Implementasi Algoritma CART dan CMAR untuk Menentukan Status Resiko Kredit

Skripsi

Oleh :

RESTU FUJI RAHAYU
0610963050

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG
2011

Implementasi Algoritma CART dan CMAR untuk Menentukan Status Resiko Kredit

Skripsi

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

Oleh :

RESTU FUJI RAHAYU
0610963050

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG
2011

LEMBAR PENGESAHAN SKRIPSI

Implementasi Algoritma CART dan CMAR untuk Menentukan
Status Resiko Kredit

Oleh :

RESTU FUJI RAHAYU
0610963050-96

Setelah dipertahankan di depan Majelis Penguji
Pada tanggal 19 Januari 2011

dan dinyatakan memenuhi syarat untuk memperoleh gelar
Sarjana Komputer dalam bidang Ilmu Komputer

Pembimbing I,	Pembimbing II,
Candra Dewi, S.Kom.,MSc	Dewi Yanti L.,S.Kom,M.Kom
NIP. 197711142003122001	NIP. 198111162005012004

Mengetahui,
Ketua Jurusan Matematika
Fakultas MIPA Universitas Brawijaya

Dr. Abdul Rouf A., MSc
NIP. 196709071992031001

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini :

Nama	: Restu Fuji Rahayu
NIM	: 0610963050-96
Jurusan	: Matematika
Program Studi	: Ilmu Komputer
Penulis skripsi berjudul	: Implementasi Algoritma CART dan CMAR untuk Menentukan Status Resiko Kredit

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 dikemudian 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, 19 Januari 2011
Yang menyatakan,

Restu Fuji Rahayu
NIM. 0610963050-96

Implementasi Algoritma CART dan CMAR untuk Menentukan Status Resiko Kredit

ABSTRAK

Dalam dunia perbankan, pemberian kredit merupakan kegiatan usaha yang mengandung resiko tinggi yang berpengaruh terhadap kelangsungan usaha perbankan. Oleh karena itu, seorang pembuat keputusan *(decision maker)* harus melakukan evaluasi untuk pemohon kredit secara objektif, akurat dan konsisten. Pada pengambilan keputusan untuk pemohon kredit, dapat dilakukan dengan *data mining* menggunakan metode klasifikasi.

Dalam penelitian ini, digunakan algoritma CART *(Classification and Regression Tree)* dan algoritma CMAR *(Classification Based on Multiple Class-Association Rules).* CART merupakan *binary tree* yang hanya memilikai dua cabang untuk setiap *decision node*, sehingga *decision node* harus dipartisi menjadi dua bagian dan membentuk *candidate split*. *Candidate split* dipilih untuk penyusunan inisial partisi pada *root node* dan *decision node.* Kriteria pemilihan tersebut berdasarkan nilai *goodness of split* (Φ(s/t)) terbesar. Pada algoritma CMAR, klasifikasi dilakukan dengan memangkas beberapa aturan dan hanya memilih *subset* aturan klasifikasi berkualitas tinggi, kemudian menemukan aturan lengkap melalui *support* dan *confidence*.

Dari hasil analisa, didapatkan tingkat akurasi sebesar 68% untuk algoritma CART dan 80% untuk algoritma CMAR. Sedangkan waktu kinerja program, 2127.84 *ms* untuk algoritma CART dan 949.2 *ms* untuk algoritma CMAR. Dari hal tersebut dapat disimpulkan bahwa algoritma CMAR lebih layak dipakai untuk penentuan status resiko kredit dibandingkan algoritma CART.

Key Words : Kredit, Klasifikasi, *Decision Tree,* CART, CMAR.

Implementation CART and CMAR Algorithm to Determine Status of Credit Risk

ABSTRACT

Business lending is an activity that contains a high risk for effect on the continuity of the banking business. Therefore, a decision maker should do evaluation for credit applicants objectively, accurately and consistently. In the decision making, the decision to loan applicants, it can be done with data mining using classification methods.

In this research, using CART (Classification and Regression Tree) algorithm and CMAR (Classification Based on Multiple Class) algorithm. CART is a binary tree that only has two branches for each decision node, so the decision nodes must be partitioned two parts and form a candidate split. Candidate split selected for the preparation of the initial split partition for the root nodes and decision nodes. Selection criteria were based on the largest value of *goodness of split* (Φ(s/t)). At the CMAR algorithm, classification is done be cutting a few rules and choose only a subset of high-quality classification rules, then fins the complete rules through the support and confidence.

From the analysis result, the accuracy rate of 68% for the CART algorithm and 80% for CMAR algorithm. While the performance of programs, 2127.84 ms for CART algorithm and 949.2 ms for CMAR algorithm. From that, can be concluded that CMAR algorithm more proper to use for determining the status of credit risk than CART algorithm.

Keywords: Credit, Classification, Decision Tree, CART, CMAR.

KATA PENGANTAR

Alhamdulillahi rabbil 'alamin. Puji syukur penulis panjatkan kehadirat Allah SWT, karena atas segala rahmat dan limpahan hidayahnya, Tugas Akhir yang berjudul "Implementasi Algoritma CART dan CMAR untuk Menentukan Status Resiko Kredit" ini dapat terselesaikan. Skripsi ini disusun dan diajukan sebagai syarat untuk memperoleh gelar sarjana pada program studi Ilmu Komputer, jurusan Matematika, fakultas MIPA, Universitas Brawijaya.

Semoga Allah melimpahkan rahmat atas Nabi Muhammad SAW, makhluk paling mulia yang senantiasa memberikan cahaya petunjuk, dan atas keluarganya dan sahabat-sahabatnya.

Dalam penyelesaian tugas akhir ini, penulis telah mendapat begitu banyak bantuan baik moral maupun materiil dari banyak pihak. Atas bantuan yang telah diberikan, penulis ingin menyampaikan penghargaan dan ucapan terima kasih yang sedalam-dalamnya kepada:

1. Candra Dewi, S.Kom., M.Sc dan Dewi Yanti Liliana,S.Kom., M. Kom, selaku dosen pembimbing terima kasih atas semua saran, bantuan, kritikan, waktu, dorongan semangat dan bimbingannya.
2. Drs. Marji, MT selaku Ketua Program Studi Ilmu Komputer Universitas Brawijaya Malang.
3. Dr. Abdul Rouf A., MSc selaku Ketua Jurusan Matematika Fakultas MIPA Universitas Brawijaya Malang.
4. Yusi Tyroni Mursityo,S.Kom., PM dan Muhammad Tanzil Furqon, S.Kom selaku dosen penasehat akademik.
5. Segenap Bapak dan Ibu dosen yang telah mendidik dan mengamalkan ilmunya kepada penulis.
6. Segenap staf dan karyawan di Jurusan Matematika Fakultas MIPA Universitas Brawijaya
7. Kedua Orang Tua, Mbak, Mas yang selalu memberikan nasehat. Keponakan-keponakan yang selalu menghibur. Terima kasih atas cinta, kasih sayang, doa, dukungan dan semangat yang tiada henti.
8. Teman-teman Ilkomers, khususnya Ilkomers B 2006 "Babi Inside", terima kasih untuk semangat, cinta dan kebersamaan yang diberikan.

9. Teman-teman kost Tangerang 6 yang telah menjadi keluarga keduaku, terima kasih untuk pengalaman dan cerita hidup yang telah diberikan.
10. Pihak lain yang telah membantu terselesaikannnya Tugas Akhir ini yang tidak bisa penulis sebutkan satu-persatu.

Semoga penulisan laporan tugas akhir ini bermanfaat bagi pembaca sekalian. Dengan tidak lupa kodratnya sebagai manusia, penulis menyadari bahwa tugas akhir ini masih jauh dari kesempurnaan, dan mengandung banyak kekurangan, sehingga dengan segala kerendahan hati penulis mengharapkan kritik dan saran yang membangun dari pembaca.

Malang, Januari 2011

Penulis

DAFTAR ISI

HALAMAN

HALAMAN JUDUL i
HALAMAN PENGESAHAN iii
LEMBAR PERNYATAAN v
ABSTRAK vii
ABSTRACT ix
KATA PENGANTAR xi
DAFTAR ISI xiii
DAFTAR GAMBAR xv
DATAR TABEL xvii
DATAR KODE PROGRAM xix
DAFTAR LAMPIRAN xxi
DATAR RUMUS xxiii

BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 2
1.3 Batasan Masalah 2
1.4 Tujuan 3
1.5 Manfaat 3
1.6 Metode Penyelesaian Masalah 3
1.5 Sistematika Penulisan 4

BAB II TINJAUAN PUSTAKA 5
2.1 Kredit 5
2.2 Data 5
2.3 Data Mining 7
2.3.1 Teknik Data Mining 9
2.3.2 Klasifikasi 10
2.3.3 Decision Tree 11
2.3.4 Associative Classification 11
2.4 Pruning Tree 12
2.5 Algoritma CART 12
2.6 Algoritma CMAR 16
2.7 Akurasi 21

BAB III METODOLOGI DAN PERANCANGAN SISTEM . 23
3.1 Analisis Data 25
3.2 Analisis Sistem 25
3.2.1 Deskripsi Sistem 25
3.2.2 Batasan Sistem 26
3.3 Perancangan Sistem 26
3.3.1 Perancangan Proses 26
3.3.2 Perancangan Basis Data 38
3.3.3 Perhitungan Manual 45
3.4 Perancangan Antarmuka 62
3.5 Perancangan Uji Coba 65
3.5.1 Data Yang Digunakan Dalam Penelitian 65
3.5.2 Pengujian Pada Saat Prediksi Sistem 68
3.5.3 Skenario Uji Coba 70

BAB IV IMPLEMENTASI DAN PEMBAHASAN 73
4.1 Lingkungan Implementasi 73
4.1.1 Lingkungan Implementasi Perangkat Keras 73
4.1.2 Lingkungan Implementasi Perangkat Lunak 73
4.2 Implementasi Program 73
4.2.1 Form Input Data 73
4.2.2 Form Data Training 76
4.2.3 Algoritma CART 78
4.2.4 Algoritma CMAR 84
4.3 Hasil Uji 91
4.4 Analisa Hasil 101

BAB V KESIMPULAN DAN SARAN 103
5.1 Kesimpulan 103
5.2 Saran 103

DAFTAR PUSTAKA 105

LAMPIRAN 109

DAFTAR GAMBAR

Gambar 2.1 Proses KDD 6
Gambar 2.2 Iterasi 1 Root Node 15
Gambar 2.3 Cart Decision Tree 16
Gambar 2.4 Contoh FP-Tree 17
Gambar 2.5 FP-Tree Setelah Penggabungan 19
Gambar 2.6 Proses Konstruksi CR-Tree 20
Gambar 3.1 Alur Penelitian 24
Gambar 3.2 Alur Proses Utama 27
Gambar 3.3 Proses Request Data 28
Gambar 3.4 Proses Learning 29
Gambar 3.5 Konversi Model Tree 30
Gambar 3.6 Alur Proses Algoritma CART 31
Gambar 3.7 Alur Proses Algoritma CMAR 32
Gambar 3.8 Proses Pembangunan FP-Tree 33
Gambar 3.9 Proses Pembangunan CR-Tree 34
Gambar 3.10 Proses Pruning 35
Gambar 3.11 Proses Data Testing 36
Gambar 3.12 Proses Prediksi 37
Gambar 3.13 Struktur Dan Relasi Antar Tabel 45
Gambar 3.14 Iterasi 1 Algoritma CART 52
Gambar 3.15 Decision Tree CART Utuh 54
Gambar 3.16 FP-Tree 58
Gambar 3.17 CR-Tree 60
Gambar 3.18 CR-Tree Setelah Pruning 61
Gambar 3.19 Antarmuka Input Data 62
Gambar 3.20 Antarmuka Data Training 63
Gambar 3.21 Antarmuka Algoritma CART 64
Gambar 3.22 Antarmuka Algoritma CMAR 65
Gambar 4.1 Form Input Data 74
Gambar 4.2 Form Data Training 76
Gambar 4.3 Form Algoritma CART 78
Gambar 4.4 Form Algoritma CMAR 84
Gambar 4.5 Grafik Waktu Algoritma CART 92
Gambar 4.6 Grafik Akurasi Algoritma CART 93
Gambar 4.7 Grafik Waktu Algoritma CMAR 96
Gambar 4.8 Grafik Akurasi Algoritma CMAR 99

Gambar 4.9 Grafik Waktu CART dan CMAR 100
Gambar 4.10 Grafik Akurasi CART dan CMAR 101

DAFTAR TABEL

Tabel 2.1 Contoh Data Training 13
Tabel 2.2 Alternatif Candidate Split 13
Tabel 2.3 Hasil Perhitungan Split 14
Tabel 2.4 Contoh Data Training_2 17
Tabel 2.5 Aturan Yang Ditemukan Pada Training Set 20
Tabel 3.1 Tabel T_Transaction 38
Tabel 3.2 Tabel T_Checking_Account 39
Tabel 3.3 Tabel T_Duration 39
Tabel 3.4 Tabel T_Credit History 39
Tabel 3.5 Tabel T_Purpose 40
Tabel 3.6 Tabel T_Credit_Amount 40
Tabel 3.7 Tabel T_Saving_Account 40
Tabel 3.8 Tabel T_Employment 41
Tabel 3.9 Tabel T_Installment Rate 41
Tabel 3.10 Tabel T_Personal_Status 41
Tabel 3.11 Tabel T_Guarantor 42
Tabel 3.12 Tabel T_Residence_Time 42
Tabel 3.13 Tabel T_Property 42
Tabel 3.14 Tabel T_Age 42
Tabel 3.15 Tabel T_Installment 43
Tabel 3.16 Tabel T_Housing 43
Tabel 3.17 Tabel T_Existing_Credit 43
Tabel 3.18 Tabel T_Job 44
Tabel 3.19 Tabel T_Liable_Person 44
Tabel 3.20 Tabel T_Telp 44
Tabel 3.21 Tabel T_Foreigner 44
Tabel 3.22 Data Training Uji Coba Manual 47
Tabel 3.23 Tabel Candidate Split 48
Tabel 3.24 Perhitungan Split Iterasi 1 50
Tabel 3.25 Perhitungan Split Iterasi 2 52
Tabel 3.26 Atribut dan Jumlah Frekuensi 55
Tabel 3.27 F-List dengan Jumlah Frekuensi 57
Tabel 3.28 F-List yang Dimasukkan Data Training 57
Tabel 3.29 Subset Tiap-Tiap Rule 58
Tabel 3.30 Rule, Support dan Confidence 59
Tabel 3.31 Rule, Support, Confidence Setelah Pruning 61

Tabel 3.32 Rancangan Pengujian CART 70
Tabel 3.27 Rancangan Pengujian CMAR 71
Tabel 4.1 Keterangan Variabel Input Data 74
Tabel 4.2 Deskripsi Fungsi `Class Datatraining` 77
Tabel 4.3 Deskripsi Fungsi `Class Cart` 79
Tabel 4.4 Deskripsi Fungsi `Class Prunerulecart` 79
Tabel 4.5 Deskripsi Fungsi `Class Objrulewincart` 80
Tabel 4.6 Deskripsi Fungsi `Class Cmar` 85
Tabel 4.7 Deskripsi Fungsi `Class Prunerulecmar` 85
Tabel 4.8 Deskripsi Fungsi `Class Objcmar` 85
Tabel 4.9 Perhitungan Waktu CART 91
Tabel 4.10 Perhitungan Akurasi CART 92
Tabel 4.11 Perhitungan Waktu CMAR 94
Tabel 4.12 Perhitungan Akurasi CMAR 97

DAFTAR KODE PROGRAM

Kode Program 4.1	Variabel Form Input Data	74
Kode Program 4.2	Input Data Text Field	75
Kode Program 4.3	Input Data Combo Box	75
Kode Program 4.4	Fungsi `Class Datatraining`	76
Kode Program 4.5	Swing Controls Radio Button	77
Kode Program 4.6	Proses Pengiriman Data Prediksi	78
Kode Program 4.7	Perhitungan Split	81
Kode Program 4.8	Pemilihan Rule	83
Kode Program 4.9	Proses Prediksi	84
Kode Program 4.10	Hitung Frekuensi	85
Kode Program 4.11	Pilih Rule yang Sesuai Frequent Threshold	87
Kode Program 4.12	Penentuan Subset Tiap Rule	87
Kode Program 4.13	Menghitung Support dan Confidence	88
Kode Program 4.14	Pilih Rule yang Sesuai Conf Threshold	88
Kode Program 4.15	Pruning Rule	89
Kode Program 4.16	Rule Terakhir	90
Kode Program 4.17	Proses Prediksi	90

DAFTAR LAMPIRAN

Lampiran 1 DataPengujian CART dan CMAR 109

DAFTAR RUMUS

Rumus 2.1 Hitung Split 13
Rumus 2.2 Hitung Support 21
Rumus 2.3 Hitung Confidence 21
Rumus 2.4 Hitung Akurasi 22

BIODATA MAHASISWA

Nama	: Restu Fuji Rahayu
NIM	: 0610963050-96
Tempat dan Tanggal Lahir	: Blitar, 27 September 1988
Alamat Asal	: Dsn. Mandesan RT 01, RW 01, Ds. Mandesan, Kec. Selopuro, Kab. Blitar – 66184
Alamat Sementara	: Jl. Tangerang 06 Malang
Telp / HP	: 0342-691671 / 081 334 736 447
E-mail	: restu.fuji@gmail.com

BAB I
PENDAHULUAN

1.1 Latar Belakang

Banyaknya nasabah yang mengajukan permohonan kredit pada bank menuntut kreditor harus mampu mengevaluasi pemohon kredit dengan objektif, akurat, dan konsisten. Hal ini perlu dilakukan untuk menghindari kredit macet yang dapat menimbulkan kerugian pada bank. Pada kasus permohonan kredit oleh nasabah, seorang *decision maker* (pembuat keputusan) pada suatu perbankan harus mampu mengambil keputusan yang tepat untuk menerima atau menolak permohonan kredit tersebut.

Pada pengambilan keputusan selama ini, perbankan biasanya meminta pemohon (calon nasabah) mengisi formulir berupa daftar pertanyaan dan melengkapi permohonan kredit dengan berkas-berkas yang diperlukan oleh perbankan, untuk kemudian dilakukan penilaian permohonan kredit tersebut (Tole Sutikno, 2007). Berdasarkan penilaian parameter nasabah tersebut, maka bank dapat membuat peringkat (*rating*) calon debitur untuk memutuskan kelayakan debitur menerima kredit (Emanuel Kristijadi, 2003).

Salah satu metode dalam *data mining* adalah klasifikasi. Pada pengklasifikasian, ada beberapa macam metode yang digunakan, yaitu metode *decision tree* dan metode klasifikasi *associative*.

Salah satu algoritma pada *decision tree* yang sering digunakan adalah algoritma CART (*Classification and Regression Tree*). Pada algoritma CART, akan menghasilkan *binary tree* dengan memecah *record* pada tiap *node* berdasarkan fungsi variabel *input* tunggal. Penelitian dengan menggunakan algoritma CART sebelumnya pernah dilakukan pada judul penelitian *"Credit Risk Assessment Using Statistical and Machine Learning: Basic Methodology and Risk Modeling Applications"* (J. Galindo, 2000). Pada penelitian ini, diimplementasikan algoritma CART pada penilaian resiko perantara keuangan. Hasil penelitian menunjukkan bahwa algoritma CART menyediakan estimasi terbaik untuk standar dengan tingkat kesalahan yang relatif kecil dibandingkan algoritma Jaringan Syaraf Tiruan dan algoritma *K-Nearest Neighbour*.

Sedangkan pada metode klasifikasi *associative*, pada penelitian ini akan diimplementasikan algoritma CMAR (*Classification Based*

on Multiple Class-Association Rules). Algoritma CMAR mengadopsi dari metode *FP-growth* untuk generasi *frequent itemset* dan menggunakan pendekatan yang berbasis logika untuk masalah klasifikasi. Jadi, *node* pertama untuk algoritma CMAR, diisi berdasarkan atribut yang sering muncul.

Penelitian dengan menggunakan algoritma CMAR sebelumnya pernah dilakukan dengan judul penelitian *"Accurate and Efficient Classification Based on Multiple Class-Association Rules"* (Wenmin Li, 2001). Dalam penelitian ini, algoritma CMAR diimplementasikan pada banyak kasus yang diambil dari UCI *repository*. Pada penelitian tersebut, menunjukkan bahwa CMAR terbukti lebih efektif dan memiliki rata-rata keakuratan lebih baik dibandingkan dengan algoritma C4.5 dan CBA (*Classification Based on Associations*).

Karena algoritma CART dan CMAR sama-sama unggul pada dua pengujian yang berbeda, maka dalam penelitian ini, akan dibahas implementasi algoritma CART dan algoritma CMAR untuk menentukan status resiko kredit, yang selanjutnya akan dianalisis tingkat keakuratannya serta waktu komputasi kedua algoritma tersebut.

Berdasarkan latar belakang yang telah diuraikan, maka skripsi ini diberi judul "Implementasi Algoritma CART dan CMAR untuk Menentukan Status Resiko Kredit".

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, maka masalah yang akan dibahas dalam tugas akhir ini adalah:

1. Bagaimana cara kerja algoritma CART dan CMAR dalam penentuan status resiko kredit
2. Bagaimana waktu kinerja algoritma CART dan CMAR.
3. Bagaimana akurasi algoritma CART dan CMAR.

1.3 Batasan Masalah

Mengacu pada masalah yang telah dirumuskan, maka batasan masalah dalam skripsi ini adalah:

1. *Dataset* yang dipergunakan dalam analisa adalah data nasabah pengajuan kredit perbankan di Jerman, yang diambil dari situs http:// mlearn.ics.uci.edu/databases/statlog/german/

2. Menggunakan model klasifikasi yang berbentuk *decision tree* untuk memprediksi resiko kredit pada calon nasabah.

1.4 Tujuan

Tujuan yang ingin dicapai dari penelitian ini, yaitu :

1. Mengetahui cara kerja algoritma CART dan CMAR (dalam pembentukan *rule*) untuk penentuan status resiko kredit,
2. Mengetahui rata-rata waktu kinerja algoritma CART dan CMAR.
3. Mengetahui rata-rata tingkat akurasi algoritma CART dan CMAR.

1.5 Manfaat

Manfaat dari penulisan skripsi ini adalah untuk menentukan status resiko kredit pada calon nasabah, dan membantu kreditor dalam efisiensi waktu untuk mengevaluasi pemohon kredit dengan objektif, akurat, dan konsisten.

1.6 Metode Penyelesaian Masalah

Metode penyelesaian masalah yang dilakukan pada penelitian ini, yaitu :

1. Studi literatur.
   Mempelajari dan mengkaji beberapa literatur (jurnal, buku, dan artikel dari *website*) mengenai *data mining* dan algoritma CART dan CMAR.
2. Perancangan dan implementasi sistem.
   Mengimplementasikan algoritma CART dan CMAR dengan merancang dan membangun sebuah perangkat lunak untuk mengklasifikasikan data nasabah pengajuan kredit perbankan.
3. Uji coba dan analisis hasil implementasi.
   Menganalisa perbandingan perfomansi model *tree* yang dihasilkan oleh algoritma CART dan CMAR pada masalah penentuan resiko kredit.

1.7 Sistematika Penulisan

Sistematika penulisan tugas akhir ini dibagi menjadi lima bab dengan masing-masing bab diuraikan sebagai berikut:

1. BAB I PENDAHULUAN

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

2. BAB II DASAR TEORI

   Bab ini berisi teori-teori dari berbagai pustaka yang menunjang penelitian dalam penulisian skripsi. Adapun teori yang tercakup dalam bab ini yaitu mengenai definisi dan konsep kredit, data, *data mining*, *decision tree*, *association classification*, algoritma CART dan CMAR.

3. BAB III METODOLOGI DAN PERANCANGAN SISTEM

   Bab ini berisi mengenai perancangan perangkat lunak yang dibangun, meliputi perancangan proses, perancangan tabel dan perancangan uji coba.

4. BAB IV IMPLEMENTASI DAN PEMBAHASAN

   Bab ini berisi hasil dari implementasi perangkat lunak yang digunakan untuk mengukur akurasi hasil klasifikasi, pembahasan analisa hasil uji coba dan evaluasi hasil uji coba.

5. BAB V KESIMPULAN DAN SARAN

   Bab ini memuat kesimpulan dari hasil penelitian dan saran-saran untuk pengembangan penelitian selanjutnya.

BAB II
TINJAUAN PUSTAKA

Pembahasan pada bab ini meliputi dasar teori yang menunjang penelitian, yaitu mengenai kredit, data, *data mining*, *decision tree*, *association classification,* algoritma CART dan CMAR.

2.1 Kredit

Menurut Undang-Undang Nomor 7 tahun 1992, tentang Perbankan, kredit adalah penyediaan uang atau tagihan yang dapat dipersamakan dengan itu, berdasarkan persetujuan atau kesepakatan pinjam meminjam antara bank dengan pihak lain, yang mewajibkan pihak lain untuk melunasi utangnya setelah jangka waktu tertentu dengan pemberian bunga.

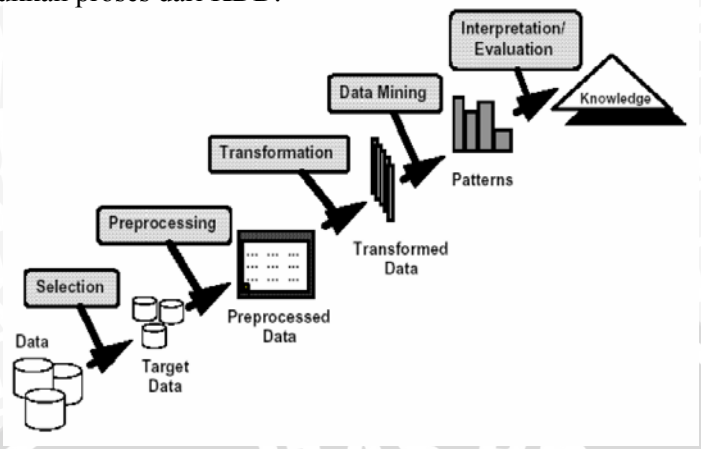
Pemberian kredit merupakan kegiatan usaha yang mengandung resiko tinggi dan berpengaruh terhadap kesehatan dan keberlangsungan usaha perbankan. Pada kasus permohonan kredit oleh nasabah, seorang *decision maker* pada suatu perbankan harus mampu mengambil keputusan yang tepat untuk menerima atau menolak permohonan kredit tersebut.

Pada pengambilan keputusan selama ini, perbankan biasanya meminta pemohon (calon nasabah) mengisi formulir berupa daftar pertanyaan dan melengkapi permohonan kredit dengan berkasberkas yang diperlukan oleh perbankan, untuk kemudian dilakukan penilaian permohonan kredit tersebut. Jika *decision maker* tepat dalam mengambil keputusan, maka pihak bank mendapatkan nasabah yang menyokong kesehatan dan keberlangsungan usaha perbankan, dan sebaliknya jika salah membuat keputusan maka akan menjatuhkan kelangsungan usaha perbankan.

2.2 Data

Data adalah fakta yang terungkap atau representasi fakta dunia nyata yang mewakili suatu objek. KDD *(Knowledge Discovery In Database)* berhubungan dengan teknik integrasi dan penemuan ilmiah, interprestasi anvisualisasi dari pola-pola sejumlah kumpulan data.

Knowledge discovery in databases (KDD) adalah keseluruhan proses *non-trivial* untuk mencari dan mengidentifikasi pola (*pattern*) dalam data, dimana pola yang ditemukan bersifat sah, baru, dapat bermanfaat dan dapat dimengerti (Usama Fayyad, 1996). Gambar 2.1 menunjukkan proses dari KDD.



Gambar 2.1 Proses *Knowledge discovery in databases* (KDD)
(Usama Fayyad, 1996)

Berikut ini adalah proses dari KDD (Kantardzic, 2003):

1. *Data Selection*
   Menciptakan himpunan data target, pemilihan himpunan data, atau memfokuskan pada subset variabel atau sampel data, dimana penemuan (*discovery*) dilakukan. pemilihan (seleksi) data dari sekumpulan data operasional perlu dilakukan sebelum tahap penggalian informasi dalam KDD dimulai. Data hasil seleksi yang digunakan untuk proses *data mining*, disimpan dalam suatu berkas, terpisah dari basis data operasional.
2. *Pre-processing/Cleaning*
   Pemrosesan pendahuluan dan pembersihan data merupakan operasi dasar seperti penghapusan *noise* dilakukan. Sebelum proses *data mining* dapat dilaksanakan, perlu dilakukan proses *cleaning* pada data yang menjadi fokus KDD. Proses *cleaning* mencakup antara lain membuang duplikasi data, memeriksa data yang inkonsisten, dan memperbaiki

kesalahan pada data, seperti kesalahan cetak (tipografi). Dilakukan proses *enrichment*, yaitu proses "memperkaya" data yang sudah ada dengan data atau informasi lain yang relevan dan diperlukan untuk KDD, seperti data atau informasi eksternal.

