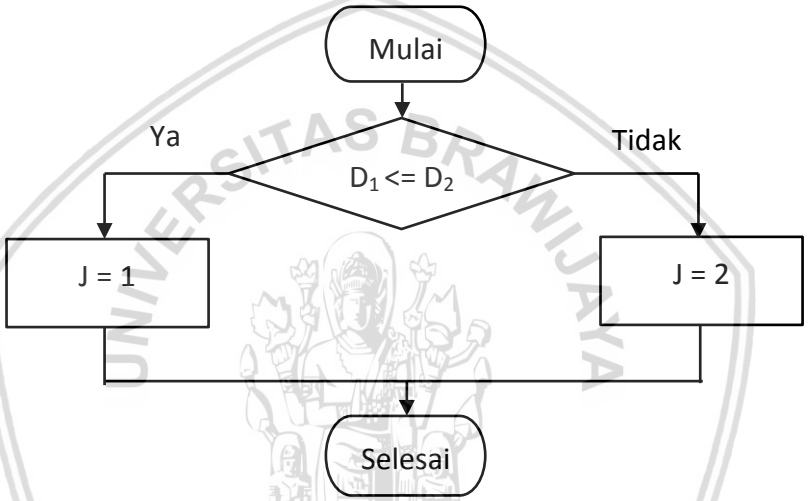
Gambar 4.6 Diagram alur menghitung jarak data

Hitung jarak data dilakukan pada setiap data. Proses ini dilakukan untuk menghitung jarak tiiap data terhadap masing-masing kelas. Langkah-langkah menghitung jarak data pada Gambar 4.6 dijelaskan sebagai berikut:

1. Perulangan sebanyak kolom (jumlah kriteria).
2. Perhitungan pada data terhadap masing-masing bobot kelas, sehingga didapatkan jarak data terhadap masing-masing kelas.

4.2.5 Membandingkan jarak

Langkah yang dilakukan setelah menghitung jarak terhadap masing-masing kelas adalah membandingkan jarak data yang telah didapatkan dari langkah sebelumnya.



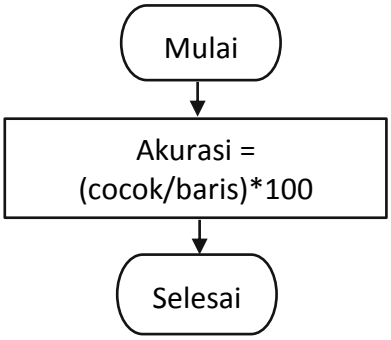
Gambar 4.7 Diagram alur membandingkan jarak

Langkah pembandingan jarak dapat dilihat pada gambar 4.7 dijelaskan sebagai berikut:

1. Kondisi jika nilai D1 lebih kecil atau samadengan nilai D2 maka kelas prediksi data akan memilih kelas pertama.
2. Kondisi jika nilai D1 lebih besar dari nilai D2 maka kelas prediksi data akan memilih kelas kedua.

4.2.6 Akurasi

Menghitung nilai akurasi dilakukan pada setiap individu bobot yang terpilih pada satu generasi. Untuk mencari nilai *fitness* dilakukan dengan cara membandingkan hasil keputusan dari pakar dengan hasil yang diperoleh sistem.



Gambar 4.8 Diagram alur menghitung akurasi

Langkah-langkah mencari nilai akurasi ditunjukkan pada Gambar 4.8. Proses menghitung akurasi dengan cara membandingkan jumlah data yang memiliki hasil kelas prediksi sama dengan kelas target dari pakar dibagi jumlah keseluruhan data.

4.3 Perhitungan manual

Pada penelitian ini proses pengolahan data calon debitur menggunakan algoritma genetika dan *Learning Vector Quantization* (LVQ) akan diselesaikan melalui langkah-langkah yang telah dijelaskan pada sub bab sebelumnya. Sebagai contoh menggunakan data dengan 7 variabel untuk perhitungan algoritma genetika ditunjukkan pada Tabel 4.1.

Tabel 4.1 Contoh permasalahan perhitungan algoritma genetika

Kriteria	*Popsize*	Nilai cr	Nilai mr	Jumlah generasi
Karakter	5	0.7	0.3	1
Modal	5	0.7	0.3	1
Kapasitas	5	0.7	0.3	1
Kondisi	5	0.7	0.3	1
Jaminan	5	0.7	0.3	1
Usia	5	0.7	0.3	1
Tanggungan	5	0.7	0.3	1

4.3.1 Pembangkitan populasi awal

Pembangkitan populasi awal adalah proses pembangkitan populasi awal yang disesuaikan dengan *popsize* yang telah ditentukan sebelumnya. Pembentukan representasi kromosom setiap individu dilakukan menggunakan pengkodean desimal yang dilakukan secara *random* menggunakan bilangan *real* pada rentang -1 hingga 1. Contoh pembangkitan populasi awal akan ditunjukkan pada Tabel 4.2.

Tabel 4.2 Pembangkitan populasi awal

K1	K2	K3	K4	K5	K6	K7	K8	K9	K10	K11	K12	K13	K14
0.1	0.2	0.3	0.4	0.5	-0.6	-0.7	0.1	0.2	0.3	0.4	0.5	-0.6	-0.7
0.2	0.3	0.4	0.5	0.6	-0.7	-0.1	0.2	0.3	0.4	0.5	0.6	-0.7	-0.1
0.3	0.4	0.5	0.6	0.7	-0.1	-0.2	0.3	0.4	0.5	0.6	0.7	-0.1	-0.2
0.4	0.5	0.6	0.7	0.1	-0.2	-0.3	0.4	0.5	0.6	0.7	0.1	-0.2	-0.3
0.5	0.6	0.7	0.1	0.2	-0.3	-0.4	0.5	0.6	0.7	0.1	0.2	-0.3	-0.4

4.3.2 Reproduksi

Reproduksi adalah proses pada algoritma genetika yang digunakan untuk mendapatkan *child* dari proses *crossover* dan *mutasi*. Reproduksi dilakukan setelah melakukan tahap pembangkitan populasi awal, selanjutnya reproduksi dilakukan dengan menggunakan metode *two-cut-pint crossover* dan *random mutation* untuk menghasilkan *child*. Untuk proses crossover, rumus yang digunakan yaitu *cr x popsize*. Sedangkan untuk mutasi menggunakan metode *random mutation* dengan menggunakan rumus *offspring* yaitu *mr* x *popsize*.

4.3.3 Pembentukan bobot

Pembentukan bobot diperoleh dari nilai kromosom tiap individu pada *populasi, child crossover*, dan *child mutasi*. Setelah diperoleh nilai kromosom tiap individu untuk bobot kelas diterima, selanjutnya membentuk bobot untuk kelas ditolak berdasarkan pembangkitan populasi awal kelas ditolak. Proses pembentukan bobot untuk kelas ditolak sama dengan proses pembentukan bobot untuk kelas diterima. Pembentukan bobot untuk kelas ditolak dapat dilihat pada Tabel 4.3.

