IMPLEMENTASI ALGORITMA GA-KNN UNTUK PENGKLASIFIKASIAN STATUS RESIKO KREDIT FINANSIAL

SKRIPSI

Sebagai salah satu syarat untuk memperoleh gelar
Sarjana dalam bidang Ilmu Komputer

Disusun oleh :

MUHAMMAD RIZKY WAHYU PRATAMA

NIM. 0810960053

PROGRAM STUDI TEKNIK INFORMATIKA

PROGRAM TEKNOLOGI INFORMASI

DAN ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

MALANG

2013

LEMBAR PERSETUJUAN

IMPLEMENTASI ALGORITMA GA-KNN UNTUK PENGKLASIFIKASIAN STATUS RESIKO KREDIT FINANSIAL

SKRIPSI

Disusun oleh :

MUHAMMAD RIZKY WAHYU PRATAMA

NIM. 0810960053

Telah diperiksa dan disetujui oleh :

Pembimbing I,	Pembimbing II,
Drs. Achmad Ridok, M.Kom	**Drs. Muh. Arif Rahman, M.Kom**
NIP. 196808251994031002	**NIP. 196604231991111001**

LEMBAR PENGESAHAN

IMPLEMENTASI ALGORITMA GA-KNN UNTUK PENGKLASIFIKASIAN STATUS RESIKO KREDIT FINANSIAL

SKRIPSI

Sebagai salah satu syarat untuk memperoleh gelar
Sarjana dalam bidang Ilmu Komputer

Disusun oleh :

MUHAMMAD RIZKY WAHYU PRATAMA

NIM. 0810960053

Skripsi ini telah diuji dan dinyatakan lulus pada tanggal 8 April 2013

Penguji I	Penguji II
Drs. Marji, M.T.	**Dian Eka Ratnawati, S.Si., M.Kom**
NIP. 19670801199203100 1	**NIP. 197306192002122001**

Penguji III

Ahmad Afif Supianto, S.Si., M.Kom
NIK. 82062316110425

Mengetahui
Ketua Program Studi Teknik Informatika

Drs. Marji, M.T.
NIP. 19670801199203100 1

PERNYATAAN ORISINALITAS SKRIPSI

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 dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan 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, April 2013

Muhammad Rizky Wahyu Pratama
NIM 0810960053

IMPLEMENTASI ALGORITMA GA-KNN UNTUK PENGKLASIFIKASIAN STATUS RESIKO KREDIT FINANSIAL

ABSTRAK

Kredit merupakan elemen yang penting dalam perbankan. Melalui kredit, nasabah dapat menerima pinjaman dana dari bank yang harus dikembalikan pada tenggat waktu tertentu. Pada suatu kasus kredit dapat terjadi suatu kredit macet atau ketidaklancaran pembayaran hutang oleh debitur atau nasabah. Kredit macet dapat disebabkan oleh faktor eksternal seperti krisis moneter, atau faktor internal yaitu kekurangmampuan pihak bank dalam menilai resiko calon debitur. Faktor eksternal sulit dikontrol, sementara faktor internal dapat dikontrol oleh pihak bank.

Penelitian ini mengimplementasikan metode *Genetic Algorithm K-Nearest Neighbor* (GA/KNN) pada pengklasifikasian status resiko kredit debitur. GA/KNN bekerja dengan cara membangun vektor bobot secara acak sebanyak ukuran populasi, kemudian vektor tersebut akan diregenerasikan hingga mencapai vektor bobot terbaik pada suatu kriteria berhenti. Vektor bobot terbaik yang diperoleh selanjutnya akan digunakan sebagai dasar pembobotan dalam pengklasifikasian berbobot untuk pengambilan keputusan.

Pengujian dilakukan terhadap vektor bobot terbaik yang telah terbentuk berdasarkan kombinasi nilai probabilitas *crossover* (PC), probabilitas mutasi (PM), dan ukuran populasi. Pengujian ini menggunakan metode pengujian akurasi yang dilakukan pada 200 data uji yang berbeda dari data latih. Hasil pengujian yang dilakukan menunjukkan tingkat akurasi tertinggi dengan prosentase 65%. Hasil ini diperoleh pada nilai PC sebesar 70%, nilai PM sebesar 30%, dan ukuran populasi sebesar 100.

Kata Kunci : GA/KNN, *Data Mining*, *Genetic Algorithm,K-Nearest Neighbor, credit scoring*.

IMPLEMENTATION OF GA-KNN ALGORITHM FOR FINANCIAL CREDIT RISK STATUS CLASSIFICATION

ABSTRACT

Credit is an essential element in banking. Through loans, customers can receive a loan from the bank to be returned on a specific deadline. In the case of credit can occur a bad credit oruncertainty debt repayment by borrowers or customers. Bad credit can be caused by external factors such as financial crisis, or internal factors, namely incapacity banks in assessing the risk of borrowers. External factors difficult to control, while internal factors can be controlledby the bank.

This study implements the method of Genetic Algorithm K-Nearest Neighbor (GA / KNN) classification status on credit risk borrowers. GA / KNN works by building a weight vector randomly as population size, then the vector will be regenerated to achieve the best weight vector at a stop criterion. Best weighting vector obtained will then be used as the basis for weighting in the weighted classification decision.

Tests conducted on the best weighting vector that has been formed by the combination of crossover probability value (PC), the probability of mutation (PM), and population size. This test method accuracy tests performed on 200 test data differ from the data train. The results of the tests performed showed the highest level of accuracy with a percentage of 65%. These results were obtained on the PC by 70%, the value of PM by 30%, and the population size of 100.

Key Word : GA/KNN, *Data Mining*, *Genetic Algorithm,K-Nearest Neighbor, credit scoring*.

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat TuhanYang Maha Esa yang telah melimpahkan segala kasih sayang dan rahmat-Nya sehingga penulis dapat menyelesaikan tugas akhir dengan judul "**Implementasi Algoritma GA-KNN untuk Pengklasifikasian Status Resiko Kredit Finansial**".

Skripsi ini diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer Strata Satu (S-1) Program Studi Teknik Informatika, Program Teknologi Informasi & Ilmu Komputer Universitas Brawijaya, Malang.Penulis menyadari bahwa tugas akhir ini tidak dapat terealisasikan tanpa bantuan dari berbagai pihak, untuk itu penulis menyampaikan penghargaan dan ucapan terima kasih yang sedalam-dalamnya kepada :

1. Drs. A. Ridok, M.Kom selaku pembimbing utama yang telah meluangkan waktu memberikan pengarahan dan bimbingan kepada penulis.
2. Drs. Muh.Arif Rahman, M.Kom selaku pembimbing pendamping yang telah meluangkan waktu memberikan pengarahan dan bimbingan kepada penulis.
3. Drs. Marji, MT.,selakuKetua Program Studi Teknik Informatika, Program Teknologi Informasi & Ilmu Komputer Universitas Brawijaya.
4. Candra Dewi, S.Kom, M.Scselaku dosen pembimbing akademik, atas nasehat dan bimbingan akademik yang telah diberikan.
5. Segenap bapak dan ibu dosen yang telah mendidik dan mengajarkan ilmunya kepada penulis selama menempuh pendidikan di Program Teknologi Informasi & Ilmu Komputer Universitas Brawijaya.
6. Segenap staf, karyawan dan civitas diProgram Teknologi Informasi & Ilmu Komputer Universitas Brawijaya yang telah banyak membantu penulis dalam pelaksanaan penyusunan tugas akhir ini.
7. Kedua orang tua yang penulis cintai, yang telah memberikan dukungan baik materi maupun non-materi selama penulis menyelesaikan tugas akhir ini.
8. Rina Ayoni Dewi, yang senantiasa menemani, mendengarkan keluhan, memberikan motivasi, doa dan semangat kepada penulis
9. Rekan-rekan mahasiswa Program Studi Teknik Informatika Universitas Brawijaya yang telah memberikan dukungan, semangat dan kebersamaan.

10. Semua pihak yang telah membantu terselesaikannya penyusunan tugas akhir ini yang tidak dapat disebutkan satu per satu.

Penulis menyadari bahwa tugas akhir ini masih jauh dari kesempurnaan, untuk itu dengan segala kerendahan hati penulis mengharapkan saran dan kritik yang membangun demi kesempurnaan penulisan selanjutnya. Semoga laporan tugas akhir ini dapat bermanfaat untuk semua pihak.

Malang, April 2013

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR SOURCE CODE

BAB I
PENDAHULUAN

1.1Latar Belakang

Perbankan merupakan salah satu institusi yang mempunyai peran dalam upaya untuk meningkatkan kesejahteraan bangsa dan negara. Melalui perbankan dana masyarakat dapat dihimpun melalui tabungan, deposito, giro dan selanjutnya disalurkan kembali ke pihak-pihak yang membutuhkan dana dalam bentuk pinjaman. Pada suatu kasus pinjaman atau kredit dapat terjadi suatu kredit macet atau ketidaklancaran pembayaran hutang oleh debitur atau nasabah. Kredit macet ini dapat disebabkan oleh faktor eksternal seperti kondisi ekonomi yang tidak kondusif, debitur yang nakal, atau faktor internal yaitu kekurangmampuan pihak bank dalam menilai resiko calon debitur. Faktor eksternal sulit dikontrol oleh pihak bank, sementara faktor internal dapat dikontrol oleh pihak bank.

Berkembangnya teknologi informasi yang pesat menyebabkan penilaian resiko yang semula dilakukan secara *human judgment* bergeser ke arah formal dan objektif yaitu melalui *credit scoring*. Tujuan dari *credit scoring* adalah membantu pihak penyedia kredit mengkuantifikasi resiko finansial sehingga keputusan dapat diambil dengan cepat dan akurat [CHY-04]. Pada era ini, telah banyak terknik yang dapat digunakan untuk membantu dalam pembangunan model *credit scoring*, salah satunya yaitu teknik klasifikasi yang terdapat di dalam*data mining*.

Salah satu teknik klasifikasi yang sederhana akan tetapi memiliki performa yang cukup baik adalah Algoritma *K-Nearest Neighbour* [CUN-07]. Algoritma ini berfungsi untuk memprediksi kelas dari suatu *record* baru dengan mencari k *training record* (tetangga) yang memiliki jarak terdekat dari *record*baru tersebut[SAR-00].Algoritma *K-Nearest Neighbour* (KNN) memiliki prosedur yang efektif akan tetapi sensitif terhadap *noise* atau fitur yang menyimpang, sehinggasalah satu cara untuk mengurangi tingkat *noise* tersebut adalah dengan memberikan bobot. Metode ini biasa disebut *weighted K-Nearest Neighbour* (W-KNN)[WRO-10].

Pemberian bobot yang optimal pada tiap atribut sangat penting karena dapat mempengaruhi performa klasifikasi, sehinggauntuk menemukan vektor bobot

yang optimal diperlukan komputasi yang cukup kompleks apabila diterapkan data yang sangat besar. Permasalahan menemukan vektor bobot yang optimal ini dapat diatasi dengan Algoritma Genetika [WRO-10]. Algoritma genetika adalah algoritma pencarian heuristik yang didasarkan atas mekanisme seleksi dan genetika alami. Konsep dasar yang mengilhami timbulnya algoritma genetika adalah teori evolusi, yang dikemukakan oleh Charles Darwin. Algoritma genetika dimulai dengan membentuk sejumlah alternative solusi yang disebut sebagai populasi. Setiap solusi pada algoritma genetika diwakili oleh satu individu atau satu kromosom [KUS-03]. Penggabungan dua algoritma ini memiliki tujuan masing-masing. Algoritma K-Nearest Neighbour mengklasifikasikan suatu hal menggunakan jarak diantaranya dan Algoritma Genetika mencari vektor bobot 'terbaik' yang membantu untuk mengurangi kesalahan klasifikasi yaitu kumpulan bobot yang diterapkan untuk meningkatkan atau mengurangi kepentingan jarak diantara setiap nilai atribut dari data latih dan nilai atribut sama yang meragukan [PRO-09].

Penelitian tentang GA *K-Nearest Neighbour* (GA/KNN) pernah dilakukan oleh Irina Provorova dkk (2009) yang berjudul *UsingGenetic Algorithm to Optimize Weights in Data Mining Task.* Dalam penelitian ini, GA *K-Nearest Neighbour* diimplementasikan pada banyak kasus yang di ambil dari UCI *repository*. Pada penelitian tersebut, menunjukkan bahwa GA *K-Nearest Neighbour* terbukti lebih efektifdan memiliki rata-rata keakuratan lebih baik (akurasi sebesar 83%) dibandingkan dengan algoritma *K-Nearest Neighbour* (akurasi sebesar 70%).

Berdasarkan latar belakang yang telah dipaparkan maka judul yang diambil pada skripsi ini adalah "**ImplementasiAlgoritma GA-KNNuntuk Pengklasifikasian Status Resiko Kredit Finansial**".

1.2Rumusan Masalah

Berdasarkan uraian pada latar belakang, maka dalam skripsi ini dapat dirumuskan permasalahan sebagai berikut:

1. Bagaimana menerapkan *Genetic Algorithm K-Nearest Neighbor* pada kasus status risiko kredit.

2. Seberapa besar akurasi dalam klasifikasi menggunakan teknik *Genetic Algorithm*pada *K-Nearest Neighbor Classifier* dibandingkan dengan teknik *K-Nearest Neighbor Classifier* murni untuk status risiko kredit.

1.3Batasan Masalah

Untuk membatasi masalah yang dikaji, maka penulis membatasi dan memfokuskan permasalahan sebagai berikut:

1. Pengujian sistem menggunakan data-contoh (dataset) yaitu*Statlog* (German Credit data). Dataset ini dapat diunduh dari situs http://archive.ics.uci.edu/ml/datasets/Statlog
2. Dataset yang digunakan tidak mengandung *missing value*(nilai hilang).
3. Operator genetika yang diterapkan pada sistem meliputi Persilangan Terbatas (*Restricted Crossover*), Mutasi penukaran terhadap gen (*Exchange Mutation*), dan Seleksi Roda *Roullette* (*Roullette Wheel Selection*).

1.4Tujuan Penelitian

Tujuan penelitian dalam skripsi ini adalah :

1. Menerapkan metode *Genetic AlgorithmK-Nearest Neighbor*pada kasus status risiko kredit.
2. Mengetahui besarnya akurasi dalam klasifikasi menggunakan teknik *GA K-Nearest Neighbor classifier*dan teknik *K-Nearest Neighbor classifier* murni untuk status risiko kredit.

1.5Manfaat Penelitian

Manfaat yang bisa diambil dari skripsi ini adalah untuk menentukan status risiko kredit pada calon nasabah dan membantu para kreditor untuk menggunakan waktu seefisien mungkin dalam mengevaluasi debitur (pemohon kredit) secara objektif, akurat dan konsisten.

1.6Sistematika Penulisan

Skripsi ini disusun berdasarkan sistematika penulisan sebagai berikut:

1. **BAB I PENDAHULUAN**

   Berisi latar belakang masalah, perumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi pemecahan masalah, dan sistematika penulisan.

2. **BAB II TINJAUAN PUSTAKA**

   Menguraikan teori-teori penunjang yang berhubungan dengan konsep dasar klasifikasi, Algoritma Genetika, Algoritma *K-Nearest Neighbour,* Algoritma *Weighted K-Nearest Neighbour* dan *GAK-Nearest Neighbour*.

3. **BAB III METODOLOGI DAN PERANCANGAN SISTEM**

   Pada bab ini akan dijelaskan mengenai langkah-langkah yang digunakan dalam menerapkan*GAK-Nearest Neighbour* pada klasifikasi status risiko kredit

4. **BAB IV IMPLEMENTASI DAN UJI COBA SISTEM**

   Dalam bab ini dijelaskan mengenai implementasi sistem, pengujian dan analisa sistem perangkat lunak yang dibangun.

5. **BAB V KESIMPULAN DAN SARAN**

   Berisi kesimpulan dari seluruh rangkaian penelitian serta saran kemungkinan pengembangannya.

BAB II
TINJAUAN PUSTAKA

2.1Kredit

2.1.1Pengertian Kredit

Dalam bahasa latin, kredit disebut "credere" yang artinya percaya, yaitu kepercayaan dari kreditur bahwa debiturnya akan mengembalikan pinjaman beserta bunganya sesuai dengan perjanjian kedua belah pihak [KAS-03]. Pengertian kredit menurut undang-undang Perbankan No. 10 Tahun 1998, adalah sebagai berikut :

Kredit adalah penyediaan uang atau tagihan yang dapat dipersamakan dengan itu berdasarkan persetujuan atau kesepakatan pinjam meminjam antara bank dengan pihak lain yang mewajibkan pihak peminjam untuk melunasi utangnya setelah jangka waktu tertentu dengan jumlah bunga imbalan atau pembagian hasil keuntungan [KAS-03].

Kredit merupakan pinjaman yang diberikan pihak bank selaku kreditur kepada nasabah selaku debitur. Kredit yang diberikan harus melalui beberapa proses kesepakatan yang menguntungkan bagi kedua belah pihak, sehingga tidak ada yang saling diuntungkan dan dirugikan karena pada intinya bank merupakan penolong bagi debitur [HAS-05].

2.1.2Risiko Kredit

Risiko kredit adalah risiko kerugian yang diderita bank terkait dengan kemungkinan bahwa pada saat jatuh tempo *counterparty* gagal memenuhi kewajiban-kewajibannya kepada bank. Bagi bank risiko kerugian karena terjadinya kegagalan *counterparty* memenuhi kewajibannya tersebut merupakan risiko yang wajar, mengingat hal tersebut terkait dengan bisnis inti bank yaitu *lending-based business*. Hakikat bank sebagai lembaga dengan tingkat *leverage* atau *debt-equity ratio* yang tinggi, menyebabkan permodalan bank dapat tergerus habis bila para debiturnya memiliki *default rate* yang tinggi. Risiko kredit merupakan risiko terbesar yang dihadapi perbankan, karena sebagian besar

struktur asset yang dimiliki perbankan dalam neracanya adalah berbentuk kredit. Dengan demikian menjadi hal yang penting untuk mengukur seberapa besar nilai risiko yang terkandung dalam suatu eksposure kredit. Seberapa besar tingkat akurasi risiko kredit sangat tergantung pada pemilihan metodologi yang paling sesuai dengan karakteristik dan kompleksitas kredit yang disalurkan bank [SUL-09].

2.1.3Credit Scoring

Banyaknya permohonan kredit menuntut kreditor harus mampu mengevaluasi pemohon kredit dengan objektif, akurat, dan konsisten. Evaluasi tersebut dapat dibantu dengan *credit scoring*. Isac mendefinisikan *credit scoring* sebagai *tool* yang melibatkan penggunaan model statistik untuk mengevaluasi seluruh informasi yang tersedia dengan objektif dalam pengambilan keputusan kredit [NOE-97].

Manfaat yang dapat diperoleh dari penerapan *credit scoring* adalah peningkatan kecepatan dan konsistensi proses aplikasi pinjaman dan memungkinkan otomatisasi proses peminjaman [CHY-04]; adanya kemampuan belajar sepanjang waktu karena model *credit scoring* didasarkan pada perhitungan statistik dari data masa lalu [GLA-97].

Penerapan model statistik prediktif membutuhkan dua faktor [GLA-97]:

1. Teknologi yang memungkinkan model bekerja dengan cepat sehingga kecepatan proses memberikan waktu respon yang dapat diterima, dan
2. Basis data yang menyediakan input bagi model prediktif.

Seiring dengan kemajuan teknologi informasi yang sangat pesat disertai dengan harga yang semakin terjangkau, sangat dimungkinkan bagi perusahaan kecilpun menggunakan model statistik dalam mengevaluasi kredit.

Model *credit scoring* dibangun dengan menggunakan sampel kredit masa lalu dalam jumlah yang besar. Sampel tersebut dibagi kedalam dua kelas yaitu kredit yang baik (pembayaran dilakukan tepat waktu) dan kredit yang bermasalah (pembayaran dilakukan tidak tepat waktu atau tidak dapat melakukan pembayaran). Berdasarkan pola masa lalu, kombinasi karakteristik peminjam

yang membedakan peminjam yang baik dan yang buruk menghasilkan skor sebagai estimasi resiko dari tiap peminjam baru.

2.1.4 German Credit dataset

Dataset kredit di bawah memaparkan kriteria – kriteria yang menjadi penilaian dalam mengambil keputusan untuk menentukan apakah suatu kredit yang diajukan termasuk kredit yang *Good Risk* atau *Bad Risk*. Keterangan mengenai kriteria – kriteria penilaian *credit scoring* ditampilkan pada tabel 2.1.

Tabel 2.1Dataset

Keterangan: (Tipe : N = numerik, K = Kategori)

No	Nama Atribut	Tipe		Keterangan
		N	K	
1.	Status laporan pengecekan yang ada *(Status of existing checking account)*		v	A11 : ... < 0 DM A12 : 0 <= ... < 200 DM A13 : ... >= 200 DM / *salary assignments for at least 1 year* A14 : *no checking account*
2.	Lamanya dalam bulan *(Duration in month)*	v		-
3.	Sejarah kredit *(Credit history)*		v	A30 : *no credits taken/ all credits paid back duly* A31 : *all credits at this bank paid back duly* A32 : *existing credits paid back duly till now* A33 : *delay in paying off in the past* A34 : *critical account/ other credits existing (not at this bank)*
4.	Kegunaan *(Purpose)*		v	A40 : *car (new)* A41 : *car (used)* A42 : *furniture/equipment*

				A43 : *radio/television* A44 : *domestic appliances* A45 : *repairs* A46 : *education* A47 : *(vacation - does not exist?)* A48 : *retraining* A49 : *business* A410 : *others*
5.	Jumlah Kredit *(Credit amount)*	v		-
6.	Tabungan/Surat obligasi *(Savings account/bonds)*		v	A61 : ... < 100 DM A62 : 100 <= ... < 500 DM A63 : 500 <= ... < 1000 DM A64 : .. >= 1000 DM A65 : *unknown/ no savings account*
7.	Pekerjaan sekarang sejak *(Present employment since)*		v	A71 : *unemployed* A72 : ... < 1 year A73 : 1 <= ... < 4 years A74 : 4 <= ... < 7 years A75 : .. >= 7 years
8.	Tarif angsuran dalam prosentase dari pendapatan bersih setelah pajak *(Installment rate in percentage of disposable income)*	v		-
9.	Status pribadi dan jenis kelamin *(Personal status and sex)*		v	A91 : *male : divorced/separated* A92 : *female :divorced/separated/married* A93 : *male : single* A94 : *male : married/widowed* A95 : *female : single*
10.	Debitor / penjamin lain *(Other debtors / guarantors)*		v	A101 : *none* A102 : *co-applicant* A103 : *guarantor*
11.	Tepat tinggal sekarang sejak*(Present residence since)*	v		-

12.	Kekayaan *(Property)*		v	A121 : *real estate* A122 : *if not A121 : building society savings agreement/ life insurance* A123 : *if not A121/A122 : car or other, not in attribute 6* A124 : *unknown / no property*
13.	Umur dalam tahun *(Age in years)*	v		-
14.	Rencana angsuran yang lain *(Other installment plans)*		v	A141 : *bank* A142 : *stores* A143 : *none*
15.	Tempat tinggal *(Housing)*		v	A151 : *rent* A152 : *own* A153 : *for free*
16.	Jumlah kredit yang ada di bank ini *(Number of existing credits at this bank)*	v		-
17.	pekerjaan *(Job)*		v	A171 : *unemployed/ unskilled - non-resident* A172 : *unskilled - resident* A173 : *skilled employee / official* A174 : *management/self- employed/ highly qualified employee/ officer*
18.	Jumlah orang yang menjadi tanggung jawab untuk menyediakan biaya hidup untuk *(Number of people being liable to provide maintenance for)*	v		-
19.	Telepon *(Telephone)*		v	A191 : *none* A192 : *yes, registered under the customers name*
20.	Pekerja asing *(foreign worker)*		v	A201 *: yes* A202 : *no*

Dataset kredit di atas terdiri dari data numerik dan data kategori. Pada data numerik dapat menggunakan normalisasi atau standarisasi sebelum dilakukan klasifikasi. Salah satu metode untuk normalisasi adalah *Min-max normalization* yang dirumuskan sebagai berikut :

$$X^* = \frac{X - \min(X)}{\max(X) - \min(X)} \quad (2.1)$$

Keterangan :

X* : hasil normalisasi

X : nilai pada data yang akan dinormalisasi

Min(X) : nilai terkecil pada data yang akan dinormalisasi

Max(X) : nilai terbesar pada data yang akan dinormalisasi

Sedangkan untuk data kategori menggunakan persamaan sebagai berikut [LAR-05] :

$$different(x_i, y_i) = \begin{cases} 0 & if\ x_i = y_i \\ 1 & otherwise \end{cases} \quad (2.2)$$

2.2 Data Mining

2.2.1 Definisi Data Mining

Menurut Han dan Kamber (2006), *data mining* merupakan solusi yang mampu menemukan kandungan informasi yang tersembunyi berupa pola dan aturan dari sekumpulan data yang besar. Informasi yang tersembunyi ini menguntungkan dari sudut pandang penelitian, bisnis dan lainnya. *Data mining* diharapkan menghasilkan informasi yang sangat akurat dan mudah dipahami.

