# MENILAI KELAYAKAN DAN KEAKURATAN ATURAN *MINING* YANG DIHASILKAN ALGORITMA GENETIKA

**SKRIPSI**

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


oleh :
**Indra Arta Kusuma**
0410960026-96



**PROGRAM STUDI ILMU KOMPUTER**
**JURUSAN MATEMATIKA**
**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**
**UNIVERSITAS BRAWIJAYA**
**MALANG**

**2009**

# DAFTAR ISI

Halaman

**BAB III METODOLOGI DAN PERANCANGAN**

## DAFTAR GAMBAR

# DAFTAR TABEL

## DAFTAR LAMPIRAN

# MENILAI KELAYAKAN DAN KEAKURATAN ATURAN *MINING* YANG DIHASILKAN ALGORITMA GENETIKA

**SKRIPSI**

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


oleh :
**Indra Arta Kusuma**
0410960026-96




**PROGRAM STUDI ILMU KOMPUTER**
**JURUSAN MATEMATIKA**
**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**
**UNIVERSITAS BRAWIJAYA**
**MALANG**

**2009**

UNIVERSITAS BRAWIJAYA

**LEMBAR PENGESAHAN**

**MENILAI KELAYAKAN DAN KEAKURATAN ATURAN *MINING* YANG DIHASILKAN ALGORITMA GENETIKA**

Oleh :
**Indra Arta Kusuma**
**0410960026-96**

**Setelah dipertahankan di depan Majelis Penguji**
**Pada tanggal 4 Februari 2009**
**dan dinyatakan memenuhi syarat untuk memperoleh gelar**
**Sarjana dalam bidang Ilmu Komputer**

| Pembimbing I | Pembimbing II |
|---|---|
| **Dian Eka R, S.Si., M.Kom.** | **Djoko Pramono, S.T.** |
| **NIP. 132 300 224** | **NIP. 132 313 603** |

**Mengetahui,**
**Ketua Jurusan Matematika**
**Fakultas MIPA Universitas Brawijaya**

**Dr. Agus Suryanto, M.Sc.**
**NIP. 132 126 049**

UNIVERSITAS BRAWIJAYA

**LEMBAR PERNYATAAN**

**Saya yang bertanda tangan di bawah ini :**

**Nama : Indra Arta Kusuma**
**NIM : 0410960026-96**
**Jurusan : Matematika**
**Penulis skripsi berjudul : Menilai Kelayakan dan Keakuratan Aturan *Mining* yang Dihasilkan Algoritma Genetika**

**Dengan ini menyatakan bahwa :**

1. **Isi dari skripsi yang saya buat adalah benar-benar karya sendiri dan tidak menjiplak karya orang lain, selain nama-nama yang termaktub di isi dan tertulis di daftar pustaka dalam skripsi ini.**
2. **Apabila di kemudian hari ternyata skripsi yang saya tulis terbukti hasil jiplakan, maka saya akan bersedia menanggung segala resiko yang akan saya terima.**

**Demikian pernyataan ini dibuat dengan segala kesadaran.**

**Malang, 9 Februari 2009**
**Yang menyatakan,**

**( Indra Arta Kusuma )**
**NIM. 0410960026-96**

**MENILAI KELAYAKAN DAN KEAKURATAN ATURAN *MINING* YANG DIHASILKAN ALGORITMA GENETIKA**

## ABSTRAK

Telah terjadi pertumbuhan data yang sangat besar pada segala bidang. Realitanya, hanya sedikit data yang akan digunakan karena ukuran datanya yang terlalu besar dan terlalu kompleks untuk diolah. Padahal kebutuhan untuk menggali informasi yang tersembunyi pada data adalah sangat penting.

*Data Mining* bertujuan menggali informasi yang akurat dan lengkap dari *database* yang besar. Klasifikasi adalah salah satu metode *Data Mining* yang menghasilkan aturan dari *database*. Dalam banyak penelitian tentang klasifikasi, hanya berkisar pada pencarian keakuratan aturan yang terukur dengan tingkat kesalahan klasifikasi. Maka pada skripsi ini, penekanannya adalah menilai kelayakan sekaligus keakuratan aturan *IF-THEN*.

Pada sistem, Algoritma Genetika mengimplementasikan teknik klasifikasi aturan dengan pembangkitan populasi awal secara terstruktur dan perhitungan nilai *fitness* berparameter kelayakan *antecedent-consequent* dan keakuratan prediksi untuk menghasilkan aturan *IF-THEN* dengan atribut tujuan tertentu.

Tahapan awal akan dilakukan pengkondisian data-contoh sesuai tipe masukan dari Algoritma Genetika. Tahapan pengkondisian data ini terdiri dari pengisian nilai hilang dan pendiskritan nilai kontinu. Pada tahapan kedua akan dilakukan operasi genetika untuk mencari aturan serta menghitung nilai kelayakan dan keakuratannya dari data-contoh.

Pengujian pada sistem dilakukan untuk mengetahui pengaruh parameter jumlah populasi terhadap nilai kelayakan *antecedent* (*AI*), nilai kelayakan *consequent* (*CI*), nilai keakuratan prediksi (*PA*), dan nilai *Fitness*. Dan untuk mengetahui pengaruh parameter persentase data-latih dan persentase data-uji terhadap nilai persentase kebenaran aturan dari data-uji (*PC*).

Hasil dari penelitian ini yakni kenaikan nilai parameter jumlah populasi mempengaruhi tingginya nilai kelayakan *antecedent* (*AI*) dari aturan dengan nilai *AI* paling maksimal=0,99 pada jumlah populasi ke-80.

Perubahan nilai parameter jumlah populasi tidak mempengaruhi nilai kelayakan *consequent* (*CI*) dari aturan. Begitu juga perubahan nilai persentase data-latih tidak mempengaruhi perubahan nilai kelayakan *CI*.

Kenaikan nilai parameter jumlah populasi mempengaruhi tingginya nilai prediksi keakuratan (*PA*) dari suatu aturan dengan nilai *PA* paling maksimal=0,96 pada jumlah populasi ke-60 sampai ke-80.

Kenaikan nilai parameter jumlah populasi mempengaruhi tingginya nilai *fitness* dari aturan, dengan nilai *fitness* paling maksimal =0,93 pada jumlah populasi ke-70.

Pada data-contoh *Crx*, kenaikan nilai parameter jumlah populasi menghasilkan rendahnya nilai persentase kebenaran aturan dari data-uji (*PC*). Hal ini disebabkan adanya nilai hilang, nilai kontinu, dan atribut berisi sebaran banyak nilai; sedang yang paling besar pengaruhnya terhadap rendahnya nilai *PC* pada data-contoh *Crx* yakni atribut berisi sebaran banyak nilai.

Kenaikan nilai persentase data-uji mempengaruhi terhadap tingginya nilai persentase kebenaran aturan dari data-uji (*PC*), terutama pada posisi persentase data-uji 100% dengan nilai *PC*=11,02.

**MINING RULES OF INTERESTING AND ACCURATING VALUATION WITH GENETIC ALGORITHM**

## ABSTRACT

We have grown tremendous volumes of data filling of all fields. In reality, only a small amount of these data will ever be used because the volumes are too large and too complicated to manage.

Data mining consists of the discovery of highly accurate and comprehensible knowledge from large databases. Classification Task consists of Data Mining methods that induce rules from a database. However, in many classification research, the emphasis is discovery of accurate knowledge as measured by the classification error rate. In this undergraduate thesis, the emphasis is discovering interesting and accurate IF-THEN rules.

In the system, Genetic Algorithm is implemented for classication rule technique using a generalized uniform population method and a fitness function with the measures interestingness of the antecedent-consequent of the rule and the measures its predictive accuracy, that discovers comprehensible IF-THEN rules of the value of a goal attribute.

The first term is working in preprocessing the dataset appropriate with input characteristic of Genetic Algorithm. In preprocessing the dataset consist of the miss values filling and the continuous values discretization. And the second term is working in genetic processing to discovers and measures the degree of interestingness and accurateness of the rule of dataset.

The testing of system as a purpose to research the parameter influence of the number of populations concerning the degree of antecedent interestingness (*AI*), the degree of consequent interestingness (*CI*), the predictive accuracy (*PA*), and the fitness function values. And to research the parameter influence of the percentage of training sets and the percentage of testing sets concerning the percentage of probability correct of testing sets (*PC*).

The result of this research is the increases of parameter values of the number of populations influences the height of the degree of antecedent interestingness (*AI*) of rules with the maksimum of *AI*=0,99 in the number of populations 80$^{th}$.

The variation of parameter value of the number of populations not influences of the degree of consequent interestingness (*CI*) of rules. And the same conclusions goes for the variation of parameter values of the percentage of training sets not influences of the degree of consequent interestingness (*CI*) of rules.

The increases of parameter values of the number of populations influences the height of the degree of predictive accuracy (*PA*) of rules with the maksimum of *PA*=0,96 in the number of populations 60$^{th}$ until the number of populations 80$^{th}$.

The increases of parameter values of the number of populations influences the height of the fitness function values of rules with the maksimum of fitness value=0,93 in the number of populations 70$^{th}$.

In the *Crx* dataset, the increases of parameter values of the number of populations influences the low of the degree of the probability correct of rules of testing test (*PC*). It happened because is contained the missing values, the continuous value, and the attributes is consist of distributions many of values; whereas the highest of influential of concerning the low of values of *PC* is the attributes is consist of distributions many of values.

The increases of the percentage of testing test influences the height of the values of percentage of the probability correct of testing sets, especially in the percentage of testing test is 100% with the values of *PC* =11,02.

## Kata Pengantar

Segala puji bagi Alloh yang tiada henti memberikan nikmat-Nya kepada penulis sehingga skripsi dengan judul "Menilai Kelayakan dan Keakuratan Aturan *Mining* yang Dihasilkan Algoritma Genetika" ini dapat terselesaikan. Sholawat dan salam terus terucap untuk Rosulullah, keluarga, serta para sahabatnya.

Skripsi ini disusun dan diajukan sebagai syarat untuk memperoleh gelar sarjana pada Program Studi Ilmu Komputer, Jurusan Matematika, Fakultas MIPA, Universitas Brawijaya.

Sebagai ungkapan rasa syukur, penulis perlu mengucapkan banyak terima kasih kepada semua pihak yang sangat membantu baik moral dan materiil dalam penyelesaian skripsi ini. Ucapan terima kasih, penulis sampaikan kepada :

1. Dian Eka R., S.Si., M.Kom., selaku pembimbing utama penulisan skripsi dan Penasihat Akademik. Terima kasih atas arahan dan masukannya selama penulisan skripsi berlangsung. Dan terima kasih pula atas bimbingannya dalam kegiatan keakademikan.
2. Djoko Pramono, S.T., selaku pembimbing pendamping penulisan skripsi, yang banyak membantu dalam mengarahkan penulisan sesuai sistematika penulisan skripsi yang benar.
3. Dr. Agus Suryanto, M.Sc., selaku ketua Jurusan Matematika Fakultas MIPA Universitas Brawijaya.
4. Wayan Firdaus Mahmudy, S.Si., M.T., selaku Ketua Program Studi Ilmu Komputer Universitas Brawijaya.
5. Seluruh Bapak dan Ibu dosen yang dengan sabar mendidik dan memberikan banyak ilmu yang teramat berharga kepada penulis.
6. Segenap staf dan karyawan Jurusan Matematika Fakultas MIPA Universitas Brawijaya atas semua kerjasamanya.
7. Ibu, Bapak, dan Saudara yang memberikan dukungan luar biasa yang terus-menerus meskipun telah selesai pengerjaan skripsi.
8. Ustadz Abdulloh Sholeh al-Hadromi dengan Radio Dakwah 100.5 FM dan Ustadz Agus Hasan dengan Qiblatinya, terima kasih atas kunci sukses hidupnya. *Uhibbukuma fillah*.
9. Ikhwan-ikhwan eks Asrama Kopma, Anif Sumenep, Qolbun Salim, Muhajirin, AsSalam, AsSunnah, Mujahidin, Manarul, Forkalam, dan semuanya, terima kasih atas *Ukhuwwah Islamiyyah*-nya.

10. Teman-teman Ilkom semua angkatan, lebih khusus pinjaman laptop kencengnya Anung Hendhi S.Kom., laptop antiknya Andi Reza, monitor 14" milik Muhlasin, dan rekan main badminton Rabu pagi, terima kasih atas kekompakannya.
11. Serta semua pihak yang tidak bisa penulis sebutkan satu-persatu, atas seluruh bantuannya dalam penyelesaian skripsi ini, semoga Alloh membalasnya dengan sebaik-baik balasan, amin.

Penulis sangat berharap, agar skripsi ini tidak hanya tersimpan dalam lemari buku yang terkunci rapat. Tiada satupun pembaca yang berminat. Akan tetapi, semoga skripsi ini bisa bermanfaat. Bagi yang ingin meraih penjelasan Algoritma Genetika secara singkat. Penulis menyadari, bahwa skripsi ini masih banyak hal yang kurang tepat. Dalam penjelasan dan penulisannya yang mengandung banyak kejanggalan dan cacat. Oleh karenanya, penulis menyatakan tiada kata terlambat. Bagi para pembaca yang ingin menyampaikan saran dan kritik yang membangun, bukan menggugat. Semoga balasan yang terpuji dari yang Maha Rahmat. Kepada para pembaca atas kesediaan waktunya yang singkat. Dalam kebersamaannya untuk menggulirkan suatu karya tulis yang lebih hebat. Dari apa yang bisa penulis buat. Terima kasih sekali lagi untuk para pembaca yang cermat.

Malang, 9 Februari 2009

Penulis

# BAB I
# PENDAHULUAN

## 1.1 Latar Belakang Masalah

Teknologi *database* telah berkembang dari pemrosesan *file* yang sederhana menuju pengembangan sistem manajemen *database* dengan *query*. Pemrosesan *file* selanjutnya dituntut untuk meningkatkan nilai efesien dan efektif dalam analisa data. Hal ini dibutuhkan karena pertumbuhan data yang sangat besar yang terkumpul dari banyak aplikasi seperti manajemen dan bisnis, administrasi pemerintahan, ilmu pengetahuan teknik, dan kontrol lingkungan (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 sangat menguntungkan dari sudut pandang penelitian, bisnis, dan lainnya. *Data Mining* diharapkan menghasilkan informasi yang sangat akurat dan mudah dipahami. Salah satu teknik *Data Mining* yaitu teknik klasifikasi. Teknik klasifikasi bertujuan menemukan sekumpulan aturan yang mendefinisikan label kelas yang tidak diketahui (Han dan Kamber, 2006).

Metode penyelesaian Algoritma Genetika akan diterapkan untuk menemukan aturan-aturan dari data-contoh. Algoritma Genetika memiliki pendekatan solusi terbaik pada banyak kasus yang berbeda, sehingga sering diterapkan pada banyak aplikasi. Algoritma Genetika menemukan aturan yang lebih umum, berbeda dengan metode *Data Mining* lain, dimana algoritmanya melakukan jenis pencarian lokal (Korkut, 2004).

Dalam banyak literatur, pembahasan teknik klasifikasi hanya menekankan pada perhitungan keakuratan aturan, yang terukur dengan tingkat kesalahan klasifikasi (Alatas, 2002). Seperti penelitian teknik klasifikasi dengan Algoritma Genetika oleh Ngurah Putu Sumantrika (2005), yang hanya menggunakan perhitungan keakuratan aturan untuk menilai aturan. Kelemahan dari penelitian oleh Sumantrika ini, tidak menghasilkan aturan yang akurat sekaligus layak. Maka dalam skripsi ini, penekanannya adalah menilai kelayakan sekaligus keakuratan aturan *IF antecedent* (kondisi) *THEN consequent* (label hasil) menurut parameter kelayakan *antecedent-consequent* dan keakuratan prediksi (*predictive accuracy*) untuk mendapatkan aturan yang akurat dan layak..

Pada tahapan awal akan dilakukan pengkondisian data-contoh sesuai masukan dari Algoritma Genetika. Tahapan pengkondisian data ini terdiri dari pengisian nilai hilang (*missing value*) dan pendiskritan nilai kontinu. Pada tahapan kedua akan dilakukan operasi genetika untuk mencari aturan serta menghitung kelayakan dan keakuratannya dari data-contoh. Operasi genetika ini dimulai dengan pembangkitan populasi awal (*initial population*) berupa aturan-aturan secara terstruktur (*uniform*) bukan secara acak (*random*). Pembangkitan populasi awal secara acak terdapat banyak kekurangan, yakni bisa membentuk kromosom pada daerah yang berdekatan dan jauh dari daerah solusi atau pencarian solusi hanya berkisar pada daerah solusi lokal (Korkut, 2004).

Berikutnya, dari populasi awal yang dibangkitkan tersebut akan diproduksi banyak generasi turunan baru yang diharapkan memiliki nilai *fitness* lebih baik dari induknya. Nilai *fitness* pada tiap kromosom dihitung dari parameter *antecedent, consequent,* dan *predictive accuracy,* sehingga dari pengamatan terhadap ketiga parameter ini dan parameter lainnya yakni persentase kebenaran aturan bentukan data-latih terhadap data-uji maka dapat diketahui dengan mudah nilai kelayakan dan keakuratan dari aturan (representasi dari kromosom).

Pada sistem yang akan dibangun pada skripsi ini, aturan-aturan yang berhasil dibentuk akan menyimpulkan nilai label untuk satu atribut tujuan (*goal attributes*) yang secara bebas telah ditentukan pengguna, tidak seperti umumnya teknik klasifikasi dimana semua aturan hanya menyimpulkan satu atribut tujuan tertentu saja, misalnya *class*.

Berdasarkan latar belakang yang dipaparkan tersebut, maka skripsi ini diberi judul **"Menilai Kelayakan dan Keakuratan Aturan *Mining* yang Dihasilkan Algoritma Genetika"**.

## 1.2 Rumusan Masalah

Rumusan masalah dalam skripsi ini yakni bagaimana kelayakan dan keakuratan aturan untuk atribut tujuan tertentu yang dihasilkan oleh Algoritma Genetika, dengan pembangkitan populasi awal secara terstruktur dan perhitungan nilai *fitness* menggunakan paramater kelayakan *antecedent-consequent* serta parameter keakuratan prediksi.

## 1.3 Batasan Masalah

Batasan masalah dalam skripsi ini meliputi :

1. Pengujian sistem menggunakan data-contoh (*dataset*) yaitu *Crx*. Data-contoh ini bisa didownload dari situs http://www.ics.uci.edu/~mlearn/MLRepository.html.
2. Tahapan pengkondisian data-contoh meliputi pengisian nilai hilang yang disimbolkan '?' dan pendiskritan nilai kontinu.
3. Pembangkitan populasi awal dilakukan secara terstruktur untuk menghasilkan aturan dengan atribut tujuan tertentu.
4. Operator Genetika yang diterapkan meliputi Persilangan Terstruktur; mutasi acak terhadap bit, gen, atau kromosomnya; dan Seleksi Roda *Roullette* (*Roullette Wheel Selection*).

## 1.4 Tujuan Penelitian

Tujuan yang hendak dicapai pada skripsi ini meliputi :

1. Pengimplementasian Teknik Klasifikasi *Data Mining* menggunakan Metode Algoritma Genetika dengan pembangkitan populasi awal secara terstruktur dan perhitungan nilai *fitness* berparameter kelayakan dan keakuratan aturan, untuk menghasilkan aturan dengan atribut tujuan tertentu.
2. Penganalisaan nilai kelayakan dan keakuratan aturan berdasarkan parameter kelayakan *antecedent-consequent*, keakuratan prediksi (*predictive accuracy*), dan persentase kebenaran terhadap data-uji.

## 1.5 Manfaat Penelitian

Manfaat dari skripsi ini meliputi :

1. Manfaat akademis :
   a. Memahami dan menerapkan Teknik Klasifikasi Aturan *Data Mining* dengan Algoritma Genetika ke dalam perangkat lunak dengan beberapa pengembangan kasus.
   b. Memberikan tambahan referensi bagi para peneliti untuk mengembangkan lebih lanjut mengenai penerapan Teknik Klasifikasi Aturan *Data Mining* dengan Algoritma Genetika.
2. Manfaat praktis ialah memberikan pendekatan solusi terbaik berupa klasifikasi pola dan aturan pada database yang besar untuk mendapatkan suatu informasi tersembunyi yang sangat berharga.

## 1.6 Metodologi Penelitian

Metodologi untuk menyelesaikan rumusan masalah dalam skripsi ini meliputi :

1. Studi Literatur
   Melakukan pendalaman materi yang berhubungan dengan Algoritma Genetika dan Klasifikasi Aturan *Data Mining*.
2. Pendefinisian dan Analisa Masalah
   Mendefinisikan dan menganalisa rumusan masalah untuk mencari solusi yang tepat sesuai studi literatur.
3. Perancangan dan Implementasi Sistem
   Merancang dan mengimplementasikan solusi atas rumusan masalah berupa perangkat lunak.
4. Pengujian Sistem dan Analisa Hasil
   Menguji coba perangkat lunak yang telah dibangun dan menganalisa hasil yang didapat sesuai parameter yang telah ditentukan sebelumnya.

## 1.7 Sistematika Penulisan

Sistematika penulisan untuk menyusun skripsi ini meliputi :

1. BAB I *PENDAHULUAN*
   Merinci latar belakang masalah, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian dan sistematika penulisan.
2. BAB II *TINJAUAN PUSTAKA*
   Menyertakan berbagai teori pendukung untuk menyelesaikan rumusan masalah dalam skripsi.
3. BAB III *METODE DAN PERANCANGAN*
   Membahas penerapan teori-teori pada tinjauan pustaka untuk merancang perangkat lunak serta rancangan alur kerja sistem.
4. BAB IV *HASIL DAN PEMBAHASAN*
   Melakukan pengujian data-contoh dan menganalisa hasil yang didapat oleh sistem sesuai parameter yang telah ditentukan.
5. BAB V *PENUTUP*
   Mencari kesimpulan dari keseluruhan tahapan metodologi penelitian dan memberikan saran untuk pengembangan sistem lebih lanjut.

# BAB II
# TINJAUAN PUSTAKA

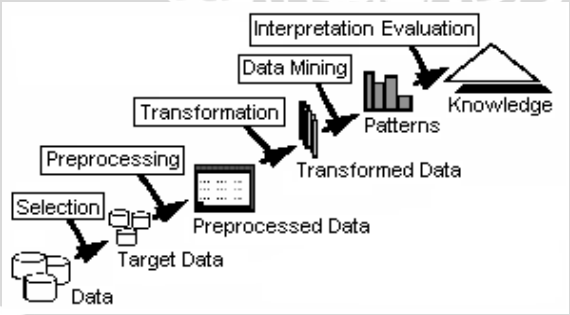
Pembahasan dalam bab ini mencakup tinjauan pustaka yang dijadikan dasar acuan dalam penyelesaian rumusan masalah dalam skripsi ini.

## 2.1 *Data Mining*

Telah terjadi pertumbuhan volume data yang sangat besar pada segala bidang. Realitanya, hanya sedikit data yang akan digunakan. Karena pada banyak kasus, data yang ada terlalu besar sehingga sulit dianalisa. Padahal dalam dunia bisnis, informasi tersembunyi pada data tersebut merupakan aset yang berharga dan sangat penting bagi suatu kompetisi dunia saat ini. Maka segala proses yang menerapkan metodologi berbasis komputer mencakup teknik baru dalam pencarian informasi dari data disebut Data Mining (Kantardzic, 2003).

### 2.1.1 Definisi *Data Mining*

*Data Mining* adalah proses pencarian berbagai jenis model, ringkasan dan nilai dari koleksi data yang diberikan (Kantardzic, 2003). Atau menurut Tan (2004), *Data Mining* adalah proses penggalian dan analisis, dengan menggunakan peranti otomatis atau semi otomatis, dari sejumlah besar data yang bertujuan untuk menemukan bentuk yang bermanfaat. Gambar 2.1 merupakan tahapan-tahapan untuk menggali informasi tersembunyi dalam *Data Mining*.

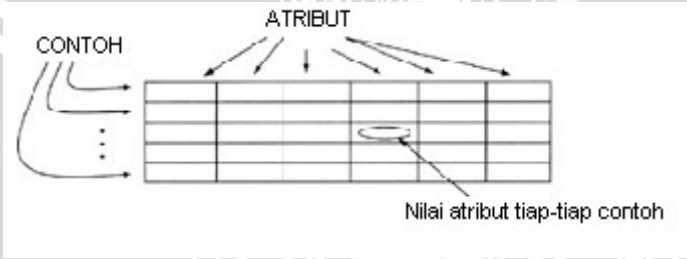



Gambar 2.1 Proses Penggalian Informasi Tersembunyi

### 2.1.2 Definisi Data Contoh (*Dataset*)

Menurut Kantardzic (2003), dalam teori kebanyakan *Data Mining* sangat diuntungkan dengan data-contoh berukuran besar. Data-contoh mempunyai potensi informasi yang berharga. Data-contoh ada yang terstruktur (mayoritas data), semi terstruktur (dokumen web, hasil laboratorium, dan lain-lain), dan tidak terstruktur (rekaman video pada supermarket). Kebanyakan metode *Data Mining* dan aplikasi komersial menggunakan data-contoh terstruktur.

Model umum dari data-contoh terstruktur untuk *Data Mining* terdiri dari banyak kasus dengan atribut yang telah terspesifikasi. Biasanya representasi dari data-contoh terstruktur untuk *Data Mining* berupa bentuk tabular, seperti pada gambar 2.2 atau hanya tabel berelasi satu (pada relasi database), dimana kolom sebagai atribut dari obyek pada tabel dan baris adalah nilai dari atribut entitas.




Gambar 2.2 Representasi Data-data Tabel pada Data Contoh6

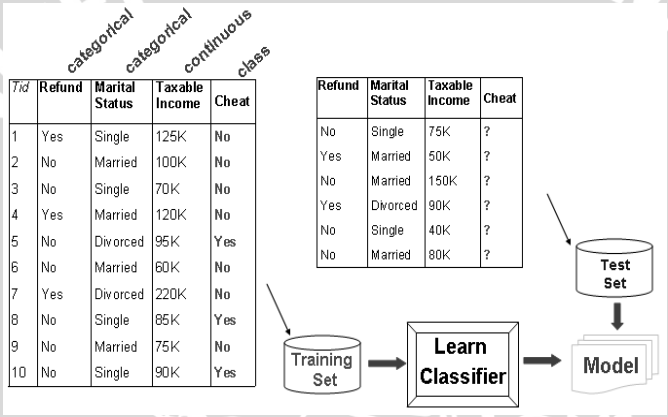
Banyak tipe berbeda dari atribut dari data-contoh terstruktur untuk *Data* Mining, yang tidak semua metode *Data Mining* bisa mengatasi hal tersebut dengan sama baiknya.

### 2.1.3 Teknik Klasifikasi Aturan

Diantara teknik *Data Mining* yaitu pengklasifikasian aturan. Klasifikasi adalah proses menemukan suatu model (garis besar struktur, ringkasan relasi banyak kasus- Kantardzic, 2003) untuk memprediksi kelas dari obyek yang belum diketahui namanya. Model didasarkan dari analisa sekumpulan data-latih (*training set)* (Han, 2006).

Dalam *Data Mining,* klasifikasi termasuk ke dalam kelompok teknik dengan metode prediksi, yakni menggunakan beberapa variabel untuk memprediksi nilai yang tidak diketahui atau nilai di masa mendatang dari variabel lain.

Pencarian informasi salah satunya ditunjukkan dengan bentuk aturan *IF-THEN*, yang memiliki keuntungan berupa level yang tinggi, representasi simbol informasi, dan mendapatkan informasi yang bisa dimengerti. Pencarian aturan bisa dievaluasi menurut beberapa kriteria, seperti nilai kelayakan prediksi, nilai keakuratan klasifikasi pada kelas yang belum diketahui, dan lain-lain (Fidelis, 1999).




Gambar 2.3 Gambaran Umum Teknik Klasifikasi

Pada gambar 2.3 ditunjukkan tentang gambaran teknik klasifikasi secara umum. Bahwa dalam data-contoh, terdapat sekumpulan baris data-latih (*training set*) yang setiap barisnya terdiri dari sekumpulan atribut, satu dari atribut merupakan atribut tujuan. Ditentukan suatu model untuk atribut tujuan sebagai suatu fungsi nilai dari atribut lain.

Umumnya, data-contoh yang diberikan dibagi ke dalam data-latih (*training set*) dan data-uji (*test set*). Data-latih digunakan untuk membentuk model dan data-uji digunakan untuk mengujinya yakni menentukan keakuratan suatu model. Dengan kata lain, algoritma *Data Mining* (pada teknik klasifikasi) diterapkan pada sekumpulan data-latih, dengan kelas yang diketahui, untuk mencari aturan sesuai relasi antara atribut prediksi dan atribut tujuan. Relasi ini kemudian digunakan untuk memprediksi kelas dari atribut tujuan yang belum diketahui (Tan, 2004).

### 2.1.4 Pengkondisian Data-Contoh

Umumnya *database* sangat rentan dengan *noisy* (data rusak, nilai *outlier*), *missing* (atribut tidak bernilai) dan data tak berkonsisten (ketidakcocokan nilai) dikarenakan ukurannya yang sangat besar (beberapa *gigabytes* bahkan lebih) dengan inputan data dari berbagai sumber. Data berkualitas rendah akan menghasilkan kualitas hasil *mining* yang rendah pula.

Ada beberapa teknik untuk pengkondisian data. *Data cleaning* diterapkan untuk mengisi nilai hilang, identifikasi dan memindahkan *noise*, serta membetulkan data yang tidak konsisten. Pendiskritan Data *(Data Discretization)* diterapkan untuk mengurangi nilai pada suatu atribut kontinu dengan membagi nilai atribut ke dalam *interval-interval*. Kemudian identitas *interval* akan menggantikan nilai sebenarnya. Teknik pengkondisian data ketika diterapkan sebelum proses *mining*, akan memperbaiki keseluruhan kualitas dari pola *mining* dan/atau waktu yang dibutuhkan untuk *mining* (Han, 2006). Teknik pengisian nilai hilang dan pendiskritan dijelaskan dalam subbab 2.1.4.1 dan 2.1.4.2.

#### 2.1.4.1 Mengisi Nilai Hilang

Menurut Kantardzic (2003), pada data yang sangat besar, kadang dijumpai kasus dimana data tidak terisi dengan lengkap (nilai hilang). Sehingga sangat mempengaruhi pemrosesan data untuk memaksimalkan kesimpulan informasi. Solusinya tidak membuang data berisi nilai hilang, tetapi dengan mendefinisikan sebuah nilai untuk mengisinya.

Klasifikasi *Bayesian* menggunakan pendekatan statistika untuk menginduksi kesimpulan masalah klasifikasi. Klasifikasi *Bayesian* mempunyai tingkat kesalahan yang kecil dibanding semua metode klasifikasi yang dikembangkan *Data Mining*. Tetapi tidak terjadi untuk semua kasus, karena ketidakakuratan pada asumsi atribut dan kondisi kelas yang bebas.

Semisal $X$ adalah data yang tidak diketahui label kelasnya. $H$ adalah hipotesanya. Data $X$ diinginkan termasuk kelas $C$. Kemudian ditentukan $P(H|X)$, peluang hipotesa $H$ berdasar pengamatan data $X$. $P(H)$ adalah peluang dari $H$ untuk semua sampel. Maka dirumuskan Klasifikasi *Bayesian* seperti pada persamaan 2.1.

$$P(H/X) = [P(X/H) \cdot P(H)] / P(X) \quad (2.1)$$

Andaikata terdapat *m* data-latih S = { $S_1$, $S_2$, ....,$S_m$ } dimana setiap $S_i$ sebagai vektor *n*-dimensi {$x_1$,$x_2$,....,$x_n$}. Nilai $x_i$ berkoresponden dengan atribut $A_1$, $A_2$, ....,$A_n$.

Terdapat pula *k* kelas $C_1$,$C_2$, ....,$C_n$ dan tiap-tiap sampel menuju kepada satu kelas. Peluang prediksi kelas data *X* dengan kondisi peluang $P(C_i|X)$ dimana $i$=1,...,$k$. Maka inilah dasar dari Klasifikasi *Naïve Bayesian*. Peluangnya dihitung dengan Klasifikasi *Bayes* yakni dengan persamaan 2.2,

$$P(C_i / X) = [P(X / C_i) \cdot P(C_i)] / P(X) \quad (2.2)$$

dimana $P(X)$ adalah nilai konstan untuk semua kelas, hanya hasil dari $P(X/C_i)\cdot P(C_i)$ yang harus dimaksimalkan. Dan peluang kelas $P(C_i)$ adalah jumlah sampel latih dari kelas $C_i|m$ (*m* adalah jumlah keseluruhan dari sampel latih). Karena komputasi dari $P(X|C_i)$ sangat komplek, khususnya data-contoh berukuran besar, persamaan Klasifikasi *Naïve Bayesian* untuk asumsi kondisi atribut sembarang menghasilkan persamaan 2.3,

$$P(X / C_i) = \prod_{t=1}^{n} P(x_t / C_i) \quad (2.3)$$

dimana $x_i$ adalah nilai atribut pada sampel *X*. Peluang $P(x_t|C_i)$ bisa diperkirakan dari data-latih.

### 2.1.4.2 Pendiskritan Nilai Kontinu

Menurut Kantardzic (2003), teknik pendiskritan bertujuan mendiskritkan nilai kontinu ke sejumlah kecil *interval-interval*, dimana masing-masing *interval* memetakan simbol diskrit. Keuntungannya adalah pendeskripsian data menjadi lebih sederhana serta mudah untuk memahami data dan hasil akhir *Data Mining*. Juga nilai diskrit lebih aplikatif dengan sistem.

*ChiMerge* adalah salah satu algoritma pendiskritan otomatis yang menganalisa kualitas *interval* untuk atribut yang diberikan dengan statistik $\chi^2$. Algoritma ini sebagai pendekatan kesamaan antara pendistribusian data pada dua *interval* yang berdekatan. Hasil uji $\chi^2$ digunakan dalam metodologi untuk menentukan dua interval yang

berdekatan. Jika hasil uji dari $\chi^2$ tidak berada dalam atribut *interval*, maka *interval-interval* tersebut digabung; sebaliknya, bila terindikasi ada perbedaan antara *interval-interval* maka tidak digabung.

*ChiMerge* terdiri dari 3 langkah untuk pendiskritan meliputi :

1. Mengurutkan data secara menaik untuk atribut yang diberikan.
2. Mendefinisikan *interval* awal sehingga masing-masing nilai atribut berada pada interval tersendiri.
3. Diulang sampai $\chi^2$ dari dua *interval* yang berdekatan kurang dari nilai *threshold*.

Setelah masing-masing *interval* digabung, hasil uji $\chi^2$ dari selisih *interval* dikalkulasi dan 2 atribut yang berdekatan dengan nilai $\chi^2$ terkecil akan ditemukan. Jika hasil kalkulasi $\chi^2$ kurang dari *threshold*, maka *interval-interval* ini digabung. Jika tidak digabung dan jumlah *interval-interval* adalah lebih besar dari definisi maksimum oleh pengguna, maka bertambah nilai *threshold*-nya.

Hasil uji $\chi^2$ dihitung pada tabel *contingency* (ditunjukkan pada tabel 2.1), dengan persamaan 2.4.

Tabel 2.1 Contoh Tabel *Contingency* untuk Data Kategori 2 x 2

| | Kelas 1 | Kelas 2 | $\sum$ |
|---|---|---|---|
| *Interval*-1 | $A_{11}$ | $A_{12}$ | $R_1$ |
| *Interval*-2 | $A_{21}$ | $A_{22}$ | $R_2$ |
| $\sum$ | $C_1$ | $C_2$ | N |

$$\chi^2 = \sum_{i=1}^{2} \sum_{j=1}^{k} (A_{ij} - E_{ij})^2 / E_{ij} \qquad (2.4)$$

dimana

k = jumlah kelas

$A_{ij}$ = jumlah data pada *interval* ke-i, kelas ke-j

$E_{ij}$ = frekuensi dari $A_{ij}$, yang dihitung dengan $(R_i \, . \, C_j)N$

$R_i$ = jumlah data pada *interval* ke-i = $\sum A_{ij}$, j = 1,...,k

$C_j$ = jumlah data pada kelas ke-j = $\sum A_{ij}$, i = 1,2

N = jumlah keseluruhan data = $\sum R_i$, i = 1,2

### 2.1.5 Menangani Atribut Berisi Sebaran Banyak Nilai

Dalam data-contoh terkadang dibutuhkan pengecekan kembali terhadap nilai label; hasil dari proses klasifikasi, agar didapatkan nilai label yang akurat. Diantara yang menyebabkan ketidakakuratan nilai label hasil klasifikasi adalah jumlah persebaran nilai pada atribut tertentu yang sangat besar. Seperti pada tabel 2.2, diberikan sebuah atribut *origin* berisi sebaran banyak jenis nilai.