Tabel 4.3 Representasi bobot

K1	**K2**	**K3**	**K4**	**K5**	**K6**	**K7**	**K8**	**K9**	**K10**	**K11**	**K12**	**K13**	**K14**
0,1	0,2	0,3	0,4	0,5	-0,6	-0,7	0,4	-0,3	0,2	-0,1	-0,7	0,6	0,5

4.3.4 Menentukan jarak data terhadap masing-masing kelas

Langkah awal pada metode *Learning Vector Quantization* (LVQ) setelah mendapatkan bobot adalah mencari jarak tiap data terhadap masing-masing kelas. Perhitungn jarak terhadap masing-masing kelas ditunjukkan pada Tabel 4.4.

Tabel 4.4 Representasi perhitungan jarak terhadap masing-masing kelas

X_{ij}	W_1	W_2	$X_{ij}W_1$	$X_{ij}W_2$
15	0.1	0.4	222,01	213,16
353,55	0.2	-0.3	124856,22	125209,82
8,36	0.3	0.2	64,96	66,59
100,00	0.4	-0.1	16,79	21,14
4,50	0.5	-0.7	9900,25	10140,49
37	-0.6	0.6	1413,76	1324,96
3	-0.7	0.5	13,69	6,25
Jarak data terhadap kelas			369,44	370,11

4.3.5 Membandingkan jarak

Langkah yang dilakukan setelah menghitung jarak terhadap masing-masing kelas adalah membandingkan jarak data yang telah didapatkan dari langkah perhitungan sebelumnya. Representasi pembandingan jarak dapat dilihat pada Gambar 4.9.

Jarak kelas-1	Jarak kelas-2	Kelas Prediksi
369,44	370,11	1

Gambar 4.9 Representasi langkah membandingkan jarak

4.3.6 Menghitung akurasi

Langkah terakhir pada proses *Learning Vector Quantization* (LVQ) adalah menghitung akurasi dari bobot yang terpilih dengan cara membandingkan jumlah data yang hasil perkiraan kelasnya cocok terhadap target target, dengan jumlah keseluruhan data. Representasi perhitungan akurasi dapat dilihat pada Tabel 4.5.

Tabel 4.5 Tabel perhitungan akurasi

Target	Hasil	Cocok
1	1	1
2	1	0
2	2	1
2	2	1
1	2	0
Akurasi(%):		60

4.4 Perancangan pengujian dan evaluasi

Perancangan pengujian adalah tahap untuk menguji semua parameter pada algoritma genetika dan *learning vector quantization* dalam permasalahan penentuan kelayakan kredit diantaranya yaitu jumlah generasi, jumlah *popsize*, *crossover rate*, *mutation rate, alpha, decerease alpha,* dan *minimum alpha*.

Tahapan pada proses pengujian yang dilakukan dalam mengevaluasi sistem rekomendasi penentuan kelayakan kredit yang dibuat yaitu :

1. Perancangan Pengujian *Popsize*

Pengujian pada *popsize* digunakan untuk mencari nilai *popsize* yang optimal. Semakin besar nilai *popsize* yang digunakan, maka peluang untuk menemukan hasil yang optimal juga semakin besar. Pengujian ini akan dilakukan sebanyak 6 kali dengan nilai *popsize* dimulai dari 10 kemudian akan bertambah sejumlah 2 hingga mencapi jumlah *popsize* 20, setelah mencapai jumlah *popsize* 20, maka *popsize* akan bertambah 20 hingga mencapai 100. Jumlah generasi yang akan digunakan pada pengujian ini sebanyak 2 generasi. Untuk nilai *cr* ditentukan sebesar 0,5 dan nilai *mr* sebesar 0,5. Dari hasil pengujian *popsize* nantinya akan dipilih nilai rata-rata *fitness* tertinggi yang kemudian akan dijadikan acuan untuk perancangan pengujian pada tahap selanjutnya. Perancangan uji coba ukuran populasi akan ditunjukkan pada Tabel 4.6.

Tabel 4.6 Perancangan pengujian jumlah populasi

Popsize	**Cr = 0,5**		**Mr = 0,5**	**Generasi = 2**		**Rata-rata nilai *fitness***
	Percobaan ke-i					
	1	**2**	**3**	**4**	**5**	
10						
12						
14						
16						
18						
20						

2. Perancangan Pengujian Kombinasi Nilai *Cr* dan *Mr*

Pengujian yang akan dilaukan pada kombinasi nilai *cr* dan *mr* digunakan untuk mengetahui kombinasi yang cocok yang dapat memberikan nilai rata-rata *fitness* tertinggi. Pengujian akan dilakukan sebanyak 10 kali dengan rentang nilai antara [0; 1]. Ukuran populasi yang akan digunakan berdasarkan hasil dari pengujian ukuran populasi yang memiliki nilai paling optimal. Perancangan pengujian kombinasi nilai *crossover rate* (cr) dan *mutation rate* (mr) ditunjukkan pada tabel 4.7.

Tabel 4.7 Perancangan pengujian kombinasi *Cr* dan *Mr*

Kombinasi		*Popsize* =			Generasi =		Rata-rata nilai *fitness*
		Percobaan ke-i					
Cr	*Mr*	1	2	3	4	5	
0,9	0,1						
0,8	0,2						
0,7	0,3						
0,6	0,4						
0,5	0,5						
0,4	0,6						
0,3	0,7						
0,2	0,8						
0,1	0,9						

3. Perancangan Pengujian Jumlah Generasi

Perancangan pengujian selanjutnya dilakukan terhadap jumlah generasi yang terbentuk pada algoritma genetika. Pengujian tersebut dilakukan untuk menentukan jumlah generasi yang dianggap optimal yang nantinya akan digunakan untuk menentukan bobot optimal pada metode *Learning Vector Quantization*(LVQ). Jumlah generasi yang akan diuji yaitu memiliki rentang nilai antara [1; 20] generasi dimulai dari jumlah generasi 1 sampai 5, kemudian bertambah dengan kelipatan 5 hingga mencapai jumlah generasi sebanyak 20, dengan menggunakan hasil dari ukuran *popsize*, *cr*, dan *mr* pada penelitian sebelumnya. Perancangan pengujian parameter algoritma genetika untuk menentukan jumlah generasi terbaik ditunjukkan pada Tabel 4.8.

Tabel 4.8 Perancangan pengujian jumlah generasi

Jumlah Generasi	*Popsize* =		*Cr* =	*Mr* =		Rata-rata nilai *fitness*
	Percobaan ke-i					
	1	2	3	4	5	
1						
2						
3						
4						
5						
10						
15						
20						

4. Perancangan Pengujian Akurasi Metode LVQ tanpa Optimasi

Perancangan pengujian keempat ini dilakukan untuk menguji nilai akurasi metode *Learning Vector Quantization*(LVQ) tanpa menggunakan bobot optimal hasil dari optimasi bobot menggunakan metode algoritma genetika. Perancangan pengujian akurasi metode LVQ tanpa optimasi dapat dilihat pada Tabel 4.9.

Tabel 4.9 Perancangan pengujian akurasi metode LVQ tanpa optimasi

Data debitur ke – i	Target (Pakar)	Target (Sistem)	Kecocokan
1			
2			
3			
4			
5			
. .	...	...	...
61			
62			
63			
Cocok			...
Akurasi			...