Data mining dapat dikategorikan sebagai *supervised* dan *unsupervised*. Dalam metode *unsupervised*, tidak ada variabel tujuan yang diidentifikasikan. Kebanyakan data mining menggunakan metode *supervised*, yang berarti ada variabel target yang dispesifikasi sebelumnya. Algoritma diberikan beberapa contoh dimana nilai dari variabel target disediakan sehingga algoritma dapat mempelajari yang mana nilai variabel target berhubungan dengan yang mana nilai variabel pemrediksi [LAR-05].

2.2.2 Teknik Data Mining

Pengelompokan *data mining* berdasarkan tugas yang dapat dilakukan,antara lain [LAR-05] :

a. Deskripsi

Deskripsi menggambarkan pola dan kecenderungan yang terdapat dalam data yang memungkinkan memberikan penjelasan dari suatu pola ataukecenderungan tersebut.

b. **Prediksi**.

Dalam prediksi nilai dari hasil akan terwujud di masa yang akan datang.

c. **Klasifikasi**

Klasifikasi adalah proses pengelompokan *record* ke dalam kelas objek yang sama yang memiliki variabel target ke arah kategori.

d. **Estimasi**

Estimasi hampir sama dengan klasifikasi, tetapi variabel target estimasi lebih ke arah numerik.

e. ***Clustering***

*Clustering*hampir sama dengan klasifikasi, tetapi *Clustering* tidak memiliki variabel target.

f. **Asosiasi**

Asosiasi adalah menemukan atribut yang muncul dalam satu waktu.

2.2.3 Klasifikasi

Klasifikasi adalah proses untuk menemukan model atau fungsi yang menggambarkan dan membedakan kelas data atau konsep dengan tujuan memprediksikan kelas untuk data yang tidak diketahui kelasnya [HAN-06].

Proses klasifikasi biasanya dibagi menjadi dua fase yaitu pembelajaran dan uji coba. Pada fase pembelajaran, sebagian data yang telah diketahui kelas datanya diumpankan untuk membentuk model perkiraan. Kemudian pada fase uji coba, model yang sudah terbentuk diuji dengan sebagian data lainnya untuk mengetahui akurasi dari model tersebut. Bila akurasinya mencukupi model ini dapat dipakai untuk prediksi kelas data yang belum diketahui [PRA-03].

Tujuan dari klasifikasi adalah untuk menganalisa data latih dan membentuk sebuah deskripsi yang akurat atau sebuah model untuk setiap kelas berdasarkan fitur-fitur yang tersedia di dalam data itu. Deskripsi dari masing-masing kelas itu nantinya akan digunakan untuk mengklasifikasikan data yang hendak diuji dalam basis data, atau untuk membangun suatu deskripsi yang lebih baik untuk setiap kelas dalam basis data.

Metode yang biasa digunakan untuk klasifikasi antara lain K-Nearest Neighbour, Naïve Bayes, Decision Tree Based Methods, Rule-Based Methods, Support Vector Machines, Neural Network dan lain sebagainya.

2.2.4 Algoritma K-Nearest Neighbor (KNN)

Algoritma k-nearest neighbourmerupakan salah algoritma pembelajaran terarah (*supervised learning*) yang mencari k training record (tetangga) yang memiliki jarak terdekat dari suatu record baru untuk memeprediksi kelas dari record baru tersebut. Data dari KNN dapat berupa data kategorik, dan atau data numerik. Kelebihan dari algoritma KNN adalah sederhana dan mudah diimplementasikan [SAR-00].

2.2.4.1Proses K-Nearest Neighbor

Pada skripsi ini metode yang digunakan untuk membandingkan kemiripan data yaitu menggunakan metrik *Euclidian Distance*. Jarak antara dua titik data x dan y dapat dihitung dengan rumus sebagai berikut :

$$\boldsymbol{dist_{x,y} = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}} \quad \boldsymbol{(2.3)}$$

Setelah diketahui jarak antar record, kemudian diambil sebanyak k tetangga terdekat untuk memprediksi label kelas dari record baru menggunakan label kelas tetangga. Prediksi tersebut diketahui dengan fungsi kombinasi yang dibahas pada subbab 2.2.4.2.

2.2.4.2Fungsi Kombinasi

Untuk memberikan keputusan klasifikasi bagi record baru dilakukan kombinasi terhadap record yang serupa, dengan fungsi kombinasi. Pada skripsi ini menggunakan metode *Weighted Voting*. Pada Weighted Voting dilakukan dengan memberikan bobot terhadap beberapa tetangga yang dekat dengan record yang baru. Pemberian bobot ini dapat memberikan pengaruh lebih dalam menentukan label kelas. Persamaan metode *Weighted Voting* adalah sebagai berikut :

$$Weight\ (neighbor) = \frac{1}{Distance(neighbor, newRecord)^2} \mathbf{(2.4)}$$

Distance(neighbor,newRecord) merupakan jarak antara record yang baru dengan tetangganya. Persaman 2.4 dapat menghitung bobot dari semua tetangganya, dengan menjumlahkan bobot dari tetangga yang memiliki label kelas yang sama. Label kelas dari record yang baru adalah label kelas dari record yang jumlah bobot tetangganya paling besar [MOR-09].

2.2.5Algoritma Weighted K-Nearest Neighbour (WK-NN)

Algoritma *K-Nearest Neighbour* memiliki prosedur yang efektif akan tetapi sensitif terhadap *noise* (fitur yang menyimpang). Salah satu cara untuk mengurangi tingkat noise tersebut adalah dengan memberikan bobot. Metode yang digunakan salah satunya yaitu teknik *feature extraction* (Ekstraksi Fitur).

Dalam hal ini algoritma *K-Nearest Neighbour* berubah menjadi *weighted K-Nearest Neighbour*.Kedekatan antara dua pola x dan y diukur (dalam *Euclidian Metric*) dengan:

$$\boldsymbol{dist_{x,y}} = \sqrt{\sum_{i=1}^{n} \left(\boldsymbol{w_i(x_i - y_i)}\right)^2} \ \mathbf{(2.5)}$$

dimana w_i adalah bobot atribut ke-i. bobot dapat berupa bilangan real atau integer dari interval yang telah ditetapkan.

Pemberian bobot yang optimal pada tiap atribut sangat penting karena dapat mempengaruhi performa klasifikasi. Untuk menemukan vektor bobot yang optimal diperlukan komputasi yang cukup kompleks apabila diterapkan data yang

sangat besar. Permasalahan ini dapat diatasi dengan Algoritma Genetika [WRO-10].

2.3Algoritma Genetika

2.3.1Definisi Algoritma Genetika

Algoritma genetika adalah algoritma pencarian heuristik yang didasarkan atas mekanisme seleksi dan genetika alami [GEN-97]. Algoritma genetika pertama kali diperkenalkan oleh John Holland dari Universitas Michigan pada awal 1970 dengan tulisannya berjudul *Adapted in Natural and Artificial System* yang cara kerjanya berdasarkan seleksi alam. John Holland menyatakan bahwa setiap masalah yang berbentuk adaptasi (alami maupun buatan) dapat diformulasikan dalam terminologi genetika [KUS-03].

Individu yang lebih kuat akan memiliki tingkat survival dan tingkat reproduksi yang lebih tinggi jika dibandingkan dengan individu yang kurang fit atau kecocokannya rendah. Pada kurun waktu tertentu (seringkali disebut generasi), populasi secara keseluruhan akan lebih banyak memuat organisme yang fit [GEN-97].

2.3.2Struktur Umum Algoritma Genetika

Algoritma genetika merupakan teknik pencarian yang didasarkan pada mekanisme seleksi dan genetika alami. Berbeda dengan teknik pencarian konvensional, algoritma genetika dimulai dengan membentuk kumpulan solusi (kromosom) yang dinamakan populasi. Solusi dari suatu populasi akan diambil untuk membentuk populasi baru. Solusi ini diambil berdasarkan nilai *fitness*. Proses pembentukan populasi baru ini dilakukan dengan harapan bahwa populasi baru yang terbentuk merupakan populasi yang lebih baik dari sebelumnya.

Secara garis besar, algoritma genetika dapat dijelaskan dengan algoritma berikut [OBI-98] :

1. [**Mulai**] Membangun populasi yang beranggotakan n kromosom secara acak (sesuai dengan permasalahan).
2. [***Fitness***] Evaluasi setiap *fitnessf(x)* dari setiap kromosom x dalam populasi tersebut.

3. [**Populasi baru**] Membuat populasi baru dengan cara mengulang langkah-langkah berikut sampai sebuah populasi baru terbentuk.
    1. [**Seleksi**] Memilih dua kromosom induk (*parent*) dari populasi berdasarkan nilai *fitness*nya (semakin besar *fitness*nya semakin besar pelungnya untuk terpilih).
    2. [***Crossover***] Sesuai dengan besarnya Probabilitas *crossover*, lakukan pindah silang antara dua *parent* untuk membentuk *offspring* baru.
    3. [**Mutasi**] Sesuai dengan besarnya Probabilitas mutasi, lakukan mutasi pada *offspring* untuk membentuk *offspring* baru.
4. [**Ganti**] Gunakan populasi yang baru dibentuk untuk proses algoritma selanjutnya.
5. [**Tes**] Jika kondisi akhir terpenuhi, berhenti dan hasilnya adalah solusi terbaik dari populasi saat itu.

    [**Ulangi**] Ulangi langkah 2.

Beberapa kriteria berhenti yang sering digunakan antara lain :

1. Berhenti pada generasi tertentu.
2. Berhenti setelah dalam beberapa generasi berturut-turut didapatkan nilai *fitness* tertinggi tidak berubah.
3. Berhenti bila *N* generasi berikutnya tidak didapatkan nilai *fitness* yang lebih tinggi [NUR-06].

2.3.3Komponen-komponen Algoritma Genetika

2.3.3.1Pengkodean Kromosom

Pengkodean kromosom berbeda untuk setiap permasalahan yang berbeda. Proses pengkodean akan menghasilkan suatu deretan yang disebut kromosom. Sebuah kromosom terdiri dari sekumpulan bit yang disebut gen. Satu gen umumnya akan mewakili satu variabel [LUK-05].

Pada Skripsi ini akan digunakan pengkodean bilangan real. Pada pengkodean ini, nilai gen berada dalam interval [0,R], dimana R adalah bilangan real positif dan biasanya nilai R=1 [SUY-05]. Selain itu, pada penyusunan kromosom ini, ketika semua gen dijumlahkan total nilainya harus sama dengan 1. Hal ini khusus

untuk metode yang diterapkan pada skripsi ini yaitu metode GA/KNN yang akan dijelaskan kembali pada subbab 2.4 [PRO-09].

Kromosom 1

0.13	0.04	0.09	0.01	0.14	0.1	0.01	0.19	0.02	0.001
0.006	0.009	0.045	0.023	0.065	0.045	0.002	0.003	0.011	0.06

Kromosom 2

0.21	0.001	0.034	0.005	0.001	0.006	0.15	0.11	0.021	0.009
0.1	0.001	0.11	0.022	0.049	0.033	0.034	0.012	0.002	0.09

2.3.3.2Fungsi Fitness

Fungsi *fitness* merupakan fungsi yang digunakan untuk menghitung *fitness* atau tingkat kebaikan suatu individu untuk bertahan hidup. Fungsi ini mengambil parameter berupa individu dan menghasilkan keluaran nilai *fitness* dari individu tersebut. Untuk setiap permasalahan yang akan diselesaikan dengan algoritma genetika harus terdefinisi suatu *fitness function* [NUR-07].

Pada permasalahan yang berbeda fungsi *fitness* yag digunakan juga berbeda. Pada algoritma genetika sederhana fungsi *fitness* akan menghasilkan nilai *fitness* yang mempunyai *range* antara 0 dan 1 [KUS-03]. Pada skripsi ini menggunakan fungsi fitness sebagai berikut :

$$F = \frac{TotalTest - Incorrect}{TotalTest} \quad (2.6)$$

Total Test : jumlah data yang diuji

Incorrect : jumlah data yang salah diklasifikasi

2.3.4Operator Genetika

1. Seleksi induk (*parent*)

Proses pemilihan dua individu sebagai orang tua biasanya dilakukan secara proporsional berdasarkan nilai-nilai *fitness*-nya. Salah satu metode seleksi yang umum digunakan adalah ***Roulette-Wheel***. Sesuai dengan namanya, metode ini menirukan permainan *roulette-wheel* dimana masing-masing individu menempati potongan lingkaran pada roda roulette secara proporsional sesuai dengan nilai *fitness*-nya [SUY-05]. Kromosom yang memiliki nilai fitness lebih besar

menempati potongan lingkaran yang lebih besar dibandingkan dengan kromosom bernilai fitness rendah [SAN-06].

Langkah-langkah seleksi *Roulette Wheel* adalah sebagai berikut [KUS-03]:

1. Hitung *fitness* (*Fk*) setiap individu pada suatu populasi (k=1,2,3,…,n) sesuai persamaan 2.3.
2. Hitung jumlah total *fitness* dari semua individu pada populasi (k=1,2,3,…,n)

$$totFitness = \sum Fk \ (\mathbf{2.7})$$

3. Hitung peluang (*Pk*) dari setiap individu yang ada (k=1,2,3,…n)

$$Pk = \frac{Fk}{totFitness} (\mathbf{2.8})$$

4. Hitung peluang kumulatif (*qk*) dari setiap individu (k=2,3,4,…n)

$$q_1 = p_1$$

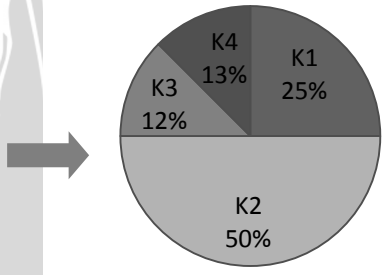
$$q_k = q_{k-1} + p_k (\mathbf{2.9})$$

5. Dibangkitkan bilangan acak *(r)* agar diketahui individu mana yang terpilih dalam proses seleksi. Nilai bilangan acak antara 0 sampai 1.
6. Jika $q_k \leq r$ dan $q_{k+1} > r$, maka pilih individu ke (k+1) sebagai kandidat induk.

Keterangan:

k	: individu ke-k
F	: nilai *fitness*
totFitness	: total *fitness*
Pk	: peluang individu ke-k
qk	: peluang kumulatif individu ke-k

Kromosom	NIlai Fitness
K1	1
K2	2
K3	0,5
K4	0,5
Jumlah	4



Gambar 2.1 Contoh penggunaan metode seleksi roulette-wheel [SAN-06]

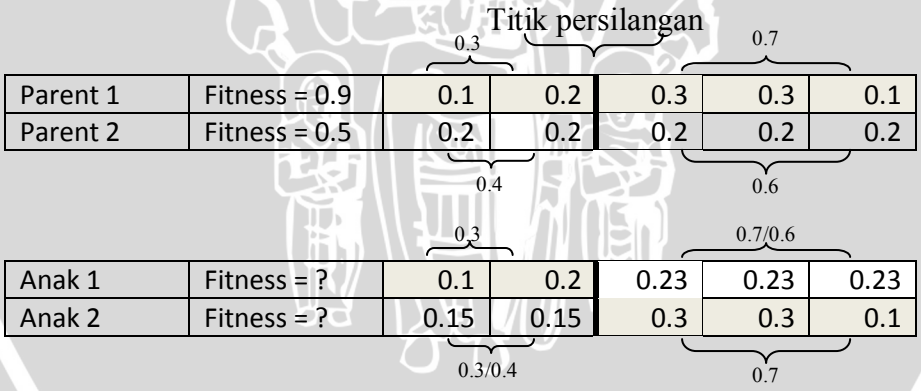
2. Persilangan (*Crossover*)

Proses persilangan berfungsi untuk menghasilkan keturunan dari dua buah kromosom induk yang terpilih. Kromosom anak yang dihasilkan merupakan kombinasi gen-gen yang dimiliki oleh kromosom induk [KUS-03]. Pada skripsi

ini menggunakan metode ***Restricted Crossover***. Menurut Provorova (2009), biasanya setelah persilangan individu, jumlah (total bobot) dari kromosom tersebut mungkin berubah, dan untuk menjaga jumlah nilai bobot, operator crossover memiliki beberapa modifikasi yaitu :

- Memasangkan secara acak 2 individu
- Memilih titik *crossover* secara acak
- Menghitung jumlah bobot dari individu yang terletak sebelum dan sesudah titik *crossover*
- Menggeser kebalikan dari masing-masing bagian dari pasangan individu pada titik *crossover*
- Membandingkan jumlah nilai bobot yang terletak di bagian yang digeser dari individu lain sebelum dan sesudah *crossover*. Dalam hal ini jika jumlahnya berbeda, bagian yang termasuk individu dengan fitness terburuk akan diperbaiki (setelah *crossover* selisih nilai akan dikurangkan atau ditambahkan ke bagian individu dengan fitness terburuk yang terletak sebelum atau sesudah titik *crossover*).

Contoh *Restricted Crossover* dapat dilihat pada Gambar 2.2



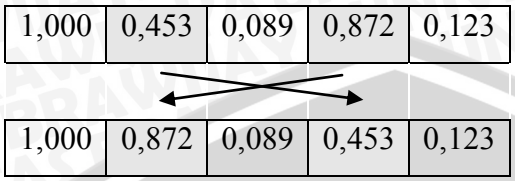
Gambar 2.2 Contoh penggunaan metode *Restricted Crossover*

3. Mutasi

Mutasi sangat jarang dalam meningkatkan keefektifan Algoritma Genetika. Sebabnya mutasi membentuk konfigurasi-konfigurasi genetik baru yang memperlebar kemungkinan dalam menemukan solusi optimal dan pencegahan konvergensi prematur[OBI-98].

Pada Skripsi ini mutasi yang digunakan adalah ***Exchange Mutation***. Proses mutasi ini dilakukan dengan cara memilih dua gen secara acak kemudian posisi gen pertama ditukar dengan posisi gen kedua [YEN-08].

Contoh *Exchange Mutation* dapat dilihat pada Gambar 2.3



Gambar 2.3 Contoh penggunaan metode Exchange Mutation

2.3.5 Parameter Genetika

Parameter genetik berguna dalam pengendalian operator–operator genetik yang digunakan. Parameter yang sering digunakan adalah sebagai berikut :

1. Ukuran Populasi

   Ukuran populasi menyatakan berapa banyak individu (solusi) pada satu generasi, jika terlalu sedikit individu, algoritma genetika mempunyai beberapa kemungkinan untuk melakukan persilangan dan hanya bagian kecil yang ditemukan. Sebaliknya, jika ada terlalu banyak individu, algoritma genetika menjadi lambat [TRO-00].

2. Peluang Persilangan

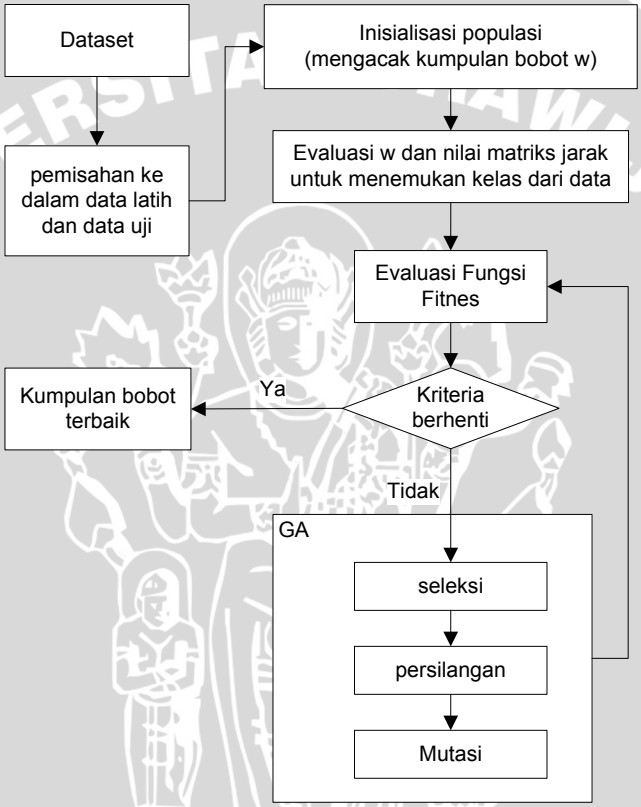
   Peluang persilangan berfungsi untuk menentukan seberapa sering proses c*rossover* dilakukan [OBI-98]. Tidak semua pasangan induk mengalami proses persilangan, banyaknya pasangan induk yang mengalami persilangan ditentukan dengan nilai peluang persilangan *(Pc)* [ALI-10].

3. Peluang Mutasi

   Peluang mutasi akan mengendalikan operator mutasi pada setiap generasi. Jika peluang mutasi terlalu kecil, banyak gen yang mungkin berguna tidak dievaluasi, tetapi bila peluang mutasi ini terlalu besar maka akan terlalu banyak gangguan acak, sehingga anak akan kehilangan kemiripan dari induknya dan algoritma juga akan kehilangan kemampuan untuk belajar dari *history* pencarian [KUS-03].

2.4 Genetic Algorithm K-Nearest Neighbour (GA/KNN)

Algoritma GA/KNN berdasarkan pada dua algoritma yang memiliki tujuan masing-masing. Algoritma K-Nearest Neighbour mengklasifikasikan suatu hal menggunakan jarak diantaranya dan Algoritma Genetika mencari bobot 'terbaik' yang membantu untuk mengurangi kesalahan klasifikasi yaitu kumpulan bobot yang diterapkan untuk meningkatkan atau mengurangi kepentingan jarak diantara setiap nilai atribut dari data latih dan nilai atribut sama yang meragukan [PRO-09]. Struktur dari GA/KNN ditunjukkan pada Gambar 2.4



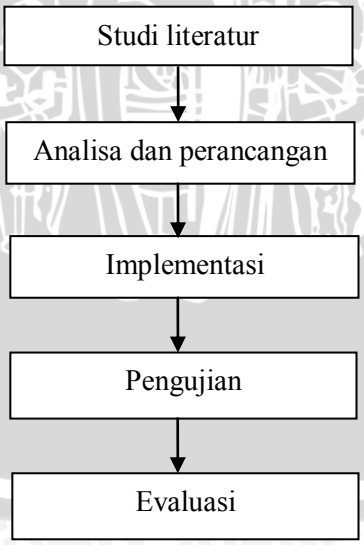
Gambar 2.4Struktur dari Algoritma GA/KNN

BAB III
METODOLOGI DAN PERANCANGAN

Pada bab metodologi dan perancangan ini akan dibahas metode, rancangan yang digunakan, dan langkah-langkah yang dilakukan dalam penelitian untuk mengklasifikasikan kredit dengan menggunakan *Genetic Algorithm-K-Nearest Neighbour*. Penelitian dilakukan dengan tahapan sebagai berikut :

1. Mempelajari metode yang digunakan dari berbagai sumber seperti yang telah dijelaskan pada bab 2.
2. Menganalisa dan merancang perangkat lunak dengan metode yang akan digunakan.
3. Membuat perangkat lunak berdasarkan analisis dan perancangan yang dilakukan.
4. Uji coba klasifikasi data kredit menggunakan perangkat lunak yang telah dibuat.
5. Evaluasi hasil uji coba.

