IJCCS, Vol.8, No.2, July 2014, pp. 121~132
ISSN: 1978-1520

Aplikasi Algoritma CBA untuk Klasifikasi Resiko Pemberian Kredit

(Studi kasus: PT. Telkom CDC Sub Area Kupang)

Robynson Amseke*[1], Edi Winarko[2]
[1]UPT Komputer, Politeknik Pertanian Negeri Kupang, Kupang
[2]Jurusan Ilmu Komputer dan Elektronika, FMIPA UGM, Yogyakarta
email: *[1]**robinsonwillson77@gmail.com,** [2]ewinarko@ugm.ac.id

Abstrak

Salah satu penyebab kredit bermasalahberasal dari pihak internal, yaitu kurang telitinya timdalam melakukan survei dan analisis, atau bisa juga karena penilaian dan analisis yang bersifat subjektif.Penyebab ini dapat diatasi dengan sistem komputer, yaitu aplikasi komputer yang menggunakan teknik data mining.Teknik data mining digunakan dalam penelitian ini untuk klasifikasi resiko pemberian kredit dengan menerapkan algoritma Classification Based On Association (CBA). Algoritma ini merupakan salah satu algoritma klasifikasi dalam data mining yang mengintegrasikan teknik asosiasi dan klasifikasi. Data kredit awal yang telah di-preprocessing, diproses menggunakan algoritma CBA untuk membangun model, lalu model tersebut digunakan untuk mengklasifikasi data pelaku usaha baru yang mengajukan kredit ke dalam kelas lancar atau macet.Teknik Pengujian akurasi model diukur menggunakan 10-fold cross validation. Hasil pengujian menunjukkan bahwa rata-rata nilai akurasi menggunakan algoritma CBA (57,86%), sedikit lebih tinggi dibandingkan rata-rata nilai akurasi menggunakan algoritma Naive Bayes dan SVM dari perangkat lunak Rapid Miner 5.3 (56,35% dan 55,03%).

***Kata kunci**—classification based on association, CBA, data mining, klasifikasi, resiko pemberian kredit*

Abstract

One of the causes of non-performing loans come from the internal, that is caused by a lack of rigorous team in conducting the survey and analysis, or it could be due to subjective evaluation and analysis. The cause of this can be solved by a computer system, the computer application that uses data mining techniques. Data mining technique, was usedin this study toclassifycreditriskby applyingalgorithmsClassificationBasedonAssociation(CBA). This algorithm is an algorithm classification of data mining which integratingassociationandclassificationtechniques. Preprocessed initial-credit data, will be processed using theCBAalgorithmto create a model of which is toclassifythe newloandata into swift class or bad one. Testing techniques the accuracy of the model was measured by 10-fold cross validation. The resultshowsthatthe accuracy averagevalue using theCBAalgorithm(57,86%), was slightly higher than those using thealgorithmsofSVM andNaiveBayes from RapidMiner5.3software(56,35% and55,03%, respectively).

***Keywords**—classification based on association, CBA, data mining, classification, credit risk*

Received August 27th,2013; Revised April 17th, 2014; Accepted July 10th, 2014

1. PENDAHULUAN

Salah satu tanggung jawab sosial PT. Telkom dalam program kemitraan dengan masyarakat sekitarnya adalah memberikan kredit lunak bagi pelaku Usaha Kecil Menengah (UKM). Kredit ini bisa diperoleh melalui tahapan pengajuan proposal, survei ke lokasi usaha, wawancara, dan analisis terhadap kelayakan usaha. Semua tahapan ini sebagai proses seleksi untuk meminimalkan terjadinya kredit bermasalah, karena adanya kredit bermasalah akan mengganggu kondisi keuangan PT. Telkom dan menyebabkan berkurangnya jumlah dana yang disalurkan kepada para pelaku UKM periode berikutnya. Sistem seleksi pemberian kredit yang ada saat ini belum efektif menurunkan angka kredit bermasalah, hal ini terlihat pada data kredit PT. Telkom CDC (*Community Development Centre*) Sub Area Kupang dari tahun 2003 s/d triwulan II tahun 2010, dari 1.054 Mitra Binaan (MB) yang kreditnya disetujui, 46,8% (493 MB) berstatus bermasalah, dan 53,2% (561 MB) berstatus lancar. Pengertian bermasalah adalah MB yang setelah tanggal jatuh tempo pelunasan kredit (per triwulan II tahun 2012), belum melunasi kreditnya.

Kurang efektifnya sistem penerimaan saat ini, maka diperlukan sistem lain untuk seleksi pemberian kredit. Sistem lain yang digunakan adalah sistem komputer yang mampu mengklasifikasi resiko pemberian kredit terhadap pelaku usaha baru yang mengajukan kredit.Sistem komputer yang dimaksud adalah sebuah aplikasi komputer yang menggunakan teknik *data mining*. Teknik inidigunakan karena mampu mengekstrak *knowledge* dari data-data pelaku usaha sebelumnya [1]. *Knowledge* ini selanjutnyadigunakan untuk menyelesaikan permasalahan. Data kredit pelaku usaha sebelumnya diproses menggunakan sebuah algoritma untuk membangun model, lalu model ini digunakan untuk mengklasifikasi kemungkinan kredit bermasalah terhadap pelaku usaha baru yang mengajukan kredit.