5. Perancangan pengujian optimasi bobot optimal pada LVQ dengan algoritma genetika

Perancangan pengujian kelima merupakan pengujian yang dilakukan terhadap metode *Learning Vector Quantization* (LVQ) dengan melakukan optimasi bobot menggunakan algoritma genetika untuk mendapatkan bobot optimal. Perancangan pengujian ini dilakukan untuk melihat kemampuan metode optimasi bobot optimal menggunakan algoritma genetika. Apakah dengan melakukan optimasi bobot dapat memberikan bobot optimal yang lebih tepat. Perancangan pengujian optimasi bobot optimal pada LVQ dengan algoritma genetika dapat dilihat pada Tabel 4.10.

Tabel 4.10 Perancangan pengujian optimasi bobot optimal pada LVQ dengan algoritma genetika

Data debitur ke - i	Target (Pakar)	Target (Sistem)	Kecocokan
1			
2			
3			
4			
5			
. .	...	...	...
61			
62			
63			
Cocok			...
Akurasi			...

BAB 5 IMPLEMENTASI

5.1 Struktur program

Program yang digunakan dalam penelitian ini menggunakan bahasa pemrograman java dengan menggunakan *tools* NetBeans. Struktur program yang dibentuk pada proses pembangunan sistem untuk mengoptimasi bobot metode *Learning Vector Quantization*(LVQ) menggunakan algoritma genetika diantaranya adalah:

1. *Class* Alev yang dibangun dengan membentuk *method-method* dan *source code* implementasi proses algoritma genetika sampai dengan melakukan tahap optimasi bobot optimal metode *Learning Vector Quantization*(LVQ) pada permasalahan penentuan kelayakan kredit.
2. *Class* LVQ berisikan *method-method* yang dibangun untuk mengimplementasikan perhitungan pada metode Learning Vector Quantization dengan masukkan hasil bobot optimal dari class Alev dan menghitung hasil akurasi tiap bobot optimal yang digunakan.

5.2 Potongan implementasi program

Setelah melakukan tahap implementasi pada permasalahan yang terjadi kedalam suatu program, maka diperoleh hasil dari proses tersebut yaitu berupa sistem pendukung keputusan kelayakan kredit dengan menerapkan proses algoritma genetika dan *Learning Vector Quantization* (LVQ) diantaranya yaitu pembentukan populasi awal, proses reproduksi *crossover two-cut-point* dan *random mutation*, proses memasukkan bobot, proses menentukan jarak data terhadap masing-masing kelas, tahap membandingkan jarak, perubahan nilai bobot, penghitungan akurasi, dan proses seleksi. Dari hasil implementasi tersebut selanjutnya akan dijelaskan potongan Source Code tiap tahapan optimasi bobot optimal *Learning Vector Quantization* (LVQ) penentuan kelayakan kredit yang telah dibuat.

5.2.1 Implementasi pembangkitan populasi awal

Pembangkitan populasi awal dilakukan dengan rentang [-1, 1] dan panjang kromosom sebanyak dua kali jumlah kriteria. Source code pembangkitan populasi awal ditentukan berdasarkan masukan jumlah populasi (popsize) untuk mengetahui jumlah individu yang akan dibentuk. Potongan implementasi code pembentukan inisialisasi populasi awal ditunjukkan pada *Source Code* 5.1.

```
1  public void PembangkitanPopulasi(){
2          System.out.println("\n======================");
3          System.out.println("    POPULASI AWAL");
4          System.out.println("======================");
5          dl.Inisialisasi1();
```

```
6           System.out.println(dl.baris);
7           kolom = 2*dl.kolom;
8           popcr = (int)(Math.round(popsize*cr));
9           popmr = (int)(Math.round(popsize*mr));
10          jmlindividu = popsize + popcr +popmr;
11          Populasi = new double[popsize][kolom];
12          individu = new double[jmlindividu][kolom];
13          fitness = new double[jmlindividu];
14          indexfitness = new int[jmlindividu];
15          for (int i = 0; i < popsize; i++) {
16              for (int j = 0; j < kolom; j++) {
17                  Populasi[i][j] = ((2*Math.random()-1));
18              }
19          }
20
21          for (int i = 0; i < popsize; i++) {
22              System.out.print("Parent ke-"+(i+1)+":\t");
23              for (int j = 0; j < kolom; j++) {
24                  individu[i][j] = Populasi[i][j];
25                  System.out.print(df.format(Populasi[i][j])+"\t");
26              }System.out.println("");
27          }
28      }
```

Source Code 5.1 Implementasi pembangkitan populasi

Penjelasan *source code* proses pembentukan populasi awal:

- 6 – 9 : Inisialisasi variabel untuk panjang kolom array, jumlah child hasil *crossover* dan mutasi, serta jumlah individu yang dihasilkan.
- 10 – 13: Inisialisasi array untuk menampung populasi awal, semua individu, serta *fitness* tiap generasi.
- 14 – 18: Perulangan untuk mengisi array populasi dengan bobot dengan *range* nilai -1 hingga 1.
- 20 – 26: Perulangan untuk mencetak populasi awa yang terbentuk.

5.2.2 Implementasi proses reproduksi *Crossover Two-cut-point*

Pada tahap reproduksi, salah satu proses yang akan dilakukan pada penelitian ini yaitu menggunakan *Two-Cut-Point Crossover*. Pada proses ini, teknik yang dilakukan yaitu memilih dua induk dan dua titik potong secara acak untuk dilakukan proses pindah silang pada gen yang memiliki index diantara dua titik potong yang terpilih. Dari proses ini akan menghasilkan jumlah *child* atau *offspring* yang terbentuk berdasarkan *crossover rate* (*cr*) dan *popsize* yang telah