Langkah-langkah penelitian ini ditunjukkan oleh Gambar 3.1



Gambar 3.1Tahapan penelitian

3.1 Deskripsi Data

Dataset yang akan digunakan dalam skripsi ini adalah *dataset GermanCredit*. Kumpulan data tersebut diperoleh dari situs UCI *Machine Learning Repository*(http://archive.ics.uci.edu/ml/datasets/Credit)dan terdiri dari20atribut. Atribut-atribut tersebut meliputi :

1. Status laporan pengecekan yang ada *(Status of existing checking account)*
2. Lamanya dalam bulan *(Duration in month)*
3. Sejarah kredit *(Credit history)*
4. Kegunaan *(Purpose)*
5. Jumlah Kredit *(Credit amount)*
6. Tabungan/Surat obligasi *(Savings account/bonds)*
7. Pekerjaan sekarang sejak *(Present employment since)*
8. Tarif angsuran dalam prosentase dari pendapatan bersih setelah pajak *(Installment rate in percentage of disposable income)*
9. Status pribadi dan jenis kelamin *(Personal status and sex)*
10. Debitor / penjamin lain *(Other debtors / guarantors)*
11. Tepat tinggal sekarang sejak *(Present residence since)*
12. Kekayaan *(Property)*
13. Umur dalam tahun *(Age in years)*
14. Rencana angsuran yang lain *(Other installment plans)*
15. Tempat tinggal *(Housing)*
16. Jumlah kredit yang ada di bank ini *(Number of existing credits at this bank)*
17. Pekerjaan *(Job)*
18. Jumlah orang yang menjadi tanggung jawab untuk menyediakan biaya hidup untuk *(Number of people being liable to provide maintenance for)*
19. Telepon *(Telephone)*
20. Pekerja asing *(foreign worker)*

Atribut ke-1 sampai 20 ada yang bertipe data numerik dan ada yang bertipe kategori. Daftar tipe *dataset*kredittersebut dapat dilihat pada Tabel 2.1. *Dataset* ini dikelompokkan menjadi 2 kelas yaitu kelas Baik *(Good)* yang disimbolkan dengan angka 1 dan Buruk *(Bad)* yang disimbolkan dengan angka 2.

Dataset kredit ini terdiri dari 1000*record*. Dari 1000*record* tersebut, terdapat rincian sebagai berikut :

- Kelas Baik (*Good Risk*) sebanyak700 *record*
- Kelas Buruk (*Bad Risk*) sebanyak 300 *record*

3.2Deskripsi Sistem

Sistem yang dibuat merupakan perangkat lunak yang menerapkan *Genetic Algorithm K-Nearest Neighbour*(GA/KNN)untuk klasifikasi kredit.GA (*Genetic Algorithm*) digunakan untuk menentukan bobot optimal pada tiap-tiap atribut. Bobot tersebut akan digunakan untuk proses klasifikasi dengan mencari k tetangga terdekat.

Sistem yang dibuat memiliki keterbatasan sebagai berikut :

1. Jenis file data latih,dan data uji yang digunakan oleh sistem berasal dari file dengan ekstensi *.csv.
2. Kelas data latih dan data uji berada pada kolom terakhir.
3. Nilai k (tetangga), parameter genetika seperti ukuran populasi, generasi maksimal, Probabilitas*crossover* dan Probabilitas mutasi dimasukkan secara manual oleh *user*.

3.3 Perancangan Sistem

Subbab ini menjelaskan tentang proses-proses yang dilakukan oleh sistem. Alur proses dari sistem ditampilkan oleh Gambar 3.2

Berdasarkan Gambar 3.2, proses pada sistem meliputi 4 proses utama, yaitu :

1. Proses normalisasi data

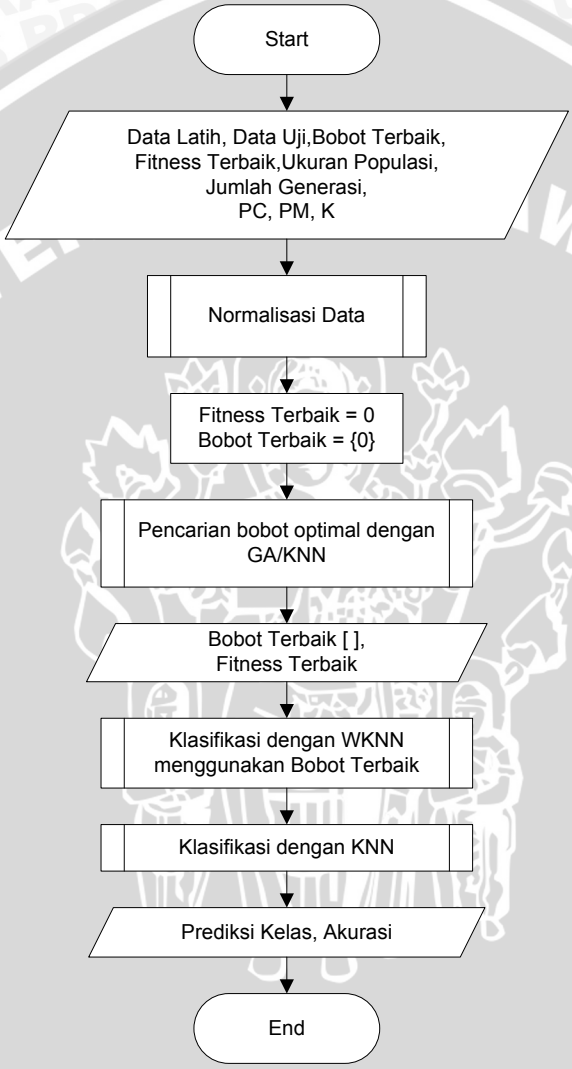
   Merupakan proses untuk menormalisasi data sebelum dibagi menjadi data latih dan data uji. Proses normalisasi ini menggunakan metode *MinMax Normalization*.Proses ini dijelaskan pada subbab 3.3.1.
2. Proses pencarian bobot optimal menggunakan GA/KNN

   Merupakan proses untuk menentukan bobot pada tiap-tiap atribut menggunakan GA. Proses ini dijelaskan pada subbab 3.3.2.
3. Proses klasifikasi menggunakan WKNN dengan bobot optimal hasil pencarian dengan GA/KNN

Merupakan proses klasifikasi data uji menggunakan WKNN dengan bobot hasil pencarian dan voting terhadap k tetangga terdekat. Proses ini dijelaskan pada subbab 3.3.3.

4. Proses klasifikasi menggunakan KNN

   Merupakan proses klasifikasi data uji menggunakan KNN murnisebagai pembanding untuk metode GA/KNN.



Gambar 3.2*Flowchart*proses klasifikasi kredit secara umum

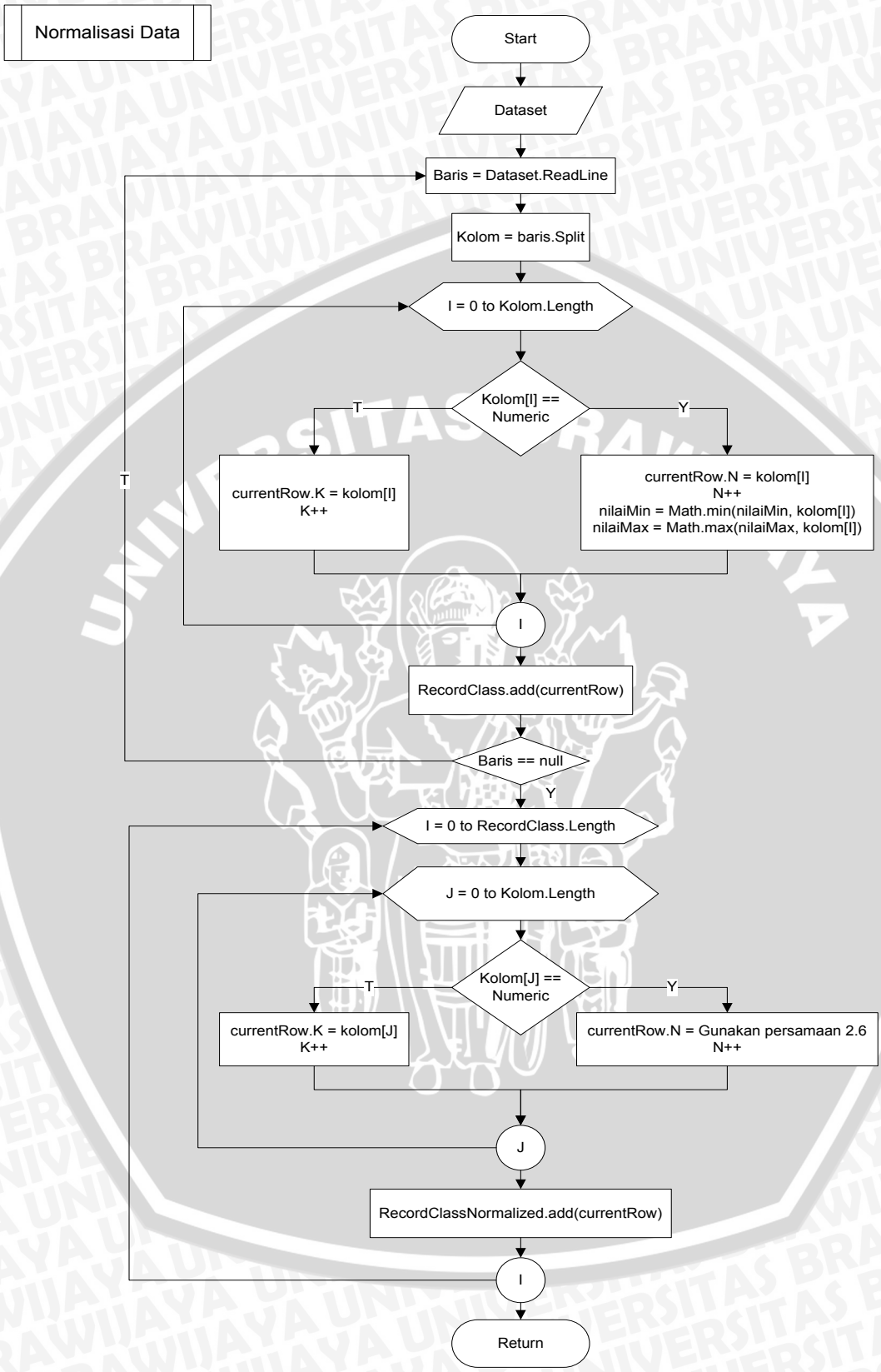
3.3.1 Normalisasi Data

Proses normalisasi data menggunakan metode *Minmax Normalization* dilakukan dengan langkah-langkah sebagai berikut:

1. Membaca dataset perbaris dan memisahkannya per-kolom-kolom.

2. Untuk kolom-kolom yang terdiri dari data numerik akan diambil nilai minimum dan nilai maksimumnya untuk proses normalisasi.
3. Dataset yang terbagi atas baris dan kolom disimpan ke dalam record class.
4. Data-data numerik di dalam record class dinormalisasi dengan persamaan 2.1.

Flowchart untuk proses normalisasi data ditampilkan pada Gambar 3.3.



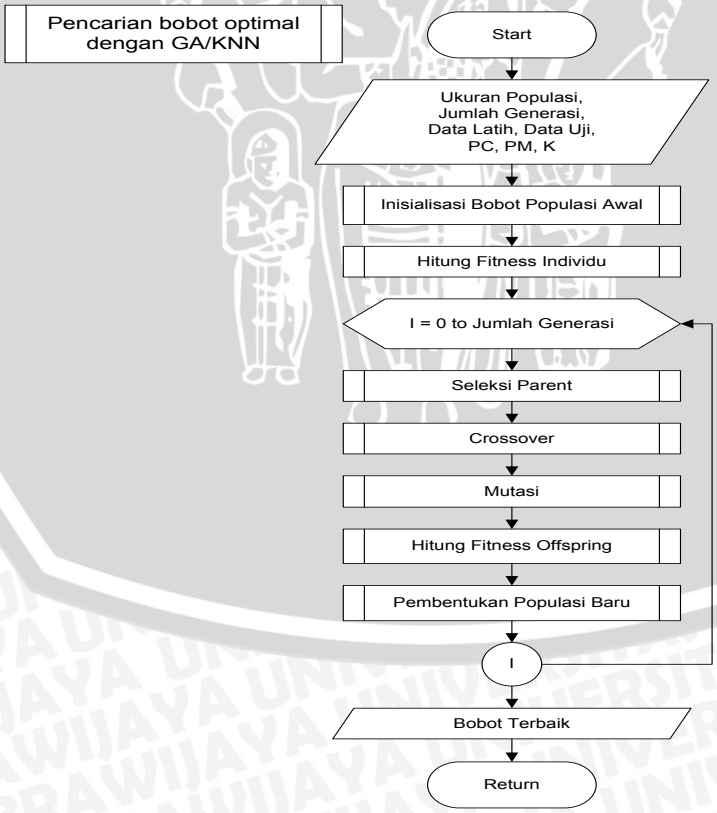
Gambar 3.3*Flowchart* normalisasi data menggunakan *Minmax Normalization*

3.3.2Pencarian Bobot Optimal dengan GA/KNN

Proses pencarian bobot menggunakan GA dilakukan dengan langkah-langkah sebagai berikut :

1. Menentukan parameter awal yang akan digunakan pada proses genetik, yaitu ukuran populasi, Generasi maksimalProbabilitas*crossover*, mutasi dan nilai k serta data latih dandata uji.
2. Membangkitkan populasi awal sesuai ketentuan yang ditetapkan sebanyak ukuran populasi yang telah ditentukan.
3. Mengevaluasi nilai *fitness* masing-masing kromosom pada populasi.
4. Mengecek kriteria berhenti, jika belum sesuai akan dilakukan proses GA yang akan membangkitkan populasi baru, atau jika sudah sesuai maka akan langsung didapatkan kromosom terbaik sebagai bobot optimal.
5. Menggunakan populasi baru untuk melakukan kembali langkah no.4 hingga jumlah generasi terpenuhi.

Flowchart untuk proses pencarian bobot menggunakan GA ditampilkan oleh Gambar 3.4.



Gambar 3.4*Flowchart*pencarian bobot menggunakan GA/KNN

3.3.2.1 InisialisasiPopulasi

Kromosom dalam Populasi yang digunakan, diinisialisasi menggunakan pengkodean real dengan range 0 sampai1yang mewakili masing-masing bobot setiap atribut kredit sehingga panjang kromosom sebanyak jumlah atribut yang ada yaitu20gen. Gen–gen dalam 1 kromosom tersebut apabila dijumlahkan harus bernilai 1. Representasi kromosom yang dihasilkan seperti berikut ini :

Kromosom 1

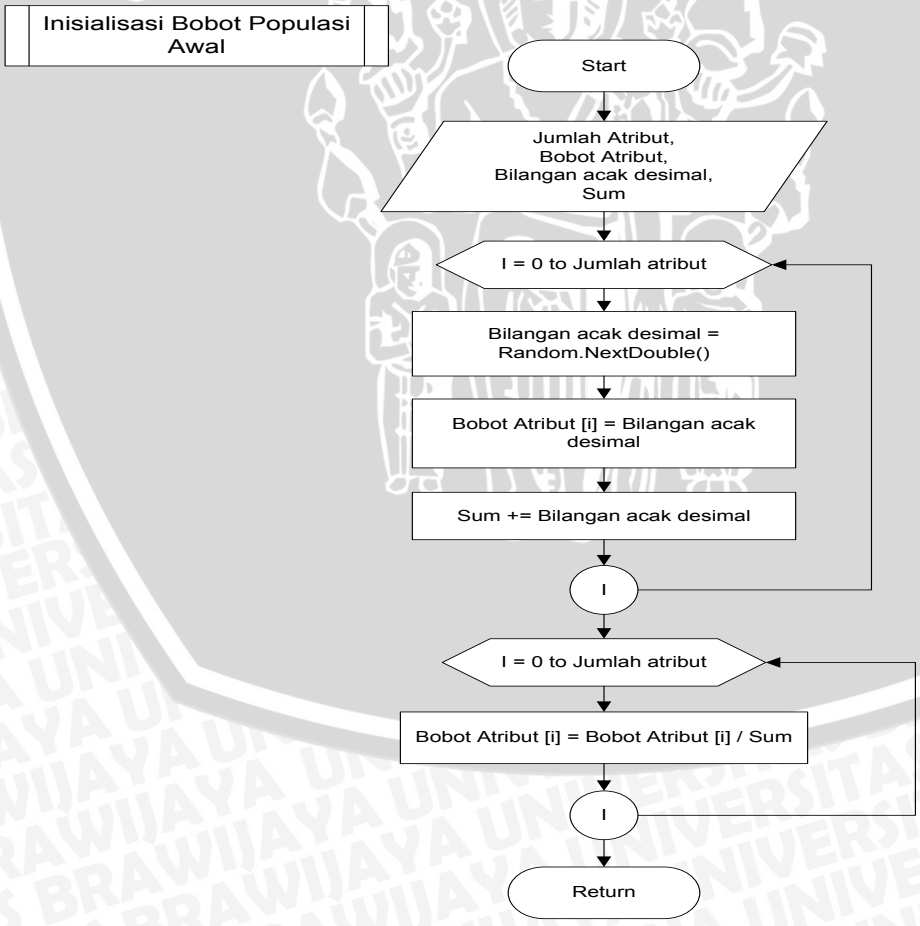
0.13	0.04	0.09	0.01	0.14	0.1	0.01	0.19	0.02	0.001
0.006	0.009	0.045	0.023	0.065	0.045	0.002	0.003	0.011	0.06

Total dari penjumlahan semua gen dalam 1 kromosom

= 0.13 + 0.04 + 0.09 + 0.01 + 0.14 + 0.1 + 0.01 + 0.19 + 0.02 + 0.001+ 0.006 +0.009+ 0.045 + 0.023 + 0.065 + 0.045 + 0.002 + 0.003 + 0.011 + 0.06

= 1

Flowchart untuk proses inisialisasi populasi ditampilkan oleh Gambar 3.5.



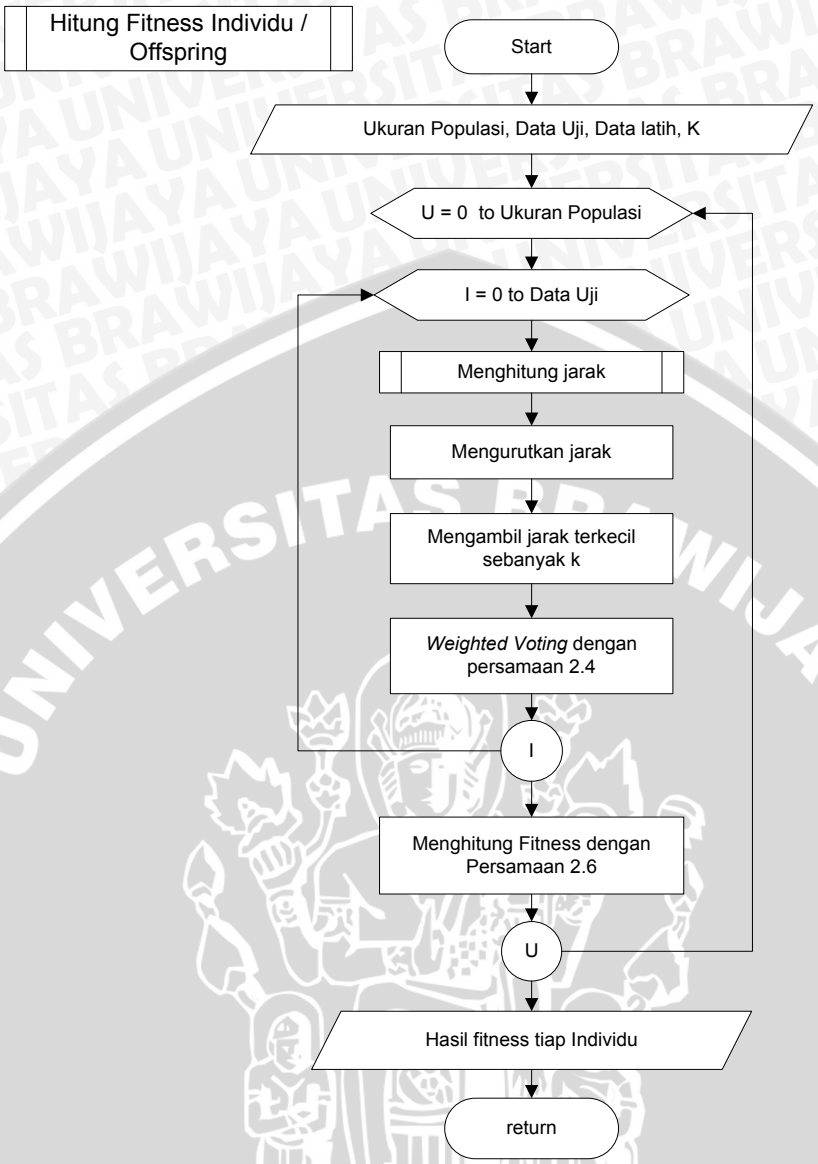
Gambar 3.5*Flowchart*proses inisialisasi populasi

3.3.2.2Evaluasi Fitness

Proses penghitungan nilai fitness dilakukan dengan langkah-langkah sebagai berikut:

1. Memasukkanpopulasi, data latih, dan data uji serta nilai k.
2. Pada populasi dari kromosom pertama sampai kromosom terakhir akan dilakukan :
   - Untuk setiap data uji dari data uji pertama sampai data uji terakhir akan dilakukan:
     a) Menghitung jarak pada semua data latih terhadap sebuah kromosom yang akan dijelaskan selanjutnya.
     b) Mengurutkan jarak
     c) Mengambil sejumlah record dengan jarak terkecil sebanyak k (tetangga terdekat)
     d) Melakukan proses prediksi kelas dengan *Weighted Voting* dengan persamaan 2.4
3. Menghitung *fitness* dengan persamaan 2.6
4. Jika Kromosomterakhir telah selesai diproses maka proses hitung *fitness* berhenti.

Flowchart untuk proses perhitungan nilai fitnessditampilkan oleh Gambar 3.6.

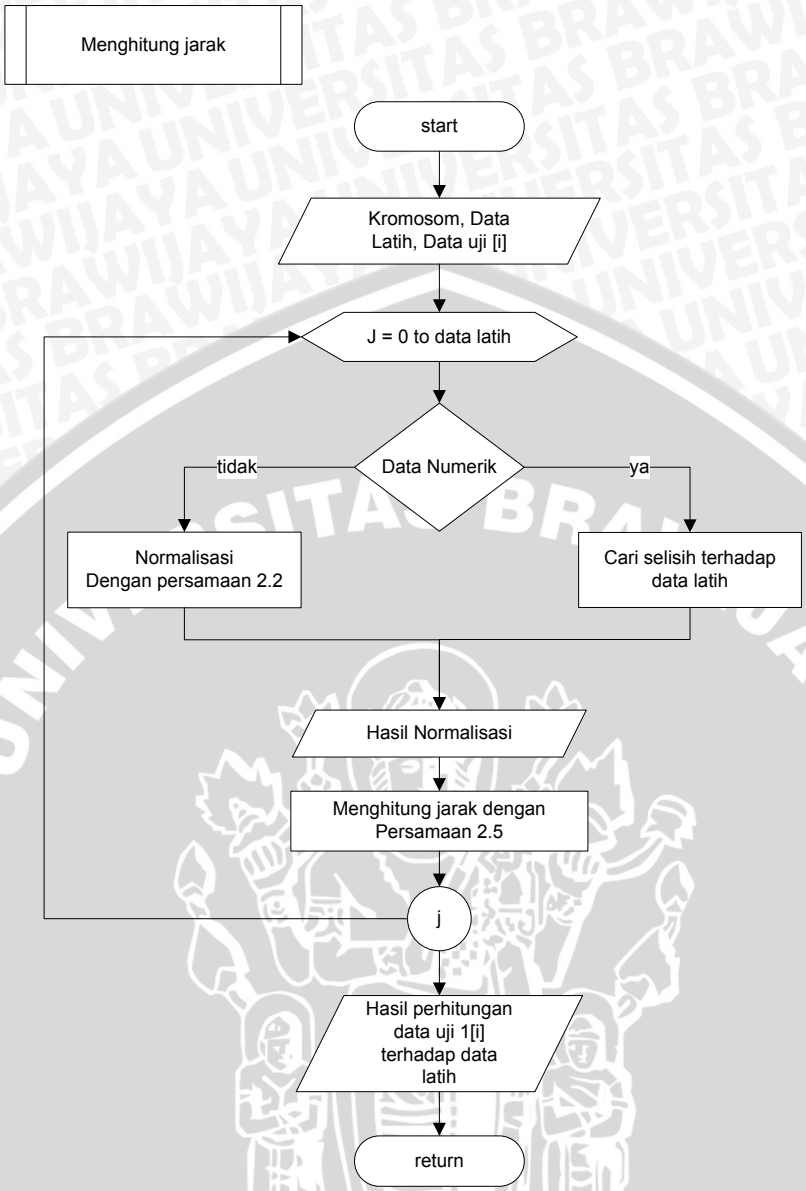


Gambar 3.6*Flowchart* proses evaluasi *fitness*

Proses menghitung jarak dilakukan dengan langkah-langkah sebagai berikut :

1. Memasukkan kromosom, datalatih, dan data uji [i]
2. Untuk setiap data latih dari data latih pertama sampai data latih terakhir akan dilakukan :
    a) Memilih tipe data. Jika berupa tipe data numerik maka dicari nilai selisihnya, jika tidak, akan dinormalisasi dengan persamaan 2.2
    b) Menghitung jarak dengan persamaan 2.5

Flowchart untuk proses perhitungan jarak pada semua data latihditampilkan oleh Gambar 3.7.