Metode yang digunakan dalam *data mining* untuk klasifikasi disebut metode klasifikasi atau *Supervised Learning*. Beberapa algoritma dari metode ini yang telah digunakan untuk mengklasifikasi resiko pemberian kredit antara lain *Naive Bayes* [2], dan *Support Vector Machnine* (SVM) [3,4,5]. Terdapat algoritma lain untuk klasifikasi, salah satu diantaranya adalah *Classification Based On Assosiations* (CBA) [6]. Algoritma ini memiliki 2 kelebihan, yaitu (1) memiliki akurasi yang lebih baik dibandingkan beberapa algoritma klasifikasi lainnya (C4.5, RIPPER, NB, PART, SVM, REG, dan HIVdb) [7,8,9,10], dan (2) menghasilkan sekumpulan *rule* yang mudah dipahami dan diinterpretasikan oleh pengguna, dan mampu menggunakan atribut literal untuk membangun model [10].

Penelitian ini menggunakan algoritma CBA, sebuah pendekatan dari teknik *data mining* untuk mengklasifikasi resiko pemberian kredit terhadap pelaku usaha baru yang mengajukan kredit di PT. Telkom CDC Sub Area Kupang. Hasil pengujian pengaplikasian algoritma CBA pada aplikasi yang dibangun, akan dibandingkan akurasinya dengan algoritma *Naive Bayes* dan SVM dari perangkat lunak Rapid Miner 5.3. Perbandingan akurasi ini untuk mengetahui performa algoritma CBA jika dibandingkan dengan algoritma *Naive Bayes* dan SVM untuk klasifikasi resiko pemberian kredit di PT. Telkom CDC Sub Area Kupang.

2. METODE PENELITIAN

2.1 Pengumpulan Data

Data kredit awal yang dikumpulkan adalah data kredit UKM PT.Telkom CDC Sub Area Kupang dari tahun 2003 s/d bulan Juli tahun 2010 yang bersumber dari SIM PKBL PT. Telkom. SIM mengeksport data calon Mitra Binaan (MB) yang mengajukan proposal dan data MB yang proposalnya diterima ke file yang berformat .xls atau .xlsx (file Excel). Kedua file ini kemudian digabung, lalu dipilih atribut-atribut yang sesuai dengan kebutuhan penelitian. Hasilnya, diperoleh 12 atribut dengan jumlah data sebanyak 1.054 record. Contoh data kredit awal yang dikumpulkan, ditunjukkan pada Tabel 1.

Tabel 1 Contoh data kredit awal UKM PT.Telkom CDC Sub Area Kupang

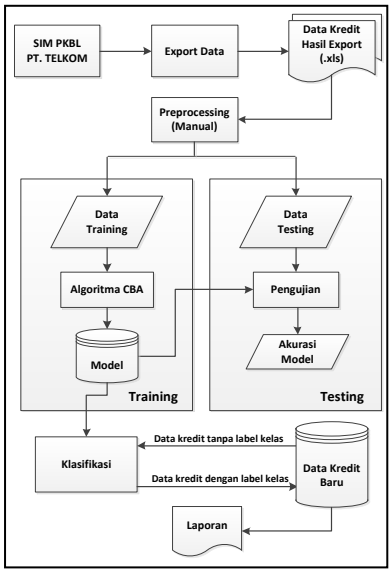
THN	TRIW	STATUS RUMAH	AGUNAN	PINJAMAN	SEKTOR USAHA	JENIS USAHA	BENTUK USAHA	KABUPATEN/K	KECAMATAN	KELURAHAN	KELAS
2004	IV	Milik Sendiri	SERTIFIKAT	KECIL	PERDAGANGAN	PERDAGANGAN	Perorangan Tidak Berbai	Ngada	AESESA	DANGA	LANCAR
2004	IV	Milik Sendiri	-	KECIL	JASA	JASA	Perorangan Tidak Berbai	Kota Kupang	-	-	DIRAGUKAN
2004	IV	Milik Sendiri	SERTIFIKAT	KECIL	JASA	SALON	Perorangan Tidak Berbai	Ngada	NGADA BAWA	LEBIJAGA	DIRAGUKAN
2004	IV	Milik Sendiri	SERTIFIKAT	KECIL	PERDAGANGAN	PERDAGANGN	Perorangan Tidak Berbai	Ngada	AESESA	DANGA	DIRAGUKAN
2004	IV	Milik Sendiri	SERTIFIKAT	KECIL	PERIKANAN	NELAYAN	Perorangan Tidak Berbai	Ngada	RIUNG	NANGAMESE	DIRAGUKAN
2004	IV	Milik Sendiri	SERTIFIKAT	KECIL	PERDAGANGAN	PERDAGANGAN	Perorangan Tidak Berbai	Ngada	AESESA	LAPE	DIRAGUKAN
2004	IV	Milik Sendiri	-	KECIL	JASA	JASA	Perorangan Tidak Berbai	Kota Kupang	-	-	DIRAGUKAN
2004	IV	Milik Sendiri	BPKB	KECIL	INDUSTRI	MEBEL	Perorangan Tidak Berbai	Ngada	AESESA	WAEKOKOK	MACET