dimasukkan. proses implementasi *Two-Cut-Point Crossover* ditunjukkan pada *Source Code* 5.2.

```
1   public void Crossover(){
2            System.out.println("\n======================");
3            System.out.println("      CROSSOVER");
4            System.out.println("======================");
5
6            System.out.println("\nInduk Terpilih:");
7            System.out.println("===============");
8           double tukar, parent [][]= new double[2][kolom];
9           int index, counter=0;
10          int loop = (int)(Math.round((popcr+1)/2));
11          childcr = new double [popcr][kolom];
12
13          //Menentukan titik potong CR
14          int temp;
15          point1 = (int)(7*Math.random()+0);
16          point2 = (int)(7*Math.random()+0);
17           if (point1>point2) {
18              temp = point1;
19              point1 = point2;
20              point2 = temp;
21           }
22
23           for (int h = 0; h < loop; h++) {
24           //Proses untuk child dengan jumlah genap
25           if (popcr%2==0) {
26               for (int i = 0; i < 2; i++) {
27                           index=(int) (popsize*Math.random()+0);
28                           System.out.print("Index-"+(index+1)+": ");
29                       for (int j = 0; j < dl.kolom; j++) {
30                           parent[i][j] = Populasi[index][j];
31
32  System.out.print(df.format(parent[i][j])+"\t");
33                       } System.out.println("");
34                   }
35
36              //System.out.println("\n");
37                  for (int j = 0; j < kolom; j++) {
38                      if (j>=point1 && j<=point2) {
39                          tukar = parent[0][j];
```

```
                        parent[0][j] = parent[1][j];
                        parent[1][j] = tukar;
                    }
                    childcr[counter+0][j] = parent[0][j];
                    childcr[counter+1][j] = parent[1][j];
                }
                counter+=2;
                //System.out.println("counter"+counter);

//                for (int i = 0; i < popcr; i++) {
//                    for (int j = 0; j < dl.kolom; j++) {
//
System.out.print(df.format(childcr[i][j])+"\t");
//                    }System.out.println("");
//            }
        //Proses untuk child dengan jumlah ganjil
        } else if (popcr%2==1) {
                for (int i = 0; i < 2; i++) {
                        index=(int) (popsize*Math.random()+0);
                        System.out.print("Index-"+(index+1)+": ");
                    for (int j = 0; j < kolom; j++) {
                        parent[i][j] = Populasi[index][j];

System.out.print(df.format(parent[i][j])+"\t");
                    } System.out.println("");
                }

            //System.out.println("\n");
                for (int j = 0; j < kolom; j++) {
                    if  ((j>=point1  &&  j<=point2)||(j>=(point1+7)
&& j<=(point2+7))) {
                        tukar = parent[0][j];
                        parent[0][j] = parent[1][j];
                        parent[1][j] = tukar;
                    }
                    childcr[counter+0][j] = parent[0][j];
                    if (h<(loop-1)) {
                    childcr[counter+1][j] = parent[1][j];
                    }

                }
                counter+=2;
```

```
80          }
81
82              for (int i = 0; i < popcr; i++) {
83                  for (int j = 0; j < kolom; j++) {
84                      individu[i+popsize][j] = childcr[i][j];
85                  }
86              }
87          }
88
89
90          System.out.println("Loop = "+loop);
91          System.out.println("Popcr= "+popcr);
92          System.out.println("Point1 = "+point1);
93          System.out.println("Point2 = "+point2);
94      }
```

Source Code 5.2 Implementasi reproduksi *crossover two-cut-point*

Penjelasan *source code* Proses reproduksi *crossover two-cut-point.*

- 9 – 11 : Deklarasi variabel index untuk memilih individu yang akan dijadikan *parent*, serta inisialiasasi jumlah perulangan yang dibutuhkan serta array untuk menampung *child*.
- 14 – 21: Menentukan titik potong awal dan titik potong akhir.
- 25 – 53: Proses crossover jika *child* yang dihasilkan berjumlah genap.
- 54 – 80: Proses *crossover* jika *child* yang dihasilkan berjumlah ganjil.
- 82 – 87: Memasukkan *child* kedalam *array* kumpulan individu.

5.2.3 Implementasi proses reproduksi *Random Mutation*

Proses reproduksi yang kedua adalah mutasi yang menggunakan metode *random mutation*. Proses ini dengan cara memilih satu induk secara acak, kemudian memilih satu pointer secara acak pada masing-masing variabel untuk dilakukan proses *random mutation*. Nilai bilangan *random* yang akan digunakan untuk melakukan proses *random mutation* yaitu antara [-0.1, 0.1]. Kemudian pada proses *random mutation* ini menghasilkan satu *child* atau *offspring* yang terbentuk dari tiap *parent* yang terpilih dan sebanyak jumlah *mutation rate (mr)* dan *popsize*. Proses implementasi *random mutation* ditunjukkan pada *Source Code* 5.3.

```
1       public void Mutasi(){
2           System.out.println("\n======================");
3           System.out.println("        MUTASI");
4           System.out.println("======================");
5           int index=0;
```

```
6           double r, parent[][]=new double[popmr][kolom];
7           childmr = new double [popmr][kolom];
8           point1 = (int)(7*Math.random()+0);
9           r = (double)(2*Math.random()+(-1));
10          System.out.println("Nilai R = "+df.format(r));
11          System.out.println("Point1 = "+point1);
12          double min = 10000, min2 = 10000;
13          double max = 0, max2 = 0;
14
15          //Mencari nilai min dan max pada gen yang terpilih pada
16  index Poin1
17          for (int i = 0; i < popsize; i++) {
18              for (int j = 0; j < kolom; j++) {
19                  if (j==point1) {
20                      if (Populasi[i][j]<min) {
21                          min = Populasi[i][j];
22                      }
23                      if (Populasi[i][j]>max) {
24                          max = Populasi[i][j];
25                      }
26                  }
27                  if (j==point1+7) {
28                      if (Populasi[i][j]<min2) {
29                          min2 = Populasi[i][j];
30                      }
31                      if (Populasi[i][j]>max2) {
32                          max2 = Populasi[i][j];
33                      }
34                  }
35              }
36          }
37          System.out.println("Min = "+df.format(min));
38          System.out.println("Max = "+df.format(max));
39          System.out.println("Min2 = "+df.format(min2));
40          System.out.println("Max2 = "+df.format(max2));
41
42          //Proses Mutasi Gen
43          for (int i = 0; i < popmr; i++) {
44              index=(int) (popsize*Math.random()+0);
45              System.out.print("Index-"+(index+1)+": ");
46              for (int j = 0; j < kolom; j++) {
```

```
47                  //x'i = x'i + r (maxi - minj)
48                  parent[i][j]=Populasi[index][j];
49                  if (j==point1) {
50                      parent[i][j] = parent[i][j]+r*(max-min);
51                  }else if (j==point1+7) {
52                      parent[i][j] = parent[i][j]+r*(max2-min2);
53                  }
54                  childmr[i][j] = parent[i][j];
55                  individu[i+popsize+popcr][j] = childmr[i][j];
56                  System.out.print(df.format(childmr[i][j])+"\t");
57
58              }System.out.println("");
59          }
60      }
```

Source Code 5.3 Implementasi reproduksi *random mutation*

- 5 – 13 : Deklarasi dan inisialisasi variabel untuk nilai random, *array* untuk menampung *parent* dan *child* hasil mutasi, serta nilai minimum dan maksimum untuk menentukan batas titik potong.
- 17 – 35: Mencari titik potong awal dan akir pada index *array*.
- 43 – 53: Proses mutasi dengan mengubah nilai yang ada di dalam batasan dengan rumus *random* mutasi.
- 54 – 55: Memasukkan hasil mutasi ke dalam *array* kumpulan individu.