Tabel 2.2 Contoh Atribut Berisi Sebaran Banyak Nilai

| Jenis Nilai | Jumlah Baris |
|---|---|
| USA | 1 |
| France | 1 |
| US | 156 |
| Europe | 46 |
| Japan | 51 |
| German | 30 |
| Korea | 39 |

Tabel 2.2 menunjukkan 7 jenis nilai dari atribut *origin*: *USA, France, US, Europe, Japan, German* dan *Korea*. Dua jenis nilai diantaranya hanya terdiri dari satu baris data pada atribut origin, yakni *USA* dan *France*. Total 2 baris data ini ikut terklasifikasi pula dengan nilai-nilai lainnya yang memiliki jumlah baris lebih banyak, sehingga berakibat pada ketidakmaksimalan untuk mengklasifikasikan atribut *origin*. Agar lebih maksimal untuk mengklasifikasi data-contoh, khususnya atribut *origin*, baris data dengan jenis nilai *USA* diberi label *US* dan baris data dengan jenis nilai *France* diberi label *Europe* (Daniel, 2005).

Agar lebih efektif dan efesien dalam klasifikasi data-contoh, dipilihlah atribut yang lebih maksimum dalam aplikasi Data Mining. Sehingga dalam tabel kasus 2.2, penghilangan atribut secara manual oleh analis, akan diterapkan untuk meningkatkan keakuratan proses klasifikasi (Kantardzic, 2003).

## 2.2 Algoritma Genetika

Algoritma Genetika dimulai dengan inisialisasi solusi baik secara acak atau terstruktur yang disebut populasi. Solusi dari suatu populasi diambil dan digunakan untuk membentuk populasi baru berikutnya. Dengan harapan, populasi baru berikutnya akan lebih baik dari populasi sebelumnya. Solusi yang terpilih untuk membentuk solusi baru (*offspring*) berdasarkan nilai *fitness*-nya, yakni makin baik nilainya, maka makin besar kesempatannya untuk diproduksi (Sumantrika, 2005). Pada subbab ini akan dijelaskan secara singkat tentang Algoritma Genetika berkaitan dengan rumusan masalah dalam skripsi.

### 2.2.1 Definisi Algoritma Genetika

Algoritma Genetika adalah suatu algoritma pencarian data dan optimasi yang bersifat umum, berbasis pada proses mekanika alamiah di mana sifat spesies sangat bergantung pada gen-gen dan susunannya. Algoritma Genetika sangat tepat digunakan untuk penyelesaian masalah optimasi yang kompleks dan sukar diselesaikan dengan menggunakan metode yang konvensional (Nurjaya, 2006).

### 2.2.2 Istilah dalam Algoritma Genetika

Istilah Algoritma Genetika pada skripsi ini yakni :

1. Satu generasi menyatakan kumpulan populasi.
2. Satu populasi menyatakan kumpulan individu.
3. Satu individu menyatakan satu kesatuan kromosom.
4. Satu kromosom menyatakan kumpulan gen.
5. Satu gen menyatakan satu atribut dan terdiri dari 3 bit.
6. Nilai *fitness* menyatakan nilai kelangsungan hidup individu.

### 2.2.3 Siklus Algoritma Genetika

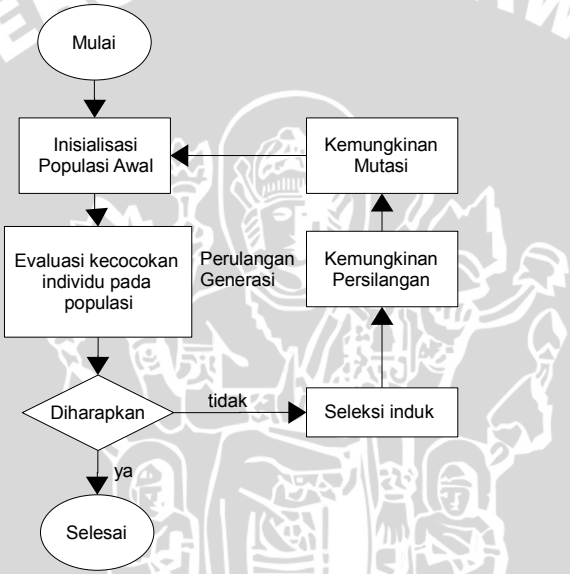
Seperti ditunjukkan pada gambar 2.4, siklus umum dari suatu Algoritma Genetika yakni :

1. Membangkitkan populasi awal secara acak atau berdasarkan kriteria tertentu. Populasi merepresentasikan solusi yang diinginkan.
2. Membentuk generasi baru dengan seleksi, persilangan, dan mutasi.
3. Mengevaluasi populasi dengan menghitung nilai kecocokan setiap kromosom dan mengevaluasinya sampai terpenuhi kriteria berhenti.

Apabila kriteria berhenti belum terpenuhi maka dibentuk lagi generasi baru dengan mengulangi langkah 2.

Beberapa kriteria berhenti yang sering digunakan antara lain:

- Berhenti pada generasi tertentu.
- Berhenti setelah dalam beberapa generasi berturut-turut didapatkan nilai *fitness* tertinggi tidak berubah.
- Berhenti bila dalam *N* generasi berikutnya tidak didapatkan nilai kecocokan yang lebih tinggi (Nurjaya, 2006).




Gambar 2.4 Proses Umum Algoritma Genetika

### 2.2.4 Algoritma Genetika pada Skripsi

Dalam subbab ini akan dijelaskan beberapa pendalaman teori tentang komponen-komponen Algoritma Genetika yang berkaitan dengan penyelesaian rumusan masalah pada skripsi ini yakni untuk memperoleh pendekatan solusi optimal berupa aturan klasifikasi *mining* yang layak dan akurat.

### 2.2.4.1 Pengkodean *Binary*

Sebagai langkah awal dalam proses genetika adalah mengkodekan nilai-nilai atribut dari data-contoh yang telah dikondisikan datanya ke dalam bit-bit; berfungsi untuk memaksimalkan proses genetika.

Ada beberapa macam pengkodean kromosom dalam Algoritma Genetika, diantaranya Pengkodean *Binary*. Pengkodean *Binary* adalah pengkodean yang sering digunakan dalam Algoritma Genetika, dimana setiap kromosom terdiri dari bit 0 dan 1. Contoh masalah dari Pengkodean *Binary s*eperti masalah *knapsack* (ransel). Yakni ada benda yang diberikan nilai dan panjang. *Knapsack* memberikan kesempatan untuk menyeleksi benda dan memaksimumkan nilai dari benda tersebut. Pengkodeannya, setiap masalah dinyatakan sebagai bit untuk menyesuaikannya dengan *knapsack* (Nurjaya, 2006). Gambar 2.5 menunjukkan tabel hasil Pengkodean *Binary*.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Kromosom A | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| Kromosom B | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |

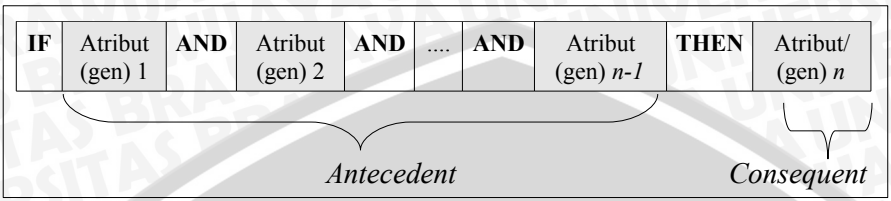
Gambar 2.5 Contoh Pengkodean *Binary*

### 2.2.4.2 Representasi Kromosom

Setelah nilai-nilai atribut dari data-contoh dikodekan, maka diperlukan upaya merepresentasikan solusi masalah dalam bentuk satu kesatuan kromosom. Kesesuaian representasi kromosom dengan kasus Algoritma Genetika inilah yang nantinya akan sangat mempengaruhi kualitas solusi Algoritma Genetika.

Menurut Fidelis (1999), kromosom dibagi menjadi *n* gen, dimana tiap-tiap gen sama dengan satu atribut, dan *n* adalah jumlah atribut prediksi pada data yang digali. Tergantung posisinya, seperti gen ke-1 menunjukkan atribut ke-1, gen ke-2 menunjukkan atribut ke-2, dst.

Sehingga susunan kromosom yang terdiri dari *n* gen ini sama artinya dengan sebuah aturan dengan *n* atribut yakni untuk atribut *antecedent* bagian *IF* dan atribut *consequent* bagian *THEN*, seperti ditunjukkan pada gambar 2.6.




Gambar 2.6 Struktur *IF...THEN*

Tiap-tiap gen ke-i, *i=1..n*, dibagi ke dalam 3 bagian : *flag* (*Fi*), *operator* (*Oi*) dan *value* (*Vi*), seperti tampak pada gambar 2.7.

| Gen ke-1 | | | ... | ... | Gen ke-*n* | | |
|---|---|---|---|---|---|---|---|
| $F_1$ | $O_1$ | $V_1$ | ... | ... | $F_n$ | $O_n$ | $V_n$ |

Gambar 2.7 Representasi Kromosom

Bagian *Flag* (*Fi*) berisi variabel bernilai 0 dan 1. Variabel ini mengindikasikan ada tidaknya atribut yang ditunjukkan pada aturan. Bila bit bernilai 0, maka atribut tidak terdapat pada aturan dan bila bit bernilai 1, maka atribut terdapat pada aturan.

Bagian *Operator* (*Oi*) berisi variabel yang menunjukkan relasi operator pada atribut ke-*i*. Atribut *antecedent* telah dikondisikan bernilai diskrit, maka bagian *operator* hanya memuat operator "<>" dan "=". Bila bit bernilai 0, artinya memuat operator "<>" dan bila bit bernilai 1, maka artinya memuat operator "=".

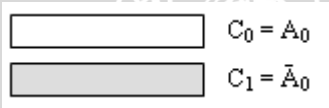
Bagian *Value* (*Vi*) berisi satu nilai dari *n* nilai pada atribut ke-*i*. Nilai *Vi* ini dikodekan menjadi string *binary* .

### 2.2.4.3 Pembangkitan Populasi Awal Terstruktur

Untuk mencari alternatif-alternatif solusi, Algoritma Genetika membangkitkan populasi awal; yang berisikan banyak alternatif solusi; untuk dijadikan solusi awal dari kasus yang diselesaikan dan dijadikan sebagai populasi induk untuk melahirkan populasi-populasi turunan berikutnya yang diharapkan memiliki nilai *fitness* lebih baik dari nilai *fitness* populasi induk.

Semua solusi Algoritma Genetika dibentuk dari pembangkitan populasi awal secara acak. Pembangkitan populasi awal secara acak memiliki banyak kekurangan, yakni bisa membentuk kromosom pada daerah yang berdekatan dan jauh dari daerah solusi; atau pencarian solusi hanya berkisar pada daerah solusi lokal. Solusi lokal pada teknik klasifikasi adalah aturan yang tidak disetujui secara umum. Sehingga pada skripsi ini, pembangkitan populasi awal akan dilakukan secara sistematis yang terinspirasi dari metode populasi terstruktur (Korkut, 2004).

Menurut (Korkut, 2004), Pembangkitan populasi awal secara terstruktur ini diterapkan untuk Pengkodean kromosom secara *Binary*. Pertama, dua kromosom dibangkitkan dengan panjang kromosom yang sama. Seperti pada gambar 2.8, terdapat kromosom $C_0$, $C_1$ dan $A_i$ yang menunjukkan kumpulan gen dengan panjang yang sama.



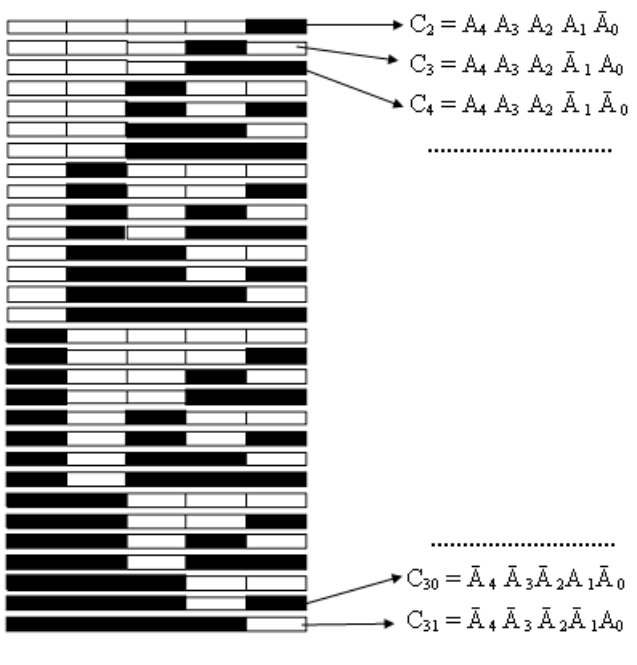

Gambar 2.8 Kromosom Awal

$C_1$ berfungsi sebagai pelengkap dari $C_0$. $C_1$ akan dijadikan acuan dalam pembentukan kromosom berikutnya. Pada pembangkitan populasi awal secara terstruktur, kromosom-kromosom baru ($2^r$-*1*) akan dibangkitkan dari sebuah kromosom yang dibentuk secara acak. Jika ukuran populasi telah ditetapkan, maka kromosom baru ($2^r$-*2)* akan dibangkitkan secara sistematik.

$$2^r \leq |P| \tag{2.5}$$

Pada persamaan 2.5, $|P|$ adalah ukuran populasi yang ditetapkan, $r$ adalah panjang maksimal model yang berubah-ubah. Kemudian $r$ berkurang satu dan kromosom baru ($2^r - 2$) dibangkitkan mengikuti nilai $r$ yang baru. Proses ini berlanjut sampai menyisakan dua buah kromosom. Pada kasus ini, hanya beberapa yang terpilih menjadi populasi awal untuk menyesuaikan ukuran populasi. Untuk lebih memperjelas tahapan pembangkitan populasi awal secara terstruktur ini akan diberikan sebuah contoh.

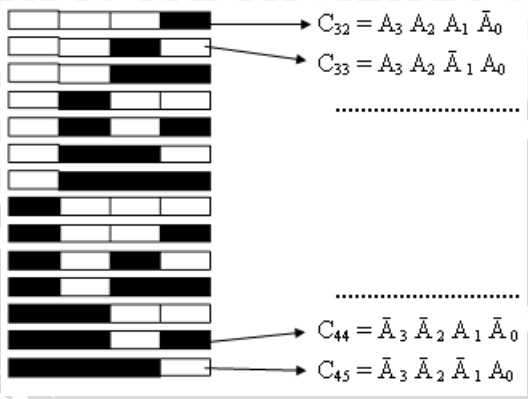
Misal ukuran populasi senilai 54. Sebuah kromosom acak akan dibangkitkan seperti gambar 2.8 sebelumnya. Kemudian mengikuti rumusan pada persamaan 2.5, didapatkan *r* senilai 5 ($2^5$<=52). Sehingga $C_0$ dibagi ke dalam 5 bagian. Ada 30 ($2^5$–2) kromosom baru yang nantinya dibangkitkan dari $C_0$, mengikuti 30 model kromosom baru. Area bayang-bayang pada 30 model kromosom tersebut menunjukkan pelengkap gen dari $C_0$. Proses ini ditunjukkan pada gambar 2.9.



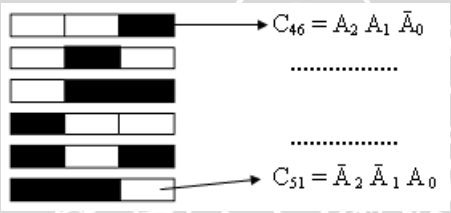

Gambar 2.9 Kromosom Baru dengan $r = 5$

Kemudian *r* berkurang satu sehingga nilainya menjadi 4. Kromosom baru $2^4$–2=14 baru akan dibangkitkan mengikuti $C_0$. Proses ini ditunjukkan pada gambar 2.10.

Langkah berikutnya *r*=3 dan 6 kromosom baru akan dibangkitkan seperti pada gambar 2.11, dan langkah berikutnya $r = 2$ dan 2 kromosom baru akan dibangkitkan seperti pada gambar 2.12. Ternyata, 2 kromosom awal yang dibangkitkan, $C_0$ dan $C_1$, sama dengan situasi dimana $r = 1$.




Gambar 2.10 Kromosom Baru dengan $r = 4$




Gambar 2.11 Kromosom Baru dengan $r = 3$


Gambar 2.12 Kromosom Baru dengan $r = 2$

#### 2.2.4.4 Operator Genetika

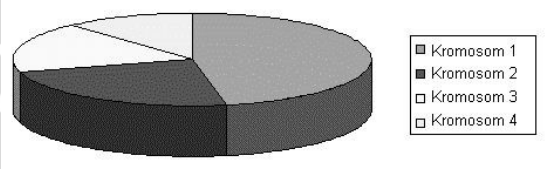
Operator genetika yang efesien yaitu yang mampu menghasilkan kromosom berkualitas tinggi pada tiap-tiap generasi. Operator-operator genetika yang dipandang efesien diterapkan pada skripsi ini yakni :

##### 2.2.4.4.1 Seleksi Roda *Roullette* (*Roullette Wheel Selection*)

Individu-individu perlu diseleksi dari populasi induk untuk mempertahankan individu terbaik dan menyisihkan individu lemah. Serta untuk memilih kromosom induk, digunakan untuk membangkitkan generasi berikutnya. Beberapa metode seleksi diantaranya Roda

*Roullette,* Seleksi Turnamen, Seleksi Peringkat, Seleksi Kondisi Tetap, Seleksi *Elitism,* dan lain-lain. (Nurjaya, 2006).

Pada Seleksi Roda *Roullette*, kromosom induk dipilih berdasar nilai *fitness*-nya. Kromosom induk dengan nilai *fitness* terbaik memiliki banyak kemungkinan untuk terpilih. Seperti pada permainan Roda *Roullette*, dimana semua kromosom ditempatkan pada populasi. Masing-masing kromosom memiliki tempat yang besar sesuai nilai *fitness*-nya, seperti tampak pada gambar 2.13.




Gambar 2.13 Seleksi Roda *Roullette*

Kemudian kelereng dilemparkan ke dalam mangkuk populasi, sehingga terpilihlah sebuah kromosom. Kromosom dengan nilai *fitness* yang besar tentu akan terpilih berulang kali (Obitko, 2008).

Seleksi Roda *Roullette* bisa disimulasikan dengan langkah :

1. Dihitung total nilai *fitness* semua kromosom pada populasi yang telah terurut menurun (*descending*).
2. Dihitung probabilitas dari nilai *fitness* semua kromosom. Yaitu *probabilitas kromosom ke-i = fitness ke-i / total fitness*
3. Dihitung probabilitas kumulatifnya tiap-tiap kromosom, yaitu *probabilitas kumulatif ke-n = total probabilitas ke-1 sampai ke-n.*
4. Dipilih angka pecahan acak dari 0 sampai 1. Angka acak yang didapat tersebut dicarikan *probabilitas kumulatif* yang sesuai (angka pecahan acak $\leq$ *probabilitas kumulatif*). Sehingga kromosom induk yang dipakai untuk membentuk kromosom turunan yakni kromosom dengan *probabilitas kumulatif* tersebut.

Kromosom dengan probabilitas terbesar memiliki banyak peluang untuk dipilih menjadi kromosom induk yang nantinya akan menurunkan sifatnya kepada generasi turunan berikutnya.

Untuk lebih memahami tentang cara kerja Seleksi Roda *Roullette*, akan diberikan contoh pada bab 3 selanjutnya.

#### 2.2.4.4.2 Persilangan Terstruktur *(Uniform Crossover)*

Persilangan merupakan proses pertukaran (bit,gen,atau kromosom -pen) antar 2 kromosom sehingga 2 kromosom tersebut akan bercampur membentuk kromosom baru atau *offspring* (Nurjaya, 2006).

Ada beberapa jenis operator persilangan dari Algoritma Genetika. Tetapi beberapa persilangan diantaranya seperti Persilangan Satu Titik dan Persilangan Dua Titik memiliki kelemahan seperti tidak bisa mengkombinasikan semua kemungkinan skema kromosom. Maka diterapkan Persilangan Terstruktur yang mampu membangkitkan semua skema kromosom induk untuk dijadikan calon kromosom baru (Melanie, 1999).

Pada Persilangan Terstruktur, dari tiap-tiap generasi, 4 atau lebih kromosom baru berkualitas tinggi dibangkitkan dari dua kromosom induk terbaik yang berbeda, dengan pencegahan sejak awal nilai yang seragam *(convergen)* pada solusi lokal. Untuk lebih mudahnya, akan diberikan sebuah contoh Persilangan Terstruktur.

Misal $C_0$ dan $C_1$ adalah dua kromosom induk terbaik dengan panjang bit senilai 20 bit seperti yang ditunjukkan pada gambar 2.14. Pada gambar 2.14 ditemukan 6 bit yang berbeda (disimbolkan bintang). Sehingga akan dibangkitkan secara acak 6-bit *array* yang tersusun dari 0 atau 1. Sebagai contoh terbentuk 6-bit *array* pertama 001101. Untuk *r* =2, 6-bit *array* kedua, ketiga, dan keempat yang terbentuk dari 6-bit *array* pertama adalah 001010, 110101, dan 110010. Keempat 6-bit *array* yang terbentuk ini didistribusikan ke posisi bit yang berbeda pada kromosom baru. Empat kromosom baru yang telah terbentuk hasil Persilangan Terstruktur seperti ditunjukkan pada gambar 2.15.

Persilangan Terstruktur ini dikerjakan di atas *flag* dari gen-gen. Sehingga faktor inilah yang menyebabkan Persilangan Terstruktur menggunakan Pengkodean *Binary*. Persilangan Terstruktur ini bisa juga menggunakan semua tipe pengkodean seperti Pengkodean *Floating Point-Based* dan Pengkodean *String-Based* dengan sedikit modifikasi, akan tetapi pada skripsi ini, pengkodean tersebut tidak dibahas.

| | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $C_0$ | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| $C_1$ | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| Y | 0 | 1 | 0 | 1 | * | 0 | * | 1 | 1 | 1 | * | 1 | 0 | 1 | 0 | 0 | * | * | * | 1 |

Gambar 2.14 Kromosom Terbaik $C_0$, $C_1$ dan Kromosom Berbeda Y

| | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| B | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| C | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| D | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |

Gambar 2.15 Kromosom-kromosom Baru

Parameter dalam Algoritma Genetika dimaksudkan untuk memberikan persentase kemungkinan dari dilakukannya suatu operasi genetika. Parameter probabilitas persilangan merupakan persentase dari seringnya dilakukan persilangan pada kromosom induk untuk melahirkan kromosom turunan. Makin besar nilai persentase dari kemungkinan persilangan maka makin tinggi tingkat kemiripan kromosom turunan dengan kromosom induk (Nurjaya, 2006).

#### 2.2.4.4.3 Mutasi Bit, Gen, atau Kromosom (*Mutation*)

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 (Obitko, 2008).

Penerapan mutasi dalam skripsi ini dilakukan secara acak pada ketiga tingkatan mutasi. Yakni mutasi bit, mutasi gen, atau mutasi kromosom, dengan definisi dari masing-masing mutasi yakni :.

1. Mutasi Bit
   Mutasi membalikan satu bit yang termutasi.

   Misal : 101 | 01**0**1 | 001 (kromosom awal) —Mutasi→ 101 | 01**1**1 | 001 (kromosom hasil)

2. Mutasi Gen
   Mutasi membalikkan semua bit yang menyusun gen termutasi.

   Misal : 101 | **0101** | 001 (kromosom awal) —Mutasi→ 101 | **1010** | 001 (kromosom hasil)

3. Mutasi Kromosom
   Mutasi membalikkan semua bit yang menyusun kromosom.

   Misal : **101** | **0101** | **001** (kromosom awal) —Mutasi→ **010** | **1010** | **110** (kromosom hasil)

(Sumantrika, 2005).

Sama halnya dengan operasi persilangan, terdapat pula kemungkinan dilakukannya mutasi. Makin besar nilai persentase dari kemungkinan mutasi maka makin tinggi tingkat kemiripan kromosom turunan dengan kromosom induk (Nurjaya, 2006).

Dalam semua kasus, sangat dianjurkan meningkatkan parameter kemungkinan mutasi. Angka mutasi yang terlalu kecil mengakibatkan penyimpan ekstrim genetik (sangat jarang terjadi pada alam) atau konvergensi prematur pada solusi lokal. Angka mutasi yang terlalu besar berkemungkinan menghilangnya solusi yang baik (Obitko, 2008).

### 2.2.4.5 Perhitungan Nilai Kelayakan dan Keakuratan Aturan

Dari perhitungan nilai kelayakan *antecedent-consequent* dan keakuratan terhadap aturan, akan menghasilkan nilai *fitness* Algoritma Genetika (sesuai rumusan masalah dalam skripsi) dari tiap-tiap kromosom (representasi dari aturan). Dengan kata lain, perhitungan nilai *fitness* dilakukan setelah 2 tahapan perhitungan sebelumnya yakni tahapan pertama untuk menghitung nilai kelayakan aturan dan tahapan kedua untuk menghitung nilai keakuratan aturan.

Tahapan perhitungan pertama untuk mendapatkan nilai *fitness* yaitu perhitungan untuk memprediksi nilai kelayakan aturan, yang dibagi dua yakni untuk memprediksi kelayakan *antecedent* dan untuk memprediksi kelayakan *consequent*. Nilai kelayakan *antecedent* (*AI*) dihitung dengan persamaan 2.6,

$$AI = 1 - \left[ \frac{\sum_{i=1}^{n} InfoGain(A_i)}{\frac{n}{\log_2(|\, dom(G_k\,|)\,)}} \right] \tag{2.6}$$

dimana *n* adalah jumlah atribut pada *antecedent*; *(| dom($G_k$ |)* adalah jumlah kemungkinan nilai dari atribut tujuan $G_k$ pada *consequent*; operasi *log* digunakan untuk menormalisasikan nilai *AI* yang menghasilkan nilai antara 0-1. Sedang untuk menghitung *InfoGain* dengan menggunakan persamaan 2.7, menghitung *Info($G_k$)* dengan persamaan 2.8, dan menghitung *Info(Gk|Ai)* dengan persamaan 2.9,

$$InfoGain(A_i) = Info(G_k)Info(G_k \mid A_i) \quad (2.7)$$

$$Info(G_k) = -\sum_{j=1}^{m_k} \left(\Pr(V_{kj})\log_2\left(\Pr(V_{kj})\right)\right) \quad (2.8)$$

$$Info(G_k \mid A_i) = \sum_{z=1}^{n_i} \left( \Pr(V_{iz}) \left( -\sum_{j=1}^{m_k} \Pr(V_{kj} \mid V_{iz}) \log_2\left(\Pr(V_{kj} \mid V_{iz})\right) \right) \right) \quad (2.9)$$

dimana $m_k$ adalah jumlah kemungkinan nilai dari atribut tujuan *Gk*, $n_i$ adalah jumlah kemungkinan nilai atribut $A_i$, *Pr(X)* adalah kemungkinan dari *X* dan *Pr(X|Y)* adalah kemungkinan dari *X* terhadap *Y*. Makin tinggi nilai *AI* yang didapat, maka makin layak *antecedent* dari aturan tersebut.

Perhitungan nilai keakuratan *consequent* berasosiasi dengan kromosom. Makin besar frekuensi relatif nilai yang diprediksi dari *consequent* pada data-latih, akan berkurang nilai *fitness*-nya. Maka jarang dihasilkan sebuah aturan yang memprediksi dengan benar nilai atribut tujuan. Oleh karenanya, untuk menghitung nilai kelayakan dari *consequent (CI)* menggunakan persamaan 2.10,

$$CI = \left(1 - \Pr(G_{kl})\right)^{1/\beta} \quad (2.10)$$

dimana $Pr(G_k)$ adalah frekuensi relatif pada atribut tujuan $G_{kl}$ dan $\beta$ adalah parameter yang didefinisikan oleh user, bernilai 2. Dari perhitungan *CI* ini menghasilkan nilai kelayakan *consequent* dari aturan, yang mana makin tinggi nilai *CI* yang didapat, maka makin layak *consequent* dari aturan tersebut.

Tahapan perhitungan yang kedua untuk mendapatkan nilai *fitness*, yaitu perhitungan untuk memprediksi keakuratan *(Predictive Accuracy)* dari aturan. Yakni menggunakan persamaan 2.11,

$$PA = \left(X - 1/2\right)/Y \quad (2.11)$$

*X* adalah jumlah data yang memenuhi *antecedent* dan *consequent*; *Y* adalah jumlah data yang memenuhi *antecedent* dari aturan. Operasi ½ digunakan untuk mengatasi aturan yang melingkupi beberapa data-latih. Makin tinggi nilai *PA*, maka makin tinggi tingkat prediksi keakuratan dari aturan tersebut.

Akhirnya akan didapat fungsi perhitungan nilai *fitness* dari 2 tahapan perhitungan sebelumnya (perhitungan nilai kelayakan aturan dan perhitungan nilai keakuratan aturan), seperti ditunjukkan pada persamaan 2.12.

$$Fitness = \frac{w_1 \frac{AI + CI}{2} + w_2 \cdot PA}{w_1 + w_2} \quad (2.12)$$

dimana $w_1$ dan $w_2$ didefinisikan oleh pengguna sebagai nilai bobot (*weight*) yakni secara berurutan 1 dan 2 (Alatas, 2002). Dari perhitungan nilai *fitness* ini, akan didapatkan parameter penilaian kelayakan dan keakuratan dari suatu aturan. Makin tinggi nilai *fitness* dari aturan (representasi dari kromosom); yaitu makin mendekati nilai maksimal *fitness*=1, maka makin tinggi nilai kelayakan dan keakuratan dari aturan tersebut.

### 2.2.4.6 Perhitungan *Probability Correct (PC)*

Menurut Sumantrika (2005), *Probability Correct* (*PC*) merupakan parameter untuk mengukur perfomansi aturan klasifikasi yang telah dihasilkan oleh sistem dalam klasifikasi data-contoh berdasarkan atribut yang terpilih sebagai aturan IF THEN. *Probability Correct* dinyatakan sebagai persentase tingkat kebenaran aturan terhadap data-uji yang digunakan dalam memprediksi kelasnya.

Persamaan untuk menghitung nilai dari *Probability Correct* (*PC*) yakni pada persamaan 2.13.

$$PC = \frac{nTrue \times 100}{nDataTest} \quad (2.13)$$

dimana *nTrue* yakni jumlah baris dari data-uji yang secara tepat memenuhi aturan klasifikasi dan *nDataTest* yakni jumlah baris dari data-uji.

Dari nilai *PC* ini akan diketahui tingkat kebenaran aturan terhadap data-uji. Yang artinya, makin tinggi nilai *PC*, maka makin akurat aturan yang dihasilkan dari data-latih terhadap data-uji.

# BAB III
# METODOLOGI DAN PERANCANGAN

Pada bab ini akan dibahas mengenai metode dan perancangan dalam penyelesaian rumusan masalah dalam skripsi ini. Langkah-langkah yang akan dilakukan yakni :

1. Menganalisa kebutuhan sistem yang hendak dibangun.
2. Merancang proses kerja sistem sesuai analisa kebutuhan sistem.
3. Merancang antarmuka sistem perangkat lunak.
4. Menguji coba sistem dan mengevaluasi aturan yang dihasilkan.

## 3.1 Analisa Umum

Klasifikasi termasuk teknik *Data Mining* yang paling sering diaplikasikan, yang bertujuan untuk membuat model klasifikasi dari sebuah database. Pada subbab ini akan dijelaskan deskripsi umum data-contoh dan sistem klasifikasi aturan *mining* dengan Algoritma Genetika.

### 3.1.1 Data Contoh (*Dataset*)

Data-contoh yang akan diujicobakan dalam skripsi ini yakni *Crx* yang di dalamnya terdapat nilai hilang dan nilai kontinu. Data-contoh ini didapat dari data-contoh skripsi Sumantrika, yang beliau download dari http://www.ics.uci.edu/~mlearn/MLRepository.html, situs yang disediakan oleh UCI (*University of California at Irvine*). Data-contoh *Crx* ini terdiri dari 16 atribut termasuk atribut *class;* diantaranya ada 6 atribut kontinu, kemudian 690 baris data (*instances*), 67 nilai yang hilang (*miss value*), dan 2 atribut berisi sebaran banyak nilai.

Data-contoh *Crx* ini dipilih karena data-contoh ini telah memenuhi semua kriteria data-contoh yang dibutuhkan untuk menguji coba keoptimalan sistem yang telah dibangun dalam skripsi ini. Pada data-contoh *Crx* telah tercakup di dalamnya nilai hilang, atribut bertipe kontinu, dan jumlah baris data yang tidak terlampau banyak sehingga memaksimalkan dalam pengujian sistem sampai tahapan penganalisaan hasil.

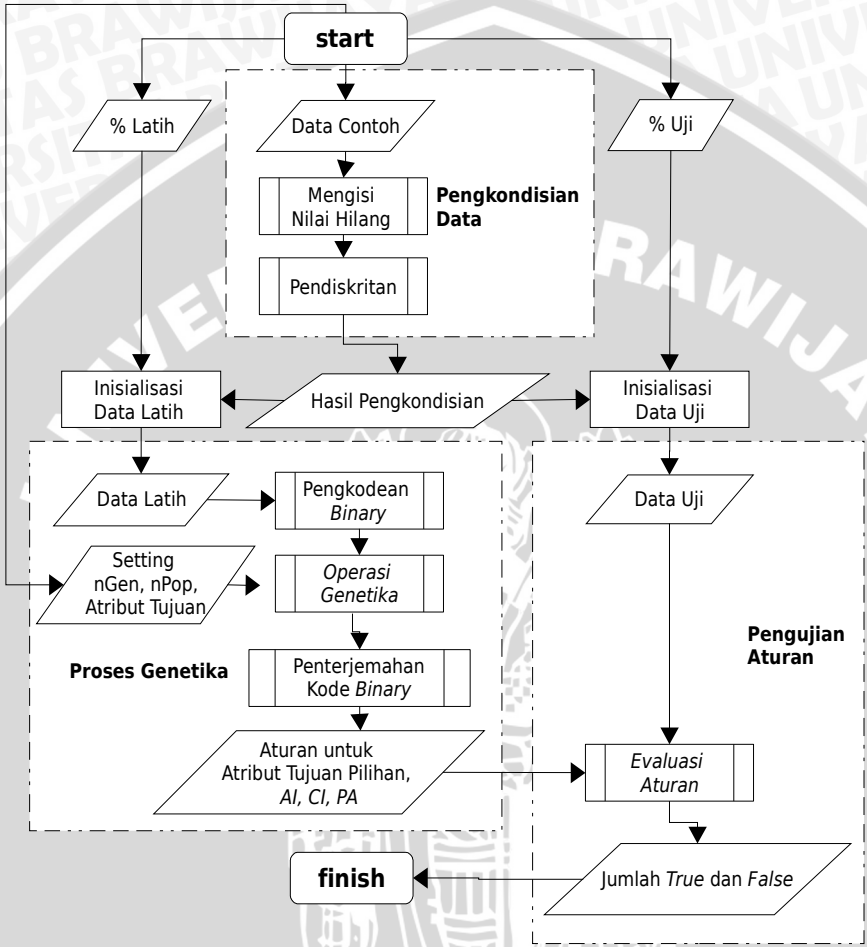
Data-contoh ini akan digunakan sebagai data-latih (*training sets*) yang berfungsi untuk membentuk model aturan klasifikasi dan digunakan sebagai data-uji (*test sets*) yang berfungsi untuk menguji aturan klasifikasi tersebut.

### 3.1.2 Deskripsi Umum Sistem

Sistem klasifikasi aturan *mining* dengan Algoritma Genetika pada skripsi ini lebih ditekankan pada penilaian kelayakan dan keakuratan aturan untuk suatu atribut tujuan berdasarkan parameter-parameter dari fungsi perhitungan nilai *fitness* yakni nilai kelayakan *antecedent, consequent,* dan nilai *predictive accuracy*. Serta berdasarkan persentase keakuratan aturan bentukan data-latih terhadap data-uji.