3. *Transformation*
   Pencarian fitur-fitur yang berguna untuk mempresentasikan data bergantung kepada *goal* yang ingin dicapai. Merupakan proses transformasi pada data yang telah dipilih, sehingga data tersebut sesuai untuk proses *data mining*. Proses ini merupakan proses kreatif dan sangat tergantung pada jenis atau pola informasi yang akan dicari dalam basis data.
4. *Data mining*
   Pemilihan tugas *data mining*, pemilihan *goal* dari proses KDD misalnya: klasifikasi, regresi, klustering dan lain-lain. Proses *data mining* yaitu proses mencari pola atau informasi menarik dalam data terpilih dengan menggunakan teknik atau metode tertentu. Teknik, metode, atau algoritma dalam *data mining* sangat bervariasi. Pemilihan metode atau algoritma yang tepat sangat bergantung pada tujuan dan proses KDD secara keseluruhan.
5. *Interpretation / Evaluation*
   Penerjemahan pola-pola yang dihasilkan dari *data mining*. Pola informasi yang dihasilkan dari proses *data mining* perlu ditampilkan dalam bentuk yang mudah dimengerti oleh pihak yang berkepentingan. Tahap ini merupakan bagian dari proses KDD yang mencakup pemeriksaan apakah pola atau informasi yang ditemukan bertentangan dengan fakta atau hipotesa yang ada sebelumnya.

2.3 *Data Mining*

Pengertian *data mining* dari beberapa referensi adalah sebagai berikut:

1. Secara sederhana dapat didefinisikan bahwa *data mining* adalah ekstraksi informasi atau pola yang penting atau menarik dari data yang ada di *database* yang besar sehingga menjadi informasi yang sangat berharga (Yudho Giri Sucahyo, 2003).

2. *Data mining* merupakan proses penemuan yang efisien sebuah pola terbaik yang dapat menghasilkan sesuatu yang bernilai dari suatu koleksi data yang sangat besar (Kurt Thearling, 2000).
3. *Data mining* adalah suatu pola yang menguntungkan dalam melakukan *search* pada sebuah *database* yang terdapat pada sebuah model. Proses ini dilakukan berulang-ulang (*iterasi*) hingga didapat satu set pola yang memuaskan yang dapat berfungsi sesuai yang diharapkan (Christina Chung, 1998).
4. *Data mining* adalah sebuah *class* dari suatu aplikasi *database* yang mencari pola-pola yang tersembunyi di dalam sebuah kumpulan data yang dapat digunakan untuk memprediksi perilaku yang akan datang (Khusnawi, 2007).

Berdasarkan beberapa pengertian tersebut, dapat ditarik kesimpulan bahwa *data mining* adalah suatu algoritma di dalam menggali informasi berharga yang terpendam atau tersembunyi pada suatu koleksi data (*database)* yang sangat besar sehingga ditemukan suatu pola yang menarik yang sebelumnya tidak diketahui.

Menurut Zaiane (1999), metode *data mining* dibagi ke dalam dua macam, yaitu:

1. *Supervised Learning* (proses belajar terawasi)
   Pada pembelajaran ini target tersedia. Dengan kata lain, label atau nama dan nomor kelas pada *data training* (supervisi) diketahui, kemudian data baru diklasifikasikan berdasarkan *data training* tersebut. Contoh : Klasifikasi.
2. *Unsupervised Learning* (proses belajar tak terawasi)
   Pada proses pembelajaran ini, tidak membutuhkan target untuk keluarannya. Dengan kata lain, tidak terdapat label atau nama kelas pada *data training*, atau bahkan tidak diketahui nomor kelasnya. *Data training* dikelompokkan berdasarkan ukuran kesamaan (*similarity*). Oleh karena itu tidak ada perbandingan yang dilakukan dengan respon ideal yang ditetapkan sebelumnya. Rangkaian pelatihan hanya berisi vektor masukan saja. Contoh : *Clustering*.

Kebanyakan tugas *data mining* merupakan *supervised data mining*.

2.3.1 Teknik *data mining*

Berdasarkan Daniel Larose (2005) pada modul Laboratorium Sistem Informasi & Keputusan ITB (2009), disebutkan beberapa jenis pekerjaan yang dapat dilakukan dengan teknik *data mining* yaitu:

1 *Description*
Deksripsi pola dan *trend* seringkali memberikan penjelasan yang masuk akal untuk pola dan *trend*. Model *data mining* harus dibuat sejelas (transparan) mungkin, yang berarti hasil dari model *data mining* harus mendeskripsikan pola jelas yang sesuai dengan interpretasi dan penjelasan intuitif. Metode *data mining* tertentu lebih sesuai dari metode lain dalam hal interpretasi transparan. Deskripsi yang berkualitas tinggi seringkali diperoleh melalui *exploratory data analysis,* metode grafis dalam eksplorasi data dalam pencarian pola dan *trend*.

2 *Estimation*
Estimasi hampir sama dengan klasifikasi kecuali bahwa variabel targetnya berupa numerik bukan kategori.

3. *Prediction*
Prediksi hampir sama dengan klasifikasi dan estimasi. Perbedaan mendasar yaitu, hasil dari prediksi adalah di masa depan. Contoh dari prediksi adalah memprediksi harga saham selama 3 bulan mendatang. Semua metode dan teknik yang digunakan untuk klasifikasi dan estimasi dapat pula digunakan untuk prediksi dalam situasi yang sesuai.

4. *Classification*
Dalam klasifikasi terdapat sebuah target variabel kategori, misalnya *income bracket*, dimana misalnya dapat dipartisi menjadi 3 kelas atau kategori: *high income, middle income,* dan *low income.* Model *data mining* meneliti *set record* dalam jumlah besar, dimana tiap *record* berisi informasi mengenai variabel target serta satu *set input.*

5. *Clustering*
   Clustering merupakan pengelompokkan *record*, observasi, atau kasus ke dalam kelas-kelas dengan objek yang serupa. Sebuah *cluster* adalah koleksi *record* yang sama satu sama lain, dan tidak sama dengan *record* di *cluster* lain. *Clustering* berbeda dengan *classification* karena tidak ada variabel target dalam *clustering*. *Clustering* tidak mengklasifikasi, estimasi ataupun prediksi nilai dari variabel target. Akan tetapi algoritma *clustering* mencari segmen dari keseluruhan *set* data ke dalam subgrup yang relatif homogen atau *cluster* di mana keserupaan (*similarity*) *record* dalam *cluster* adalah maksimal dan keserupaan *record* di luar *cluster* adalah minimal.
6. *Assosiation*
   Asosiasi merupakan sebuah teknik *data mining* yaitu melakukan pencarian atribut mana yang digabungkan bersama.

2.3.2 Klasifikasi

Menurut Khusnawi (2007), klasifikasi adalah suatu fungsionalitas *data mining* yang akan menghasilkan model untuk memprediksi kelas atau kategori dari objek-objek di dalam basis data. Klasifikasi merupakan proses yang terdiri dari dua tahap, yaitu tahap pembelajaran dan tahap pengklasifikasian.

Pada tahap pembelajaran, sebuah algoritma klasifikasi membangun sebuah model klasifikasi dengan cara menganalisis *data training*. Tahap pembelajaran dapat juga dipandang sebagai tahap pembentukan fungsi atau pemetaan $y = f(X)$ di mana y adalah kelas hasil prediksi dan X adalah *tuple* yang ingin diprediksi kelasnya. Pada skripsi ini, fungsi atau pemetaan tersebut digambarkan dalam bentuk pohon keputusan.

Selanjutnya, pada tahap pengklasifikasian, model yang telah dihasilkan digunakan untuk melakukan klasifikasi terhadap *unknown data*. Akan tetapi, sebuah model hanya boleh digunakan untuk klasifikasi jika akurasi model tersebut cukup tinggi. Akurasi dapat diketahui dengan cara menguji model tersebut dengan *data test*. *Data test* terdiri dari *tuple-tuple* yang

kelasnya sudah diketahui, namun *data test* tidak boleh sama dengan *data training* karena menyebabkan pengujian tersebut menunjukkan akurasi yang tinggi, padahal belum tentu demikian.

2.3.3 *Decision tree*

Decision tree merupakan salah satu teknik yang digunakan dalam klasifikasi, yang berbentuk struktur *flowchart* yang menyerupai *tree* (pohon) (Han, 2001). *Decision tree* dibangun oleh dua macam simpul, yaitu simpul (*node*) internal dan simpul daun (*leaf*), serta cabang. Setiap simpul internal (simpul yang memiliki cabang) merepresentasikan *attribute test*, sedangkan cabang dari sebuah simpul internal berkorespondensi dengan semua hasil yang mungkin dari tes pada simpul. Sedangkan, simpul daun (*leaf*) merepresentasikan nilai dari kelas (Kantardzic, 2003). Pada penelitian ini digunakan algoritma *decision tree* CART.

Decision tree menangani dua macam tipe atribut, yaitu:

1. Numeris atau kontinyu : Domain memiliki nilai tak hingga, direpresentasikan dalam bentuk *real* number (contoh : umur, gaji).
2. Nominal atau kategoris : domain memiliki nilai terhingga (dalam bentuk *finite set*) (contoh : pekerjaan, status pernikahan, ras).

Pada penelitian yang dilakukan oleh Yogi Yusuf (2007), disebutkan bahwa persayaratan yang harus dipenuhi dalam penerapan algoritma *decision tree* meliputi:

- Algoritma *decision tree* merepresentasikan *supervised learning,* dan oleh karena itu membutuhkan variabel target *preclassified*.
- *Training data set* harus kaya dan bervariasi.
- Kelas atribut target harus diskrit.

2.3.4 *Associative classification*

Pada penelitian ini, digunakan metode klasifikasi *associative* CMAR. Menurut Thabtah (2007), klasifikasi *associative* merupakan pendekatan yang menjanjikan dalam *data mining*

yang memanfaatkan *rule* asosiasi penemuan terbaik untuk membangun sistem klasifikasi.

Klasifikasi *associative* mencapai akurasi klasifikasi yang tinggi, *rule* pada pengklasifikasian ini diinterpretasi dan diberikan probabilitas *confidence* ketika mengelompokkan objek yang dapat digunakan untuk memecahkan klasifikasi masalah ketidakpastian (Zhonghua Tang, 2007).

2.4 *Pruning tree* (pemangkasan pohon)

Pruning dilakukan untuk menghindari *overfitting*. *Overfitting* sering terjadi karena terlalu banyak cabang yang telah dibuat, guna untuk memperoleh akurasi tinggi pada tahap *training*. (Budi Santoso, 2007).

Pruning terdiri dari beberapa langkah pemilihan secara berulang simpul yang dijadikan simpul daun (dilambangkan dengan kotak). Dengan mengubah simpul menjadi simpul daun artinya tidak dilakukan pemecahan lagi sesudah itu. Dengan demikian, ukuran *tree* berkurang. Proses *pruning* menghasilkan tawar-menawar antara kesalahan klasifikasi (*misclassification error)* dalam data validasi dan jumlah simpul keputusan dalam *tree* yang dipangkas untuk mencapai *tree* yang bisa menangkap pola sesungguhnya dan bukan hanya *noise* dalam *data training*.

2.5 Algoritma CART

Algoritma CART ditemukan oleh Breiman pada tahun 1984. Adapun karakteristik dari algoritma ini, yaitu:

- Bersifat biner
- Hanya memiliki 2 cabang untuk setiap *decision node*, sehingga setiap kemungkinan nilai untuk *decision node* harus dipartisi menjadi 2 bagian, misalnya untuk *predictor variable saving* yang memiliki nilai *low*, *med fan high* dibagi menjadi *saving = low* dan *saving = med / high* atau kombinasi lainnya. Setiap kombinasi tersebut membentuk alternatif *candidate split* yang dipilih untuk penyusunan inisial partisi pada *root node* dan *decision node* lainnya. Adapun kriteria pemilihan tersebut berdasarkan nilai *goodness of split* (Φ(s/t)) yang terbesar.

Adapun langkah-langkah penyusunan CART, yaitu (Laboratorium Sistem Informasi & Keputusan ITB, 2009):

1. Diasumsikan telah memiliki *data training*.
   Sebagai contoh, diberikan tabel *data training* seperti pada Tabel 2.1.
   Tabel 2.1 Contoh *data training*

customer	savings	assets	income	credit risk
1	medium	high	75	good
2	Low	low	50	bad
3	high	medium	25	bad
4	medium	medium	50	good
5	low	medium	100	good
6	high	high	25	good
7	low	low	25	bad
8	medium	medium	75	good

2. Dari *data training* tersebut, disusunlah alternatif untuk *candidate split*, sehingga setiap nilai untuk *predictor variable* hanya membentuk 2 cabang seperti yang terdapat pada Tabel 2.2.
   Tabel 2.2 Alternatif *candidate split*

candidate split	left child node (tL)	right child node, tR
1	saving = low	savings = (medium, high)
2	saving = medium	savings = (low, high)
3	saving = high	savings = (low, medium)
4	assets = low	assets = (medium, high)
5	assets = medium	assets = (low, high)
6	assets = high	assets = (low, medium)
7	income <= $25000	income > $25000
8	income <= $50000	income > $50000
9	income <= $75000	income > $75000

3. Kemudian untuk setiap *candidate split*, dihitung variabel-variabel berikut berdasarkan *data training* yang dimiliki. Adapun variabel-variabel tersebut, yaitu :

$$\varphi\left(\frac{s}{t}\right) = 2P_L P_R \sum_{j=1}^{\#\,classes} \left|P\left(\frac{j}{t_L}\right) - P\left(\frac{j}{t_L}\right)\right| \quad (2.1)$$

   dimana,

$$t_R = left\_child\_node\_of\_node\_t$$

$$t_R = right\_child\_node\_of\_node\_t$$

$$P_L = \frac{number\_of\_records\_at\_t_L}{number\_of\_records\_in\_training\_set}$$

$$P_R = \frac{number\_of\_records\_at\_t_R}{number\_of\_records\_in\_training\_set}$$

$$P\left(\frac{j}{t_L}\right) = \frac{number\_of\_class\_j\_records\_at\_t_L}{number\_of\_records\_at\_t}$$

$$P\left(\frac{j}{t_R}\right) = \frac{number\_of\_class\_j\_records\_at\_t_R}{number\_of\_records\_at\_t}$$

Contoh hasil perhitungannya ditunjukkan pada Tabel 2.3 berikut:

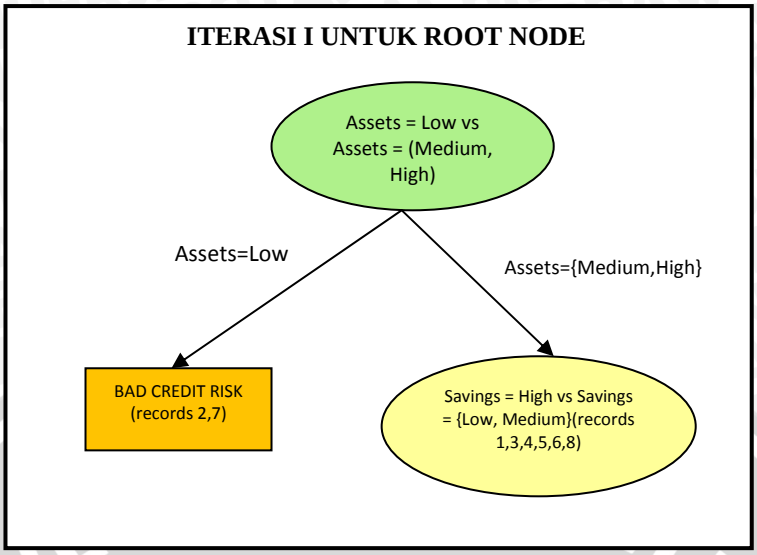
Tabel 2.3 Hasil perhitungan *split*

split	PL	PR	P(j/tL)		P(j/tR)		2PLPR	Q(s/t)	Φ(s/t)
			G	*B*	*G*	*B*			
1	0.375	0.625	0.333	0.667	0.8	0.2	0.46875	0.934	0.4378
2	0.375	0.625	1	0	0.4	0.6	0.46875	1.2	0.5625
3	0.25	0.75	0.5	0.5	0.667	0.333	0.375	0.334	1.1252
4	0.25	0.75	0	1	0.883	0.167	0.375	1.666	0.6247
5	0.5	0.5	0.75	0.25	0.5	0.5	0.5	0.5	0.25
6	0.25	0.75	1	0	0.5	0.5	0.375	1	0.375
7	0.375	0.625	0.333	0.667	0.8	0.2	0.46875	0.934	0.4378
8	0.625	0.375	0.4	0.6	1	0	0.46875	1.2	0.5625
9	0.875	0.125	0.571	0.429	2	0	0.21875	0.858	0.1876

Dapat dilihat dari Tabel 2.3, bahwa yang memiliki nilai *goodness of split* (Φ(s/t)) yang terbesar, yaitu *split* 4 dengan nilai 0.64275. Oleh karena itu *split* 4 yang digunakan pada *root node*, yaitu *split* dengan : *assets = low* dengan *assets = {medium, high}*.

Untuk penentuan pencabangan, dapat dilihat bahwa dengan *assets=low* maka didapatkan *pure node leaf*, yaitu *bad risk* (untuk *record* 2 dan *7*). Sedangkan untuk *assets = {medium, high}* masih terdapat 2 nilai, yaitu *good credit risk* dan *bad credit risk*. Sehingga pencabangan untuk *assets = {medium, high}*

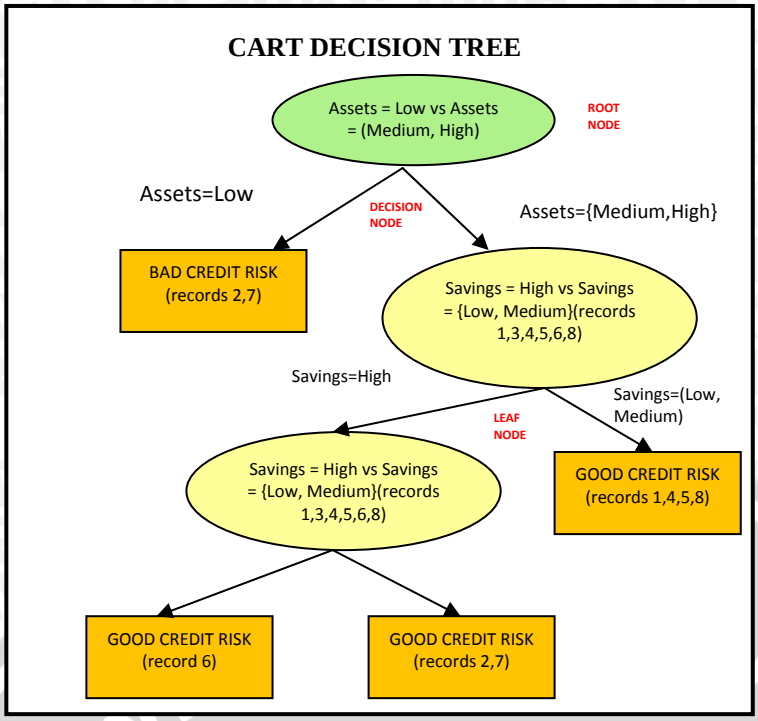
memiliki *decision node* baru. Adapun pemilihan *split* yang digunakan, yaitu dengan menyusun perhitungan nilai Φ(s/t) yang baru tanpa melihat *split* 4, *record* 2 dan 7. Pada Gambar 2.2 ditunjukkan iterasi 1 untuk *root node*.



Gambar 2.2 Iterasi 1 untuk *root node*

Demikian seterusnya hingga akhirnya dibentuk *leaf node* dan membentuk *decision tree* yang utuh (*fully grown form*) seperti ditunjukkan pada Gambar 2.3.

Pada Gambar 2.6 tersebut, terdapat *root node, decision node* dan *leaf node*. *Root node* merupakan cabang utama dari suatu pohon, yang ditandai dengan bentuk oval berwarna hijau. *Decision node* ditandai dengan bentuk oval berwarna kuning, dan *leaf node* merupakan cabang terakhir, ditandai dengan bentuk kotak berwarna *orange*.



Gambar 2.3 CART *decision tree*

2.6 Algoritma CMAR

CMAR (*Classification based on Multiple Association Rules*) adalah metode klasifikasi yang diadopsi dari metode *FP-growth* untuk *frequent itemset*. Algoritma CMAR terdiri dari dua tahap, yaitu *rule generation* atau *training* dan klasifikasi (John Wiley, 2003).

Pada fase pertama, *rule generation,* CMAR menghitung aturan dalam bentuk $R : P \rightarrow c$, P merupakan sebuah pola dalam data *training*, c merupakan label kelas, seperti *sup (R)* dan *con f(R)* melewati *support* dan *confidence thresholds* masing-masing. Selanjutnya, CMAR memangkas beberapa aturan dan hanya memilih *subset* aturan klasifikasi berkualitas tinggi (Wenmin Li, 2001).

Untuk fase kedua, klasifikasi, diberikan objek data obj, CMAR mengekstrak *subset* dari *rule* yang cocok dengan objek dan memprediksi label kelas dari objek dengan menganalisis *subset* dari *rule*.

Untuk menemukan aturan klasifikasi, langkah yang pertama yaitu menyiapkan *data training*, selanjutnya menemukan aturan lengkap melalui *support* dan *confidence*. Untuk membuat *mining* yang *scalable* dan efisien, CMAR mengadopsi metode *FP-growth.*

FP-growth dilakukan dengan cara membangkitkan struktur data *tree* atau disebut dengan *FP-Tree* dengan penggalian *itemset* yang *frequent* (J. Han, 2000).

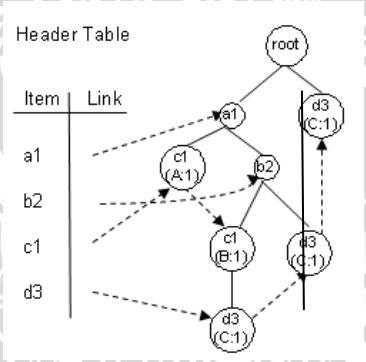
Sebagai contoh untuk aturan klasifikasi pada algoritma CMAR, diberikan contoh data *training* pada Tabel 2.4, dengan ketentuan *support threshold =2* dan *confidence threshold = 50%.*

Tabel 2.4 Contoh data *training_2*

Row-id	A	B	C	D	Class Label
1	a_1	b_1	c_1	d_1	*A*
2	a_1	b_1	c_1	d_1	*B*
3	a_2	b_2	a_2	d_2	*A*
4	a_1	b_1	c_1	d_1	*C*
5	a_1	b_1	c_1	d_1	*C*

Langkah pertama, *scan* data *training* kemudian tetapkan sebagai *T*, dan temukan atribut yang minimal muncul dua kali. Selanjutnya, tetapkan sebagai *frequent itemset* $F = \{a_1, b_2, c_1, d_3\}$. Nilai atribut lainnya yang gagal pada *support threshold*, tidak dapat berperan dalam *class-association rules*, dan selanjutnya dapat dipangkas.

Kemudian, urutkan nilai atribut pada *F* pada *support*-nya secara *descending*, *F-list* = a_1 - b_2 - c_1 - d_3 dan *scan* kembali data *training* untuk membangun *FP-tree* seperti yang ditunjukkan pada Gambar 2.4.



Gambar 2.4 Contoh *FP-tree* (J. Han, 2000)

Untuk setiap *tuple* dalam *data training set* pada *FP-growth*, nilai atribut yang muncul dalam *F-list* diambil dan disortir dari *F-list* itu sendiri. Sebagai contoh, untuk *tuple* pertama (a_1, c_1), yang diambil dan dimasukkan dalam *tree* sebagai cabang paling kiri pada *tree*. Label kelas melekat pada *node* terakhir pada *path*.

Tuple pada data *training set* membagi *prefix*. Untuk contoh, *tuple* kedua membawa nilai atribut (a_1, b_2, c_1) pada *F-list* dan membagi sama dengan *prefix* a_1 dengan *tuple* pertama, juga membagi *sub-path* a_1 dengan cabang paling kiri.

Semua *node* dengan nilai atribut yang sama dihubungkan bersama sebagai sebuah antrian dimulai dari tabel *header*. Ketiga, berdasarkan *F-list*, himpunan dari kelas aturan asosiasi dapat dibagi menjadi 4 *subset* tanpa tumpang tindih.

a. Yang memiliki d_3
b. Yang memiliki c_1 tetapi tidak memilki d_3
c. Yang memilki b_2 tetapi tidak memilki d_3 ataupun c_1
d. Hanya memilki a_1

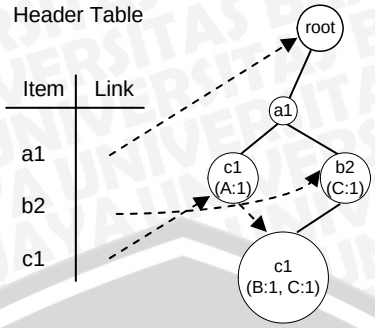
CMAR mencari himpunan bagian tersebut satu persatu.

Keempat, untuk menemukan subset atau *rules* yang mempunyai d_3, CMAR melintasi *nodes* yang memilki nilai atribut d_3 dan melihat ke atas untuk mengumpulkan d_3 untuk diproyeksikan ke *database*. Yang berisi *tuple* $(a_1, b_2, c_1, d_3) : C$, $(a_1, b_2, d_3) : C$ dan $d_3 : A$. Ketiganya mengandung *tuple* yang memiliki *frequent pattern* d_3.

Masalah untuk mencari semua pola yang sering di *set* pelatihan dapat dikurangi untuk pertambangan pola yang sering terjadi dalam d_3-*project database*. Sebagai contoh, dalam d_3-*project database*, pola (a_1, b_2, d_3) terjadi dua kali *support* adalah sama dengan nilai ambang batas / *threshold* yang diperlukan 2. Selain itu, aturan tersebut didasarkan pada *frequent pattern*, $(a_1, b_2, d_3) \rightarrow C$ memiliki keyakinan 100% (diatas nilai *threshold*) dan itu adalah satu-satunya aturan yang dihasilkan dalam proyeksi tertentu *database*.

Setelah mencari aturan yang memiliki nilai d_3, semua *node* dari d_3 dan yang berhubungan dengan label kelas digabung ke simpul induknya *FP-tree*. *FP-tree* menyusut seperti ditunjukkan pada Gambar 2.5. Sisanya seperangkat aturan yang bisa ditambang sama mengulangi prosedur sebelumnya untuk c_1- *project database*, kemudian untuk b_2- *project database*, dan akhirnya untuk a_1-*project database*. Dalam analisis, (a_1, c_1) adalah pola yang sering dengan *support* 3, tapi semua aturan dengan *confidence* kurang dari nilai

threshold. Kesimpulan yang sama dapat ditarik untuk pola (a_1, b_2), dan untuk (a_1). Oleh karena itu, aturan asosiasi hanya dihasilkan melalui proses pelatihan dengan T database $(a_1, b_2, d_3) \rightarrow C$ dengan *support* sama dengan 2 dan 100% *confidence*.



Gambar 2.5 *FP-tree* setelah penggabungan *node* d_3 (J. Han, 2000)

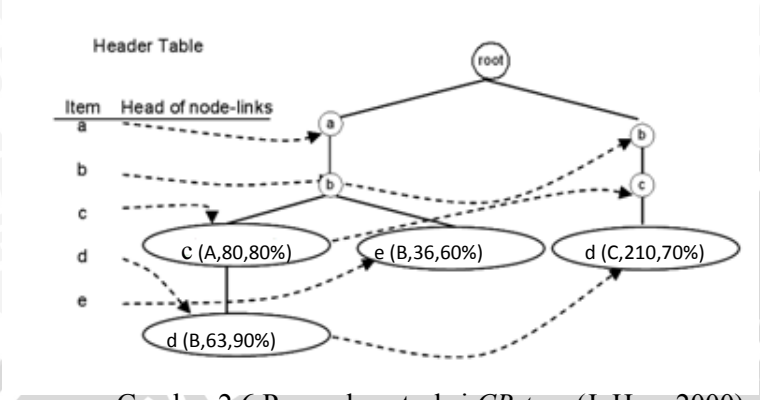
Ketika seperangkat aturan dipilih untuk klasifikasi, CMAR siap untuk mengklasifikasikan sampel baru. Untuk sampel baru, CMAR mengumpulkan *subset* dari aturan pencocokan sampel dari total set aturan. Trivial, jika semua aturan yang memiliki kelas yang sama, CMAR hanya memberikan label yang terhadap sampel baru. Jika aturan tidak konsisten dalam label kelas, membagi CMAR aturan menjadi kelompok-kelompok menurut label kelas dan hasil label kelompok "kuat". Untuk membandingkan kekuatan kelompok, perlu untuk mengukur efek "gabungan" dari masing-masing kelompok. Intuitif, jika aturan dalam suatu kelompok sangat berkorelasi positif dan memiliki dukungan yang baik, kelompok harus memiliki pengaruh yang kuat. CMAR menggunakan aturan terkuat dalam grup sebagai wakil, yaitu, aturan dengan nilai tes tertinggi x^2 (diadopsi untuk algoritma untuk perhitungan sederhana).

Setelah *rule* dibuat, *rule* disimpan di *CR-tree,* yang merupakan struktur pohon awalan. Tabel 2.5 menunjukkan 4 aturan yang telah ditemukan pada *training set*.