2.2 Analisis Sistem

Sistem yang dirancang dan dibangun pada penelitian ini merupakan sistem yang dapat menggunakan file data kredit awal (file excel) untuk membangun model, yang kemudian digunakan untuk mengklasifikasi data kredit baru yang label kelasnya belum diketahui ke dalam kelas lancar atau macet. Data file kredit awal dibaca oleh sistem untuk disimpan ke dalam database, lalu data tersebut diproses menggunakan teknik *k-fold cross validataion* untuk menghasilkan data *training* dan *testing*. Data *training* diproses menggunakan algoritma CBA untuk menghasilkan model, lalu model ini diuji menggunakan data *testing* untuk memperoleh nilai akurasi model. Model yang dihasilkan ini, lalu digunakan untuk mengklasifikasi data kredit baru (data kredit tanpa label kelas) ke dalam kelas lancar atau macet.Data kredit yang sudah diklasifikasi, selanjutnya oleh sistem dibuatkan laporan ke Manajer.

2.3 Garis Besar Sistem

Sistem yang dirancang bertujuan membuat model yang akan digunakan untuk mengklasifikasi pelaku usaha baru yang mengajukan kredit ke dalam kelas lancar atau macet. Gambaran umum alur kerja sistem penelitian ini ditunjukkan pada Gambar 1.



Gambar 1 Garis besar sistem

Tampak pada Gambar 1 bahwa yang pertama kali dilakukan adalah mengumpulkan data yang bersumber dari SIM PKBL PT. Telkom.SIM mengeksport data ke MSExcel berupa data calon mitra binaan yang mengajukan proposal dan data mitra binaan yang proposalnya disetujui dari tahun 2003 s/d triwulan II tahun 2010.Hasil ekspor kedua data ini, lalu digabungkan untuk memperoleh sebuah data kredit awal. Data kredit awal yang telah diperoleh, lalu di *preprocessing* (secara manual) agar data siap digunakan untuk proses selanjutnya, yaitu digunakan oleh sistem untuk membangun model. Pertama-tama data kredit awal dibagimenjadi 2 bagian, yaitu data *training* dan data *testing* menggunakan teknik *k-fold cross validation*, lalu data *training* diproses menggunakan algoritma CBA untuk menghasilkan model, dan model ini selanjutnya diuji menggunakan data *testing* untuk menghasilkan nilai akurasi model. Data kredit

baru tanpa label kelas yang tersimpan di tabel kreditbaru lalu diklasifikasi menggunakan model yang telah dihasilkan. Hasilnya, diperoleh label kelas bagi data kreditbaru tanpa label kelas tersebut untuk disimpan ke tabel kreditbaru, yang selanjutnya oleh sistem dibuatkan laporan kepada Manajer.

2.4 Preprocessing

Data kredit awal yang telah dikumpulkan, selanjutnya dilakukan proses awal (*preprocessing*) untuk mengurangi berbagai masalah dari data sehingga data tersebut siap digunakan untuk proses *datamining*. Kegiatan *preprocessing* yang dilakukan sebanyak 4 kegiatan, yaitu (1) pemilihan data, (2) pembersihan data, (3) transformasi data, dan(4) mengidentifikasikesalahan klasifikasi. Semua kegiatan ini dilakukan secara manual.

2.4.1 Pemilihan Data

Atribut-atribut yang telah diperoleh pada Tabel 1, lalu dipilih atribut untuk digunakan dalam proses *datamining*. Hasilnya, terpilih 9 atribut dari 12 atribut yang ada, yaitu (1) STATUS RUMAH, (2) AGUNAN, (3) PINJAMAN, (4) SEKTOR USAHA, (5) JENIS USAHA, (6) BENTUK USAHA, (7) KECAMATAN, (8) KELURAHAN, dan (9) KELAS.Atribut yang tidak dipilih yaitu TAHUN, TRIWULAN, dan KABUPATEN/KOTA. Atribut TAHUN dan TRIWULAN, tidak dipilih karena kedua atribut ini hanya menginformasikan tentang waktu awal peminjaman, sedangkan atribut KABUPATEN/KOTA, tidak dipilih karena atribut ini mempunyai jangkauan wilayah yang luas sehingga *rule* yang dihasilkan terhadap wilayah usaha menjadi kurang spesifik, oleh karena itu atribut yang wilayah usahanya lebih spesifik yang digunakan, yaitu kecamatan dan kelurahan.