Langkah-langkah proses sistem untuk menyelesaikan rumusan masalah pada skripsi ini beserta diagram alur prosesnya (gambar 3.1) yakni :

1. Sistem dijalankan dan ditentukan parameter-parameternya oleh pengguna seperti persentase data-latih dan persentase data-uji terhadap data-contoh, jumlah generasi, jumlah populasi, serta sebuah atribut tujuan. Kemudian sistem dijalankan lagi oleh pengguna dengan menekan sebuah tombol *Run* untuk memasuki tahapan pengkondisian data-contoh.
2. Sistem mengkondisikan data-contoh yaitu dengan mengisi nilai hilang dan mendiskritkan nilai kontinu apabila ditemukan.
3. Sistem memproses data-contoh hasil pengkondisian untuk inisialisasi data-latih senilai persentase data-latih dan untuk inisialisasi data-uji senilai persentase data-uji.
4. Sistem mengkodekan data-latih ke dalam *binary* untuk menyesuaikan masukan yang dibutuhkan operasi genetika.
5. Sistem menjalankan operasi genetika terhadap data-latih yang telah dikodekan ke dalam *binary*, dengan parameter-parameter yang telah ditentukan sebelumnya oleh pengguna yakni jumlah generasi, jumlah populasi, dan atribut tujuan. Proses genetika lebih detilnya telah dijelaskan pada bab 2 sebelumnya.
6. Sistem menguraikan kembali data hasil operasi genetika berupa *binary* ke dalam string.
7. Sistem menerjemahkan string menjadi sebuah aturan yang bisa dipahami oleh pengguna dengan struktur *IF (antecedent)* disertai nilai kelayakan dan keakuratannya, yaitu *AI, CI, PA*.
8. Sistem menguji aturan yang telah terbentuk dengan data-uji senilai persentase data-uji, yang menghasilkan jumlah benar dan salah aturan terhadap data-uji.
9. Proses sistem telah selesai.




Gambar 3.1 Diagram Umum Proses Sistem

### 3.1.3 Batasan Sistem

Diantara batasan-batasan dalam sistem yang dibangun yakni :

1. *Database* hanya digunakan untuk menyimpan data pada sebuah tabel sehingga tidak ada tabel yang saling berelasi.

2. Data-contoh yang digunakan dalam pengujian sistem yakni *Crx* dengan atribut tujuan bertipe diskrit (tanpa nilai hilang) yaitu *A9* berjenis nilai 't' sebanyak 361 baris data dan jenis nilai 'f' sebanyak 329 baris data.
3. Parameter jumlah generasi dibatasi maksimal 30 generasi dan proses pembentukan generasi bisa berakhir sebelum mencapai generasi ke-30 apabila tidak ada perubahan nilai *fitness* pada generasi-generasi berikutnya.
4. Aturan yang dihasilkan dari tiap-tiap jenis nilai pada atribut tujuan *A9* dibatasi *n* aturan, dimana *n* kurang dari sama dengan jumlah populasi.

## 3.2 Perancangan Proses Sistem

Dalam subbab ini akan dijelaskan rancangan tahapan-tahapan yang akan diproses oleh sistem mulai dari struktur data, tahapan pengkondisian data sampai tahapan pengimplementasian Algoritma Genetika sesuai rumusan masalah dalam skripsi.

### 3.2.1 Rancangan Struktur Data

Rancangan struktur data yang akan diterapkan dalam sistem pada skripsi ini dibagi menjadi rancangan untuk *record* data (lihat tabel 3.1), rancangan konstanta (lihat tabel 3.2), rancangan pembentukan tipe data baru (lihat tabel 3.3), dan rancangan variabel (lihat tabel 3.4).

#### 3.2.1.1 Rancangan *Record* Data pada Sistem

Tabel 3.1 Tabel Rancangan *Record* Data

| Nama | Variabel | Tipe |
|---|---|---|
| TClass | value | single |
| | class | string |
| TAttrib | name | string |
| | nDist | byte |
| TBit | value, bit | string |

| | | |
|---|---|---|
| | Oper, flag | char |
| TChrom | gens | array [0..30] of Tbit |
| | bit, ant | string |
| | fit | single |
| TRule | rule, consequent | string |
| | fit | single |
| TInterval | d1,d2 | Single |
| TIntrvl | value | Single |
| | EndPos | Word |
| | intv | array[0..1] of TInterval |

a. Untuk *TClass*, variabel *value* untuk menyimpan banyaknya nilai kelas dari data-contoh, *class* untuk menyimpan nama kelasnya.
b. Untuk *TAttrib*, variabel *name* untuk menyimpan nama atribut, *nDist* untuk menyimpan banyaknya nilai unik dari atribut tersebut.
c. Untuk *TBit*, variabel *value* untuk menyimpan nilai aslinya, *bit* untuk menyimpan data bit, *oper* untuk menyimpan bit *operator*, *flag* untuk menyimpan bit *flag*.
d. Untuk *TChrom*, variabel *gens* bertipe *array* dari TBit sepanjang 30 untuk menyimpan data gen pada sebuah kromosom, *bit* untuk menyimpan data bit keseluruhan dari kromosom, *ant* untuk menyimpan *antecedent*, *fit* untuk menyimpan nilai *fitness*
e. Untuk *TRule*, variabel *rule* untuk menyimpan data *rule*, *consequent* untuk menyimpan data *consequent*, *fit* untuk menyimpan nilai *fitness*-nya.
f. Untuk *TInterval*, variabel *d1* dan *d2* untuk menyimpan nilai batas atas dan batas bawah interval
g. Untuk *TIntrvl*, variabel *value* untuk menyimpan nilai perhitungan akhir dari teknik pendiskritan, *EndPos* untuk menyimpan nilai batas terakhir interval yang telah dihitung dalam teknik pendiskritan, *intv* bertipe *array* dari *Tinterval* untuk menyimpan batasan interval.

#### 3.2.1.2 Rancangan Konstanta pada Sistem

Tabel 3.2 Tabel Rancangan Konstanta Data

| Nama | Nilai |
|---|---|
| nDistinct | 700 |
| nAttrb | 20 |
| nIntrvl | 5000 |
| MutRate | 0.9 |
| w1 | 1 |
| w2 | 2 |
| beta | 2 |
| ThresHold | 5 |

a. *nDistinct* untuk menentukan panjang alokasi memori penyimpan nilai unik dari suatu atribut.
b. *nAttrb* untuk menentukan panjang alokasi memori penyimpan atribut-atribut.
c. *nIntrvl* untuk menentukan panjang alokasi memori untuk penyimpan interval-interval yang terbentuk dari pendiskritan.
d. *MutRate* untuk inisialisasi persentase dilakukannya mutasi dari operasi genetika.
e. *w1* untuk inisialisasi *w1* dari fungsi perhitungan nilai *fitness*.
f. *w2* untuk inisialisasi *w2* dari fungsi perhitungan nilai *fitness*.
g. *beta* untuk inisialisasi beta dari fungsi perhitungan nilai *fitness*.
h. *ThresHold* untuk inisialisasi *ThresHold* dari teknik pendiskritan.

#### 3.2.1.3 Rancangan Tipe Data Baru pada Sistem

Tabel 3.3 Tabel Rancangan Tipe Data Baru

| Nama | Tipe |
|---|---|
| TArClass | array [0..nRecrd] of TClass; |
| TArIntrvl | array [0..nIntrvl] of TIntrvl; |

| | |
|---|---|
| TArAttrib | array [0..nAttrb] of TAttrib; |
| TArFlt | array [0..nRecrd] of single; |
| TArArInt | array [0..2,0..10] of word; |
| TArInt | array [0..10] of word; |
| TArChrom | array [0..100] of TChrom; |
| TArRule | array [0..1000] of TRule; |

a. Untuk *TArClass*, yakni tipe baru berupa *array* sepanjang *nRecrd* bertipe *TClass*.
b. Untuk *TArIntrvl*, yakni tipe baru berupa *array* sepanjang *nIntrvl* bertipe *TIntrvl*.
c. Untuk *TArAttrib*, yakni tipe baru berupa *array* sepanjang *nAttrb* bertipe *TAttrib*.
d. Untuk *TArFlt*, yakni tipe baru berupa *array* sepanjang *nRecrd* bertipe *single*.
e. Untuk *TArArInt*, yakni tipe baru berupa *array* dimensi 2 sepanjang 2 berkedalaman 10 bertipe *word*.
f. Untuk *TArInt*, yakni tipe baru *array* sepanjang 10 bertipe *word*.
g. Untuk *TArChrom*, yakni tipe baru berupa *array* sepanjang 100 bertipe *TChrom*.
h. Untuk *TArRule*, yakni tipe baru berupa *array* sepanjang 1000 bertipe *TRule*.

#### 3.2.1.4 Rancangan Variabel pada Sistem

Tabel 3.4 Tabel Rancangan Variabel

| Nama | Tipe |
|---|---|
| ContAttrib, DiscAttrib | array [0..20] of byte; |
| nRecord | Word |
| Attrib | TArAttrib |
| nAttrib | Byte |
| Distinct | array[0..nAttrb,0..nDistinct] of Tbit; |

| | |
|---|---|
| DataBase | array [0..nAttrb,0..2000] of String; |
| nRecTrain, nRecTest, nRules | Word |
| nGoalDistinct, IDGoal_ | Byte |
| nGen, nPop, nAttrib, ChromLen | Byte |
| Chrom, Child | TArChrom |
| Rules | TArRule |
| Model | array [0..100] of string; |
| nModelUnif,nInitPop | byte |

a. *ContAttrib* bertipe *array* digunakan untuk menyimpan no. id atribut bertipe kontinu, untuk *DiscAttrib* bertipe *array* digunakan untuk menyimpan no. id atribut bertipe diskrit.
b. *nRecord* digunakan untuk menyimpan jumlah *record* data-contoh.
c. *Attrib* bertipe *ArAttrib* digunakan untuk menyimpan nama-nama atribut dari data-contoh.
d. *nAttrib* digunakan untuk menyimpan banyaknya atribut.
e. *Distinct* bertipe *array TBit* digunakan untuk menyimpan nilai-nilai unik dari tiap-tiap atribut.
f. *DataBase* untuk menyimpan nilai keseluruhan dari data-contoh.
g. *nRecTrain* untuk menyimpan jumlah *record* dari data-latih, *nRecTest* untuk menyimpan jumlah *record* dari data-uji, *nRules* untuk menyimpan banyaknya aturan yang berhasil terbentuk.
h. *nGoalDistinct* untuk menyimpan jumlah nilai unik dari atribut tujuan, dan *IDGoal_* untuk menyimpan no. id atribut tujuan.
i. *nGen* digunakan untuk menyimpan jumlah generasi, *nPop* untuk menyimpan jumlah populasi, *nAttrib* untuk menyimpan jumlah atribut, dan *ChromLen* untuk menyimpan panjang kromosom.
j. *Chrom* digunakan untuk menyimpan kromosom-kromosom induk yang telah terbentuk dan *Child* untuk menyimpan kromosom-kromosom turunan.
k. *Rules* untuk menyimpan aturan yang berhasil terbentuk.
l. *Model* digunakan untuk menyimpan model pembangkitan populasi awal terstruktur.

m. *nModelUnif* digunakan untuk menyimpan banyak model dan *nInitPop* untuk menyimpan banyak populasi sekarang saat sistem sedang berjalan.

### 3.2.2 Perancangan Proses Pengkondisian Data-Contoh

Tahapan pengkondisian data merupakan tahapan awal dari proses sistem klasifikasi menggunakan Algoritma Genetika. Tahapan ini bertujuan untuk mengkondisikan data-contoh sesuai masukan yang dibutuhkan oleh sistem dalam rangka untuk memaksimalkan pencarian aturan yang layak dan akurat dengan Algoritma Genetika.

#### 3.2.2.1 Proses Mengisi Nilai Hilang

Pada data-contoh terdapat beberapa nilai hilang yang perlu diinisialisasi dengan sebuah nilai. Nilai hilang ini dalam banyak literatur biasanya disimbolkan dengan '?', dan dalam skripsi ini juga menerapkan simbol '?' sebagai penanda nilai hilang dalam data-contoh. Sehingga tahapan pengisian nilai hilang pada data-contoh hanya dikhususkan pada atribut yang memiliki nilai hilang '?'.

Untuk mengisi nilai hilang pada suatu atribut bernilai '?' dipilih sebuah metode yang terkenal yaitu persamaan Klasifikasi *Naïve Bayesian* (persamaan 2.3). Untuk lebih memahami proses pengisian nilai hilang, akan diberikan contoh perhitungannya.

Tabel 3.5 Tabel Kasus untuk Pengisian Nilai Hilang

| Age | Sex | ChestPain | RestBP | Cholesterol | BloodSugar | ECG | Class |
|---|---|---|---|---|---|---|---|
| 60 | 1 | 4 | 45 | 206 | 0 | 2 | 2 |
| 54 | 1 | **?** | 1 | 239 | 0 | 0 | 2 |
| 54 | 1 | 4 | 39 | 286 | 0 | 2 | 1 |
| 52 | 1 | 4 | 3 | 255 | 0 | 0 | 2 |
| 68 | 1 | 3 | 37 | 274 | 1 | 2 | 2 |
| 42 | 1 | 4 | 7 | 0 | 0 | 0 | 2 |
| 47 | 1 | 4 | 23 | 275 | 0 | 2 | 1 |
| 67 | 1 | 4 | 8 | 299 | 0 | 2 | 2 |
| 55 | 1 | 4 | 9 | 353 | 0 | 0 | 2 |
| 55 | 1 | 4 | 11 | 353 | 0 | 0 | 1 |

1. Pada tabel 3.5, ditemukan nilai hilang '?' pada atribut *ChestPain,* baris ke-2. Dengan persamaan Klasifikasi *Naïve Bayesian*, dihitung

peluang untuk semua jenis nilai pada atribut *ChestPain* yaitu nilai '4' dan nilai '3' seperti pada tabel 3.6. Tabel 3.6 ini untuk menghitung P(Ci) yaitu jumlah data dari $C_i$ terhadap jumlah keseluruhan dari sampel data.

Tabel 3.6 Contoh Tabel Perhitungan P(Ci)

| | n | P(Ci) |
|---|---|---|
| Nilai = 4 | 8 | 8/9 = 0,89 |
| Nilai = 3 | 1 | 1/9 = 0,11 |
| Total | 9 | |

Pada tabel 3.6, untuk atribut *ChestPain* dengan nilai=4, didapatkan 4 baris data yang memenuhi *ChestPain*=4 dari tabel kasus 3.5. Dan dihitung nilai P(Ci)-nya ternyata diperoleh 0,89. Sedang untuk nilai=3 didapatkan 3 baris data yang memenuhi *ChestPain*=3 dari tabel kasus 3.5, dan P(Ci)-nya diperoleh 0,11.

2. Dihitung peluang untuk semua jenis nilai pada atribut *ChestPain* yaitu nilai '4' dan nilai '3' dengan kelas = '2' seperti pada tabel 3.7. Tabel 3.7 ini untuk menghitung P(Ci/xt).P(xt).

Tabel 3.7 Contoh Tabel Perhitungan P(Ci/xt).P(xt)

| | Kelas = 2 | P(Ci/xt).P(xt) |
|---|---|---|
| Nilai = 4 | 5 | 5/6=0,83 |
| Nilai = 3 | 1 | 1/6=0,17 |
| Total | 6 | |

Pada tabel 3.7 untuk atribut *ChestPain*=4 dengan *kelas*=2 diperoleh 5 baris data dari tabel kasus 3.5 dan P(Ci=0/xt).P(xt) diperoleh 0,83. Untuk *ChestPain*=3 dengan kelas=2 diperoleh 1 baris data dari tabel 3.5 dan P(Ci=0/xt).P(xt) diperoleh 0,17.

3. Terakhir, dihitung nilai akhir dari contoh kasus untuk persamaan Klasifikasi *Naïve Bayesian* dari tabel kasus 3.5 pada atribut *ChestPain* baris ke-2 yaitu seperti ditunjukkan pada tabel 3.8. Tabel 3.8 digunakan untuk menghitung P(xt/Ci) yaitu persamaan akhir dari Klasifikasi *Naïve Bayesian*.

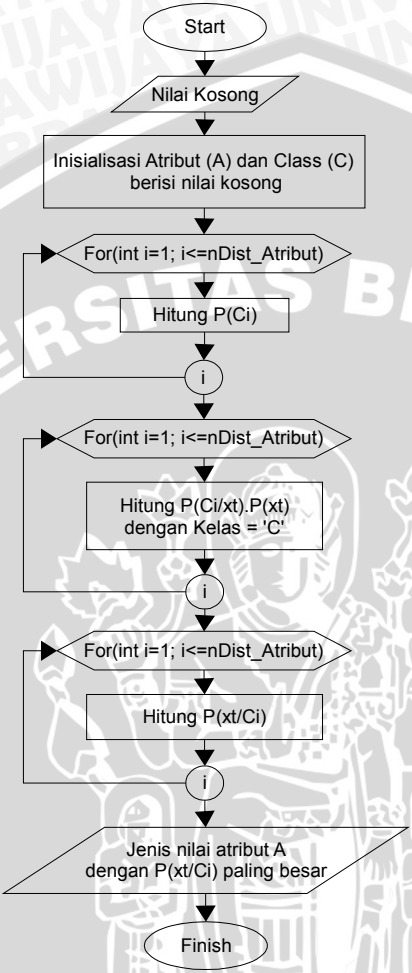
Tabel 3.8 Contoh Tabel Perhitungan P(xt/Ci)

| | P(xt/Ci) = [P(Ci/xt).P(xt)]/P(Ci) | |
|---|---|---|
| **Nilai = 4** | **0,74** | >> lebih besar |
| Nilai = 3 | 0,02 | |

Pada tabel 3.8 untuk *ChestPain*=4 didapatkan nilai akhir P(xt/Ci) senilai 0,74, sedang untuk *ChestPain*=3 didapatkan nilai akhir P(xt/Ci) senilai 0,02. Dari kedua nilai ini dibandingkan, ternyata *ChestPain*=4 memiliki nilai akhir P(xt/Ci) lebih besar dari *ChestPain*=3. Maka yang menginisialisasi nilai hilang '?' pada atribut *ChestPain* baris ke-2 dari tabel kasus 3.5 adalah nilai '4'.

Langkah-langkah no.1 sampai no.3 tersebut dilakukan berulang kali setiap ditemukan nilai hilang '?' pada keseluruhan atribut. Untuk skripsi ini dilakukan pengisian nilai hilang pada data-contoh yang telah ditentukan, yakni data-contoh *Crx*. Pada data-contoh *Crx* terdapat 67 nilai hilang yang tersebar pada banyak atribut. Sehingga proses pengisian nilai hilang dengan teknik Klasifikasi *Naïve Bayesian* ini dilakukan sebanyak 67 kali proses.

Untuk menunjukkan gambaran secara umum tentang proses pengisian nilai hilang dengan teknik Klasifikasi *Naïve Bayesian,* seperti yang telah diuraikan pada langkah no.1 sampai no.3 pada contoh pengisian nilai hilang sebelumnya, maka disertakan diagram alirnya. Gambar 3.2 merupakan diagram alir tentang proses pengisian nilai hilang dengan teknik Klasifikasi *Naïve Bayesian*. Proses pengisian nilai hilang '?' ini dilakukan pada tiap-tiap atribut satu-persatu yang mengandung nilai hilang.




Gambar 3.2 Diagram Proses Pengisian Nilai Hilang

### 3.2.2.2 Proses Pendiskritan Nilai Kontinu

Setelah menginisialisasi nilai hilang pada suatu atribut data-contoh, sistem melanjutkan tahapan pengkondisian data berikutnya yakni pendiskritan. Atribut yang dikenai proses pendiskritan yakni atribut yang ditetapkan sebagai atribut bertipe kontinu menurut deskripsi dari data-contoh yang dilampirkan.

Untuk mendiskritkan nilai kontinu pada atribut bertipe kontinu, dipilih sebuah metode pendiskritan yaitu Teknik *ChiMerge*. Untuk contoh perhitungannya yakni :

Tabel 3.9 Tabel Kasus untuk Pendiskritan

| Age | Sex | ChestPain | **RestBP** | Cholesterol | BloodSugar | ECG | Class |
|---|---|---|---|---|---|---|---|
| 60 | 1 | 4 | **45** | 206 | 0 | 2 | 2 |
| 54 | 1 | 4 | **1** | 239 | 0 | 0 | 2 |
| 54 | 1 | 4 | **39** | 286 | 0 | 2 | 1 |
| 52 | 1 | 4 | **3** | 255 | 0 | 0 | 2 |
| 68 | 1 | 3 | **37** | 274 | 1 | 2 | 2 |
| 42 | 1 | 4 | **7** | 0 | 0 | 0 | 2 |
| 47 | 1 | 4 | **23** | 275 | 0 | 2 | 1 |
| 67 | 1 | 4 | **8** | 299 | 0 | 2 | 2 |
| 55 | 1 | 4 | **9** | 353 | 0 | 0 | 2 |
| 55 | 1 | 4 | **11** | 353 | 0 | 0 | 1 |

Pada tabel kasus 3.9 misal ditetapkan tiga atribut bertipe kontinu. Yaitu *Age*, *RestBP*, dan *Cholesterol*. Berikutnya akan dipakai atribut *RestBP* untuk contoh perhitungan pendiskritan atribut bertipe kontinu.

1. Dilakukan pengurutan menaik (*ascending*) terhadap atribut *RestBP* sebagai variabel kontinu *F* dengan kelas *K*, kemudian dilakukan bagi 2 terhadap *F* setelah pengurutan. Kedua proses perhitungan ini seperti pada tabel 3.10.

Tabel 3.10 Contoh Tabel Pengurutan dan Bagi Dua Variabel Kontinu F

**(a)**

| **no.** | **F** | **K** |
|---|---|---|
| 1 | 45 | 2 |
| 2 | 1 | 2 |
| 3 | 39 | 1 |
| 4 | 3 | 2 |
| 5 | 37 | 2 |
| 6 | 7 | 2 |
| 7 | 23 | 1 |
| 8 | 8 | 2 |
| 9 | 9 | 2 |
| 10 | 11 | 1 |
| 11 | 59 | 1 |
| 12 | 46 | 1 |

>>

**(b)**

| **no.** | **Urut F** | **Bagi2 F** |
|---|---|---|
| 1 | 1 | 0 |
| 2 | 3 | ( 3+1 ) / 2 = 2 |
| 3 | 7 | ( 7+3 ) / 2 = 5 |
| 4 | 8 | ( 8+7 ) / 2 = 7,5 |
| 5 | 9 | ( 9+8 ) / 2 = 8,5 |
| 6 | 11 | ( 11+9 ) / 2 = 10 |
| 7 | 23 | ( 23+11 ) / 2 = 17 |
| 8 | 37 | ( 37+23 ) / 2 = 30 |
| 9 | 39 | ( 39+37 ) / 2 = 38 |
| 10 | 45 | ( 45+39 ) / 2 = 42 |
| 11 | 46 | ( 46+45 ) / 2 = 68,5 |
| 12 | 59 | ( 59+46 ) / 2 = 82 |

Pada tabel 3.10a berisi data asli variabel kontinu *F* dan kelas *K* sesuai tabel kasus 3.9. Setelah data variabel kontinue *F* dilakukan pengurutan menaik hasilnya seperti pada tabel 3.10b yakni pada kolom 'Urut F', sedang pada kolom 'Bagi2 F' merupakan hasil dari $Bagi2F_i = (UrutF_i + UrutF_{i-1})/2$.

2. Selanjutnya didefinisikan dua *interval* awal yang berdekatan, dengan batasan-batasan interval yang diambil dari nilai 'Bagi2 F' yang telah dihitung pada langkah no.1. Dimisalkan interval awalnya **[0;2]** ('0' diambil dari nilai 'Bagi2 F' baris ke-1 dan '2' diambil dari nilai 'Bagi2 F' baris ke-2) dan **[2;5]** ('2' diambil dari nilai 'Bagi2 F' baris ke-2 dan '5' diambil dari nilai 'Bagi2 F' baris ke-3). Kemudian semua nilai atribut *RestBP* dikelompokkan ke dalam *interval* [0;2] dan [2;5] yang bersesuaian dengan kelompok kelasnya seperti pada tabel 3.11.

Tabel 3.11 Contoh Tabel *Contingency Interval* [0;2] dan [2;5]

| **Interval** | **K = 1** | **K = 2** | **Total** |
|---|---|---|---|
| [ 0 ; 2 ] | $A_{11} = 0$ | $A_{12} = 1$ | $R_1 = 1$ |
| [ 2 ; 5 ] | $A_{21} = 0$ | $A_{22} = 1$ | $R_2 = 1$ |
| **Total** | $C_1 = 0$ | $C_2 = 2$ | $N = 2$ |

Dari tabel 3.10a ditemukan baris data untuk :

- $A_{11}$ yakni nilai *F* diantara 0-2 dengan kelas=1 sebanyak 0 data
- $A_{12}$ yakni nilai *F* diantara 0-2 dengan kelas=2 sebanyak 1 data
- $A_{21}$ yakni nilai *F* diantara 2-5 dengan kelas=1 sebanyak 0 data
- $A_{22}$ yakni nilai *F* diantara 2-5 dengan kelas=2 sebanyak 1 data

Kemudian dihitung masing nilai dari $R_1$, $R_2$, $C_1$, $C_2$, dan N.

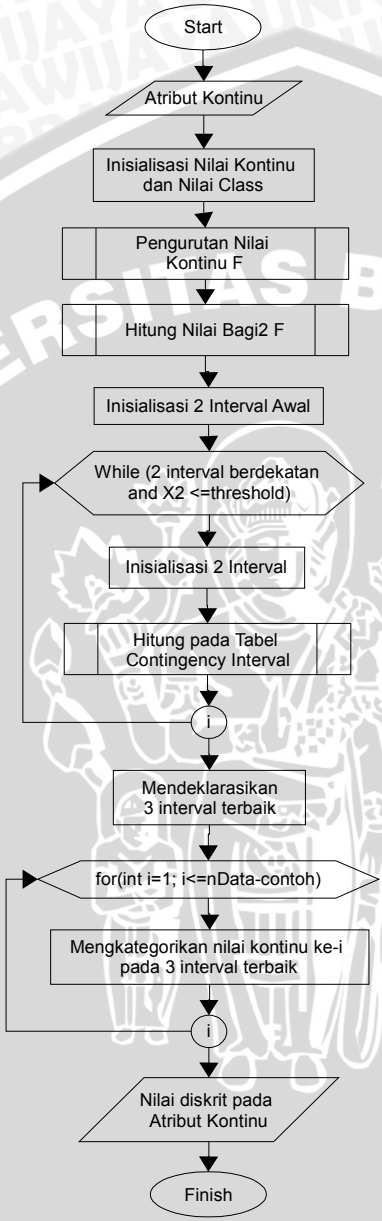
3. Dihitung frekuensi dan kualitas atribut $A_{ij}$ yang berturut-turut dilambangkan dengan variabel $E_{ij}$ dan $\chi^2$ (perhitungan menggunakan persamaan 2.4).

$$E_{11} = R_1 x C_1 / N = 1x0/2 \approx 0,1$$
$$E_{12} = R_1 x C_2 / N = 1x2/2 = 1$$
$$E_{21} = R_2 x C_1 / N = 1x0/2 \approx 0,1$$
$$E_{22} = R_2 x C_2 / N = 1x2/2 = 1$$
$$\chi^2 = (A_{11} - E_{11})^2 / E_{11} + (A_{12} - E_{12})^2 / E_{12} + (A_{21} - E_{21})^2 / E_{21} + (A_{22} - E_{22})^2 / E_{22}$$
$$= (0 - 0,1)^2 / 0,1 + (0 - 1)^2 / 1 + (0 - 0,1)^2 / 0,1 + (1 - 1)^2 / 1$$
$$= 1,2$$

4. Perhitungan $\chi^2$ diulang sampai ditemukan dua-*interval* yang saling berdekatan dan paling mendekati *threshold.* Semisal ditemukan *interval* [**0**;7,5] dan [7,5;**10**] dengan nilai $\chi^2 = 0,834$ dan *threshold* = 2,706. Karena [**0**;7,5] dan [7,5;**10**] merupakan dua-*interval* yang saling berdekatan dan paling mendekati *threshold* (0,834 < 2,706) maka kedua *interval* ini digabung menjadi [**0**;**10**].
5. Kemudian pencarian *interval* terus dilanjutkan hingga pada kasus permisalan ini akan ditemukan 3 *interval* yakni [0;10], [10;42], [42;59]. Dimana '59' adalah nilai maksimum pada atribut *RestBP*. Ketiga interval tersebut bisa dikodekan dengan nilai '1','2','3' atau dideskripsikan dengan nilai 'low', 'med', 'high'.
6. Selanjutnya semua data kontinu pada atribut *RestBP* dikategorikan berdasarkan 3 interval yang telah didapatkan pada langkah no.5. Bila nilai kontinu berada diantara rentang 0-10, maka nilai kontinu tersebut didefinisikan dengan nilai 'low'; bila berada diantara rentang 10-42, maka nilai kontinu tersebut didefinisikan dengan nilai 'med'; dan bila berada diantara rentang 42-59, maka nilai kontinu tersebut didefinisikan dengan nilai 'high'.

   Langkah no.1 sampai no.6 tersebut dilakukan berulang kali pada masing-masing atribut bertipe kontinu.

Gambar 3.3 merupakan diagram alir tentang proses pendiskritan nilai kontinu dengan teknik *ChiMerge,* seperti yang telah diuraikan masing-masing langkahnya pada contoh perhitungan pendiskritan atribut bertipe kontinu dari langkah no.1 sampai langkah no.6 sebelumnya.




Gambar 3.3 Diagram Proses Pendiskritan Nilai Kontinu

### 3.2.3 Perancangan Proses Algoritma Genetika

Dalam praktiknya, implementasi Algoritma Genetika disesuaikan dengan kasus yang ada untuk mendapatkan solusi maksimal yang diinginkan. Dalam subbab ini, akan dijelaskan komponen-komponen dari Algoritma Genetika yang telah disesuaikan menurut rumusan masalah dalam skripsi.

#### 3.2.3.1 Proses Pengkodean *Binary*

Tahapan pertama dalam operasi genetika adalah mengkodekan nilai-nilai atribut pada data-contoh ke dalam *binary*. Hal ini untuk menyesuaikan dengan masukan sistem yang nantinya akan memper-mudah dalam operasi genetika. Disertakan tabel 3.12 sebagai tabel kasus operasi genetika dengan kondisi nilai-nilai atribut belum dikodekan. Setelah dikodekan, terlihat pada tabel 3.13.

Kesimpulannya, makin banyak jenis nilai (*distinct value*) dari suatu atribut, berakibat makin panjang string *bit*-nya.

Tabel 3.12 Tabel Kasus untuk Operasi Genetika

| Age | Sex | ChestPain | RestBP | Cholesterol | BloodSugar | ECG | Class |
|---|---|---|---|---|---|---|---|
| high | 1 | 4 | high | low | 0 | 2 | 2 |
| med | 1 | 4 | low | low | 0 | 0 | 2 |
| med | 1 | 4 | high | med | 0 | 2 | 1 |
| med | 1 | 4 | low | low | 0 | 0 | 2 |
| high | 1 | 3 | high | med | 1 | 2 | 2 |
| low | 1 | 4 | low | low | 0 | 0 | 2 |
| low | 1 | 4 | med | med | 0 | 2 | 1 |
| high | 1 | 4 | low | med | 0 | 2 | 2 |
| med | 1 | 4 | low | high | 0 | 0 | 2 |
| med | 1 | 4 | med | high | 0 | 0 | 1 |

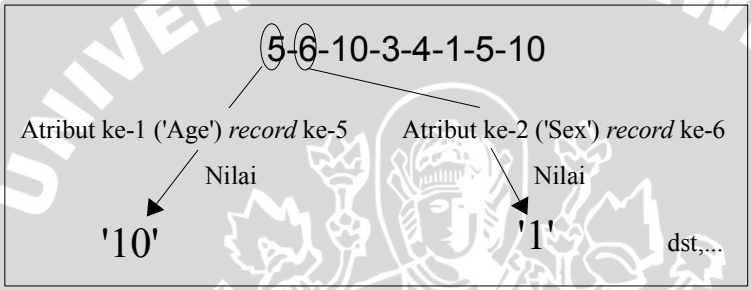
Tabel 3.13 Contoh Tabel Hasil Pengkodean *Binary*

| No | Age | Sex | ChestPain | RestBP | Cholesterol | BloodSugar | ECG | Class |
|---|---|---|---|---|---|---|---|---|
| 1 | 10 | 1 | 1 | 10 | 00 | 0 | 1 | 1 |
| 2 | 01 | 1 | 1 | 00 | 00 | 0 | 0 | 1 |
| 3 | 01 | 1 | 1 | 10 | 01 | 0 | 1 | 0 |
| 4 | 01 | 1 | 1 | 00 | 00 | 0 | 0 | 1 |
| 5 | 10 | 1 | 0 | 10 | 01 | 1 | 1 | 1 |
| 6 | 00 | 1 | 1 | 00 | 00 | 0 | 0 | 1 |
| 7 | 00 | 1 | 1 | 01 | 01 | 0 | 1 | 0 |
| 8 | 10 | 1 | 1 | 00 | 01 | 0 | 1 | 1 |
| 9 | 01 | 1 | 1 | 00 | 10 | 0 | 0 | 1 |
| 10 | 01 | 1 | 1 | 01 | 10 | 0 | 0 | 0 |

### 3.2.3.2 Proses Representasi Kromosom

Untuk langkah-langkah dalam representasi kromosom Algoritma Genetika akan dijelaskan dengan sebuah permisalan, yakni :

1. Dari tabel 3.13 dibangkitkan baris data secara acak untuk tiap-tiap atribut ke-*i*. Pada tabel 3.13 terdapat 8 atribut, artinya akan dibangkitkan *string* sepanjang 8 karakter yang berisi susunan baris data secara acak. Karakter ke-*i* akan mewakili atribut ke-*i*. Semisal didapatkan susunan *string* sepanjang 8 karakter dari 8 atribut yaitu 5-6-10-3-4-1-5-10 yang bisa diinterpretasikan seperti gambar 3.4.




Gambar 3.4 Interpretasi Susunan Baris Data Acak

Dari susunan 8 karakter string '5 – 6 – 10 – 3 – 4 – 1 – 5 - 10' bisa diinterpretasikan dengan lengkap (merujuk pada tabel 3.13) yakni :

- '5' : atribut ke-1 yakni *Age,* baris data ke-5 ⇒ bernilai '10'
- '6' : atribut ke-2 yakni *Sex,* baris data ke-6 ⇒ bernilai '1'
- '10' :atribut ke-3 yakni *Chestpain*, baris data ke-10 ⇒ bernilai '1'
- '3' : atribut ke-4 yakni *RestBp*, baris data ke-3 ⇒ bernilai '10'
- '4' : atribut ke-5 yakni *Cholesterol*, baris data ke-4⇒bernilai '00'
- '1' : atribut ke-6 yakni *BloodSugar*, baris data ke-1⇒ bernilai '0'
- '5' : atribut ke-7 yakni *ECG*, baris data ke-5 ⇒ bernilai '1'
- '10' : atribut ke-8 yakni *Class*, baris data ke-10 ⇒ bernilai '0'

Nilai akhir *binary* yaitu '10', '1','1','10','00','0','1','0' sebagai nilai *Vi* untuk tiap-tiap gen ke-*i*. Gen ke-*i* mewakili atribut ke-*i*.

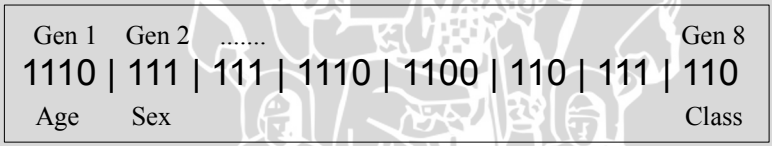
2. Setelah didapatkan masing-masing nilai *Vi* pada gen ke-*i*, maka dirangkai menjadi sebuah kromosom lengkap yang terdiri dari flag

*Fi*, operator *Oi*, dan value *Vi*. Untuk nilai *Fi* dan *Oi*, sebagai inisialisasi awal diisi '1', yang berarti (*Fi*='1') artinya atribut tersebut terdapat pada aturan, dan (*Oi*='1') artinya menggunakan operator "=". Rincian gen-gen penyusun sebuah kromosom tampak seperti pada tabel 3.14.

Tabel 3.14 Contoh Tabel Gen-gen Pembentuk Kromosom

| Nama atribut | Baris ke- | Nilai *Vi* | Kromosom '*Fi-Oi-Vi*' |
|---|---|---|---|
| Age | 5 | 10 | 1-1-10 |
| Sex | 6 | 1 | 1-1-1 |
| ChestPain | 10 | 1 | 1-1-1 |
| RestBP | 3 | 10 | 1-1-10 |
| Cholesterol | 4 | 00 | 1-1-00 |
| BloodSugar | 1 | 0 | 1-1-0 |
| ECG | 5 | 1 | 1-1-1 |
| Class | 10 | 0 | 1-1-0 |

Maka didapatkan representasi kromosom yang tersusun dari gen-gen menurut tabel 3.14 yang ditunjukkan pada gambar 3.5.