Tabel 2.5 Aturan yang telah ditemukan pada *training set*

Rule-id	Rule	Support	Confidence
1	*abc* → *A*	80	80%
2	*abcd* → *A*	63	90%
3	*abc* → *B*	36	60%
4	*bcd* → *C*	210	70%

Sebuah *CR-tree* dibangun untuk seperangkat aturan, dan proses konstruksi ditunjukkan pada Gambar 2.6 berikut:



Gambar 2.6 Proses konstruksi *CR-tree* (J. Han, 2000)

Sebuah *CR-tree* memiliki akar. Semua nilai atribut yang muncul di sisi kiri aturan diurutkan menurut frekuensinya. Frekuensi yang paling sering muncul hilang terlebih dahulu.

Pada *rule* pertama, *abc* → *A* dimasukkan ke dalam pohon sebagai *path* dari *root node.* Label kelas ataupun *support* dan *confidence* dari aturan dinotasikan sebagai (*A, 80, 80%)*, yang tersusun pada *node* terakhir di *path* adalah *node c.*

Rule kedua, *abcd* → *A,* membagi *prefix abc* pada *rule* pertama. Oleh karena itu, *abcd* dimasukkan ke dalam pohon dengan memperluas *node* baru *d* ke *path* yang dibentuk oleh aturan pertama. Jadi, label kelas, *support* dan *confidence* dari *rule* yang tersusun terakhir adalah *node d.*

Untuk *rule* ketiga dan keempat, dapat dimasukkan dengan cara yang sama. Semua *node* dengan nilai atribut yang sama dihubungkan bersama oleh *node-link* ke antrian. Masing-masing *head* antrian disimpan ke dalam tabel *Header.*

Untuk menyimpan *set* aturan asli, 13 *cells* dibutuhkan untuk *hand side* sebelah kiri aturan. Penggunaan *CR-tree* hanya 9 *node* yang dibutuhkan.

CMAR menghitung *set* lengkap aturan dalam bentuk R: P c →, seperti yang *sup* (R) dan *conf* (R) melewati ambang batas yang diberikan. Untuk ambang batas dukungan yang diberikan dan keyakinan, metode asosiatif-klasifikasi menemukan set lengkap peraturan-peraturan kelas-asosiasi (CAR) melewati batas. Dalam tahap pengujian, ketika sampel (*unclassified*) baru datang, pemilah, diwakili oleh seperangkat aturan asosiasi, memilih aturan yang sesuai dengan sampel dan memiliki keyakinan tertinggi, dan menggunakannya untuk memprediksi klasifikasi sampel baru.

Untuk menghitung support dan confidence, ditunjukkan pada rumus 2.2 dan 2.3 (Sanjay Chawla, 2007).

$$Support, s\ (X \rightarrow Y) = \frac{\sigma(X \cup Y)}{N} \quad (2.2)$$

$$Confidence, c\ (X \rightarrow Y) = \frac{\sigma(X \cup Y)}{X} \quad (2.3)$$

Pemangkasan pohon pada algoritma CMAR, dapat dipenuhi dari ketiga syarat berikut, diasumsikan jika terdapat dua *rule, R_1 dan R_2*:

1. Jika *confidence* dari R_1 > R_2, maka *rule* yang harus dipangkas adalah R_2.
2. Jika *confidence* dari R_1 dan R_2 sama, maka dilihat dari nilai support-nya, *rule* R_2 dipangkas jika support $R_1 > R_2$.
3. Jika *confidence* dari R_1 dan R_2 sama, dan *support* dari R_1 dan R_2 sama, maka dilihat dari *rule* yang memiliki nilai atribut lebih sedikit, dengan syarat, *rule* $R_1 : P \rightarrow c$ dan $R_2 : P' \rightarrow c'$, *jika dan hanya jika* P adalah *subset* dari P'.

2.7 Akurasi

Untuk menganalisa hasil dari klasifikasi algoritma CART dan CMAR, maka perlu diketahui tingkat akurasi dan *error rate* dari kedua algoritma tersebut.

Secara garis besar, untuk mengevaluasi model yang dibangun algoritma klasifikasi tersebut dapat dilakukan dengan menghitung jumlah dari *test record* yang diprediksi secara benar (akurasi) oleh

model tersebut. Akurasi dinyatakan pada persamaan beikut (Tan, 2004).

$$Akurasi = \frac{jumlah\ prediksi\ benar}{jumlah\ total\ prediksi} \quad (2.4)$$

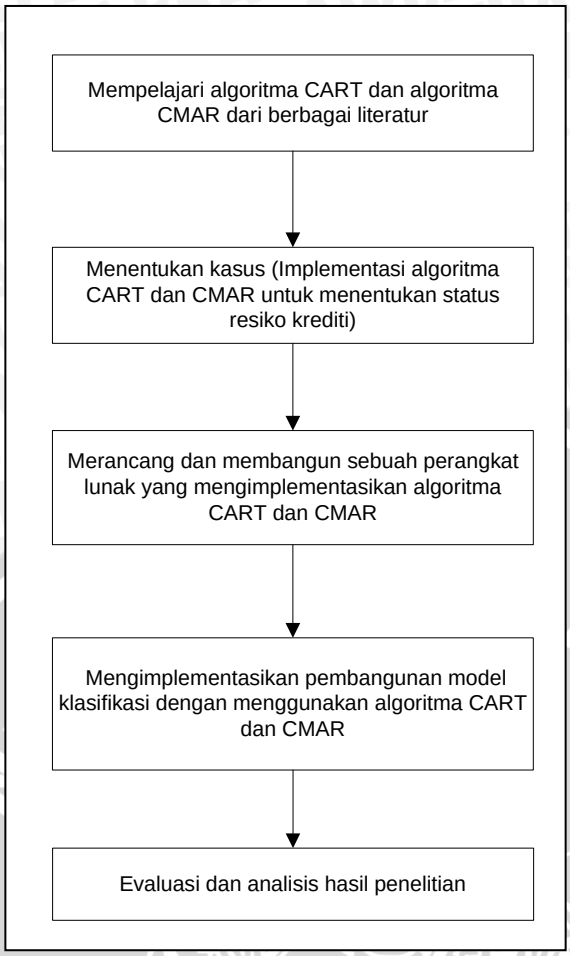
Jumlah prediksi benar, yaitu jumlah *record data testing* yang memiliki hasil prediksi kelas menggunakan model klasifikasi yang dibangun dari *data training*, sama dengan nilai kelas *data testing* semula. Sedangkan jumlah total prediksi, yaitu jumlah keseluruhan *record* yang diklasifikasikan menggunakan model klasifikasi yang dibangun dari data *training* (jumlah prediksi benar + jumlah prediksi salah).

BAB III
METODOLOGI DAN PERANCANGAN SISTEM

Pada bab metodologi dan perancangan, akan dibahas langkah-langkah yang dilakukan dalam pengimplementasian algoritma CART dan CMAR dalam menetukan status resiko kredit. Adapun langkah-langkah yang dilakukan dalam penelitian, yaitu:

1. Mempelajari algoritma CART dan algoritma CMAR dari berbagai literatur yang nantinya akan digunakan sebagai acuan untuk mendapatkan solusi dari studi kasus yang digunakan dalam penelitian ini.
2. Menentukan kasus dalam implementasi algoritma CART dan CMAR, yaitu untuk menentukan status resiko kredit yang diambil dari data nasabah pengajuan kredit perbankan di Jerman, yang diambil dari situs http://mlearn.ics.uci.edu/databases/statlog/german/.
3. Merancang dan membangun sebuah perangkat lunak yang mengimplementasikan proses pembangunan model klasifikasi data dengan menggunakan algoritma CART dan CMAR untuk memprediksi resiko kredit pada calon nasabah.
4. Mengimplementasikan pembangunan model klasifikasi dari *data training* dengan algoritma CART dan algoritma CMAR, serta membandingkan akurasi dan kinerja dari kedua algoritma tersebut.
5. Menguji hasil dari implementasi, yaitu perbandingan akurasi antara algoritma CART dan algoritma CMAR dari *rule-rule* yang telah terbentuk pada *tree*.

Adapun langkah-langkah yang dilakukan dalam penelitian dapat ditunjukkan pada Gambar 3.1.



Gambar 3.1 Alur Penelitian

Dari Gambar 3.1, alur penelitian yang pertama dilakukan adalah mempelajari algoritma CART dan algoritma CMAR dari berbagai literatur, dan selanjutnya menentukan kasus, kasus yang ditentukan adalah penentuan status resiko kredit. Setelah penentuan status resiko kredit, proses selanjutnya adalah merancang dan membangun sebuah perangkat lunak yang mengimplementasikan algoritma CART dan CMAR. Kemudian, setelah perancangan selesai dilakukan, perlu

dilakukan proses implementasi tersebut, dan langkah terakhir adalah evaluasi dan analisis hasil penelitian.

3.1 Analisis Data

Data yang digunakan pada penelitian ini yaitu kumpulan data tentang data nasabah pengajuan kredit perbankan di Jerman. Data merupakan data nyata yang diambil dari situs *UCI Machine Learning Repository* (http:// mlearn.ics.uci.edu/databases/statlog/german/) yang menyediakan data-data nyata untuk digunakan menguji model-model klasifikasi.

Pada data Jerman tersebut, data disimpan pada sebuah tabel. Data yang dipergunakan untuk penelitian ini adalah 21 atribut (termasuk atribut kelas).

3.2 Analisis Sistem

Pada tahap ini, akan dijelaskan tentang deskripsi umum sistem dan batasan sistem.

3.2.1 Deskripsi sistem

Sistem yang dibangun merupakan sistem yang mengimplementasikan algoritma CART dan CMAR untuk menentukan status resiko kredit nasabah. Dalam penentuan status resiko kredit ini, akan digunakan pengklasifikasian menggunakan *tree*. Parameter uji coba yang digunakan yaitu akurasi dari *model tree* yang dihasilkan antara kedua algoritma tersebut. Selanjutnya akan dianalisis juga waktu kinerja kedua algoritma tersebut.

Sistem yang dibangun ditujukan untuk pihak penyedia kredit, untuk mengkuantifikasi resiko finansial sehingga keputusan dapat diambil dengan cepat dan lebih akurat.

Adapun proses yang terjadi ketika seorang *user* menggunakan sistem yang dibangun, yaitu:

1. *User* menginputkan data *testing* nasabah pengajuan kredit yang selanjutnya akan disimpan dalam *temporary array*.
2. Sementara itu, sistem melakukan pembangunan model klasifikasi dengan menggunakan algoritma CART bila *user* memilih algoritma tersebut untuk pengklasifikasian, dan sistem akan melakukan klasifikasi menggunakan algoritma CMAR jika *user* memilih algoritma tersebut untuk

pengklasifikasian.

3. Sistem menghasilkan model klasifikasi berupa *model tree* dalam bentuk *rule*.
4. Sistem memprediksi 20 atribut yang telah dimasukkan pada pengisian *data testing*.
5. Sistem akan menampilkan hasil prediksi, *rule* yang terbentuk dan waktu kinerja sistem.

3.2.2 Batasan sistem

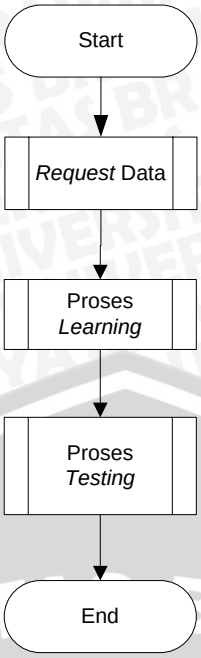
Adapun batasan pada sistem yang akan dikembangkan yaitu:

1. Proses pengklasifikasian dibagi menjadi 2, yaitu pengklasifikasian menggunakan algoritma CART dan algoritma CMAR.
2. Setelah memasukkan *data testing* dan sebelum sistem dijalankan, *user* harus memilih algoritma yang ingin digunakan dalam pengklasifikasian.
3. *Data training* dan *data testing* tidak memiliki *missing value*.
4. Untuk mengetahui akurasi masing-masing algoritma, dihitung tingkat *error*-nya dan dilihat dari proses pembentukan *tree.*
5. Sistem akan menampilkan hasil prediksi status resiko kredit berdasarkan pengklasifikasian, serta analisa perbandingan, waktu kinerja dan panjang *node* yang terpakai dari kedua algoritma tersebut.

3.3 Perancangan Sistem

3.3.1 Perancangan proses

Pada sistem yang dibangun, terdapat tiga buah proses utama yaitu : proses *request* data, proses pembangunan model (proses *learning*), dan proses *testing*. Gambar 3.2 menunjukkan alur proses utama.



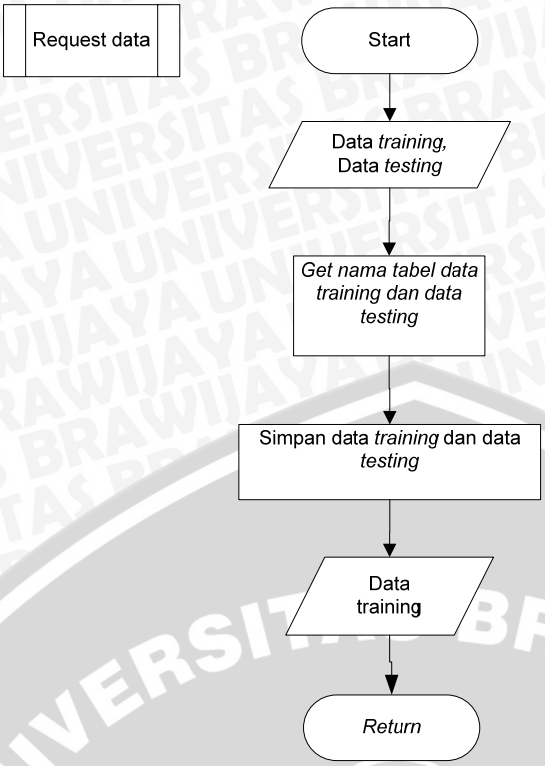
Gambar 3.2 Alur proses utama

1. Proses *request data*

Proses yang pertama dilakukan oleh sistem yaitu proses *request data*, dimana sistem akan mendapatkan data untuk diolah, yaitu *data training* dan *data testing* beserta informasinya.

Pada saat *user* pertama kali menjalankan program, *user* akan memasukkan atribut persyaratan kredit, dan memilih *data training* yang ingin digunakan untuk *testing*. Kemudian sistem akan menyimpan *data testing* ke dalam *array*.

Setelah sistem menyimpan informasi tabel *data training* dan informasi tabel *data testing* tersebut. Proses terakhir pada *request data* yaitu sistem menampilkan semua informasi *data training* yang telah disimpan kepada *user*. Proses *request* data ditunjukkan pada Gambar 3.3.



Gambar 3.3 Proses *request* data

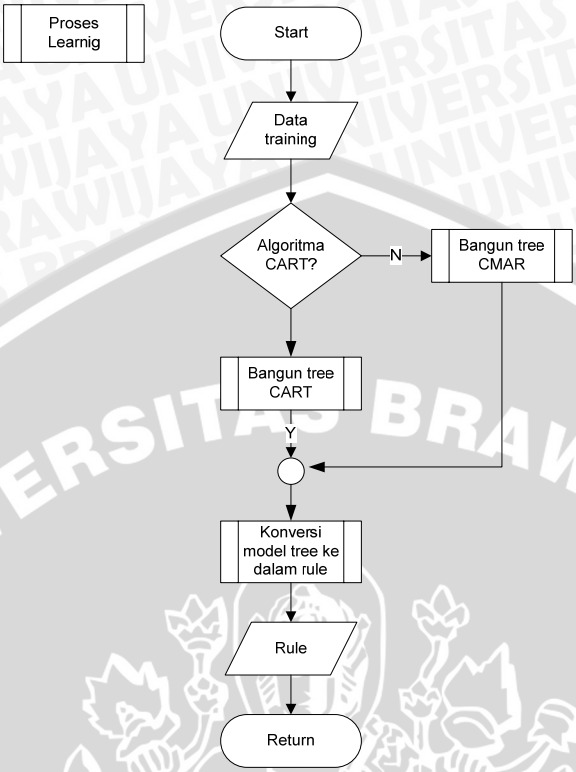
2. Proses pembangunan model (proses *learning*)

Setelah sistem mendapatkan data untuk diolah (*data training*), proses yang akan terjadi pada sistem selanjutnya yaitu proses *learning*, yang dalam permasalahan ini, diberikan 2 macam algoritma, yaitu algoritma CART dan CMAR. *Data training* dibuat bervariasi dengan menggunakan 200, 300, 500, 600, dan 700 *record data training*. Adapun langkah-langkah pada proses *learning*, yaitu:

- Sistem mendapatkan *input* berupa *data training*
- Dilakukan pemilihan algoritma yang ingin digunakan dalam pengujian
- Sistem membangun model *tree* dari *data training* (sesuai algoritma CART atau CMAR)

- Sistem menghasilkan model *tree*
- Sistem mengkonversi model *tree* yang terbentuk ke dalam bentuk *rule* dan menyimpannya
- Sistem menampilkan *rule* yang terbentuk

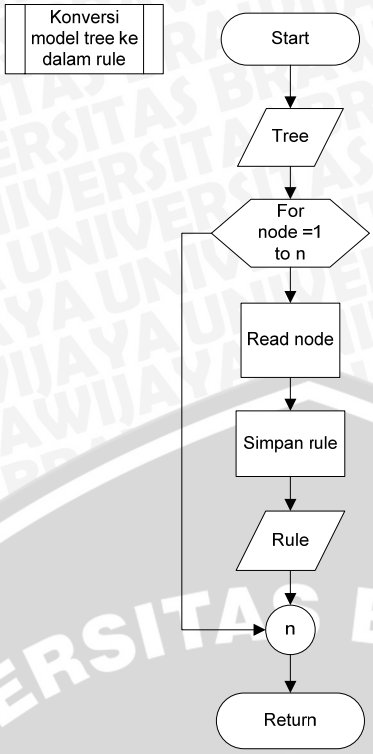
Gambar 3.4 menunjukkan proses *learning*.



Gambar 3.4 Proses *learning*

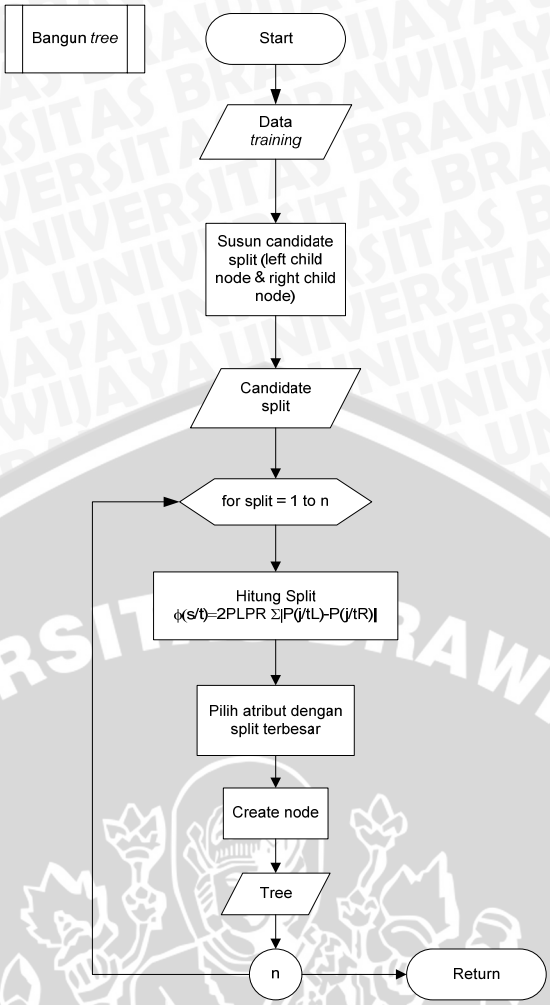
Gambar 3.5 menunjukkan proses dari konversi model *tree* ke dalam *rule*, dengan skenario sebagai berikut:

- *Input* adalah *tree*.
- Dimulai dari *node tree* = 1 sampai *node tree* ke-n.
- *Output* berupa *rule*.



Gambar 3.5 Konversi model *tree* ke dalam *rule*

Di dalam proses *learning* diterapkan 2 algoritma, yaitu algoritma CART dan CMAR. CART merupakan *binary decision tree*, yaitu suatu *tree* yang hanya memiliki 2 cabang untuk setiap *decision node*. Gambar 3.6 menunjukkan alur proses algoritma CART.



Gambar 3.6 Alur proses algoritma CART

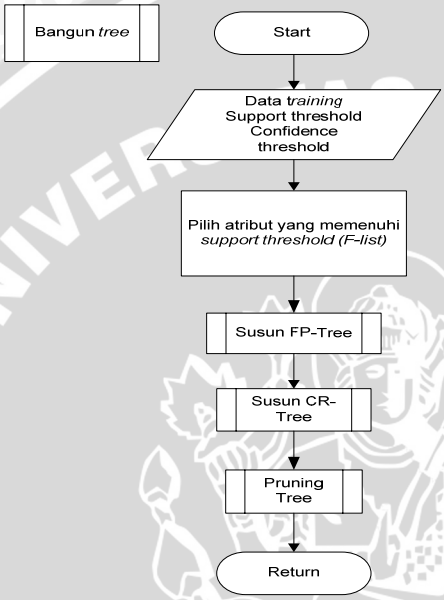
Pada algoritma CART, alur prosesnya adalah sebagai berikut:

- Data *input* berupa data *training*
- Proses menyusun *candidate splite* (*left child node* dan *right child node*) yang mengeluarkan *output candidate split*.

- Hitung *split*
- Menentukan *split* terbesar dan membentuk *node* hingga *split* ke-n

Sedangkan untuk algoritma CMAR akan ditunjukkan pada Gambar 3.7. Berikut adalah proses pembangunan algoritma CMAR:

- Data *input* berupa data *training*, *support threshold* dan *confidence threshold.*
- Memilih atribut yang memenuhi *support threshold.*
- Menyusun *FP-Tree.*
- Menyusun *CR-Tree*.
- *Pruning tree*

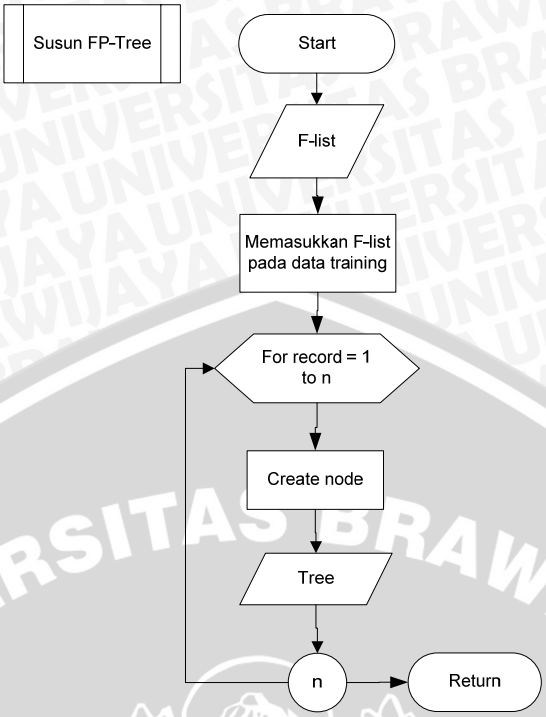


Gambar 3.7 Alur proses algoritma CMAR

Untuk proses penyusunan *FP-Tree* pada Gambar 3.8, skenario prosesnya adalah sebagai berikut:

- *Input* berupa *F-list*
- Memasukkan *F-List* pada *data training*

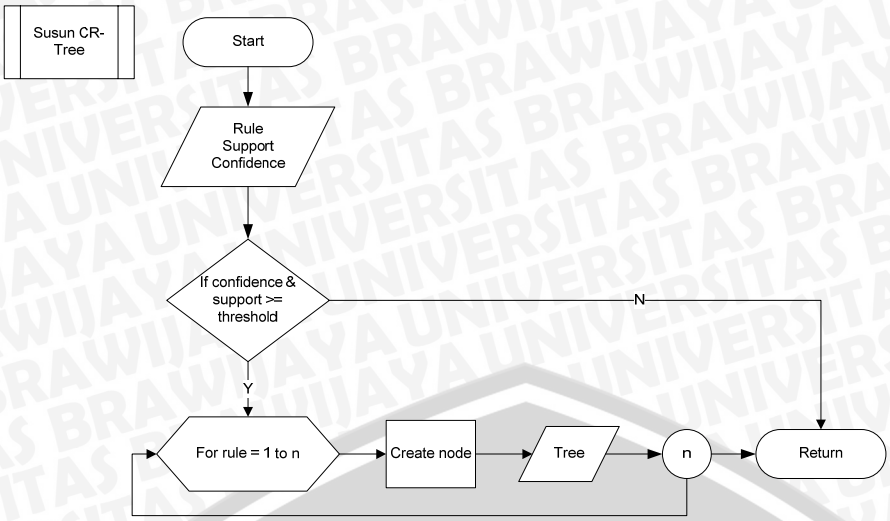
- Menyusun *tree,* dimulai dari *record* = 1 sampai *record* ke-n



Gambar 3.8 Proses pembangunan *FP-Tree*

Untuk proses penyusunan *CR-Tree* pada Gambar 3.8, skenario prosesnya adalah sebagai berikut:

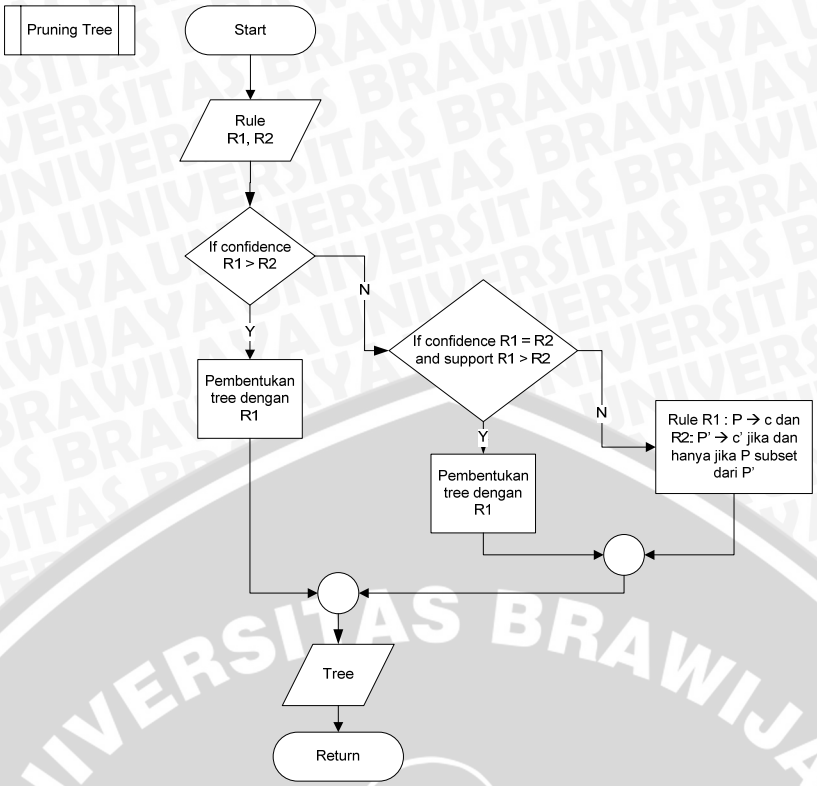
- *Input* berupa *rule*, *support* dan *confidence threshold.*
- *Jika support* dan *confidence* lebih besar dari *threshold*, maka *rule* membentuk *tree* dimulai dari *rule* = 1 sampai *rule* ke-n.
- *Output* berupa *tree.*



Gambar 3.9 Proses pembangunan *CR-Tree*

Untuk proses pruning, akan ditunjukkan pada Gambar 3.10 dengan skenario proses sebagai berikut:

- *Input* berupa rule, misal R1 dan R2.
- Jika *confidence* dari R_1 > R_2, maka *rule* yang harus dipangkas adalah R_2, dengan syarat, *rule* $R_1 : P \rightarrow c$ dan $R_2 : P' \rightarrow c'$, jika dan hanya jika P adalah *subset* dari P'
- Jika *confidence* dari R_1 dan R_2 sama, maka dilihat dari nilai support-nya, *rule* R_2 akan dipangkas jika support R_1 > R_2, dengan syarat, *rule* $R_1 : P \rightarrow c$ dan $R_2 : P' \rightarrow c'$, jika dan hanya jika P adalah *subset* dari P'.
- Jika *confidence* dari R_1 dan R_2 sama, dan *support* dari R_1 dan R_2 sama, maka dilihat dari *rule* yang memiliki nilai atribut lebih sedikit, dengan syarat, *rule* $R_1 : P \rightarrow c$ dan $R_2 : P' \rightarrow c'$, jika dan hanya jika P adalah *subset* dari P'



Gambar 3.10 Proses *pruning*

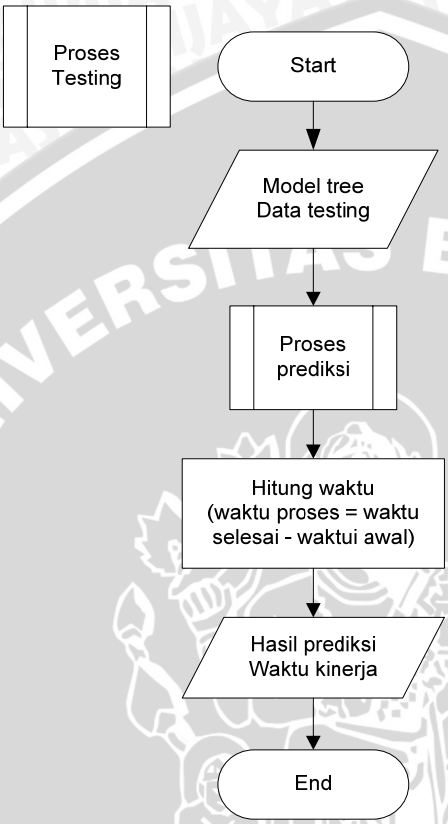
3. Proses *testing*

Proses prediksi dilakukan pada proses *testing*. Hal ini bertujuan untuk mengetahui prediksi status resiko kredit seorang pemohon kredit. Pada proses *testing*, yang dijadikan masukan adalah data nasabah pengajuan kredit perbankan di Jerman yang diambil secara *random*.