2.4.2 Pembersihan Data

Pembersihan data dilakukan dengan cara mengisi nilai data yang hilang dan memperbaiki nilai data yang tidak konsisten. Menurut [1,11], pengisian terhadap nilai data yang hilang dapat menggunakan salah satu dari 6 cara berikut: (1) mengisi nilai secara manual, atau (2) menggunakan nilai konstanta tertentu, atau (3) menggunakan nilai *mean* atau rata-rata untuk variabel numerik, atau (4) menggunakan *mode* atau nilai yang paling sering muncul untuk variabel kategori, atau (5) mengisi dengan nilai random berdasarkan nilai-nilai variabel yang ada, atau (6) mengisi data secara proporsional berdasarkan nilai-nilai variabel yang ada.

Perbaikan data yang tidak konsisten dilakukan pada atribut KELAS, AGUNAN, JENIS USAHA, KECAMATAN, dan KELURAHAN.Berdasarkan PERMEN BUMN Nomor PER-05/MBU/2007, kualitas pinjaman disebut Macet apabila keterlambatan pembayaran angsuran pokok dan/atau jasa administrasi pinjaman yang telah melampaui 270 hari dari tanggal jatuh tempo pembayaran angsuran, karena itu data kredit awal pada atribut KELAS dari tahun 2003 s/d triwulan III tahun 2009 yang kualitas pinjamnya KURANGLANCAR atau DIRAGUKAN diubah menjadi MACET (322 record).Perbaikan data pada atribut AGUNANdilakukan dengan mengubah 2 nilai data, yaitu BPKB (70 record) dan SERTIFIKAT (414 record). Data dengan nilai BPKB diubah berdasarkan nilai dari atribut PINJAMAN, jika nilai atribut PINJAMAN sama dengan KECIL, maka data dengan nilai BPKB diubah menjadi BPKB MOTOR, jika tidak, maka diubah menjadi BPKB MOBIL. Data dengan nilai SERTIFIKAT diubah menjadi SERTIFIKAT TANAH.

Perbaikan data pada atribut JENIS USAHA, KECAMATAN, dan KELURAHAN, dilakukan dengan memperbaiki kesalahan ketik, seperti pada atribut JENIS USAHA, data MEUBEL diubah menjadi MEBEL; pada atribut KECAMATAN, data PEBOBO diubah menjadi OEBOBO; pada atribut KELURAHAN, data LILIB diubah menjadi LILIBA. Selain data yang tidak konsisten, perbaikan data juga dilakukan pada data yang nilainya hilang.Data yang nilainya hilang terdapat pada atribut AGUNAN, JENIS USAHA, KECAMATAN, dan KELURAHAN.Cara pengisiannya dilakukan secara proporsional berdasarkan nilai-nilai data yang ada pada masing-masing atribut.

2.4.3 Transformasi Data

Transformasi data adalah proses mengubah data ke dalam bentuk yang dapat digunakan untuk proses *data mining*. Terdapat 2 kegiatan yang dilakukan:

1. Melakukan pengkategorian terhadap data yang bersifat kontinyuatau numerik.Berdasarkan hasil pemilihan atribut, bentuk data yang atributnya masih kontinyu hanyalah pinjaman sehingga perlu dikategorikan atau didiskritkan.Hasilnya, diperoleh data yang terbagi menjadi tiga kategori yaitu KECIL (0 s/d 15 juta), SEDANG (>15 s/d 30 juta), dan BESAR (>30 juta). Fungsi yang digunakan adalah fungsi *IF*, selengkapnya dari fungsi ini:
   IF(A1<=15000000;"KECIL";IF(A1<=30000000;"SEDANG";"BESAR"))
2. Melakukan pengelompokan data pada atribut JENIS USAHA, pengelompokan dilakukan karena nilai data pada atribut ini sangat banyak sehingga nilai data yang mirip dikelompokan ke sebuah nilai.Potongan hasil pengelompokan ditunjukkan pada Tabel 2.

Tabel 2 Potongan transformasi data pada atribut jenis usaha

Sebelum Transformasi		Sesudah Transformasi	
Nilai Data	**Jumlah Record**	**Nilai Data**	**Jumlah Record**
WARUNG	2	RUMAH MAKAN	20
WARUNG MAKAN	1		
RUMAH MAKAN	17		
SALON	14	SALON	20
SALON KECANTIKAN	4		
SALON KENCANTIKAN	1		
PANGKAS RAMBUT	1		

2.4.4 Mengidentifikasi Kesalahan Data

Setelah data dibersihkan dan ditransformasi, langkah selanjutnya merekapitulasi setiap nilai data dari masing-masing atribut. Dari hasil rekapitulasi ini, lalu diidentifikasi nilai data yang jumlah recordnya dibawah 1% dari total record data (kurang atau sama dengan 10 record). Nilai 1% digunakan karena dari hasil pengujian oleh [7], nilai *minimum support* terbaik pada 1%-2%. Nilai-nilai data hasil identifikasi dari masing-masing atribut ini, lalu diubah ke nilai data lain dari atribut bersangkutan yang jumlah recordnya banyak (lebih dari 10 record) dan memiliki kemiripan dengan nilai data hasil identifikasi pada atribut bersangkutan. Selengkapnya perbaikan kesalahan klasifikasi ditunjukkan pada Tabel 3.