Gambar 3.7*Flowchart* proses menghitung jarak

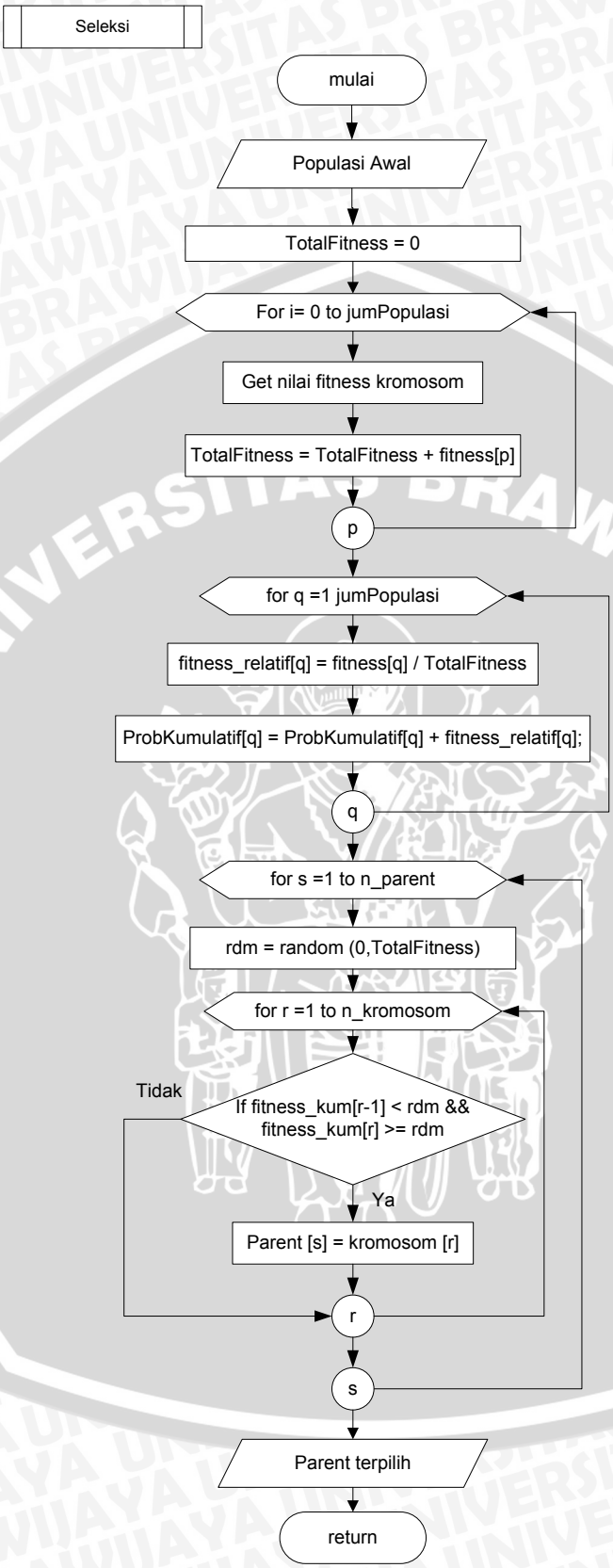
3.3.2.3 Proses Seleksi

Proses untuk menentukan *parent* dengan metode *roulette wheel* dilakukan dengan langkah-langkah sebagai berikut :

1. Mengambil nilai *fitness* masing-masing kromosom pada populasi.
2. Menjumlahkan seluruh *fitness* untuk mendapatkan total *fitness.*
3. Menghitung peluangsetiap individudengan membagi nilai *fitness* dengan total *fitness.*

4. Menghitungpeluangkumulatif tiap kromosom, peluang kumulatif digunakan untuk menentukan batasan daerah tiap kromosom.
5. Melakukan proses *roulette* dengan membangkitkan bilangan acak antara 0 sampai nilai total *fitness*.
6. Jika nilai random berada pada salah satu daerah kromosom, maka kromosom itu terpilih menjadi *parent* untuk melakukan persilangan.
7. Melakukan proses *roulette wheel* sebanyak jumlah *parent* yang dihitung berdasarkan peluang persilangan yang telah ditentukan sebelumnya.

Flowchart untuk proses seleksi dengan metode*roulette wheel*ditampilkan oleh Gambar 3.8.



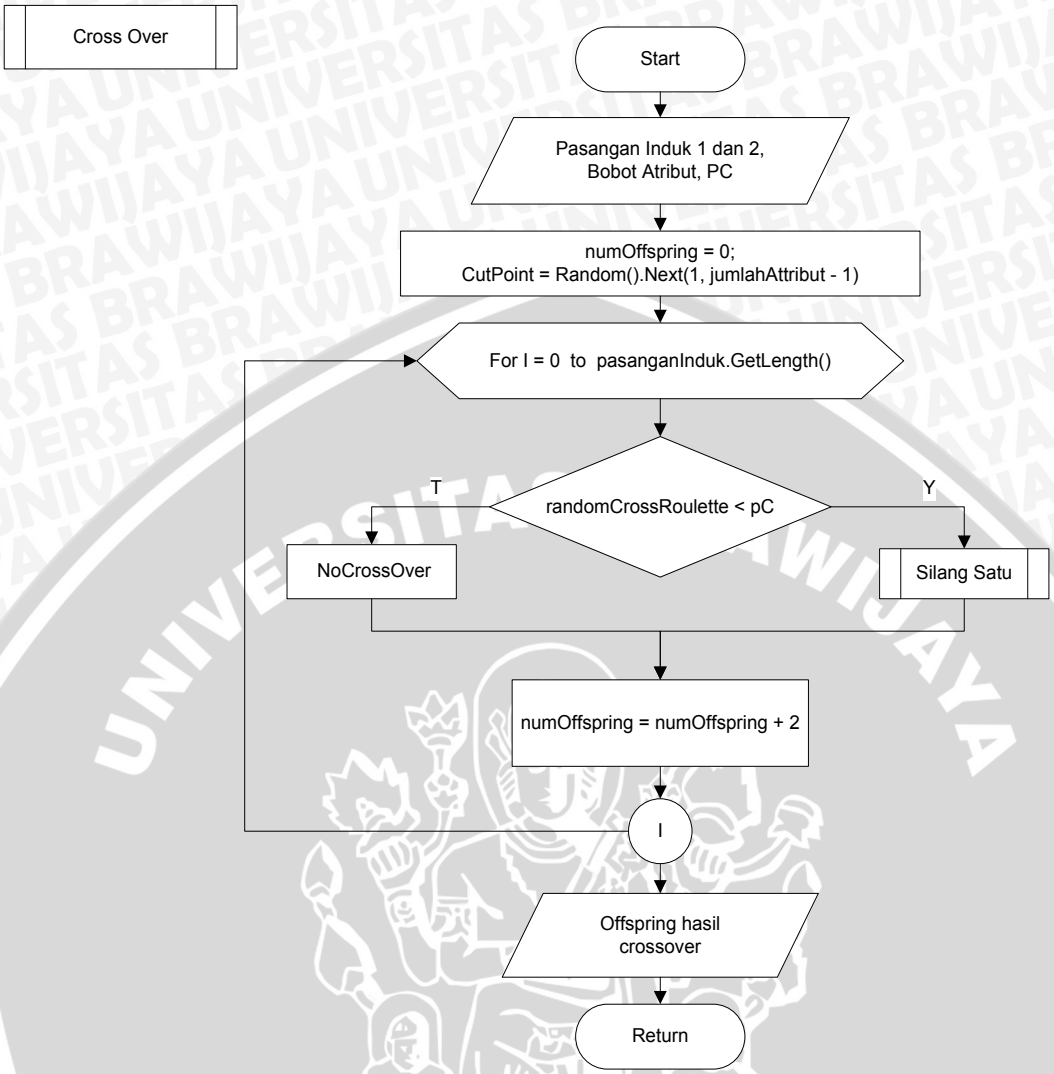
Gambar 3.8*Flowchart* Metode *Roulette Wheel*

3.3.2.4 Persilangan (Cross Over)

Proses untuk menentukan *crossover*dengan metode *Restricted Crossover* dilakukan dengan langkah-langkah sebagai berikut :

1. Memasukkan *parent* dari hasil seleksi dengan metode *Roulette Wheel* dan Pc (Peluang Crossover)
2. Mengacak nilai dengan rentang antara 0 dan 1 (p)
3. Apabila nilai p lebih kecil dari Pc maka akan dilakukan persilangan, jika tidak maka tidak dilakukan persilangan.
4. Proses persilangan dilakukan dari kromosom pertama sampai kromosom terakhir yang mengalami persilangan
5. Menentukan titik potong atau *cut point* yang dilakukan secara acak dengan rentang antara 1 dan jumlah gen.
6. Menghitung jumlah bobot sebelum dan sesudah *cut point* pada kedua *parents*.
7. Menukar pasangan gen sampai batas titik potong yang ditentukan.
8. Untuk gen yang berasal dari *parent* yang memiliki nilai *fitness* lebih rendah, akan mengalami perbaikan bobot sesuai skala jumlah bobot dari parent ber-*fitness* tinggi dibanding jumlah bobot dari parent ber-*fitness* rendah pada gen-gen sebelum dan sesudah titik *crossover*.
9. Proses berhenti setelah kromosom mencapai batas akhir.

Flowchart untuk proses *crossover*ditampilkan oleh Gambar 3.9.



Gambar 3.9*Flowchart* proses Restricted Crossover

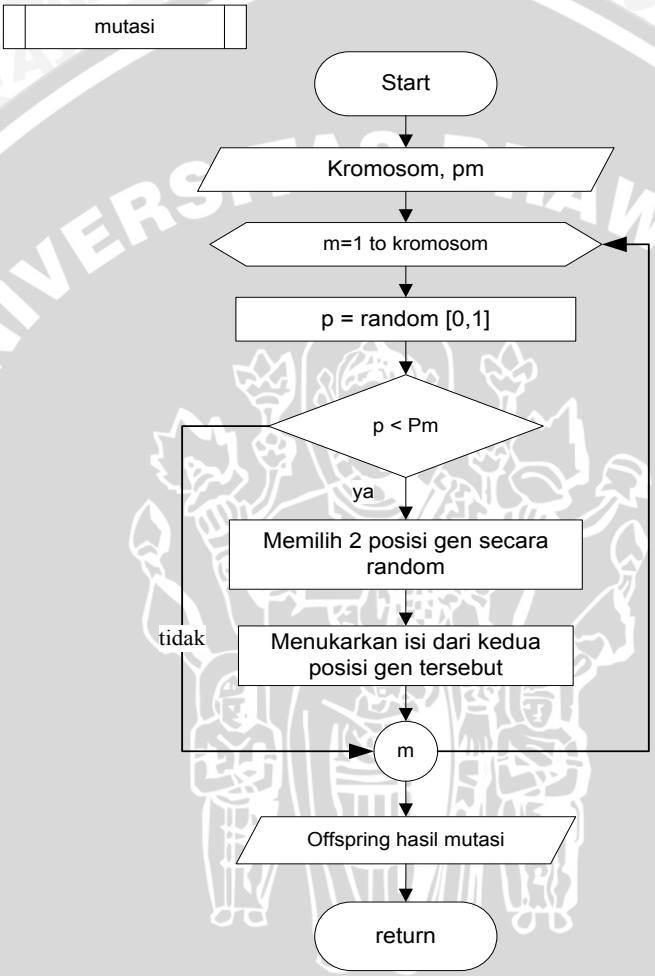
3.3.2.5 Mutasi

Proses mutasi dengan metode *Exchange Mutation* dilakukan dengan langkah-langkah sebagai berikut :

1. Memasukkan kromosom dan nilai Pm (probabilitas mutasi)
2. Mengacak nilai dengan rentang antara 0 dan 1 (p)
3. Apabila nilai p lebih kecil dari Pm maka akan dilakukan mutasi, jika tidak maka tidak dilakukan mutasi
4. Proses Mutasi akan dilakukan dari kromosom pertama sampai kromosom terakhir yang mengalami mutasi

5. Memilih 2 posisi gen secara random pada setiap kromosom yang mengalami mutasi
6. Menukarkan isi dari kedua posisi gen tersebut
7. Proses berhenti jika telah sampai pada kromosom terakhir yang akan dimutasi.

Flowchart untuk proses mutasiditampilkan oleh Gambar 3.10.



Gambar 3.10 Flowchart proses *Exchange Mutation*

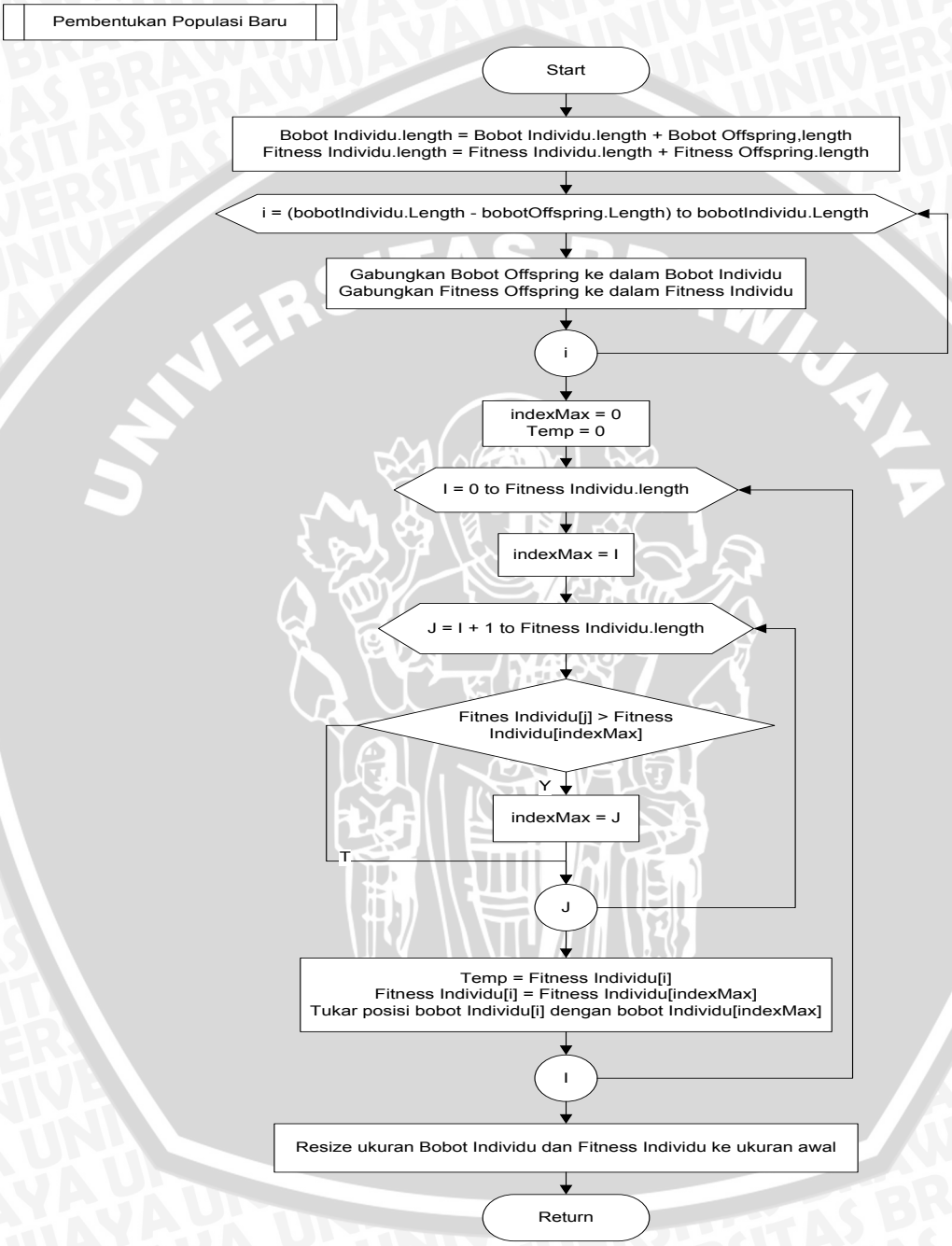
3.3.2.6 Pembentukan Populasi Baru

Proses pembentukan populasi baru dilakukan dengan langkah-langkah sebagai berikut :

1. Menggabungkan vektor bobot dari offspring dan parent, diikuti penggabungan nilai fitness dari offspring dan parent.
2. Mengurutkan vektor bobot berdasarkan nilai fitnessnya.

3. Mengambil sejumlah vektor bobot terbaik sebanyak ukuran populasi sebagai parent untuk digunakan pada siklus berikutnya.

Flowchart untuk proses pembentukan populasi baruditampilkan oleh Gambar 3.11.



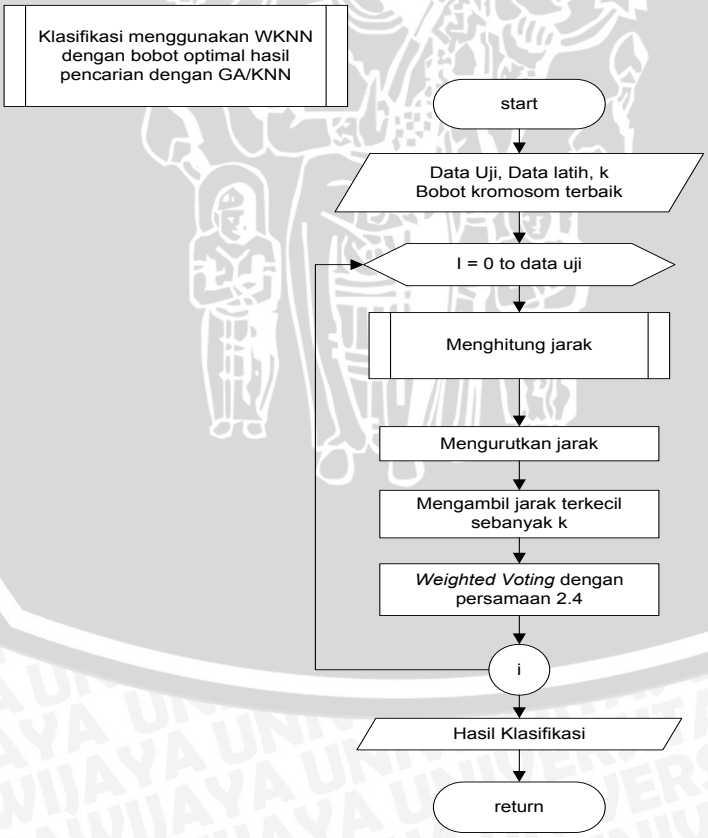
Gambar 3.11 Flowchart proses pembentukan populasi baru

3.3.3 Proses Klasifikasi dengan WKNN

Proses klasifikasi dengan WKNN dilakukan dengan langkah – langkah sebagai berikut:

1. Memasukkan data latih, data uji, nilai k (ketetanggaan) serta kromosom (bobot) terbaik yang telah berhasil ditemukan.
2. Untuk setiap data uji dari data uji pertama sampai data uji terakhir akan dilakukan :
   a. Menghitung jarak
   b. Mengurutkan jarak
   c. Mengambil sejumlah record dengan jarak terkecil sebanyak k (tetangga terdekat)
   d. Melakukan proses prediksi kelas dengan *Weighted Voting* dengan persamaan 2.4
3. Keluaran (output) merupakan hasil dari klasifikasi.

Proses klasifikasi dengan WKNN ditampilkan oleh Gambar 3.12.



Gambar 3.12 Flowchart Proses Klasifikasi Menggunakan WKNN dengan bobot optimal hasil pencarian dengan GA/KNN

3.4Perhitungan Manual

Perhitungan manual ini menggunakan sampel data yang diambil dari *datasetCredit* dan digunakan seluruh atribut *dataset* tersebut yang berjumlah 20 atribut. Diambil sebanyak 20 *record* dengan rincian : *record* ke-1 sampai 15 merupakan data latih dan *record* ke-16 sampai 20 merupakan data uji. Atribut yang digunakan adalah :

1. Status laporan pengecekan yang ada *(Status of existing checking account)*
2. Lamanya dalam bulan *(Duration in month)*
3. Sejarah kredit *(Credit history)*
4. Kegunaan *(Purpose)*
5. Jumlah Kredit *(Credit amount)*
6. Tabungan/Surat obligasi *(Savings account/bonds)*
7. Pekerjaan sekarang sejak *(Present employment since)*
8. Tarif angsuran dalam prosentase dari pendapatan bersih setelah pajak *(Installment rate in percentage of disposable income)*
9. Status pribadi dan jenis kelamin *(Personal status and sex)*
10. Debitor / penjamin lain *(Other debtors / guarantors)*
11. Tepat tinggal sekarang sejak *(Present residence since)*
12. Kekayaan *(Property)*
13. Umur dalam tahun *(Age in years)*
14. Rencana angsuran yang lain *(Other installment plans)*
15. Tempat tinggal *(Housing)*
16. Jumlah kredit yang ada di bank ini *(Number of existing credits at this bank)*
17. Pekerjaan *(Job)*
18. Jumlah orang yang menjadi tanggung jawab untuk menyediakan biaya hidup untuk *(Number of people being liable to provide maintenance for)*
19. Telepon *(Telephone)*
20. Pekerja asing *(foreign worker)*

Contoh *dataset* Kredit dapat dilihat pada Tabel 3.1. Sedangkan keterangan mengenai data kredit ini dapat dilihat pada Tabel 2.1.

Tabel 3.1 Data kredit

	Atribut kredit ke -																				kelas
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
1	A14	48	A34	A43	618	A61	A75	4	A93	A101	4	A121	56	A143	A152	1	A173	1	A191	A201	1
2	A13	10	A32	A43	625	A61	A72	4	A94	A103	1	A121	26	A141	A152	1	A172	1	A191	A201	1
3	A14	30	A34	A43	629	A63	A75	4	A93	A101	3	A122	32	A141	A152	2	A174	1	A192	A201	1
4	A12	18	A32	A40	640	A61	A73	4	A91	A101	2	A121	49	A143	A152	1	A172	1	A191	A201	1
5	A11	30	A32	A42	652	A61	A75	4	A92	A101	4	A122	24	A143	A151	1	A173	1	A191	A201	1
6	A14	36	A32	A45	660	A63	A74	2	A94	A101	4	A121	23	A143	A151	1	A172	1	A191	A201	1
7	A11	6	A32	A40	662	A61	A72	3	A93	A101	4	A121	41	A143	A152	1	A172	2	A192	A201	1
8	A14	9	A32	A43	433	A61	A71	3	A92	A102	4	A121	22	A143	A151	1	A173	1	A191	A201	2
9	A11	6	A31	A46	433	A64	A72	4	A92	A101	2	A122	24	A141	A151	1	A173	2	A191	A201	2
10	A11	10	A32	A46	448	A61	A72	4	A92	A101	4	A122	23	A143	A152	1	A173	1	A191	A201	2
11	A13	12	A31	A49	609	A61	A72	4	A92	A101	1	A121	26	A143	A152	1	A171	1	A191	A201	2
12	A12	9	A31	A43	626	A61	A73	4	A92	A101	4	A121	24	A141	A152	1	A172	1	A191	A201	2
13	A12	12	A32	A45	639	A61	A73	4	A93	A101	2	A123	30	A143	A152	1	A173	1	A191	A201	2
14	A14	11	A32	A40	654	A61	A73	4	A93	A101	3	A123	28	A143	A152	1	A172	1	A191	A201	2
15	A12	24	A32	A43	674	A62	A74	4	A94	A101	1	A122	20	A143	A152	1	A173	1	A191	A201	2
16	A12	21	A32	A40	2782	A63	A74	1	A92	A101	2	A123	31	A141	A152	1	A174	1	A191	A201	1
17	A14	24	A34	A42	2788	A61	A74	2	A92	A102	3	A123	24	A141	A152	2	A173	1	A191	A201	1
18	A11	6	A33	A42	1808	A61	A74	4	A92	A101	1	A121	22	A143	A152	1	A173	1	A191	A201	2
19	A14	6	A32	A46	1819	A61	A73	4	A93	A101	4	A124	37	A142	A153	1	A173	1	A192	A201	2
20	A14	24	A32	A43	1823	A61	A71	4	A93	A101	2	A123	30	A142	A152	1	A174	2	A191	A201	2

Keterangan warna:

: Data latih

: Data uji

3.4.1 Inisialisasi Populasi Awal

Inisialisasi populasi awal dilakukan dengan membangkitkan bilangan random antara 0 dan 1. Banyaknya bilangan random ini sebanyak jumlah gen dalam satu kromosom. Kemudian dibuat lagi sebanyak jumlah populasi yang akan dibentuk. Semua bilangan random dalam satu kromosom ini apabila dijumlahkan harus sebesar 1. Misalnya telah dibangkitkan kromosom sepanjang 20 (atribut) pada sebuah populasi sebanyak 4 kromosom sebagai berikut :

Kromosom 1

0.033	0.014	0.083	0.032	0.038	0.077	0.005	0.065	0.088	0.045
0.002	0.078	0.096	0.049	0.079	0.002	0.095	0.028	0.067	0.023

Kromosom 2

0.031	0.094	0.063	0.017	0.079	0.017	0.085	0.007	0.026	0.02
0.056	0.038	0.039	0.063	0.047	0.076	0.075	0.022	0.051	0.094

Kromosom 3

0.052	0.066	0.043	0.029	0.031	0.039	0	0.028	0.107	0.104
0.099	0.094	0.003	0.066	0.033	0.058	0.035	0.061	0.008	0.044

Kromosom 4

0.007	0.067	0.004	0.002	0.049	0.026	0.041	0.024	0.043	0.002
0.028	0.071	0.076	0.055	0.165	0.143	0.043	0.013	0.073	0.067

3.4.2 Perhitungan Fitness

Langkah 1(normalisasi data).