Pada proses *testing*, sistem akan menyimpan data atribut persyaratan kredit yang dimasukkan oleh *user*. Sementara itu, sistem akan mengambil *data training* yang telah disediakan, yang terdiri dari 200, 300, 500, 600 dan 700 *record*. Pengujian dengan 400 *record* data tidak dilakukan karena untuk mengetahui rata-rata waktu dan tingkat akurasi dengan kelipatan jumlah data yang berbeda. Untuk rata-rata akurasi, akan dilakukan beberapa kali percobaan.

Akurasi dihitung dengan menghitung jumlah *record* yang diklasifikasikan oleh *model tree* dengan benar (nilai kelas baru *record* sama dengan nilai kelas lama *record*) dibagi dengan jumlah *record* data *testing* yang diklasifikasikan dikalikan 100%. Sehingga, akurasi yang dihasilkan dalam satuan persen (%).

Selain menganalisis akuarsi, sistem akan menghitung waktu kinerja dari kedua algoritma, CART dan CMAR. Proses *data testing* ditunjukkan pada Gambar 3.11, dan pengembangan dari proses prediksi ditunjukkan oleh Gambar 3.12.



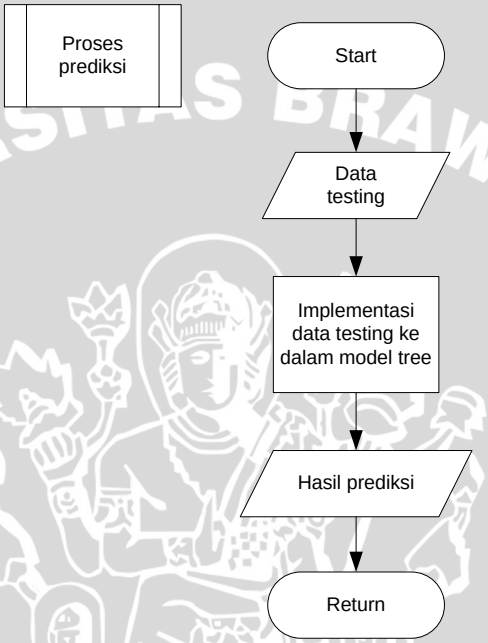
Gambar 3.11 Proses data *testing*

Pada Gambar 3.11, skenario prosesnya adalah sebagai berikut:

- *Input* berupa model *tree* dan data *testing.*
- Proses prediksi.
- Menghitung waktu, akurasi dan *node* yang terpakai.
- *Output* berupa hasil prediksi, waktu kinerja, akurasi, dan jumlah *node* yang terpakai.

Dari Gambar 3.12 berikut, skenario prosesnya adalah sebagai berikut:

- Input berupa data *testing.*
- Proses implementasi data *testing* ke dalam model *tree.*
- *Output* berupa hasil prediksi.



Gambar 3.12 Proses prediksi

3.3.2 Perancangan basis data

Setelah melalui berbagai tahapan data *preprocessing*, sistem ini memerlukan 21 tabel yang terdiri dari sebuah tabel transaksi *t_transaction* dan 20 tabel master. Berikut ini struktur dan fungsi dari tabel-tabel tersebut:

1. Tabel *t_transaction*

Tabel ini digunakan untuk menyimpan data nasabah pengajuan kredit dan berfungsi sebagai tabel transaksi yang terhubung dengan 20 tabel master. Tabel 3.1 menunjukkan tabel *t_transaction*.

Tabel 3.1 Tabel *t_transaction*

No	Nama Field	Tipe Data	Keterangan
1	**id**	varchar(5)	Nomor id (PK)
2	**kd_checking_account**	varchar(5)	Kode besar tabungan yang dijadikan jaminan
3	**kd_duration**	varchar(5)	Kode jangka waktu kredit
4	**kd_credit_history**	varchar(5)	Kode sejarah kredit yang pernah dimiliki
5	**kd_purpose**	varchar(5)	Kode tujuan pengauan kredit
6	**kd_credit_amount**	varchar(5)	Kode jumlah kredit yang diajukan
7	**kd_saving_account**	varchar(5)	Kode besar tabungan yang dimiliki
8	**kd_employment**	varchar(5)	Kode lama bekerja
9	**kd_installment_rate**	varchar(5)	Kode besar persentase bunga kredit terhadap penghasilan
10	**kd_personal_status**	varchar(5)	Kode status dan jenis kelamin
11	**kd_guarantor**	varchar(5)	Kode penjamin
12	**kd_residence_time**	varchar(5)	Kode lama tinggal
13	**kd_property**	varchar(5)	Kode asset yang dijadikan sebagai jaminan
14	**kd_age**	varchar(5)	Kode usia nasabah
15	**kd_installment**	varchar(5)	Kode rencana angsuran
16	**kd_housing**	varchar(5)	Kode status kepemilikan rumah
17	**kd_existing_kredit**	varchar(5)	Kode jumlah kredit yang dimiliki
18	**kd_job**	varchar(5)	Kode pekerjaan

19	**kd_liable_person**	varchar(5)	Kode jumlah orang tertanggung
20	**kd_telp**	varchar(5)	Kode kepemilikan telepon
21	**kd_foreigner**	varchar(5)	Kode status kewarganegaraan
22	Class	varchar(5)	Kategori resiko kredit

2. Tabel *t_checking_account*

Tabel ini digunakan untuk menyimpan data kategori besar jaminan yang dialokasikan untuk kredit lain. Tabel 3.2 menunjukkan tabel *t_checking_account*.

Tabel 3.2 Tabel *checking_account*

No	Nama Field	Tipe Data	Keterangan
1	kd_checking_account	varchar(5)	Kode besar tabungan yang dijadikan jaminan (PK)
2	checking_account	varchar(25)	Jumlah tabungan yang dijadikan jaminan

3. Tabel *t_duration*

Tabel ini digunakan untuk menyimpan data kategori jangka waktu kredit. Tabel 3.3 menunjukkan tabel *t_duration*.

Tabel 3.3 Tabel *t_duration*

No	Nama Field	Tipe Data	Keterangan
1	kd_duration	varchar(5)	Kode jangka waktu kredit (PK)
2	Duration	varchar(25)	Jangka waktu kredit

4. Tabel *t_credit_history*

Tabel ini digunakan untuk menyimpan data kategori sejarah kredit yang pernah dimiliki oleh nasabah. Tabel 3.4 menunjukkan tabel *t_credit_history*.

Tabel 3.4 Tabel *t_credit_history*

No	Nama Field	Tipe Data	Keterangan
1	kd_credit_history	varchar(5)	Kode sejarah kredit yang dimiliki (PK)
2	credit_history	varchar(50)	Sejarah kredit yang pernah dimiliki

5. Tabel *t_purpose*

Tabel ini digunakan untuk menyimpan data kategori tujuan pengajuan kredit. Tabel 3.5 menunjukkan tabel *t_purpose*.

Tabel 3.5 Tabel *t_purpose*

No	Nama Field	Tipe Data	Keterangan
1	kd_purpose	varchar(5)	Kode tujuan pengauan kredit (PK)
2	purpose	varchar(50)	Tujuan pengauan kredit

6. Tabel *t_credit_amount*

Tabel ini digunakan untuk menyimpan data kategori besar kecilnya kredit yang diajukan oleh nasabah. Tabel 3.6 menunjukkan tabel *t_credit_amount*.

Tabel 3.6 Tabel *t_credit_amount*

No	Nama Field	Tipe Data	Keterangan
1	kd_credit_amount	varchar(5)	Kode jumlah kredit yang diajukan (PK)
2	credit_amount	varchar(25)	Jumlah kredit yang diajukan

7. Tabel *t_saving_account*

Tabel ini digunakan untuk menyimpan data kategori jumlah tabungan yang dimiliki oleh nasabah. Tabel 3.7 menunjukkan tabel *t_saving_account.*

Tabel 3.7 Tabel *t_saving_account*

No	Nama Field	Tipe Data	Keterangan
1	kd_saving_account	varchar(5)	Kode jumlah tabungan yang dimiliki (PK)
2	saving_account	varchar(25)	Jumlah tabungan yang dimiliki

8. Tabel *t_employment*

Tabel ini digunakan untuk menyimpan data kategori lama bekerja seorang nasabah. Tabel 3.8 menunjukkan tabel *t_employment.*

Tabel 3.8 Tabel *t_employment*

No	Nama Field	Tipe Data	Keterangan
1	kd_employment	varchar(5)	Kode lama bekerja (PK)
2	employment	varchar(25)	Lama bekerja

9. Tabel *t_installment_rate*

Tabel ini digunakan untuk menyimpan data kategori persentase bunga kredit terhadap penghasilan. Tabel 3.9 menunjukkan tabel *t_installment_rate.*

Tabel 3.9 Tabel *t_installment_rate*

No	Nama Field	Tipe Data	Keterangan
1	kd_installment_rate	varchar(5)	Kode besar persentase bunga kredit terhadap penghasilan (PK)
2	installment_rate	integer	Persentase bunga kredit terhadap penghasilan

10. Tabel t_*personal_status*

Tabel ini digunakan untuk menyimpan data kategori status perkawinan dan jenis kelamin nasabah. Tabel 3.10 menunjukkan tabel *t_personal_status.*

Tabel 3.10 Tabel *t_personal_status*

No	Nama Field	Tipe Data	Keterangan
1	kd_personal_status	varchar(5)	Kode status dan jenis kelamin (PK)
2	personal_status	varchar(75)	Status dan jenis kelamin

11. Tabel *t_guarantor*

Tabel ini digunakan untuk menyimpan data kategori penjamin dalam transaksi kredit. Tabel 3.11 menunjukkan tabel *t_guarantor.*

Tabel 3.11 Tabel *t_guarantor*

No	Nama Field	Tipe Data	Keterangan
1	kd_guarantor	varchar(5)	Kode penjamin (PK)
2	guarantor	varchar(25)	Penjamin

12. Tabel *t_residence_time*

Tabel ini digunakan untuk menyimpan data kategori lama tinggal nasabah. Tabel 3.12 menunjukkan tabel *t_residence_time.*

Tabel 3.12 Tabel *t_residence_time*

No	Nama Field	Tipe Data	Keterangan
1	kd_residence_time	varchar(5)	Kode lama tinggal (PK)
2	residence_time	Integer	Lama tinggal

13. Tabel *t_property*

Tabel ini digunakan untuk menyimpan data kategori aset yang akan dijadikan jaminan. Tabel 3.13 menunjukkan tabel *t_property.*

Tabel 3.13 Tabel *t_property*

No	Nama Field	Tipe Data	Keterangan
1	kd_property	varchar(5)	Kode asset yang dijadikan sebagai jaminan (PK)
2	property	varchar(50)	Asset yang dijadikan sebagai jaminan

14. Tabel *t_age*

Tabel ini digunakan untuk menyimpan data kategori usia nasabah saat pengajuan kredit. Tabel 3.14 menunjukkan tabel *t_age.*

Tabel 3.14 Tabel *t_age*

No	Nama Field	Tipe Data	Keterangan
1	kd_age	varchar(5)	Kode usia nasabah (PK)
2	Age	varchar(25)	Usia nasabah

15. Tabel *t_installment*

Tabel ini digunakan untuk menyimpan data kategori rencana angsuran yang dipilih nasabah. Tabel 3.15 menunjukkan tabel t_*installment.*

Tabel 3.15 Tabel *t_installment*

No	Nama Field	Tipe Data	Keterangan
1	kd_installment	varchar(5)	Kode rencana angsuran (PK)
2	installment	varchar(25)	Rencana angsuran

16. Tabel *t_housing*

Tabel ini digunakan untuk menyimpan data kategori status kepemilikan rumah. Tabel 3.16 menunjukkan tabel *t_housing.*

Tabel 3.16 Tabel *t_housing*

No	Nama Field	Tipe Data	Keterangan
1	kd_housing	varchar(5)	Kode status kepemilikan rumah (PK)
2	Housing	varchar(25)	Status kepemilikan rumah

17. Tabel *t_existing_credit*

Tabel ini digunakan untuk menyimpan data kategori usia jumlah kredit yang dimiliki oleh nasabah. Tabel 3.17 menunjukkan tabel *t_existing_credit.*

Tabel 3.17 Tabel *t_existing_credit*

No	Nama Field	Tipe Data	Keterangan
1	kd_existing_credit	varchar(5)	Kode jumlah kredit yang dimiliki (PK)
2	existing_credit	Integer	Jumlah kredit yang dimiliki

18. Tabel *t_job*

Tabel ini digunakan untuk menyimpan data kategori pekerjaan nasabah. Tabel 3.18 menunjukkan tabel *t_job.*

Tabel 3.18 Tabel *t_job*

No	Nama Field	Tipe Data	Keterangan
1	kd_job	varchar (5)	Kode pekerjaan (PK)
2	job	varchar(75)	Pekerjaan

19. Tabel *t_liable_person*

Tabel ini digunakan untuk menyimpan data kategori jumlah orang tertanggung. Tabel 3.19 menunjukkan tabel *t_liable_person.*

Tabel 3.19 Tabel *t_job*

No	Nama Field	Tipe Data	Keterangan
1	kd_liable_person	varchar(5)	Kode jumlsh orsng tertanggung (PK)
2	liable_person	Integer	Jumlah orang tertanggung

20. Tabel *t_telp*

Tabel ini digunakan untuk menyimpan data kategori jumlah orang tertanggung. Tabel 3.20 menunjukkan tabel *t_telp.*

Tabel 3.20 Tabel *t_job*

No	Nama Field	Tipe Data	Keterangan
1	kd_telp	varchar(5)	Kode kepemilikan telepon (PK)
2	Telp	varchar (50)	Kepemilikan telepon

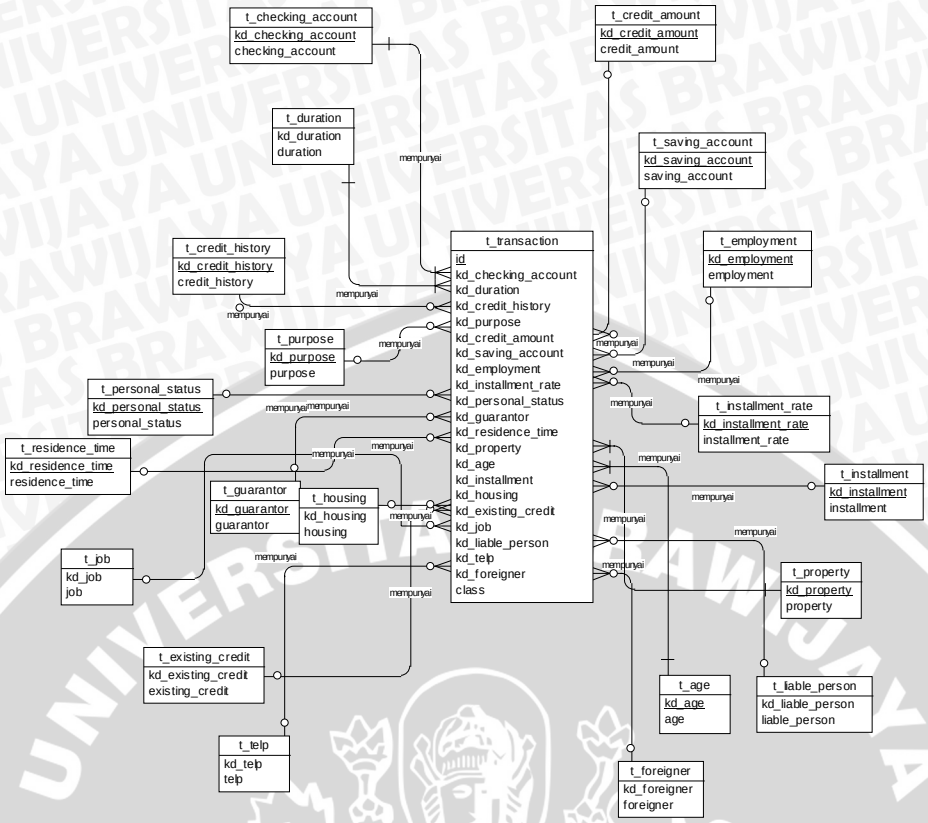
21. Tabel *t_foreigner*

Tabel ini digunakan untuk menyimpan data kategori status kewarganegaraan. Tabel 3.21 menunjukkan tabel *t_foreigner.*

Tabel 3.21 Tabel *t_foreigner*

No	Nama Field	Tipe Data	Keterangan
1	kd_foreigner	varchar(5)	Kode status kewarganegaraan (PK)
2	foreigner	varchar (25)	Status Kewarganegaraan

Pada Gambar 3.13 menunjukkan struktur relasi antar tabel.



Gambar 3.13 Struktur dan relasi antar tabel

3.3.3 Perhitungan manual

1. Perhitungan manual algoritma CART

CART merupakan *binary decision tree*, yaitu suatu *tree* yang hanya memiliki 2 cabang untuk setiap *decision node*. Proses pertama pada algoritma ini adalah menyiapkan *data training*, jika *data training* tidak sama dengan 0, maka atribut *split* harus ditentukan. Proses selanjutnya adalah menghitung *split*, dan menetapkan *split* tertinggi sebagai *root node*. Berikut adalah algoritma dari CART.

```
Training data
Susun candidate split
(predictor variable = 2)
Hitung split
```

$$\varphi\left(\frac{s}{t}\right) = 2P_L P_R \sum_{j=1}^{\#\,classes} |P\left(\frac{j}{t_L}\right) - P(j/t_R)$$

```
Sort  split
       for (atribut : sort split)
       if (sort split =1)
       root node
   else node
end
```

Langkah-langkah algoritma CART dalam contoh perhitungan *split* untuk membangun *tree* adalah sebagai sebagai berikut:

1. Sistem menerima masukan *data training*. Contoh tabel *data training* dapat dilihat pada Tabel 3.22

2. Pada pembuatan *candidate split*, ditentukan atribut yang masuk *node* sebelah kiri atau *node* sebelah kanan. Pada Tabel 3.23, merupakan tabel *candidate split* dari *data training.*

Tabel 3.23 Tabel *candidate split*

candidate split	left child node (tL)	right child node (tR)
1	checking account=a11	checking account={a12,a13,a14}
2	checking account=a12	checking account={a11,a13,a14}
3	checking account=a13	checking account={a11,a12,a14}
4	checking account=a14	checking account={a11,a12,a13}
5	duration =a21	duration ={a22,a23,a24}
6	duration =a22	duration ={a21,a23,a24}
7	duration =a23	duration ={a21,a22,a24}
8	duration =a24	duration ={a21,a22,a23}
9	credit_history=a32	credit_history={a33,a34}
10	credit_history=a33	credit_history={a32,a34}
11	credit_history=a34	credit_history={a32,a33}
12	purpose=a40	purpose={a42,a43,a46}
13	purpose=a42	purpose={a40,a43,a46}
14	purpose=a43	purpose={a40,a42,a46}
15	purpose=a46	purpose={a40,a42,a43}
16	credit_amount = a51	credit_amount = {a52,a53,a54}
17	credit_amount = a52	credit_amount = {a51,a53,a54}
18	credit_amount = a53	credit_amount = {a51,a52,a54}
19	credit_amount = a54	credit_amount = {a51,a52,a53}
20	saving acount=a61	saving account={a63, a65}
21	saving acount=a63	saving account={a61, a65}
22	saving acount=a65	saving account={a61,a63}
23	employment =a71	employment ={a73,a74,a75}
24	employment =a73	employment ={a71,,a74,a75}
25	employment =a74	employment ={a71,a73,a75}
26	employment =a75	employment ={a71,a73,a74}
27	instalment rate =2	instalment rate ={3,4}
28	instalment rate =3	instalment rate ={2,4}
29	instalment rate =4	instalment rate ={2,3}
30	sex=a92	sex={a93,a94}
31	sex=a93	sex={a92,a94}
32	sex=a94	sex={a92,a93}
33	guarantor=a101	guarantor={a102,a103}

34	guarantor=a102	guarantor={a101,a103}
35	guarantor=a103	guarantor={a101,a102}
36	residence=1	residence={2,3,4)
37	residence=2	residence={1,3,4}
38	residence=3	residence={1,2,4}
39	residence=4	residence={1,2,3}
40	property = a121	property = {a122,a123,a124}
41	property = a122	property = {a121,a123,a124}
42	property = a123	property = {a121,a122,a124}
43	property = a124	property = {a121,a122,a123}
44	age = a131	age = {a132,a133,a134}
45	age = a132	age = {a131,a133,a134}
46	age = a133	age = {a131,a132,a134}
47	age = a134	age = {a131,a132,a133}
48	installment=a142	installment =a143
49	installment=a143	installment =a142
50	housing=a152	housing=a153
51	housing=a153	housing=a152
52	existing credit = 1	existing credit = 2
53	existing credit = 2	existing credit = 1
54	job=a172	job={a173,a174}
55	job=a173	job={a172,a174}
56	job=a174	job={a172,a173}
57	liable person =1	liable person =2
58	liable person =2	liable person =1
59	telp=a191	telp=a192
60	telp=a192	telp=a191
61	foreigner=a201	foreigner=a202
62	foreigner=a202	foreigner=a201

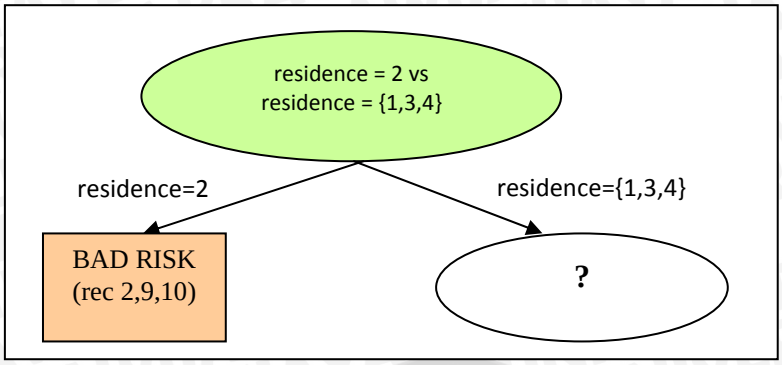
3. Kemudian, untuk *candidate split* tersebut, dihitung variabel-variabelnya berdasarkan *data training* yang dimiliki. Dalam perhitungan variabel-variabel ini, data dimasukkan dalam *temporary array*. Rumus untuk menghitung *split* seperti ditunjukkan pada persamaan 2.1. Setelah perhitungan *split* selesai, harus ditentukan *split* tertinggi, yang akan dijadikan sebagai *root node*. Pada Tabel 3.24, akan ditunjukkan perhitungan *split* dari *data training* pada Tabel 3.22.

Tabel 3.24 Perhitungan *split iterasi* 1

split	pl	pr	P(j/tL)		P(j/tR)		2plpr	q(s/t)	Φ(s/t)
			G	B	G	B			
1	0.3	0.7	0.67	0.33	0.57	0.43	0.42	0.19	0.08
2	0.2	0.8	0	1	0.75	0.25	0.32	1.5	0.48
3	0.1	0.9	0	1	0.67	0.33	0.18	1.33	0.24
4	0.4	0.6	1	0	0.33	0.67	0.48	1.33	0.64
5	0.3	0.7	1	0	0.43	0.57	0.42	1.14	0.48
6	0.3	0.7	0.33	0.67	0.71	0.29	0.42	0.76	0.32
7	0.2	0.8	0.5	0.5	0.625	0.375	0.32	0.25	0.08
8	0.2	0.8	0.5	0.5	0.625	0.375	0.32	0.25	0.08
9	0.5	0.5	0.6	0.4	0.6	0.4	0.5	0	0
10	0.1	0.9	0	1	0.67	0.33	0.18	1.33	0.24
11	0.4	0.6	0.75	0.25	0.5	0.5	0.48	0.5	0.24
12	0.2	0.8	0	1	0.75	0.25	0.32	1.5	0.48
13	0.3	0.7	1	0	0.43	0.57	0.42	1.14	0.48
14	0.3	0.7	0.33	0.67	0.71	0.29	0.42	0.76	0.32
15	0.2	0.8	1	0	0.5	0.5	0.32	1	0.32
16	0.4	0.6	0.75	0.25	0.5	0.5	0.48	0.5	0.24
17	0.2	0.8	0.5	0.5	0.625	0.375	0.32	0.25	0.08
18	0.2	0.8	0	1	0.75	0.25	0.32	1.5	0.48
19	0.2	0.8	1	0	0.5	0.5	0.32	1	0.32
20	0.6	0.4	0.33	0.67	1	0	0.48	1.33	0.64
21	0.1	0.9	1	0	0.56	0.44	0.18	0.89	0.16
22	0.3	0.7	1	0	0.43	0.57	0.42	1.14	0.48
23	0.1	0.9	0	1	0.67	0.33	0.18	1.33	0.24
24	0.5	0.5	0.4	0.6	0.8	0.2	0.5	0.8	0.4
25	0.2	0.8	1	0	0.5	0.5	0.32	1	0.32
26	0.2	0.8	1	0	0.5	0.5	0.32	1	0.32
27	0.5	0.5	0.8	0.2	0.4	0.6	0.5	0.8	0.4
28	0.2	0.8	0.5	0.5	0.63	0.38	0.32	0.25	0.08
29	0.3	0.7	0.33	0.67	0.71	0.29	0.420	0.76	0.32
30	0.1	0.9	0	1	0.67	0.33	0.18	1.33	0.24
31	0.7	0.3	0.71	0.29	0.33	0.67	0.42	0.76	0.32
32	0.2	0.8	0.5	0.5	0.63	0.38	0.32	0.25	0.08
33	0.8	0.2	0.63	0.38	0.5	0.5	0.32	0.25	0.08
34	0.1	0.9	0	1	0.67	0.33	0.18	1.33	0.24
35	0.1	0.9	1	0	0.56	0.44	0.18	0.89	0.16
36	0.1	0.9	1	0	0.56	0.44	0.18	0.89	0.16
37	0.3	0.7	0	1	0.86	0.14	0.42	1.71	0.72
38	0.1	0.9	1	0	0.56	0.44	0.18	0.89	0.16
39	0.5	0.5	0.8	0.2	0.4	0.6	0.5	0.8	0.4
40	0.4	0.6	0.5	0.5	0.67	0.33	0.48	0.33	0.16
41	0.2	0.8	1	0	0.5	0.5	0.32	1	0.32
42	0.2	0.8	0.5	0.5	0.63	0.38	0.32	0.25	0.08

43	0.2	0.8	0.5	0.5	0.63	0.38	0.32	0.25	0.08
44	0.3	0.7	0.33	0.67	0.71	0.29	0.42	0.76	0.32
45	0.4	0.6	0.75	0.25	0.5	0.5	0.48	0.5	0.24
46	0.2	0.8	0.5	0.5	0.625	0.375	0.32	0.25	0.08
47	0.1	0.9	1	0	0.56	0.44	0.18	0.89	0.16
48	0.1	0.9	0	1	0.67	0.33	0.18	1.33	0.24
49	0.9	0.1	0.67	0.33	0	1.000	0.18	1.33	0.24
50	0.7	0.3	0.57	0.43	0.67	0.33	0.42	0.19	0.08
51	0.3	0.7	0.67	0.33	0.57	0.43	0.42	0.19	0.08
52	0.6	0.4	0.67	0.33	0.50	0.50	0.48	0.33	0.16
53	0.4	0.6	0.5	0.5	0.67	0.33	0.48	0.33	0.16
54	0.2	0.8	1	0	0.5	0.5	0.32	1	0.32
55	0.7	0.3	0.57	0.43	0.67	0.33	0.42	0.19	0.08
56	0.1	0.9	0	1	0.67	0.33	0.18	1.33	0.24
57	0.6	0.4	0.5	0.5	0.75	0.25	0.48	0.5	0.24
58	0.4	0.6	0.75	0.25	0.5	0.5	0.48	0.5	0.24
59	0.8	0.2	0.5	0.5	1	0	0.32	1	0.32
60	0.2	0.8	1	0	0.5	0.5	0.32	1	0.32
61	0.9	0.1	0.56	0.44	1	0	0.18	0.89	0.16
62	0.1	0.9	1	0	0.56	0.44	0.18	0.89	0.16

4. Dari contoh perhitungan di atas, yang memiliki nilai *goodness of split* (Φ(s/t)) terbesar, yaitu *split* 37 dengan nilai 0.72, jadi *split* 37 yang akan digunakan pada *root node*, yaitu *split* dengan *residence = 2 vs residence = {1,3,4}*.
5. Untuk penentuan percabangan, dapat dilihat bahwa dengan *residence = 2*, didapatkan *pure node leaf*, yaitu *bad risk* (untuk *record 2, 9,* dan *10).* Sedangkan untuk *residence = {1,3,4},* masih terdapat dua nilai, yaitu *good risk* dan *bad credit risk.* Sehingga percabangan untuk *residence = {1,3,4}* memiliki *decision node* baru. Adapun pemilihan *split* yang aka digunakan, yaitu dengan menyusun perhitungan nilai Φ(s/t) yang baru, tanpa melihat *split 37, record 2, 9,*dan *10.* Gambar 3.14 menunjukkan *tree* pada *iterasi* 1 untuk *root node.*



Gambar 3.14 Iterasi 1 untuk *root node* algoritma CART

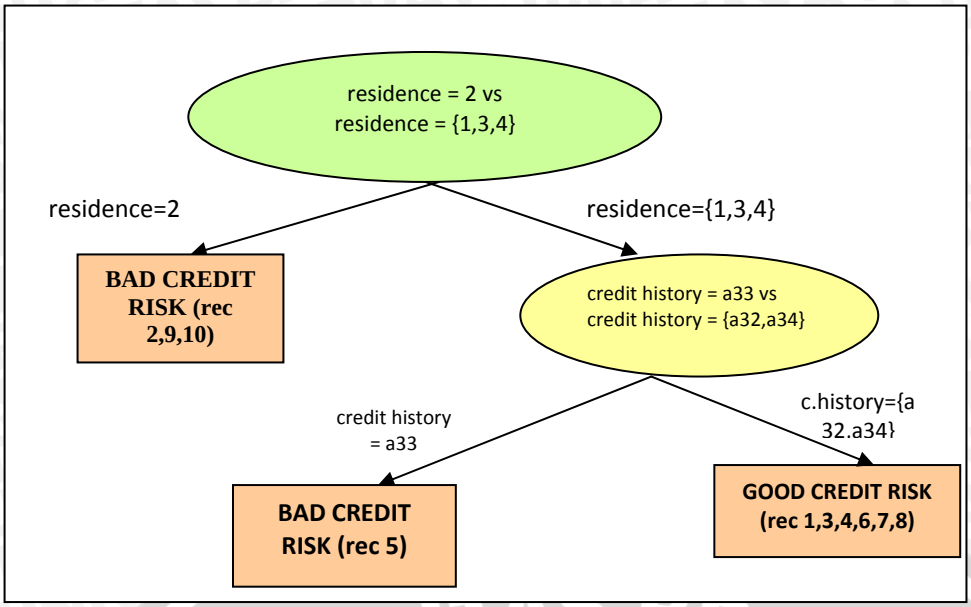
6. Untuk penentuan percabangan *residence = {1,3,4}*, maka perlu dihitung *split* yang baru tanpa melihat *split* 37, *record 2, 9,* dan *10*. Tabel 3.25 menunjukkan perhitungan *split* iterasi *2* yang merupakan *split* baru, tanpa melihat *split* 37.