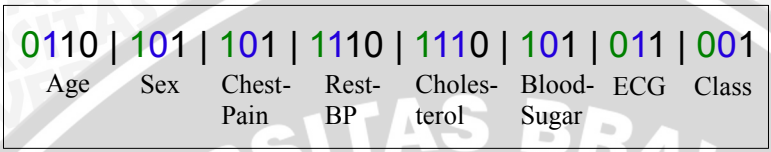



Gambar 3.5 Contoh Representasi Kromosom

Dari representasi kromosom pada gambar 3.5, bila diterjemahkan menjadi sebuah aturan yang utuh beratribut tujuan *Class*, dengan merujuk pada tabel kasus 3.12 dan hasil pengkodean *binary*-nya pada tabel 3.13, maka hasil aturannya yakni :

```
IF Age='high' AND Sex='1' AND ChestPain='4'
AND RestBP='high' AND Cholesterol='low' AND
BloodSugar='0' AND ECG='2' THEN Class='1'
```

Setelah dilakukan operasi genetika, semisal terbentuk kromosom turunan dengan string '01101011011110111010101 1001', maka bila diterjemahkan menjadi sebuah aturan dengan penjelasannya yang ditunjukkan pada gambar 3.6.




Gambar 3.6 Contoh Penterjemahan Kromosom Turunan

Untuk tiap-tiap gen (atribut) pada gambar 3.6, bit yang berwarna hijau berupa bagian *flag*, bit berwarna biru berupa bagian *operator*, dan bit berwarna hitam berupa bagian *value*. Dengan merujuk pada tabel kasus 3.12 dan hasil pengkodean *binary*-nya pada tabel 3.13, maka aturan hasil terjemahan dari kromosom turunan '0110101 10111101110111111001', yakni :

```
IF Sex<>'1' AND ChestPain<>'4' AND RestBP=
'high' AND Cholesterol='high' AND BloodSugar
<>'1' THEN Class='2'
```

### 3.2.3.3 Proses Pembangkitan Populasi Awal Terstruktur

Populasi awal Algoritma Genetika yang digunakan dalam skripsi ini mengimplementasikan pembangkitan populasi awal secara terstruktur, untuk dijadikan solusi awal dari penyelesaian kasus dan sebagai populasi induk untuk melahirkan populasi-populasi turunan.

Langkah-langkah untuk membangkitkan populasi awal terstruktur Algoritma Genetika akan dijelaskan dengan sebuah permisalan, yakni :

1. Dimisalkan kromosom awal '10101000010101010010' yang memiliki panjang kromosom=20 dan ditentukan jumlah populasi=10. Kemudian dihitung nilai *r*-nya, yaitu panjang maksimal model.

   $2^r \leq |\text{jumlah populasi}|$
   $2^r \leq |10|$
   $r = 3$

   Diperoleh r=3 atau panjang maksimal model = 3.

2. Selanjutnya dirancang model penyusunan kromosom-kromosom untuk pembangkitan populasi awal terstruktur dengan *r* maksimal =3 (hasil perhitungan langkah no.1). Model penyusunan kromosom dengan *r*=3 ditunjukkan pada tabel 3.15.

Tabel 3.15 Model Kromosom Terstruktur dengan r maksimal = 3

| Populasi | Model | | | |
|---|---|---|---|---|
| 1 | X | X | O | r = 3 |
| 2 | X | O | X | |
| 3 | O | X | X | |
| 4 | X | O | O | |
| 5 | O | X | O | |
| 6 | O | O | X | |
| 7 | X | O | | r = 2 |
| 8 | O | X | | |
| 9 | X | | | r = 1 |
| 10 | O | | | |

Model tersusun sebanyak jumlah populasi, dimana kotak berlatar abu-abu dengan simbol 'X' artinya bit-bit dalam kromosom dibalikkan nilainya, '0' jadi '1' dan '1' jadi '0'. Sedang kotak berlatar putih dengan simbol 'O' artinya bit-bit dalam kromosom tidak berubah nilainya.

Untuk penyusunan kromosom untuk r=3 maka mengacu pada model dengan populasi nomor 1 sampai 6. Untuk r=2 mengacu pada model dengan populasi nomor 7 sampai 8. Sedang untuk r=1 maka mengacu pada model dengan populasi nomor 9 sampai 10. Lebih mudahnya akan dijelaskan pada langkah no.3.

3. Setelah terbentuk model penyusunan kromosom dengan *r* maksimal=3, maka mulai dibangkitkan populasi awal terstruktur mengacu pada model tersebut. Untuk *r*=3, kromosom awal yang telah dimisalkan sebelumnya yaitu '10101000010101010010', dibagi menjadi 3 daerah seperti ditunjukkan pada gambar 3.7.

| 1010100 | 0010101 | 010010 |
|---|---|---|

Gambar 3.7 Contoh Pembagian Kromosom Awal dengan r=3

Mengacu pada model penyusunan kromosom (tabel 3.15) dengan r=3 yakni model dengan nomor populasi 1 sampai 6 maka dari kromosom awal yang telah terbagi menjadi 3 daerah (gambar 3.7) akan terbentuk $(2^3-2) = 6$ kromosom baru, seperti pada tabel 3.16.

Tabel 3.16 Contoh Kromosom Terstruktur Baru dengan r=3

| No. | Daerah Kromosom Baru | | |
|---|---|---|---|
| 1 | **0101011** | **1101010** | 010010 |
| 2 | **0101011** | 0010101 | **101101** |
| 3 | 1010100 | **1101010** | **101101** |
| 4 | **0101011** | 0010101 | 010010 |
| 5 | 1010100 | **1101010** | 010010 |
| 6 | 1010100 | 0010101 | **101101** |

Untuk r=2, mengacu pada model penyusunan kromosom (tabel 3.15) dengan r=2 yakni model dengan nomor populasi 7 sampai 8, maka dari kromosom awal yang telah terbagi menjadi 2 daerah seperti pada gambar 3.8 akan terbentuk $(2^2-2)=2$ kromosom baru, seperti pada tabel 3.17.

| | |
|---|---|
| 1010100001 | 0101010010 |

Gambar 3.8 Contoh Pembagian Kromosom Awal dengan r=2

Tabel 3.17 Contoh Kromosom Terstruktur Baru dengan r=2

| No. | Daerah Kromosom Baru | |
|---|---|---|
| 7 | **0101011110** | 0101010010 |
| 8 | 1010100001 | **1010101101** |

Sedang untuk r=1, mengacu pada model penyusunan kromosom (tabel 3.15) dengan nomor populasi 9 sampai 10, maka terbentuk dua kromosom baru seperti pada tabel 3.18.

Tabel 3.18 Contoh Kromosom Terstruktur Baru dengan r=1

| No. | Kromosom Baru |
|---|---|
| 9 | **01010111101010101101** |
| 10 | 10101000010101010010 |

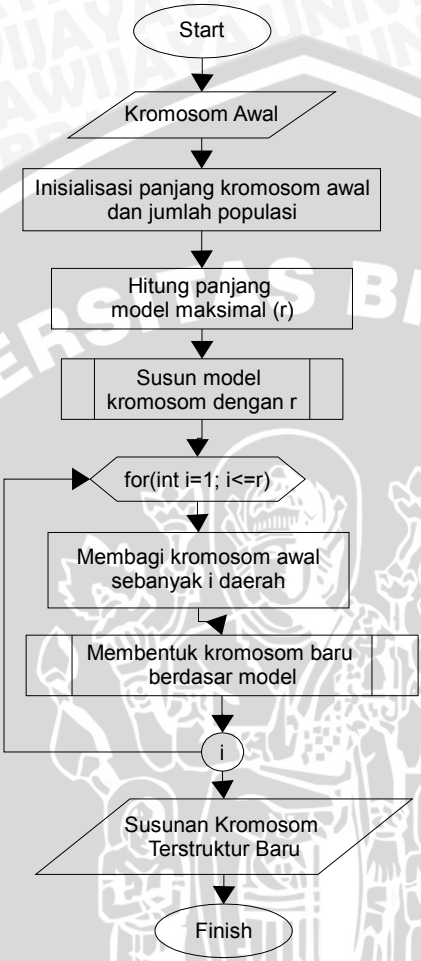
Sehingga semua kromosom baru hasil dari pembangkitan secara terstruktur dengan *r* maksimal=3, yang terbentuk dari kromosom awal '10101000010101010010' dengan panjang kromosom=20 dan jumlah populasi=10 ditunjukkan pada tabel 3.19.

Tabel 3.19 Contoh Kromosom Baru Terstruktur

| No. | Kromosom Baru |
|---|---|
| 1 | 01010111101010010010 |
| 2 | 01010110010101101101 |
| 3 | 10101001101010101101 |
| 4 | 01010110010101010010 |
| 5 | 10101001101010010010 |
| 6 | 10101000010101101101 |
| 7 | 01010111100101010010 |
| 8 | 10101000011010101101 |
| 9 | 01010111101010101101 |
| 10 | 10101000010101010010 |

Gambaran umum tentang pembangkitan kromosom awal terstruktur akan ditampilkan dalam bentuk diagram alir, seperti yang ditunjukkan pada gambar 3.9.

Gambar 3.9 merupakan diagram alir tentang proses pembangkitan kromosom awal terstruktur pada Algoritma Genetika dengan langkah-langkahnya yang telah diuraikan pada sebuah permisalan pembangkitan populasi awal terstruktur Algoritma Genetika langkah no.1 sampai langkah no.3 sebelumnya.




Gambar 3.9 Diagram Proses Pembangkitan Populasi Awal Terstruktur

### 3.2.3.4 Proses Operasi Genetika

Operator yang dipilih pada skripsi ini yakni operator yang diharapkan mampu untuk mendapatkan kromosom berkualitas tinggi pada tiap-tiap generasi turunan. Pada subbab ini, akan dijelaskan operator-operator genetika yang diterapkan pada skripsi ini.

#### 3.2.3.4.1 Proses Seleksi Roda *Roullette*

Individu-individu pada populasi induk diseleksi untuk mempertahankan individu terbaik dan menyisihkan individu lemah. Pada Seleksi Roda Roullette individu dengan nilai *fitness* yang tinggi berkemungkinan banyak untuk terpilih dibanding individu dengan *fitness* rendah. Langkah-langkah dari Seleksi Roda *Roullette* pada Algoritma Genetika, yakni :

1. Misal ada 10 individu baru dengan masing-masing bit kromosom dan nilai *fitness*-nya yang telah terurut menurun (*descending*) seperti ditunjukkan pada tabel 3.20. Kemudian dihitung nilai probabilitasnya dari tiap-tiap kromosom.

Tabel 3.20 Contoh Perhitungan Probabilitas Kromosom

| Individu | Kromosom | Fitness | Probabilitas |
|---|---|---|---|
| 1 | 101010100010100101010 | 0,083 | 0,083/0,402 = 0,21 |
| 2 | 101010100000101010010 | 0,067 | 0,067/0,402 = 0,17 |
| 3 | 010101001010100000101 | 0,05 | 0,050/0,402 = 0,12 |
| 4 | 010100100001111100101 | 0,045 | 0,045/0,402 = 0,11 |
| 5 | 101010101111010101010 | 0,038 | 0,038/0,402 = 0,10 |
| 6 | 010101000111101010100 | 0,029 | 0,029/0,402 = 0,07 |
| 7 | 111010101000101010101 | 0,025 | 0,025/0,402 = 0,06 |
| 8 | 110101010101001010001 | 0,024 | 0,024/0,402 = 0,06 |
| 9 | 110101010001010101010 | 0,022 | 0,022/0,402 = 0,06 |
| 10 | 101111010101010100101 | 0,018 | 0,018/0,402 = 0,05 |
| Total | | **0,402** | |

2. Setelah didapatkan nilai probabilitas dari tiap-tiap kromosom maka dilanjutkan dengan menghitung probabilitas kumulatif dari tiap-tiap kromosom. Langkah ini seperti ditunjukkan di dalam tabel 3.21, yakni tabel contoh perhitungan probabilitas kumulatif.

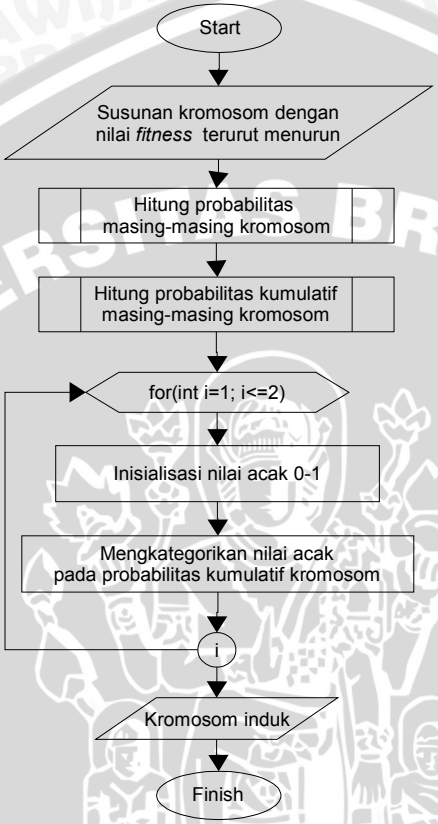
Tabel 3.21 Contoh Perhitungan Probabilitas Kumulatif

| Individu | Fitness | Prob | Probabilitas Kumulatif |
|---|---|---|---|
| 1 | 0,083 | 0,21 | 0,21 |
| 2 | 0,067 | 0,17 | 0,21+0,17 = 0,38 |
| 3 | 0,050 | 0,12 | 0,21+0,17+0,12 = 0,50 |
| 4 | 0,045 | 0,11 | 0,21+0,17+0,12+0,11 = 0,61 |
| 5 | 0,038 | 0,10 | 0,21+0,17+0,12+0,11+0,10 = 0,71 |
| 6 | 0,029 | 0,07 | 0,21+0,17+0,12+0,11+0,10+0,07 = 0,78 |
| 7 | 0,025 | 0,06 | 0,21+0,17+0,12+0,11+0,10+0,07+0,06 = 0,84 |
| 8 | 0,024 | 0,06 | 0,21+0,17+0,12+0,11+0,10+0,07+0,06+0,06 = 0,90 |
| 9 | 0,022 | 0,06 | 0,21+0,17+0,12+0,11+0,10+0,07+0,06+0,06+0,06 = 0.95 |
| 10 | 0,018 | 0,05 | 0,21+0,17+0,12+0,11+0,10+0,07+0,06+0,06+0,06+0,05 = 1,00 |
| Total | **0,402** | | |

3. Setelah didapatkan probabilitas kumulatif untuk tiap-tiap individu, dipilihlah angka pecahan acak antara 0-1. Semisal didapatkan angka pecahan acak 0,17. Kemudian nilai 0,17 dikategorikan ke dalam probabilitas kumulatif individu yang tersusun pada tabel 3.21, dimana nilai acak $\leq$ probabilitas kumulatif individu. Akhirnya ditemukan individu terpilih yang didapatkan dari nilai 0,17 yaitu individu ke-1 ($0,17 \leq 0,21$).
4. Langkah no.3 dilakukan dua kali, yakni inisialisasi individu untuk induk 1 dan induk 2. Misal didapatkan dua angka acak 0,17 dan 0,60. Maka setelah dikategorikan ke dalam tabel 3.21 yang berisi probabilitas kumulatif individu, maka ditemukan individu terpilih untuk masing-masing nilai acak. Yakni individu ke-1 untuk nilai acak 0,17 ($0,17 \leq 0,21$) dan individu ke-4 untuk nilai acak 0,60 ($0,60 \leq 0,61$). Sehingga didapatkan dua induk untuk pembentukan individu-individu turunan, yakni individu ke-1 dan individu ke-4.

Untuk memperjelas tentang gambaran umum Seleksi Roda Roullette maka disertakan diagram alirnya, dengan langkah-langkah yang telah diuraikan sebelumnya pada permisalan Seleksi Roda Roullette dari langkah no.1 sampai langkah no.4 sebelumnya.

Pada gambar 3.10 ditunjukkan diagram alir tentang proses Seleksi Roda *Roullette* Algoritma Genetika.




Gambar 3.10 Diagram Proses Seleksi Roda *Roullette*

### 3.2.3.4.2 Proses Persilangan Terstruktur

Teknik Persilangan Terstruktur akan meningkatkan kemampuan Algoritma Genetika dalam menemukan solusi optimum global dengan mencegah konvergensi prematur. Persilangan Terstruktur selalu diterapkan pada sistem ketika terjadi operasi genetika, artinya Persilangan Terstruktur ini tidak dibatasi dengan persentase kemungkinan persilangan.

Langkah-langkah dari Persilangan Terstruktur Algoritma Genetika akan dijelaskan dengan sebuah permisalan, yakni :

1. Dimisalkan kromosom awal $C_0$ = '10101000010101010010' dan $C_1$ = '01010100101101110011'. Kemudian dibandingkan dan dideteksi bit-bit yang berbeda antara kedua kromosom tersebut, sehingga nantinya akan membentuk sebuah model kromosom baru Y, seperti ditunjukkan pada gambar 3.11.

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $C_0$ | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| $C_1$ | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| Y | * | * | * | * | * | * | 0 | 0 | * | * | * | 1 | 0 | 1 | * | 1 | 0 | 0 | 1 | * |

Gambar 3.11 Contoh Kromosom Awal $C_0$, $C_1$ dan Kromosom Baru Y

2. Bit-bit berbeda pada model kromosom baru Y pada gambar 3.11 (yang disimbolkan *) terdapat 11 bit. Maka dibangkitkan 11 bit acak untuk mengisi 11 bit tersebut. Semisal 11 bit acak tersebut yakni '01001110101'. Dengan mengacu pada model r=2 (r yaitu panjang maksimal model, yang telah dibahas pada pembangkitan populasi awal terstruktur sebelumnya), maka kumpulan bit acak '01001110101' dibagi 2 daerah, seperti pada gambar 3.12.

| | |
|---|---|
| 010011 | 10101 |

Gambar 3.12 Contoh Pembagian Kumpulan Bit Acak dengan r=2

3. Kemudian, bit-bit berbeda pada kromosom Y (disimbolkan *) diisi dengan kombinasi dari 2 daerah pembagian kumpulan bit acak (mengacu pada gambar 3.11), yakni 010011|10101, 010011|01010, 101100|10101, 101100|01010. Maka kromosom baru hasil persilangan terstruktur kromosom C0 dan C1 seperti pada gambar 3.13.

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| B | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| C | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| D | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |

Gambar 3.13 Contoh Kromosom Baru Hasil Persilangan Terstruktur

4. Setelah didapat 4 kromosom baru hasil persilangan terstruktur, maka dipilih secara acak dari ke-4 kromosom tersebut untuk dijadikan calon kromosom turunan. Bila *fitness*-nya dinilai baik, maka calon kromosom turunan ini akan menggantikan kromosom induk yang kurang baik nilai *fitness*-nya.

#### 3.2.3.4.3 Proses Mutasi Bit, Gen, atau Kromosom

Mutasi sebagai upaya menggabungkan konfigurasi genetik baru untuk menemukan solusi optimal dan mencegah konvergensi prematur. Parameter kemungkinan mutasi pada sistem ditentukan senilai 0,9. Sistem akan mengacak nilai sembarang, bila didapat nilai acak ≤ 0,9 maka sistem melakukan mutasi, bila > 0,9 tidak dilakukan mutasi.

Penerapan mutasi dalam skripsi ini dilakukan secara acak untuk ketiga jenis mutasi. Yakni mutasi bit, mutasi gen, atau mutasi kromosom. Konsep dari ketiga jenis mutasi ini telah dijelaskan pada subbab 2.2.4.4.3 sebelumnya.

Sistem membangkitkan nilai acak *x* antara 0-2. Bila didapatkan *x*=0 maka sistem melakukan mutasi bit, bila *x*=1 maka sistem melakukan mutasi gen, bila *x*=2 maka sistem melakukan mutasi kromosom.

#### 3.2.3.5 Proses Perhitungan Nilai Kelayakan dan Keakuratan Aturan

Dari perhitungan nilai kelayakan *antecedent-consequent* dan keakuratan terhadap aturan, akan menghasilkan nilai *fitness* Algoritma Genetika untuk tiap-tiap kromosom (representasi dari aturan).

Langkah-langkah untuk menghitung nilai *fitness* Algoritma Genetika akan dijelaskan dengan sebuah permisalan, yakni :

1. Dengan merujuk pada tabel 3.12 sebagai tabel kasus untuk operasi genetika, dimisalkan aturan yang terbentuk beserta penentuan parameter-parameter lain yakni :

   Aturan : **IF ChestPain = 4 AND BloodSugar = 0 THEN Class = 1**
   Parameter : Atribut tujuan = 'CLASS'
   Jenis nilai atribut tujuan = '1','2'

2. Sesuai persamaan fungsi nilai *fitness* (persamaan 2.12), tahapan awal dihitung nilai *Info(Gk='CLASS')* (persaman 2.8), yang diperoleh nilai=0,88, perhitungannya ditunjukkan pada tabel 3.22.

Tabel 3.22 Contoh Perhitungan nilai *Info*(Gk='CLASS')

| value Class | n | Pr | INFO(gk) |
|---|---|---|---|
| 2 | 7 | 7/10 = 0,7 | 0,7*LOG2(0,7) = -0,36 |
| 1 | 3 | 3/10 = 0,3 | 0,3*LOG2(0,3) = -0,52 |
| **Total** | 10 | Tot INFO(gk) | -1*(-0,36+(-0,52)) = **0,88** |

Berdasar tabel kasus 3.12, untuk *Class*=2 ditemukan 7 *record* dan untuk *Class*=1 ditemukan 3 *record*. Kemudian dihitung masing-masing nilai *Pr*-nya (nilai peluang), untuk *Class*=2 dihasilkan 0,7 dan *Class*=1 dihasilkan 0,3. Terakhir, dihitung masing-masing nilai *INFO(gk)*-nya, untuk *Class*=2 dihasilkan -0,36 dan untuk *Class*=1 dihasilkan -0,52. Dari kedua nilai *INFO* ini dihitung untuk mendapatkan nilai *Info(Gk='CLASS')* senilai **0,88**.

3. Dihitung nilai *Info(Gk='CLASS'|Ai='ChestPain')* (persamaan 2.9) yang ditunjukkan pada tabel 3.23 dan perhitungan setelahnya.

Tabel 3.23 Contoh Perhitungan *Info(Gk='CLASS'|Ai='ChestPain')*

| ChestPain | n | Pr | value Class | n | Pr | Pr log2(Pr) |
|---|---|---|---|---|---|---|
| 4 | 9 | 9/10 = **0,9** | 2 | 6 | 6/9 = 0,67 | 0,67*log2(0,67) ~ 0,49 |
| | | | 1 | 3 | 3/9 = 0,33 | 0,33*log2(0,33) ~ 0,14 |
| | | | *Total* | 9 | *Tot Info* | -1*(0,49+0,14) = **0,63** |
| 3 | 1 | 1/10 = **0,1** | 2 | 1 | 1/1 = 1 | 1*log2(1) ~ 1 |
| | | | 1 | 0 | 0/1 = 0 | 0*log2(0) ~ 0 |
| *Total* | 10 | | *Total* | 1 | *Tot Info* | -1*(1+0) = **-1** |

*Info(Gk='CLASS'|Ai='ChestPain')* = (0,9*-0,63) + (0,1*-1)
= **-0,67**

Perhitungan *Info* terhadap atribut *antecedent* dimulai dari atribut *antecedent* yang pertama yakni *ChestPain*. Dari tabel kasus 3.12, untuk *ChestPain*=4 ditemukan 9 *record* dan untuk *ChestPain*=3 ditemukan 1 *record*. Lalu dihitung masing-masing *Pr*-nya, yang dihasilkan untuk *ChestPain*=4 yakni 0,9 dan untuk *ChestPain*=3 yakni 0,1.

Selanjutnya, dicari *ChestPain*=4 dengan *Class*=2 yang ditemukan 6 *record* dan *ChestPain*=4 dengan *Class*=1 yang ditemukan 3 *record*. Lalu dihitung masing-masing nilai *Pr*-nya, yang dihasilkan untuk *ChestPain*=4 dengan *Class*=2 yakni 0,67, sehingga *Pr log2(Pr)*-nya dihasilkan nilai 0,49; dan untuk *ChestPain*=4 dengan *Class*=1 yakni 0,33 sehingga *Pr log2(Pr)*-nya dihasilkan nilai 0,14. Kemudian nilai dari kedua *Pr log2(Pr)* dihitung *Total Info*-nya yang menghasilkan nilai -0,63.

Sedang untuk *ChestPain*=3 dengan *Class*=2 ditemukan 1 *record* dan *ChestPain*=3 dengan *Class*=1 ditemukan 0 *record*. Lalu dihitung masing-masing nilai *Pr*-nya, yang dihasilkan untuk *ChestPain*=3 dengan *Class*=2 yakni 1, sehingga *Pr log2(Pr)*-nya dihasilkan nilai 1; dan untuk *ChestPain*=3 dengan *Class*=1 yakni 0 sehingga *Pr log2(Pr)*-nya dihasilkan nilai 0. Kemudian nilai dari kedua *Pr log2(Pr)* dihitung *Total Info*-nya yang menghasilkan nilai 1.

Hasil akhir dari *Info(Gk='CLASS'|Ai='ChestPain')* dihasilkan dari total antara *Pr* dikali *Pr log2(Pr)* untuk *ChestPain*=4 dan *ChestPain*=3 yakni **-0,67**.

4. Dihitung nilai *InfoGain (Ai='ChestPain')* yakni perkalian antara *Info(Gk='CLASS')* dengan *Info(Gk='CLASS'|Ai='ChestPain')*, yang dihasilkan nilai akhir *InfoGain (Ai='ChestPain')* yakni **-0,59**.

   *InfoGain(Ai='ChestPain')=Info(Gk='CLASS')*Info(Gk='CLASS'|Ai='ChestPain')*
   = 0,88 * -0,67
   = **-0,59**

5. Dihitung nilai *Info(Gk='CLASS'|Ai='BloodSugar')*. Perhitungan untuk atribut *antecedent BloodSugar* ini sama dengan perhitungan untuk atribut antecedent *ChestPain* pada langkah 3 dan 4 sebelumnya. Perhitungan dari *InfoGain (Ai='BloodSugar')* ditunjukkan pada tabel 3.24 dan perhitungan setelahnya.

Tabel 3.24 Contoh Perhitungan *Info(Gk='CLASS'|Ai='BloodSugar')*

| BloodSugar | n | Pr | value Class | n | Pr | Pr log2(Pr) |
|---|---|---|---|---|---|---|
| 0 | 9 | 9/10 = **0,9** | 2 | 6 | 6/9 = 0,67 | 0,67*log2(0,67) ~ 0,5 |
| | | | 1 | 3 | 3/9 = 0,33 | 0,33*log2(0,33) ~ 0,14 |
| | | | *Total* | 9 | *Tot Info* | -1*(0,5+0,14) = **-0,64** |
| 1 | 1 | 1/10 = **0,1** | 1 | 0 | 0/1 = 0 | 0*log2(0) ~ 0 |
| | | | 2 | 1 | 1/1 = 1 | 1*log2(1) ~ 1 |
| *Total* | 10 | | *Total* | 1 | *Tot Info* | -1*(0+1) = **-1** |

*Info(Gk='CLASS'|Ai='BloodSugar')* = (0,9*-0,64) + (0,1*-1)
= **-0,67**

Hasil akhir dari *Info(Gk='CLASS'|Ai='BloodSugar')* didapatkan dari total antara *Pr* dikali *Pr log2(Pr)* untuk *BloodSugar*=0 dan *BloodSugar*=1 yakni **-0,67**

6. Dihitung nilai *InfoGain (Ai='BloodSugar')* yakni perkalian *Info(Gk='CLASS')* dan *Info(Gk='CLASS'|Ai='BloodSugar')*, yang dihasilkan nilai akhir *InfoGain (Ai='BloodSugar')* yakni **-0,59**.

*InfoGain(Ai='BloodSugar')=Info(Gk='CLASS')*Info(Gk='CLASS'*
*|Ai='BloodSugar')*
= 0,88 * -0,67
**= -0,59**

7. Dihitung jumlah atribut pada *antecedent* (disimbolkan *n*), jumlah kemungkinan nilai pada atribut tujuan *Gk* pada *consequent* (disimbolkan *(|dom(Gk)|)* ), dan nilai akhir dari kelayakan *antecedent* (disimbolkan *AI*) yakni persamaan 2.6.

*n* = **2**
didapatkan dua atribut pada *antecedent* yakni *ChestPain* dan *BloodSugar*.

*|dom(Gk)|)* = **2**
didapatkan dua nilai untuk atribut tujuan *Class* yakni '1' dan '2'.

$$AI = 1 - \left[ \frac{\frac{\sum_{i=1}^{n} InfoGain(A_i)}{n}}{\log_2(|dom(G_k|))} \right]$$

$$= 1 - \left[ \frac{\frac{-059 + (-0{,}59)}{2}}{\log_2(2)} \right]$$

$$= 1 - \left[ \frac{-059}{1} \right]$$

$$= \mathbf{0{,}41}$$

Untuk menghitung nilai akhir kelayakan *antecedent* dihitung dari 1 dikurang nilai total InfoGain untuk *InfoGain(Ai='ChestPain')* yang dihitung pada langkah no.4 dan ditambah *InfoGain (Ai='BloodSugar')* yang dihitung pada langkah no.6, lalu dibagi *n* dan dibagi lagi dengan log2 dari *dom(Gk)*. Maka didapatkan hasil akhir kelayakan *antecedent* senilai **0,41**.

8. Dihitung frekuensi relatif atribut tujuan *Gk* atau *Pr(Gk)* dan ditentukan nilai $\beta$ oleh user yakni 2. Nilai *Pr(Gk)* dan $\beta$ untuk menghitung nilai kelayakan *consequent* atau *CI* yakni persamaan 2.10..

$Pr(Gkl = 'Class=1') = 2$

$\beta = 2$

$$CI = \left(1 - \Pr(G_{kl})\right)^{1/\beta}$$

$= (1\text{-}2^{1/2})$

$= \mathbf{0{,}84}$

Didapatkan hasil akhir dari nilai kelayakan *consequent CI* yakni senilai **0,84**.

9. Selanjutnya, dihitung jumlah *record* dari tabel kasus 3.12 yang memenuhi *antecedent* dan *consequent* dari aturan, jumlah *record* dari tabel kasus 3.12 yang memenuhi *antecedent* dari aturan, dan nilai keakuratan prediksi yang berturut-turut dilambangkan dengan *X, Y, PA* (persamaan 2.11).

   Dari aturan yang dimisalkan sebelumnya yakni :

   **IF ChestPain = 4 AND BloodSugar = 0 THEN CLASS = 1**
   antecedent — consequent

   didapatkan $X$ = 3, yakni record ke-3, 7, dan 10,
   didapatkan $Y$ = 9, yakni semua record kecuali ke-5,
   maka $PA = (X\text{-}1/2) / Y$
   $= (3\text{-}1/2) / 9$
   $= \mathbf{0,28}$

10. Terakhir, dihitung nilai fungsi *fitness* dengan $w_1$ dan $w_2$ yang telah didefinisikan oleh user untuk bobot (*weight*) berturut-turut 1 dan 2.

    $w_1 = 1,\ w_2 = 2$

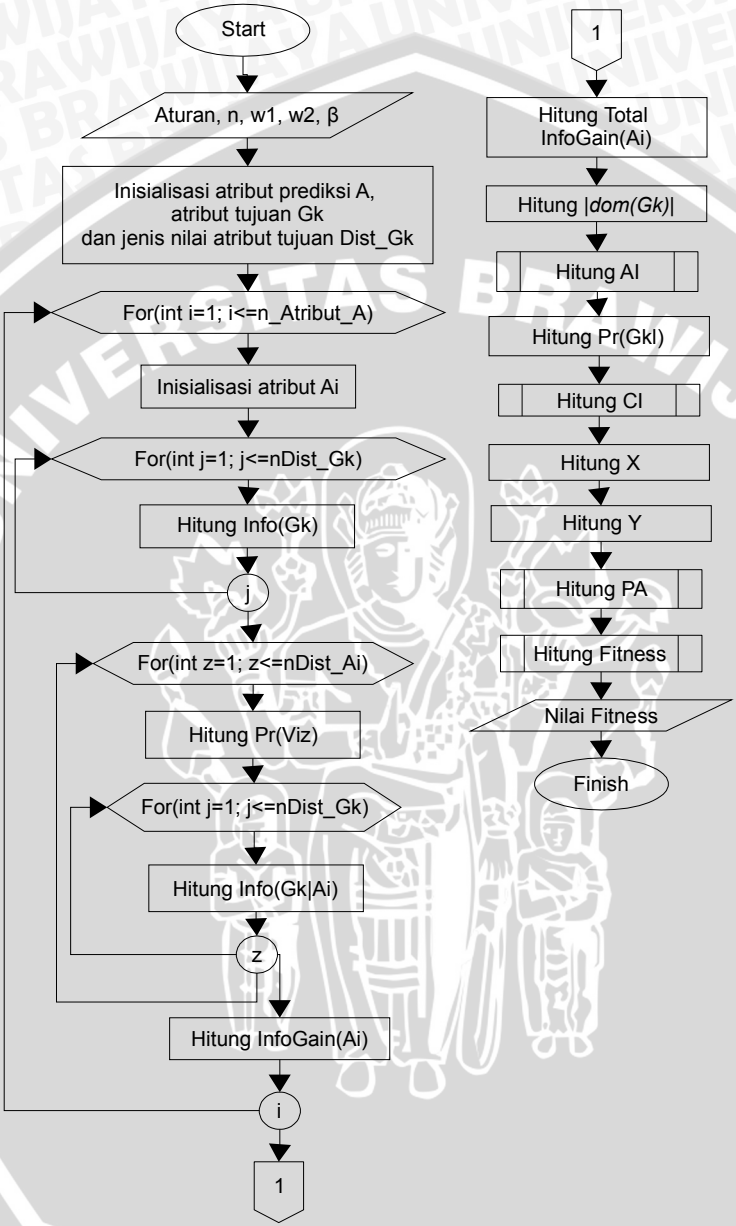
    $$Fitness = \frac{w_1 \frac{AI + CI}{2} + w_2 \cdot PA}{w_1 + w_2}$$

    $$= \frac{1 \cdot \frac{0,41 + 0,84}{2} + 2 \cdot 0,28}{1 + 2}$$

    $$= \mathbf{0,395}$$

    Dan didapatkan nilai akhir fungsi *fitness* untuk aturan *IF ChestPain=4 AND BloodSugar=0 THEN CLASS=1* berdasarkan tabel kasus 3.12 senilai **0,395**.

Pada gambar 3.14 merupakan diagram alir tentang proses perhitungan nilai kelayakan dan keakuratan aturan untuk menghasilkan nilai *fitness* dari sebuah kromosom untuk kasus Algoritma Genetika sesuai rumusan masalah dalam skripsi.




Gambar 3.14 Diagram Proses Perhitungan Nilai Kelayakan dan Keakuratan Aturan untuk Menghasilkan Nilai *Fitness* Kromosom

## 3.3 Perancangan Tabel Data-Latih dan Data-Uji

Tabel database yang digunakan untuk data-latih dan data-uji berupa satu tabel yang dipakai berdua untuk data-latih dan data-uji. Tabel ini tidak berelasi, karena hanya untuk menyimpan data saja. Ketika hendak menguji data-latih, maka sistem akan mengambil *record* (baris data) dari tabel sebanyak *n* persentase data-latih mulai dari *record* awal. Sedang ketika hendak menguji data-uji, maka sistem akan mengambil *record* tabel sebanyak *n* persentase data-uji dimulai dari *record* paling akhir. Untuk nilai *n* persentase data-latih dan *n* persentase data-uji ditentukan oleh pengguna.

Sedang format tabel yang dipakai untuk data-latih dan data-uji seperti ditunjukkan pada gambar 3.15, dengan tipe semua kolom bertipe char atau varchar. Dari atribut-atribut ini, nanti pengguna memilih satu atribut sebagai atribut tujuan, tidak selalu harus menggunakan atribut tujuan 'Kelas'.

| **Atribut ke-1** | **Atribut ke-2** | **.....** | **Atribut ke-n** | **Kelas** |
|---|---|---|---|---|

Gambar 3.15 Format Tabel untuk Data-Latih dan Data-Uji

## 3.4 Perancangan Antarmuka

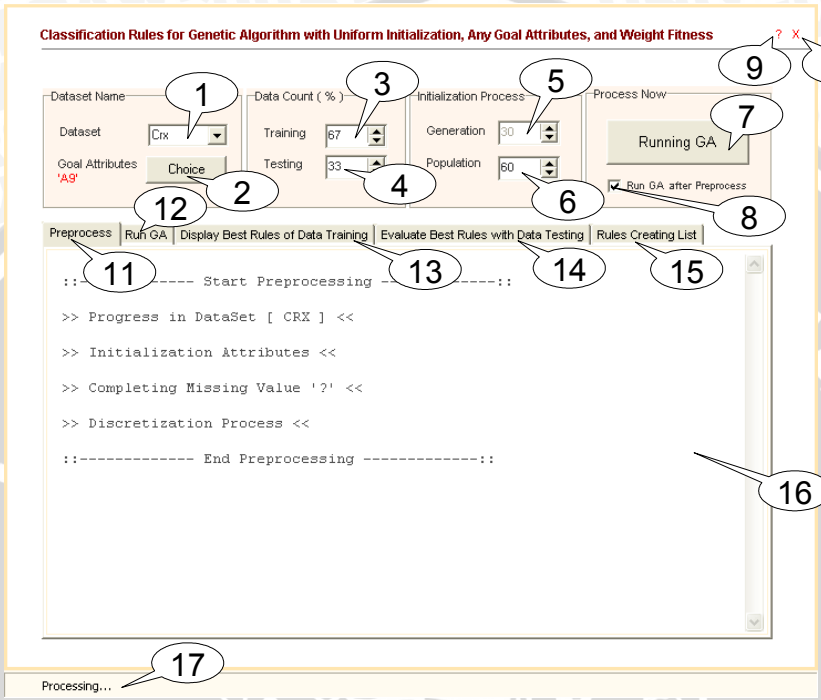
Perancangan antarmuka dari sistem yang dibuat untuk menghasilkan aturan-aturan beserta penilaian kelayakan dan keakuratannya pada skripsi ini terdiri dari 2 *form*.