5.2.4 Implementasi proses memasukkan bobot

Proses ini digunakan untuk memasukkan bobot kelas pertama dan kedua ke dalam dua *array* yang berbeda yang nantinya akan di proses menggunakan metode *Learning Vector Quantization*(LVQ). Proses implementasi *random mutation* ditunjukkan pada tabel *Source Code* 5.4.

```
1       public void Bobotawal(Alev a, int baris) {
2           kolom = dl.kolom * 2;
3           dl.Inisialisasi1();
4           dl.Inisialisasi2();
5           w1 = new double[dl.kolom];
6           w2 = new double[dl.kolom];
7           T = new int[dl.baris2][dl.kolom2];
8           for (int i = 0; i < dl.baris; i++) {
9               T[i][0] = dl.target[i][0];
10          }
11
12          for (int i = 0; i < dl.kolom; i++) {
```

```
13              w1[i] = a.individu[baris][i];
14              w2[i] = a.individu[baris][i + 7];
15          }
16      }
```

Source Code 5.4 Implementasi memasukkan bobot

- 2 – 7 : Inisialisasi panjang *array*, data, deklarasi *array* bobot kelas pertama, kedua, dan *array* untuk menyimpan nilai target dari tiap data.
- 8 – 10 : Memasukkan data target ke dalam *array*.
- 12 – 15: Proses memasukkan nilai bobot dari *array* individu ke dalam *array* w1 dan w2.

5.2.5 Implementasi proses menentukan jarak

Proses ini digunakan untuk menentukan jarak data terhadap kelas pertama dan kedua. Dari proses ini akan dihasilkan jarak data terhadap masing-masing kelas yang nantinya akan dibandingkan nilainya. Proses implementasi menentukan jarak ditunjukkan pada *Source Code* 5.5.

```
1   public void HitungJarak(int baris) {
2           D1=0; D2=0;
3
4           for (int i = 0; i < 7; i++) {
5               D1 += Math.pow((w1[i] - dl.latih[baris][i]), 2);
6               D2 += Math.pow((w2[i] - dl.latih[baris][i]), 2);
7           }
8           D1 = Math.sqrt(D1);
9           D2 = Math.sqrt(D2);
10          System.out.println("Baris: " + baris + "\tD1 = " +
     df.format(D1) + "\tD2 = " + df.format(D2));
11      }
```

Source Code 5.5 Implementasu hitung jarak data terhadap kelas

- 2 : Deklarasi variabel untuk jarak kelas pertama dan kedua.
- 4 – 7 : Perulangan untuk menghitung jarak berdasarkan bobot dan data.
- 8 – 9 : Mengakar hasil perhitungan jarak dari perulangan.
- 10 : Cetak hasil jarak yang telah diakar.

5.2.6 Implementasi proses membandingkan jarak

Proses ini digunakan untuk meandingkan jarak data terhadap kelas pertama dan kedua yang telah didapatkan dari proses sebelumnya. Dari proses ini akan dihasilkan jarak terkecil dari data terhadap masing-masing kelas yang nantinya akan menentukan di kelas mana data tersebut akan dikelasifikasikan. Proses implementasi membandingkan jarak ditunjukkan pada *Source Code* 5.6.

```
1     public void Compare(int baris) {
2         if (D1 < D2) {
3             J = 1;
4         } else if (D2 <= D1) {
5             J = 2;
6         }
7         System.out.println("J: " + J +"\tT: "+T[baris][0]);
8     }
```

Source Code 5.6 Implementasi membandingkan jarak data terhadap masing-masing kelas

- 1 – 6 : Proses pembandingan nilai variabel D1 dan D2, nilai yang terkecil akan menjadi nilai jarak yang terdekat.
- 7 : Cetak kelas hasil prediksi berdasarkan jarak terdekat yang terpilih.

5.2.7 Implementasi proses menghitung akurasi

Proses ini digunakan untuk menentukan akurasi atau jumlah kecocokan hasil perbandingan kelas target data dengan kelas prediksi sistem, dibagi jumlah data. Dari proses ini akan dihasilkan *fitness* tiap bobot optimal yang nantinya akan digunakan pada proses seleksi. Proses implementasi menghitung akurasi ditunjukkan pada tabel *Source Code* 5.7.

```
1     public void Akurasi(LVQ lvq, int individu){
2         akurasi = (double)lvq.cocok/dl.baris*100;
3         fitness[individu] = akurasi;
4         System.out.println("Fitness = "+df.format(akurasi));
5     }
```

Source Code 5.7 Implementasi menghitung akurasi

- 2 : Menghitung akurasi dengan membagi variabel cocok dengan jumlah data.
- 3 : Memasukkan nilai akurasi ke dalam array *fitness* dengan index sesuai urutan individu.

5.2.8 Implementasi proses seleksi

Proses ini digunakan untuk menyeleksi populasi awal dan *child* dari semua prose reproduksi berdasarkan nilai akurasi yang diurutkan dari yang tertinggi. Dari proses ini akan dihasilkan individu bobot optimal yang telah terseleksi. Proses implementasi seleksi ditunjukkan pada *Source Code* 5.8.

```
1     public void Seleksi(){
2         System.out.println("==================");
3         System.out.println("     SELEKSI");
4         System.out.println("==================");
```

```
5           int pos = 0, temp = 0;
6           double max=0;
7
8           for (int i = 0; i < jmlindividu; i++) {
9               indexfitness[i] = i;
10          }
11          for (int i = 0; i < jmlindividu; i++) {
12              for (int j = i; j < jmlindividu; j++) {
13                  if (fitness[j] > max) {
14                      max = fitness[j];
15                      temp = indexfitness[j];
16                      pos = j;
17                  }
18              }
19              fitness[pos] = fitness[i];
20              indexfitness[pos] = indexfitness[i];
21              fitness[i] = max;
22              indexfitness[i] = temp;
23              max=0;
24          }
25
26          for (int i = 0; i < jmlindividu; i++) {
27              System.out.print(df.format(fitness[i])+"\t");
28          }
29          System.out.println("");
30          for (int i = 0; i < jmlindividu; i++) {
31              System.out.print(indexfitness[i]+"\t");
32          }
33
34          for (int i = 0; i < popsize; i++) {
35              for (int j = 0; j < kolom; j++) {
36                  Populasi[i][j] = individu[indexfitness[i]][j];
37                  individu[i][j] = Populasi[i][j];
38              }
39          }
40      }
```

Source Code 5.8 Implementasi proses seleksi

- 5 – 6 : Inisialisasi variabel untuk posisi, temporari, dan *maximum fitness.*
- 8 – 10 : Mengisi nilay *array* indexfitness.

- 11 – 24 : Proses pengurutan individu sesuai *fitness*.
- 34 – 39 : Proses pembaruan pada populasi awal setelah individu dengan *fitness* tinggi terpilih.

BAB 6 PENGUJIAN DAN ANALISIS

Tahap pengujian terhadap optimasi bobot optimal *Learning Vector Quantization* (LVQ) menggunakan algoritma genetika yang akan dilakukan sesuai dengan perancangan pengujian yang telah dijelaskan pada subbab 4.4. Terdapat 5 skenario pengujian, yaitu pengujian *popsize*, pengujian kombinasi nilai *Cr* dan *Mr*, pengujian jumlah generasi, pengujian akurasi bobot optimal *Learning Vector Quantization*(LVQ) menggunakan algoritma genetika, dan pengujian hasil akurasi *Learning Vector Quantization*(LVQ) tanpa optimasi bobot optimal menggunakan algoritma genetika.