Tabel 3.25 Perhitungan *split* iterasi 2

split	pl	pr	P(j/tL)		P(j/tR)		2plpr	q(s/t)	Φ(s/t)
			G	B	G	B			
1	0.43	0.57	0.67	0.33	1	0	0.489796	0.667	0.326531
2	0	1	0	0	0.86	0.14	0	1	0
3	0	1	0	0	0.86	0.14	0	1	0
4	0.6	0.43	1	0	0.67	0.33	0.49	0.67	0.33
5	0.43	0.57	1	0	0.75	0.25	0.49	0.5	0.24
6	0.29	0.71	0.5	0.5	1	0	0.41	1	0.41
7	0.14	0.86	1	0	0.83	0.17	0.24	0.33	0.08
8	0.1	0.86	1	0	0.83	0.17	0.24	0.33	0.08
9	0.43	0.57	1	0	0.75	0.25	0.49	0.5	0.24
10	0.14	0.86	0	1	1	0	0.24	2	0.49
11	0.43	0.57	1	0	0.75	0.25	0.49	0.5	0.24
12	0.14	0.86	0	1	1	0	0.24	2	0.49
13	0.43	0.57	1	0	0.75	0.25	0.49	0.5	0.24
14	0.14	0.86	1	0	0.83	0.17	0.24	0.33	0.08
15	0.29	0.71	1	0	0.8	0.2	0.41	0.4	0.16
16	0.43	0.57	1	0	0.75	0.25	0.49	0.5	0.24
17	0.29	0.71	0.5	0.5	1	0	0.41	1	0.41
18	0	1	0	0	0.86	0.14	0	1	0
19	0.29	0.71	1	0	0.8	0.2	0.41	0.4	0.16
20	0.43	0.57	0.67	0.33	1	0	0.49	0.67	0.33
21	0.14	0.86	1	0	0.83	0.17	0.24	0.33	0.08
22	0.43	0.57	1	0	0.75	0.25	0.49	0.5	0.24

23	0	1	0	0	0.86	0.14	0	1	0
24	0.43	0.57	0.67	0.33	1	0	0.49	0.67	0.33
25	0.29	0.71	1	0	0.8	0.2	0.41	0.4	0.16
26	0.29	0.71	1	0	0.8	0.2	0.41	0.4	0.16
27	0.57	0.43	1	0	0.67	0.33	0.49	0.67	0.33
28	0.29	0.71	0.5	0.5	1	0	0.41	1	0.41
29	0.14	0.86	1	0	1	0.17	0.24	0.33	0.08
30	0	1	0	0	0.86	0.14	0	1	0
31	0.86	0.14	0.83	0.17	1	0	0.24	0.33	0.08
32	0.14	0.86	1	0	0.83	0.17	0.24	0.33	0.08
33	0.86	0.14	0.83	0.17	1	0	0.24	0.33	0.08
34	0	1	0	0	0.86	0.14	0	1	0
35	0.14	0.86	1	0	0.83	0.17	0.24	0.33	0.08
36	0.14	0.86	1	0	0.83	0.17	0.24	0.33	0.08
37									
38	0.14	0.86	1	0	0.83	0.17	0.24	0.33	0.08
39	0.71	0.29	0.8	0.2	1	0	0.41	0.4	0.16
40	0.29	0.71	1	0	0.8	0.2	0.41	0.4	0.16
41	0.29	0.71	1	0	0.8	0.2	0.41	0.40	0.16
42	0.14	0.86	1	0	0.83	0.17	0.24	0.33	0.08
43	0.29	0.71	0.5	0.5	1	0	0.41	1	0.41
44	0.14	0.86	1	0	0.83	0.167	0.24	0.33	0.08
45	0.43	0.57	1	0	0.75	0.25	0.49	0.5	0.24
46	0.29	0.71	0.5	0.5	1	0	0.41	1	0.41
47	0.14	0.86	1	0	0.83	0.167	0.24	0.33	0.08
48	0	1	0	0	0.86	0.14	0	1	0
49	1	0	0.86	0.14	0	0	0	1	0
50	0.57	0.43	1	0	0.67	0.33	0.49	0.67	0.33
51	0.43	0.57	0.67	0.33	1	0	0.49	0.67	0.33
52	0.57	0.43	1	0	0.67	0.33	0.49	0.67	0.33
53	0.43	0.57	0.67	0.33	1	0	0.49	0.67	0.33
54	0.29	0.71	1	0	0.75	0.25	0.41	0.5	0.20
55	0.71	0.29	0.8	0.2	1	0	0.41	0.4	0.16
56	0	1	0	0	0.86	0.14	0	1	0
57	0.43	0.57	1	0	0.75	0.25	0.49	0.5	0.24
58	0.57	0.43	0.75	0.25	1	0	0.49	0.5	0.24
59	0.71	0.29	0.8	0.2	1	0	0.41	0.4	0.16
60	0.29	0.71	1	0	0.8	0.2	0.41	0.4	0.16
61	0.86	0.14	0.83	0.17	1	0	0.24	0.33	0.08
62	0.14	0.86	1	0	0.83	0.17	0.24	0.33	0.08

7. Dari perhitungan di atas, yang memiliki nilai *goodness of split* ($\Phi_{(s/t)}$) terbesar, yaitu *split* 10 dengan nilai 0.49. Oleh karena itu, *split* 10 yang akan digunakan untuk menentukan cabang dari *purpose*. Gambar 3.15 menunjukkan *decision tree* setelah *record* yang mengandung *split* 10 dimasukkan.



Gambar 3.15 *Decision tree* CART utuh

8. Karena struktur *tree* sudah seimbang dan masing-masing *credit hidtory = a33* dan *credit history = {a32, a34}* sudah memiliki kelas yang berbeda, yaitu *bad risk* dan *good risk*, maka penyusunan *tree* telah selesai dilakukan. Oleh karena itu, Gambar 3.15 disebut *decision tree* utuh.

2. Perhitungan manual algoritma CMAR

Pada pengklasifikasian menggunakan proses algoritma CMAR, prosedur perhitungan manualnnya adalah sebagai berikut:

- *Training* data dan menentukan *support* dan *confidence threshold.*
- Menghitung jumlah atribut yang sering muncul (*frequent item set* yang kemudian disebut *support)*, kemudian mengurutkan secara *descending.*
- Menetukan atribut yang memenuhi *support threshold.*
- Memasukkan *F-List* ke dalam data *training*.
- Menyusun *FP-Tree*.

- Menghitung *support rule* dan *confidence.*
- Menentukan atribut yang nenenuhi *confidence threshold.*
- *Pruning* tree.

Prosedur tersebut jika diimplementasikan dalam kasus penentuan resiko kredit adalah sebagai berikut:

1. Sistem menerima masukan data *training* dan ditentukan nilai *support threshold* = 5 dan *confidence threshold* = 50%. Contoh tabel *data training* dapat dilihat pada Tabel 3.22
2. Mencari *frequent item set* pada *data training. Frequent item set* merupakan nilai dari atribut yang paling sering muncul. Tabel 3.26 menunjukkan *frequent item set* pada masing-masing atribut.

Tabel 3.26 Atribut dan jumlah *frequent item set*

id_atribut	range atribut	Frequent Item set
1	checking account=a11	3
2	checking account=a12	2
3	checking account=a13	1
4	checking account=a14	4
5	duration 0 – 12	3
6	duration 13- 24	3
7	duration 25-36	2
8	duration 37-48	2
9	credit_history=a32	5
10	credit_history=a33	1
11	credit_history=a34	4
12	purpose=a40	2
13	purpose=a42	3
14	purpose=a43	3
15	purpose=a46	2
16	credit_amount 0-2500	4
17	credit_amount 2501- 5000	2
18	credit_amount 5001-7500	2
19	credit_amount 7501-10000	2
20	saving acount=a61	6
21	saving acount=a63	1
22	saving acount=a65	3
23	employment =a71	1

24	employment =a73	5
25	employment =a74	2
26	employment =a75	2
27	instalment rate =2	5
28	instalment rate =3	2
29	instalment rate =4	3
30	personal status=a92	1
31	personal status =a93	7
32	personal status =a94	2
33	guarantor=a101	8
34	guarantor=a102	1
35	guarantor=a103	1
36	residence time=1	1
37	residence time=2	3
38	residence time=3	1
39	residence time=4	5
40	property = a121	4
41	property = a122	2
42	property = a123	2
43	property = a124	2
44	age 19-34	3
45	age 35-49	4
46	age 50-64	2
47	age 65-75	1
48	installment=a142	1
49	installment =a143	9
50	housing=a152	7
51	housing=a153	3
52	existing credit = 1	6
53	existing credit = 2	4
54	job=a172	2
55	job=a173	7
56	job=a174	1
57	liable person=1	6
58	liable person=2	4
59	telp=a191	8
60	telp=a192	2
61	foreigner=a201	9
62	foreigner=a202	1

3. Setelah *data training* ditentukan *frequent item set*-nya, maka dapat ditentukan atribut yang memenuhi *support threshold*.

Atribut yang memenuhi *support threshold* terdapat pada Tabel 3.27.

Tabel 3.27 *F-list* dengan jumlah *frequent item set*

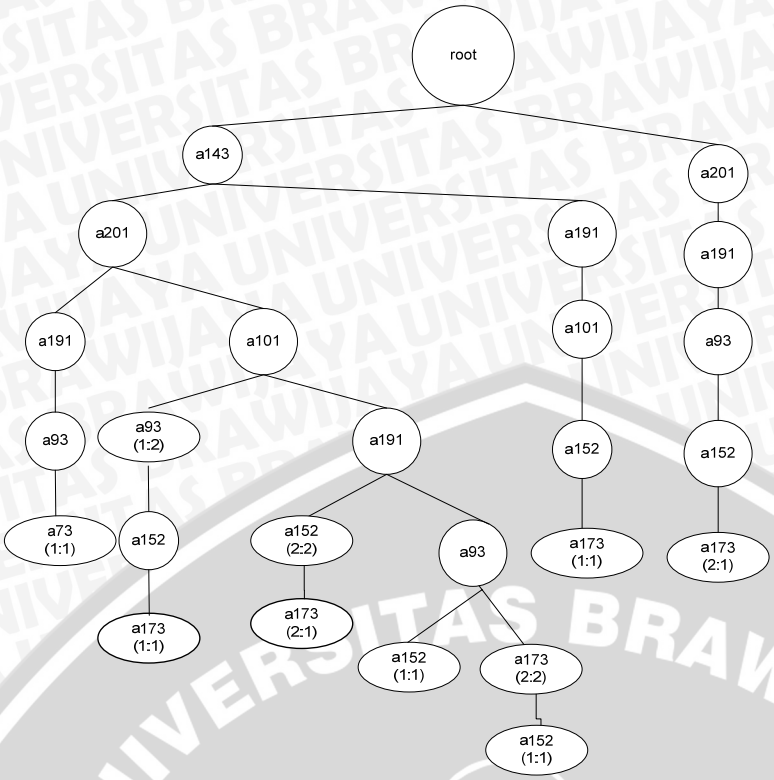
Id_atribut	range atribut	frequent itemset
1	installment=a143	9
2	foreigner=a201	9
3	guarantor=a101	8
4	telp=a191	8
5	personal status=a93	7
6	housing=a152	7
7	job=a173	7
8	saving acount=a61	6
9	existing credit= 1=a161	6
10	liable person=1=a181	6
11	credit_history=a32	5
12	employment =a73	5
13	installment rate =2=a82	5
14	residence time=4=a114	5

4. Setelah *F-list* ditemukan, maka atribut-atribut tersebut dimasukkan ke dalam *data training*. Tabel 3.28 menunjukkan atribut dari *F-list* yang telah dimasukkan ke dalam *data training* dan membentuk *rule.*

Tabel 3.28 *F-list* yang telah dimasukkan ke dalam data *training*

Record	rule	Class
1	a93,a191,a143,a152,a173,a201	1
2	a101,a143,a152,a173,a191,a201	2
3	a93,a101,a143,a152,a191,a201	1
4	a93,a143,a173,a191,a201	1
5	a93,a101,a143,a173,a191,a201	2
6	a93,a101,a143,a201	1
7	a93,a101,a143,a152,a173,a191,a201	1
8	a101,a143,a152,a173,a191	1
9	a93,a152,a173,a191,a201	2
10	a101,a143,a152,a191,a201	2

5. Setelah *F-list* dimasukkan ke dalam data *training*, maka dapat disusun *FP-tree* seperti Gambar 3.16.



Gambar 3.16 *FP-tree*

6. Setelah terbentuk *FP-tree* seperti Gambar 3.16, maka langkah selanjutnya adalah menentukan subset dari tiap-tiap rule untuk memudahkan perhitungan *support* dan *confidence.* Tabel 3.29 menunjukkan *subset* dari tiap-tiap *rule*.

Tabel 3.29 *Subset* tiap-tiap *rule*

rule id	Subset dengan kelas yang sama	Subset dengan kelas yang berbeda
1	1,7	1,7,8
2	2.3.6.7	2.3.6.7.8
3	3.7	3.7
4	4.5	4.5.6.7
5	5	5.7
6	6.7	6.7

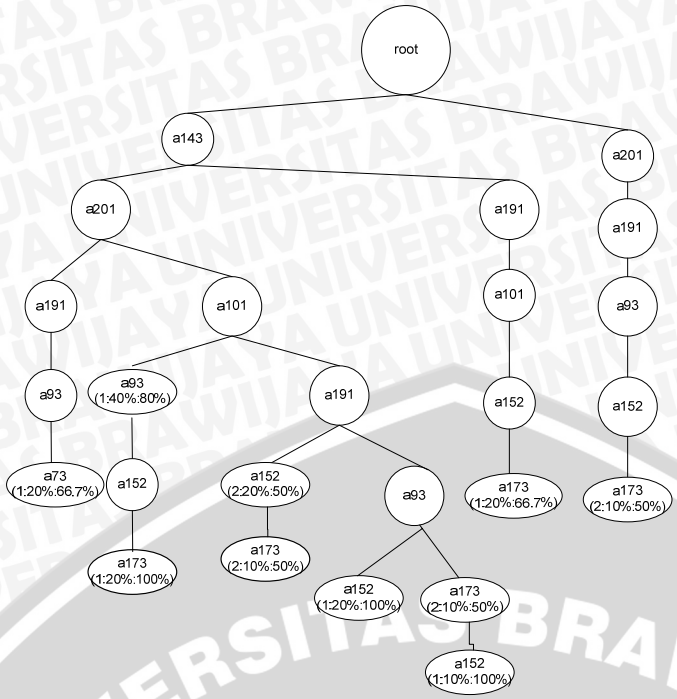
7	7	7
8	8	7.8
9	7.9	5.7.9
10	10	7.10

7. Menghitung *support rule* dan *confidence*. Tabel 3.18 menunjukkan *rule, support rule* dan *confidence*.

Tabel 3.30 *Rule, support rule* dan *confidence*

rule id	Rule	Support rule (%)	Conf (%)
1	a143,a201,a191,a93,a173 → 1	20	66.66667
2	a143,a201,a101,a93 → 1	40	80
3	a143,a201,a101,a93,a152,a173 → 1	20	100
4	a143,a201,a101,a191,a152 → 2	20	50
5	a143,a201,a101,a191,a152,a173 → 2	10	50
6	a143,a201,a101,a191,a93,a152 → 1	20	100
7	a143,a201,a101,a191,a93,a152,a173 → 1	10	100
8	a143,a201,a101,a191,a93,a173 → 2	10	50
9	a143,a191,a101,a152,a173 → 1	20	66.66667
10	a201,a191,a93,a152,a173 → 2	10	50

8. Dari *confidence* yang telah ditemukan, maka dapat dibentuk *CR-tree*, karena syarat *confidence* telah memenuhi *threshold confidence*, yaitu 50%. Gambar 3.17 menunjukkan gambar dari *CR-tree*.



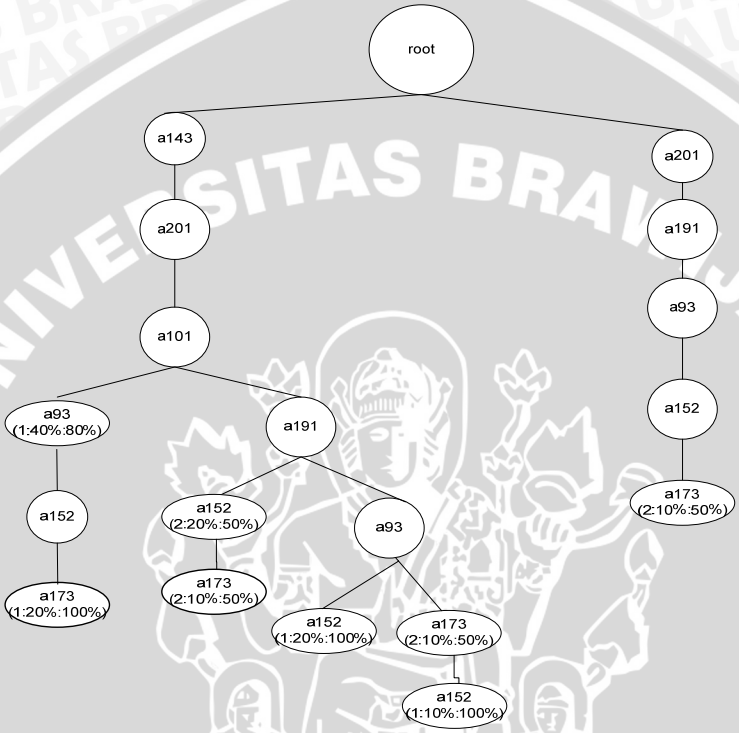
Gambar 3.17 *CR-tree*

9. Untuk *pruning* pada CMAR, dapat dipenuhi dari ketiga syarat berikut:
   - Jika *confidence* dari R_1 > R_2, maka *rule* yang harus dipangkas adalah R_2.
   - Jika *confidence* dari R_1 dan R_2 sama, maka dilihat dari nilai support-nya, *rule* R_2 akan dipangkas jika support $R_1 > R_2$.
   - Jika *confidence* dari R_1 dan R_2 sama, dan *support* dari R_1 dan R_2 sama, maka dilihat dari *rule* yang memiliki nilai atribut lebih sedikit, dengan syarat, *rule* $R_1 : P \rightarrow c$ dan $R_2 : P' \rightarrow c'$, jika dan hanya jika P adalah *subset* dari P'.
10. Setelah *pruning*, maka didapatkan *rule* yang tidak ikut dipangkas, yaitu *rule 3, 4, 5, 6, 7, 8* ,dan *10*. Kemudian, dapat dibuat *tree* baru setelah proses pemangkasan. Tabel

3.31 menunjukkan *rule-rule* setelah *pruning* dan Gambar 3.18 menunjukkan *tree* setelah *pruning.*

Tabel 3.31 *Rule, support rule* dan *confidence* setelah *pruning*

rule id	Rule	Support rule (%)	Conf (%)_
3	a143,a201,a101,a93,a152,a173 → 1	20	100
4	a143,a201,a101,a191,a152 → 2	20	50
5	a143,a201,a101,a191,a152,a173 → 2	10	50
6	a143,a201,a101,a191,a93,a152 → 1	20	100
7	a143,a201,a101,a191,a93,a152,a173 → 1	10	100
8	a143,a201,a101,a191,a93,a173 → 2	10	50
10	a201,a191,a93,a152,a173 → 2	10	50



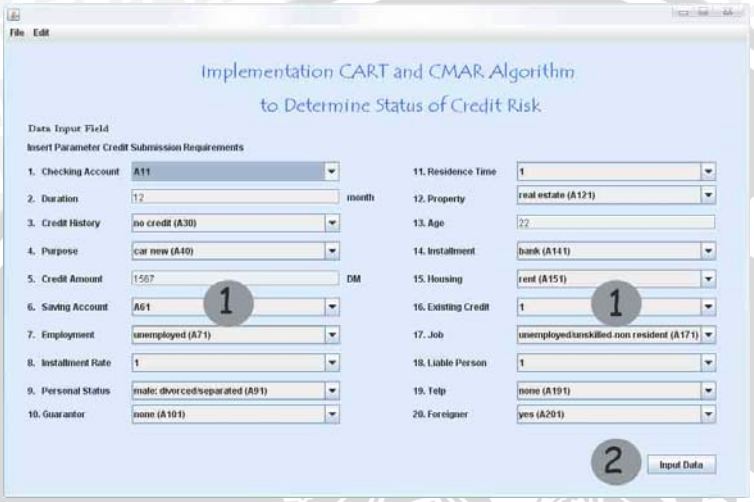
Gambar 3.18 *CR-tree* setelah *pruning*

3.4 Perancangan Antarmuka

Untuk perancangan antarmuka pada sistem ini, terdiri dari 4 *form*, yaitu *form input*, *form data training*, form algoritma CART dan *form* algoritma CMAR. *Form home* diilustrasikan pada Gambar 3.19, dan *form* algoritma CART dan CMAR akan diilustrasikan secara terurut pada Gambar 3.20 dan Gambar 3.21.

a. Form input data

Pada *form input data*, user mengisikan atribut persyaratan pengajuan kredit, kemudian memilih algoritma yang ingin dipakai untuk prediksi. Sementara itu, data *input*-an disimpan ke dalam *temporary array* dan sistem akan melatih data (*data training).*



Gambar 3.19 Antarmuka *input data*

Atribut yang diisikan pada persyaratan kredit (1) meliputi:

1. checking_account: besar tabungan yang dijadikan jaminan
2. duration: jangka waktu kredit
3. credit history: sejarah kredit yang pernah dimiliki
4. purpose: tujuan pengajuan kredit
5. credit amount: besar kredit yang diajukan
6. saving account: besar tabungan yang dimiliki
7. employment: lama bekerja
8. installment rate: besar presentase bunga kredit terhadap penghasilan

9. personal status: status pernikahan dan jenis kelamin
10. guarantor: penjamin
11. residence time: lama tinggal
12. property: asset yang dijadikan sebagai jaminan
13. age: usia nasabah
14. installment: rencanan angsuran
15. housing: status kepemilikan rumah
16. existing credit: jumlah kredit yang dimiliki
17. job: pekerjaan
18. liable person: jumlah orang yang tertanggung
19. telp: kepemilikan telepon
20. foreigner: status kewarganegaraan

Sedangkan pada nomor 2 digunakan untuk melanjutkan proses ke *data training.*

b. *Form data training*

Pada *form data training*, *user* dapat memilih *data training* sesuai dengan kehendak. *Form data training* ditunjukkan pada Gambar 3.20.

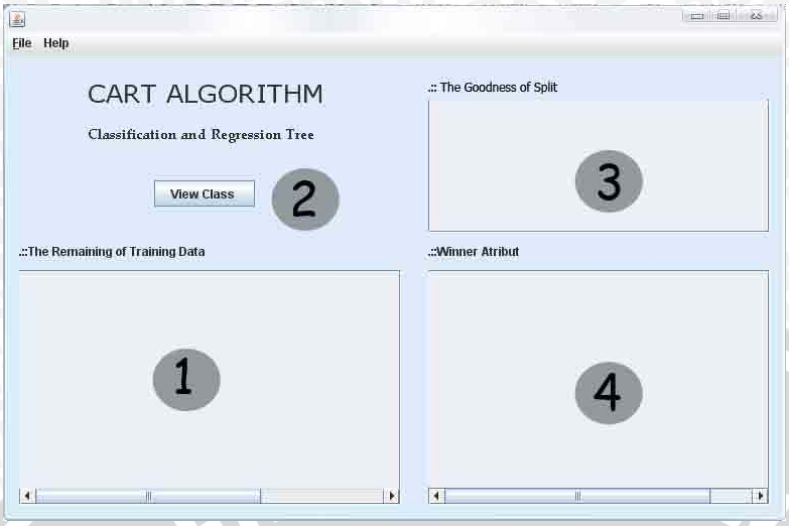
Gambar 3.20 Antarmuka *data training*

Keterangan Gambar :

1. Untuk memilih *data training*, yang terdiri dari 200, 300, 500, 600, dan 700 *record data training.*
2. Tombol untuk memilih, berlanjut ke proses CART atau CMAR.

c. *Form* algoritma CART

Pada *form* algoritma CART, akan ditampilkan *split* tertinggi dan *rule* yang paling cocok dengan data prediksi. *Form* algoritma CART ditunjukkan pada Gambar 3.21.



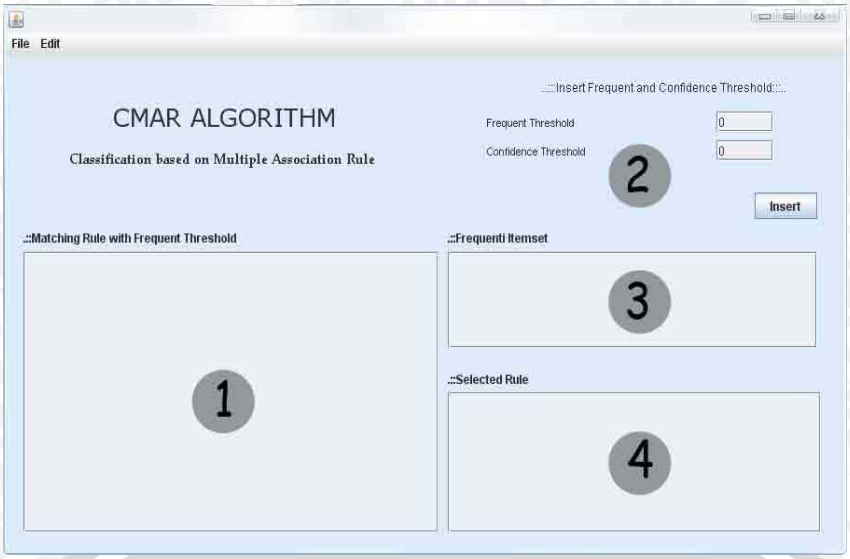
Gambar 3.21 Antarmuka algoritma CART

Keterangan Gambar :

1. *Rule* terakhir yang terpilih
2. Tombol untuk melihat kelas.
3. Menampilkan nilai *split*
4. Menampilkan atrbut pemenang.