Tabel 3 Perbaikan kesalahan klasifikasi

Sebelum Perbaikan Kesalahan Klasifikasi			Setelah Perbaikan Kesalahan Klasifikasi		Keterangan
Atribut	**Nilai Data**	**Jumlah Record**	**Nilai Data**	**Jumlah Record**	
KELAS	KURANG LANCAR	3	MACET	493	Nilai data KURANG LANCAR dan DIRAGUKAN diubah nilainya menjadi MACET
	DIRAGUKAN	4	LANCAR	561	
	MACET	486			
	LANCAR	561			

Tampak pada Tabel 3 bahwa nilai data KURANGLANCAR dan DIRAGUKAN pada atribut KELAS, diubah nilai datanya menjadi MACET karena jumlah recordnya hanya 3 dan 4 record (kurang atau sama dengan 10 record) dan juga memiliki kemiripan dengan nilai data MACET (KURANG LANCAR, DIRAGUKAN, dan MACET merupakan nilai data dari kualitas kredit bermasalah).

2.5 Classification Based On Association (CBA)

AlgoritmaCBA terdiri atas 2 algoritma, yaitu (1) CBA-*Rule Generator* (CBA-RG) dan (2) CBA-*Classifier Building* (CBA-CB). Algoritma CBA-RG digunakan untuk menghasilkan sekumpulan *Class Association Rules* (CARs) yang memenuhi nilai ambang *minimum support*

dan *minimum confidence*. CARs yang telah dihasilkan,selanjutnya digunakan oleh algoritma CBA-CB untuk membangun model.

2.5.1 Konsep Dasar

Data *training* atau dataset *D* mempunyai *m* atribut $A_1, A_2, \ldots, A_m$ dan *Y* adalah daftar label kelas. Atribut bisa berupa atribut kategori (diskrit) atau kontinyu (numerik), atribut kontinyu perlu didiskritisasi.

Definisi 3.1 *Sebuah record data dalam D merupakan kombinasi dari nama atribut A_i dan nilainya a_i serta label kelas y_j.*

Definisi 3.2 *Sebuah item merupakan pasangan dari (A_i, a_i).*

Definisi 3.3 *Sebuah itemset merupakan kumpulan dari satu atau lebih item yang terdapat dalam D.*

Definisi 3.4 *Sebuah ruleitem r berbentuk <itemset, y> dimana $y \in Y$ merupakan label kelas, r disajikan dengan bentuk: itemset $\rightarrow y$.*

Definisi 3.5 *support count dari itemset (itemsetsuppcount) adalah jumlah record dalam D yang memuat itemset.*

Definisi 3.6 *support count dari ruleitem (rulesuppcount) adalah jumlah record dalam D yang memuat itemset dan dilabelkan dengan kelas y.*

Definisi 3.7 *Sebuah ruleitem r memenuhi nilai ambang minsupp jika (rulesuppcount(r)/|D|) ≥ minsupp, dimana |D| adalah jumlah total record dalam D.*

Definisi 3.8 *Sebuah ruleitem r memenuhi nilai ambang minconf jika (rulesuppcount(r)/itemsetsuppcount(r)) ≥ minconf.*

Definisi 3.9 *class association rules (CARs) terdiri atas semua rule yang memenuhi nilai ambang minimum support dan minimum confidence.*

Definisi 3.10 *ruleitem yang memiliki itemset sebanyak k items disebut k-ruleitems.*

Definisi 3.11 *frequent ruleitems F_k menunjukkan sekumpulan frequent k-ruleitems, setiap elemen dari kumpulan ini berbentuk:<(itemset, itemsetsuppcount), (y, rulesuppcount)>*

Definisi 3.12 *frequent ruleitems baru yang dihasilkan dari frequent ruleitems pada tahap sebelumnya disebut candidate ruleitems C_k.*

Definisi 3.13 *Sebuah CARs berbentuk $(A_{i1}, a_{i1}) \wedge (A_{i2}, a_{i2}) \wedge \ldots \wedge (A_{im}, a_{im}) \rightarrow y_j$, dimana antecedent dari ruleitem adalah itemset dan consequent adalah label kelas.*

2.5.2 Algoritma CBA-RG

Algoritma CBA-RG terdiri atas 2 tahapan, yaitu (1) menemukan semua *frequent ruleitems* melalui serangkaian tahapan yang mirip dengan algoritma Apriori dan (2) membangun CARs menggunakan *frequent ruleitems* yang diperoleh. Selengkapnya Algoritma CBA-RG ditunjukkan pada Gambar 2.

```
1   F1 = {large 1-ruleitems};
2   CAR1 = genRules(F1);
3   for (k = 2; Fk-1 ≠ ∅; k++) do
4       Ck = candidateGen(Fk-1);
5       for each data case d ∈ D do
6           Cd = ruleSubset(Ck, d);
7           for each candidate c ∈ Cd do
8               c.itemsetsuppcount++;
9               if d.class = c.class then c.rulesuppcount++
10          end
11      end
12      Fk = {c ∈ Ck | c.rulesuppcount ≥ minsup};
13      CARk = genRules(Fk);
14  end
15  CARs = ∪k CARk;
```

Gambar 2 Algoritma CBA-RG tanpa *prunning*[7]

Tampak pada Gambar 2 bahwa baris 1-2 merupakan tahap pertama algoritma. Tahap ini menghitung jumlah data berdasarkan *item* dan kelas-nya untuk memperoleh *frequent 1-ruleitem* (baris 1). Kumpulan *frequent1-ruleitem* ini, lalu diproses menggunakan fungsi *genRules* untuk menghasilkan sekumpulan CAR_1 (baris 2).