Pertama, *form* utama sebagai *form* untuk mengolah data-contoh sampai menampilkan hasil akhir berupa aturan-aturan. *Form* ini dibagi menjadi 5 tab yakni tab untuk menampilkan proses pengkondisian data-contoh, tab untuk menampilkan proses Algoritma Genetika, tab untuk menampilkan analisa aturan terbaik berdasar data-contoh, tab untuk menampilkan evaluasi aturan terbaik berdasar data-uji, dan tab untuk menampilkan sejumlah aturan terbaik dari tiap-tiap jenis nilai atribut tujuan.

Kedua, *form* statistik untuk mengatur atribut tujuan dari data-contoh dan menampilkan deskripsi dari data-contoh.

Gambar ilustrasi dari *form-form* yang dibuat untuk sistem ditunjukkan pada subbab 3.4.1 sampai 3.4.5.

### 3.4.1 Rancangan *Form* Utama Tab Pengkondisian Data-Contoh



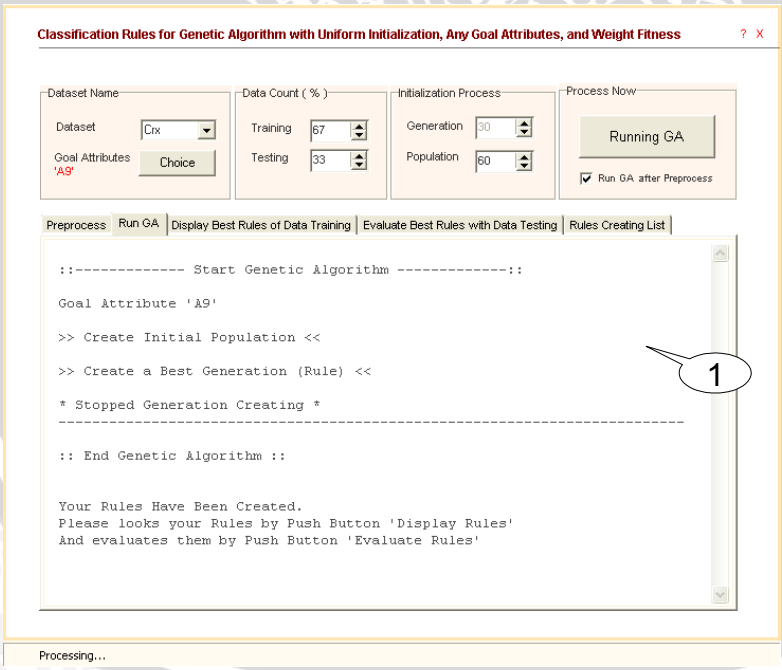

Gambar 3.16 Rancangan Form Utama Tab Pengkondisian Data-Contoh

Pada gambar 3.16 ditampilkan rancangan *form* utama pada tab pengkondisian data. Secara terurut, penjelasan nomor-nomor penunjuk pada gambar 3.16 yakni :

1. Pilihan data-contoh untuk diterapkan proses. Sementara untuk pengujian pada sistem hanya menggunakan 1 pilihan data-contoh yakni *Crx*.
2. Pilihan atribut tujuan untuk dijadikan *consequent* pada aturan. Bila tombol *Choice* di-*klik*, akan muncul *form* statistik dari data-contoh yang dipilih.
3. Penentuan persentase data-latih untuk pembuatan aturan.
4. Penentuan persentase data-uji untuk pengujian aturan.
5. Penentuan jumlah generasi Algoritma Genetika untuk membatasi jumlah generasi maksimal yang terbentuk.

6. Penentuan jumlah populasi Algoritma Genetika untuk membatasi jumlah populasi dalam suatu generasi.
7. Tombol untuk mengeksekusi tahapan-tahapan sistem.
8. Pengecek untuk mengotomatisasi tahapan proses genetika langsung setelah tahapan pengkondisian data telah selesai.
9. Tombol untuk membuka halaman profil programmer.
10. Tombol keluar dari sistem.
11. Tab khusus untuk proses pengkondisian data.
12. Tab khusus untuk proses Algoritma Genetika.
13. Tab khusus untuk menganalisa aturan terbaik dengan data-latih.
14. Tab khusus untuk mengevaluasi aturan terbaik.dengan data-uji.
15. Tab khusus untuk proses mencetak aturan-aturan yang terbentuk tiap-tiap jenis nilai atribut tujuan.
16. Memo mencetak kerja sistem pada tahap pengkondisian data.
17. Status aksi sistem sekarang.

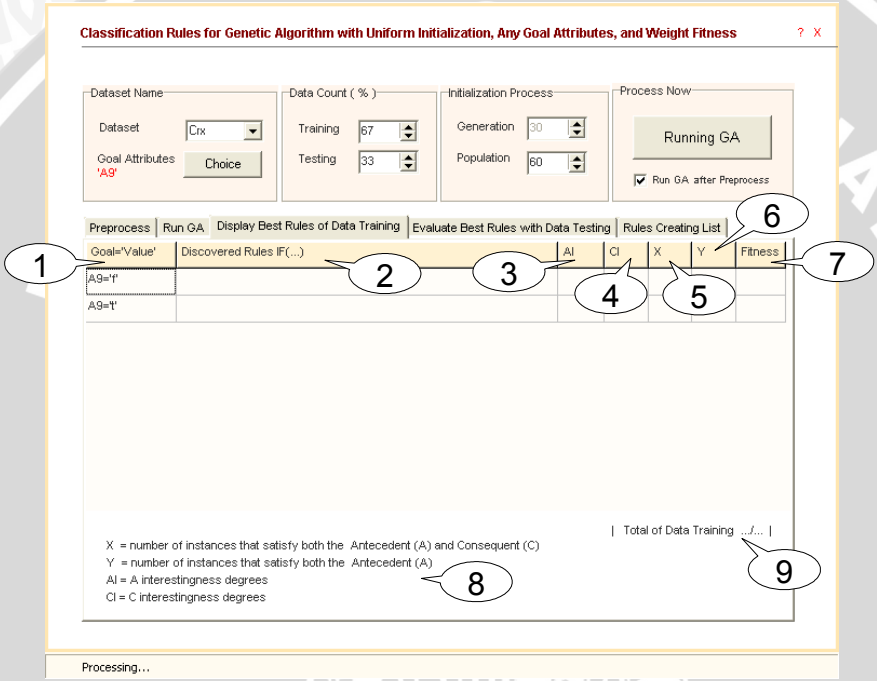
### 3.4.2 Rancangan *Form* Utama Tab Algoritma Genetika




Gambar 3.17 Rancangan *Form* Utama Tab Algoritma Genetika

Pada gambar 3.17 ditampilkan rancangan *form* utama pada tab Algoritma Genetika. Pada gambar 3.17, penunjuk no.1 menunjukkan memo khusus untuk mencetak kerja sistem ketika tahap pengoperasian genetik. Sedang bagian-bagian yang lain telah dijelaskan pada gambar 3.16 sebelumnya.

### 3.4.3 Rancangan *Form* Utama Tab Analisa Aturan



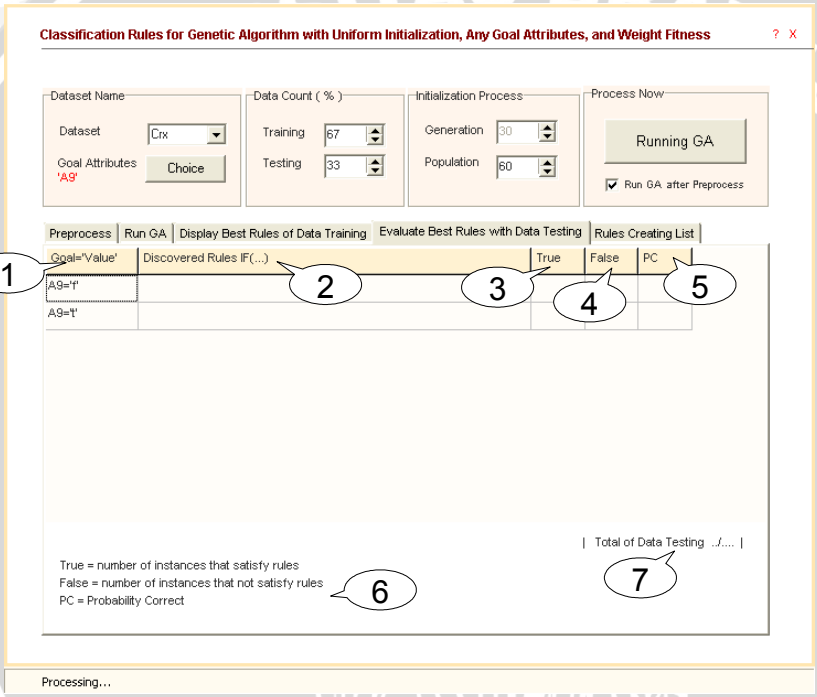

Gambar 3.18 Rancangan *Form* Utama Tab Analisa Aturan

Pada gambar 3.18, ditampilkan rancangan *form* utama pada tab analisa aturan dari data-latih senilai persentase data-latih. Secara terurut, penjelasan nomor-nomor penunjuk pada gambar 3.18 yakni :

1. Kolom mencetak atribut tujuan dan semua jenis nilainya dari data-contoh yang dipilih pengguna.
2. Kolom mencetak aturan *antecedent.*
3. Kolom mencetak nilai kelayakan *antecedent*.
4. Kolom mencetak nilai kelayakan *consequent*.

5. Kolom mencetak jumlah *record* yang memenuhi *antecedent* dan *consequent*.
6. Kolom mencetak jumlah *record* yang memenuhi *antecedent.*
7. Kolom mencetak nilai *fitness* dari aturan.
8. Beberapa baris keterangan tentang fungsi *X*, *Y*, *AI*, dan *CI*.
9. Keterangan persentase data-latih dan total data-latih pada sistem.

### 3.4.4 *Form* Utama Tab Evaluasi Aturan

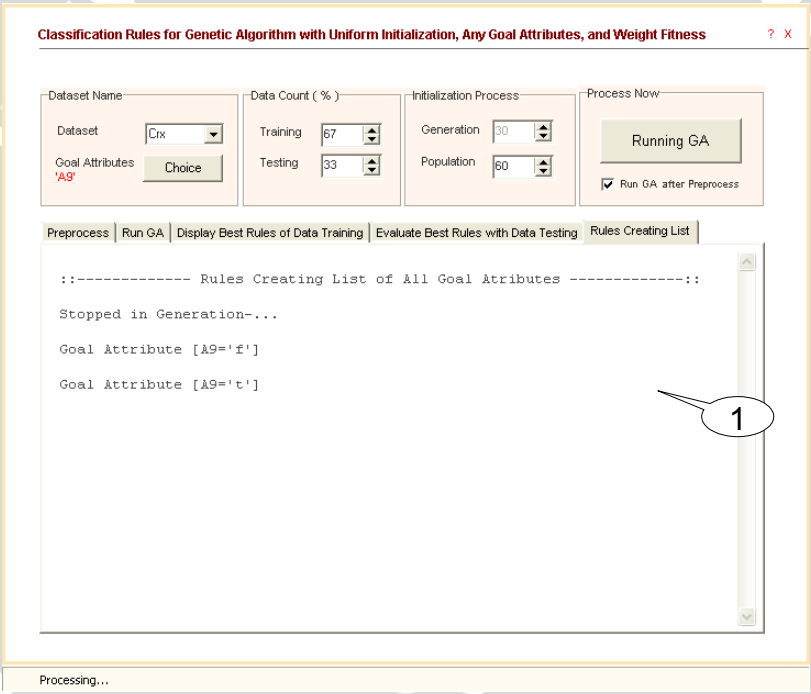



Gambar 3.19 Rancangan *Form* Utama Tab Evaluasi Aturan

Pada gambar 3.19, ditampilkan rancangan form utama tab evaluasi aturan senilai persentase data-uji. Secara terurut, penjelasan nomor-nomor penunjuk pada gambar 3.19 yakni :

1. Kolom mencetak atribut tujuan dan semua jenis nilainya dari data-contoh yang telah dipilih pengguna.
2. Kolom mencetak aturan *antecedent*.
3. Kolom mencetak jumlah data-uji benar dari data-contoh (*True*).
4. Kolom mencetak jumlah data-uji salah per jenis nilai kelas dari data-contoh (*False*).
5. Kolom mencetak persentase *Probability Correct* (*PC*).
6. Beberapa baris keterangan tentang *True*, *False*, dan *PC*.
7. Keterangan persentase data-uji dan total data-uji pada sistem.

### 3.4.5 *Form* Utama Tab Daftar Aturan

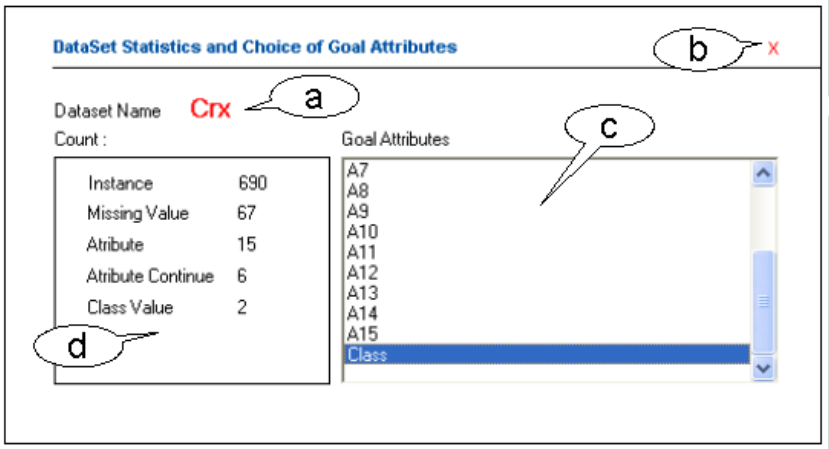



Gambar 3.20 Rancangan *Form* Utama Tab Daftar Aturan

Pada gambar 3.20 ditampilkan rancangan *form* utama pada tab daftar aturan. Pada gambar 3.20 tersebut, penunjuk no.1 menunjukkan memo khusus untuk mencetak aturan-aturan terbaik dan nilai *fitness*-nya untuk semua jenis nilai pada atribut tujuan tertentu.

Sedang bagian-bagian yang lain telah dijelaskan pada gambar 3.16 sebelumnya.

### 3.4.6 *Form* Statistik Data-Contoh




Gambar 3.21 Rancangan Form Statistik Data-Contoh

Penjelasan nomor-nomor penunjuk dari gambar 3.21 yakni :

**a.** Nama data-contoh yang sedang aktif dilihat deskripsinya.
**b.** Tombol untuk menutup *form* statistik.
**c.** Daftar atribut tujuan data-contoh yang hanya bisa dipilih 1.
**d.** Statistik data-contoh yang sedang aktif dilihat deskripsinya.

## 3.5 Perancangan Uji Coba

Sistem yang telah dibuat akan diujicobakan dengan data-contoh yang telah ditentukan yakni *Crx* dengan atribut tujuan *A9* yaitu atribut bertipe diskrit dengan jenis nilai 't' dan 'f'. Sehingga dari data-latih akan didapatkan aturan-aturan untuk atribut tujuan *A9* dengan nilai kelayakan dan keakuratannya masing-masing. Kemudian, akan dilakukan evaluasi terhadap keseluruhan aturan-aturan tersebut menggunakan data-uji.

### 3.5.1 Perancangan Skenario Uji Coba

Penilaian terhadap kelayakan dan keakuratan suatu aturan dengan Algoritma Genetika harus mempertimbangkan banyak parameter dengan nilai-nilainya yang berubah-ubah.

Parameter-parameter yang akan dirancang untuk diujicobakan dalam sistem ini yakni persentase data-latih (rentang antara 60%-100%), persentase data-uji (rentang antara 60%-100%), dan jumlah populasi (kelipatan 10, antara 10-100). Pengujian keseluruhan parameter ini akan dilakukan secara bergantian terhadap masing-masing parameter dengan berbagai variasi nilainya, sementara parameter-parameter lainnya bernilai tetap, dengan rinciannya yakni :

a. Untuk menguji parameter jumlah populasi antara rentang 10-100, maka parameter yang bernilai tetap yakni persentase data-latih senilai 67% dari data-contoh dan persentase data-uji senilai 33% dari data-contoh.
b. Untuk menguji parameter persentase data-latih antara rentang 60%-100%, maka parameter yang bernilai tetap yakni persentase data-uji senilai 10% dari data-contoh; dan jumlah populasi 20.
c. Untuk menguji parameter persentase data-uji antara rentang 60%-100%, maka parameter yang bernilai tetap yakni persentase data-latih senilai 67% dan jumlah populasi 20.

Pengujian masing-masing parameter persentase data-latih, persentase data-uji, atau jumlah populasi dilakukan untuk mengetahui :

a. Pengaruh parameter jumlah populasi terhadap nilai kelayakan a*ntecedent* (*AI*). Pengujian dilakukan 10 kali.
b. Pengaruh parameter jumlah populasi terhadap nilai kelayakan *consequent* (*CI*). Pengujian dilakukan 10 kali.
c. Pengaruh parameter jumlah populasi terhadap nilai prediksi keakuratan (*PA*). Pengujian dilakukan 10 kali.
d. Pengaruh parameter jumlah populasi terhadap nilai kelayakan *fitness*. Pengujian dilakukan 10 kali.
e. Pengaruh parameter persentase data-latih terhadap *Probabilty Correct* (*PC*). Pengujian dilakukan 10 kali.
f. Pengaruh parameter persentase data-uji terhadap *Probabilty Correct* (*PC*). Pengujian dilakukan 10 kali.

Dari rancangan-rancangan pengujian ini nantinya dapat mengevaluasi kelayakan dan keakuratan dari aturan yang dihasilkan oleh sistem.

### 3.5.2 Perancangan Hasil Evaluasi

Untuk mengetahui nilai kelayakan dan keakuratan aturan yakni dengan melakukan pengamatan terhadap nilai kelayakan *antecedent* (*AI*), nilai kelayakan *consequent* (*CI*), keakuratan prediksi (*PA*), nilai *fitness*, dan persentase kebenaran aturan dari data-uji (*PC*).

Tabel-tabel yang dirancang untuk mengevaluasi hasil dari aturan-aturan yang terbentuk dari data-contoh *Crx* dengan atribut tujuan *A9* diantaranya yakni :

1. Tabel 3.25 untuk mencatat pengaruh parameter jumlah populasi terhadap nilai *antecedent* (*AI*).

Tabel 3.25 Pengaruh Jumlah Populasi Terhadap Nilai *AI*

| Populasi | AI | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | Rerata |
| 10 | | | | | | | | | | | |
| 20 | | | | | | | | | | | |
| 30 | | | | | | | | | | | |
| ... | | | | | | | | | | | |
| 100 | | | | | | | | | | | |

2. Tabel 3.26 untuk mencatat pengaruh parameter jumlah populasi terhadap nilai *consequent* (*CI).*

Tabel 3.26 Pengaruh Jumlah Populasi Terhadap Nilai *CI*

| Populasi | CI | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | Rerata |
| 10 | | | | | | | | | | | |
| 20 | | | | | | | | | | | |
| 30 | | | | | | | | | | | |
| ... | | | | | | | | | | | |
| 100 | | | | | | | | | | | |

3. Tabel 3.27 untuk mencatat pengaruh parameter jumlah populasi terhadap nilai keakuratan prediksi (*PA*).

Tabel 3.27 Pengaruh Jumlah Populasi Terhadap Nilai *PA*

| Populasi | PA | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | Rerata |
| 10 | | | | | | | | | | | |
| 20 | | | | | | | | | | | |
| 30 | | | | | | | | | | | |
| ... | | | | | | | | | | | |
| 100 | | | | | | | | | | | |

4. Tabel 3.28 untuk mencatat pengaruh parameter jumlah populasi terhadap nilai *fitness.*

Tabel 3.28 Pengaruh Jumlah Populasi Terhadap Nilai *Fitness*

| Populasi | Fitness | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | Rerata |
| 10 | | | | | | | | | | | |
| 20 | | | | | | | | | | | |
| 30 | | | | | | | | | | | |
| ... | | | | | | | | | | | |
| 100 | | | | | | | | | | | |

5. Tabel 3.29 untuk mencatat pengaruh parameter persentase data-latih terhadap nilai persentase kebenaran aturan dari data-uji (*PC*).

Tabel 3.29 Pengaruh Persentase Data-Latih Terhadap *PC*

| % Data Latih | PC | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | Rerata |
| 60 | | | | | | | | | | | |
| 70 | | | | | | | | | | | |
| 80 | | | | | | | | | | | |
| 90 | | | | | | | | | | | |
| 100 | | | | | | | | | | | |

6. Tabel 3.30 mencatat pengaruh parameter persentase data-uji terhadap nilai persentase kebenaran aturan dari data-uji (*PC*).

Tabel 3.30 Pengaruh Persentase Data-Uji Terhadap *PC*

| % Data Uji | PC | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | Rerata |
| 60 | | | | | | | | | | | |
| 70 | | | | | | | | | | | |
| 80 | | | | | | | | | | | |
| 90 | | | | | | | | | | | |
| 100 | | | | | | | | | | | |

7. Tabel 3.31 untuk mencatat contoh 20 aturan terbaik dari data-contoh *Crx* untuk semua jenis nilai atribut tujuan *A9* yakni 't', 'f'.

Tabel 3.31 Aturan-aturan Data Contoh *Crx*

| Nilai | No. | Aturan | *Fitness* |
|---|---|---|---|
| *'t'* | 1 | | |
| | 2 | | |
| | ... | | |
| | 20 | | |
| *'f'* | 1 | | |
| | 2 | | |
| | ... | | |
| | 20 | | |

# BAB IV
# IMPLEMENTASI DAN PEMBAHASAN

## 4.1 Lingkungan Implementasi

Lingkungan implementasi yang akan dijelaskan dalam subbab ini adalah lingkungan implementasi perangkat keras dan perangkat lunak yang diterapkan pada sistem klasifikasi aturan *mining* dengan Algoritma Genetika sesuai rumusan masalah dalam skripsi.

### 4.1.1 Lingkungan Perangkat Keras

Perangkat keras yang digunakan untuk membangun sistem klasifikasi aturan *mining* dengan Algoritma Genetika ini yakni :

1. Prosesor Intel Pentium4 1,4 GHz
2. Memori SD Ram 256 MB
3. Harddisk berkapasitas 80 GB
4. Monitor 17”
5. Keyboard
6. Mouse

### 4.1.2 Lingkungan Perangkat Lunak

Perangkat lunak yang digunakan untuk membangun sistem klasifikasi aturan *mining* dengan Algoritma Genetika ini yakni :

1. Sistem operasi menggunakan Microsoft Windows XP.
2. Sistem dibangun dengan Borland Delphi Enterprise 6.0 dengan sistem manajemen databasenya Microsoft Access XP.

## 4.2 Implementasi Program

Pada subbab implementasi program ini akan dijelaskan tentang implementasi dari masing-masing rancangan proses sistem yang telah dibahas sebelumnya pada subbab 3.2.

### 4.2.1 Implementasi Pengkondisian Data-Contoh

Pada implementasi bagian proses pengkondisian data terdiri dari proses pengisian nilai hilang yang ditemukan pada data-contoh dan proses pendiskritan nilai kontinu.

#### 4.2.1.1 Mengisi Nilai Hilang

Tahapan pertama dalam pengkondisian data-contoh yakni mengisi nilai hilang yang telah ditemukan pada data-contoh, menggunakan metode Klasifikasi Naïve Bayesian (persamaan 2.3). Fungsi yang menunjukkan penggunaan metode Klasifikasi Naïve Bayesian untuk mengisi nilai hilang pada data-contoh ditunjukkan pada gambar 4.1.

```
35  function TPreprocess.Bayess(pos:byte; j,nRec:word):string;
36  var Max,PCX,PXC,PC : single;    nPerDist : word;
37       i : byte;                  Q : string;
38  begin
39   Max := 0;
40   for i:=0 to Attrib[pos].nDist-1 do
41   begin
42     PC := CalcPC(i,pos,nPerDist,nRec);  // Hitung PC
43     { Hitung PCX }
44     (*  Mencari Bayess berdasar 'CLASS' *)
45     Q := 'SELECT * FROM '+DataSet_+' WHERE Class'+'='
46             +''''+ DataBase[ IDClass,j ] + '''';
47     Q := Q + ' AND '+Attrib[pos].name+'='+
48             '''+ Distinct[pos,i].Value +'''';
49     DB.Query2(Q);
50  PCX := Dmodule1.DataSource2.DataSet.RecordCount/nPerDist;
51     { Hitung PXC }
52     PXC := PC * PCX;
53     if PXC >= Max then
54     begin
55       Max := PXC;
56       Result := Distinct[pos,i].Value;
57     end;
58    end;
58  end;
```

Gambar 4.1 Fungsi Mengisi Nilai Hilang

Fungsi mengisi nilai hilang dengan metode Klasifikasi Naïve Bayesian diimplementasikan dalam fungsi `Bayess`. Fungsi `Bayess` berfungsi untuk mengisi nilai hilang pada posisi baris data tertentu dan atribut tertentu. Fungsi `Bayess` memiliki parameter yakni :

a. `Pos` bertipe `byte` berfungsi untuk mendefinisikan no.atribut yang akan diterapkan pengisian nilai hilang.

b. `j` bertipe `word` berfungsi untuk mendefinisikan no.baris-data yang akan diterapkan pengisian nilai hilang.
c. `nRec` bertipe `word` berfungsi untuk mendefinisikan jumlah baris data dari data-contoh.
d. Keluaran dari fungsi `Bayess` bertipe `string` yang berisi inisialisasi nilai hilang untuk atribut dan baris data tertentu.

Pada gambar 4.1 tepatnya pada baris 40-58 dijelaskan tentang implementasi metode Klasifikasi Naïve Bayesian. Pada baris 40 ditunjukkan tentang pencarian inisialisasi terbaik untuk mengisi nilai hilang dari semua jenis nilai dari atribut Kelas. Sehingga nilai terbaik yang bisa mengisi nilai hilang dari atribut dan baris tertentu ialah jenis nilai dari atribut Kelas yang memiliki peluang lebih banyak pada data contoh *Crx*, dan penjelasan ini ditunjukkan pada pada baris 53-57.

#### 4.2.1.2 Pendiskritan Nilai Kontinu

Tahapan kedua dalam pengkondisian data-contoh yakni pendiskritan nilai kontinu. Proses pendiskritan ini terjadi pada atribut-atribut yang telah ditentukan dalam deskripsi data-contoh (bisa dilihat dalam lampiran) sebagai atribut bertipe kontinu. Hasil akhir dari pendiskritan nilai kontinu yang diterapkan dalam sistem yakni 3 jenis nilai diskrit yakni 'low', 'med', dan 'high'.

Pendiskritan nilai kontinu dalam sistem menggunakan Teknik *Chimerge,* yakni algoritma pendiskritan dengan menganalisa kualitas *interval* dengan parameter statistik $\chi^2$. Prosedur dari teknik *Chimerge* ini diimplementasikan pada prosedur `CHIMERGE` seperti yang ditunjukkan pada gambar 4.2. Prosedur `CHIMERGE` ini berfungsi untuk mendiskritkan semua atribut yang bertipe kontinu, tetapi dalam kerjanya prosedur `CHIMERGE` mendiskritkan atribut kontinu satu per-satu atribut.

```
105  procedure Tpreprocess.CHIMERGE(Temp:TArClass;
106        var Intv_:TArIntrvl; var B:array of Tinterval);
107  var L,Start: word;
108      i: byte;
109      Max,x2 : single;
110      Div2_ : TArFlt;
111  begin
112    ContinueSort(Temp,0,nRecord-1);  // Urut nilai cont
113    FuncDiv2(nRecord-1,Temp,Div2_); // Hit nilai bagi 2
114    Start := 0;
115    { Mencari 2 grup interval }
116    for i:=0 to 1 do
117    begin
118     PosIntrvl := 0;     L := 0;       Max := 0;
119     { Hit selisih interval }
120     DiffInterval(nRecord-1,Start,Intv_,Div2_);
121     while L<nRecrd do   // mengurangi komputasi
122     begin
123      x2 := ContingencyTabel(nRecord-1,L,Temp
124            ,Intv_[PosIntrvl]);
125      inc(PosIntrvl);
126      if (x2>Max)and(x2<=ThresHold) then
127      Begin      // ThresHold ditentukan pengguna=5
128       Max := x2;
129       B[i].d1 := Intv_[PosIntrvl].intv[0].d1;
130       B[i].d2 := Intv_[PosIntrvl].intv[1].d2;
131       Start := Intv_[PosIntrvl].EndPos;
132      end;
133      inc(L);
134     end;
135    end;
136    { Mencari grup ke-3 }
137    B[2].d1 := B[1].d2;
138    B[2].d2 := Temp[nRecord-1].value;
139  end;
```

Gambar 4.2 Prosedur Pendiskritan Nilai Kontinu

Prosedur `CHIMERGE` ini memiliki parameter yakni :

a. `Temp` bertipe `TArClass` berfungsi untuk mendefinisikan data-data bayangan dari atribut tertentu yang bertipe kontinu yang hendak dilakukan proses pendiskritan nilai kontinu. Data bayangan

`Temp` ini digunakan untuk proses pengurutan menaik dari data-data kontinu yakni pada prosedur `ContinueSort` (gambar 4.2 baris 112). Setelah dilakukan pengurutan menaik, dilanjutkan dengan penghitungan nilai bagi 2 dari data-data kontinu tersebut yakni pada prosedur `FuncDiv2` (gambar 4.2 baris 113). Konsep dari pengurutan menaik dan penghitungan nilai bagi 2 ini telah dijelaskan sebelumnya dalam subbab 3.2.2.2.

b. Variabel `Intv_` bertipe `TArIntrvl` berfungsi untuk menyimpan *interval-interval* dari semua data kontinu pada atribut yang sedang didiskritkan yakni pada prosedur `DiffInterval` (gambar 4.2 baris 120). Nilai *interval-interval* ini nantinya digunakan ketika mencari $\chi^2$ terbaik pada tabel *Contingency* yakni pada prosedur `ContingencyTabel` (gambar 4.2 baris 123-124), sehingga dari $\chi^2$ terbaik ini ditemukan *interval-interval* terbaik.

c. Variabel `B` bertipe `array of Tinterval` berfungsi untuk menyimpan *interval-interval* terbaik yang dijadikan sebagai hasil akhir dari Teknik *Chimerge*. Dikatakan *interval* terbaik apabila ditemukan dua-*interval* yang saling berdekatan dan paling mendekati *threshold.* Nantinya semua data kontinu pada atribut yang sedang didiskritkan akan dipetakan menurut *interval-interval* terbaik ini. Langkah-langkah menyimpan interval-interval terbaik pada variabel B ditunjukkan pada gambar 4.2 baris 126-138.

### 4.2.2 Implementasi Algoritma Genetika

Setelah dilakukan proses pengkondisian data pada data-contoh, berikutnya dilakukan proses Algoritma Genetika untuk mencari aturan serta menghitung kelayakan dan keakuratan aturan yang dihasilkan tersebut dari data-contoh. Pada subbab ini akan dijelaskan tentang implementasi dari rancangan proses Algoritma Genetika yakni Pengkodean *Binary*, representasi kromosom, pembangkitan populasi terstruktur, operasi genetika (Seleksi Roda *Roullette*, persilangan terstruktur, mutasi) dan perhitungan nilai *fitness*.

#### 4.2.2.1 Pengkodean *Binary*

Untuk memulai tahapan operasi genetika pada sistem, data-contoh harus disesuaikan dengan masukan pada Algoritma Genetika, yaitu

data-contoh harus dikodekan menjadi *binary,* sehingga hasil yang diinginkan dari Algoritma Genetika bisa maksimal. Implementasi dari rancangan pengkodean datacontoh menjadi binary ditunjukkan pada gambar 4.3.

```
144  procedure EncodeDistinct;
145  var i,j,lnRange : byte;
146  begin
147   for i:=0 to Pre.nAttrib-1 do
148   begin
149    // rumus panjang range bit
150    lnRange:=Round(Int(log2(Pre.Attrib[i].nDist)+1));
151    for j:=0 to Pre.Attrib[i].nDist-1 do
152    { mis.00000000000000000000000101 -> 0101}
153    Pre.Distinct[i,j].bit:=Copy(InttoBin(j+1),
154                           32-lnRange+1,lnRange);
155   end;
156  end;
```

Gambar 4.3 Prosedur Pengkodean *Binary*

Implementasi dari Pengkodean *Binary* yakni pada prosedur `EncodeDistinct`. Prosedur `EncodeDistinct` ini berfungsi untuk mengkodekan ke dalam *binary* semua jenis nilai dari semua atribut data-contoh yang kemudian disimpan di *array Distinct*, seperti ditunjukkan pada gambar 4.3 baris 147-155. Prosedur `EncodeDistinct` ini tidak memiliki parameter di dalamnya.

#### 4.2.2.2 Representasi Kromosom

Solusi masalah yang diinginkan dari Algoritma Genetika perlu untuk direpresentasikan dalam bentuk kromosom. Sehingga kesesuaian representasi kromosom terhadap rumusan masalah dalam skripsi akan sangat mempengaruhi kualitas solusi dari masalah. Implementasi dari rancangan representasi kromosom ke dalam sistem ditunjukkan pada gambar 4.4 yakni pada fungsi `InitRandom1thPop`.

```
117  function Tgenetic.InitRandom1thChrom:string;
118   function GetBit(idAtt:byte; value:string):string;
119   var i : byte;
120   begin
121   for i:=0 to Pre.Attrib[idAtt].nDist-1 do
122    if value = Pre.Distinct[idAtt,i].Value then
123    begin
124     result := Pre.Distinct[idAtt,i].Bit;
125     exit;
126    end;
127   end;
128
129  var i : byte;
130      idRecord : word;
131      value : string;
132  begin
133   Result := '';
134   for i:=0 to nAttrib-1 do
135   begin
136    idRecord := Random(nRecTrain);
137    Q := 'SELECT * FROM PreDB WHERE ID='
138        +IntToStr(idRecord);
139    DB.Query1(Q);
140    value :=  Dmodule1.DataSource1.DataSet.
141       Fields.FieldByName(Pre.Attrib[i].name).AsString;
142    { diisi 'f=1', 'r=1', 'v=bit hasil encoding }
143    Result := Result + '11'+ GetBit(i,value);
144   end;
145  end;
```

Gambar 4.4 Prosedur Representasi Kromosom

Fungsi `InitRandom1thPop` ini sebagai fungsi yang digunakan untuk merepresentasikan kromosom awal. Kromosom awal ini tersusun dari sejumlah atribut-atribut yang dimiliki data-contoh, seperti yang ditunjukkan pada gambar 4.4 baris 134-144.

Seperti telah dijelaskan sebelumnya pada subbab 2.2.4.2, bahwa satu gen (sama artinya dengan satu atribut) tersusun dari *flag-operator-value*. Pada gambar 4.4 baris 143, tampak bahwa kromosom awal disusun dari *flag* yang bernilai '1' artinya atribut tersebut terdapat pada aturan, *operator* yang bernilai '1' artinya menggunakan *operator* '=', dan *value* yang didapat dari fungsi `GetBit` yang menghasilkan nilai dari

atribut tersebut yang telah dikodekan.

Fungsi `InitRandom1thPop` ini tidak memiliki parameter di dalamnya. Sedang fungsi `GetBit` memiliki parameter `idAtt` bertipe `byte`, sebagai identifikasi no.atribut berisi nilai *value* yang akan dicarikan kode bit dari nilai *value* tersebut, dan parameter `value` bertipe `string`, berisi nilai asli yang akan dicarikan kode bit-nya. Dan fungsi `GetBit` ini mengembalikan *value* bertipe `string` yang telah dikodekan.