***d. Form* algoritma CMAR**

Pada *form* algoritma CMAR, akan ditampilkan hasil data *training* berupa *rule* klasifikasi menggunakan algoritma CMAR yang ditunjukkan oleh nomor 1. Selain itu, akan ditampilkan pula *support threshold* dan *confidence threshold* yang telah dimasukkan oleh *user*, hasil prediksi, akurasi, *error rate*, dan waktu komputasi dari algoritma CMAR yang ditunjukkan oleh nomor 2. *Form* algoritma CMAR ditunjukkan pada Gambar 3.22.



Gambar 3.22 Antarmuka algoritma CMAR

Keterangan Gambar :

1. *Rule-rule* yang memenuhi syarat *pruning rule*
2. *Input* dari *user* berupa frekuensi dan *confidence threhold*
3. Frekuensi munculnya atribut.
4. Menampilkan *rule* yang terpilih setelah terjadi *pruning,*

3.5 Perancangan Uji Coba

Pada sub bab ini akan dijelaskan mengenai data yang digunakan dalam penelitian dan uji coba pada waktu prediksi.

3.5.1 Data yang digunakan dalam penelitian

Data yang digunakan dalam penelitian ini merupakan data pengajuan kredit pada suatu bank di Jerman. Pada data nasabah pada bank Jerman tersebut, terdapat 1000 *record* data.

Data yang digunakan dalam penelitian ini adalah data kategoris. Jadi, data Jerman akan dikategorikan dengan aturan pengelompokan sebagai berikut:

1. Status of existing checking account
   A11: < 0 DM
   A12: 0 – 200 DM
   A13: > 200 DM
   A14: no checking account
2. Duration in month
   A21: 0-12
   A22: 13-24
   A23: 25-36
   A24: 37-48
   A25: 49-60
3. Credit history
   A30: no credit
   A31: all credit at this bank paid back duly
   A32: existing credits paid back duly till now
   A33: delay in paying off in the past
   A34: critical account
4. Purpose
   A40: car new
   A41: car used
   A42: furniture/ equipment
   A43: radio/ TV
   A44: domestic appliances
   A45: repairs
   A46: education
   A47: vacation
   A48: retraining
   A49: business
   A419:others
5. Credit amount
   A51: 0-2500
   A52: 2501-5000
   A53: 5001-7500
   A54: 7501-10000
   A55: 10001-12500
   A56: 12501-15000
   A57: 15001-17500
6. Savings account/bonds
   A61: 0-100 DM
   A62: 101-500 DM
   A63: 501-1000 DM
   A64: > 1000 DM

7. Present employment since
   A71 : unemployed
   A72 : < 1 year
   A73 : 1 - 4 years
   A74 : 4,1 - 7 years
   A75 : > 7 years
8. Installment rate in percentage of disposable income
   A81 : 1
   A82 : 2
   A83 : 3
   A84 : 4
9. Personal status and sex
   A91 : male   : divorced/separated
   A92 : female : divorced/separated/married
   A93 : male   : single
   A94 : male   : married/widowed
   A95 : female : single
10. Other debtors / guarantors
   A101 : none
   A102 : co-applicant
   A103 : guarantor
11. Present residence since
   A111 : 1
   A112 : 2
   A113 : 3
   A114 : 4
12. Property
   A121: real estate
   A122: building society/ life insurance
   A123: car or other
   A124: no property/ unknown
13. Age
   A131: 19-34
   A132: 35-49
   A133: 50-64
   A134: 65-75
14. Other installment plans
   A141 : bank
   A142 : stores
   A143 : none
15. Housing
   A151 : rent

A152 : own
A153 : for free

16. Number of existing credits at this bank
    A161 :1
    A162 :2
    A163 :3
17. Job
    A171 : unemployed/ unskilled - non-resident
    A172 : unskilled - resident
    A173 : skilled employee / official
    A174 : management/ self-employed/highly qualified employee/ officer
18. Number of people being liable to provide maintenance for
    A181: 1
    A182: 2
19. Telephone
    A191 : none
    A192 : yes, registered under the customers name
20. Foreign worker
    A201 : yes
    A202 : no

3.5.2 Pengujian pada saat prediksi sistem

Data prediksi dibuat secara *random*, yang diambil dari 1000 *record* dari data bank Jerman. Untuk proses prediksi, misalkan diberikan data sebagai berikut:

a1: checking_account = a11	a11: residence time = 4
a2: duration = 42	a12: property = a122
a3: credit history = a32	a13: age = 45
a4: purpose = a42	a14: installment = a143
a5: credit_amount = 7882	a15: housing = a153
a6: saving account = a61	a16: existing credit = 1
a7: employment = a74	a17: job = a173
a8: installment rate = 2	a18: liable person = 2
a9: personal status = a93	a19: telp = a191
a10: guarantor = a103	a20: foreigner = a201

Dari data tersebut, akan diprediksi *class* untuk nasabah yang memiliki syarat seperti ketentuan tersebut, dan proses prediksi akan dilakukan menggunakan 2 macam algoritma, yaitu algoritma CART dan CMAR.

- Prediksi dengan menggunakan algoritma CART
  Untuk prediksi dengan algoritma CART, langkah pertama yang dilakukan oleh sistem adalah melihat *rule-rule* yang talah terbentuk oleh data *training.* Pada pengujian sistem prediksi ini, akan digunakan *rule* yang terbentuk berupa *tree* pada proses perancangan algoritma CART, yang diambil dari 10 *record* dari data *training. Tree* yang telah terbentuk dengan algoritma CART seperti ditunjukkan pada Gambar 3.15. Pada *tree* tersebut, *rule* pertama yang terbentuk adalah *residence time=2 vs residence time= {1,3,4}*. Karena pada data *testing residence time*-nya= 4, maka proses dilanjutkan ke *right cgild node*. Kedua, pengecekan atribut *credit history*. Pada *tree* tersebut, telah terbentuk *credit history= a33 vs credit history= {a32,a34}*. Pada data *testing, credit history*-nya = a32. Menurut tree pada Gambar 3.15, atribut dengan *credit history = a32*, memiliki *class* = 1, yaitu *good risk*. Jadi, dapat diputuskan bahwa nasabah dengan data tersebut layak mendapatkan kredit.
- Prediksi dengan menggunakan algoritma CMAR
  Pada prediksi dengan menggunakan algoritma CMAR, perlu dilihat hasil *rule* berupa *tree* yang ditunjukkan pada Gambar 3.18. Berdasarkan *tree* tersebut, pengecekan pertama yang dilihat adalah, apakah data *testing* memiliki atribut *a143* atau *a121*? Data *testing* memiliki atribut *a143*. Dilanjutkan dengan pengecekan, apakah data *testing* memiliki atribut *a201* atau *a191* atau keduanya? Data *testing* memiliki keduanya. Setelah itu dilakukan pengecekan, apakah *data testing* memiliki atribut *a93*? Data *testing* memiliki atribut *a93*. Menurut tree pada Gambar 3.18, atribut dengan *atribut a93*, memiliki *class* = 1, yaitu *good risk*. Jadi, dapat diputuskan bahwa nasabah dengan data tersebut layak mendapatkan kredit.

3.5.3 Skenario uji coba

Untuk pengujian, dilakukan perhitungan waktu kinerja dan akurasi. Akan digunakan variasi jumlah *data training* untuk pengujian tersebut. Variasi *data training* itu terdiri dari 200, 300, 500, 600 dan 700 *record.* Khusus untuk algoritma CMAR, diberikan variasi frekuensi dan *confidence threshold* untuk pengujian.

- *Data training*
  Semakin bervariasi *data training* yang digunakan untuk uji coba, diharapkan akan mendapatkan nilai akurasi yang semakin tinggi.
- Waktu kinerja sistem
  Dari pengujian tersebut akan dianalisa pengaruh waktu terhadap akurasi, dan dilihat apa yang terjadi pada akurasi, jika waktunya bertambah atau berkurang.
- Frekuensi dan *confidence threshold*
  Nilai *threshold* hanya berpengaruh pada algoritma CMAR saja. Jadi, untuk pengujian akurasi, akan dilihat dari nilai *threshold* yang diberikan dengan jumlah yang berbeda-beda pada setiap prediksi.

Rancangan tabel untuk analisa, akan dibuat 2 macam tabel. Tabel 3.32 digunakan untuk menampung akurasi dan waktu kinerja program pada algoritma CART terhadap jumlah *record data training.*

Tabel 3.32 Rancangan pengujian algoritma CART

200 *data train*			....			700 *data train*		
Rec	Waktu	Kelas	Rec	Waktu	Kelas	Rec	Waktu	Kelas
1			1			1		
...			...			...		
...			...			...		
...			...			...		
15			15			15		
Akurasi (%)			Akurasi (%)			Akurasi (%)		
Rt waktu			Rt waktu			Rt waktu		

Tabel 3.33 adalah tabel untuk menampung akurasi dan waktu kinerja program pada algoritma CMAR terhadap jumlah *record data training* dan kombinasi frekuensi , *confidence threshold.*

Tabel 3.33 Rancangan pengujian algoritma CMAR

∑ data training	(0,0)		....		(225,75)	
	Waktu	Akurasi	Waktu	Akurasi	Waktu	Akurasi
200						
300						
500						
600						
700						
Rt akur						
Rt wktu						

BAB IV
IMPLEMENTASI DAN PEMBAHASAN

4. 1 Lingkungan Implementasi

Lingkungan implementasi yang dijelaskan dalam Sub Bab ini adalah lingkungan implementasi perangkat keras dan perangkat lunak.

4.1.1 Lingkungan implementasi perangkat keras

Perangkat keras yang digunakan dalam implementasi algoritma CART dan CMAR ini adalah:

1. Prosesor Intel ® Core™ Duo
2. Memori 512 MB
3. *Harddisk* dengan kapasitas 120 GB
4. Monitor 14”
5. *Keyboard*
6. *Mouse*

4.1.2 Lingkungan implementasi perangkat lunak

Perangkat lunak yang digunakan dalam pengembangan sistem prediksi resiko kredit dengan menggunakan algoritma CART dan CMAR adalah :

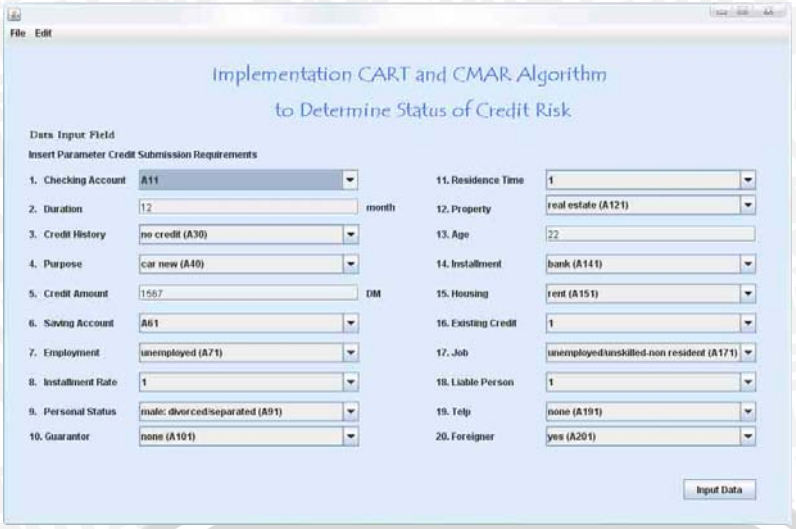
1. Sistem operasi yang digunakan adalah *Microsoft Windows XP Professional.*
2. Sistem dibuat menggunakan perangkat lunak *NetBeans* IDE 6.7.1 dan MySQL *database.*

4. 2 Implementasi Program

Pada Sub Bab implementasi program, dijelaskan mengenai implementasi dari rancangan perangkat lunak.

4. 2. 1 *Form input data*

Pada *form input data*, digunakan untuk memasukkan parameter persyaratan kredit nasabah. Pada *form* ini, terdapat 20 variabel masukan. Pada Gambar 4.1 ditunjukkan *form input data*.



Gambar 4.1 *Form input data*

Digunakan dua variabel dalam *form input data* pada `class input` yang ditampilkan pada Kode Program 4.1 dan diterangkan lebih lanjut pada Tabel 4.1.

```
1  public class input extends javax.swing.JFrame
2  {
3     public String p[]= new String[21];
4     private int n;
```

Kode Program 4.1 Variabel *form input data*

Tabel 4.1 Keterangan Variabel *form input data*

p	Suatu string yang digunakan untuk menyimpan data input / variabel data prediksi
n	Variabel untuk menampung nilai parameter persyaratan kredit

Pada *form input data*, terdapat 2 macam *swing controls* yang digunakan, yaitu *text field* dan *combo box*. Pada Kode Program 4.2 adalah contoh kode program yang menggunakan *text field*,

dan pada Kode Program 4.3 adalah *source code* dengan menggunakan *combo box* yang menggunakan variabel `p` dan `n`.

```
n=Integer.parseInt(a1.getText());
    if(n<0){
       p[1]="A11";
        }
    if (n>= 0 && n <= 200){
        p[1]="A12";
        }
    if (n > 200){
        p[1]="A13";
        }
    if (n==0){
        p[1]="no checking account";
        }
```

Kode Program 4.2 *Input data* menggunakan Text Field

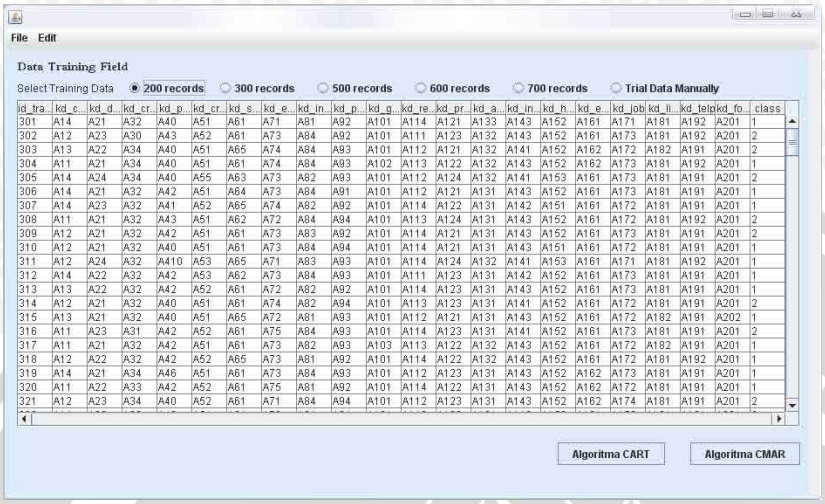
```
if(a3.getSelectedItem()=="no credit"){
     p[3]="A31";
}  else  if  (a3.getSelectedItem()==  "credit  at  this
   bank paid back dully"){
     p[3]="A32";
} else if (a3.getSelectedItem()== "delay in paying off
   in the past"){
     p[3]="A33";
}  else if (a3.getSelectedItem()=="critical
   account" ){
     p[3]="A34";
}
```

Kode Program 4.3 *Input data* menggunakan Combo Box

Pada Kode Program 4.2 dan 4.3, terdapat variabel `n` yang digunakan untuk menampung nilai dari salah satu parameter persyaratan kredit. Sedangkan variabel `p` digunakan untuk menampung atribut persyaratan kredit. Setelah parameter persyaratan kredit dimasukkan, data disimpan pada variabel sementara `p`. Setelah itu, proses berlanjut pada *data training*.

4. 2. 2 *Form data training*

Form data training digunakan *user* untuk memilih *data training* yang ingin digunakan untuk prediksi. Pada aplikasi ini, disiapkan 6 macam *data training*. Pemilihan 6 macam *data training* tersebut bervariasi berdasarkan jumlah *record*-nya, yaitu data dengan jumlah *record* 200, 300, 500, 600, 700 dan 10 *record* data yang digunakan untuk perhitungan manual. *Form data training* ditunjukkan pada Gambar 4.2.



Gambar 4.2. *Form data training*

Pada *form data training* ini, terdapat `class DataTraining` yang terdiri dari 5 variabel, yaitu, `training`, `prediksi`, `data`, `data2` dan `p`. Pada Kode Program 4.4 ditunjukkan variabel yang tersusun pada *data training*. Untuk deskripsi mengenai prosedur dan fungsi-fungsi pada `class DataTraining`, disajikan pada Tabel 4.2.

```
1  public class DataTraining extends javax.swing.JFrame {
2      public String training[][] = new String[501][23];
3      public String prediksi[]= new String[20];
4       String data = "Select * from t_transaction where
5       id_transaction between 501 and 1000";
6      public String data2 = "where id_transaction between 501
```

```
7       and 1000";
8
9     public DataTraining(String[] p) {
10          initComponents();
11          System.arraycopy(p, 1, prediksi, 0, p.length-  1);
12          tampilTabel();
13
```

Kode Program 4.4 Prosedur / fungsi `class DataTraining`

Tabel 4.2 Deskripsi prosedur / fungsi `class DataTraining`

`public class DataTraining extends javax.swing.JFrame`	Class *data training*
`public String training[][] = new String[501][23]`	Fungsi untuk mengambil *data training*.
`public String prediksi[]= new String[20];`	Fungsi untuk mengambil data prediksi.
`String data = "Select * from t_transaction where id_transaction between 501 and 1000";`	Fungsi untuk menampilkan *data training* yang dipilih *user* pada form *data training*.
`public String data2 = "where id_transaction between 501 and 1000";`	Variabel `data2` dikirim ke dalam proses prediksi CART dan CMAR
`System.arraycopy(p, 1, prediksi, 0, p.length-1)`	Digunakan untuk meng-copy nilai input-an prediksi.

Untuk memilih *data training*, digunakan *swing controls radio button*, yang ditunjukkan pada Kode Program 4.5.

```
1   private void aActionPerformed(java.awt.event. ActionEvent
2   evt) {
3       if(a.isSelected()){
4        data = "Select * from t_transaction where
5        id_transaction between 501 and 1000";
6        data2 = "where id_transaction between 501 and 1000";
7       tampilTabel();
8   }
```

Kode Program 4.5 *Swing controls radio button* untuk *data training*

Selanjutnya, jika *data training* sudah dipilih, *user* memilih metode yang ingin digunakan, yaitu metode CART atau CMAR.

Data yang dikirim ke metode CART dan CMAR adalah data prediksi (`prediksi`) dan *data training* (`data2`). Pada Kode Program 4.6 ditunjukkan proses pengiriman *data training* ke dalam proses CART dan CMAR.

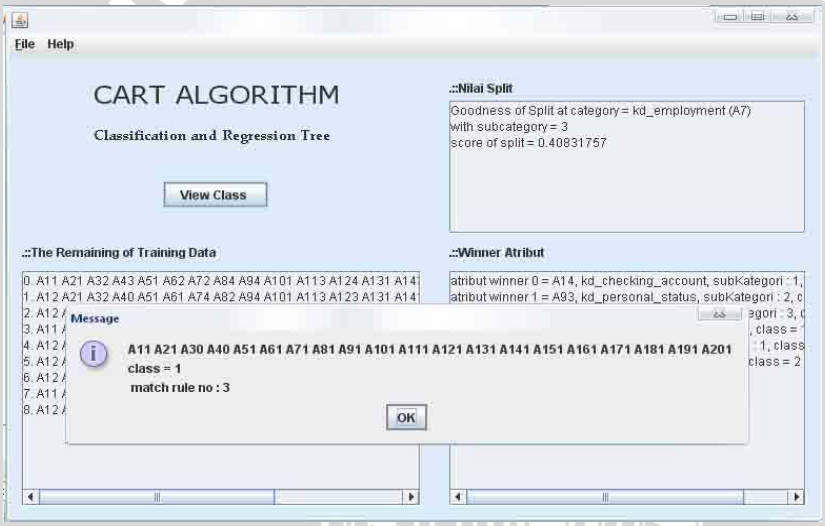
```
1   private void jButton1ActionPerformed
2   (java.awt.event.ActionEvent evt) {
3       cart fi= new cart(prediksi,data2);
4       fi.setVisible(true);this.setVisible(false);
5       }
6
7   private void jButton2ActionPerformed
8   (java.awt.event.ActionEvent evt) {
9       cmar fi= new cmar(prediksi,data2);
10      fi.setVisible(true);this.setVisible(false);
```

Kode Program 4.6 Proses pengiriman *data training* ke dalam proses CART dan CMAR

4. 2. 3 Algoritma CART

a. Form algoritma CART

Pada form algoritma CART, ditampilkan nilai *split* tertinggi dan *rule* yang terpilih. *Form* algoritma CART ditunjukkan pada Gambar 4.3.



Gambar 4.3 *Form* algoritma CART

b. Kelas pada algoritma CART.

Pada algoritma CART, terdapat 3 kelas, yaitu `class cart`, `class pruneRuleCART`, dan `classObjRuleWinCart`. Pada `class pruneRuleCART`, terdiri dari `class SubKategori` dan `class classObjectCART`. Penjelasan kelas-kelas pada algoritma CMAR dideskripsikan pada Tabel 4.3, 4.4 dan 4.5.

Tabel 4.3 Deskripsi prosedur/fungsi pada `class cart`

`private String a[],sqlstatment="";`	Fungsi untuk menyimpan data *training*
`private String[][] dataTraining;`	Fungsi mengambil data *training*
`private String[] dataInput;`	Fungsi untuk menyimpan data prediksi
`private pruneRuleCART totalProcessedRule;`	Fungsi untuk mengambil proses pembentukan *rule*

Tabel 4.4 Deskripsi prosedur/fungsi pada `classPruneRuleCART`

`private String[][] dataAwal, data;`	Untuk menyimpan rule yang dikirim dari konstruktor
`private String[] ketKategori`	Untuk menyimpan nama kategori dari konstruktor
`private ArrayList<String[]> listData;`	Modifikasi rule untuk penyederhanaan.
`private ArrayList<classObjectCART> preProcessedRule;`	`preProcessedRule` digunakan untuk menampung objek yang dibentuk dari kelas `classObjectCART`, yaitu objek `namaKategori`, `kategoriType` dan `subKategori`.
`private ArrayList<classObjRuleWinCart> winner;`	`winner` digunakan untuk menampung kategori pemenang
`public pruneRuleCART(String[][] bufInput, String[] ketKategori`	Memilih atribut-atribut yang masuk dalam kategori pemenang
`private ArrayList <classObjectCART>`	Digunakan untuk

`preProcessingData(ArrayList<String[]> listData)`	menginisialisaisi nama kategori ke sub kategorinya.
`private int[] processData(ArrayList<classObjectCART> processedRule, ArrayList<String[]> bufListData)`	Menghitung nilai *split.*
`private ArrayList<classObjectCART> processData(ArrayList<classObjectCART> processedRule, int totalBufRule)`	Pemilihan atribut yang mengandung *split* tertimggi
`private ArrayList<String[]> createInvRule(int highestKat, int highestSubKat, ArrayList<String[]> sourceRule, ArrayList<classObjectCART> processedRule)`	Mencari atribut dalam *rule* yang mengandung *split* tertinggi.
`public String defineClass(String[] input)`	Untuk mencocokkan data prediksi dengan *rule*.
`private String showRule(ArrayList<String[]> rul`	Memanggil *fix rule* yang sudah terbentuk.

Tabel 4.5 Deskripsi prosedur/fungsi pada `class classObjRuleWinCart`

`int kategori;`	Variabel untuk menyimpan kategori
`int subKategori;`	Variabel untuk menyimpan sub kategori
`String classType;`	Variabel untuk menyimpan kelas suatu *rule*
`String atributeName;`	Variabel untuk atribut nama.

c. Penentuan *candidate split* dan hitung nilai *split*

Setelah ditentukan *candidate split,* kemudian dihitung *split* berdasarkan training data yang dimiliki. Untuk menghitung split, yang pertama dilakukan adalah menghitung *pl*. *pl* adalah frekuensi munculnya atribut *x* yang muncul pada cabang kiri, di

bagi jumlah total data. Kemudian, yang dihitung selanjutnya adalah *pr*. *pr* merupakan frekuensi atribut *x* yang mucul pada cabang kanan, di bagi dengan jumlah total data.

Setelah itu, dihitung *p (j/tL) G*. *p (j/tL) G* merupakan jumlah atribut pL yang memiliki kelas *G*, di bagi jumlah atribut *pL* yang memiliki kelas *G* dan *B*. Dan dilanjutkan dengan menghitung *p (j/tL) B*, *p (j/tR) G*, dan *p (j/tR) B*.

Setelah ditemukan nilai *split*-nya, maka di cari *split* tertinggi. Dan proses terus berulang sampai ditemukan suatu kelas yang sama pada suatu *candidate split*.

Penentuan *candidate split* dan perhitungan *split* diimplementasikan pada Kode Program 4.7.

```
1     private int[] processData(ArrayList<classObjectCART>
2     processedRule,
3                 ArrayList<String[]> bufListData){
4       float highestSplit = 0;
5       int highestKat = 0;
6       int highestSubKat = 0;
7       String classType;
8       for(int i=0; i<processedRule.size(); i++){
9          for(int j=0; j<processedRule.get(i).subKategori.size();
10    j++){
11         processedRule.get(i).subKategori.get(j).pl =
12         (float)processedRule.get(i).subKategori.get(j).total
13         / (float)listData.size();
14         processedRule.get(i).subKategori.get(j).pr =
15         1 - (float)processedRule.get(i).subKategori.get(j).pl;
16         processedRule.get(i).subKategori.get(j).pjtl[0] =
17         (float)processedRule.get(i).subKategori.get(j).gbClass[0]
18         / (float)processedRule.get(i).subKategori.get(j).total;
19         processedRule.get(i).subKategori.get(j).pjtl[1] =
20        (float)processedRule.get(i).subKategori.get(j).gbClass[1]
21       / (float)processedRule.get(i).subKategori.get(j).total;
22       int totalRGood = 0, totalRBad = 0;
23          if(processedRule.get(i).subKategori.size() > 1){
24            for(int k=0; k<processedRule.get(i).subKategori.size();
25    k++){
26              if (k != j){
27                totalRGood +=
28    processedRule.get(i).subKategori.get(k).gbClass[0];
29                                totalRBad +=
30    processedRule.get(i).subKategori.get(k).gbClass[1];
31                            }
32                        }
33                    } else{
34                        totalRBad = 0;
35                        totalRGood = 0;
```

```
                }
   processedRule.get(i).subKategori.get(j).rGood = totalRGood;
   processedRule.get(i).subKategori.get(j).rBad = totalRBad;
   processedRule.get(i).subKategori.get(j).rTotal = totalRBad +
totalRGood;
      if (processedRule.get(i).subKategori.get(j).rTotal != 0){
          processedRule.get(i).subKategori.get(j).pjtr[0] =
          (float)totalRGood / (float)(totalRGood + totalRBad);
          processedRule.get(i).subKategori.get(j).pjtr[1] =
          (float)totalRBad / (float)(totalRGood + totalRBad);
          } else{
    processedRule.get(i).subKategori.get(j).pjtr[0]
                          =
processedRule.get(i).subKategori.get(j).pjtr[1]
                          = 0;
                }
   processedRule.get(i).subKategori.get(j)._2plpr =
(float)processedRule.get(i).subKategori.get(j).pl
                          *
(float)processedRule.get(i).subKategori.get(j).pr * 2;
                processedRule.get(i).subKategori.get(j).qst =

Math.abs(processedRule.get(i).subKategori.get(j).pjtl[0]
                          -
processedRule.get(i).subKategori.get(j).pjtr[0])
                          +
Math.abs(processedRule.get(i).subKategori.get(j).pjtl[1]
                          -
processedRule.get(i).subKategori.get(j).pjtr[1]);
                processedRule.get(i).subKategori.get(j).split =

(float)processedRule.get(i).subKategori.get(j)._2plpr
                          *
(float)processedRule.get(i).subKategori.get(j).qst;

    if (highestSplit <
processedRule.get(i).subKategori.get(j).split){
    highestSplit =
processedRule.get(i).subKategori.get(j).split;
    highestKat = i;
    highestSubKat = j;
                    if
(processedRule.get(i).subKategori.get(j).pjtl[0] >

processedRule.get(i).subKategori.get(j).pjtl[1])
    classType = "1";
       else
         classType = "2";
                }
           }
```

Kode Program 4.7 Pemilihan *candidate split* dan perhitungan *split*

Proses penentuan *candidate split*, perhitungan split, dan pemilihan nilai *split* tertinggi diimplementasikan dalam fungsi `processData`. Fungsi `processData` yaitu fungsi untuk mendapatkan *candidate* split dan menghitung *split*. Pada proses ini, iterasi pencarian *split* tertinggi dilakukan perulangan sampai ditemukan *puritiy class*.

d. Pemilihan *rule*

Pemilihan *rule* diimplementasikan dalam fungsi `createInvRule`. Pada fungsi ini, memiliki parameter data masukan berupa nilai *split* tertinggi (`highestKat`) dan atribut yang mengandung *split* tertinggi (`highestSubKat`). Fungsi `createInvRule` dibuat member variabel bernama `sourceRule` yang digunakan untuk menampung objek arrayList baru, yaitu `bufRule`. Implementasi program pemilihan *rule* dapat dilihat pada Kode Program 4.8.

```
1   private ArrayList<String[]> createInvRule(int highestKat,
2   int highestSubKat,
3     ArrayList<String[]> sourceRule,
4     ArrayList<classObjectCART> processedRule){
5     ArrayList<String[]> bufRule = new ArrayList<String[]>();
6     String subKat = processedRule.get(highestKat).kategoriType
7     + processedRule.get(highestKat).
8   subKategori.get(highestSubKat).subKategoriType;
9           for(int i=0; i<sourceRule.size(); i++){
10  if(!(sourceRule.get(i)[highestKat].equals(subKat))){
11          bufRule.add(sourceRule.get(i));
12              }
13          }
14          return bufRule;
15      }
```

Kode Program 4.8 Pemilihan *rule*

e. Proses prediksi

Pada proses prediksi, dilakukan pencocokan antara data prediksi dengan *fix rule* yang telah terbentuk. . Kelas *data testing* mengikuti kelas data *rule(x)* apabila jumlah atribut data *testing* banyak yang sama dengan atribut data *rule(x)*.