Tahap-tahap berikutnya (tahap *k*) terbagi atas 4 langkah. Pertama, *frequent ruleitem* F_{k-1} yang diperoleh pada tahap (k-1) digunakan untuk menghasilkan *candidate ruleitem* C_k menggunakan fungsi *candidateGen* (baris 4), selanjutnya data *training* di proses satu per satu untuk mengupdate nilai *itemsetsuppcount* dan *rulesuppcount* dari *candidate* C_k (baris 5-11). Setelah itu, *frequent ruleitem* baru teridentifikasi untuk membentuk F_k (baris 12), lalu F_k yang diperoleh digunakan oleh fungsi *genRules* (baris 13) untuk menghasilkan *ruleitem*CAR_k.Terakhir, semua CAR_k yang diperoleh digabung dan disimpan di *CARs* (baris 15).

2.5.3 Algoritma CBA-CB

Algoritma CBA-CB digunakan untuk membangun model menggunakan CARs yang dihasilkan oleh algoritma CBA-RG. CARstersebut perlu diurutkan dari besar ke kecil menggunakan kriteria pengurutan: diberikan 2 buah *ruleitem*, yaitu r_1 dan r_2, urutan r_1 lebih tinggi daripada urutan r_2 (r_1 lebih diprioritaskan dibandingkan r_2) jika:

1. ***confidence*:** $conf(r_1) > conf(r_2)$, atau
2. ***support*:** $conf(r_1) = conf(r_2)$, tetapi $supp(r_1) > supp(r_2)$, atau
3. $conf(r_i) = conf(r_j)$ dan $supp(r_i) = supp(r_j)$, tetapi r_1 diperoleh terlebih dahulu dibandingkan r_2.

Salah satu Algoritma CBA-CB yang digunakan untuk membangun model adalah Algoritma M2.

Algoritma M2

Menurut [12], konsep dasar Algoritma ini terdiri atas 4 tahap:

Tahap 1. Mengidentifikasi *ruleitem r* pada CARs yang benar dan salah mengklasifikasi record data *training d*. Setiap *d* diproses untuk memperoleh 2 *r* dengan prioritas tertinggi, yaitu *cRule* dan *wRule*. *r*yang benar mengklasifikasi *d* disebut *cRule*, sedangkan *r* yang salah mengklasifikasi *d* disebut *wRule*. Tandai *cRule* untuk menunjukkan bahwa *cRule* benar mengklasifikasi *d* jika prioritas *cRule*>*wRule*. Sebaliknya, jika *wRule*>*cRule* maka perlu dibuat sebuah struktur data yang berbentuk: <*d.id*, *d.class*, *cRule*, *wRule*>, dimana *d.id*adalah nomor identifikasi unik dari *d*, dan*d.class* adalah label kelas dari*d*. Algoritma tahap ini ditunjukkan pada Gambar 3.

```
1.  sortRuleitem(r)
2.  for each case d in D do
3.      wRule = getWrule(d, d.class)
4.      cRule = getCrule(d, d.class)
5.      if cRule != null then
6.          cRule.isAcRule ← true
7.          cRule.classCasesCovered[d.class]++
8.          if wRule = null then
9.              cRule.isAstrongCrule ← true
10.         else
11.             if ruleIsCBAgreater(cRule,wRule) then
12.                 cRule.isAstrongCrule ← true
13.             else
14.                 insertIntoSetA(d.id, d.class, cRule, wRule)
15.             end
16.         end
17.     end
18. endfor
```

Gambar 3 Algoritma M2 tahap 1

Tampak pada Gambar 3 bahwa prosedur *sortRuleitem* digunakan untuk mengurutkan *r* berdasarkan skala prioritas. Fungsi *getWrule* digunakan untuk menemukan *r* dengan prioritas tertinggi yang salah mengklasifikasi *d*. Fungsi *getCrule* digunakan untuk menemukan *r* dengan

prioritas tertinggi yang benar mengklasifikasi *d*. Field *isAcRule* dari *r* digunakan untuk menunjukan bahwa *r* tersebut pernah mengklasifikasi secara benar *d*. Field *classCasesCovered* dari *r* digunakan untuk mengetahui berapa banyak *r* tersebut benar mengklasifikasi *d* pada setiap label kelas. Field *isAstrongCrule* dari *r* digunakan untuk menunjukkan bahwa *r* tersebut pernah mempunyai prioritas *cRule* lebih tinggi daripada prioritas *wRule*. Fungsi *ruleIsCBAgreater* digunakan untuk mengetahui apakah prioritas *cRule* lebih tinggi daripada prioritas *wRule*.Prosedur *insertIntoSetA* digunakan untuk menyimpan *d* dan *r* terkait apabila *d* tersebut ketika diproses ke *r* menghasilkan prioritas *wRule* lebih tinggi daripada *cRule*.