### 4.2.2.3 Pembangkitan Populasi Awal Terstruktur

Untuk mendapatkan solusi dari Algoritma Genetika maka perlu dibangkitkan kromosom-kromosom yang dikelompokkan menjadi sebuah populasi. Perlu dibangkitkan populasi awal untuk membentuk populasi-populasi turunan dengan harapan menghasilkan nilai *fitness* pada kromosom-kromosomnya yang lebih baik dari kromosom-kromosom pada populasi awal.

```
160  procedure Tgenetic.InitUniform(IDGoal:byte);
161  var r : word;
162      Chromosome : string;
163  begin
...   ...
166   r:=Round(Int(log2(nPop))); // Hit r dari 2^r <= |P|
167   InitModelUnif(r,Model); // buat model uniform
168   nInitPop := 0;
169   while nInitPop<nPop do
170   begin
171    Chromosome := InitRandom1thChrom;
172    InitPop(Chromosome); //inisialisasi pop awal unif
173   end;
...  ...
177  end;
```

Gambar 4.5 Prosedur Pembangkitan Populasi Awal Terstruktur

Pembangkitan populasi awal dalam sistem ini dikerjakan dengan terstruktur artinya dirancang suatu model struktur populasi sebagai acuan dalam pembangkitan populasi awal sehingga pembangkitan populasi secara awal ini bisa terhindar dari pembentukan kromosom

berkisar pada daerah solusi lokal atau jauh dari daerah solusi. Implementasi dari rancangan proses pembangkitan populasi awal secara terstruktur ini ditunjukkan pada gambar 4.5 yakni pada prosedur `InitUniform`.

Di dalam prosedur `InitUniform` ini akan mengeksekusi dua prosedur lain yakni `InitModelUnif` (gambar 4.5 baris 167) dan `InitPop` (gambar 4.5 baris 172). Kedua prosedur ini merupakan bagian dari sistem untuk pembangkitan populasi awal terstruktur.

Prosedur `InitModelUnif` berfungsi untuk merancang model struktur dari populasi awal, sehingga pembangkitan populasi awal akan mengikuti model yang telah dibangun pada prosedur ini. Untuk lebih jelasnya, prosedur `InitModelUnif` ini ditunjukkan pada gambar 4.6.

Sedang prosedur `InitPop` sebagai prosedur pembangkitan kromosom untuk populasi awal terstruktur. Prosedur `InitPop` ini ditunjukkan pada gambar 4.7.

```
58  procedure InitModelUnif(r:word;
59                          var Model:array of string);
60  var x : string;   i : word;
61  begin
62   if r=2 then begin
63    Model[nModelUnif] := '-0';  inc(nModelUnif);
64    Model[nModelUnif] := '0-';  inc(nModelUnif);
65    Model[nModelUnif] := '0';   inc(nModelUnif);
65    Model[nModelUnif] := '-';
66    exit;
67   end;
68   x := '-';
69   for i:=1 to r-1 do   x := x + '0';
70   PermutationStr(x,Model,1,r);
71   for i:=0 to r-3 do // r-3 = jml pop unif dlm 1 gen
72   begin
73     x[r-i] := '-';
74     PermutationStr(x,Model,1,r);
75   end;
76   InitModelUnif(r-1,Model);  // rekursif
77  end;
```

Gambar 4.6 Prosedur Perancangan Model Populasi Terstruktur

Seperti telah dijelaskan sebelumnya bahwa prosedur `Init-ModelUnif` ini berfungsi untuk merancang model struktur dari populasi awal. Seperti yang ditunjukkan pada gambar 4.6, prosedur `InitModelUnif` memiliki parameter - parameter diantaranya :

a. `r` bertipe `byte` berfungsi untuk nilai rekursif dari r (panjang maksimal model).
b. Variabel `Model` bertipe `array of string` berfungsi untuk menyimpan model struktur dari populasi.

Prosedur `InitModelUnif` bersifat *rekursif*, sehingga prosedur ini akan terus dieksekusi sampai kondisi berhenti yakni jika *r* atau jumlah maksimum modelnya telah mencapai nilai minimal yakni 2, seperti pada gambar 4.6 baris 62-67. Sedang prosedur *rekursif* lainnya yang berada di dalam prosedur `InitModelUnif` yakni prosedur `PermutationStr` (gambar 4.6 baris 70 dan 74), berfungsi untuk mencari semua kemungkinan model struktur populasi yang terbentuk dengan nilai *r* tertentu.

Sedang untuk prosedur `InitPop` seperti yang tampak pada gambar 4.7 memiliki parameter `Chr_` bertipe `string` yang berfungsi untuk sebagai masukan kromosom yang telah dibangkitkan pertama kali untuk membentuk kromosom-kromosom berikutnya dengan maksud untuk memenuhi sejumlah kromosom pada populasi awal. Pada gambar 4.7 baris 95-100 ditunjukkan tentang pembentukan kromosom-kromosom mengikuti model struktur populasi yang telah dirancang pada prosedur `InitModelUnif`. Dan pada gambar 4.7 baris 101-107 merupakan pendefinisian kromosom-kromosom sekaligus perhitungan nilai *fitness-fitness*-nya untuk membentuk sebuah populasi awal terstruktur dengan menyertakan persyaratan antisipatif tidak boleh melebihi jumlah kromosom dalam sebuah populasi.

```
 86  procedure TGenetic.InitPop(Chr_:string);
 87  var i,j,m,k : word;
 88      n : string;
 89  begin
 90   for i:=0 to nModelUnif do
 91   begin
 92   k := 1;   n := '';
 93   for j:=0 to length(Model[0])-1 do
 94    for m:=1 to Round(length(Chr_)/length(Model[i]))+1
 95     if k<=length(Chr_) then
 96     begin
 97      if copy(Model[i],j+1,1)='0' then n := n + Chr_[k]
 98      else n := n + Invers(Chr_[k]);
 99      inc(k);
100     end;
101     if nInitPop<nPop then
102     begin
103      Chrom[nInitPop].Bit := n;
104      CalcFitness(DecodeAntecedent(Chrom[nInitPop]),
105                  Chrom[nInitPop]);
106      inc(nInitPop);
107     end
108     else exit;
109   end;
109  end;
```

Gambar 4.7 Prosedur Pembangkitan Kromosom untuk Populasi Awal Terstruktur

#### 4.2.2.4 Operasi Genetika

Setelah dibangkitkan sebuah populasi awal dengan sejumlah kromosomnya, langkah selanjutnya yaitu melakukan proses operasi genetika sebagaimana telah dirancang sebelumnya pada subbab 3.2.3. Operator Algoritma Genetika yang diterapkan dalam sistem ini dipilih menurut keoptimalannya untuk menghasilkan solusi berupa aturan *mining* yang layak dan akurat. Maka operator-operator yang dipakai dalam sistem ini diantaranya Seleksi Roda *Roullette*, Persilangan Terstruktur, dan Mutasi. Pada subbab ini akan dijelaskan implementasi dari ketiga operator tersebut.

#### 4.2.2.4.1 Seleksi Roda *Roullette*

Operator Algoritma Genetika yang pertama digunakan dalam sistem ini yakni Seleksi Roda *Roullette*. Operator ini menyeleksi dari sekian banyak kromosom untuk dipilih antara kromosom yang bernilai *fitness* tinggi dengan yang rendah. Kromosom yang bernilai *fitness* tinggi berkemungkinan besar untuk dipilih daripada kromosom dengan nilai *fitness* yang rendah. Implementasi dari rancangan Seleksi Roda *Roullette* ditunjukkan pada gambar 4.8 yakni prosedur `GAOperate`.

```
21  procedure TGenetic.GAOperate(nPop:byte);
22  var i,p1,p2 : word;
23      ProbCum : array [0..100] of single;
24    procedure CreateProbCum;
25    var i : word;  TotProbCum : single;
26    begin
27      TotProbCum := 0;    ProbCum[0] := 0;
28      for i:=0 to nPop-1 do
29      begin
30        TotProbCum := TotProbCum + Chrom[i].Fit;
31        ProbCum[i] := ProbCum[i-1] + Chrom[i].Fit;
32      end;
33      for i:=0 to nPop-1 do
34        ProbCum[i] := ProbCum[i] / TotProbCum;
35    end;
36    function GetParent(nPop:byte):byte;
37    var r:single; i:word;
38    begin
39      r := Random;    i:= 0;
40      while (i<=nPop-1) and (r>ProbCum[i]) do Inc(i);
41      result := i;
42    end;
...   .......
72  begin
73        CreateProbCum;
74        for i:=0 to nPop-1 do
75        begin
76          p1 := GetParent(nPop);
77          p2 := GetParent(nPop);
78          .......
79        end;
80  end;
```

Gambar 4.8 Prosedur Seleksi Roda *Roullette*

Prosedur `GAOperate` ini memiliki parameter `nPop` yang bertipe `byte`, berfungsi untuk mendefinisikan jumlah populasi yang diterapkan dalam sistem. Untuk proses Seleksi Roda *Roullette* pada prosedur `GAOperate`, yakni ditunjukkan pada gambar 4.8 baris 73-77. Pada baris 73-77, eksekusi prosedur `CreateProbCum` (gambar 4.8 baris 73) dan fungsi `GetParent` (gambar 4.8 baris 76-77).

Untuk prosedur `CreateProbCum` berfungsi untuk menghitung nilai probabilitas dan probabilitas kumulatif terhadap *fitness* masing-masing kromosom. Sedang fungsi `GetParent` dengan parameter `nPop` bertipe `byte` berfungsi untuk mendapatkan kromosom induk yang telah dilakukan proses Seleksi Roda *Roullette*. Fungsi `GetParent` ini mengembalikan nilai bertipe `byte` berupa no.id kromosom induk pada populasi awal.

#### 4.2.2.4.2 Persilangan Terstruktur

Operator Algoritma Genetika yang kedua yang digunakan dalam sistem ini yakni Persilangan Terstruktur. Persilangan ini dipilih dari banyak persilangan yang ada dikarenakan persilangan ini mampu menghasilkan semua skema kromosom hasil dari persilangan antar 2 kromosom. Pada sistem ini selalu terjadi Persilangan Terstruktur antara kromosom-kromosomnya karena tidak adanya batasan kemungkinan terjadinya persilangan. Implementasi dari rancangan proses Persilangan Terstruktur yakni pada prosedur `UnifCrossOverRUN`, gambar 4.9.

Prosedur UnifCrossOverRUN memiliki parameter `p1` bertipe `string` yang berfungsi sebagai pendefinisian dari kromosom induk ke-1 yang akan disilangkan, `p2` bertipe `string` berfungsi sebagai pendefinisian kromosom induk ke-2 yang akan disilangkan. Prosedur ini mengembalikan nilai bertipe `TChr` yaitu sebuah kromosom turunan hasil Persilangan Terstruktur.

Pada gambar 4.9 baris 133, dieksekusi prosedur `Uniform-CrossOver` dengan parameter `C1`, `C2`, `C3`, dan `C4` bertipe `string` untuk mengembalikan nilai berupa 4 kromosom yang terbentuk dari Persilangan Terstruktur. Pada gambar 4.9 baris 134-136, ditunjukkan tentang pemilihan sebuah kromosom acak dari 4 kromosom yang terbentuk dari Persilangan Terstruktur.

```
 93  function TGenetic.UnifCrossOverRUN(p1,p2:string):TChr
 94    procedure UniformCrossOver(var C1,C2,C3,C4:string);
 95    var i,j,nPos : byte;
 96        Bit : string[100];
 97        C1_,C2_,C3_,C4_ : array [0..100] of string;
 98        PosCross : array [0..100] of byte;
 99    begin
100    Bit := '';                 nPos := 0;
101    C1 := '';  C2 := '';       C3 := '';  C4 := '';
102    for i:=0 to ChromLen-1 do
103    //  Menyimpan Kesamaan 2 Chrom
104     if p1[i+1]=p2[i+1] then begin
105       C1_[i] := p1[i+1];    C2_[i] := p1[i+1];
106       C3_[i] := p1[i+1];    C4_[i] := p1[i+1];
107     end
108     else begin // bila tdk sama, maka disimpan pos-nya
109       PosCross[nPos] := i;
110       inc(nPos);
111     end;
112    for i:=1 to nPos do   // Menyusun Bit Random 0&1
113      Bit := Bit + IntToStr(Random(2));
114    for i:=1 to nPos do begin // isi random bit&invers
115      j := PosCross[i-1];
116      if i<Round(nPos/2) then begin
117       C1_[j] := Bit[i];        C2_[j] := Bit[i];
118       C3_[j]:=Invers(Bit[i]);C4_[j]:=Invers(Bit[i]);
119      end
120      else begin
121       C1_[j] := Bit[i]; C2_[j] := Invers(Bit[i]);
122       C3_[j] := Bit[i]; C4_[j] := Invers(Bit[i]);
123      end
124     end;
125    for i:=0 to ChromLen-1 do begin // Susun dr Array
126     C1 := C1 + C1_[i];     C2 := C2 + C2_[i];
127     C3 := C3 + C3_[i];     C4 := C4 + C4_[i];
128    end;
129    end;
130  var y : array[0..3] of string;
131      C : Tchrom;     x : byte;
132  begin
133    UniformCrossOver(y[0],y[1],y[2],y[3]);
134    x := random(4);  // 4 child hasil dipilih acak 1
135    C.Bit := y[x];
136    result := C;
137  end;
```

Gambar 4.9 Prosedur Persilangan Terstruktur

#### 4.2.2.4.3 Mutasi Bit, Gen, atau Kromosom

Operator Algoritma Genetika yang ketiga atau yang terakhir digunakan dalam sistem ini yakni mutasi. Bila kromosom terkena mutasi maka bit yang bernilai '0' menjadi '1' dan bit yang bernilai '1' menjadi '0'. Terdapat 3 jenis mutasi terhadap suatu kromosom, yaitu mutasi terhadap bit, gen, atau kromosom. Yang diterapkan dalam sistem ini yaitu ketiga jenis mutasi tersebut dengan penggunaan ketiga jenis mutasi tersebut secara acak. Kemungkinan terjadinya mutasi telah ditentukan senilai 0,8. Semisal telah dilakukan pengambilan nilai acak dari 0-1, kemudian didapatkan nilai acak 0,7. Karena 0,7 ≤ 0,8 maka akan dilakukan proses mutasi.

Implementasi dari rancangan proses mutasi ditunjukkan pada gambar 4.10 yakni pada prosedur `MutationRUN`.

Prosedur MutationRUN memiliki parameter variabel `Chr_` bertipe `TChrom` berupa kromosom yang akan dilakukan mutasi dan mengembalikan kromosom hasil mutasi juga pada variabel `Chr_` bertipe `TChrom` tersebut.

Untuk selanjutnya, akan dijelaskan beberapa baris *source code* pada gambar 4.10 mengenai prosedure `MutationRUN`. Pada gambar 4.10 baris 179-183 ditunjukkan tentang penggunaan operasi mutasi untuk bit, gen, atau kromosom secara acak. Dilakukan pengacakan angka 0 sampai 2 (seperti pada gambar 4.10 baris 179-183), bila didapatkan angka 0 maka dilakukan mutasi kromosom (seperti pada gambar 4.10 baris 155-158); bila didapatkan angka 1 maka dilakukan mutasi gen (seperti pada gambar 4.10 baris 159-171); dan bila didapatkan angka 2 maka dilakukan mutasi bit (seperti pada gambar 4.10 baris 172-177).

```
144  procedure TGenetic.MutationRUN(var Chr_:TChrom);
145    function Mutate(a,b:byte):string;
146    var i : byte;
147    begin
148     result := '';
149     for i:=1 to ChromLen do
150      if (i>=a)and(i<=b) then
151       result:=result+Invers(Chr_.Bit[i])
152      else
153       result := result + Chr_.Bit[i];
154     end;
155    procedure ChromMutation;
156    begin
157        Chr_.Bit := Mutate(1,ChromLen);
158    end;
159    procedure GenMutation;
160    var L,R,tmp : byte;
161    begin
162        L := Random(ChromLen)+1;
163        R := Rnd(L,ChromLen)+1;
164        if L>R then
165        begin
166          tmp := R;
167          R := L;
168          L := tmp;
169        end;
170        Chr_.Bit := Mutate(L,R);
171    end;
172    procedure BitMutation;
173    var no : byte;
174    begin
175        no := random(ChromLen)+1;
176        Chr_.Bit := Mutate(no,no);
177    end;
178  begin
179       case Random(3) of
180         0 : ChromMutation;
181         1 : GenMutation;
182         2 : BitMutation;
183       end;
184  end;
```

Gambar 4.10 Prosedur Mutasi Bit, Gen, atau Kromosom

#### 4.2.2.5 Perhitungan Nilai Kelayakan dan Keakuratan Aturan

Dari perhitungan nilai kelayakan *antecedent-consequent* dan keakuratan terhadap aturan, akan menghasilkan nilai *fitness* Algoritma Genetika dari tiap-tiap kromosom (representasi dari aturan). Dengan kata lain, perhitungan nilai *fitness* dilakukan setelah tahapan perhitungan nilai kelayakan *antecedent-consequent* dan tahapan perhitungan nilai keakuratan aturan. Tiap-tiap rancangan proses perhitungan pada fungsi *fitness* akan dijelaskan dalam gambar 4.11 yakni pada prosedur `FuncFitness`.

```
129  procedure Tgenetic.FuncFitness(Antecedent:string;
130                        var Chr_:TChrom);
...    .......
148  var dom,n : byte;
149      X,Y : word;
150      InfoGain,InfoGk,AI,CI,PA,Fit : single;
151      Ant : string;
152  begin
153    Y:=CalcSatisfyY;  //Y=jumlah rec yg memenuhi A
154    X:=CalcSatisfyX;  //X=jumlah rec yg memenuhi A & C
155      if (X=0)or(Y=0) then
156      begin
157        Chr_.Ant := '';
158        Chr_.Fit := 0;
159        exit;
160      end;
161      PA := (X-1/2)/Y;
162      // dom = jmlh kemungkinan nilai consequent
163      dom := Pre.Attrib[IDGoal_].nDist;
164      InfoGk := FuncInfoGk;          // hitung Info(Gk)
165      // n = jumlah atribut antecedent dg flag='1'
166      InfoGain := FuncInfoGain(InfoGk,Chr_,n);
167      // log2 utk normalisasi AI=> 0-1
168      AI := 1 - (InfoGain/(n+1)/log2(abs(dom))); // AI
169      CI := power(1-funcPrGkl,1/beta);           // CI
170      Fit := ((w1*(AI+CI)/2)+(w2*PA))/(w1+w2);
...      ...
174      Chr_.Fit := Fit;
175  end;
```

Gambar 4.11 Prosedur Perhitungan Nilai Kelayakan dan Keakuratan Aturan untuk Menghasilkan Nilai *Fitness* Kromosom

Prosedur `FuncFitness` ini memiliki parameter `Antecedent` bertipe `string` untuk mendefinisikan *antecedent* dari aturan sebagai hasil penerjemahan dari kromosom, dan parameter variabel `Chr_` bertipe `TChrom` untuk mendefinisikan krosomosom yang akan dihitung nilai *fitness*-nya.

Pada prosedur `FuncFitness` terdapat banyak eksekusi pada prosedur dan fungsi. Diantaranya yaitu :

a. Fungsi `CalcSatisfyY` (gambar 4.11 baris 153)
   Untuk menghitung jumlah baris data-latih yang memenuhi *antecedent* dari aturan. Dan mengembalikan nilainya berupa variabel `Y` bertipe `word`.
b. Fungsi `CalcSatisfyX` (gambar 4.11 baris 154)
   Untuk menghitung jumlah baris data-latih yang memenuhi *antecedent* dan *consequent* dari aturan. Dan mengembalikan nilainya berupa variabel `X` bertipe `word`.
c. Fungsi `FuncInfoGk` (gambar 4.11 baris 164)
   Untuk menghitung *info(Gk)* dan mengembalikan nilainya berupa variabel `InfoGk` bertipe `single`.
d. Fungsi `FuncInfoGain` (gambar 4.11 baris 166)
   Untuk menghitung *infoGain* dan mengembalikan nilainya berupa variabel `InfoGain` bertipe `single`. Fungsi `FuncInfoGain` ini memiliki parameter `InfoGk` bertipe `single`, `Chr_` bertipe `TChr`, dan `n` bertipe `byte`.

Sedang untuk menghitung nilai *PA* seperti ditunjukkan pada gambar 4.11 baris 161, untuk menghitung nilai *AI* ditunjukkan pada gambar 4.11 baris 168, menghitung nilai *CI* ditunjukkan pada gambar 4.11 baris 169, untuk menghitung nilai *fitness* ditunjukkan pada gambar 4.11 baris 170, dan pada gambar 4.11 baris 174 ditunjukkan tentang pendefinisian nilai *fitness* yang telah dihitung untuk kromosom `Chr_`.

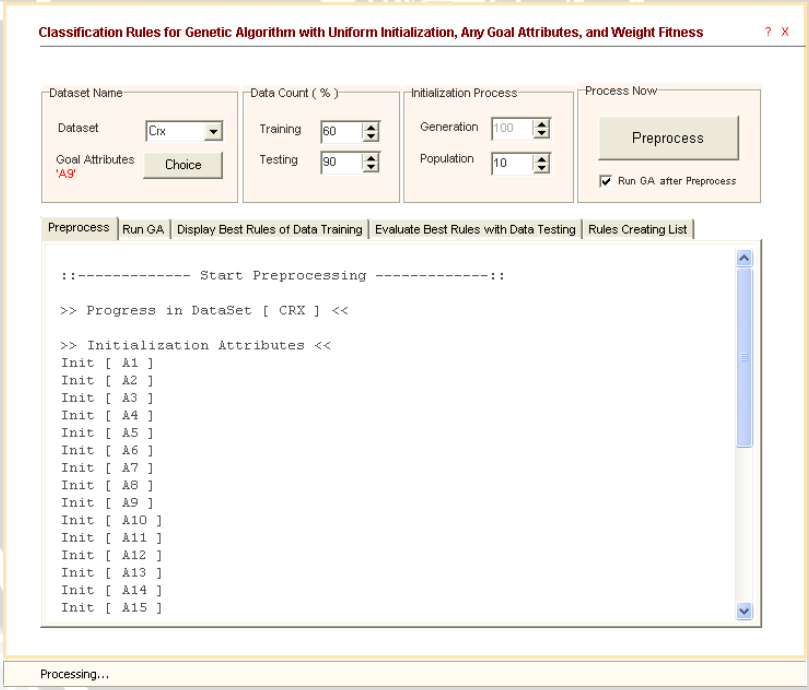
## 4.3 Implementasi Antarmuka

Pada implementasi antarmuka ini akan dijelaskan tentang *form-form* dalam sistem yang telah dibuat sebagai media dalam mendapatkan aturan-aturan *mining* serta menghitung kelayakan dan keakuratannya.

*Form* yang telah dibuat dalam sistem terdiri dari 2 *form*. *Form* utama yang terdiri dari 5 tab, yakni tab untuk menampilkan proses pengkondisian data-contoh, tab untuk menampilkan proses Algoritma Genetika, tab untuk menampilkan analisa aturan terbaik berdasar data-contoh, tab untuk menampilkan evaluasi aturan terbaik berdasar data-uji, dan tab untuk menampilkan sejumlah aturan terbaik dari tiap-tiap jenis nilai atribut tujuan. Kedua, *form* statistik untuk mengatur atribut tujuan dari data-contoh dan menampilkan deskripsi dari data-contoh.

### 4.3.1 *Form* Utama Tab Pengkondisian Data-Contoh

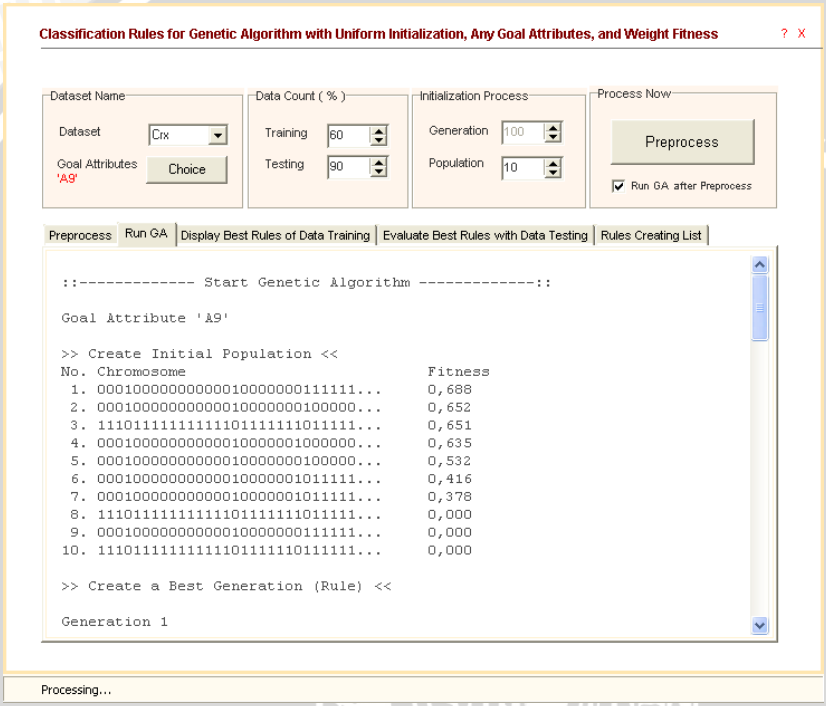
*Form* Utama pada tab pengkondisian data-contoh ini untuk mengimplementasikan proses pengkondisian data-contoh. Dalam tab ini terdapat *memo* untuk mencetak proses pengkondisian data-contoh. Implementasi dari proses pengkondisian data-contoh pada tab *form* pengkondisian ini ditunjukkan pada gambar 4.12.




Gambar 4.12 Implementasi *Form* Utama Tab Pengkondisian Data-Contoh

### 4.3.2 *Form* Utama Tab Algoritma Genetika

*Form* Utama pada tab Algoritma Genetika ini untuk meng-implementasikan proses Algoritma Genetika. Dalam tab ini terdapat memo untuk mencetak proses sistem ketika sedang berada pada tahapan operasi genetika. Implementasi dari proses Algoritma Genetika pada tab *form* Algoritma Genetika ini seperti pada gambar 4.13.

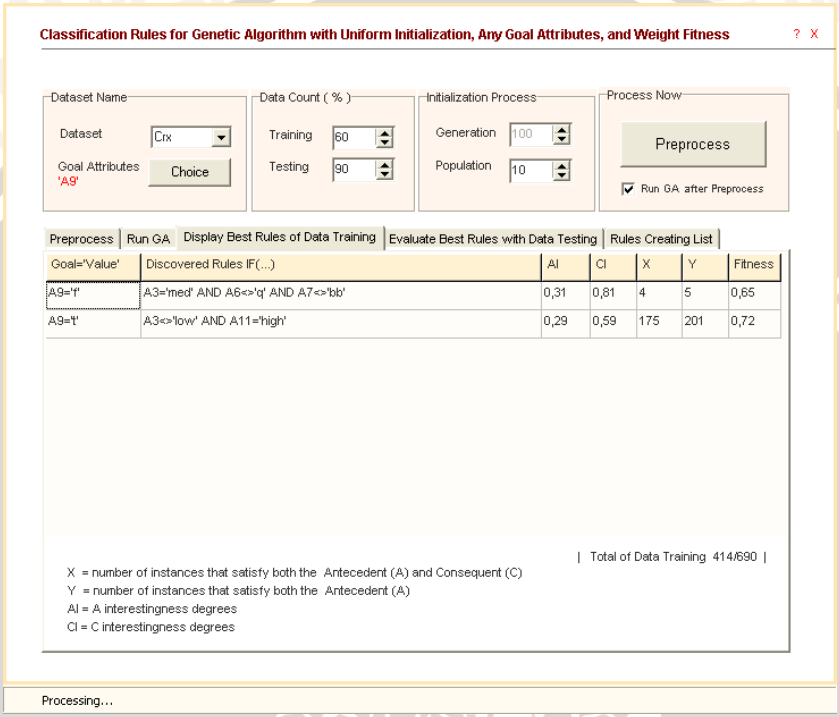



Gambar 4.13 Implementasi *Form* Utama Tab Algoritma Genetika

### 4.3.3 *Form* Utama Tab Analisa Aturan

*Form* Utama pada tab analisa aturan dengan data-latih ini untuk mengimplementasikan proses pengujian aturan dan menilai keakuratan dan kelayakan aturan dari parameter-parameter *AI, CI, PA,* dan *fitness*. Dalam tab ini terdapat tabel untuk mencetak aturan-aturan terbaik dari tiap-tiap jenis nilai dari atribut tujuan, sekaligus mencetak nilai *AI, CI,*

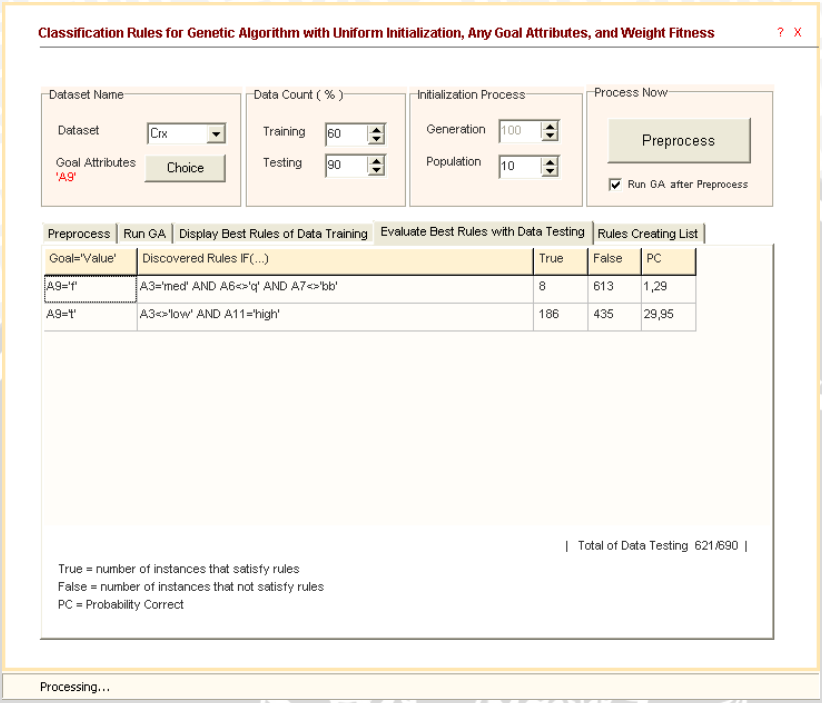
*PA*, dan *fitness* dari masing-masing aturan tersebut. Implementasi dari proses pengujian aturan dengan data-latih pada tab *form* pengujian aturan ini ditunjukkan pada gambar 4.14.




Gambar 4.14 Implementasi *Form* Utama Tab Analisa Aturan

### 4.3.4 *Form* Utama Tab Evaluasi Aturan

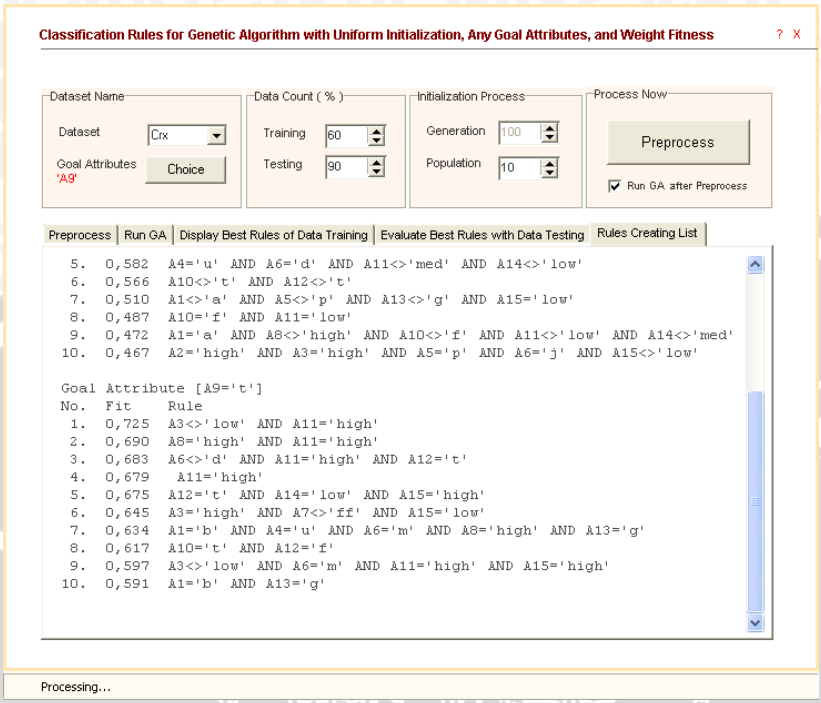
*Form* Utama pada tab mengevaluasi aturan dengan data-uji ini untuk mengimplementasikan proses evaluasi aturan dengan data-uji yang terukur dari parameter *True, False*, dan *PC*. Dalam tab ini terdapat tabel untuk mencetak aturan-aturan terbaik dari tiap-tiap jenis nilai dari atribut tujuan, beserta nilai dari *True, False*, dan *PC* dari aturan-aturan tersebut. Implementasi dari proses evaluasi aturan dengan data-uji pada tab *form* evaluasi aturan ini seperti pada gambar 4.15.




Gambar 4.15 Implementasi *Form* Utama Tab Evaluasi Aturan

### 4.3.5 *Form* Utama Tab Daftar Aturan

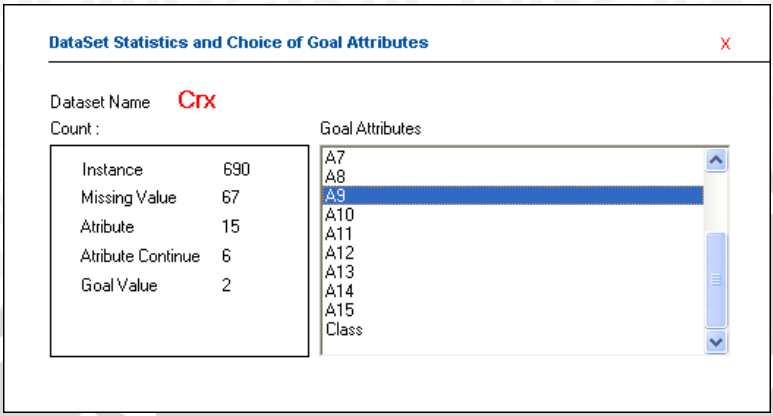
*Form* Utama pada tab daftar aturan untuk mencetak sejumlah aturan (dibatasi ≤ jumlah populasi) yang terbentuk dari operasi genetika untuk tiap-tiap jenis nilai dari atribut tujuan tertentu. Dalam tab ini terdapat *memo* untuk mencetak aturan-aturan dan nilai *fitness*-nya. Implementasi dari proses mencetak aturan-aturan pada tab *form* daftar aturan ini ditunjukkan pada gambar 4.16.




Gambar 4.16 Implementasi *Form* Utama Tab Daftar Aturan

### 4.3.6 *Form* Statistik Data-Contoh

*Form* Statistik Data-Contoh berfungsi untuk menentukan pilihan atribut tujuan pada data-contoh dan untuk menampilkan deskripsi dari data-contoh, yaitu jumlah barisnya, jumlah keseluruhan nilai hilang, jumlah atribut, jumlah atribut bertipe kontinu, dan banyak jenis nilai pada atribut tujuan. Implementasi dari pemilihan atribut tujuan dan menampilkan deskripsi data-contoh pada *form* statistik data-contoh ditunjukkan pada gambar 4.17.




Gambar 4.17 Implementasi *Form* Statistik Data-Contoh

## 4.4 Implementasi Uji Coba

Sistem yang menghasilkan aturan *mining* serta nilai kelayakan dan keakuratannya dari metode Algoritma Genetika, membutuhkan pengujian terhadap kualitas hasil yang diperoleh sistem tersebut. Telah dirancang skenario uji coba dan analisanya terhadap kualitas hasil dari sistem dengan tujuan menilai keoptimalan sistem untuk menghasilkan aturan yang layak dan akurat. Diantara uji cobanya yaitu melakukan pengujian terhadap parameter-parameter sistem yang menentukan hasil akhir dari sistem. Secara lebih lengkapnya akan dijelaskan pada subbab-subbab tersendiri tentang skenario uji coba serta evaluasi dan analisa hasilnya.