Implementasi proses predisi ditunjukkan pada Kode Program 4.9.

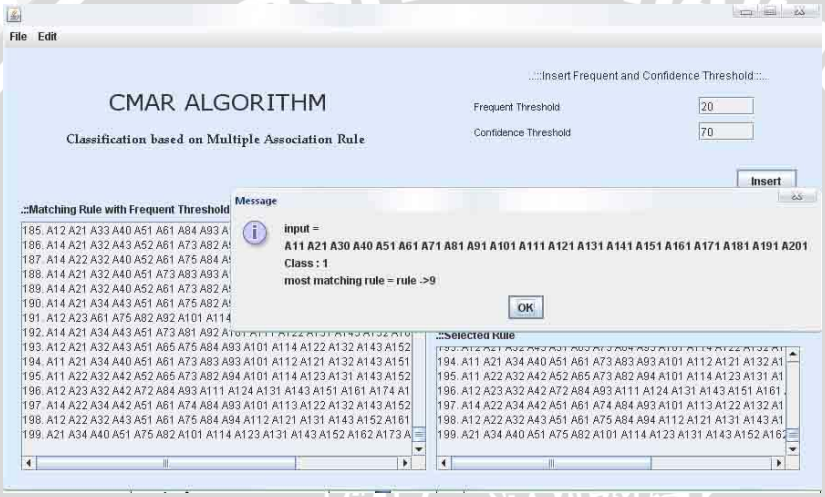
```
1     private boolean cekClassRule(ArrayList<String[]> rule){
2       boolean sameClass = true;
3       String cekClass = rule.get(0)[rule.get(0).length-1];
4       if(rule.size()>1){
5          for(int i=1; i<rule.size(); i++){
6          if(!(cekClass.equals(rule.get(i)[rule.get(i).length-
7     1]))){
8             sameClass = false;
9               break;
10                }
11             }
12          }
13          return sameClass;
14       }
```

Kode Program 4.9 Proses prediksi

4. 2. 4 Algoritma CMAR

a. Form algoritma CMAR

Pada algoritma CMAR, *user* memasukkan frekuensi dan *confdence threshold.* Gambar *form* algoritma *CMAR,* ditunjukkan pada Gambar 4.4.



Gambar 4.4 *Form* algoritma CMAR

b. Kelas pada algoritma CMAR

Pada algoritma CMAR, terdapat 3 kelas, yaitu `class cmar`, `class pruneRuleCMAR`, dan `classObjCMAR`. Penjelasan

kelas pada algoritma CMAR dideskripsikan pada Tabel 4.6, 4.7 dan 4.8.

Tabel 4.6 Deskripsi prosedur/fungsi pada `class cmar`

`public cmar(String p[],String data2)`	Untuk menyimpan data prediksi
`private ArrayList<List> dataSupport(Object[][] rawData){ class AttAndFrek`	Fungsi menghitung frekuensi masing-masing atribut

Tabel 4.7 Deskripsi prosedur/fungsi pada `pruneRuleCMAR`

`private double thresConf, thresSupp;`	Menyimpan batasan terendah *support* dan *confidence*
`private ArrayList<classObjectCMAR> processedRule;`	Untuk menyimpan urutan *rule*
`private ArrayList<classObjectCMAR> finalRule;`	Untuk menyimpan *rule* terakhir yang terbentuk
`private ArrayList<classRule> classType;`	Menyimpan kelas suatu atribut.
`private ArrayList<Integer> highestConf;`	Menyimpan nilai confidence paling tinggi.
`private ArrayList<Integer> deletedRule;`	Menghapus *rule* yang sudah tidak terpakai.

Tabel 4.8 Deskripsi prosedur/fungsi pada `classObjCMAR`

`List seqRule;`	
`ArrayList<Integer> ruleComplementAll;`	Digunakan untuk menampung komplemen *rule*
`ArrayList<Integer> ruleComplementClass;`	Digunakan untuk menampung komplemen *rule*, yang sekelas.

c. Hitung *frequent itemset*

Perhitungan *frequent itemset*, dalam program ditunjukkan pada Kode Program 4.10.

```
1       private ArrayList<List> dataSupport(Object[][] rawData){
2          class AttAndFrek{
3            String att;
```

```
4               int frek;
5           }
6       ArrayList<AttAndFrek> attAndFrek = new
7   ArrayList<AttAndFrek>();
8       ArrayList<List> bufData = new ArrayList<List>();
9       int thresSupp = Integer.valueOf(insupport.getText());
10         for(int i=0; i<rawData.length; i++){
11              for(int j=1; j<rawData[i].length-1; j++){
12                  if(attAndFrek.isEmpty()){
13  AttAndFrek buf = new AttAndFrek();
14                      buf.att = (String)rawData[i][j];
15                      buf.frek = 1;
16                      attAndFrek.add(buf);
17                  } else{
18                      boolean found = false;
19                      for(int k=0; k<attAndFrek.size(); k++){
20                          String strAtt =
21  attAndFrek.get(k).att;
22                          String strRaw =
23  (String)rawData[i][j];
24                          if(strAtt.equals(strRaw)){
25                              attAndFrek.get(k).frek++;
26                              found = true;
27                              break;
28                          }
29                      }
30                      if(!found){
31                          AttAndFrek buf = new AttAndFrek();
32                          buf.att = (String)rawData[i][j];
33                          buf.frek = 1;
34                          attAndFrek.add(buf);
35                      }
```

Kode Program 4.10 Hitung *frequent itemset*

Fungsi `AttAndFrek` digunakan untuk perhitungan frekuensi tiap-tiap atribut pada data *training*. Fungsi `AttAndFrek` memiliki parameter masukan berupa data *training* (`dataSupport`).

d. Fungsi untuk memilih *rule* yang sesuai dengan frekuensi *threshold*.

Fungsi untuk memilih *rule* yang sesuai dengan diimplementasikan pada Kode Program 4.11.

```
for(int i=0; i<rawData.length; i++){
            List buf = new List();
            for(int j=1; j<rawData[i].length-1; j++){
                String strRawData = (String)rawData[i][j];
                for(int k=0; k<attAndFrek.size(); k++){
                    if(attAndFrek.get(k).frek < thresSupp){
                        break;
                    } else{
                        String strAtt =
attAndFrek.get(k).att;
                        if(strRawData.equals(strAtt)){
                            buf.add(strRawData);}
                    }
                }
            }
            if(buf.getItemCount() > 0){
                buf.add((String)rawData[i][21]);
                bufData.add(buf);
          }
```

Kode Program 4.11 Pilih *rule* yang sesuai dengan *frekuensi threshold*

Rule yang memenuhi frekuensi *threshold* disimpan pada variabel `bufData`.

e. Penentuan *subset*/komplemen tiap-tiap *rule.*

Fungsi untuk penentuan *subset/*komplemen diimplementasikan pada Kode Program 4.12.

```
public void searchComp(){
        for(int i=0; i<processedRule.size(); i++){
            processedRule.get(i).ruleComplementAll = new
ArrayList<Integer>();
            processedRule.get(i).ruleComplementClass = new
ArrayList<Integer>();
            for(int j=0; j<processedRule.size(); j++){
                if(i==j){                   //rule-nya
berkomplemen dgn dirinya sendiri

processedRule.get(i).ruleComplementAll.add(j);

processedRule.get(i).ruleComplementClass.add(j);
                }
                else if(processedRule.get(i).seqRule.size()
<=

processedRule.get(j).seqRule.size()){
                    testComplement(i, j);
                }
```

Kode Program 4.12 Penentuan komplemen tiap-tiap *rule*

`ruleComplementAll` digunakan untuk menyimpan subset/komplemen untuk tiap-tiap *rule*. Sedangkan `ruleComplementClass` digunakan untuk menyimpan komplemen *rule* untuk kelas yang sama saja.

f. Hitung *support* dan *confidence*.

Perhitungan *support* dan *confidence* dihitung berdasarkan rumus 2.2 dan rumus 2.3. Perhitungan nilai *support* dan *confidence* ditunjukkan pada Kode Program 4.13.

```
1   processedRule.get(i).support =
2   (double)processedRule.get(i).ruleComplementClass.size()
3                       / (double)processedRule.size() * 100;
4               int indexClass =
5   processedRule.get(i).seqRule.size()-1;
6               for(int j=0; j<classType.size(); j++){
7                   if(classType.get(j).className.equals(
8
9   processedRule.get(i).seqRule.get(indexClass))){
10                      processedRule.get(i).conf =
11
12  (double)processedRule.get(i).ruleComplementClass.size()
13                                  /
14  (double)processedRule.get(i).ruleComplementAll.size() * 100;
```

Kode Program 4.13 Menghitung *threshold*

g. Memilih *rule* yang sesuai dengan *frequent* dan *confidence threshold*

Pemilihan *rule* yang sesuai dengan *frequent* dan *confidence threshold* ditunjukkan pada Kode Program 4.14.

```
1   for(int i=0; i<processedRule.size(); i++){
2               processedRule.get(i).confPass = true;
3               processedRule.get(i).suppPass = true;
4               for(int j=0;
5   j<processedRule.get(i).ruleComplementClass.size(); j++){
6                   int complement =
7   processedRule.get(i).ruleComplementClass.get(j);
8                       double confRule =
9   processedRule.get(i).conf;
10                      double confComp =
11  processedRule.get(complement).conf;
12                              double suppRule =
13  processedRule.get(i).support;
```

```
                                double suppComp =
processedRule.get(complement).support;
                    if(i != complement){
                        if(confRule < thresConf){
                            processedRule.get(i).confPass =
false;
                        } else{
                            if(confRule < confComp){
                                processedRule.get(i).confPass =
false;
                            }else if(confRule == confComp){
                                if(suppRule < suppComp){

processedRule.get(i).suppPass = false;
                                }else{
processedRule.get(i).suppPass = true;
```

Kode Program 4.14 Pilih *rule* yang sesuai dengan *frequent* dan *confidence threshold*.

h. *Pruning rule* CMAR

Pada *pruning* algoritma CMAR, dilakukan dengan cara membandingkan nilai *confidence* yang sekelas, dan dilanjutkan dengan membandingkan nilai *support*. Fungsi *pruning rule* diimplementasikan pada Kode Program 4.15.

```
public pruneRuleCMAR(){
   processedRule = new ArrayList<classObjectCMAR>();
    classType = new ArrayList<classRule>();
    }
public void addPruneRule(classObjectCMAR newRule){
   processedRule.add(newRule);
    int posClassIndex = newRule.seqRule.size() - 1;
     if(classType.isEmpty()){
            classRule bufClass = new classRule();
            bufClass.className =
(String)newRule.seqRule.get(posClassIndex);
            bufClass.frekuensi = 1;
            classType.add(bufClass);
}
        else{
            boolean find = false;
            for(int i=0; i<classType.size(); i++){
                if(classType.get(i).className.equals(
                    newRule.seqRule.get(posClassIndex))){
classType.get(i).frekuensi++;
                    find = true;
                }
            }
            if(!find){
```

```
25  classRule bufClass = new classRule();
26  bufClass.className =
27  (String)newRule.seqRule.get(posClassIndex);
28                  bufClass.frekuensi = 1;
29                  classType.add(bufClass);
```

Kode Program 4.15 Pruning *rule* CMAR

i. *Rule* terakhir

Setelah proses *pruning* selesai dilakukan, maka dicari *rule* terakhir yang digunakan untuk proses prediksi. Implementasi pogram ditunjukkan pada Kode Program 4.16

```
1   public String getFinalRule(){
2           String s = new String();
3           for(int i=0; i<finalRule.size(); i++){
4               s += i + ". ";
5               for(int j=0; j<finalRule.get(i).seqRule.size();
6   j++){
7                   s += finalRule.get(i).seqRule.get(j) + " ";
8               }
```

Kode Program 4.16 *Rule* terakhir

j. Proses prediksi

Pada fungsi ini, proses berjalan dengan menyamakan atribut data *testing/*prediksi dengan *fix rule* yang telah terentuk. Kelas data *testing* mengikuti kelas data *rule(x)* apabila jumlah atribut data *testing* banyak yang sama dengan atribut data *rule(x)*. Proses prediksi dalam program, ditunjukkan pada Kode Program 4.17.

```
1   for(int i=0; i<finalRule.size(); i++){
2               int lenPair = 0;
3               String s = new String();
4               for(int j=0;
5   j<finalRule.get(i).seqRule.size(); j++){
6                   s += finalRule.get(i).seqRule.get(j)
7   + " ";
8                   boolean foundPair = false;
9                   for(int k=0; k<input.length; k++){
10                      foundPair =
11  comparingObject(finalRule.get(i).seqRule.get(j),
12                                  input[k]);
13                      if(foundPair){
14
15
```

```
16                          break;
17                      }
18                  }
19                  if(foundPair){
20                      lenPair++;
21                  }else{
22                      break;
```

Kode Program 4.17 Proses prediksi

4. 3 Hasil Uji

Dalam pengujian program algoritma CART dan CMAR, digunakan 15 *record* data yang diambil dari data nasabah pengajuan kredit perbankan di Jerman, yang masing-masing datanya sudah memiliki *class* resiko kredit (*good* atau *bad*). Data pengujian CART dan CMAR ditunjukkan pada Lampiran 1.

Parameter pengujian untuk algorima CART adalah waktu kinerja pembentukan *rule* dalam program dan tingkat akurasi terhadap jumlah *record data training*. Sedangkan pada algoritma CMAR, parameter pengujiannya yaitu waktu kinerja pembentukan *rule* dalam program, tingkat akurasi, dan kombinasi frekuensi *threshold* dan *confidence threshold*

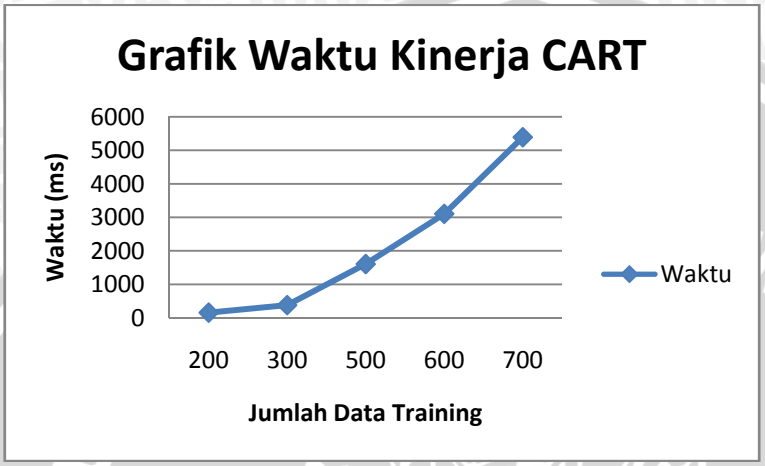
Pada algoritma CART, pengujian dilakukan dengan variasi jumlah *record data training* yang terdiri dari 200 *record*, 300 *record*, 500 *record*, 600 *record* dan 700 *record data training*. Hasil uji coba algoritma CART dengan parameter waktu kinerja program ditunjukkan pada Tabel 4.9.

Tabel 4.9 Perhitungan waktu CART

Rec	Waktu Kinerja (ms)				
	200	300	500	600	700
1	156	328	1563	3171	5406
2	156	375	1547	3062	5484
3	156	391	1672	3187	5453
4	172	437	1594	3050	5484
5	140	406	1578	3062	5250
6	156	422	1641	3188	5250
7	156	390	1625	3140	5407
8	141	406	1620	3171	5516
9	157	390	1656	3110	5422

10	156	406	1609	3063	5266
11	156	328	1624	3033	5375
12	156	406	1609	3187	5422
13	156	329	1594	3141	5437
14	156	391	1531	3078	5422
15	172	328	1625	2921	5269
Rata-rata	156.1	382.2	1605.8	3104.2	5390.8

Berdasarkan data hasil uji coba pada Tabel 4.9, pola kenaikan waktu kinerja terhadap jumlah record dapat dilihat pada Gambar 4.5.



Gambar 4.5 Grafik Waktu Kinerja Algoritma CART

Dari Gambar 4.5, didapatkan bahwa semakin banyak *data training* yang digunakan pengujian, maka semakin banyak waktu yang diperlukan program dalam pembentukan *rule*.

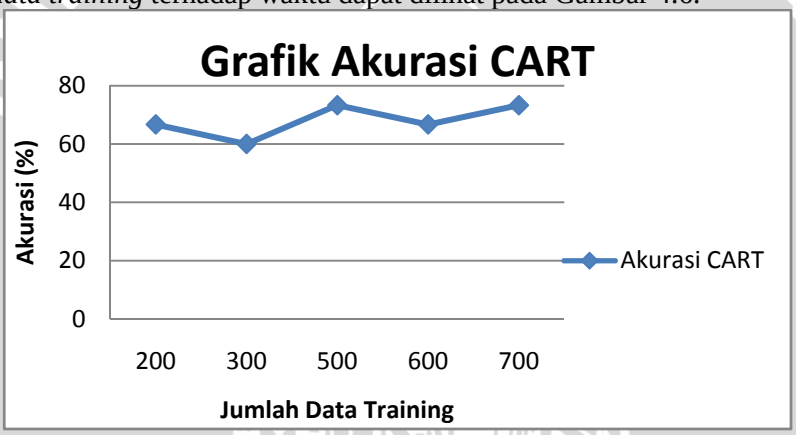
Perhitungan akurasi menggunakan algoritma CART ditunjukkan pada Tabel 4.10.

Tabel 4.10 Perhitungan akurasi CART

Rec	Kelas	Kelas Hasil Uji Coba				
		200	300	500	600	700
1	1	1	1	2	2	2
2	2	1	2	1	1	1
3	1	1	1	1	1	1

4	1	1	1	1	1	1
5	2	1	1	2	2	2
6	1	1	1	1	1	1
7	1	1	1	1	1	1
8	1	1	1	1	1	2
9	1	1	1	1	1	2
10	2	2	1	1	1	2
11	2	2	2	2	1	2
12	2	1	1	2	2	2
13	1	2	2	1	1	1
14	2	1	1	2	2	2
15	1	1	2	2	2	2
Jum Kelas Benar		10	14	11	10	11
Akurasi (%)		66.7	60	73.3	66.7	73.3

Dari data pada Tabel 4.10, angka 1 dan 2 merupakan kelas yang dihasilkan pada pengujian. Angka 1 adalah kelas dengan *good credit risk* dan angka 2 adalah kelas dengan *bad credit risk.* Jumlah kelas benar pada keterangan Tabel 4,10 merupakan jumlah kelas data pengujian yang bernilai sama dengan kelas data asli. Prosentase akurasi pada Tabel 4.10 dihitung berdasarkan Rumus 2.4. Berdasarkan data hasil uji coba pada Tabel 4.10, pengaruh jumlah *data training* terhadap waktu dapat dilihat pada Gambar 4.6.



Gambar 4.6 Grafik akurasi algoritma CART

Dari Gambar 4.6, didapat bahwa tingkat akurasi pada algoritma CART dari 5 macam variasi *data training* cenderung konstan. Akurasi pada algoritma CART tidak dipengaruhi dari jumlah *data training* yang digunakan dalam pengujian.

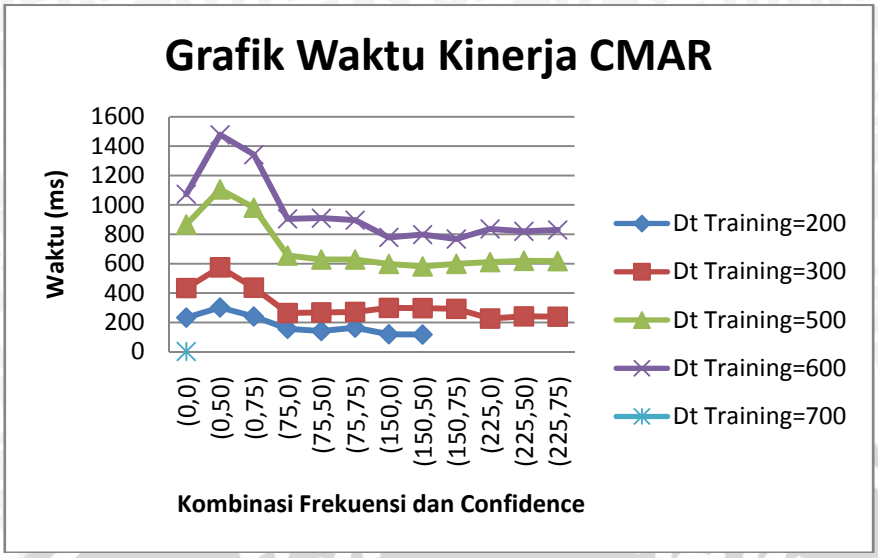
Pengujian pengaruh *data training* dan kombinasi frekuensi *confidence threshold* terhadap waktu kinerja untuk algoritma CMAR ditunjukkan pada Tabel 4.11.

Tabel 4.11 Perhitungan Waktu CMAR

D.T	R	Waktu											
		0 0	0 50	0 75	7 50	75 50	75 75	150 0	150 50	150 75	225 0	225 50	225 75
200	1	266	219	219	156	187	141	93	172				
	2	218	219	203	125	125	125	156	110				
	3	281	297	235	140	188	125	156	110				
	4	219	266	266	140	125	141	172	109				
	5	218	203	218	219	125	297	156	110				
	6	219	359	219	125	140	125	110	172				
	7	203	360	218	140	188	140	94	109				
	8	219	359	218	141	125	125	110	94				
	9	281	281	266	125	125	125	93	109				
	10	219	359	266	188	141	188	109	109				
	11	219	218	219	187	125	360	109	125				
	12	219	265	281	125	125	141	110	110				
	13	266	360	203	141	125	187	109	109				
	14	219	375	281	203	141	125	110	109				
	15	219	375	281	188	125	109	94	94				
Rata-rata		232.3	301	239.5	156.2	140.6	163.6	118.7	116.7				
300	1	390	562	485	250	250	250	281	282	344	188	204	235
	2	422	563	406	313	297	297	344	281	282	234	234	235
	3	407	578	406	310	290	280	289	280	281	210	230	230
	4	453	578	406	297	312	250	266	314	266	172	234	204
	5	469	547	407	250	297	297	344	281	266	203	266	266
	6	469	500	485	234	235	297	281	297	297	203	266	265
	7	406	515	406	250	250	250	282	328	281	281	281	203
	8	422	422	469	234	250	250	280	265	328	266	203	203
	9	453	672	390	266	234	250	344	281	281	250	266	281
	10	438	578	421	250	250	250	281	280	282	203	203	203
	11	453	969	468	250	235	266	265	265	266	250	265	266
	12	406	532	422	313	297	297	344	281	265	266	266	281
	13	390	562	500	250	250	235	281	281	312	203	203	203
	14	469	562	469	250	250	297	266	312	281	204	219	219
	15	454	484	406	235	313	297	328	422	344	250	266	281
Rata-rata		433.4	574.9	836.4	263.4	267.3	270.8	298.4	296.6	291.7	225.5	240.4	238.3
5	1	782	1140	969	625	625	609	625	609	609	609	609	609

0	2	828	1109	937	687	625	625	578	562	578	594	609	593
0	3	750	1125	969	703	688	640	578	562	649	656	578	594
	4	797	1109	1000	625	625	625	562	609	563	594	641	594
	5	1484	1093	937	672	610	609	579	563	563	578	656	641
	6	843	1109	954	609	609	609	579	641	641	610	656	657
	7	1656	1156	953	703	609	657	687	563	578	593	609	657
	8	765	1125	953	625	579	625	640	562	562	578	625	687
	9	750	1015	1016	625	687	687	630	547	625	594	656	593
	10	766	1141	1000	703	609	625	563	641	625	656	609	578
	11	781	1141	1031	625	615	625	562	578	578	640	594	500
	12	750	1141	953	625	672	610	562	563	563	594	594	594
	13	750	1109	1015	687	625	625	640	578	594	656	594	610
	14	781	969	1015	625	609	625	578	563	625	610	593	594
	15	542	1110	1016	688	641	625	609	578	625	594	672	750
Rata-rata		868.3	1106.1	981.2	655.1	628.5	628	598.1	581.2	598.5	610.4	619.6	616.7
	1	1078	1531	1375	907	906	855	859	828	750	875	844	828
	2	1063	1453	1297	953	922	937	750	766	750	813	781	812
	3	1063	1500	1312	937	954	860	765	812	782	844	844	859
	4	1125	1485	1297	875	906	978	812	828	781	859	875	844
	5	1062	1516	1297	875	900	890	750	750	781	859	797	797
	6	1062	1468	1312	953	859	859	828	813	750	859	859	875
6	7	1062	1469	1360	950	875	875	828	750	750	843	781	813
0	8	1063	1516	1375	875	890	860	766	843	766	813	797	812
0	9	1078	1500	1422	937	937	891	750	765	750	875	850	810
	10	1078	1500	1390	875	938	860	750	828	765	797	797	860
	11	1063	1469	1313	891	875	875	781	766	781	859	890	850
	12	1063	1469	1375	906	953	875	766	812	766	813	813	813
	13	1078	1469	1360	875	938	938	766	813	766	813	797	797
	14	1078	1313	1344	891	922	969	766	796	750	797	781	797
	15	1078	1515	1312	875	875	922	750	797	828	828	797	875
Rata-rata		1072.9	1478.2	1342.7	905	910	896.2	779.1	797.8	767.7	836.4	820.2	829.4
	1	1453	1550	1703	1234	1005	1141	1000	1078	1078	1062	1000	1094
	2	1468	1906	1828	1203	1100	1187	1016	1078	1078	1063	1625	1047
	3	1437	1891	1750	1141	1250	1219	1078	1031	1016	1000	1062	1063
	4	1469	1920	1703	1235	1156	1141	1031	1000	1062	1031	1062	1078
	5	1438	1950	1719	1219	1235	1219	1063	1094	1078	1078	1000	1031
	6	1468	1906	1790	1272	1219	1219	1063	1015	1063	1078	1094	1000
7	7	1469	1891	1703	1219	1140	1204	1000	1000	1016	1047	1000	1047
0	8	1437	1906	1730	1100	1156	1219	1078	1016	1078	1016	1015	1016
0	9	1454	1906	1750	1203	1156	1218	1063	1015	1000	1078	1047	1047
	10	1031	1875	1735	1140	1140	1187	1063	1010	1078	1000	1016	1047
	11	1469	1900	1797	1141	1141	1171	1000	1047	1015	1040	1000	1000
	12	1953	1890	1781	1157	1172	1156	1063	1000	1000	1000	1016	1063
	13	1469	1903	1719	1156	1141	1234	1000	1000	1062	1063	1000	1016
	14	1516	1922	1719	1204	1219	1203	984	1062	1078	1063	1078	1031
	15	1453	1875	1766	1172	1204	1141	1063	1000	1031	1031	1041	1032
Rata-rata		1465.6	1879.4	1746.2	1186.4	1162.2	1190.6	1037.6	1029.7	1048.8	1043.3	1070.4	1040.8

Perhitungan waktu untuk algoritma CMAR, jika digambarkan dalam bentuk grafik, dapat dilihat pada Gambar 4.7.



Gambar 4.7 Grafik waktu kinerja algoritma CMAR

Dari Gambar 4.7, didapatkan bahwa semakin banyak *data trainig* yang dipergunakan untuk pengujian, maka semakin besar waktu yang diperlukan program dalam pembentukan *rule*. Selain jumlah *data training*, faktor lain yang berpengaruh besar terhadap waktu kinerja program adalah frekuensi *threshold*. Semakin kecil nilai frekuensi yang dipergunakan dalam pengujian, maka semakin banyak waktu yang dipergunakan program dalam pembentukan *rule*. Begitu pula sebaliknya, semakin besar frekuensi *threshold*, maka semakin cepat waktu program dalam pembentukan *rule*. Hal ini terjadi karena semakin banyak atribut yang terseleksi dalam proses pembetukan *rule* jika frekuensi *threshold*-nya semakin kecil.

Dari lima kali pengujian dengan menggunakan 200, 300, 500, 600 dan 700 *data training*, waktu rata-rata yang dibutuhkan program dalam pembentukan *rule* paling tinggi adalah data pengujian dengan menggunakan frekuensi=0, dan *confidence*=50.