Prosedur *sortRuleitem* pada baris 1 digunakan untuk mengurutkan *r* berdasarkan skala prioritas, lalu untuk setiap *d* dicari *r* dengan prioritas tertinggi yang salah mengklasifikasi *d* (*wRule*) menggunakan fungsi *getWrule* (baris 3), dan *r* dengan prioritas tertinggi yang benar mengklasifikasi *d* (*cRule*) menggunakan fungsi *getCrule* (baris 4). Baris 5-17 mengikuti aturan:

1. Tidak ada *cRule*, maka tidak melakukan apa-apa.
2. *cRule* ada, maka field *isAcRule* dari *cRule* diupdate nilainya menjadi *true* (baris 6), dan field *classCasesCovered* yang indexnya sesuai dengan label kelas dari *d*, nilainya ditambahkan 1 (baris 7).
3. *cRule* ada tetapi *wRule* tidak ada, maka tandai *cRule* sebagai sebuah *"Strong"cRule* (baris 8-9).
4. *cRule* dan *wRule* ada dan prioritas *cRule* lebih tinggi daripada *wRule*, maka tandai *cRule* sebagai sebuah *"Strong"cRule* (baris 10-12).
5. *cRule* dan *wRule* ada dan prioritas *wRule* lebih tinggi daripada *cRule*, maka tambahkan record tersebut ke struktur data *SetA* (baris 13-14).

Tahap 2. Mempertimbangkan *ruleitem r* yang salah mengklasifikasi record data *training d*. Tahap ini dilakukan proses terhadap sekumpulan data di *SetA* yang telah diperoleh pada tahap 1. Algoritma tahap ini ditunjukkan pada Gambar 4.

```
1.  for each entry <dID, y, cRule, wRule> in A do
2.     if wRule.isAcRule = true then
3.          cRule.classCasesCovered[y]--
4.          wRule.classCasesCovered[y]++
5.     else
6.          for each ruleitem r beetwen wRule and cRule do
7.               if r.isAcRule = true then
8.                    if isSubset(r.antecedent, dID) then
9.                         Overrides ← insertIntoOverrides(cRule,diD,y)
10.                        Overrides.next ← r.replaceList
11.                        r.replaceList ← Overrides
12.                        r.classCasesCovered[y]++
13.                        r.isAstrongCrule ← true
14.                   end
15.              end
16.         endfor
17.    end
18. endfor
```

Gambar 4 Algoritma M2 tahap 2

Tampak pada Gambar 4 bahwa pemrosesan dilakukan pada setiap record di *SetA* (baris 1). Pemrosesan inimemiliki 2 kemungkinan:

1. **Kemungkinan 1**: *wRule* tersebut pernah menjadi *cRule* untuk *d* yang lain (baris 2), maka untuk kemungkinan ini, nilai data pada field *classCasesCovered* dari *cRule* yang indexnya sesuai dengan label kelas pada *SetA*,dikurangkan 1 (baris 3), dan pada field *classCasesCovered* dari *wRule* yang indexnya sesuai dengan label kelas pada *SetA*,ditambah 1 (baris 4).
2. **Kemungkinan 2**: *wRule* tersebut tidak pernah menjadi *cRule*untuk *d*yang lain (baris 5), maka untuk kemungkinan ini, dilakukan pembacaan terhadap *r* diantara *wRule* dan *cRule* (baris 6). Jika field *isAcRule* dari *r* bernilai *true* (baris 7) dan *antecedent* dari *r* merupakan anggota bagian dari *d* yang sesuai (baris 8), maka tambahkan data ke *List<Overrides> Overrides* menggunakan prosedur insertIntoOverrides (baris 9), lalu tambahkan data

r.replaceList ke *Overrides* (baris 10), setelah itu data pada *Overrides* tersebut disimpan ke field *replaceList* dari *r* (baris 11).Selanjutnya, nilai data pada field *classCasesCovered* dari *r* yang indexnya sesuai dengan label kelas pada *SetA*,ditambahkan 1 (baris 12), dan pada field *isAstrongCrule* dari *r*, nilainya diupdate menjadi *true* (baris 13).