6.1 Hasil pengujian dan analisis variabel *popsize*

Hasil dan analisis pengujian yang pertama dilakukan pada *popsize* untuk mencari ukuran populasi terbaik yang mempunyai nilai *fitness* paling optimal. Ukuran populasi terbaik akan dijadikan sebagai parameter untuk pengujian selanjutnya. Ukuran populasi yang digunakan sebanyak 5 dimulai dari jumlah *popsize ebanyak* 10 kemudian bertambah 2 hingga mencapi jumlah *popsize* 20, setelah mencapai jumlah *popsize* 20, maka *popsize* akan bertambah sebanyak 20 hingga mencapai jumlah 100. Setiap *popsize* melakukan pengujian sebanyak 5 kali. Kemudian nilai *fitness* tiap percobaan akan dihitung untuk mencari rerata *fitness* dari setiap *popsize*. Pada pengujian *popsize,* jumlah generasi yang digunakan adalah 2, kombinasi nilai *Cr* = 0,5 dan *Mr* = 0,5. Hasil pengujian *popsize* yang telah dilakukan, dapat dilihat pada Tabel 6.1.

Tabel 6.1 Hasil pengujian variabel *popsize*

Popsize	Cr=0,5		Mr =0,5	Generasi=2		Rata-rata nilai *fitness*
	Percobaan ke-i					
	1	2	3	4	5	
10	93,65	90,48	88,89	92,06	88,89	90,79
12	92,06	88,89	90,48	92,06	93,65	91,43
14	92,06	88,89	92,06	92,06	93,65	91,74
16	93,65	93,65	93,65	90,48	92,06	92,70
18	92,06	92,06	92,06	93,65	93,65	92,70
20	93,65	93,65	93,65	93,65	93,65	93,65
40	93,65	93,65	93,65	93,65	93,65	93,65
60	93,65	93,65	93,65	93,65	93,65	93,65
80	93,65	93,65	93,65	93,65	93,65	93,65
100	93,65	93,65	93,65	93,65	93,65	93,65

Pada hasil pengujian variabel *popsize* dapat kita lihat bahwa rata-rata akurasi semakin bertambah seiring bertambahnya *popsize*. Akurasi tertinggi didapatkan saat *popsize* mencapi jumlah 20 yaitu akurasi sebesar 93,65%, dan akurasi tidak berubah meskipun *popsize* ditambah.

6.2 Hasil pengujian dan analisis kombinasi variabel *CR* dan *MR*

Pengujian berikutnya dilakukan terhadap kombinasi nilai *Cr* dan *Mr*. Pengujian ini dilakukan untuk mencari kombinasi nilai yang sesuai dan dapat menghasilkan nilai *fitness* yang optimal. Pada pengujian kombinasi nilai *Cr* dan *Mr*, parameter algoritma genetika untuk *popsize* menggunakan 20 populasi, yang diperoleh dari *popsize* akurasi yang stabil di setiap pengujiannya pada pengujian *popsize* pada subbab 6.1 dan jumlah generasi sebanyak 2 generasi. Setiap kombinasi nilai *Cr* dan *Mr* dilakukan pengujian sebanyak 5 kali. Tabel 6.2 merupakan hasil uji coba kombinasi nilai *Cr* dan *Mr*.

Tabel 6.2 Hasil pengujian variabel *Cr* dan *Mr*

Cr & Mr	*Popsize* = 20; Generasi = 2; Perulangan ke – i					Nilai rata-rata akurasi (*fitness*)
	1	2	3	4	5	
0,9 ; 0,1	93,65	93,65	93,65	93,65	93,65	93,65
0,8 ; 0,2	93,65	92,06	93,65	93,65	93,65	93,33
0,7 ; 0,3	92,06	92,06	92,06	93,65	93,65	92,70
0,6 ; 0,4	92,06	93,65	92,06	93,65	92,06	92,70
0,5 ; 0,5	92,06	93,65	92,06	92,06	92,06	92,38
0,4 ; 0,6	93,65	92,06	88,89	93,65	92,06	92,06
0,3 ; 0,7	93,65	92,06	93,65	92,06	92,06	92,70
0,2 ; 0,8	92,06	92,06	92,06	93,65	93,65	92,70
0,1 ; 0,9	92,06	92,06	93,65	92,06	92,06	92,38

Pada hasil pengujian variabel *cr* dan *mr* rata-rata akurasi tertinggi didapatkan saat *cr* = 0,9 dan *mr* = 0,1 yaitu akurasi sebesar 93,65%. Sebagian besar pengujian mendapatkan nilai akurasi yang tidak optimal ketika nilai *mr* dinaikkan dan nilai *cr* diturunkan.

6.3 Hasil pengujian dan analisis jumlah generasi

Perancangan pengujian selanjutnya dilakukan terhadap jumlah generasi yang terbentuk pada algoritma genetika. Pengujian tersebut dilakukan untuk menentukan jumlah generasi yang dianggap optimal yang nantinya akan digunakan untuk menentukan bobot optimal pada metode *Learning Vector Quantization* (LVQ). Jumlah generasi yang akan diuji yaitu memiliki rentang nilai antara [1; 20] generasi dimulai dari jumlah generasi 1 hsampai 5, kemudian bertambah dengan kelipatan 5 hingga mencapai jumlah generasi sebanyak 20,

dengan menggunakan hasil dari ukuran *popsize*, *cr*, dan *mr* pada penelitian sebelumnya. Hasil skenario pengujian jumlah generasi ditunjukkan pada Tabel 6.3.

Tabel 6.3 Hasil pengujian variabel generasi

Generasi	Cr=0,9		Mr =0,1	*Popsize* = 20		Rata-rata nilai *fitness*
	Percobaan ke-i					
	1	2	3	4	5	
1	92,06	92,06	92,06	93,65	93,65	92,70
2	93,65	93,65	93,65	92,06	92,06	93,01
3	93,65	93,65	95,24	93,65	93,65	93,97
4	93,65	92,06	92,06	93,65	93,65	93,01
5	93,65	93,65	93,65	92,06	93,65	93,33
10	93,65	93,65	93,65	93,65	93,65	93,65
15	93,65	93,65	93,65	93,65	93,65	93,65
20	93,65	93,65	93,65	93,65	93,65	93,65

Pada hasil pengujian variabel generasi rata-rata akurasi tertinggi didapatkan saat generasi mencapi jumlah 3 yaitu akurasi sebesar 93,97%. Tetapi pada percobaan berikutnya rata-rata akurasi masih mengalami penurunan. Sehingga pengujian dilakukan hingga rata-rata akurasi tidak berubah nilai. Sehingga nilai generasi yang dianggap optimal berjumlah 10.

6.4 Hasil pengujian dan analisis akurasi metode LVQ tanpa optimasi

Pengujian keeempat adalah pengujian yang dilakukan terhadap metode *Learning Vector Quantization*(*LVQ*) dalam menentukan kelayakan kredit. Dari 63 data calon debitur akan diuji kedalam sistem *Learning Vector Quantization* (*LVQ*) dengan menggunakan bobot dari calon debitur sesuai kelasnya. Untuk menghitung nilai *fitness* yang berasal dari akurasi sistem didapatkan dengan cara membandingkan hasil target kelas yang diperoleh dari sistem dengan target kelas menurut keputusan pakar. Jika target kelas dari sistem dengan target kelas dari pakar memiliki kecocokan maka akan diberi nilai kecocokan 1(*true*). Sedangkan jika target kelas dari sistem yang dibandingkan dengan target kelas dari pakar tidak sesuai maka diberi nilai 0(false). Kemudian nilai kecocokan akan dijumlahkan dan dibagi dengan jumlah data dan dikalikan 100 seperti yang telah dijelaskan pada subbab 4.3.6 sebelumnya. Pengujian akurasi *Learning Vector Quantization* (LVQ) yang telah dilakukan, diperlihatkan pada Tabel 6.4.