Pada Gambar 4.7, garis pada *data training*=200 terputus karena, frekuensi yang digunakan melebihi jumlah *data training* yang

digunakan. Data yang digunakan sebesar 200, namun frekuensi yang dimasukkan adalah sebesar 150 dan 225. Berarti, dalam 200 *data training* tersebut, tidak ditemukan atribut yang memiliki frekuensi yang lebih besar dari 150.

Hasil perhitungan akurasi algoritma CMAR, ditunjukkan pada Tabel 4.12.

Tabel 4.12 Perhitungan akurasi algoritma CMAR

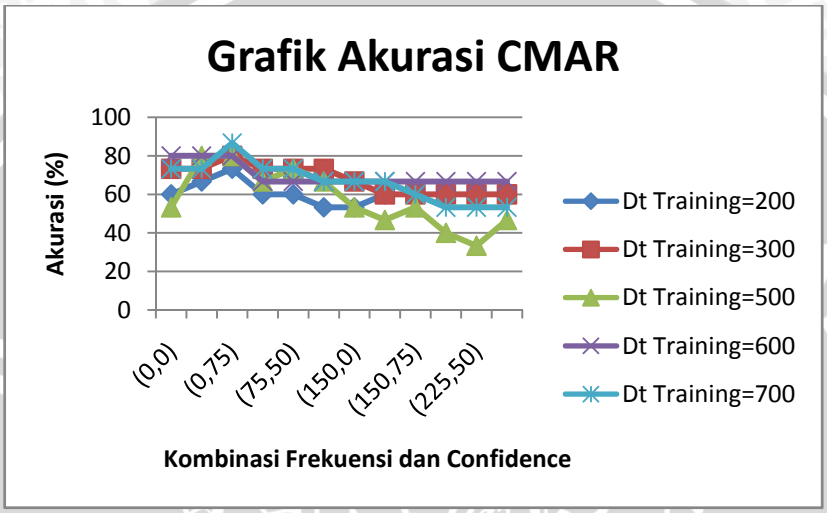
D.T	R	KI	Kelas Hasil Uji Coba											
			0 0	0 50	0 75	7 50	75 50	75 75	150 0	150 50	150 75	225 0	225 50	225 75
200	1	1	1	1	1	2	2	2	1	1				
	2	2	1	1	1	1	1	1	1	1				
	3	1	1	1	1	1	1	1	2	1				
	4	1	2	2	2	1	1	1	1	1				
	5	2	1	1	1	1	1	1	2	1				
	6	1	1	1	1	1	1	1	2	1				
	7	1	1	1	1	1	1	1	1	1				
	8	1	2	2	1	1	1	1	1	1				
	9	1	1	1	1	1	1	1	1	1				
	10	2	2	2	2	2	2	2	1	1				
	11	2	1	2	2	2	2	1	1	1				
	12	2	2	2	2	1	1	1	1	1				
	13	1	1	1	1	2	2	2	1	1				
	14	2	1	1	1	1	1	1	1	1				
	15	1	1	1	1	1	1	1	1	1				
Jum Kls Benar			9	10	11	9	9	8	8	9				
Akurasi (%)			60	66.7	73.3	60	60	53.3	53.3	60				
300	1	1	1	1	1	2	2	2	1	1	1	1	1	1
	2	2	2	2	2	2	2	2	2	1	1	1	1	1
	3	1	1	1	1	1	1	1	1	1	1	1	1	1
	4	1	1	1	1	1	1	1	2	1	1	1	1	1
	5	2	2	2	2	2	2	2	2	1	1	1	1	1
	6	1	1	1	1	1	1	1	1	1	1	1	1	1
	7	1	1	1	1	1	1	1	1	1	1	1	1	1
	8	1	2	2	1	2	2	2	1	1	1	1	1	1
	9	1	1	1	1	1	1	1	2	1	1	1	1	1
	10	2	2	2	2	2	2	2	2	1	1	1	1	1
	11	2	1	1	1	2	2	2	2	1	1	1	1	1
	12	2	1	1	1	2	2	2	2	1	1	1	1	1
	13	1	1	1	1	2	2	2	2	1	1	1	1	1
	14	2	1	1	1	1	1	1	1	1	1	1	1	1
	15	1	1	1	1	1	1	1	2	1	1	1	1	1
Jum Kls Benar			11	11	12	11	11	11	10	9	9	9	9	9
Akurasi (%)			73.3	73.3	80	73.3	73.3	73.3	66.7	60	60	60	60	60

500	1	1	1	1	1	1	1	1	1	1	1	2	2	1
	2	2	1	1	1	2	2	2	2	2	2	2	2	2
	3	1	2	1	1	1	1	1	1	1	1	2	2	1
	4	1	2	1	1	1	1	1	1	1	1	1	2	2
	5	2	2	2	2	1	1	1	2	2	2	1	1	1
	6	1	1	1	1	1	1	1	2	2	2	2	2	2
	7	1	1	1	1	1	1	1	2	2	2	1	1	1
	8	1	1	1	1	2	2	2	2	2	2	2	2	2
	9	1	2	1	1	1	1	1	1	1	1	2	2	2
	10	2	2	2	2	2	2	2	2	2	2	2	2	1
	11	2	2	2	2	1	1	1	1	1	1	2	2	2
	12	2	2	2	2	2	2	2	2	2	2	2	2	2
	13	1	2	1	1	1	1	1	2	2	2	2	2	2
	14	2	1	1	1	1	1	1	1	1	1	1	1	1
	15	1	2	2	2	2	2	2	2	2	2	2	2	1
Jum Kls Benar			8	12	12	10	11	10	8	7	8	6	5	7
Akurasi (%)			53.3	80	80	66.7	73.3	66.7	53.3	46.7	53.3	40	33.3	46.7
600	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	2	2	1	1	1	2	2	2	2	2	2	2	2	2
	3	1	1	1	1	2	2	2	1	1	1	1	1	1
	4	1	2	1	1	1	1	1	1	1	1	1	1	1
	5	2	2	2	2	1	1	1	2	2	2	2	2	2
	6	1	1	1	1	1	1	1	1	1	1	1	1	1
	7	1	1	1	1	1	1	1	1	1	1	1	1	1
	8	1	1	1	1	2	2	2	2	2	2	2	2	2
	9	1	1	1	1	1	1	1	1	1	1	1	1	1
	10	2	2	2	2	2	2	2	2	2	2	1	1	1
	11	2	2	2	2	1	1	1	1	1	1	2	2	2
	12	2	2	2	2	2	2	2	2	2	2	2	2	2
	13	1	1	1	1	1	1	1	2	2	2	2	2	2
	14	2	1	1	1	1	1	1	1	1	1	1	1	1
	15	1	1	1	1	1	1	1	2	2	2	2	2	2
Jum Kls Benar			12	12	12	10	10	10	10	10	10	10	10	10
Akurasi (%)			80	80	80	66.7	66.7	66.7	66.7	66.7	66.7	66.7	66.7	66.7
700	1	1	1	1	1	1	1	1	2	2	2	2	2	2
	2	2	1	1	1	2	2	2	2	2	2	2	2	2
	3	1	1	1	1	1	1	1	1	1	1	1	1	1
	4	1	2	1	1	1	1	1	1	1	1	1	1	1
	5	2	2	2	2	1	1	1	2	2	2	2	2	2
	6	1	1	1	1	1	1	1	1	1	1	2	2	2
	7	1	1	1	1	1	1	1	1	1	1	2	2	2
	8	1	1	1	1	2	2	2	2	2	2	2	2	2
	9	1	1	1	1	1	1	1	1	1	1	1	1	1
	10	2	2	2	2	2	2	2	2	2	2	2	2	2
	11	2	1	1	1	1	1	1	1	1	1	1	1	1
	12	2	2	2	2	2	2	2	2	2	2	2	2	2
	13	1	1	1	1	1	1	1	2	2	2	2	2	2

	14	2	1	1	2	1	1	1	1	1	1	1	1	1
	15	1	1	1	1	1	1	2	2	2	2	1	1	1
Jum Kls Benar			11	11	13	11	11	10	10	10	9	8	8	8
Akurasi (%)			73.3	73.3	86.7	73.3	73.3	66.7	66.7	66.7	60	53.3	53.3	53.3

Dari data pada Tabel 4.12, angka 1 dan 2 merupakan kelas yang dihasilkan pada pengujian. Angka 1 adalah kelas dengan *good credit risk* dan angka 2 adalah kelas dengan *bad credit risk.* Jumlah kelas benar pada keterangan Tabel 4,12 merupakan jumlah kelas data pengujian yang bernilai sama dengan kelas data asli. Prosentase akurasi pada Tabel 4.12 dihitung berdasarkan Rumus 2.4.

Perhitungan algoritma CMAR dengan kombinasi kombinasi frekuensi dan *confidence threshold*, jika digambarkan dalam grafik, dapat dilihat pada Gambar 4.8.

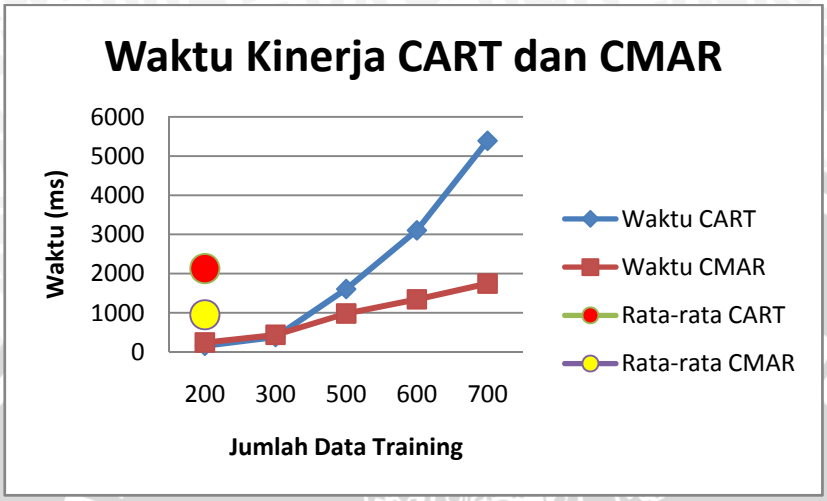


Gambar 4.8 Grafik akurasi algoritma CMAR

Berdasarkan Gambar 4.8, dapat dilihat bahwa jumlah *data training* tidak berpengaruh terhadap tingkat akurasi. Yang mempengaruhi tingkat akurasi adalah kombinasi frekuensi dan *confidence threhold.* Nilai akurasi tertinggi terdapat pada data yang menggunakan kombinasi nilai frekuensi=0 dan *confidence*=75. Rata-rata nilai akurasi rendah terdapat pada data yang menggunakan nilai frekuensi tinggi dan nilai *confidence* yang cenderung kecil. Dari pernyataan tersebut, memiliki arti lain bahwa semakin kecil nilai

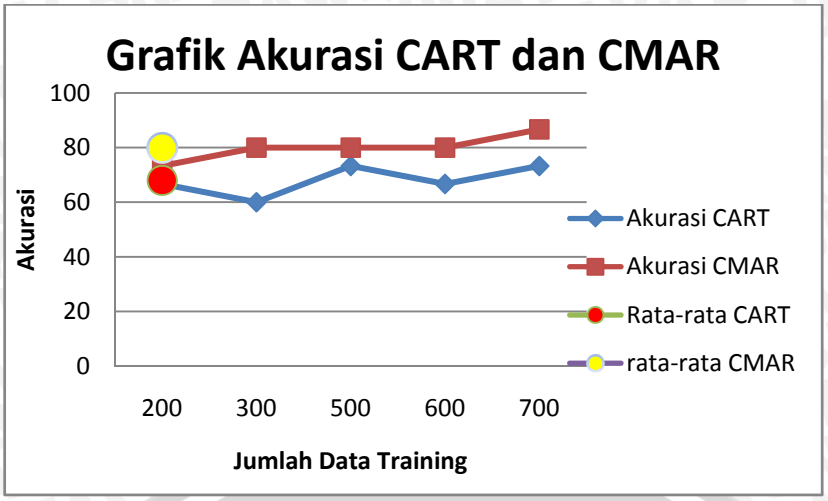
frekuensi dan semakin tinggi nilai *confidence*, menghasilkan tingkat akurasi yang semakin tinggi.

Untuk membandingkan waktu kinerja dan akurasi dari kedua algoritma tersebut, untuk CART dicari dengan nilai rata-rata pada pengujian tiap jumlah *record data training*. Sedangkan untuk CMAR, dipilih pengujian dengan kombinasi frekuensi dan *confidence threshold* yang menghasilkan akurasi tertinggi. Perbandingan waktu kinerja dan tingkat akurasi algoritma CART dan CMAR disajikan pada Gambar 4.9 dan Gambar 4.10.



Gambar 4.9 Grafik Waktu Kinerja CART dan CMAR

Dari Gambar 4.9, terlihat bahwa waktu kinerja algoritma CART cenderung mengalami kenaikan dibandingkan dengan waktu kinerja CMAR. Dari grafik, didapatkan rata-rata waktu kinerja algoritma CART jauh lebih tinggi dibandingkan waktu kinerja algoritma CMAR. Algoritma CART memiliki waktu rata-rata 2127.84 *ms* dan CMAR 949.2 *ms*.



Gambar 4.10 Grafik Akurasi CART dan CMAR

Dari grafik, didapatkan rata-rata akurasi kinerja algoritma CART jauh lebih rendah dibandingkan waktu kinerja algoritma CMAR. Algoritma CART memiliki akurasi sebesar 68% dan CMAR sebesar 80%.

4. 4 Analisa Hasil

Berdasarkan hasil uji coba algoritma CART, dapat diketahui bahwa waktu kinerja algoritma CART tergantung pada jumlah *data training* yang digunakan dalam pengujian. Semakin banyak *data training* yang digunakan untuk pengujian, maka semakin tinggi pula waktu yang dipergunakan program untuk pembentukan *rule*. Dari Gambar 4.5, grafik pengujian CART mengalami peningkatan waktu kinerja rata-rata sebesar dua kali besar waktu awal. Dari Gambar 4.5 tersebut, waktu dari 300 *record* sama dengan dua kali waktu dari 200 *record*. Karena pada pengujian tidak digunakan uji coba dengan 400 *record*, maka dari Gambar 4.5, terlihat kenaikan grafik dari 300 *record* ke 500 *record* sebesar tiga kali lipat. Hal ini terjadi karena, semakin banyak *record* yang digunakan untuk pengujian, maka semakin banyak pula *candidate split* yang digunakan dalam pembentukan *rule*. Tingkat akurasi algoritma CART tidak dipengaruhi oleh jumlah *data training* dan waktu kinerja program. Dari pengujian algoritma CART, didapatkan rata-rata waktu kinerja

program adalah 2127.84 *ms* dan tingkat akurasi rata-rata sebesar 68%.

Pada pengujian algoritma CMAR, didapatkan kesimpulan sama halnya seperti pengujian algoritma CART. Yaitu, semakin banyak *data training* yang digunakan untuk pengujian, maka semakin tinggi pula waktu yang dipergunakan program untuk pembentukan *rule*. Hal ini terjadi karena, semaikn banyak jumlah parameter yang digunakan dalam pengujian. Faktor lain yang mempengaruhi besarnya waktu kinerja adalah nilai frekuensi *threshold*. Semakin kecil nilai frekuensi yang dipergunakan dalam pengujian, maka semakin banyak waktu yang dipergunakan program dalam pembentukan *rule*. Begitu pula sebaliknya, semakin besar frekuensi *threshold*, maka semakin cepat waktu program dalam pembentukan *rule*. Pada frekuensi yang semakin besar, semakin sedikit *rule* yang terbentuk dalam pengujian. Hal inilah yang menyebabkan program tidak membutuhkan waktu tinggi dalam pengujian.

Akurasi pada algoritma CMAR tidak dipengaruhi oleh jumlah *record data training* yang digunakan dalam pengujian, melainkan dipengaruhi oleh kombinasi frekuensi dan *confidence threshold*. Semakin kecil nilai frekuensi dan semakin tinggi nilai *confidence*, maka dalam uji coba dihasilkan tingkat akurasi yang semakin tinggi, dan begitu pula sebaliknya. Jika frekuensi *threshold* yang digunakan semakin kecil, maka semakin bervariasi parameter pengujian yang digunakan dalam pengujian, hal ini yang menyebabkan pengujian semakin mendekati tingkat kebenaran. Semakin tinggi nilai *confidence threshold* yang digunakan, maka *rule* yang terpilih dalam proses prediksi adalah *rule subset* yang memiliki kelas sama. Tingkat akurasi terbesar, diperoleh dari data yang menggunakan frekuensi dan *confidence threshold* sebesar 0 dan 75%.

Dari pengujian algoritma CMAR dengan menggunakan frekuensi dan *confidence threshold* sebesar 0 dan 75% didapatkan rata-rata waktu kinerja program adalah 949.2 *ms* dan tingkat akurasi rata-rata sebesar 80%.

Berdasarkan pernyataan tersebut, dapat disimpulkan bahwa, algoritma CMAR memiliki waktu kinerja dalam pembentukan *rule* relatif lebih cepat dan tingkat akurasi lebih tinggi dibandingkan dengan algoritma CART.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Kesimpulan yang didapat dari skripsi ini adalah :

1. Algoritma CART dan CMAR memiliki cara kerja yang berbeda dalam penentuan status resiko kredit. Algoritma CART menentukan *candidate split* dan perhitungan *split*, sedangkan algoritma CMAR menentukan nilai *support* dan *confidence.*
2. Algoritma CMAR memiliki waktu kinerja program dalam pembentukan *rule* relatif lebih cepat dan tingkat akurasi lebih tinggi dibandingkan dengan algoritma CART. Pada algoritma CART, waktu kinerja program adalah 2127.84 *ms* dan tingkat akurasi rata-rata sebesar 68%. Sedangkan pada algoritma CMAR, waktu kinerja program adalah 949.2 *ms* dan tingkat akurasi rata-rata sebesar 80%. Hal tersebut menunjukkan bahwa algoritma CMAR cukup layak dipakai untuk prediksi status resiko kredit.
3. Tingkat akurasi pada algoritma CART dari 5 macam variasi *data training,* cenderung konstan. Akurasi dari algoritma CART tidak dipengaruhi dari jumlah *data training* yang digunakan dalam pengujian. Jumlah *data training* tidak berpengaruh terhadap tingkat akurasi algoritma CMAR. Yang mempengaruhi tingkat akurasi adalah kombinasi frekuensi dan *confidence threshold.* Semakin kecil nilai frekuensi dan semakin tinggi nilai *confidence,* maka akan menghasilkan tingkat akurasi yang semakin tinggi. Tingkat akurasi terbaik pada algoritma CMAR dihasilkan pada pengujian dengan menggunakan frekuensi dan *confidence threshold* sebesar 0 dan 75% dengan tingkat akurasi rata-rata sebesar 80%.

5.2. Saran

Beberapa saran untuk pengembangan lebih lanjut yang dapat diberikan oleh penulis adalah:

1. Pada penelitian ini, data yang digunakan adalah data nasabah pengajuan kredit perbankan di Jerman, dengan variabel-variabel pengujian standart Jerman. Oleh karena itu,

disarankan untuk penelitian selanjutnya yaitu dengan menggunakan variabel-variabel persyaratan kredit standart bank di Indonesia.

DAFTAR PUSTAKA

Chawla, Sanjay. 2007. *Classification Using Statistically Significant Rules*. University of Sydney. Sydney. http://www.slidefinder.net/c/classification_using_statistically_significant_rules/12910902/p2., tanggal akses: 20 Desember 2010.

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

Fayyad, Usama., dkk., 1996. *From Data Mining to Knowledge Discovery in Databases*. American Association for Artificial Intelligence.

Galindo, J., Tamayo, P., 2000. *Credit Risk Assessment Using Statistical and Machine Learning: Basic Methodology and Risk Modeling Applications*. Kluwer Academic Publisher. USA. http://portal.acm.org/citation.cfm?id=344512., tanggal akses: 20 Februari 2010.

Giri, Yudho, S., 2003. *Data mining – Menggali Informasi yang Terpendam*. http://ikc.cbn.net.id/populer/yudho/yudho-datamining.zip., tanggal akses: 17 Februari 2010.

Glassman, C.A., dan Wilkins, H.M., 1997. *Credit Scoring: Probabilities and Pitfalls*. Journal of Retail Banking Services 19 (2): 53-56.

Han, J., Pie, J., Yin, Y., 2000. *Mining Frequent Patterns without Candidate Generation*, School of Computing Science, Simon Fraser University.

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

Khusnawi. 2007. *Pengantar Solusi data Mining*. STMIK AMIKOM. Yogyakarta. http://p3m.amikom.ac.id/p3m/56%20-

%20PENGANTAR%20SOLUSI%20DATA%20MINING.pdf., tanggal akses: 18 Februari 2010.

Kristijadi, Emanuel. 2003. *Sistem Informasi Credit Scoring Berbasis Database Untuk Retail Banking*. STIE Perbanas. Surabaya. http://ekristijadi.files.wordpress.com/2007/10/sniktiviireff07037-emanuel-kristijadi.pdf., tanggal akses: 3 Februari 2010.

Laboratorium Sistem Informasi & Keputusan ITB. 2009. *Modul 5 Analisis dan Perancangan Sistem Informasi Manajemen 3: Pengenalan Sistem Pendukung Keputusan*. http://www.ti.itb.ac.id/~myti/files/Semester%206/PTI%202/Modul%205/modul%205.pdf., tanggal akses 1 April 2010.

Larose, Daniel T. 2005. *Discovering Knowledge in Data*: *An Introduction to Data Mining*. John Wiley & Sons, Inc.

Li, Wenmin., Han, Jiawei., Pei, Jian,. 2001. *Accurate and Efficient Classification Based on Multiple Class-Association Rules*. Simon Fraser University. Canada.

Santoso, Budi. 2007. *Data Mining Teknik Pemanfaatan Data untuk keperluan Bisnis*. Graha Ilmu. Yogyakarta.

Sutikno, T., Pujianta, A., Tri Y. S. 2007. *Prediksi Resiko Kredit Dengan Jaringan Syaraf Tiruan Backpropagation*. Universitas Ahmad Dahlan. Yogyakarta. http://journal.uii.ac.id/index.php/Snati/article/viewFile/1679/1461., tanggal akses: 27 November 2009.

Tang, Zhonghua. 2007. *A New Class Based Associative Classification Algorithm*. Santa Clara.

Thabtah, F., A., 2007. *A Greedy Classification Algorithm Based on Association Rule*. Bradford University. Amsterdam.

Thearling, Kurt,. 2000. *Data Mining and Customer Relationship*. http://www.thearling.com/index.htm)., tanggal akses: 21 Februari 2010.

Yusuf, Y. W. 2007. *Perbandingan Scoring Algoritma Decision Tree C5.0, CART, dan CHAID: Kasus Prediksi Status Resiko Kredit di Bank X*. Universtias Katolik Parahyangan. Yogyakarta. http://journal.uii.ac.id/index.php/Snati/article/view/1628/1403., tanggal akses: 27 November 2009.

Zaiane, Osmar R. 1999. *Principles of Knowledge Discovery in Databases*. University of Alberta. http://www.cs.ualberta.ca/~zaiane/courses/cmput690/slides/Chapter8., tanggal akses: 19 Februari 2010.

Lampiran 2. Data Hasil Pengujian CART

1. Menggunakan *data training* 200 *record*

Record ke-	Waktu Kinerja	Kelas
1	156	1
2	156	1
3	156	1
4	172	1
5	140	1
6	156	1
7	156	1
8	141	1
9	157	1
10	156	2
11	156	2
12	156	1
13	156	2
14	156	1
15	172	1
Akurasi (%)		66.7
Rata-rata waktu (ms)		156.1

2. Menggunakan *data training* 300 *record*

Record ke-	Waktu Kinerja	Kelas
1	328	1
2	328	2
3	375	1
4	391	1
5	437	1
6	406	1
7	422	1
8	390	1
9	406	1
10	390	1
11	406	2
12	328	1

13	406	2
14	329	1
15	391	2
Akurasi (%)		60
Rata-rata waktu (ms)		382.2

3. Menggunakan *data training* 500 *record*

Record ke-	Waktu Kinerja	Kelas
1	1563	2
2	1547	1
3	1672	1
4	1594	1
5	1578	2
6	1641	1
7	1625	1
8	1620	1
9	1656	1
10	1609	1
11	1624	2
12	1609	2
13	1594	1
14	1531	2
15	1625	2
Akurasi (%)		73.3
Rata-rata waktu (ms)		1605.8

4. Menggunakan *data training* 600 *record*

Record ke-	Waktu Kinerja	Kelas
1	3171	2
2	3062	1
3	3187	1
4	3050	1
5	3062	2
6	3188	1
7	3140	1
8	3171	1
9	3110	1

10	3063	1
11	3033	1
12	3187	2
13	3141	1
14	3078	2
15	2921	2
Akurasi (%)		66.7
Rata-rata waktu (ms)		3104.2

5. Menggunakan *data training* 700 *record*

Record ke-	Waktu Kinerja	Kelas
1	5406	2
2	5484	1
3	5453	1
4	5484	1
5	5250	2
6	5250	1
7	5407	1
8	5516	2
9	5422	2
10	5266	2
11	5375	2
12	5422	2
13	5437	1
14	5422	2
15	5269	2
Akurasi (%)		73.3
Rata-rata waktu (ms)		5390.8

Tabel 3.22 *Data training* uji coba manual

REC	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20	A21
1	A11	A21	A34	A43	A51	A65	A75	4	A93	A101	4	A121	A134	A143	A152	2	A173	1	A192	A201	1
2	A12	A24	A32	A43	A53	A61	A73	2	A92	A101	2	A121	A131	A143	A152	1	A173	1	A191	A201	2
3	A14	A21	A34	A46	A51	A61	A74	2	A93	A101	3	A121	A132	A143	A152	1	A172	2	A191	A201	1
4	A11	A24	A32	A42	A54	A61	A74	2	A93	A103	4	A122	A132	A143	A153	1	A173	2	A191	A201	1
5	A11	A22	A33	A40	A52	A61	A73	3	A93	A101	4	A124	A133	A143	A153	2	A173	2	A191	A201	2
6	A14	A23	A32	A46	A54	A65	A73	2	A93	A101	4	A124	A132	A143	A153	1	A172	2	A192	A201	1
7	A14	A22	A32	A42	A52	A63	A75	3	A93	A101	4	A122	A133	A143	A152	1	A173	1	A191	A201	1
8	A14	A21	A34	A42	A51	A65	A73	2	A94	A101	1	A123	A131	A143	A152	2	A173	1	A191	A202	1
9	A13	A22	A32	A43	A51	A61	A73	4	A93	A102	2	A121	A132	A142	A152	1	A173	1	A191	A201	2
10	A12	A23	A34	A40	A53	A61	A71	4	A94	A101	2	A123	A131	A143	A152	2	A174	1	A191	A201	2

Keterangan:
a1: checking_account
a2: duration
a3: credit history
a4: purpose
a5: credit_amount
a6: saving account
a7: employment
a8: installment rate
a9: personal status
a10: guarantor
a11: residence time
a12: property
a13: age
a14: installment
a15: housing
a16: existing credit
a17: job
a18: liable person
a19: telp
a20: foreigner
a21: class

Lampiran 1. Data Pengujian Algoritma CART dan CMAR

A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16	A17	A18	A19	A20	A21
A11	6	A34	A43	1169	A65	A75	4	A93	A101	4	A121	67	A143	A152	2	A173	1	A192	A201	1
A12	48	A32	A43	5951	A61	A73	2	A92	A101	2	A121	22	A143	A152	1	A173	1	A191	A201	2
A14	12	A34	A46	2096	A61	A74	2	A93	A101	3	A121	49	A143	A152	1	A172	2	A191	A201	1
A11	42	A32	A42	7882	A61	A74	2	A93	A103	4	A122	45	A143	A153	1	A173	2	A191	A201	1
A11	24	A33	A40	4870	A61	A73	3	A93	A101	4	A124	53	A143	A153	2	A173	2	A191	A201	2
A14	36	A32	A46	9055	A65	A73	2	A93	A101	4	A124	35	A143	A153	1	A172	2	A192	A201	1
A14	24	A32	A42	2835	A63	A75	3	A93	A101	4	A122	53	A143	A152	1	A173	1	A191	A201	1
A12	36	A32	A41	6948	A61	A73	2	A93	A101	2	A123	35	A143	A151	1	A174	1	A192	A201	1
A14	12	A32	A43	3059	A64	A74	2	A91	A101	4	A121	61	A143	A152	1	A172	1	A191	A201	1
A12	30	A34	A40	5234	A61	A71	4	A94	A101	2	A123	28	A143	A152	2	A174	1	A191	A201	2
A12	12	A32	A40	1295	A61	A72	3	A92	A101	1	A123	25	A143	A151	1	A173	1	A191	A201	2
A11	48	A32	A49	4308	A61	A72	3	A92	A101	4	A122	24	A143	A151	1	A173	1	A191	A201	2
A12	12	A32	A43	1567	A61	A73	1	A92	A101	1	A123	22	A143	A152	1	A173	1	A192	A201	1
A11	24	A34	A40	1199	A61	A75	4	A93	A101	4	A123	60	A143	A152	2	A172	1	A191	A201	2
A11	15	A32	A40	1403	A61	A73	2	A92	A101	4	A123	28	A143	A151	1	A173	1	A191	A201	1