Menormalisasi semua data numerik pada dataset menggunakan persamaan 2.1 sehingga range dari semua data numerik pada data kredit di tabel 3.1 berubah menjadi bilangan real dengan [0,1] yang ditunjukkan pada tabel 3.2.

Tabel 3.2 Data kredit ternormalisasi

	Atribut kredit ke -																				kelas
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
1	A14	1	A34	A43	0.079	A61	A75	1	A93	A101	1	A121	1	A143	A152	0	A173	0	A191	A201	1
2	A13	0.095	A32	A43	0.081	A61	A72	1	A94	A103	0	A121	0.167	A141	A152	0	A172	0	A191	A201	1
3	A14	0.571	A34	A43	0.083	A63	A75	1	A93	A101	0.667	A122	0.333	A141	A152	1	A174	0	A192	A201	1
4	A12	0.286	A32	A40	0.088	A61	A73	1	A91	A101	0.333	A121	0.805	A143	A152	0	A172	0	A191	A201	1
5	A11	0.571	A32	A42	0.093	A61	A75	1	A92	A101	1	A122	0.111	A143	A151	0	A173	0	A191	A201	1
6	A14	0.714	A32	A45	0.096	A63	A74	0.333	A94	A101	1	A121	0.083	A143	A151	0	A172	0	A191	A201	1
7	A11	0	A32	A40	0.097	A61	A72	0.667	A93	A101	1	A121	0.583	A143	A152	0	A172	1	A192	A201	1
8	A14	0.071	A32	A43	0	A61	A71	0.667	A92	A102	1	A121	0.056	A143	A151	0	A173	0	A191	A201	2
9	A11	0	A31	A46	0	A64	A72	1	A92	A101	0.333	A122	0.111	A141	A151	0	A173	1	A191	A201	2
10	A11	0.095	A32	A46	0.006	A61	A72	1	A92	A101	1	A122	0.083	A143	A152	0	A173	0	A191	A201	2
11	A13	0.143	A31	A49	0.075	A61	A72	1	A92	A101	0	A121	0.167	A143	A152	0	A171	0	A191	A201	2
12	A12	0.071	A31	A43	0.082	A61	A73	1	A92	A101	1	A121	0.111	A141	A152	0	A172	0	A191	A201	2
13	A12	0.143	A32	A45	0.087	A61	A73	1	A93	A101	0.333	A123	0.278	A143	A152	0	A173	0	A191	A201	2
14	A14	0.119	A32	A40	0.094	A61	A73	1	A93	A101	0.667	A123	0.222	A143	A152	0	A172	0	A191	A201	2
15	A12	0.428	A32	A43	0.102	A62	A74	1	A94	A101	0	A122	0	A143	A152	0	A173	0	A191	A201	2
16	A12	0.357	A32	A40	0.997	A63	A74	0	A92	A101	0.333	A123	0.306	A141	A152	0	A174	0	A191	A201	1
17	A14	0.428	A34	A42	1	A61	A74	0.333	A92	A102	0.667	A123	0.111	A141	A152	1	A173	0	A191	A201	1
18	A11	0	A33	A42	0.584	A61	A74	1	A92	A101	0	A121	0.056	A143	A152	0	A173	0	A191	A201	2
19	A14	0	A32	A46	0.588	A61	A73	1	A93	A101	1	A124	0.472	A142	A153	0	A173	0	A192	A201	2
20	A14	0.429	A32	A43	0.590	A61	A71	1	A93	A101	0.333	A123	0.278	A142	A152	0	A174	1	A191	A201	2

Keterangan warna:

: Data latih

: Data uji

Menghitung fitness untuk kromosom 1 dengan menggunakandata numerik yang ternormalisasi dan untuk data kategorik akan dinormalisasi dengan menggunakan persamaan 2.2.

Data latih ke-1

A14	1	A34	A43	0.079	A61	A75	1	A93	A101
1	A121	1	A143	A152	0	A173	0	A191	A201

Datauji ke-1

A12	0.357	A32	A40	0.997	A63	A74	0	A92	A101
0.333	A123	0.306	A141	A152	0	A174	0	A191	A201

Hasil dari selisih data latih dan data uji :

1	0.643	1	1	0.918	1	1	1	1	0
0.667	1	0.693	1	0	0	1	0	0	0

Langkah 2 (menghitung jarak antar data menggunakan vektor bobot).

Menghitung jarak antara data latih 1dan data uji1 dengan persamaan 2.4. Nilai w (bobot) adalah nilai kromosom 1.

$$\boldsymbol{jarak_{11}} = \sqrt{\sum_{i=1}^{20}\left(\boldsymbol{w_i}(\boldsymbol{x_i}-\boldsymbol{y_i})\right)^2}$$

$$= \sqrt{\begin{array}{l}(0.033*1)^2+(0.014*0.643)^2+(0.083*1)^2+(0.032*1)^2+\\(0.038*0.918)^2+(0.077*1)^2+(0.005*1)^2+(0.065*1)^2+\\(0.088*1)^2+(0.045*0)^2+(0.002*0.667)^2+(0.078*1)^2+\\(0.096*0.693)^2+(0.049*1)^2+(0.079*0)^2+(0.002*0)^2+\\(0.095*1)^2+(0.028*0)^2+(0.067*0)^2+(0.023*0)^2\end{array}}$$

$$= \sqrt{0.050181}$$

$$= 0.224012$$

Dilakukan hal yang sama pada semua data latih dan data uji. Hasil perhitungan jarak ditampilkan pada Tabel.3.3.

Tabel 3.3 Perhitungan jarak data latih dengan data uji

(A = Data Latih ke- , B = DataUji ke-)

A \ B	1	2	3	4	5
1	0.224012	0.155705	0.142309	0.168225	0.165025
2	0.193051	0.189282	0.173998	0.201275	0.170993
3	0.177754	0.196049	0.212188	0.194223	0.163161

4	0.190812	0.200125	0.167182	0.196309	0.171024
5	0.190294	0.163267	0.138869	0.172114	0.185828
6	0.185191	0.217697	0.199678	0.213217	0.20321
7	0.199439	0.207211	0.177531	0.164909	0.158065
8	0.189712	0.152254	0.13332	0.176542	0.187025
9	0.203697	0.17891	0.171456	0.205381	0.215821
10	0.173266	0.146468	0.11848	0.169286	0.16834
11	0.191803	0.174593	0.13456	0.213019	0.187415
12	0.182683	0.167545	0.14309	0.21328	0.184835
13	0.174355	0.152347	0.151819	0.146565	0.119835
14	0.174609	0.176318	0.178869	0.171777	0.115256
15	0.19185	0.187438	0.169677	0.189461	0.1827

Langkah 3 (Mengambil sejumlah record dengan jarak terkecil sebanyak k (tetangga terdekat)).

Jika nilai ketetanggaan (k) = 3 maka dipilih 3 jarak terkecil pada setiap data uji. Hasilnya ditampilkan pada tabel 3.4. Kelas Data Uji berdasarkan jarak yang terpilih ditampilkan pada tabel 3.5.

Tabel 3.4 Jarak yang dipilih berdasarkan nilai k = 3

	Data uji				
K	1	2	3	4	5
1	0.173266	0.146468	0.11848	0.146565	0.115256
2	0.174355	0.152254	0.13332	0.164909	0.119835
3	0.174609	0.152347	0.13456	0.168225	0.158065

Tabel 3.5 Kelas Data Uji berdasarkan jarak

	Data uji				
K	1	2	3	4	5
1	2	2	2	2	2
2	2	2	2	1	2
3	2	2	2	1	1

Keterangan : 1 : Beresiko baik (*Good Risk*)

2 : Beresiko buruk (*Bad Risk*)

Langkah 4 (Melakukan proses prediksi kelas).

Prediksi kelas digunakan fungsi kombinasi yaitu metode Weighted Voting pada persamaan 2.4 yang hasilnya ditampilkan pada tabel 3.6.

Tabel 3.6 Perhitungan Weighted Vote untuk menentukan kelas

	Data uji				
Kelas	1	2	3	4	5
1	0	0	0	2600.793	1601.969
2	3267.456	5890.111	11290.45	2167.073	10516.06
Vote	2	2	2	1	2

Tabel 3.7 Perbandingan kelas hasil klasifikasi dengan kelas aktual

Datauji	Kelas hasil klasifikasi	Kelas aktual
1	2	1
2	2	1
3	2	2
4	1	2
5	2	2

Keterangan : 1 : Beresiko baik (*Good Risk*)

2 : Beresiko buruk (*Bad Risk*)

Langkah 5 (menghitung fitness).

Dari tabel 3.7 tersebut diketahui bahwa terdapat beberapa kelas hasil klasifikasi dan kelas aktual memiliki nilai yang sama sehingga nilai *incorrect* adalah 0. Fungsi Fitnes dari kromosom ini dapat dihitung dengan persamaan 2.6.

$$F = \frac{TotalTest - Incorrect}{TotalTest}$$

$$= \frac{5 - 3}{5}$$

$$= 0.4$$

Lakukan cara yang sama untuk menghitung kromosom yang ada dalam populasi. Hasil *fitness* seluruh kromosom dalam populasi, dapat dilihat pada Tabel 3.8.

Tabel 3.8 Hasil *fitness* tiap kromosom

Kromosom ke-	***Fitness***
1	**0.4**
2	**1**
3	**0.4**
4	**0.4**

Pada perhitungan ini nilai rentang fitness kromosom berkisar diantara nilai terendah yaitu 0 dan nilai tertinggi adalah 1.

3.4.3 Seleksi Parent dengan Metode Roulette Wheel

Langkah 1 (mengambil *fitness* masing-masing kromosom)

Mengambil nilai *fitness* masing-masing kromosom pada populasi.

Kromosom 1 = 0.4

Kromosom 2 = 1

Kromosom 3 = 0.4

Kromosom 4 = 0.4

Langkah 2 (menghitung total *fitness*)

Menjumlahkan seluruh *fitness* untuk mendapatkan total *fitness* dengan persamaan 2.4.

Total fitness = 0.4 + 1 + 0.4 +0.4 = 2.2

Langkah 3 (menghitung peluang individu)

Menghitung masing-masing *fitness relative* terhadap total *fitness* yaitu dengan membagi masing-masing *fitness* individu dengan total *fitness* dengan persamaan 2.5.

Fitness relatif kromosom 1 = 0.4/2.2 = 0.1818

Fitness relatif kromosom 2 = 1/2.2 = 0.4545

Fitness relatif kromosom 3 = 0.4/2.2 = 0.1818

Fitness relatif kromosom 4 = 0.4/2.2 = 0.1818

Langkah 4 (menghitung peluang kumulatif)

Menghitung *fitness* kumulatif tiap kromosom untuk memperoleh skala atau daerah *roulette wheel* dengan persamaan 2.6.

Kromosom 1 =0 + 0.1818 = 0.1818

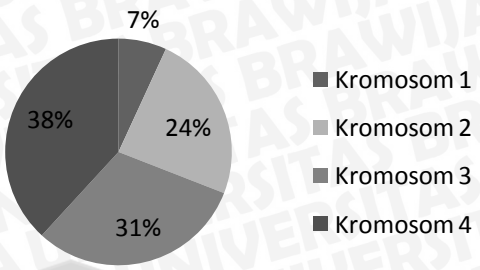
Kromosom 2 =0.1818+ 0.4545 = 0.6363

Kromosom 3 =0,6363+ 0.1818 = 0.8181

Kromosom 4 =0.8181 + 0.1818 = 0.9999 ≈ 1

Dari *fitness* kumulatif tersebut maka masing masing kromosom memiliki daerah sebagai berikut :

Kromosom 1	0 - 0.1817
Kromosom 2	0.1818 - 0.6362
Kromosom 3	0.6363 - 0.8180
Kromosom 4	0.8181 – 1



Langkah 5 (membangkitkan bilangan acak)

Membangkitkan nilai random antara 0 sampai 1 untuk menentukan kromosom yang akan terpilih melakukan *crossover*. Misalnya nilai random yang terpilih adalah sebagai berikut:

Random 1 = 0.5150

Random 2 = 0.0455

Langkah 6 (menentukan parent)

Berdasarkan batasan atau daerah mesing-masing kromosom, maka untuk random pertama yang terpilih adalah kromosom ke-2 sedangkan random kedua yang terpilih adalah kromosom ke-1. Maka selanjutnya proses *crossover* antara kromosom 2 dan 1 akan dilakukan.

Langkah 7 (mengulangi proses)

Melakukan proses *roulette wheel* sebanyak jumlah *parent* yang dihitung berdasarkan peluang persilangan yang telah ditentukan sebelumnya.

3.4.4 Restricted Crossover

Proses *crossover* dilakukan dengan cara mengkombinasikan 2 kromosom *parent* yang terpilih dengan satu titik potong yang ditentukan secara acak.

Langkah 1 (menetapkan pasangan kromosom hasil seleksi).

***Parent* 1** :Kromosom 2

0.031	0.094	0.063	0.017	0.079	0.017	0.085	0.007	0.026	0.02
0.056	0.038	0.039	0.063	0.047	0.076	0.075	0.022	0.051	0.094

Parent 2: Kromosom 1

0.13	0.04	0.09	0.01	0.14	0.1	0.01	0.19	0.02	0.001
0.006	0.009	0.045	0.023	0.065	0.045	0.002	0.003	0.011	0.06

Langkah 2 (menentukan titik potong).

Menentukan bilangan random antara 1 sampai (banyaknya gen – 1) yaitu 19. Misalnya nilai random yang muncul adalah 9. Maka titik potong berada pada posisi ke 9.

Langkah 3 (menghitung jumlah bobot sebelum dan sesudah titik potong)

***Parent* 1** :Kromosom 2, *fitness* = 1

0.419

0.031	0.094	0.063	0.017	0.079	0.017	0.085	0.007	0.026	0.02
0.056	0.038	0.039	0.063	0.047	0.076	0.075	0.022	0.051	0.094

0.581

Parent 2: Kromosom 1, *fitness* = 0.4

0.435

0.13	0.04	0.09	0.01	0.14	0.1	0.01	0.19	0.02	0.001
0.006	0.009	0.045	0.023	0.065	0.045	0.002	0.003	0.011	0.06

0.564

Langkah 4 (menukar bagian dari *parents* sebelum dan sesudah titik potong kepada *offspring* dan melakukan perbaikan pada bobot dari parent bernilai *fitness* rendah)

Child 1

0.419

0.031	0.094	0.063	0.017	0.079	0.017	0.085	0.007	0.026	0.046
0.002	0.0803	0.0989	0.050	0.081	0.002	0.0979	0.0288	0.069	0.023

0.581
0.564

Child 2

0.419
0.435

0.032	0.013	0.08	0.031	0.037	0.074	0.005	0.062	0.085	0.02
0.056	0.038	0.039	0.063	0.047	0.076	0.075	0.022	0.051	0.094

0.581

Keterangan warna :

: Kromosom dari *parent* 1

: Kromosom dari *parent* 2

3.4.5 Exchange Mutation

Proses mutasi diawali dengan membangkitkan nilai acak untuk menetukan apakah individu tersebut dimutasi atau tidak. Apabila nilai acak yang muncul lebih kecil dari probabilitas mutasi, maka individu tersebut akan dimutasi.

Langkah 1 (memilih random 2 posisi gen)

Memilih secara acak posisi 2 gen. Misalnya posisi gen ke-5 dan ke-9.

0.031	0.094	0.063	0.017	0.079	0.017	0.085	0.007	0.026	0.02

Langkah 2 (menukar isi gen)

Menukar isi gen pada posisi gen ke-5 dan posisi gen ke-9.

0.031	0.094	0.063	0.017	0.026	0.017	0.085	0.007	0.079	0.02

3.4.6Pembentukan Populasi baru

Individu-individu yang telah melalui proses *crossover* dan mutasi kemudian dimasukkan kedalam satu populasi sementara untuk diseleksi membentuk satu populasi baru. Pembentukan populasi baru dilakukan berdasarkan nilai *fitness* terbaik. Untuk itu perlu dilakukan perhitungan *fitness* lagi untuk masing-masing kromosom. Dengan cara yang sama seperti sebelumnya, *fitness* masing-masing kromosom dapat dilihat pada Tabel 3.9.

Tabel 3.9 Hasil *fitness* tiap kromosom setelah proses reproduksi

	Fitness
Kromosom 1	0.4
Kromosom 2	1
Kromosom 3	0.4
Kromosom 4	0.4
Kromosom 2 (mutasi)	0.4
Child 1	0.4
Child 2	0.4

Selanjutnya urutkan kromosom berdasarkan *fitness* dimulai dari yang terbesar hingga terkecil. Hasil pengurutan dapat dilihat pada Tabel 3.10.

Tabel 3.10 Hasil pengurutan individu berdasarkan *fitness*

	fitness
Kromosom 2	1
Child 1	0.4
Kromosom 1	0.4
Kromosom 4	0.4
Kromosom 3	0.4
Kromosom 2 (mutasi)	0.4
Child 2	0.4

Kemudian ambil empat kromosom teratas untuk dijadikan populasi baru, sedangkan individu yang *fitness*-nya tidak mencukupi akan dibuang. Individu yang terpilih untuk membentuk populasi baru disebut juga generasi ke-1. Proses genetika ini diulang terus hingga generasi maksimum yang ditentukan. Individu pada populasi baru dapat dilihat pada Tabel 3.11.

Tabel 3.11 Individu generasi ke-1

Individu	*Fitness*
Kromosom 2	1
Child 1	0.4
Kromosom 1	0.4
Kromosom 4	0.4

3.4.7 Klasifikasi dengan menggunakan WKNN

Misalkan bobot optimal yang dihasilkan adalah sebagai berikut :

0.13	0.04	0.09	0.01	...	...	...	0.003	0.011	0.06

Kemudian hitung jarak antara data uji dengan data latih yang ada pada tabel 3.1 dengan menggunakan bobot hasil pelatihan.

A12	21	A32	A40	2782	A63	A74	1	A92	A101	A12	21
2	A123	31	A141	A152	1	A174	1	A191	A201	2	A123

Contoh perhitungan jarak antara record pertama data latih (pada Tabel 3.1) dengan record uji berikut.

$$jarak_{11} = \sqrt{\sum_{i=1}^{20}(w_i(x_i - y_i))^2}$$

$$= \sqrt{\begin{matrix}(0.13*0)^2 + (0.04*0.2647)^2 + (0.09*1)^2 + (0.01*0)^2 + \\ (0.14*0.006)^2 + (0.1*0)^2 + (0.01*1)^2 + (0.19*0)^2 + \\ (0.02*1)^2 + (0.001*0)^2 + (0.006*0.6667)^2 + (0.009*1)^2 + \\ (0.045*0.625)^2 + (0.023*0)^2 + (0.065*0)^2 + (0.045*0.33)^2 + \\ (0.002*1)^2 + (0.003*0)^2 + (0.011*1)^2 + (0.06*0)^2\end{matrix}}$$

$$= \sqrt{0.019951}$$

$$= 0.141248$$

Selanjutnya dihitung lagi jarak antara record uji dengan data latih yang lain. Hasil perhitungan jarak ditampilkan pada Tabel 3.12.Kemudian dilakukan pengurutan terhadap jarak record lalu diambil k record dengan jarak terkecil. Apabila ditentukan k= 3, maka record yang terpilih adalah record ke-12, 14 dan 1.

Tabel 3.12 Hasil perhitungan Jarak

No Record	Jarak	Peringkat jarak terkecil	Kelas
1	0.141248	3	1
2	0.211156	10	2
3	0.228066	14	1
4	0.183811	7	1
5	0.167988	4	2
6	0.227019	13	1
7	0.178997	6	1
8	0.223059	12	1
9	0.210527	11	1
10	0.191777	9	2
11	0.188217	8	2
12	0.139	1	2
13	0.251492	15	1
14	0.139455	2	2
15	0.174438	5	1

Prediksi kelas dilakukan dengan metode *Weighted Voting* yaitu sebagai berikut :

Untuk kelas 1

$$= \frac{1}{0.141248^2}$$

$$= 50.122$$

Untuk kelas 2

$$= \frac{1}{0.139^2} + \frac{1}{0.139455^2}$$

$$= 71.76 + 51.42$$

$$= 123.18$$

Dari perhitungan di atas dapat disimpulkan bahwa kelas 2 lebih besar dari kelas 1 sehingga data uji termasuk kelas 2.

3.5Perancangan Uji Coba

Setelah sistem selesai dibuat, langkah selanjutnya adalah melakukan pengujian terhadap sistem tersebut. Pengujian dilakukan untuk mengetahui tingkat akurasi dari hasil klasifikasi kredit menggunakan *GeneticAlgorithm K-Nearest Neighbour*.

Pengujian yang dilakukan dalam penelitian ini, yaitu :

1. pengujian untuk mengetahui pengaruh peluang crossover dan peluang mutasi terhadap nilai *fitness*. Rancangan tabel yang akan digunakan untuk mencatat hasil dari pengujian ini ditampilkan oleh Tabel 3.13.
2. pengujian untuk mengetahui pengaruh jumlah populasi dan nilai k (jumlah tetangga terdekat) terhadap nilai *fitness*. Rancangan tabel yang akan digunakan untuk mencatat hasil dari pengujian ini ditampilkan oleh Tabel 3.14.
3. pengujian untuk mengetahui pengaruh jumlah data latih terhadap nilai *fitness*. Rancangan tabel yang akan digunakan untuk mencatat hasil dari pengujian ini ditampilkan oleh Tabel 3.15.
4. pengujian untuk mengetahui pengaruh jumlah k (ketetanggaan) terhadap tingkat akurasi KNN murni dan GA-WKNN. Rancangan tabel yang akan digunakan untuk mencatat hasil dari pengujian ini ditampilkan oleh Tabel 3.16.