Tabel 6.4 Tabel hasil pengujian metode LVQ tanpa optimasi

Data debitur ke - i	Target (Pakar)	Target (Sistem)	Kecocokan
1	1	1	1
2	2	1	0
3	2	1	0
4	1	1	1
5	1	1	1
6	1	1	1
7	1	2	0
8	1	1	1
9	1	1	1
10	1	2	0
11	1	1	1
12	1	2	0
13	1	1	1
14	1	1	1
15	1	1	1
16	1	1	1
17	1	1	1
18	1	1	1
19	1	1	1
20	1	1	1
21	1	1	1
22	1	1	1
23	1	1	1
24	1	1	1
25	1	1	1
26	2	1	0
27	1	1	1
28	1	1	1
29	1	1	1
30	1	1	1
31	1	1	1
32	1	2	0

33	1	1	1
34	1	1	1
35	1	2	0
36	1	1	1
37	1	1	1
38	1	2	0
39	1	1	1
40	1	1	1
41	1	1	1
42	1	1	1
43	1	1	1
44	1	1	1
45	1	1	1
46	1	2	0
47	1	1	1
48	1	1	1
49	1	1	1
50	1	1	1
51	2	2	1
52	1	1	1
53	2	2	1
54	2	2	1
55	2	1	0
56	1	1	1
57	1	1	1
58	1	2	0
59	1	1	1
60	1	2	0
61	1	1	1
62	1	1	1
63	1	1	1
Cocok			**50**
Akurasi			**79,37**

6.5 Hasil pengujian dan analisis optimasi bobot optimal pada LVQ dengan algoritma genetika

Pengujian kelima merupakan pengujian yang dilakukan terhadap metode *Learning Vector Quantization* (LVQ) dengan melakukan optimasi bobot menggunakan algoritma genetika. Pengujian ini dilakukan untuk melihat kemampuan metode optimasi bobot menggunakan algoritma genetika. Apakah dengan melakukan optimasi bobot dapat memberikan bobot optimal yang tepat. Sehingga dengan diperolehnya bobot optimal yang tepat tersebut dapat meningkatkan hasil akurasi sistem. Untuk menghitung nilai akurasi dapat dilihat pada subbab 4.3.6 sebelumnya. Pengujian optimasi bobot awal menggunakan algoritma genetika ditunjukkan pada Tabel 6.5.

Tabel 6.5 Hasil pengujian optimasi bobot optimal pada LVQ dengan algoritma genetika

Data debitur ke - i	Target (Pakar)	Target (Sistem)	Kecocokan
1	1	1	1
2	2	1	0
3	2	1	0
4	1	1	1
5	1	1	1
6	1	1	1
7	1	1	1
8	1	1	1
9	1	1	1
10	1	1	1
11	1	1	1
12	1	1	1
13	1	1	1
14	1	1	1
15	1	1	1
16	1	1	1
17	1	1	1
18	1	1	1
19	1	1	1
20	1	1	1
21	1	1	1
22	1	1	1

23	1	1	1
24	1	1	1
25	1	1	1
26	2	1	0
27	1	1	1
28	1	1	1
29	1	1	1
30	1	1	1
31	1	1	1
32	1	1	1
33	1	1	1
34	1	1	1
35	1	1	1
36	1	1	1
37	1	1	1
38	1	1	1
39	1	1	1
40	1	1	1
41	1	1	1
42	1	1	1
43	1	1	1
44	1	1	1
45	1	1	1
46	1	1	1
47	1	1	1
48	1	1	1
49	1	1	1
50	1	1	1
51	2	2	1
52	1	1	1
53	2	2	1
54	2	2	1
55	2	1	0
56	1	1	1
57	1	1	1

58	1	1	1
59	1	1	1
60	1	1	1
61	1	1	1
62	1	1	1
63	1	1	1
Cocok			**59**
Akurasi			**93,65%**

BAB 7 KESIMPULAN DAN SARAN

7.1 Kesimpulan

Penerapan algoritma genetika terhadap optimasi bobot pada *Learning Vector Quantization* (LVQ) untuk mendapatkan bobot optimal dalam menentukan kelayakan kredit menggunakan representasi kromosom pengodean bilangan real. Setiap individu memiliki 14 kromosom. Parameter algoritma genetika dapat diimplementasikan dengan metode *two-cut-point crossover*, *random mutation*, dan *elitism selection*.

Pengujian parameter algoritma genetika pada *popsize* memperoleh hasil sebesar 20, *cr* 0.9, *mr* 0.1, dan jumlah generasi 10. Semakin besar ukuran populasi dan ukuran generasi yang digunakan maka nilai rerata akurasu yang didapatkan akan semakin tinggi.

Kombinasi nilai *Cr* yahng lebih tinggi dibandingkan dengan nilai *Mr* akan menghasilkan nilai rata-rata akurasi yang baik karena algoritma genetika lebih bergantung pada proses *crossover* yang dapat menghasilkan keragaman kromosom pada *offspring* lebih banyak dari proses mutasi. Jadi dapat dikatakan bahwa nilai *Cr* yang tinggi dapat membantu mencari solusi permasalahan ke area yang lebih luas. Jika nilai *Cr* terlalu rendah, maka nilai akurasi akan bergantung pada nilai *Mr* yang dapat mengakibatkan terjadinya konvergensi dini sehingga algoritma genetika tidak bisa menemukan hasil rata-rata yang optimal.

Pengujian sistem optimasi bobot *Learning Vector Quantization* (LVQ) menggunakan algoritma genetika mendapatkan nilai akurasi sebesar 93.651%. Sementara hasil pengujian sistem dengan metode *Learning Vector Quantization* (LVQ) saja mendapatkan nilai akurasi sebesar 79.37%. Hal tersebut membuktikan bahwa optimasi bobot optimal untuk metode *Learning Vector Quantization* (LVQ) menggunakan algoritma genetika mampu memberikan akurasi lebih tinggi dan hasil akhir yang lebih optimal.

7.2 Saran

Pada penelitian berikutnya diharapkan menggunakan data yang lebih bervariasi dan memiliki jumlah data yang lebih banyak, serta menggunakan jumlah populasi dan generasi yang lebih besar. Sehingga diharapkan mendapat hasil akurasi yang lebih tinggi lagi.

DAFTAR PUSTAKA

Astiko, 1996. *Manajemen Perkreditan*. Yogyakarta: Andi Offset.

Bonita, H., 2016. *Penggunaan metode TOPSIS dan SAW untuk penentuan kredit pensiunan bagi calon nasabah (studi kasus : PT.Bank X).* S1. Malang: Universitas Brawijaya.