### 4.4.1 Skenario Uji Coba

Seperti yang telah dijelaskan dalam rancangan skenario uji coba pada bab 3.5, penilaian terhadap kelayakan dan keakuratan suatu aturan dengan Algoritma Genetika mempertimbangkan pengujian terhadap banyak parameter diantaranya persentase data-latih, persentase data-uji, dan jumlah populasi. Bila dirinci maka skenario uji coba yang telah diimplementasikan yakni :

a. Menguji parameter persentase data-latih antara rentang 60%-100% dengan persentase data-uji senilai 10% dari data-contoh *Crx* dan jumlah populasi senilai 20. Tujuan pengujian parameter persentase data-latih untuk mengetahui nilai *Probability Correct (PC).* Pengujian dilakukan 10 kali.

b. Menguji parameter persentase data-uji antara rentang 60%-100% dengan persentase data-latih senilai 67% dari data-contoh *Crx* dan jumlah populasi 20. Tujuan pengujian parameter persentase data-latih untuk mengetahui nilai *PC.* Pengujian dilakukan 10 kali.

c. Menguji parameter jumlah populasi antara rentang 10-100 dengan persentase data-latih senilai 67% dari data-contoh dan persentase data-uji senilai 33% dari data-contoh. Tujuan pengujian parameter jumlah populasi untuk mengetahui nilai kelayakan *antecedent* (*AI*), nilai kelayakan *consequent* (*CI*), keakuratan prediksi (*PA*), dan nilai *fitness*. Pengujian dilakukan 10 kali.

### 4.4.2 Evaluasi dan Analisa Hasil

Pada subbab evaluasi dan analisa hasil ini akan dijelaskan tentang evaluasi dan analisa hasil dari pengujian pengaruh parameter jumlah populasi terhadap nilai *AI* (kelayakan antecedent), nilai *CI* (kelayakan consequent), nilai *PA* (prediksi keakuratan), dan nilai *Fitness*; serta pengujian pengaruh persentase data-latih dan persentase data-uji terhadap *PC* (persentase kebenaran aturan dari data-uji).

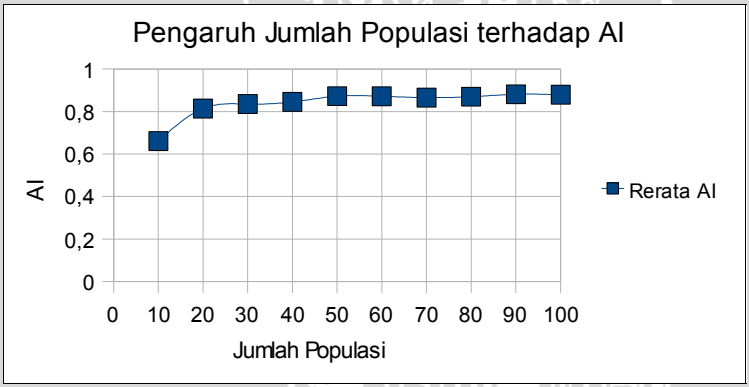
#### 4.4.2.1 Pengaruh Jumlah Populasi Terhadap *AI*

Pengujian ini dilakukan 10 kali dengan rentang jumlah populasi antara 10-100 dengan kelipatan 10, persentase data-latih 67 % dari data-contoh, dan persentase data-uji 33 % dari data-contoh. Hasil dari pengujian parameter jumlah populasi terhadap *AI* akan ditunjukkan pada tabel 4.1 dengan kolom-kolom untuk data percobaan dari 1 sampai 10 dan reratanya.

Tabel 4.1 Data Pengaruh Jumlah Populasi Terhadap *AI*

| Populasi | AI | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | Rerata |
| 10 | 0,79 | 0,41 | 0,66 | 0,41 | 0,51 | 0,82 | 0,76 | 0,84 | 0,83 | 0,6 | 0,66 |
| 20 | 0,87 | 0,54 | 0,94 | 0,92 | 0,82 | 0,86 | 0,85 | 0,85 | 0,88 | 0,63 | 0,81 |
| 30 | 0,9 | 0,78 | 0,94 | 0,65 | 0,88 | 0,89 | 0,87 | 0,8 | 0,8 | 0,86 | 0,84 |
| 40 | 0,9 | 0,9 | 0,83 | 0,81 | 0,88 | 0,88 | 0,78 | 0,76 | 0,92 | 0,81 | 0,85 |
| 50 | 0,91 | 0,87 | 0,9 | 0,87 | 0,88 | 0,84 | 0,82 | 0,9 | 0,85 | 0,9 | 0,87 |
| 60 | 0,88 | 0,89 | 0,85 | 0,88 | 0,85 | 0,83 | 0,93 | 0,89 | 0,84 | 0,89 | 0,87 |
| 70 | 0,8 | 0,78 | 0,91 | 0,87 | 0,92 | 0,89 | 0,95 | 0,88 | 0,84 | 0,85 | 0,87 |
| 80 | 0,91 | 0,88 | 0,88 | 0,92 | 0,86 | 0,86 | 0,91 | 0,79 | 0,82 | 0,9 | 0,87 |
| 90 | 0,94 | 0,84 | 0,87 | 0,87 | 0,89 | 0,91 | 0,93 | 0,84 | 0,83 | 0,91 | 0,88 |
| 100 | 0,98 | 0,81 | 0,86 | 0,9 | 0,88 | 0,86 | 0,87 | 0,9 | 0,9 | 0,84 | 0,88 |

Sedangkan representasi dari tabel 4.1 berupa grafik ditunjukkan pada gambar 4.18.




Gambar 4.18 Grafik Pengaruh Jumlah Populasi Terhadap *AI*

Dari gambar 4.18 tersebut terlihat pola grafik yang menaik seiring bertambahnya jumlah populasi, artinya kenaikan jumlah populasi mempengaruhi nilai dari *AI*. Mulai jumlah populasi ke-10 dengan rerata *AI*=0,66, pola grafik mengalami penaikan menuju jumlah populasi ke-20 dengan *AI*=0,81. Sedang pada jumlah populasi ke-20 dengan *AI*=0,81 sampai jumlah populasi ke-100 dengan *AI*=0,88, terlihat pola grafik yang stagnan, berkisar dengan nilai rerata *AI*=0,86. Grafik yang

stagnan ini menunjukkan bahwa untuk jumlah populasi ke-20 sampai ke-100, menghasilkan nilai kelayakan antecedent (*AI*) yang cukup stabil.

Jumlah populasi yang banyak memberikan lebih banyak peluang untuk menghasilkan kromosom dengan nilai *fitness*nya yang lebih baik. Sehingga perubahan parameter jumlah populasi sangat berpengaruh pada nilai kelayakan antecedent (AI).

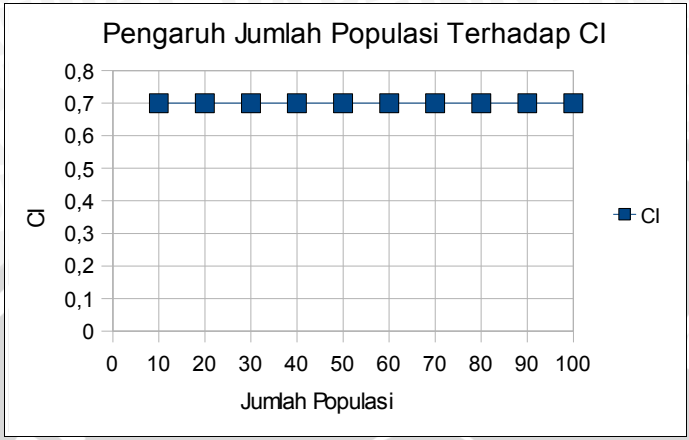
Dari gambar 4.18 bisa diambil kesimpulan bahwa perubahan nilai parameter jumlah populasi mempengaruhi nilai kelayakan *antecedent* (*AI*) dari sebuah aturan. Makin besar jumlah populasinya, maka makin layak *antecedent* dari aturan tersebut. Begitu pula sebaliknya. Dalam pengujian ini nilai kelayakan *antecedent* (*AI*) paling maksimal pada jumlah populasi ke-90 dan jumlah populasi ke-100 dengan nilai *AI*=0,88.

### 4.4.2.2 Pengaruh Jumlah Populasi Terhadap *CI*

Pengujian parameter jumlah populasi terhadap *CI* dilakukan 10 kali dengan jumlah populasi antara 10-100 dengan kelipatan 10, persentase data-latih 67 % dari data-contoh, dan persentase data-uji 33 % dari data-contoh. Hasil pengujiannya ditunjukkan pada tabel 4.2, sedangkan representasi dari tabel 4.2 berupa grafik ditunjukkan pada gambar 4.19, dengan kolom-kolom untuk data percobaan dari 1 sampai 10 dan reratanya.

Tabel 4.2 Data Pengaruh Jumlah Populasi Terhadap *CI*

| Populasi | CI | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | Rerata |
| 10 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 |
| 20 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 |
| 30 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 |
| 40 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 |
| 50 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 |
| 60 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 |
| 70 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 |
| 80 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 |
| 90 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 |
| 100 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 | 0,7 |




Gambar 4.19 Grafik Pengaruh Jumlah Populasi Terhadap *CI*

Dari gambar 4.19, tampak pola grafik yang tetap pada populasi dari 10 sampai 100, artinya perubahan nilai jumlah populasi yang meninggi dari 10 sampai 100 tidak menunjukkan perubahan sama sekali pada pola grafik yang hanya bernilai rata-rata *CI*=0,7. Akan tetapi nilai *CI* untuk masing-masing jenis nilai atribut *A9* berbeda. Seperti pada tabel 4.2, untuk nilai A9='t' bernilai CI=0,57 dan untuk nilai A9='f' bernilai CI=0,82.

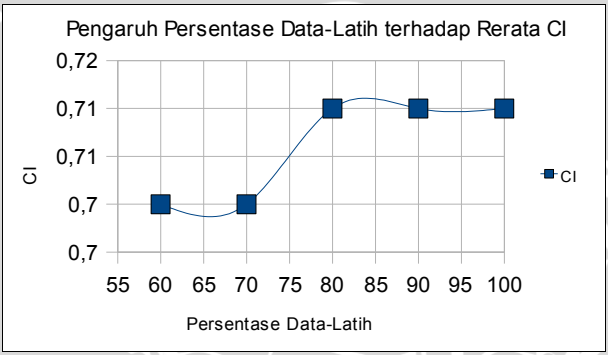
Tidak adanya perubahan pola grafik bisa terjadi karena nilai *CI* dihitung dari peluang ditemukannya nilai atribut tujuan *A9* dari sejumlah baris data-latih. Sedangkan pada pengujian kali ini, persentase data-latih telah ditentukan senilai 67 % dari data-contoh. Sehingga jelas karena penentuan nilai dari data-latih ini, mengakibatkan tidak adanya pola grafik naik-turun pada gambar 4.19.

Perlu dilakukan penelitian untuk menguji pengaruh parameter persentase data-latih terhadap *CI*. Pengujian ini dimaksudkan untuk mengetahui pengaruh perubahan persentase data-latih terhadap nilai *CI*. Pengujian ini telah dilakukan 3 kali, dengan rentang persentase data-latih 60%-100% dari data-contoh, persentase data-uji 10% dari data-contoh, dan jumlah populasi 20. Hasil pengujian persentase data-latih terhadap rerata *CI* ditunjukkan pada tabel 4.3.

Tabel 4.3 Data Pengaruh Persentase Data-Latih Terhadap Rerata *CI*

| %Data-Latih | CI |
|---|---|
| 60 | 0,7 |
| 70 | 0,7 |
| 80 | 0,71 |
| 90 | 0,71 |
| 100 | 0,71 |

Pada tabel 4.3 pada kolom Rerata *CI*, merupakan kolom untuk mencetak nilai rerata *CI* dari ketiga percobaan pada persentase data-latih yang sama. Sedang representasi dari tabel 4.3 berupa grafik ditunjukkan pada gambar 4.20.




Gambar 4.20 Grafik Pengaruh Persentase Data-Latih Terhadap Rerata *CI*

Pada gambar 4.20 dimulai dari persentase data-latih 60% bernilai *CI*=0,7. Kemudian mulai terjadi kenaikan pola grafik menuju persentase data-latih 80% dengan *CI*=0,71 meskipun tidak tinggi, yang hanya terpaut CI senilai 0,01. Dari persentase data-latih 80% inilah berikutnya sampai persentase data-latih 100% terjadi pola grafik yang tetap, hanya berkisar di nilai CI=0,71.

Hal seperti ini yakni pola grafik yang stagnan mulai persentase data-latih 70%, menunjukkan bahwa sekalipun dilakukan perubahan nilai persentase data-latih, tetap tidak berpengaruh pada nilai *CI*. Hal ini menambah kesimpulan baru setelah dilakukan evaluasi dan analisa pada pengaruh jumlah populasi terhadap *CI* sebelumnya, bahwa perubahan

nilai parameter persentase data-latih juga tidak berpengaruh pada kelayakan *consequent* (*CI*) dari sebuah aturan. Alasannya serupa dengan alasan pola grafik yang stagnan pada pengujian pengaruh parameter jumlah populasi terhadap nilai *CI*.

Dengan demikian disimpulkan bahwa perubahan nilai parameter jumlah populasi tidak mempengaruhi nilai kelayakan *consequent* (*CI*) dari aturan. Begitu pula penentuan persentase nilai data-latih dengan besaran nilai yang tetap (seperti persentase data-latih senilai 67% dari data-contoh) atau tidak ditentukan dengan besaran nilai yang tetap (seperti rentang persentase data-latih 60%-100%), tidak berimbas sama sekali pada perubahan nilai kelayakan *consequent* (*CI*) dari aturan.

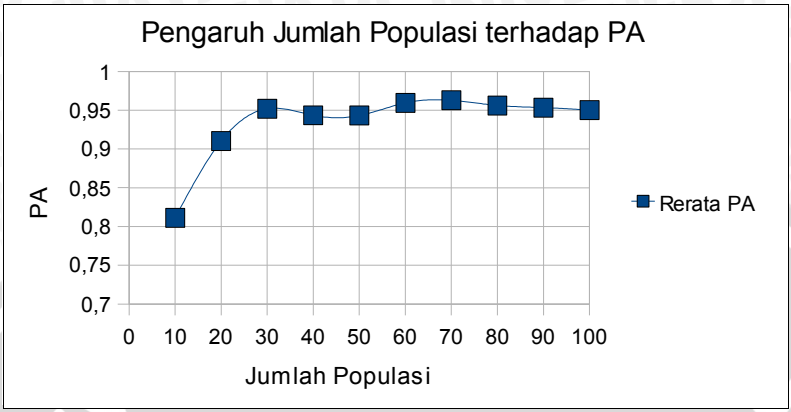
#### 4.4.2.3 Pengaruh Jumlah Populasi Terhadap *PA*

Pengujian parameter jumlah populasi terhadap *PA* dilakukan sekali dengan jumlah populasi antara 10-100 dengan kelipatan 10, persentase data-latih 67 % dari data-contoh, dan persentase data-uji 33 % dari data-contoh. Hasil pengujian parameter jumlah populasi terhadap *PA* ditunjukkan pada tabel 4.4, dengan kolom-kolom untuk data percobaan dari 1 sampai 10 dan reratanya.

Tabel 4.4 Data Pengaruh Jumlah Populasi Terhadap *PA*

| Populasi | PA | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | Rerata |
| 10 | 0,73 | 0,83 | 0,87 | 0,87 | 0,88 | 0,71 | 0,91 | 0,7 | 0,86 | 0,75 | 0,81 |
| 20 | 0,93 | 0,84 | 0,85 | 0,85 | 0,94 | 0,94 | 0,96 | 0,89 | 0,96 | 0,93 | 0,91 |
| 30 | 0,94 | 0,92 | 0,95 | 0,95 | 0,97 | 0,96 | 0,93 | 0,98 | 0,95 | 0,96 | 0,95 |
| 40 | 0,96 | 0,94 | 0,91 | 0,98 | 0,93 | 0,95 | 0,95 | 0,91 | 0,95 | 0,94 | 0,94 |
| 50 | 0,91 | 0,95 | 0,95 | 0,93 | 0,95 | 0,95 | 0,95 | 0,96 | 0,94 | 0,96 | 0,94 |
| 60 | 0,97 | 0,95 | 0,95 | 0,97 | 0,97 | 0,96 | 0,94 | 0,96 | 0,98 | 0,96 | 0,96 |
| 70 | 0,98 | 0,97 | 0,96 | 0,96 | 0,95 | 0,96 | 0,96 | 0,95 | 0,98 | 0,95 | 0,96 |
| 80 | 0,97 | 0,97 | 0,94 | 0,94 | 0,93 | 0,96 | 0,97 | 0,95 | 0,97 | 0,96 | 0,96 |
| 90 | 0,96 | 0,98 | 0,95 | 0,97 | 0,95 | 0,94 | 0,91 | 0,96 | 0,96 | 0,97 | 0,95 |
| 100 | 0,95 | 0,93 | 0,97 | 0,94 | 0,95 | 0,95 | 0,95 | 0,97 | 0,94 | 0,96 | 0,95 |

Sedangkan representasi dari tabel 4.4 dalam bentuk grafik ditunjukkan pada gambar 4.21.




Gambar 4.21 Grafik Pengaruh Jumlah Populasi Terhadap *PA*

Pada gambar 4.21 tampak kecenderungan pola grafik yang menaik. Dimulai pada jumlah populasi ke-1 dengan nilai *PA*=0,81 yang menaik terus sampai jumlah populasi ke-30 dengan nilai *PA*=0,95. Kemudian, dari jumlah populasi ke-30 ini sampai jumlah populasi ke-100 dengan nilai *PA*=0,95, pola grafik tidak mengalami perubahan atau stagnan dengan nilai rerata *PA*=0,95. Artinya, pada jumlah populasi ke-30 sampai ke-100, dihasilkan aturan-aturan dengan nilai keakuratan prediksi (*PA*) yang cukup stabil.

Dari pengujian pengaruh parameter jumlah populasi terhadap *PA* ini didapatkan kesimpulan, bahwa kenaikan jumlah populasi mempengaruhi kenaikan nilai *PA* pada aturan-aturan, dengan nilai *PA* paling tinggi=0,96 pada jumlah populasi ke-60 sampai ke-80.

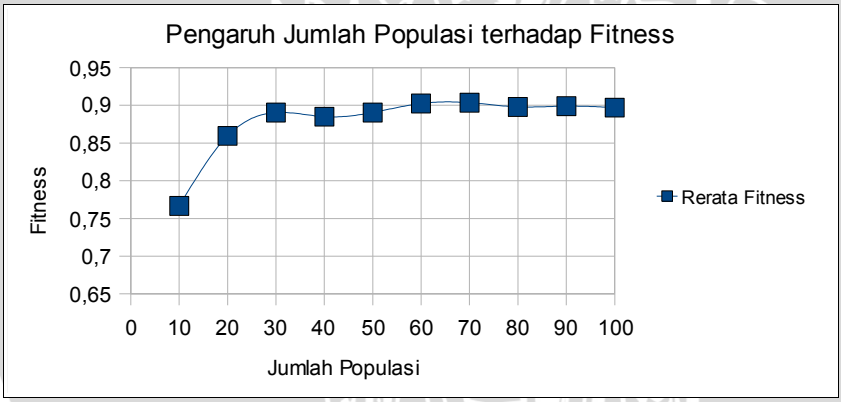
#### 4.4.2.4 Pengaruh Jumlah Populasi Terhadap *Fitness*

Pengujian terakhir pada parameter jumlah populasi yakni pengujian parameter jumlah populasi terhadap *Fitness*. Pengujian ini dilakukan 10 kali dengan jumlah populasi antara 10-100 dengan kelipatan 10, persentase data-latih 67 % dari data-contoh, dan persentase data-uji 33 % dari data-contoh. Hasil pengujian parameter jumlah populasi terhadap *Fitness* ditunjukkan pada tabel 4.5, dengan kolom-kolom untuk data percobaan dari 1 sampai 10 dan reratanya.

Tabel 4.5 Data Pengaruh Jumlah Populasi Terhadap *Fitness*

| Populasi | Fitness | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | Rerata |
| 10 | 0,73 | 0,74 | 0,81 | 0,76 | 0,79 | 0,73 | 0,85 | 0,73 | 0,83 | 0,72 | 0,77 |
| 20 | 0,88 | 0,77 | 0,84 | 0,84 | 0,88 | 0,89 | 0,91 | 0,85 | 0,91 | 0,85 | 0,86 |
| 30 | 0,9 | 0,86 | 0,91 | 0,86 | 0,91 | 0,91 | 0,89 | 0,9 | 0,88 | 0,9 | 0,89 |
| 40 | 0,91 | 0,9 | 0,87 | 0,91 | 0,88 | 0,9 | 0,88 | 0,85 | 0,91 | 0,88 | 0,89 |
| 50 | 0,88 | 0,9 | 0,9 | 0,88 | 0,9 | 0,89 | 0,89 | 0,9 | 0,89 | 0,91 | 0,89 |
| 60 | 0,91 | 0,9 | 0,9 | 0,91 | 0,9 | 0,9 | 0,9 | 0,92 | 0,91 | 0,9 | 0,9 |
| 70 | 0,91 | 0,9 | 0,91 | 0,91 | 0,91 | 0,91 | 0,92 | 0,9 | 0,91 | 0,9 | 0,9 |
| 80 | 0,91 | 0,91 | 0,89 | 0,9 | 0,89 | 0,9 | 0,91 | 0,88 | 0,9 | 0,91 | 0,9 |
| 90 | 0,91 | 0,91 | 0,9 | 0,91 | 0,9 | 0,9 | 0,88 | 0,9 | 0,9 | 0,92 | 0,9 |
| 100 | 0,91 | 0,87 | 0,91 | 0,89 | 0,9 | 0,9 | 0,9 | 0,92 | 0,9 | 0,9 | 0,9 |

Sedangkan representasi dari tabel 4.5 berupa grafik ditunjukkan pada gambar 4.22.




Gambar 4.22 Grafik Pengaruh Jumlah Populasi Terhadap *Fitness*

Pada gambar 4.22 terlihat kenaikan pola grafik, dimulai dari jumlah populasi ke-10 bernilai *fitness*=0,77 sampai jumlah populasi ke-30 dengan nilai *fitness*=0,89. Kemudian pada jumlah populasi ke-30 sampai akhir jumlah populasi ke-100 dengan nilai *fitness*=0,9, pola grafik mengalami stagnan dengan nilai rerata *fitness*=0,895. Nilai *fitness* terbaik=0,9 dihasilkan pada jumlah populasi ke-60 sampai ke-100.

Dari pengujian parameter jumlah populasi terhadap nilai *fitness* bisa diambil sebuah kesimpulan bahwa makin besarnya jumlah populasi, mengakibatkan makin tingginya nilai *fitness* dari sebuah aturan yang artinya makin baik kualitas kelayakan dan keakuratan dari aturan yang dihasilkan tersebut. Disebabkan modifikasi individu yang dilakukan terhadap jumlah populasi yang banyak akan memperbesar peluang untuk mencapai nilai *fitness* yang baik, meskipun tidak terpaut sangat jauh antara nilai-nilai *fitness*nya.

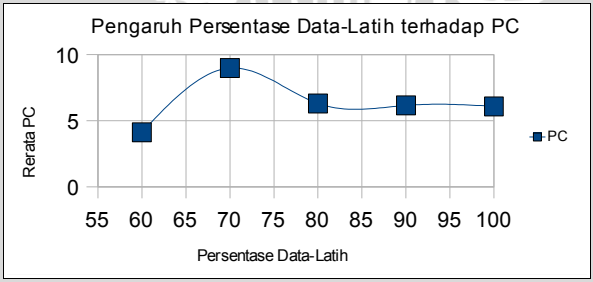
#### 4.4.2.5 Pengaruh Persentase Data-Latih terhadap Rerata *PC*

Berikutnya pengujian pada parameter persentase data-latih terhadap rerata *PC*. Pada pengujian ini dilakukan sebanyak 10 kali untuk tiap-tiap persentase data-latih antara rentang 60%-100%. Sedangkan persentase data-uji ditentukan 10% dari data-contoh dan jumlah populasinya 20. Hasil pengujian pengaruh parameter persentase data-latih terhadap rerata *PC*, ditunjukkan pada tabel 4.6, dengan kolom-kolom untuk data percobaan dari 1 sampai 10 dan reratanya.

Tabel 4.6 Data Pengaruh Persentase Data-Latih Terhadap Rerata *PC*

| % Data Latih | PC | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | Rerata |
| 60 | 18,84 | 1,45 | 0,73 | 2,18 | 4,35 | 3,63 | 2,18 | 7,25 | 0 | 0,73 | 4,13 |
| 70 | 9,42 | 22,46 | 10,15 | 10,15 | 12,32 | 4,35 | 4,35 | 7,25 | 5,08 | 4,35 | 8,99 |
| 80 | 10,87 | 6,53 | 10,87 | 1,45 | 11,6 | 2,18 | 4,35 | 1,45 | 1,45 | 12,32 | 6,31 |
| 90 | 3,63 | 10,15 | 1,45 | 5,07 | 1,45 | 28,26 | 0,73 | 2,18 | 6,52 | 2,18 | 6,16 |
| 100 | 2,18 | 5,8 | 0 | 2,18 | 2,18 | 16,67 | 2,18 | 4,35 | 21,74 | 3,63 | 6,09 |

Representasi dari tabel 4.6 berupa grafik ditunjukkan pada gambar 4.23.



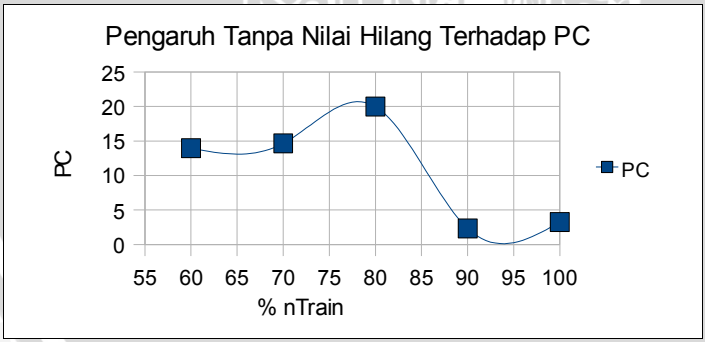

Gambar 4.23 Grafik Pengaruh Persentase Data-Latih Terhadap Rerata *PC*

Pada gambar 4.23 pada persentase data-latih 60% memiliki nilai rerata *PC*=4,13. Menuju pada persentase data-latih 70% terjadi kenaikan pola grafik dengan nilai rerata *PC* menyentuh nilai 8,99. Kemudian pola grafik menurun menuju persentase data-latih 80% dengan rerata *PC*=6,31. Dari persentase data-latih 80% dengan rerata *PC*=6,31 sampai persentase data-latih 100% dengan rerata PC=6,09 dihasilkan pola grafik yang stagnan.

Pola grafik pada gambar 4.23 secara umum mengalami pola grafik yang naik-turun. Hipotesa awal dari penelitian yakni makin tinggi persentase dari data-latih maka makin tinggi nilai persentase kebenaran aturan dari data-uji (*PC*). Ternyata hasil dari pengujian bertolak belakang dengan hipotesa penelitian. Untuk mengetahui sebab dari penurunan pola grafik pada gambar 4.23, perlu dilakukan penelitian lebih lanjut.

Seperti halnya untuk pengujian pengaruh jumlah populasi terhadap *PA*, penelitian lebih lanjut ini tertuju pada pengujian pengaruh atribut dengan sebaran banyak nilai pada data-contoh *Crx*. Telah disinggung pada subbab 3.1.1, bahwa data-contoh *Crx* terdiri dari 67 nilai hilang, 6 atribut kontinu (atribut *A2, A3, A8, A11, A14,* dan *A15*), dan 2 atribut berisi sebaran banyak nilai (atribut *A6* dan *A7*). Untuk mengetahui seberapa tinggi tingkat pengaruh adanya nilai hilang, nilai kontinu dan atribut dengan sebaran banyak nilai terhadap persentase kebenaran aturan dari data-uji, maka dilakukan pengujian terhadap masing-masing kemungkinan.

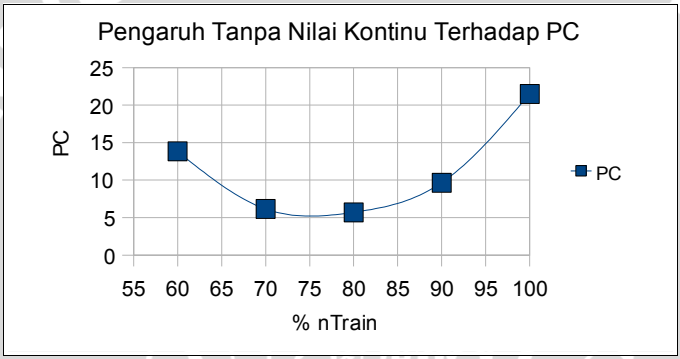
a. Pengaruh data-contoh *Crx* tanpa nilai hilang terhadap *PC*




Gambar 4.24 Grafik Pengaruh Tanpa Nilai Hilang Terhadap *PC*

Pada gambar 4.24, pola grafik menunjukkan naik-turun. Dimulai dari persentase data-latih 60% dengan *PC*=13,96 terus menaik sampai persentase data-latih 80% dengan *PC*=20. Memasuki persentase data-latih 90% dengan *PC*=2,33 sampai berakhir pada persentase data-latih 100% dengan *PC*=3,26 mengalami penurunan yang drastis. Untuk lebih lengkapnya data pengaruh *Crx* tanpa nilai hilang terhadap *PC* bisa dilihat di lampiran. Pola grafik seperti pada gambar 4.24 yang naik-turun, sulit untuk menyimpulkan ada tidaknya pengaruh data-contoh *Crx* tanpa nilai hilang terhadap *PC*.

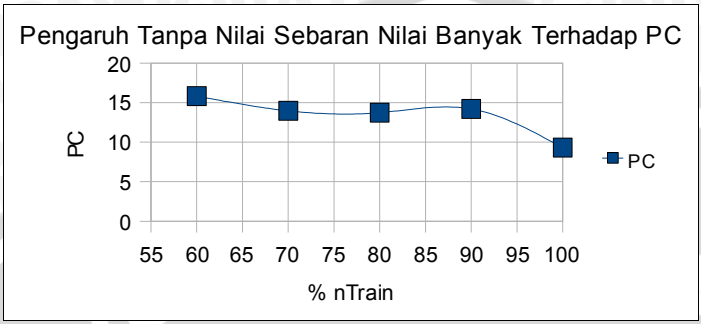
b. Pengaruh data-contoh *Crx* tanpa nilai kontinu terhadap *PC*




Gambar 4.25 Grafik Pengaruh Tanpa Nilai Kontinu Terhadap *PC*

Pada gambar 4.25, pola grafik menunjukkan kecenderungan untuk menaik. Dimulai dari persentase data-latih 60% dengan *PC*=13,82 dan menurun pada persentase data-latih berikutnya 70% dengan *PC*=6,14 sampai pada persentase data-latih 80% dengan *PC*=5,71. Mulai dari persentase data-latih 80% terjadi kenaikan pola grafik sampai berakhir di persentase data-latih 100% dengan *PC*=21,49. Untuk lebih lengkapnya data pengaruh *Crx* tanpa nilai kontinu terhadap *PC*, bisa dilihat di lampiran. Pola grafik pada gambar 4.25 yang cenderung menaik tersebut, menandakan tidak adanya nilai kontinu pada data-contoh *Crx* menyebabkan nilai *PC* menjadi tinggi. Bisa disimpulkan, salah satu yang menyebabkan nilai *PC* pada data-contoh *Crx* menjadi rendah yakni adanya nilai kontinu, apalagi pada data-contoh *Crx* terdapat 6 atribut kontinu yang makin menjadikan rendah nilai *PC*-nya.

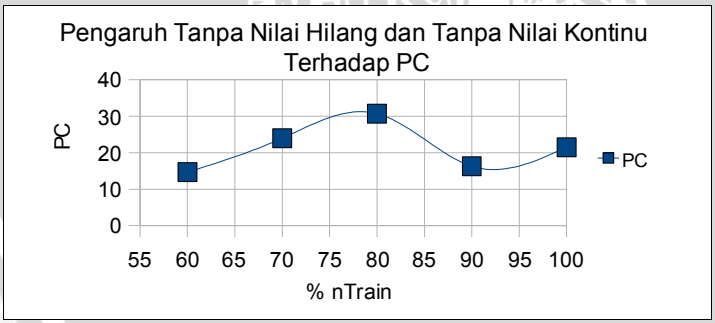
c. Pengaruh data-contoh *Crx* tanpa sebaran banyak nilai terhadap *PC*




Gambar 4.26 Grafik Pengaruh Tanpa Sebaran Banyak Nilai Terhadap *PC*

Pada gambar 4.26, tampak pola grafik yang menurun. Dimulai dari persentase data-latih 60% dengan *PC*=15,82 sampai berakhir pada persentase data-latih 100% dengan *PC*=9,3. Untuk lebih lengkapnya data pengaruh *Crx* tanpa sebaran banyak nilai terhadap *PC*, bisa dilihat di lampiran. Pola grafik yang menurun seperti pada gambar 4.26, menunjukkan bahwa ada tidaknya atribut dengan sebaran banyak nilai (atribut *A6* dan *A7*), tetap menyebabkan rendahnya nilai *PC* pada data-contoh *Crx*. Sehingga ada faktor lain, selain adanya atribut dengan sebaran banyak nilai, yang menjadikan nilai *PC* pada data-contoh *Crx* sangat rendah.

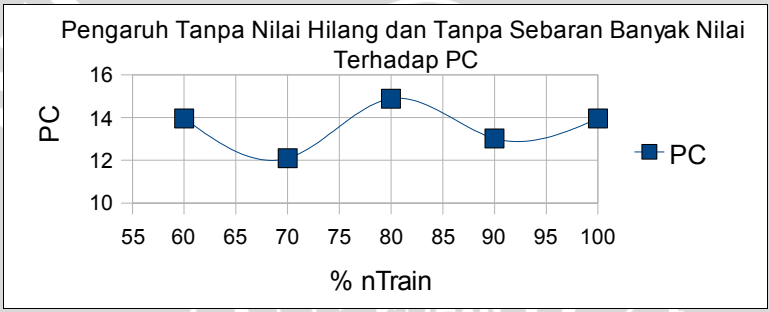
d. Pengaruh data-contoh *Crx* tanpa nilai hilang dan tanpa nilai kontinu terhadap *PC*




Gambar 4.27 Grafik Pengaruh Tanpa Nilai Hilang dan Tanpa Nilai Kontinu Terhadap *PC*

Pada gambar 4.27, terlihat pola grafik yang naik-turun. Dimulai pada persentase data-latih 60% dengan *PC*=14,66 sampai persentase data-latih 80% dengan *PC*=30,7. Berikutnya memasuki persentase data-latih 90% dengan *PC*=16,28 terjadi penurunan pola grafik dan kembali menaik pada persentase data-latih 100% dengan *PC*=21,49. Untuk lebih lengkapnya data pengaruh *Crx* tanpa nilai hilang dan tanpa nilai kontinu terhadap *PC*, bisa dilihat di lampiran. Pola grafik yang naik turun seperti pada gambar 4.27, sulit untuk menyimpulkan ada tidaknya pengaruh *Crx* tanpa nilai hilang dan nilai kontinu terhadap *PC*.

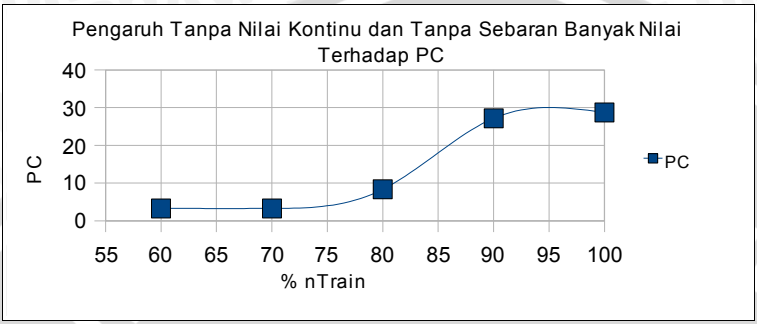
e. Pengaruh data-contoh *Crx* tanpa nilai hilang dan tanpa sebaran banyak nilai terhadap *PC*




Gambar 4.28 Grafik Pengaruh Tanpa Nilai Hilang dan Tanpa Sebaran Banyak Nilai Terhadap *PC*

Pada gambar 4.28 terlihat pola grafik yang cenderung naik-turun. Pada persentase data-latih 60% bernilai *PC*=13,96 turun menuju persentase data-latih 70% bernilai *PC*=12,1. Pola grafik menaik kembali menuju persentase data-latih 80% dengan *PC*=14,89. Kemudian menurun dan menaik lagi sampai persentase akhir data-latih 100% dengan nilai *PC*=13,96. Untuk lebih lengkapnya data pengaruh *Crx* tanpa nilai hilang dan tanpa sebaran banyak nilai terhadap *PC*, bisa dilihat di lampiran. Pola grafik yang naik turun seperti pada gambar 4.28, akan sangat sulit untuk menyimpulkan ada tidaknya pengaruh *Crx* tanpa nilai hilang dan tanpa sebaran banyak nilai terhadap *PC*.