Tabel 3.13 Tabel Uji Pengaruh Peluang Crossover Dan Peluang Mutasi Terhadap Nilai Fitness

PC	PM		

Tabel 3.14Tabel Uji pengaruh Jumlah Populasi, dan Nilai K terhadap Nilai Fitness

K	Jumlah Populasi	Nilai *Fitness*

Tabel 3.15 Tabel Uji Pengaruh Jumlah Data Latih Data Uji Terhadap Tingkat Akurasi

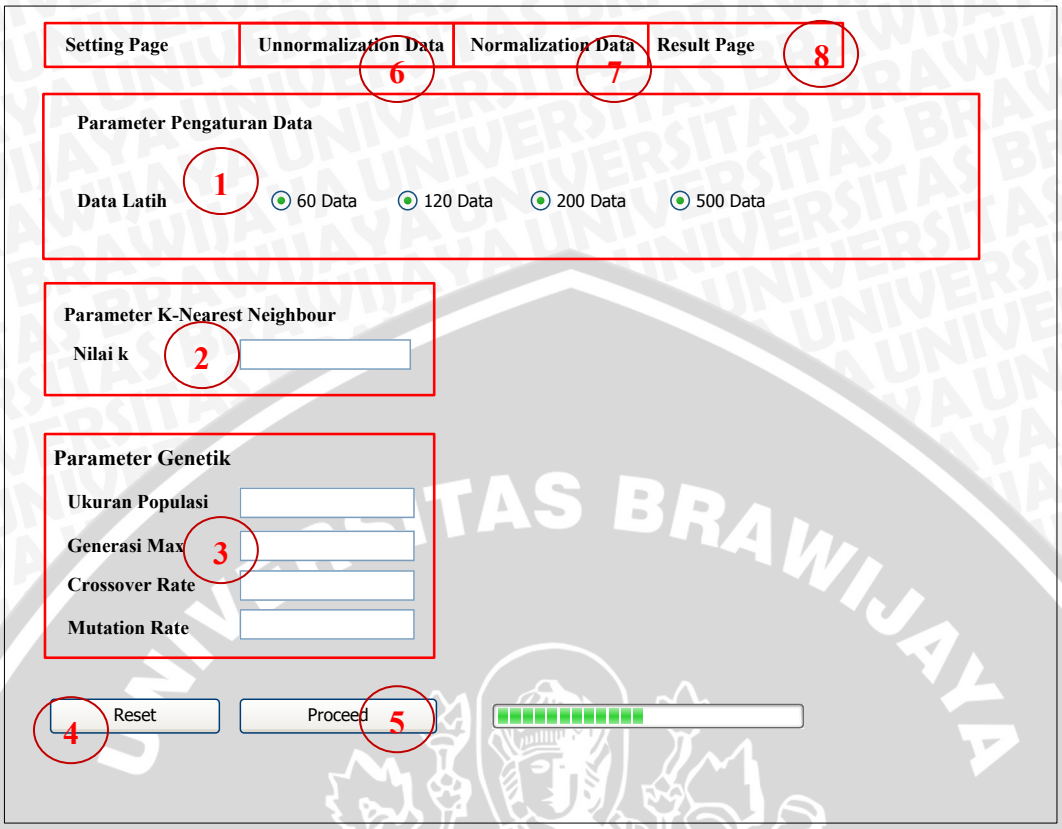
No.	Jumlah Data Latih	Jumlah Data Uji	Tingkat Akurasi (%)

Tabel 3.16 Tabel Uji Pengaruh Jumlah K (Ketetanggaan) Terhadap Tingkat Akurasi KNN Murni Dan GA/KNN

No.	Nilai k (ketetanggan)	Tingkat Akurasi KNN (%)	Tingkat Akurasi GA-WKNN (%)

3.6Rancangan Antarmuka

Antarmuka (*interface*) untuk sistem ini secara umum terdiri dari bagian bagian input, output, tombol proses dan reset. Inputan berupa dataset kredit German yang telah disediakan pilihan untuk memilih jumlahnya. Output berupa kromosom-kromosom pada populasi awal, bobot hasil GAKNN, Klasifikasi dengan WKNN beserta tingkat akurasi dan klasifikasi dengan KNN sebagai perbandingan. Gambar 3.13 menunjukkan rancangan antarmuka secara umum.



Gambar 3.13 Rancangan Antarmuka untuk Proses Pelatihan

Penjelasan dari bagian-bagian rancangan antarmuka di atas adalah sebagai berikut :

1. Tempat untuk memilih jumlah data latih yang ingin diproses. Disediakan 4*radiobutton* untuk memilih jumlah data.
2. Tempat untuk mengisi parameter KNN yang berisikan nilai k (ketetanggaan).
3. Tempat untuk mengisi parameter genetika yang terdiri dari:
   - Ukuran Populasi
   - Generasi Maksimal
   - Peluang *crossover*
   - Peluang mutasi
4. *Button* untuk mereset program.
5. *Button* untuk memproses perhitungan.
6. *Tab*untuk menampilkan data – data yang belum dinormalisasi
7. *Tab* untuk menampilkan data – data yang sudah dinormalisasi
8. *Tab* untuk menampilkan hasil perhitungan yang diantaranya terdapat:
   - *Page*populasi awal untuk menampilkan hasil dari populasi awal beserta *fitness*-nya.
   - *Page* untuk menampilkan hasil dari proses GA/KNN
   - *P*ageuntuk menampilkan hasil bobot optimal dan akurasi hasil dari klasifikasi WKNN

*Page*untuk menampilkan hasil dari klasifikasi KNN

BAB IV
HASIL DAN PEMBAHASAN

4.1 Lingkungan Implementasi

Proses implementasi merupakan tahap sistem siap dioperasikan pada keadaan yang sebenarnya, sehingga akan diketahui apakah sistem yang dibuat telah menghasilkan tujuan yang diinginkan. Lingkungan implementasi yang akan dijelaskan pada bab ini adalah lingkungan implementasi perangkat keras dan perangkat lunak.

4.1.1 Lingkungan Perangkat Keras

Perangkat keras yang digunakan dalam implementasi GA *K-Nearest Neighbour*untuk klasifikasi data kredit Germanadalah sebagai berikut :

1 Prosesor Intel Pentium Dual Core T4300 @2.10GHz
2 Memori 2GB DDR2
3 *Harddisk* dengan kapasitas 320 GB
4 Monitor 14”

4.1.2 Lingkungan Perangkat Lunak

Perangkat lunak yang digunakan dalam implementasi GA *K-Nearest Neighbour*untuk klasifikasi data kredit German adalah sebagai berikut:

1. Sistem Operasi *Microsoft Windows 7 Ultimate.*
2. *Microsoft Visual Studio* 2010*Proffesional Edition softwaredevelopment* dalam implementasi perancangan sistem.
3. *Microsoft Excel* pada penyimpanan data kredit.

4.2 Implementasi Program

Sub bab ini membahas mengenai implementasi proses-proses dari perancangan proses yang telah dijelaskan pada bab metodologi dan perancangan. Terbentuk beberapa kelas utama pada implementasi program ini. Kelas – kelas tersebut antara lain dapat dilihat pada tabel 4.1.

Tabel 4.1 Kelas Utama Implementasi Program

No.	Kelas	Keterangan
1.	`ReadData`	Membaca data dokumen
2.	`RecordClass`	*Struct class* untuk mengolah data
3.	`Kromosom`	Inisialisasi gen dalam kromosom
4.	`GA`	GA *K-Nearest Neighbour* untuk penentuan bobot optimal
5.	`WKNN`	Algoritma Klasifikasi *Weighted K-Nearest Neighbour*
6.	`KNN`	Algoritma Klasifikasi *K-Nearest Neighbour*

4.2.1 Struktur Data

Struktur data digunakan untuk menyimpan data-data yang dibutuhkan oleh sistem. Pada sistem ini digunakan struktur data berupa *struct class* yang digunakan untuk menyimpan dan mengolah variabel-variabel yang digunakan dalam sistem GA *K-Nearest Neighbour*.

Kelas *RecordClass* digunakan untuk menyimpan *node* yang selanjutnya variabel-variabel untuk menyimpan isi sebuah *node* dalam *List.* Kelas *RecordClass*direpresentasikan pada *sourcecode* 4.1.

```
1   publicclass RecordClass
2   {
3   privatestring _K1;
4   privatedouble _N2;
5   privatestring _K3;
6   privatestring _K4;
7   privatedouble _N5;
8   privatestring _K6;
9   privatestring _K7;
10  privatedouble _N8;
11  privatestring _K9;
12  privatestring _K10;
13  privatedouble _N11;
14  privatestring _K12;
15  privatedouble _N13;
16  privatestring _K14;
17  privatestring _K15;
18  privatedouble _N16;
19  privatestring _K17;
20  privatedouble _N18;
21  privatestring _K19;
22  privatestring _K20;
23  privateint _Class;
24
25  publicstring K1 { get {return _K1;} set { _K1 = value;}}
26  publicdouble N2 { get {return _N2;} set { _N2 = value;}}
27  publicstring K3 { get {return _K3;} set { _K3 = value;}}
28  publicstring K4 { get {return _K4;} set { _K4 = value;}}
29  publicdouble N5 { get {return _N5;} set { _N5 = value;}}
30  publicstring K6 { get {return _K6;} set { _K6 = value;}}
31  publicstring K7 { get {return _K7;} set { _K7 = value;}}
```

```
32  publicdouble N8 { get {return _N8;} set { _N8 = value;}}
33  publicstring K9 { get {return _K9;} set { _K9 = value;}}
34  publicstring K10 { get {return _K10;} set { _K10 = value;}}
35  publicdouble N11 { get {return _N11;} set { _N11 = value;}}
36  publicstring K12 { get {return _K12;} set { _K12 = value;}}
37  publicdouble N13 { get {return _N13;} set { _N13 = value;}}
38  publicstring K14 { get {return _K14;} set { _K14 = value;}}
39  publicstring K15 { get {return _K15;} set { _K15 = value;}}
40  publicdouble N16 { get {return _N16;} set { _N16 = value;}}
41  publicstring K17 { get {return _K17;} set { _K17 = value;}}
42  publicdouble N18 { get {return _N18;} set { _N18 = value;}}
43  publicstring K19 { get {return _K19;} set { _K19 = value;}}
44  publicstring K20 { get {return _K20;} set { _K20 = value;}}
45  publicint Class { get {return _Class;} set { _Class = value;}}
46
47  public RecordClass()
48  {
49  _K1 = _K3 = _K4 = _K6 = _K7 = _K9 = _K10 = _K12 = _K14 = _K15 = _K17 =
    _K19 = _K20 ="";
50  _N2 = _N5 = _N8 = _N11 = _N13 = _N16 = _N18 = _Class =0;
51  }
52  }
```

Sourcecode* 4.1 Kelas *RecordClass

4.2.2Implementasi Kelas *ReadData*

Kelas ReadData memiliki dua method. Method pertama untuk membaca *file*data dalam bentuk **.csv*, method kedua untuk memasukkan data yang telah dibaca dan dinormalisasi ke dalam *struct* RecordClass menggunakan *Minmax Normalization*.

4.2.2.1 Implementasi Baca *File* Data

Tahap awal yang dilakukan dalam proses pelatihan data pencarian bobot optimal ini adalah mengambil data yang dijadikan data latih dengan membaca file dalam format *csv*. Setelah itu data dalam *file* tersebut disimpan dalam *struct* RecordClassyang berisi variabel-variabel untuk penyimpanan setiap atribut dalam pencarian bobot optimal. Implementasi baca *file* data latih ditunjukkan pada sourcecode 4.2.

```
1  public List<RecordClass> parseCSVallData(string path){
2  try
3  {
4  using(StreamReader bacaFile =new StreamReader(path))
5  {
6  string baris = bacaFile.ReadLine();
7  string[] kolom = baris.Split(',');
8
   minmaxValue.Add(newdouble[3]{int.Parse(kolom[1]),int.Parse(kolom[1]),0});
```

```
9
   minmaxValue.Add(newdouble[3]{int.Parse(kolom[4]),int.Parse(kolom[4]),0});
10
   minmaxValue.Add(newdouble[3]{int.Parse(kolom[7]),int.Parse(kolom[7]),0});
11
   minmaxValue.Add(newdouble[3]{int.Parse(kolom[10]),int.Parse(kolom[10]),0});
12 minmaxValue.Add(newdouble[3]{int.Parse(kolom[12]),int.Parse(kolom[12]),0});
13 minmaxValue.Add(newdouble[3]{int.Parse(kolom[15]),int.Parse(kolom[15]),0});
14 minmaxValue.Add(newdouble[3]{int.Parse(kolom[17]),int.Parse(kolom[17]),0});
   do{
15      RecordClass currentRow =new RecordClass();
16      kolom = baris.Split(',');
17      currentRow.K1 = kolom[0];
18      currentRow.N2 =int.Parse(kolom[1]);
19      minmaxValue[0][0]= Math.Min(minmaxValue[0][0], currentRow.N2);
20    minmaxValue[0][1]= Math.Max(minmaxValue[0][1], currentRow.N2);
21   minmaxValue[0][2]+= currentRow.N2;
22
23      currentRow.K3 = kolom[2];
24      currentRow.K4 = kolom[3];
25
26      currentRow.N5 =int.Parse(kolom[4]);
27      minmaxValue[1][0]= Math.Min(minmaxValue[1][0], currentRow.N5);
28      minmaxValue[1][1]= Math.Max(minmaxValue[1][1], currentRow.N5);
29      minmaxValue[1][2]+= currentRow.N5;
30
31      currentRow.K6 = kolom[5];
32      currentRow.K7 = kolom[6];
33
34      currentRow.N8 =int.Parse(kolom[7]);
35      minmaxValue[2][0]= Math.Min(minmaxValue[2][0], currentRow.N8);
36      minmaxValue[2][1]= Math.Max(minmaxValue[2][1], currentRow.N8);
37      minmaxValue[2][2]+= currentRow.N8;
38
39      currentRow.K9 = kolom[8];
40      currentRow.K10 = kolom[9];
41
42      currentRow.N11 =int.Parse(kolom[10]);
43      minmaxValue[3][0]= Math.Min(minmaxValue[3][0], currentRow.N11);
44      minmaxValue[3][1]= Math.Max(minmaxValue[3][1], currentRow.N11);
45      minmaxValue[3][2]+= currentRow.N11;
46
47      currentRow.K12 = kolom[11];
48
49      currentRow.N13 =int.Parse(kolom[12]);
50      minmaxValue[4][0]= Math.Min(minmaxValue[4][0], currentRow.N13);
51      minmaxValue[4][1]= Math.Max(minmaxValue[4][1], currentRow.N13);
52      minmaxValue[4][2]+= currentRow.N13;
53
54      currentRow.K14 = kolom[13];
55      currentRow.K15 = kolom[14];
56
57      currentRow.N16 =int.Parse(kolom[15]);
58      minmaxValue[5][0]= Math.Min(minmaxValue[5][0], currentRow.N16);
59      minmaxValue[5][1]= Math.Max(minmaxValue[5][1], currentRow.N16);
60      minmaxValue[5][2]+= currentRow.N16;
61
```

```
62      currentRow.K17 = kolom[16];
63
64       currentRow.N18 =int.Parse(kolom[17]);
65       minmaxValue[6][0]= Math.Min(minmaxValue[6][0], currentRow.N18);
66       minmaxValue[6][1]= Math.Max(minmaxValue[6][1], currentRow.N18);
67       minmaxValue[6][2]+= currentRow.N18;
68
69      currentRow.K19 = kolom[18];
70       currentRow.K20 = kolom[19];
71
72       currentRow.Class =int.Parse(kolom[20]);
73
74      dataLatih.Add(currentRow);
75  }while((baris = bacaFile.ReadLine())!=null);
76  }
77  }
78  catch(FileNotFoundException)
79  {
80    KryptonMessageBox.Show("\nFile tak ada","perhatian",
81  MessageBoxButtons.OK, MessageBoxIcon.Warning);
82  }
83  return dataLatih;
84  }
85
```

***Sourcecode* 4.2Baca *File*Data Kelas ReadData**

4.2.2.2 Implementasi Minmax Normalization

Tahap selanjutnya adalah normalisasi atribut-atribut data numerik menggunakan metode *Minmax Normalization* sehingga memiliki skala nilai berkisar antara 0 s/d 1.Implementasi *Minmax Normalization* ditunjukkan pada *sourcecode* 4.3.

```
1   public List<RecordClass> minMaxNormalization(List<RecordClass>
    dataMentah)
2   {
3    List<RecordClass> hasilMMN =new List<RecordClass>();
4
5   foreach(RecordClass dM in dataMentah)
6   {
7   RecordClass currentRow =new RecordClass();
8    currentRow.K1 = dM.K1;
9   currentRow.N2 = Math.Round((dM.N2 -
    minmaxValue[0][0])/(minmaxValue[0][1]- minmaxValue[0][0]),3);
10  currentRow.K3 = dM.K3;
11   currentRow.K4 = dM.K4;
12   currentRow.N5 = Math.Round((dM.N5 -
    minmaxValue[1][0])/(minmaxValue[1][1]- minmaxValue[1][0]),3);
13   currentRow.K6 = dM.K6;
14   currentRow.K7 = dM.K7;
15   currentRow.N8 = Math.Round((dM.N8 -
    minmaxValue[2][0])/(minmaxValue[2][1]- minmaxValue[2][0]),3);
16   currentRow.K9 = dM.K9;
17   currentRow.K10 = dM.K10;
18   currentRow.N11 = Math.Round((dM.N11 -
```

```
    minmaxValue[3][0])/(minmaxValue[3][1]- minmaxValue[3][0]),3);
19   currentRow.K12 = dM.K12;
20   currentRow.N13 = Math.Round((dM.N13 -
    minmaxValue[4][0])/(minmaxValue[4][1]- minmaxValue[4][0]),3);
21   currentRow.K14 = dM.K14;
22   currentRow.K15 = dM.K15;
23   currentRow.N16 = Math.Round((dM.N16 -
    minmaxValue[5][0])/(minmaxValue[5][1]- minmaxValue[5][0]),3);
24   currentRow.K17 = dM.K17;
25   currentRow.N18 = Math.Round((dM.N18 -
    minmaxValue[6][0])/(minmaxValue[6][1]- minmaxValue[6][0]),3);
26   currentRow.K19 = dM.K19;
27   currentRow.K20 = dM.K20;
28  currentRow.Class = dM.Class;
29   hasilMMN.Add(currentRow);
30  }
31  return hasilMMN;
32  }
```

Sourcecode 4.3Minmax NormalizationKelas ReadData

4.2.3Implementasi Kelas Kromosom

Kelas Kromosom terdiri dari method pembentukanBobotRandomAtribut yang berguna untukinisialisasi gen dalam kromosom, dan *properties* untuk mengakses nilai dari gen-gen di dalam kromosom. Proses tersebut dapat dilihat pada sourcecode 4.4.

```
1   class Kromosom
2       {
3           private int jumlahAtribut;
4           private double[] bobotAttribut;
5           private Random acak = new Random();
6
7           public Kromosom()
8           {
9               jumlahAtribut = 0;
10              bobotAttribut = new double[0];
11          }
12
13          public Kromosom(int _jumlahAtribut)
14          {
15              jumlahAtribut = _jumlahAtribut;
16              bobotAttribut = new double[jumlahAtribut];
17              for (int j = 0; j < jumlahAtribut; j++)
18              {
19                  bobotAttribut[j] = 0;
20              }
21          }
22
23          public void pembentukanBobotRandomAtribut()
24          {
25              double sum = 0;
26
27              for (int i = 0; i < jumlahAtribut; i++)
28              {
```

```
29                  bobotAttribut[i] = Math.Round(acak.NextDouble(), 3);
30                  Thread.Sleep(1);
31                  sum += bobotAttribut[i];
32              }
33
34              for (int i = 0; i < jumlahAtribut; i++)
35                  bobotAttribut[i] = Math.Round(bobotAttribut[i] / sum, 3);
36          }
37
38          public double[] getBobotAttribut()
39          {
40              return bobotAttribut;
41          }
42
43          public void setBobotAttributPadaGen(int _position, double _value)
44          {
45              bobotAttribut[_position] = _value;
46          }
47
48          // mendapatkan posisi atribut
49          public double getBobotAttributPadaGen(int _position)
50          {
51              return bobotAttribut[_position];
52          }
53      }
```

***Sourcecode* 4.4 Kelas Kromosom**

4.2.4 Kelas GA

Kelas GAterdiri dari beberapa method yang berguna untuk proses pembentukan dan pengembangan genetika dari individu-individu yang berisi solusi pembobotan.

4.2.4.1 Inisialisasi Kromosom untuk Populasi Awal

Pada proses inisialisasi kromosom dilakukan pembangkitan populasi awal dengan nilai gen yang sama. Panjang gen kromosom sesuai dengan jumlah atribut. Proses inisialisasi kromosom dapat dilihat pada *sourcecode* 4.5.

```
1   public void initialPopulasi()
2   {
3    for (int i = 0; i < jumlahIndividuPerPopulasi; i++)
4    {
5     Kromosom individu = new Kromosom(jumlahAttribut);
6     individu.pembentukanBobotRandomAtribut();
7     bobotIndividu[i] = individu;
8    }
9   }
```

***Sourcecode* 4.5 Inisialisasi Kromosom Kelas GA**

4.2.4.2 Perhitungan *Fitness*

Perhitungan nilai *fitness* diperoleh dari perhitungan WKNN terhadap data latih dan data uji. Perhitungan *fitness* dilakukan pada setiap individu. Proses perhitungan *fitness* dapat dilihat pada *sourcecode* 4.6.

```
1   public int hitFitness(List<RecordClass> _dataLatih, List<RecordClass>
    _dataLatihUji, int k, int _noUrutIndividu)
2   {
3    double kelas1;
4    double kelas2;
5    int JumlahHasilUjiSesuaiRealClass;
6    int hasilwKNNbobotAwal;
7    double jarak;
8    double[] EuclidDistance = new double[_dataLatih.Count];
9    double[] data = new double[_dataLatih.Count];
10   double[] urutan = new double[_dataLatih.Count];
11   double[] tetangga = new double[_dataLatih.Count];
12   string[] prediksi = new string[_dataLatih.Count];
13   string[] kelasFromDataLatih = new string[_dataLatih.Count];
14
15   for (int ke = 0; ke < _dataLatihUji.Count; ke++) // baris data uji
16   {
17    for (int i = 0; i < _dataLatih.Count; i++) // baris data latih
18    {
19     jarak = compareRecordWithBobotAwal(_dataLatih[i], _dataLatihUji[ke]
    , 20, _noUrutIndividu);
20
21     EuclidDistance[i] = Math.Round((1 / jarak), 4);
22    }
23
24    for (int x = 0; x < _dataLatih.Count; x++)
25     {
26      urutan[x] = EuclidDistance[x];
27      kelasFromDataLatih[x] = Convert.ToString(_dataLatih[x].Class);
28     }
29
30     Array.Sort(urutan, kelasFromDataLatih);
31    kelas1 = 0;
32     kelas2 = 0;
33
34     for (int z = 0; z < k; z++)
35     {
36      tetangga[z] = urutan[z];
37      if (kelasFromDataLatih[z] == "1")
38         kelas1 += urutan[z];
39      else
40         kelas2 += urutan[z];
41     }
42     if (kelas1 > kelas2)
43       prediksi[ke] = "1";
44     else
45       prediksi[ke] = "2";
46    }
47
48    hasilwKNNbobotAwal = 0;
49    JumlahHasilUjiSesuaiRealClass = 0;
```

```
50   for (int ff = 0; ff < _dataLatihUji.Count; ff++)
51   {
52    if (prediksi[ff] == Convert.ToString(_dataLatihUji[ff].Class))
53      JumlahHasilUjiSesuaiRealClass++;
54   }
55   hasilwKNNbobotAwal = JumlahHasilUjiSesuaiRealClass;
56   return hasilwKNNbobotAwal;
57  }
```

***Sourcecode* 4.6 Perhitungan *Fitness*Kelas GA**

4.2.4.3 Proses Seleksi

Proses seleksi atau pemilihan individu induk (*parent*) dilakukan dengan metode *Roulette Wheel*. Pada seleksi ini semakin besar nilai *fitness* suatu individu maka akan semakin besar *range* individu tersebut untuk terpilih menjadi induk. Langkah pertama pada proses seleksi adalah menghitung total *fitness*, peluang, dan peluang kumulatif masing-masing individu. Proses perhitungan tersebut dapat dilihat pada *sourcecode* 4.7.

```
1   private void hitungProbKum()
2   {
3    double TotalFitness = 0;
4    for (int i = 0; i < jumlahIndividuPerPopulasi; i++)
5    {
6     TotalFitness += fitnessIndividu[i];
7    }
8
9    for (int i = 0; i < jumlahIndividuPerPopulasi; i++)
10   {
11    if (i == 0) // baris pertama
12     probKumulatif[i] = Math.Round(fitnessIndividu[i] / TotalFitness, 3)
     ;
13    else
14    {
15     if (i == jumlahIndividuPerPopulasi - 1)
16       probKumulatif[i] = 1;
17     else
18      probKumulatif[i] = Math.Round(probKumulatif[i -
     1] + fitnessIndividu[i] / TotalFitness, 3);
19    }
20   }
21  }
```