Hendry, M., Dayawati, R. M., dan Wibowo, A. T., 2009. *Analisis dan Implementasi Optimasi Jaringan Saraf Tiruan dengan Menggunakan Algoritma Genetika untuk Pendiagnosaan Penyakit Stroke (Studi Kasus : RS. Dr. M. Djamil Padang Sumbar)*.

Kurniawan, D. E., 2011. *Konsep Learning Vector Quantization LVQ* [Online] Tersedia di: < https://ikhs.wordpress.com/2011/07/03/konsep-learning-vector-quantization-lvq/ > [Diakses pada 22-05-2017]

Kusumadewi, S. 2003. *Artificial Intelligentce: Teknik dan Aplikasinya*. Yogyakarta: Graha Ilmu.

Mahmudy, W.F., 2015. *Modul Kuliah Semester Ganjil 2015-2016: Dasar-Dasar Algoritme Evolusi,* Cover, i-vii,1-105. Tersedia di : < https://wayanfm.lecture.ub.ac.id/2016/03/modul-algoritme-evolusi-semester-ganjil-2015-2016 > [Diakses pada 22-05-2017].

Mahmudy, W. F., dan Rahman, M. A. 2011. *Optimasi Fungsi Multi-Obyektif Berkendala Menggunakan Algoritme Genetika Adaptif Dengan Pengodean Real*. *Jurnal Ilmiah KURSOR*, *6*(1), 19–26.

Sutojo, T., Mulyanto, E., dan Suhartono, V., 2011. *Kecerdasan Buatan*. Yogyakarta: ANDI.

Wibowo, W. A., (2017). Penerapan Algoritma Jaringan Syaraf Tiruan Untuk Prediksi Status Permohonan Hutang Dan Harga Jaminan Hutang Motor.

Utama, W., 2016. *Implementasi Pengolahan Citra dan Learning Vector Qiantization (LVQ) dalam Penentuan Golongan Kendaraan*. S1. Malang : Universitas Brawijaya.

Wijayanti, D. M. P., 2016. Identifikasi Diagnosis Perubahan Hasil Perawatan Kulit Menggunakan Metode Learning Vector Quantization (LVQ). Malang : Universitas Brawijaya.

Wuryandari, D. M., dan Afrianto, I., 2012. *Perbandingan Metode Jaringan Syaraf Tiruan Backpropagation Dan Learning Vector Quantization Pada Pengenalan Wajah.*

LAMPIRAN

Lampiran 1 Data calon nasabah

No.	*Character*	*Condition*	*Capital*	*Capacity*	*Collateral*	Usia	Tanggungan	Keputusan
1	15	353550000	8360000	4497673	100000000	37	3	Diterima
2	15	330800000	18000000	4700624	90000000	46	4	Diterima
3	13	611000000	4800000	2954775	78806000	28	2	Diterima
4	15	397900000	11400000	2117673	60000000	53	3	Diterima
5	13	805500000	9600000	2983308	62310000	61	2	Diterima
6	15	510125000	8750000	3417673	14000000	55	3	Diterima
7	15	136650000	5700000	2483308	65787150	36	2	Diterima
8	15	712500000	24000000	7860428	85000000	24	2	Diterima
9	15	309625000	19500000	4417673	29476000	53	3	Diterima
10	12	69700000	1000000	2117673	32600000	39	3	Diterima
11	12	189342500	3300000	3007869	95717500	44	5	Diterima
12	13	157986000	2500000	2483306	68450000	38	2	Diterima
13	15	460000000	18000000	7259413	80000000	34	3	Diterima
14	15	278800000	13200000	3479158	59430000	37	4	Diterima
15	15	1043250000	135000000	36975007	250000000	53	6	Diterima
16	13	273500000	12000000	4423504	29500000	43	4	Diterima
17	13	590500000	51000000	3054775	62800000	44	2	Diterima

18	12	519000000	3300000	2454775	40000000	52	2	Diterima
19	13	273500000	12000000	4423504	29500000	43	4	Diterima
20	14	310300000	16000000	4667673	350000000	42	3	Diterima
21	15	258500000	12000000	1357673	450000000	43	3	Diterima
22	13	315000000	10000000	1089140	40936000	37	3	Diterima
23	13	305437500	2520000	919532	163450000	40	7	Diterima
24	12	376050000	1955000	4916438	265020000	39	4	Diterima
25	13	95735000	1875000	3104775	486000000	30	2	Diterima
26	15	358500000	34000000	7960607	150000000	39	3	ditolak
27	14	357000000	12000000	2592038	278000000	48	4	Diterima
28	14	422250000	3900000	2517673	50000000	55	3	Diterima
29	13	429800000	16000000	3389140	75000000	65	3	Diterima
30	12	294800000	14400000	550803	133100000	39	5	Diterima
31	13	339000000	14000000	2439140	23866000	59	3	Diterima
32	12	143812500	1035000	2516438	132500000	44	4	Diterima
33	15	770000000	42000000	11194971	175000000	29	4	Diterima
34	15	238500000	12800000	2702036	150000000	46	4	Diterima
35	15	166750000	3648000	-296692	75000000	64	2	Diterima
36	15	770000000	42000000	11194971	175000000	29	4	Diterima
37	15	348500000	34000000	7960607	230338000	56	3	Diterima
38	12	87543750	1237500	1061438	100000000	29	4	Diterima

39	15	358500000	34000000	7960607	100000000	54	3	Diterima
40	15	1064500000	60000000	16717673	125000000	45	3	Diterima
41	15	1400000000	140000000	52694971	190000000	59	4	Diterima
42	14	425500000	30000000	7944971	105000000	42	4	Diterima
43	15	239200000	20000000	3696474	32480000	52	6	Diterima
44	15	230500000	16000000	4223522	125000000	41	5	Diterima
45	15	245400000	16000000	3462109	40000000	44	5	Diterima
46	15	176500000	12000000	3044793	50000000	35	3	Diterima
47	15	224750000	18240000	3649140	55836000	64	3	Diterima
48	15	254500000	12000000	2733308	45438250	65	2	Diterima
49	15	194200000	6500000	5267673	49800000	37	3	Diterima
50	15	181000000	4860000	2428944	51800000	40	1	Diterima
51	0	0	0	-856888	20000000	41	0	ditolak
52	15	239200000	20000000	3696474	324800000	52	6	Diterima
53	0	0	0	-856888	22186450	55	0	ditolak
54	0	0	0	-856888	135000000	48	0	ditolak
55	15	297000000	0	510007	43275000	41	3	ditolak
56	14	236350000	0	-7039393	95000000	72	3	ditolak
57	7	179500000	0	-1789393	125000000	51	3	ditolak
58	15	176500000	12000000	3044793	50000000	35	3	Diterima
59	15	263000000	14000000	2462109	100500000	66	5	Diterima

60	15	176500000	12000000	3044793	25000000	39	3	Diterima
61	14	377100000	8360000	3494971	80000000	38	4	Diterima
62	15	327300000	25900000	5823504	212225000	38	4	Diterima
63	15	281000000	12600000	4393504	75000000	50	4	Diterima