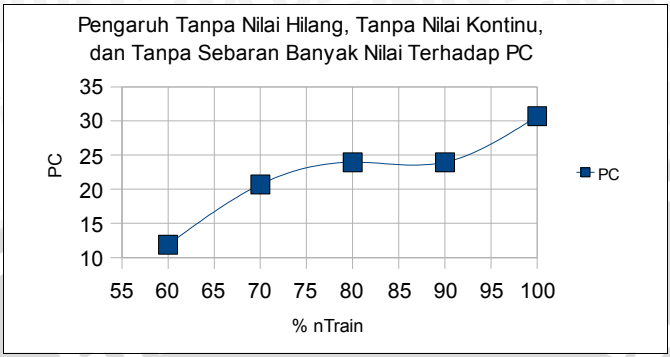
f. Pengaruh data-contoh *Crx* tanpa nilai kontinu dan tanpa sebaran banyak nilai terhadap *PC*




Gambar 4.29 Grafik Pengaruh Tanpa Nilai Kontinu dan Tanpa Sebaran Banyak Nilai Terhadap *PC*

Pada gambar 4.29 tampak pola grafik yang terus menaik. Dimulai dari persentase data-latih 60% dengan *PC*=3,29 dan berakhir pada persentase data-latih 100% dengan *PC*=28,73. Untuk lebih lengkapnya data pengaruh *Crx* tanpa nilai kontinu dan sebaran banyak nilai terhadap *PC*, bisa dilihat di lampiran. Pola grafik yang terus menaik seperti pada gambar 4.29 tersebut, menandakan pengaruh *Crx* tanpa nilai kontinu dan tanpa sebaran banyak nilai terhadap *PC* yang tinggi. Sehingga untuk mendapatkan nilai *PC* yang tinggi pada data-contoh *Crx*, dengan lebih mengkondisikan data atribut berisi nilai kontinu dan atribut berisi sebaran banyak nilai.

g. Pengaruh data-contoh *Crx* tanpa nilai hilang, tanpa nilai kontinu, dan tanpa sebaran banyak nilai terhadap *PC*




Gambar 4.30 Grafik Pengaruh Tanpa Nilai Hilang, Tanpa Nilai Kontinu, dan Tanpa Sebaran Banyak Nilai Terhadap *PC*

Pada gambar 4.30 tampak pola grafik yang terus menaik. Kenaikan pola grafik dimulai dari persentase data-latih 60% dengan *PC*=11,87 sampai berakhir pada persentase data-latih 100% dengan *PC*=30,7. Untuk lebih lengkapnya data pengaruh *Crx* tanpa nilai hilang, tanpa nilai kontinu, dan tanpa sebaran banyak nilai terhadap *PC*, bisa dilihat di lampiran. Pola grafik yang menaik ini menandakan bahwa rendahnya nilai *PC* pada data-contoh *Crx* disebabkan adanya nilai hilang, nilai kontinu, dan atribut dengan sebaran banyak nilai.

Dari ketujuh kemungkinan uji coba terhadap pengaruh nilai hilang, nilai kontinu, dan atribut berisi sebaran banyak nilai terhadap *PC* maka didapatkan kesimpulan akhir, bahwa rendahnya nilai *PC* pada data-contoh *Crx* ketika melakukan pengujian pengaruh persentase data-latih terhadap *PC*, ternyata dipengaruhi adanya nilai hilang, nilai kontinu, dan atribut berisi sebaran banyak nilai; seperti yang telah diujikan. Yang paling tampak besar pengaruhnya terhadap rendahnya nilai *PC* pada data-contoh *Crx* yakni adanya atribut yang berisi sebaran banyak nilai (atribut *A6* dan atribut *A7*), sehingga untuk mendapatkan hasil optimal berupa aturan dari data-contoh *Crx* yakni dengan mengkondisikan data-contoh *Crx* pada atribut *A6* dan *A7*, yang bisa dilakukan dengan menghilangkan kedua atribut tersebut dari data-contoh *Crx*, seperti telah dijelaskan pada subbab 2.1.5.

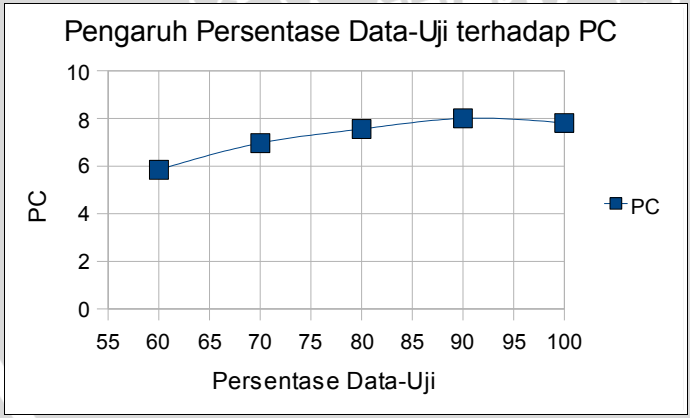
#### 4.4.2.6 Pengaruh Persentase Data-Uji Terhadap Rerata *PC*

Pengujian parameter persentase data-uji terhadap rerata PC telah dilakukan sebanyak 10 kali untuk rentang persentase data-uji antara 60%-100% dengan persentase data-latih senilai 67%, dan jumlah populasi 20. Dalam tiap-tiap pengujian pengaruh persentase data-uji terhadap *PC* (sebanyak 3 x 60%-100%), dihasilkan aturan-aturan yang berbeda, yang selanjutnya digunakan untuk menghitung persentase kebenaran aturan dari data-uji (*PC*). Hasil dari pengujian persentase data-uji terhadap *PC* ditunjukkan pada tabel 4.7.

Tabel 4.7 Data Pengaruh Persentase Data-Uji Terhadap Rerata *PC*

| % nTest | PC | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | Rerata |
| 60 | 6,88 | 8,7 | 4,47 | 9,9 | 9,55 | 7,97 | 4,59 | 1,57 | 1,21 | 3,63 | 5,85 |
| 70 | 4,14 | 8,28 | 3,73 | 16,57 | 1,04 | 13,25 | 5,08 | 2,38 | 3,63 | 11,6 | 6,97 |
| 80 | 1,18 | 3,99 | 13,95 | 13,86 | 5,89 | 15,95 | 9,15 | 5,35 | 4,62 | 1,63 | 7,56 |
| 90 | 4,99 | 11,03 | 3,79 | 15,06 | 4,03 | 14,41 | 10,31 | 12,64 | 2,18 | 1,61 | 8 |
| 100 | 21,31 | 10,58 | 17,68 | 1,67 | 7,25 | 1,74 | 9,13 | 1,45 | 2,83 | 4,5 | 7,81 |

Sedang representasi dari tabel 4.7 berupa grafik ditunjukkan pada gambar 4.31.




Gambar 4.31 Grafik Pengaruh Persentase Data-Uji Terhadap Rerata *PC*

Pada gambar 4.31 ditunjukkan tentang grafik pengaruh persentase data-uji terhadap rerata *PC*. Dimulai dari persentase data-uji 60% dengan rerata *PC*=5,85. Kemudian pola grafik mengalami kenaikan terus sampai persentase data-uji 90% dengan *PC*=8. Terakhir, pola grafik sedikit mengalami penurunan menuju jumlah populasi ke-100 dengan nilai *PC*=7,81. Akan tetapi secara garis besar, pola grafik pada pengujian pengaruh persentase data-uji terhadap rerata *PC* mengalami kenaikan. Pengujian kali ini menghasilkan nilai rerata *PC* terbaik=8 pada jumlah populasi ke-90.

Kenaikan pola grafik pada gambar 4.31 disebabkan tingginya persentase kebenaran aturan terhadap data-uji (*PC*) berbanding lurus mengikuti kenaikan persentase data-uji. Pernyataan ini sudah menjadi kesimpulan umum dari banyak penelitian tentang metode klasifikasi.

Dalam pengujian pengaruh parameter persentase data-latih terhadap *PC* (lihat subbab 4.4.2.5) didapatkan kesimpulan, bahwa untuk data-contoh *Crx* terdapat nilai hilang, nilai kontinu, dan atribut dengan sebaran banyak nilai yang menyebabkan rendahnya nilai *PC*. Kenaikan persentase data-latih mengakibatkan nilai *PC* yang cenderung naik-turun. Setelah dilakukan pengujian terhadap data-contoh *Crx* tanpa nilai hilang, tanpa nilai kontinu, dan tanpa atribut dengan sebaran banyak nilai yakni pada subbab 4.4.2.5 poin g, didapatkan kenaikan yang berbanding lurus antara nilai *PC* dan persentase data-latih.

Selanjutnya untuk menguji persentase antara data-uji dengan *PC*, dipakai aturan terbaik yang dihasilkan dari pengujian pada subbab 4.4.2.5 poin g. Dengan tujuan mengetahui pengaruh aturan terbaik tersebut pada pengujian persentase data-uji terhadap *PC*. Aturan terbaik yang dimaksud yaitu (lihat gambar 4.30) pada rerata *PC*=30,7 untuk persentase data-latih 100%, dengan rincian untuk masing-masing jenis nilai atribut tujuan 'f' dan 't' :

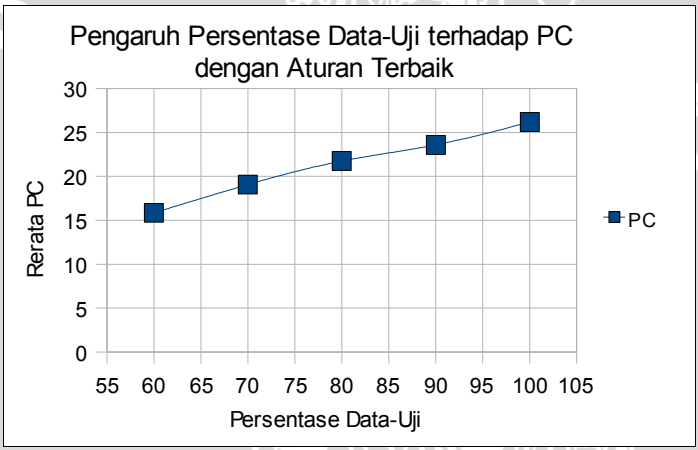
a. *IF A12<>'t' AND Class='-' THEN A9='f'*
*b. IF A10<>'f' AND A12<>'f' AND Class<>'-' THEN A9='t'*

Untuk pengujian persentase data-uji terhadap *PC* menggunakan aturan terbaik tersebut, ditunjukkan pada tabel 4.8.

Tabel 4.8 Data Pengaruh Persentase Data-Uji Terhadap Rerata *PC* dengan Aturan Terbaik

| % Data-Uji | PC | | Rerata PC |
|---|---|---|---|
| | 'f' | t' | |
| 60 | 24,04 | 7,66 | 15,85 |
| 70 | 25,57 | 12,56 | 19,07 |
| 80 | 25,57 | 17,92 | 21,75 |
| 90 | 25,73 | 21,44 | 23,58 |
| 100 | 25,88 | 26,49 | 26,19 |

Sedang representasi dari tabel 4.8 berupa grafik ditunjukkan pada gambar 4.32.




Gambar 4.32 Grafik Pengaruh Persentase Data-Uji Terhadap Rerata *PC* dengan Aturan Terbaik

Berbeda dengan gambar 4.32 yang menunjukkan pola grafik yang menurun menuju persentase data-uji 70% dengan rerata *PC*=5,38 lalu menaik sampai pada persentase data-uji 100% dengan rerata *PC*=11,02; pada gambar 4.32 menunjukkan pola grafik yang terus menaik dimulai dari persentase data-uji 60% dengan *PC*=15,85 sampai persentase data-uji paling akhir 100% dengan *PC*=26,19.

Kenaikan pola grafik pada gambar 4.32 mendukung keputusan pada langkah pengujian sebelumnya (pengujian pengaruh persentase data-uji terhadap *PC* dengan data-contoh *Crx* berisi nilai hilang, nilai kontinu, dan atribut berisi sebaran banyak nilai), yakni tingginya persentase kebenaran aturan terhadap data-uji (*PC*) berbanding lurus mengikuti kenaikan persentase data-uji, hanya saja nilai *PC* yang dihasilkannya yakni *PC*=11,02, lebih kecil daripada langkah pengujian setelahnya yakni *PC*=26,19 (dari pengujian pengaruh persentase data-uji terhadap *PC* dengan data-contoh *Crx* tanpa nilai hilang, nilai kontinu, dan atribut berisi sebaran banyak nilai).

#### 4.4.2.7 Aturan Klasifikasi Data-Contoh *Crx*

Pada subbab ini akan disertakan contoh aturan-aturan klasifikasi yang dihasilkan sistem dari data-contoh *Crx* untuk masing-masing jenis nilai atribut tujuan *A9* yakni nilai 'f' pada tabel 4.9 dan nilai 't' dalam tabel 4.10. Aturan-aturan klasifikasi ini berjumlah 20 aturan, yang dihasilkan dari percobaan ke-2 untuk pengujian parameter persentase data-uji senilai 100%, dengan persentase data-latih 67%, jumlah populasi senilai 20, dan proses pembentukan generasi turunan berhenti pada generasi ke-22.

Tabel 4.9 Aturan Data-Contoh *Crx* Atribut Tujuan *A9* Nilai 'f'

| No. | Fitness | Aturan |
|---|---|---|
| 1 | 0,92 | A3='low' AND A5='g' AND A13<>'g' |
| 2 | 0,92 | A3='low' AND A13<>'g' AND A14='low' |
| 3 | 0,89 | A3='low' AND A4<>'l' AND A14='low' AND Class='-' |
| 4 | 0,84 | A3='low' AND A4='u' AND A5='g' AND A7<>'bb' AND Class='-' |
| 5 | 0,8 | A3='med' AND A11='low' AND Class='-' |
| 6 | 0,78 | A3='low' |
| 7 | 0,78 | A5='g' AND A13='p' |
| 8 | 0,78 | A3='low' AND A5<>'gg' |
| 9 | 0,76 | A1='a' AND A3='low' AND A5='p' |
| 10 | 0,76 | A1<>'a' AND A3='low' AND A5='p' AND A8='low' AND Class<>'+' |
| 11 | 0,76 | A2<>'low' AND A4<>'y' AND A7='ff' AND A12='f' AND Class='-' |

| | | |
|---|---|---|
| 12 | 0,75 | A6='w' AND A7='v AND A14='high' AND A15='high' AND Class='-' |
| 13 | 0,75 | A3='med' AND A13<>'g' |
| 14 | 0,74 | A15='high' AND Class='-' |
| 15 | 0,74 | A11<>'high' AND A12<>'f' AND A13<>'g' AND A15<>'low' |
| 16 | 0,74 | A3='med' AND A5<>'p' AND A6<>'d' AND A14='high' |
| 17 | 0,73 | A5='g' AND A8='low' AND A11<>'med' AND Class='-' |
| 18 | 0,73 | A2<>'low' AND A11<>'high' AND A12='f' AND A15='high' AND Class='-' |
| 19 | 0,73 | A1='b' AND A2<>'med' AND A8<>'med' AND A10<>'f' AND A13='s' AND A14<>'med' |
| 20 | 0,72 | A3='low' AND A5='g' AND A7<>'bb' |

Tabel 4.10 Aturan Data-Contoh *Crx* Atribut Tujuan *A9* Nilai 't'

| No. | Fitness | Aturan |
|---|---|---|
| 1 | 0,86 | A4<>'l' AND A12<>'f' AND Class<>'-' |
| 2 | 0,83 | A7='bb' AND A10<>'f' AND Class='+' |
| 3 | 0,79 | A5='g' AND A7<>'v AND A11<>'med' AND A14<>'high' |
| 4 | 0,79 | A8<>'low' AND A12='f' AND A14='low' AND A15='low' |
| 5 | 0,78 | A7='z' |
| 6 | 0,77 | A2<>'low' AND A4<>'y' AND A7<>'o' AND A8<>'low' AND Class<>'-' |
| 7 | 0,74 | A6<>'ff' AND A12<>'t' AND Class<>'-' |
| 8 | 0,74 | A14<>'high' |
| 9 | 0,73 | A7='bb' AND A10<>'f' AND A14='high' |
| 10 | 0,72 | A6='cc' AND A12<>'t' AND A13<>'p' |
| 11 | 0,71 | A6<>'aa' AND A7='bb' AND A10<>'f' AND A12<>'f' AND A14='low' |
| 12 | 0,71 | A6<>'c' AND A12<>'f' AND A14<>'low' |
| 13 | 0,7 | A6<>'e' AND A10<>'f' AND A15<>'low' |
| 14 | 0,7 | A12<>'t' AND A14<>'low' AND Class<>'-' |
| 15 | 0,7 | A1='a' AND A3='high' AND A5='g' AND A12='f' AND A15='low' |
| 16 | 0,69 | A5='g' AND A6<>'aa' |
| 17 | 0,68 | A6='cc' AND A7<>'bb' AND A8<>'low' AND A15='high' |
| 18 | 0,68 | A7<>'bb' AND A13='g' AND Class<>'-' |
| 19 | 0,68 | A1='a' AND A6='cc' AND A13<>'s' AND A15='low' AND Class<>'-' |
| 20 | 0,68 | A10='f' AND A11='low' AND A13='g' AND Class='+' |

# BAB V
# KESIMPULAN DAN SARAN

## 5.1 Kesimpulan

Kesimpulan yang didapatkan selama pengerjaan skripsi ini yakni :

1. Kenaikan nilai parameter jumlah populasi mempengaruhi tingginya nilai kelayakan *antecedent* (*AI*) dari suatu aturan. Nilai *AI* paling maksimal pada jumlah populasi ke-80 dengan nilai *AI*=0,99.
2. Perubahan nilai parameter jumlah populasi tidak mempengaruhi nilai kelayakan *consequent* (*CI*) dari aturan. Begitu pula penentuan persentase nilai data-latih dengan nilai tertentu atau tidak ditentukan dengan nilai tertentu, tidak berimbas sama sekali pada perubahan nilai kelayakan *CI*.
3. Kenaikan nilai parameter jumlah populasi mempengaruhi tingginya nilai prediksi keakuratan (*PA*) dari suatu aturan. Nilai *PA* paling maksimal pada jumlah populasi ke-60 sampai ke-80 dengan nilai *PA*=0,96.
4. Kenaikan nilai parameter jumlah populasi mempengaruhi tingginya nilai *fitness* dari aturan. Nilai *fitness* paling maksimal pada jumlah populasi ke-70 dengan nilai *fitness*=0,93.
5. Pada data-contoh *Crx*, kenaikan nilai persentase data-latih menghasilkan rendahnya nilai rerata persentase kebenaran aturan dari data-uji (*PC*). Hal ini disebabkan adanya nilai hilang, nilai kontinu, dan atribut berisi sebaran banyak nilai; sedang yang paling besar pengaruhnya terhadap rendahnya nilai *PC* pada data-contoh *Crx* yakni atribut berisi sebaran banyak nilai.
6. Kenaikan nilai persentase data-uji mempengaruhi terhadap kenaikan nilai rerata *PC,* terutama pada posisi puncak persentase data-uji 100% dengan nilai rerata *PC*=11,02.

## 5.2 Saran

Saran yang diberikan setelah pengerjaan skripsi ini yakni :

1. Pengujian teknik klasifikasi menggunakan Algoritma Genetika pada data-contoh *Crx*, dihasilkan nilai *AI*, *CI*, *PA*, *Fitness*, dan *PC* yang tinggi-rendahnya dipengaruhi oleh nilai hilang, nilai kontinu, dan banyaknya jumlah baris data. Sehingga perlu dilakukan

penelitian dengan data-contoh yang berbeda yang lebih beragam dalam jumlah nilai hilang, jumlah atribut kontinu, dan jumlah barisnya, untuk mengetahui seberapa besar pengaruhnya terhadap tinggi-rendahnya nilai *AI, CI, PA, Fitness*, dan *PC*. Apabila didapatkan adanya pengaruh nilai hilang, nilai kontinu, dan banyaknya jumlah baris data terhadap nilai *AI, CI, PA, Fitness*, dan *PC*, maka bisa dilakukan proses pengkondisian data-contoh yang lebih tepat untuk mendapatkan aturan *Mining* yang lebih layak dan akurat dari Algoritma Genetika.

2. Seusai penelitian menggunakan data-contoh *Crx*, ditemukan adanya faktor lain yang menyebabkan rendahnya pengaruh nilai *PC* dari pengujian parameter persentase data-latih, selain karena pengaruh nilai hilang dan nilai kontinu. Yakni adanya atribut berisi sebaran banyak nilai. Oleh karenanya, perlu dilakukan penelitian lebih lanjut pada data-contoh yang berbeda, untuk mengetahui faktor-faktor lain yang menyebabkan rendahnya nilai *PC* dari pengujian persentase data-latih, selain karena pengaruh nilai hilang, nilai kontinu, dan atribut berisi sebaran banyak nilai.
3. Pengujian teknik klasifikasi dengan Algoritma Genetika pada skripsi ini telah dilakukan dengan batasan-batasan parameter persentase data-latih, persentase data-uji, jumlah populasi, dan jumlah maksimal pembentukan generasi. Yakni untuk parameter persentase data-latih dan persentase data-uji antara rentang 60%-100%, jumlah populasi=20, dan jumlah maksimal pembentukan generasi=30. Maka disarankan untuk melakukan pengujian-pengujian dengan batasan jumlah populasi dan jumlah generasi yang lebih besar lagi dalam beberapa kali percobaan.

# DAFTAR PUSTAKA


Alatas, Bilal dan Arslan, Ahmet. 2002. *Mining of Interesting Prediction Rules with Uniform Two-Level Genetic Algorithm*. International Journal of Computational Intelligence Volume 1 Number 4

Daniel, T. Larose. 2005. *Discovering Knowledge in Data An Introduction to Data Mining*. Central Connecticut State University. A John Wiley & Sons, inc., Publication.

Han, Jiawei dan Kamber, Micheline. 2006. *Data Mining : Concepts and Techniques Second Edition, University of Illinois at Urbana-Champaign*. Morgan Kaufmann Publishers San Francisco.

Kantardzic, Mehmed. 2003. *Data Mining: Concepts, Models, Methods, and Algorithms*. John Wiley & Sons.

Korkut Koray G¨UNDOGAN, Bilal ALATAS, dkk. 2004. *Mining Classication Rules by Using Genetic Algorithms with Non-random Initial Population and Uniform Operator*. Turk J Elec Engin, VOL.12, NO.1 2004. Frat University, Department of Computer Engineering.

Machine Learning Repository. http://archive.ics.uci.edu/. University of California at Irvine. Didownload tanggal 24 Mei 2008.

Melanie, Mitchell. 1999. *An Introduction to Genetic Algorithms*. A Bradford Book. MIT Press Cambridge, Massachusetts London.

M.V.Fidelis, H.S.Lopes dan A.A.Freitas. 1999. *Discovering Comprehensible Classification Rules with a Genetic Algorithm*. UEPG, CPD, Praça Santos Andrade, Ponta Grossa.

Nurjaya, Wahyu. 2006. *Analisis Proses Word Matching Problem Menggunakan Algoritma Genetika*. Majalah Ilmiah Unikom, Vol. 6. Jurusan Manajemen Informatika Univ. Komputer Indonesia.

Obitko, Marek. *Genetic Algorithms*. http://www.obitko.com/tutorials/genetic-algorithms/. Didownload tanggal 15 Agustus 2008.

Sumantrika, Ngurah Putu. 2005. *Data Mining Task Klasifikasi Menggunakan Algoritma Genetika*. Skripsi. Jurusan Teknik Informatika Sekolah Tinggi Teknologi Telkom Bandung.

Tan, Steinbach dan Kumar. 2004. *Lecture Notes for Chapter 1 Introduction to Data Mining*. Michigan State University dan University of Minnesota.


UNIVERSITAS BRAWIJAYA

# LAMPIRAN

## Lampiran 1. Deskripsi Data-Contoh *Crx*

### 1. Info Atribut Data-Contoh *Crx*

```
+,-.
A1  : b, a.
A2  : continuous.
A3  : continuous.
A4  : u, y, l, t.
A5  : g, p, gg.
A6  : c, d, cc, i, j, k, m, r, q, w, x, e, aa, ff.
A7  : v, h, bb, j, n, z, dd, ff, o.
A8  : continuous.
A9  : t, f.
A10 : t, f.
A11 : continuous.
A12 : t, f.
A13 : g, p, s.
A14 : continuous.
A15 : continuous.
```

### 2. Deskripsi Data-Contoh *Crx*

```
Number of instance : 690
Number of attributes : 15(without the class atribute)
Number of continuous attributes : 6
Number of missing value : 67
```

### 3. Beberapa Baris Data-Contoh *Crx*

```
b,30.83,0,u,g,w,v,1.25,t,t,01,f,g,00202,0,+
a,58.67,4.46,u,g,q,h,3.04,t,t,06,f,g,00043,560,+
a,24.50,0.5,u,g,q,h,1.5,t,f,0,f,g,00280,824,+
b,27.83,1.54,u,g,w,v,3.75,t,t,05,t,g,00100,3,+
b,20.17,5.625,u,g,w,v,1.71,t,f,0,f,s,00120,0,+
b,32.08,4,u,g,m,v,2.5,t,f,0,t,g,00360,0,+
b,33.17,1.04,u,g,r,h,6.5,t,f,0,t,g,00164,31285,+
a,22.92,11.585,u,g,cc,v,0.04,t,f,0,f,g,00080,1349,+
b,54.42,0.5,y,p,k,h,3.96,t,f,0,f,g,00180,314,+
```

b,42.50,4.915,y,p,w,v,3.165,t,f,0,t,g,00052,1442,+
b,22.08,0.83,u,g,c,h,2.165,f,f,0,t,g,00128,0,+
b,29.92,1.835,u,g,c,h,4.335,t,f,0,f,g,00260,200,+
a,38.25,6,u,g,k,v,1,t,f,0,t,g,00000,0,+
b,48.08,6.04,u,g,k,v,0.04,f,f,0,f,g,00000,2690,+
a,45.83,10.5,u,g,q,v,5,t,t,07,t,g,00000,0,+
b,36.67,4.415,y,p,k,v,0.25,t,t,10,t,g,00320,0,+
b,28.25,0.875,u,g,m,v,0.96,t,t,03,t,g,00396,0,+
a,23.25,5.875,u,g,q,v,3.17,t,t,10,f,g,00120,245,+
b,21.83,0.25,u,g,d,h,0.665,t,f,0,t,g,00000,0,+
a,19.17,8.585,u,g,cc,h,0.75,t,t,07,f,g,00096,0,+
b,25.00,11.25,u,g,c,v,2.5,t,t,17,f,g,00200,1208,+
b,23.25,1,u,g,c,v,0.835,t,f,0,f,s,00300,0,+
a,47.75,8,u,g,c,v,7.875,t,t,06,t,g,00000,1260,+
a,27.42,14.5,u,g,x,h,3.085,t,t,01,f,g,00120,11,+
a,41.17,6.5,u,g,q,v,0.5,t,t,03,t,g,00145,0,+
a,15.83,0.585,u,g,c,h,1.5,t,t,02,f,g,00100,0,+
a,47.00,13,u,g,i,bb,5.165,t,t,09,t,g,00000,0,+
b,56.58,18.5,u,g,d,bb,15,t,t,17,t,g,00000,0,+
b,57.42,8.5,u,g,e,h,7,t,t,03,f,g,00000,0,+
b,42.08,1.04,u,g,w,v,5,t,t,06,t,g,00500,10000,+
b,29.25,14.79,u,g,aa,v,5.04,t,t,05,t,g,00168,0,+
b,42.00,9.79,u,g,x,h,7.96,t,t,08,f,g,00000,0,+
b,49.50,7.585,u,g,i,bb,7.585,t,t,15,t,g,00000,5000,+
a,36.75,5.125,u,g,e,v,5,t,f,0,t,g,00000,4000,+
a,22.58,10.75,u,g,q,v,0.415,t,t,05,t,g,00000,560,+
b,27.83,1.5,u,g,w,v,2,t,t,11,t,g,00434,35,+
b,27.25,1.585,u,g,cc,h,1.835,t,t,12,t,g,00583,713,+
a,23.00,11.75,u,g,x,h,0.5,t,t,02,t,g,00300,551,+
b,27.75,0.585,y,p,cc,v,0.25,t,t,02,f,g,00260,500,+
b,54.58,9.415,u,g,ff,ff,14.415,t,t,11,t,g,00030,300,+
b,34.17,9.17,u,g,c,v,4.5,t,t,12,t,g,00000,221,+
b,28.92,15,u,g,c,h,5.335,t,t,11,f,g,00000,2283,+
b,29.67,1.415,u,g,w,h,0.75,t,t,01,f,g,00240,100,+
b,39.58,13.915,u,g,w,v,8.625,t,t,06,t,g,00070,0,+
b,56.42,28,y,p,c,v,28.5,t,t,40,f,g,00000,15,+
b,54.33,6.75,u,g,c,h,2.625,t,t,11,t,g,00000,284,+
a,41.00,2.04,y,p,q,h,0.125,t,t,23,t,g,00455,1236,+
b,31.92,4.46,u,g,cc,h,6.04,t,t,03,f,g,00311,300,+
b,41.50,1.54,u,g,i,bb,3.5,f,f,0,f,g,00216,0,+
b,23.92,0.665,u,g,c,v,0.165,f,f,0,f,g,00100,0,+
a,25.75,0.5,u,g,c,h,0.875,t,f,0,t,g,00491,0,+
b,26.00,1,u,g,q,v,1.75,t,f,0,t,g,00280,0,+
b,37.42,2.04,u,g,w,v,0.04,t,f,0,t,g,00400,5800,+
b,34.92,2.5,u,g,w,v,0,t,f,0,t,g,00239,200,+
b,34.25,3,u,g,cc,h,7.415,t,f,0,t,g,00000,0,+
b,23.33,11.625,y,p,w,v,0.835,t,f,0,t,g,00160,300,+

```
b,23.17,0,u,g,cc,v,0.085,t,f,0,f,g,00000,0,+
b,44.33,0.5,u,g,i,h,5,t,f,0,t,g,00320,0,+
b,35.17,4.5,u,g,x,h,5.75,f,f,0,t,s,00711,0,+
b,43.25,3,u,g,q,h,6,t,t,11,f,g,00080,0,+
b,56.75,12.25,u,g,m,v,1.25,t,t,04,t,g,00200,0,+
b,31.67,16.165,u,g,d,v,3,t,t,09,f,g,00250,730,+
a,23.42,0.79,y,p,q,v,1.5,t,t,02,t,g,00080,400,+
a,20.42,0.835,u,g,q,v,1.585,t,t,01,f,g,00000,0,+
b,26.67,4.25,u,g,cc,v,4.29,t,t,01,t,g,00120,0,+
b,34.17,1.54,u,g,cc,v,1.54,t,t,01,t,g,00520,50000,+
a,36.00,1,u,g,c,v,2,t,t,11,f,g,00000,456,+
b,25.50,0.375,u,g,m,v,0.25,t,t,03,f,g,00260,15108,+
b,19.42,6.5,u,g,w,h,1.46,t,t,07,f,g,00080,2954,+
b,35.17,25.125,u,g,x,h,1.625,t,t,01,t,g,00515,500,+
b,32.33,7.5,u,g,e,bb,1.585,t,f,0,t,s,00420,0,-
b,34.83,4,u,g,d,bb,12.5,t,f,0,t,g,?,0,-
a,38.58,5,u,g,cc,v,13.5,t,f,0,t,g,00980,0,-
b,44.25,0.5,u,g,m,v,10.75,t,f,0,f,s,00400,0,-
b,44.83,7,y,p,c,v,1.625,f,f,0,f,g,00160,2,-
b,20.67,5.29,u,g,q,v,0.375,t,t,01,f,g,00160,0,-
b,34.08,6.5,u,g,aa,v,0.125,t,f,0,t,g,00443,0,-
a,19.17,0.585,y,p,aa,v,0.585,t,f,0,t,g,00160,0,-
b,21.67,1.165,y,p,k,v,2.5,t,t,01,f,g,00180,20,-
b,21.50,9.75,u,g,c,v,0.25,t,f,0,f,g,00140,0,-
b,49.58,19,u,g,ff,ff,0,t,t,01,f,g,00094,0,-
a,27.67,1.5,u,g,m,v,2,t,f,0,f,s,00368,0,-
b,39.83,0.5,u,g,m,v,0.25,t,f,0,f,s,00288,0,-
a,?,3.5,u,g,d,v,3,t,f,0,t,g,00300,0,-
b,27.25,0.625,u,g,aa,v,0.455,t,f,0,t,g,00200,0,-
b,37.17,4,u,g,c,bb,5,t,f,0,t,s,00280,0,-
b,?,0.375,u,g,d,v,0.875,t,f,0,t,s,00928,0,-
..
..
..
b,36.42,0.75,y,p,d,v,0.585,f,f,0,f,g,00240,3,-
b,40.58,3.29,u,g,m,v,3.5,f,f,0,t,s,00400,0,-
b,21.08,10.085,y,p,e,h,1.25,f,f,0,f,g,00260,0,-
a,22.67,0.75,u,g,c,v,2,f,t,02,t,g,00200,394,-
a,25.25,13.5,y,p,ff,ff,2,f,t,01,t,g,00200,1,-
b,17.92,0.205,u,g,aa,v,0.04,f,f,0,f,g,00280,750,-
b,35.00,3.375,u,g,c,h,8.29,f,f,0,t,g,00000,0,-
```

**Lampiran 2. Hasil Uji Coba Pengaruh Nilai Hilang, Nilai Kontinu, dan Atribut Berisi Sebaran Banyak Nilai pada *PC***

Masing-masing percobaan dilakukan sekali dengan parameter jumlah populasi=20 dan persentase data-uji=33% untuk data-contoh *Crx* dengan atribut tujuan *A9* untuk jenis nilai 'f' dan 't'.

a. Pengaruh data-contoh *Crx* tanpa nilai hilang terhadap *PC*

| % nTrain | PC 'f' | PC 't' | rerata PC |
|---|---|---|---|
| 60 | 0,93 | 26,98 | 13,96 |
| 70 | 2,33 | 26,98 | 14,66 |
| 80 | 19,07 | 20,93 | 20 |
| 90 | 1,4 | 3,26 | 2,33 |
| 100 | 5,58 | 0,93 | 3,26 |

b. Pengaruh data-contoh *Crx* tanpa nilai kontinu terhadap *PC*

| % nTrain | PC 'f' | PC 't' | rerata PC |
|---|---|---|---|
| 60 | 1,32 | 26,32 | 13,82 |
| 70 | 9,21 | 3,07 | 6,14 |
| 80 | 1,32 | 10,09 | 5,71 |
| 90 | 13,6 | 5,7 | 9,65 |
| 100 | 35,96 | 7,02 | 21,49 |

c. Pengaruh data-contoh *Crx* tanpa sebaran banyak nilai terhadap *PC*

| % nTrain | PC 'f' | PC 't' | rerata PC |
|---|---|---|---|
| 60 | 4,65 | 26,98 | 15,82 |
| 70 | 10,7 | 17,21 | 13,96 |
| 80 | 0,47 | 26,98 | 13,73 |
| 90 | 1,4 | 26,98 | 14,19 |
| 100 | 5,58 | 13,02 | 9,3 |

d. Pengaruh data-contoh *Crx* tanpa nilai hilang dan nilai kontinu terhadap *PC*

| % nTrain | PC 'f' | PC 't' | rerata PC |
|---|---|---|---|
| 60 | 2,33 | 26,98 | 14,66 |
| 70 | 20,93 | 26,98 | 23,96 |
| 80 | 24,19 | 37,21 | 30,7 |
| 90 | 19,53 | 13,02 | 16,28 |
| 100 | 35,96 | 7,02 | 21,49 |

e. Pengaruh data-contoh *Crx* tanpa nilai hilang dan sebaran banyak nilai terhadap *PC*

| % nTrain | PC 'f' | PC 't' | rerata PC |
|---|---|---|---|
| 60 | 0,93 | 26,98 | 13,96 |
| 70 | 0,93 | 23,26 | 12,1 |
| 80 | 6,51 | 23,26 | 14,89 |
| 90 | 3,26 | 22,79 | 13,03 |
| 100 | 1,86 | 26,05 | 13,96 |

f. Pengaruh data-contoh *Crx* tanpa nilai kontinu dan sebaran banyak nilai terhadap *PC*

| % nTrain | PC 'f' | PC 't' | rerata PC |
|---|---|---|---|
| 60 | 1,75 | 4,82 | 3,29 |
| 70 | 1,75 | 4,82 | 3,29 |
| 80 | 1,75 | 14,91 | 8,33 |
| 90 | 44,3 | 10,09 | 27,2 |
| 100 | 44,3 | 13,16 | 28,73 |

g. Pengaruh data-contoh *Crx* tanpa nilai hilang, nilai kontinu, dan sebaran banyak nilai terhadap *PC*

| % nTrain | PC 'f' | PC 't' | rerata PC |
|---|---|---|---|
| 60 | 0,47 | 23,26 | 11,87 |
| 70 | 24,19 | 17,21 | 20,7 |
| 80 | 20,93 | 26,98 | 23,96 |
| 90 | 20,93 | 26,98 | 23,96 |
| 100 | 34,41 | 26,98 | 30,7 |