***Sourcecode* 4.7 Perhitungan peluang kumulatif individu Kelas GA**

Langkah berikutnya adalah membangkitkan bilangan *random* sebanyak jumlah *parent*. Bilangan *random* yang dibangkitkan kemudian dibandingkan dengan nilai peluang kumulatif pada suatu individu. Jika bilangan *random* tersebut lebih kecil dari peluang kumulatif individu, maka individu tersebut

terseleksi menjadi induk. Selain itu juga dilakukan proses pengecekan terhadap individu yang telah terpilih. Apabila individu tersebut telah terpilih maka dilakukan proses penyeleksian sampai diperoleh individu yang belum terseleksi sebelumya. Proses seleksi *Roulette Wheel* dapat dilihat pada *sourcecode* 4.8.

```
1   publicvoid rouletteWheel()
2   {
3    hitungProbKum();
4   double acak;
5   int[] tempInduk =newint[jumlahIndividuPerPopulasi];// membuat array
    sejumlah nPopulasi, jika terpilih diset 1. digunakan untuk penanda
    individu yg telah terdaftar sebagai induk
6   int induk_1 =0;
7   int induk_2 =0;
8
9   //inisialisasi tempInduk dengan nilai 0
10  for(int k =0; k < tempInduk.Length; k++)
11  {
12    tempInduk[k]=0;
13  }
14
15  for(int i =0; i < urutanInduk.Length; i++)
16  {
17  //membuat suatu bilangan random untuk dibandingkan dengan probkum
    suatu individu
18    Thread.Sleep(1);
19    acak =new Random().NextDouble();
20  for(int j =0; j < jumlahIndividuPerPopulasi; j++)// cari induk
21  {
22  if(acak <= probKumulatif[j])
23  {
24  if(tempInduk[j]==1)// 1 berarti individu[j] sudah terpilih menjadi
    parent di iterasi sebelunya
25  {
26        i = i -1;//iterasi batal dan diulang
27  break;
28  }
29  else// j sebagai index induk
30  {
31    urutanInduk[i]= j;//individu[j] d set sebagai induk ke-i
32       tempInduk[j]=1;// set 1 pada temp induk jika indeksnya sudah
    terpilih
33  if(i %2==0)// jika induk ke-i berindeks genap
34  {
35         pasanganInduk[induk_1, induk_2]= j;// dimasukan di pasangan
    induk kolom 0
36         induk_2++;
37  }
38  else// induk berindeks ganjil
39  {
40         pasanganInduk[induk_1, induk_2]= j;// dimasukan di pasangan
    induk kolom 1
41         induk_1++;
42         induk_2--;
43  }
44  break;
```

```
45  }
46  }
47  }// end for j
48  }// end for i
49  }
```

Sourcecode 4.8 Seleksi Roulette Wheel Kelas GA

4.2.4.4 Proses *Crossover*

Metode *crossover* yang digunakan adalah Persilangan Satu Titik. Sebelum dilakukan proses *crossover* dilakukan dahulu pemilihan individu induk yang akan mengalami *crossover* sesuai dengan nilai probabilitas *crossover* yang telah ditentukan. Proses pemilihan individu yang akan di*crossover* tersebut dapat dilihat pada *sourcecode* 4.9.

```
1   publicvoid crossover()
2   {
3   //inisialisasi offspring sebagai kromosom/individu yg memiliki
    beberapa atribut
4   for(int i =0; i < bobotOffspring.Length; i++)
5   {
6     bobotOffspring[i]=new Kromosom(jumlahAttribut);
7   }
8
9    Thread.Sleep(1);
10  int CutPointPosition =new Random().Next(1, jumlahAttribut -1);// cut
    point d harapkan berada pada setelah gen pertama dan sebelum gen
    terakhir
11  double randomCrossRoulette =new Random().NextDouble();
12  int numOffspring =0;
13  int j =0;
14  for(int i =0; i < pasanganInduk.GetLength(0); i++)
15  {
16  if(randomCrossRoulette < pC)//jika nilai ya kurang dari nilai pC, maka
    akan diadakan crossover
17  {
18     SilangSatu(numOffspring, bobotIndividu[pasanganInduk[i,
    j]].getBobotAttribut(), bobotIndividu[pasanganInduk[i, j
    +1]].getBobotAttribut(), CutPointPosition,
    fitnessIndividu[pasanganInduk[i, j]], fitnessIndividu[pasanganInduk[i,
    j +1]]);
19  }
20  else
21  {
22     NoCrossOver(numOffspring, bobotIndividu[pasanganInduk[i,
    j]].getBobotAttribut(), bobotIndividu[pasanganInduk[i, j
    +1]].getBobotAttribut());
23  }
24    numOffspring = numOffspring +2;//dalam sekali crossover akan
    menghasilkan 2 offspring
25  }
26  }
```

Sourcecode* 4.9 Pemilihan *parent* yang mengalami *crossover

Langkah selanjutnya adalah proses Persilangan Satu Titik. Pada Persilangan Satu Titik akan digunakan metode *Restricted Crossover*, di mana akan dibangkitkan satu bilangan *random* dengan nilai antara 1 sampai dengan jumlah atribut - 1. Bilangan *random* tersebut merepretasikan nilai titik potong kromosom. Kemudian dilakukan tukar silang gen. Gen pertama sampai titik potong pada parent 1 dan gen pada titik potong sampai gen terakhir dari parent 2 disalin ke offspring 1 dengan mempertahankan urutannya. Begitu juga dengan gen yang tersisa. Gen pertama sampai titik potong pada parent 2 dan gen pada titik potong sampai gen terakhir dari parent 1 disalin ke offspring 2 dengan mempertahankan urutannya. Proses tukar silang tersebut dapat dilihat pada *sourcecode* 4.10.

```
1   privatevoid SilangSatu(int _numOffspring,double[] bobotInduk1,double[]
    bobotInduk2,int CutPoint,int fitness1,int fitness2)// kalau terjadi
    crossover
2   {
3   double a =0, b =0;
4   for(int fit =0; fit < CutPoint; fit++)
5   {
6     a += bobotInduk1[fit];
7     b += bobotInduk2[fit];
8   }
9
10  double[] parent1 ={ a,1- a }, parent2 ={ b,1- b };
11
12  if(fitness1 > fitness2)// worse fitness akan dioveride oleh perbaikan
    bobot sehingga total bobot dari offspring yang bersangkutan = 1
    {
13  for(int j =0; j < CutPoint; j++)//deret bobot sebelum cutpoint
14  {
15     bobotOffspring[_numOffspring].setBobotAttributPadaGen(j,
16  bobotInduk1[j]);
       bobotOffspring[_numOffspring +1].setBobotAttributPadaGen(j,
17  Math.Round(((parent1[0]/ parent2[0])* bobotInduk2[j]),3));
    }
18
19  for(int j2 = CutPoint; j2 < jumlahAttribut; j2++)//deret bobot setelah
20  cutpoint
    {
21     bobotOffspring[_numOffspring].setBobotAttributPadaGen(j2,
22  Math.Round(((parent1[1]/ parent2[1])* bobotInduk2[j2]),3));
       bobotOffspring[_numOffspring +1].setBobotAttributPadaGen(j2,
23  bobotInduk1[j2]);
    }
24  }
25  else//if (fitness2 > fitness1)
26  {
27  for(int j =0; j < CutPoint; j++)
28  {
29     bobotOffspring[_numOffspring].setBobotAttributPadaGen(j,
30  bobotInduk2[j]);
31     bobotOffspring[_numOffspring +1].setBobotAttributPadaGen(j,
```

```
32  Math.Round(((parent2[0]/ parent1[0])* bobotInduk1[j]),3));
    }
33
34  for(int j2 = CutPoint; j2 < jumlahAttribut; j2++)
35  {
36     bobotOffspring[_numOffspring].setBobotAttributPadaGen(j2,
37  Math.Round(((parent2[1]/ parent1[1])* bobotInduk1[j2]),3));
       bobotOffspring[_numOffspring +1].setBobotAttributPadaGen(j2,
38  bobotInduk2[j2]);
    }
39  }
40  }
41
```

***Sourcecode* 4.10 Proses Persilangan Satu Titik**

4.2.4.5 Proses Mutasi

Metode mutasi yang digunakan adalah *Exchange Mutation* pada gen. Langkah awal proses mutasi hampir sama dengan proses *crossover*, yaitu melakukan pemilihan individu yang akan mengalami mutasi sesuai dengan probabilitas mutasi yang telah ditentukan. Proses pemilihan individu yang akan mengalami mutasi dapat dilihat pada *sourcecode* 4.11.

```
1   public void ExchangeMutasi(List<RecordClass> _dataLatihMutasi, List<Re
    cordClass> _dataUjiMutasi, int _k)
2   {
3   double randomMut;
4
5   for (int i = 0; i < bobotOffspring.Length; i++)
6   {
7   int fitnessSetelahMutasi = 0;
8   {
9   Thread.Sleep(1);
10  randomMut = new Random().NextDouble();
11  if (randomMut < pM)
12  exchangeMutasi(i);
13      fitnessSetelahMutasi = hitFitnessOffspring(_dataLatihMutasi, _data
    UjiMutasi, _k, i);
14  }
15  setFitness(fitnessOffspring, i, fitnessSetelahMutasi);
16  }
17  }
```

***Sourcecode* 4.11 Pemilihan *offspring* yang mengalami mutasi**

Langkah selanjutnya adalah proses *Exchange Mutation* pada Gen Pada mutasiini, dibangkitkan dua posisi *random* dengan nilai antara 1 sampai dengan panjang kromosom, dengan syarat posisi pertama dan kedua tidak sama. Kemudian dilakukan penukarann kedua posisi gen tersebut. Proses *Exchange Mutation*dapat dilihat pada *sourcecode* 4.12.

```
1  private void exchangeMutasi(int i)
2  {
3  double temp1, temp2;
4  Random acak = new Random();
5  int pos1, pos2; // posisi gen ke - 1 dan ke - 2
6  pos1 = acak.Next(0, jumlahAttribut);
7  pos2 = acak.Next(0, jumlahAttribut);
8  if (pos2 != pos1)
9  {
10 temp1 = bobotOffspring[i].getBobotAttributPadaGen(pos1);
11 temp2 = bobotOffspring[i].getBobotAttributPadaGen(pos2);
12 bobotOffspring[i].setBobotAttributPadaGen(pos1, temp2);
13 bobotOffspring[i].setBobotAttributPadaGen(pos2, temp1);
14 }
15 }
```

Sourcecode 4.12 Proses Exchange Mutation pada Gen

4.2.4.6 Proses Pembentukan Populasi Baru

Proses pembentukan populasi baru dilakukan dengan menambahkan individu baru ke dalam individu awal. Kemudian dilakukan perhitungan nilai *fitness* semua individu. Proses penggabungan individu anak dengan individu awal dapat dilihat pada *sourcecode* 4.13.

```
1  private void gabung()
2  {
3   Array.Resize(ref bobotIndividu, bobotIndividu.Length + bobotOffspring.
   Length);
4   Array.Resize(ref fitnessIndividu, fitnessIndividu.Length + fitnessOffs
   pring.Length);
5
6   for (int i = bobotIndividu.Length -
   bobotOffspring.Length; i < bobotIndividu.Length; i++)
7   {
8    bobotIndividu[i] = new Kromosom(jumlahAttribut);
9   }
10
11  for (int i = bobotIndividu.Length -
   bobotOffspring.Length; i < bobotIndividu.Length; i++)
12  {
13  for (int j = 0; j < jumlahAttribut; j++)
14   {
15   setPopulasi(bobotIndividu, i, j, bobotOffspring[i -
   jumlahIndividuPerPopulasi].getBobotAttributPadaGen(j));
16   }
17   setFitness(fitnessIndividu, i, getFitness(fitnessOffspring, i -
   jumlahIndividuPerPopulasi));
18  }
19 }
```

***Sourcecode* 4.13 Proses penggabungan individu awal dan anak**

Dari total individu tersebut dilakukan pengurutan (*sorting*) nilai *fitness* dari nilai terbesar ke kecil. Kemudian dilakukan pemilihan *fitness* terbaik sebanyak

jumlah populasi awal. Proses *sorting* dan pemilihan individu sejumlah populasi awal dapat dilihat pada *sourcecode* 4.14.

```
1   public void sorting()
2   {
3   int indexMax;
4   int temp;
5   for (int i = 0; i < fitnessIndividu.Length; i++)
6   {
7   indexMax = i;
8   for (int j = i + 1; j < fitnessIndividu.Length; j++)
9   {
10  if ((fitnessIndividu[j]) > (fitnessIndividu[indexMax]))
11  indexMax = j;
12  }
13  temp = fitnessIndividu[i];
14  fitnessIndividu[i] = fitnessIndividu[indexMax];
15  fitnessIndividu[indexMax] = temp;
16
17    replacePopulasi(bobotIndividu[i].getBobotAttribut(), bobotIndividu[i
    ndexMax].getBobotAttribut());
18  }
19  }
20
21  public void populasiBaru()
22  {
23  gabung();
24  sorting();
25  Array.Resize(ref bobotIndividu, jumlahIndividuPerPopulasi);
26  Array.Resize(ref fitnessIndividu, jumlahIndividuPerPopulasi);
27  }
```

***Sourcecode* 4.14 Proses *sorting* dan pemilihan individu baru sejumlah populasi awal**

4.2.5Implementasi Kelas WKNN

Kelas WKNNterdiri dari method klasifikasiWKNN yang berguna untuk pengklasifikasian data uji terhadap data latih menggunakan kromosom terbaik atau bobot optimal yang dihasilkan dari kelas GA. Proses klasifikasi dengan WKNN dapat dilihat pada *sourcecode* 4.15.

```
1   public string klasifikasiWKNN(KryptonRichTextBox resultLog, double _fi
    tnessOptimal)
2   {
3   double kelas1;
4   double kelas2;
5   double jarak;
6   int JumlahHasilUjiSesuaiRealClass;
7   double[] EuclidDistance = new double[dataLatih.Count];
8   double[] data = new double[dataLatih.Count];
9   double[] urutan = new double[dataLatih.Count];
10  double[] tetangga = new double[dataLatih.Count];
11  string[] prediksi = new string[dataLatih.Count];
```

```
12  string[] kelasFromDataLatih = new string[dataLatih.Count];
13  resultLog.Clear();
14  cetakKelas.Clear();
15
16   resultLog.AppendText("Bobot optimal yang digunakan adalah sebagai ber
    ikut:\n\t");
17  for (int j = 0; j < bobot.Length; j++)
18  {
19  if (j == (bobot.Length - 1) / 2)
20  resultLog.AppendText("" + bobot[j] + "\n\t");
21  else
22  resultLog.AppendText("" + bobot[j] + "\t");
23  }
24   resultLog.AppendText("\nDengan fitness optimal sebesar: "+_fitnessOpt
    imal);
25   cetakKelas.Add("\n\nKeterangan:  1 = Good Risk(beresiko baik), 2 = Ba
    d Risk(beresiko buruk) \n\n");
26   cetakKelas.Add("Data ke-
    \t" + " Kelas Hasil Prediksi \t " + "Kelas Sebenarnya\t\n");
27
28   for (int dataUjiKe = 0; dataUjiKe < dataUji.Count; dataUjiKe++)
29   {
30   for (int i = 0; i < dataLatih.Count; i++) // baris data latih
31    {
32    jarak = compareRecord(dataLatih[i], dataUji[dataUjiKe], 20);
33
34     EuclidDistance[i] = Math.Round((1 / jarak), 4);
35    }
36
37    for (int x = 0; x < dataLatih.Count; x++)
38    {
39    urutan[x] = EuclidDistance[x];
40     kelasFromDataLatih[x] = Convert.ToString(dataLatih[x].Class);
41    }
42
43    Array.Sort(urutan, kelasFromDataLatih);
44    kelas1 = 0;
45    kelas2 = 0;
46
47    for (int z = 0; z < k; z++)
48    {
49    tetangga[z] = urutan[z];
50     if (kelasFromDataLatih[z] == "1")
51     kelas1 += urutan[z];
52     else
53     kelas2 += urutan[z];
54    }
55    if (kelas1 > kelas2)
56    prediksi[dataUjiKe] = "1";
57    else
58    prediksi[dataUjiKe] = "2";
59
60    cetakKelas.Add("   " + (dataUjiKe + 1) + "\t\t" + prediksi[dataUjiKe]
     + "\t\t" + dataUji[dataUjiKe].Class + "\n");
61  }
62  JumlahHasilUjiSesuaiRealClass = 0;
63  for (int indexHasilUji = 0; indexHasilUji < dataUji.Count; indexHasilU
    ji++)
64  {
```

```
65   if (prediksi[indexHasilUji] == Convert.ToString(dataUji[indexHasilUji
     ].Class))
66   JumlahHasilUjiSesuaiRealClass++;
67   }
68   hasilwKNN = JumlahHasilUjiSesuaiRealClass;
69
70   foreach (string a in cetakKelas)
71   {
72   resultLog.AppendText(a);
73   }
74   akurasi(resultLog, dataUji.Count);
75   return "Done!!!";
76  }
```

***Sourcecode* 4.15 Proses klasifikasi dengan WKNN Kelas WKNN**

4.2.6Implementasi Kelas KNN

Kelas KNNterdiri dari method klasifikasiKNN yang berguna untuk pengklasifikasian data uji terhadap data. Proses klasifikasi dengan KNN dapat dilihat pada *sourcecode* 4.16.

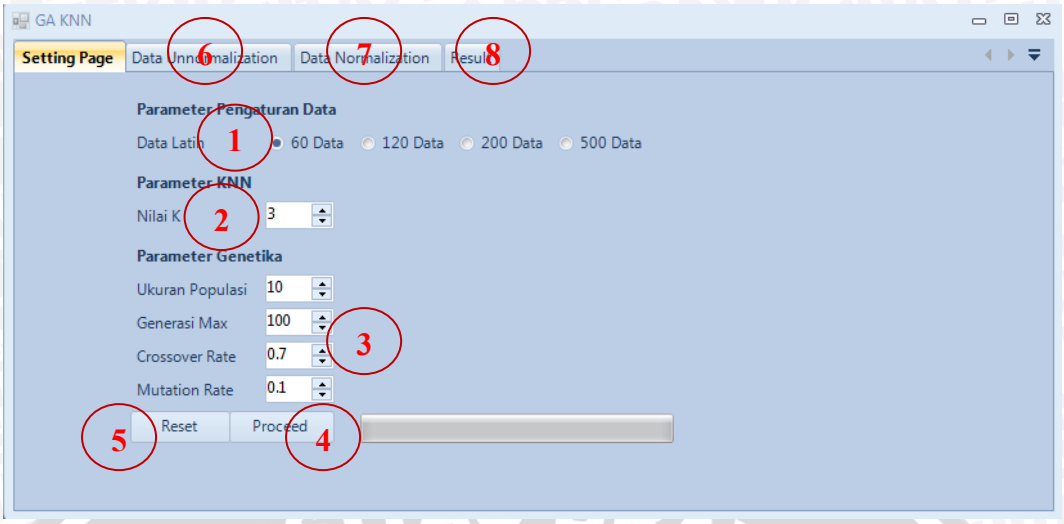
```
1   public string klasifikasiKNN(KryptonRichTextBox resultLog)
2   {
3   double kelas1;
4   double kelas2;
5   double jarak;
6   int JumlahHasilUjiSesuaiRealClass;
7   double[] EuclidDistance = new double[dataLatih.Count];
8   double[] data = new double[dataLatih.Count];
9   double[] urutan = new double[dataLatih.Count];
10  double[] tetangga = new double[dataLatih.Count];
11  string[] prediksi = new string[dataLatih.Count];
12
13  string[] kelasFromDataLatih = new string[dataLatih.Count];
14
15  resultLog.Clear();
16  cetakKelas.Clear();
17   cetakKelas.Add("Keterangan:  1 = not at risk(tak beresiko), 2 = risky(r
    iskan) \n\n");
18   cetakKelas.Add("Data ke-
    \t" + " Kelas Hasil Prediksi \t " + "Kelas Sebenarnya\t\n");
19
20   for (int dataUjiKe = 0; dataUjiKe < dataUji.Count; dataUjiKe++)
21   {
22   for (int i = 0; i < dataLatih.Count; i++) // baris data latih
23   {
24    jarak = compareRecord(dataLatih[i], dataUji[dataUjiKe]);
25
26     EuclidDistance[i] = Math.Round((1 / jarak), 4);
27    }
28
29    for (int x = 0; x < dataLatih.Count; x++)
30    {
31     urutan[x] = EuclidDistance[x];
32     kelasFromDataLatih[x] = Convert.ToString(dataLatih[x].Class); //kelas
    DariDataLatih[x];
```

```
33      }
34
35      Array.Sort(urutan, kelasFromDataLatih);
36      kelas1 = 0;
37      kelas2 = 0;
38
39      for (int z = 0; z < k; z++)
40      {
41      tetangga[z] = urutan[z];
42       if (kelasFromDataLatih[z] == "1")
43       kelas1 += urutan[z];
44       else
45       kelas2 += urutan[z];
46      }
47      if (kelas1 > kelas2)
48       prediksi[dataUjiKe] = "1";
49      else
50      prediksi[dataUjiKe] = "2";
51
52       cetakKelas.Add("  " + (dataUjiKe + 1) + "\t\t" + prediksi[dataUjiKe] +
     "\t\t" + dataUji[dataUjiKe].Class + "\n");
53      }
54      JumlahHasilUjiSesuaiRealClass = 0;
55      for (int indexHasilUji = 0; indexHasilUji < dataUji.Count; indexHasilUj
    i++)
56      {
57       if (prediksi[indexHasilUji] == Convert.ToString(dataUji[indexHasilUji]
    .Class))
58    JumlahHasilUjiSesuaiRealClass++;
59    }
60    hasilKNN = JumlahHasilUjiSesuaiRealClass;
61
62    foreach (string a in cetakKelas)
63    {
64    resultLog.AppendText(a);
65    }
66    akurasi(resultLog, dataUji.Count);
67    return "Done!!!";
68    }
```

***Sourcecode* 4.16 Proses klasifikasi dengan KNN Kelas KNN**

4.3 Implementasi Antarmuka

Tampilan utama dari implementasi *Genetic Algorithm Weighted K-Nearest Neighbour* untuk klasifikasi data kategorik dapat dilihat pada gambar 4.1.



Gambar 4.1 Tampilan Utama Program

Program tersebut terdiri dari beberapa komponen yang mempunyai fungsi sebagai berikut :

1. Tempat untuk memilih jumlah data latih yang ingin diproses. Disediakan 4 *radiobutton* untuk memilih jumlah data.
2. Tempat untuk mengisi parameter KNN yang berisikan nilai k (ketetanggaan).
3. Tempat untuk mengisi parameter genetika yang terdiri dari:
   - Ukuran Populasi
   - Generasi Maksimal
   - Peluang *crossover*
   - Peluang mutasi
4. *Button* untuk memproses perhitungan.
5. *Button* untuk mereset program.
6. *Tab* untuk melihat data – data sebelum dinormalisasi dengan metode *min-max normalization*.
7. *Tab* untuk melihat data – data yang telah dinormalisasi dengan metode *min-max normalization.*
8. *Tab* untuk menampilkan hasil perhitungan yang berupa inisialisasi bobot awal beserta nilai *fitness* untuk masing-masing indvidu awal, hasil pencarian bobot optimal dari *offspring* – *offspring*yang terbentuk melalui proses GA K-nn, hasil akurasi perbandingan metode GA/KNNdengan K-nn murni.

4.4 Sistematika Pengujian

Pada sub-bab ini akan dijelaskan mengenai sistematika pengujian. Sesuai dengan rancangan pengujian pada bab 3, terdapat 4 macam pengujian yang akan dilakukan.

4.4.1 Sistematika Uji Peluang Crossover dan Peluang Mutasi

Pengujian dilakukan dengan mengkombinasikan 3 macam peluang *crossover* dan 4 macam peluang mutasi, dengan nilai PC antara lain 0.5, 0.7 dan 0.9, dan nilai PM antara lain 0.1, 0.2, 0.3 dan 0.4. Pengujian dilakukan sebanyak 5 kali untuk setiap satu kombinasi nilai peluang *crossover* dan nilai peluang mutasi, dan kemudian dihitung nilai rata-rata *fitness*-nya. Nilai rata-rata*fitness* inilah yang akan dibandingankan dengan nilai rata-rata *fitness* pada kombinasi peluang *crossover* dan mutasi yang lain.

Uji coba dilakukan dengan menggunakan nilai K, ukuran populasi, perbandingan data latih data uji dan jumlah konvergen yang sama, berurut-turut yaitu K=3, 10 individu pada populasi, 80% data latih 20% data uji dan dengan *stop condition* yaitu individu terbaik dari tiap generasi berurutanbernilai sama (konvergen) pada 10 generasi berurutan. Kecepatan pelatihan tidak dipertimbangkan sebagai nilai yang akan dianalisis.

4.4.2 Sistematika Uji UkuranPopulasi dan Nilai K (Ketetanggaan)

Pengujian dilakukan dengan mengubah nilai ukuran populasi dari 10, 20,50, 70, 90 dan 100 individu per populasi, dan mengubah nilai K dari K=3, K=5, K=7, dan K=9 untuk mendapatkan nilai *fitness*. Tiap satu kombinasi ukuran populasi dan nilai K dilakukan ujicoba sebanyak 5 kali dengan *stop condition* yaitu individu terbaik dari tiap generasi berurutanbernilai sama (konvergen) pada 10 generasi berurutan, sehingga akan didapatkan rata-rata dari nilai *fitness*. Nilai rata-rata inilah yang akan dibandingkan dengan kombinasi ukuran populasi dan nilai K untuk dianalisa.

Nilai peluang *crossover* dan peluang mutasi yang digunakan dalam pengujian ini berurut-turut adalah 0.7 dan 0.3.

4.4.3 Sistematika Uji Pengaruh Jumlah Data Latih danData Uji

Pengujian dilakukan dengan mengubah komposisi data latih dan data uji dari 80:20, 60:40, 40:60, dan 20:80 untuk mendapatkan nilai *fitness.* Tiap komposisi data latih dan data uji dilakukan ujicoba sebanyak 5 kali dengan *stop condition* yaitu individu terbaik dari tiap generasi berurutanbernilai sama (konvergen) pada 10 generasi berurutan, sehingga akan didapatkan rata-rata dari nilai *fitness*. Nilai rata-rata inilah yang akan dibandingkan dengan komposisi data latih dan data uji untuk dianalisa.

Nilai peluang *crossover* dan peluang mutasi yang digunakan dalam pengujian ini berurut-turut adalah 0.7 dan 0.3 dengan menggunakan nilai K=9 dan 50 individu per populasi.

4.4.4Sistematika Uji Pengaruh Jumlah K (Ketetanggaan) Terhadap Tingkat Akurasi KNN Murni Dan GA/KNN

Pengujian tingkat akurasi dilakukan untuk mengetahui berapa tingkat akurasi sistem dalam melakukan pengklasifikasian. Pengujian dilakukan dengan mengubah nilai K dari K=3, K=5, K=7, dan K=9, dan digunakan konfigurasi bobot terbaik yang didapat dari pelatihanGA/KNN. Kemudian dilakukan pengklasifikasian menggunakan WKnn dan didapat kelas prediksi untuk kemudian dibandingkan dengan kelas aktual(sebenarnya). Error yang didapat akan digunakan sebagai parameter untuk menentukan seberapa besar tingkat akurasi sistem yang telah dibuat dan akan dibandingkan dengan metode Knn murni untuk menentukan seberapa besar tingkat akurasi metode GA/KNNdibandingkan metode Knn murni.

Data yang digunakan untuk pengujian sebanyak 20%dan 80% untuk data latih, nilai peluang *crossover* dan peluang mutasi yang digunakan dalam pengujian ini berurut-turut adalah 0.7 dan 0.3 dengan menggunakan 100 individu per populasi.Dataset yang digunakan merupakan data kredit German yang didapat dari repository UCI machine learning yang dapat diunduh pada alamat http://archive.ics.uci.edu/ml/datasets/Statlog.

4.5 Implementasi Uji Coba dan Analisa Hasil

Pada sub bab ini akan dijelaskan mengenai hasil uji coba yang telah dilakukan. Sesuai dengan sistematika yang telah dipaparkan sebelumnya.

4.5.1 Pengujian Peluang Crossover dan Peluang Mutasi

Berdasarkan sistematika yang telah dijelaskan sebelumnya, pengujian peluang crossover dan peluang mutasi dilakukan pada komposisi nilai PC antara lain 0.5, 0.7 dan 0.9, dengan nilai PM antara lain 0.1, 0.2, 0.3 dan 0.4. Setiap kombinasi nilai peluang crossover dan peluang mutasi dilakukan ujicoba 5 kali dan diambil rata-ratanya. Hasil uji coba dapat dilihat pada tabel 4.2.

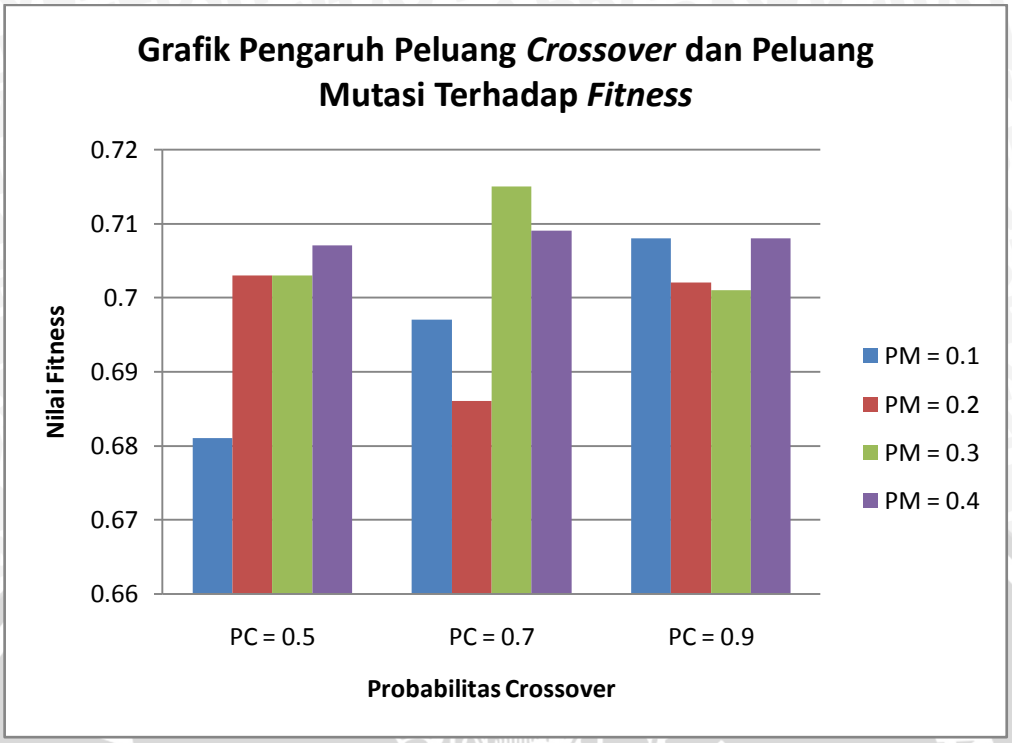
Tabel 4.2 Nilai *fitness* Peluang Crossover dan Peluang Mutasi.

	PM			
PC	0.1	0.2	0.3	0.4
0.5	0.681	0.703	0.703	0.707
0.7	0.697	0.686	0.715	0.709
0.9	0.708	0.702	0.701	0.708

Keterangan :

Data uji yang digunakan sebanyak 20% dan data latih sebanyak 80% dari dataset. Nilai fitness pada penelitian ini adalah jumlah kelas yang diklasifikasikan dengan benar dibagi dengan total data. Jadi rentang fitness antara 0 sampai 1. Nilai 1 berarti semua data uji berhasil diklasifikasi dengan benar.

Perbandingan nilai rata-rata fitness dari masing-masing peluang *crossover* dan mutasi dapat dilihat pada grafik 4.1.



Grafik 4.1 Perbandingan Peluang *Crossover* dan Peluang Mutasi terhadap nilai*fitness*

Pada grafik 4.1 diketahui bahwa peningkatan nilai peluang crossover dan mutasi terbaik pada peluang mutasi sebesar 0.3 dan peluang crossover 0.7.Selanjutnya parameter peluang mutasi dan crossover ini akan digunakan sebagai parameter untuk pengujian selanjutnya.

4.5.2 Pengujian UkuranPopulasi dan Nilai K (Ketetanggaan)

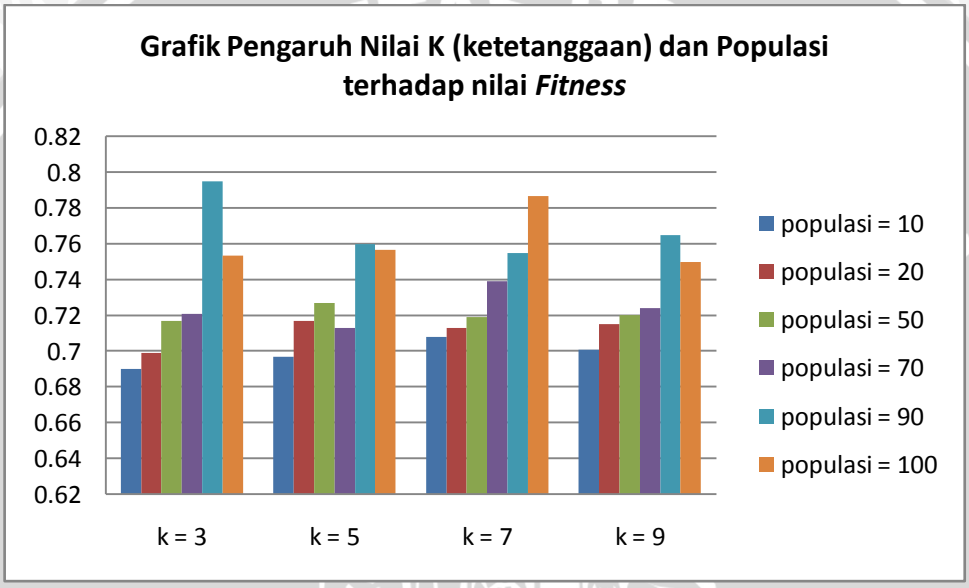
Pengujian ukuranpopulasi dan nilai K menggunakan 4 (empat) nilai K yang akan diujikan, yaitu antara 2-9 dengan ukuran populasi berturut-turut 10, 20, 50, 70, 90, 100. Sesuai dengan sistematika yang telah dijelaskan sebelumnya, tiap komposisi parameter dilakukan 5 kali uji coba untuk kemudian didapatkan rata-ratanya. Berikut hasil uji coba disajikan pada tabel 4.3.

Tabel4.3 Nilai *Fitness* dan Tingkat akurasi pada k (ketetanggan), jumlah populasi, dan jumlah iterasi

	Nilai K			
Jumlah Populasi	3	5	7	9
10	0.69	0.697	0.708	0.701

20	0.699	0.717	0.713	0.715
50	0.717	0.727	0.719	0.72
70	0.721	0.713	0.739	0.724
90	0.795	0.76	0.755	0.765
100	0.753333	0.756667	0.786667	0.75

Perbandingan nilai rata-rata tingkat akurasi dari masing-masing nilai k (ketetanggaan), jumlah populasi dan jumlah iterasi dapat dilihat pada grafik 4.2



Grafik 4.2 Perbandingan nilai k (ketetanggaan) danukuran populasi terhadap nilai *fitness*

Berdasarkan hasil uji coba diketahui bahwa ukuran populasi tidak menjamin terhadap perubahan *fitness* untuk menjadi semakin baik, tetapi akan memberikan peluang untuk menghasilkan individu unggul lebih besar.Hal ini disebabkan tiap populasi memiliki start point individu yang berbeda-beda, sehingga populasi yang memiliki individu unggul pada saat pembentukan populasi awal akan selalu diuntungkan sampai populasi mencapai titik konvergensi.

Pada grafik 4.2 diketahui bahwa individu (solusi) terbaik terdapat pada populasi sebesar 90 dengan nilai K = 3.

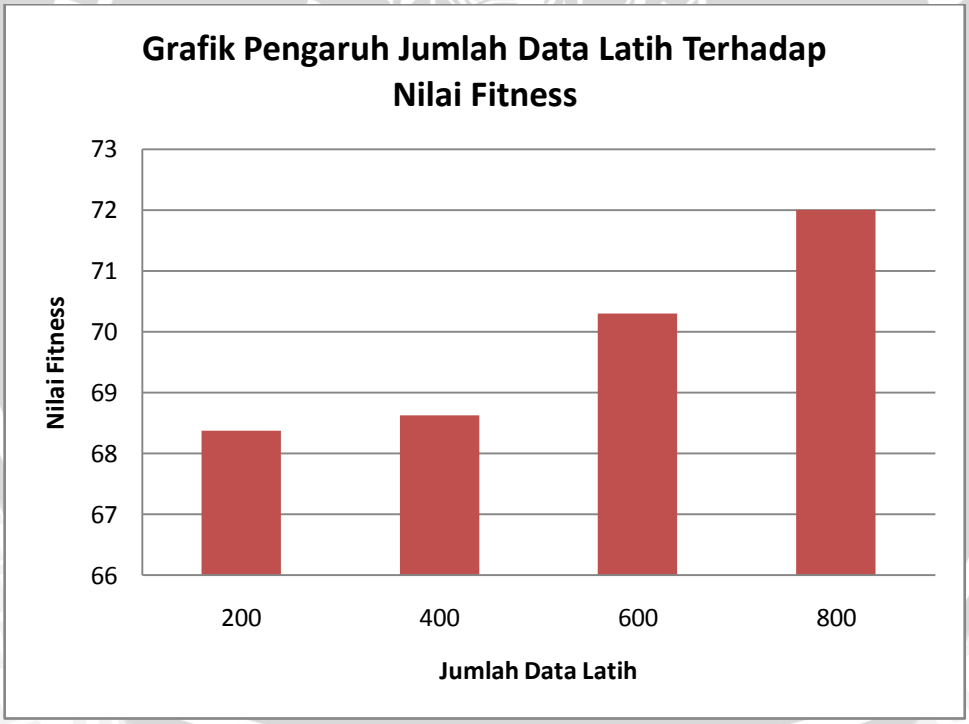
4.5.3 Pengujian Pengaruh Jumlah Data Latih danData Uji

Pengujian selanjutnya adalah uji pengaruh jumlah data latih dan data uji. Digunakan 4(empat) macam varian data latih data uji dengan komposisi berturut-turut 20:80, 40:60, 60:40, dan 80:20, dimana setiap komposisi diujikan sebanyak 5 kali untuk kemudian nilai fitnessyang didapat akan dirata-rata. Berikut hasil uji coba ditunjukkan pada tabel 4.4.

Tabel 4.4Nilai Fitness Pada Jumlah Data Latih yang Berbeda

No.	Jumlah Data Latih	Jumlah Data Uji	Nilai Fitness (%)
1	200	800	68.378
2	400	600	68.628
3	600	400	70.3
4	800	200	72

.



Grafik 4.3Perbandingan jumlah data latih terhadap nilai fitness

Berdasarkan hasil uji coba diketahui bahwa jumlah data latih yang semakin besar menjamin terhadap perubahan nilai fitness semakin besar. Hal ini disebabkan proses pembelajaran yang semakin efektif bila didukung dengan referensi yang lebih banyak.

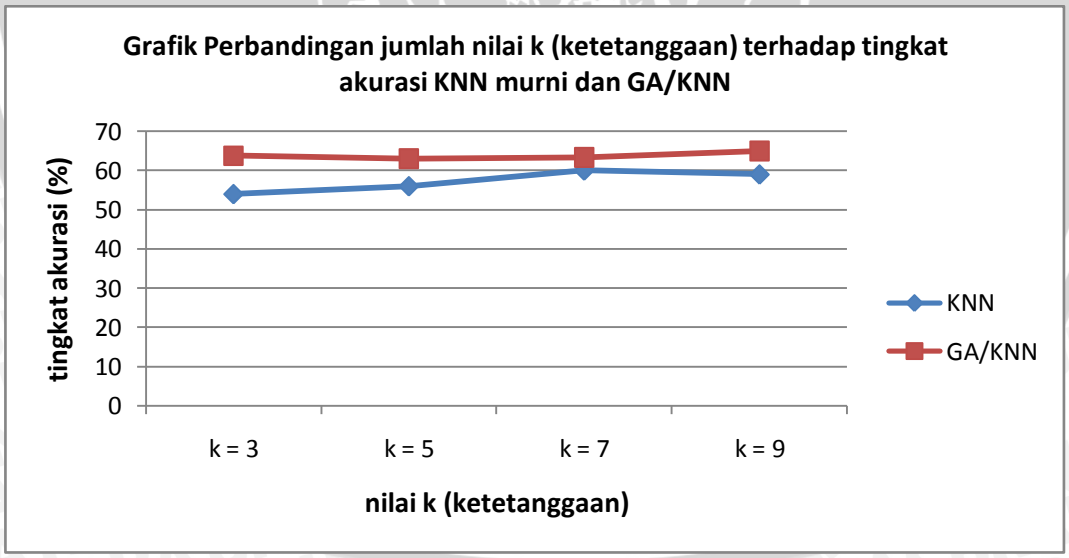
4.5.4 Pengujian Pengaruh Jumlah K (Ketetanggaan) Terhadap Tingkat Akurasi KNN Murni dan GA/KNN

Pada pengujian keempat, dilakukan perbandinganjumlah nilai k (ketetanggaan) terhadaptingkat akurasi KNN murni dan tingkat akurasi GA/KNN. Sesuai dengan sistematika yang telah dijelaskan sebelumnya, tiap komposisi parameter dilakukan 5 kali uji coba untuk kemudian didapatkan rata-ratanya. Berikut ini hasil uji coba untuk masing-masing nilai k.

Tabel 4.5Perbandingan Nilai k (ketetanggaan) terhadaptingkat akurasi KNN murni dan tingkat akurasi GA/KNN

No.	Nilai k (ketetanggaan)	Tingkat Akurasi KNN (%)	Tingkat Akurasi GA/KNN (%)
1	k = 3	54	63.75
2	k = 5	56	63
3	k = 7	60	63.333
4	k = 9	59	65

Perbandingan nilai rata-rata tingkat akurasi KNN murni dan GA/KNNterhadap jumlah k (ketetanggaan) dapat dilihat pada grafik 4.4.



Grafik 4.4Perbandingan jumlah nilai k (ketetanggaan) terhadap tingkat akurasi KNN murni dan GA/KNN.

Hasil dari pengujian ini menunjukkan bahwa algoritma GA/KNNmemiliki performa yang lebih baik daripada algoritma KNN murni

untuk mengklasifikasi data numerik dan kategorik, dikarenakan algoritmaGA/KNN mengalami pelatihan bobot terlebih dahulu, sehingga tidak terlalu sensitif terhadap noise dan memiliki bobot proporsional untuk setiap atribut yang mendekati nilai aktual dari kredit *scoring*.

BAB V
PENUTUP

5.1 Kesimpulan

Setelah melakukan penelitian maka dapat disimpulkan bahwa :

1. Sistem klasifikasi data kredit German yang telah dibuat dengan menggunakan metode *Genetic Algorithm K-Nearest Neighbour* (GA/KNN) dilakukan dengan 3 tahap utama. Tahap pertama yaitu proses normalisasi dataset pada atribut bertipe numerik menggunakan metode *minmax normalization* untuk standarisasi di antara atribut-atribut numerik. Tahap kedua yaitu pelatihan dengan menggunakan *Genetic Algorithm K-Nearest Neighbour* (GA/KNN) untuk menemukan bobot optimal dimana pada proses ini dilakukan inisialisasi populasi awal secara random, seleksi induk dengan menggunakan Roullete Wheel, persilangan dengan menggunakan *Restricted Crossover*, dan mutasi dengan menggunakan *Exchange Mutation* pada gen. Tahap ketiga yaitu klasifikasi data uji pada suatu kelas dengan menggunakan Euclidian Distance, algoritma klasifikasi WKNN dan menggunakan bobot terbaik hasil dari pelatihan pada tahap kedua.
2. Tingkat akurasi pada metode *GeneticAlgorithm K-Nearest Neighbour*(GA/KNN) dipengaruhi oleh beberapa parameter sebagai berikut :
   a. Nilai fitness terbaik yang berada pada peluang *crossover* 0.7 dan peluang mutasi 0.3 dengan nilai fitness sebesar 0.714
   b. Nilai fitness terbaik yang berada pada nilai k (ketetanggaan) sebesar 5, jumlah populasi 90 dengan tingkat konvergen 10 generasi berurutan, dengan nilai fitness sebesar 0.795.
   c. Meningkatnya jumlah data latih mempengaruhi peningkatan nilai fitness, karena dengan semakin banyaknya data latih, maka kemungkinan banyaknya jarak record yang mendekati kelas data prediksi adalah semakin tinggi.

d. Tingkat akurasi GA/KNN lebih baik dari tingkat akurasi KNN murni. Akurasi tertinggi GA/KNN mencapai 65 % sedangkan KNN murni sebesar 59 %.

5.2 Saran

Pada penelitian ini masih ada beberapa hal yang dapat dikembangkan dan digunakan untuk penelitian selanjutnya yaitu antara lain :

1. Membandingkan dengan data set bertipe kategorik yang lain.
2. Adanya proses penanganan terhadap *missing value*.
3. Penggunaan metode perhitungan kemiripan yang lain.
4. Penggunaan operator genetika yang lain.
5. Penggunaan fungsi *fitness* yang berbeda.

DAFTAR PUSTAKA

[ALI-10] Al Irsyad, A. A. 2010. *Estimasi Stok Multi Kriteria dengan Menggunakan Algoritma Genetika*. Institut Teknologi Sepuluh Nopember. Surabaya.

[CYH-04] Cyhe, K.H., Chin, T.W., dan Peng, G.C., 2004.*Credit Scoring Using Data Mining Techniques*. Singapore Management Review 26 (2): 25-47.

[CUN-07] Cunningham, P dan Delany, S.J. 2007. *K-Nearest Neighbour Classifier*. Technical Report UCD-CSI-2007-4.

[EIB-03] Eiben, A.E dan Smith J.E.2003.*Introduction to Evolutionary Computing*. Springer. Amsterdam.

[GEN-97] Gen,M dan Cheng,R.1997.*Genetic Algorithm & Engeneering Design*. Ashikawa Institut of Technology. Japan.

[HAN-06] Han,J dan Kamber,M.2006.*Data Mining: Concept and Techniques Second Edition*. University of Illinois at Urbana Champaign. Morgan Kauffman publishers San Fransisco.

[HAS-02] Hasibuan Malayu S. P. 2002. *Dasar-dasar Perbankan*. Bumi Aksara.Jakarta

[KAS-03] Kasmir. 2003. *Manajemen Perbankan, Edisi Keempat*. PT. Raja Grafindo Persada. Jakarta.

[KUS-03] Kusumadewi, S. 2003. *Artificial Intellegence (Teknik dan Aplikasi Edisi pertama)*. Graha ilmu. Jogjakarta.

[LAR-05] Larose, DT.2005. *Discovering Knowledge in Data:An Introduction to Data Mining*. John Willey & Sons, Inc New Jersey.

[LUK-05] Lukas, S. 2005. *Penerapan Algoritma Genetika untuk TSP dengan Menggunakan Metode Order Crossover dan Insertion Mutation*. Seminar Nasional Aplikasi Teknologi Informasi.

[MOR-09] Moradian, M dan Baarani, A. 2009.*KNNBA: k-Nearest Neighbor BasedAssociation Algorithm*http://www.jatit.org/volumes/research-papers/Vol6No1/14Vol6No1.pdf. Tanggal Akses 14 Mei 2012.

[NUR-06] Nurjaya, W. 2006. *Analisis Proses Word Matching Problem Menggunakan Algoritma Genetika*. Majalah Ilmiah Unikom Vol. 6. Jurusan Manajemen Informatika Univ. Komputer Indonesia.

[NUR-07] Nurwijaya. 2007. *Analisis Penggunaan Algoritma Genetika Untuk Optimalisasi Jaringan Syaraf Tiruan*. Institut Teknologi Bandung. Bandung.

[OBI-98] Obitko, M. 1998. *Introduction to Genetic Algorithm.*http://www.obitko.com/tutorials/genetic-algorithms/. Tanggal Akses 20 Mei 2012.

[PER-03] Peraturan Bank Indonesia No. 5/8/PBI/2003, *Penerapan Manajemen Risiko Bagi Bank Umum,* 2003.

[PRA-03] Pramudiono, I. 2003. *Pengantar Data Mining : Menambang Permata Pengetahuan di Gudang Data.* http:// www.ilmukomputer.com, Tanggal akses 3 Mei 2012.

[PRO-09] Provorova, I. 2009. *Using Genetic Algorithm to Optimize weights in Data Mining Task.* Riga Technical University.

[SAR-00] Sarkar, M dan Leong, T.2000. *Application of K-Nearest Neighbors Algorithm on Breast Cancer Diagnosis Problem.*The National University of Singapore. Singapore.

[SUL-09] Sulistiowati, D. 2009. *Estimasi Kolektibitas, Performance kredit Dan Perbandingan Matriks Transisi Pada Kredit Perbankan Tahun 2008*. Universitas Indonesia. Jakarta.

[SUM-05] Sumantrika, N P. 2005. *Data Mining Task Klasifikasi Menggunakan Algoritma Genetika*. Skripsi. Jurusan Teknik Informatika Sekolah Tinggi Telkom Bandung.

[SUY-05] Suyanto. 2005. *Algoritma Genetika dalam Matlab.* Andi. Yogyakarta.

[TRO-00] Trovald,N.2000. *Algoritma Genetika Dasar Komputasi Cerdas*. Universitas Udayana. Bali.

[YEN-08] Yento, N S. 2008. *Pemakaian Algoritma Genetik untuk Penjadwalan Job Shop Dinamis Non Deterministik.* Jurusan Ilmu Komputer – FMIPA, Universitas Katolik Parahyangan. Tanggal akses: 20 Mei 2012.

[WRO-10] Wroniszewska dan Dziedzic.2010. *Voice Command Recognition Using Hybrid Genetic Algorithm.*Task Quarterly 14 No 4