Tahap 3.Menandai sekumpulan *ruleitem r* untuk membentuk model.Algoritma tahap ini ditunjukkan pada Gambar 5.

```
1.  classDistr = genClassclassCasesCovered(D)
2.  ruleErrors ← 0
3.  for each strong cRule r in R do
4.      if r.classCasesCovered[r.class] != 0 then
5.          if r.replaceList != null then
6.              for each (cRule, diD, y) in r.replaceList do
7.                  if the dID case has been covered by a previous r then
8.                      r.classCasesCovered[y]--
9.                  else
10.                     cRule.classCasesCovered[y] --
11.                 end
12.             endfor
13.         end
14.         ruleErrors ← ruleErrors + errorsOfRule(r)
15.         classDistr ← updateClassDistr(r, classDistr)
16.         r.defaultClass ← selectDefault(classDistr)
17.         defaultErrors ← defErr(r.defaultClass, classDistr)
18.         r.totalErrors ← ruleErrors + defaultErrors
19.     end
20. end for
```

Gambar 5 Algoritma M2 tahap 3

Tampak pada Gambar 5 bahwa fungsi *genClassclassCasesCovered* (baris 1) digunakan untuk menghitung jumlah *d* pada setiap label kelas dari *d* yang ada.Baris 2, *ruleErrors* digunakan untuk mencatat jumlah error yang dibuat *r* terpilih pada *d*. Pemrosesan hanya dilakukan pada *r* yang merupakan *strong cRule* (field *isAstrongCrule* dari *r* bernilai *true*) dalam *R* (baris 3), lalu dicek jika nilai data field *classCasesCovered* dari *r* untuk label kelas yang sesuai dengan label kelas dari *r* tidak sama dengan 0 (baris 4), maka diproses dengan aturan:

1. Baris 5-13, memproses data yang ada pada field *replaceList* dari *r* terlebih dahulu jika nilai datanya tidak sama dengan *null*. Setiap data yang ada pada field *replaceList* dibaca lalu diproses, jika *diD* dari *r.replaceList* telah terpenuhi oleh *r* sebelumnya, maka nilai data field *classCasesCovered* dari *r* yang indexnya sesuai dengan label kelas pada *r.replaceList*, dikurangkan 1, sedangkan jika belum terpenuhi oleh *r* sebelumnya, maka nilai data field *classCasesCovered* dari *cRule* (referensi dari *r.replaceList*)yang indexnya sesuai dengan label kelas pada *r.replaceList*, dikurangkan 1.
2. Baris 14, menghitung akumulasi jumlah kesalahan klasifikasi (*ruleErrors*).
3. Baris 15, mengupdate data pada field *classDistr* dari *r* terkini,yaitu dengan mengurangkan data dari field *classDistr* pada *r* sebelumnya dengan data pada field *classCasesCovered* dari *r* terkini sesuai dengan label kelasnya masing-masing.
4. Baris 16, memilih sebuah label kelas dengan nilai tertinggi yang ada pada *classDistr* untuk dijadikan kelas default yang kemudian disimpan ke field *defaultClass* dari *r* bersangkutan.
5. Baris 17, menghitung jumlah *defaultErrors* yang diperoleh dari jumlah data pada *classDistr* yang label kelasnya tidak sama dengan nilai pada field *defaultClass* dari *r* bersangkutan.
6. Baris 18, menghitung jumlah keseluruhan error (*r.totalErrors*) yang diperoleh dari penjumlahan *ruleErrors* dan *defaultErrors*.

Tahap 4.Menghasilkan model. Tahap ini dilakukan proses terhadap sekumpulan *ruleitem r* yang telah ditandai untuk menghasilkan model. Algoritma tahap ini ditunjukkan pada Gambar 6.

```
1.  lowestTotalError ← findLowestTotalError()
2.  for each r.totalErrors ≥ lowestTotalError do
3.      if r.isAstrongCrule Then
4.          if r.classCasesCovered[r.class] != 0 Then
5.              insertRuleIntoRuleList(r.antecedent,r.consequent,
                r.rulesuppcount, r.confidence);
6.          end
7.      end
8.      if r.totalErros = lowestTotalError Then break;
9.  endfor
10. insertRuleIntoRuleList(null,r.consequent,0,0);
```

Gambar 6 Algoritma M2 tahap 4

Tampak pada Gambar 6 bahwa algoritma tahap ini terdiri atas 3 bagian:

1. Baris 1, mencari nilai terendah pada field *totalErros* dari *r* yang ada menggunakan fungsi *findLowestTotalError*, lalu hasilnya disimpan ke variabel *lowestTotalError*.
2. Baris 2-7, memproses setiap *r* pada field *totalErrors* yang nilai datanya lebih besar atau sama dengan nilai lowestTotalError, jika field *isAstrongCrule* dari *r* bernilai *true* dan nilai field *classCasesCovered* dari *r* yang indexnya sesuai dengan label kelas dari *r* tidak sama dengan 0, maka tambahkan *r* tersebut ke *RuleList* menggunakan prosedur insertRuleIntoRuleList.
3. Baris 8-10, jika nilai *r.totalErrors* sama dengan *lowestTotalError* maka proses pencarian *r* untuk dijadikan model selesai (baris 8), dan menambahkan data *kelasDefault* dari *r* (*r.consequent*) yang terakhir ke *RuleList* menggunakan fungsi *insertRuleIntoRuleList* (baris 10).

3. HASIL DAN PEMBAHASAN

Setelah sistem terbangun, selanjutnya dilakukan pengujian menggunakan sistem tersebut. Pengujian dilakukan terhadap 2 aspek, yaitu (1) pengujian waktu pemrosesan dan jumlah *ruleitem* yang dihasilkan, dan (2) Pengujian akurasi.

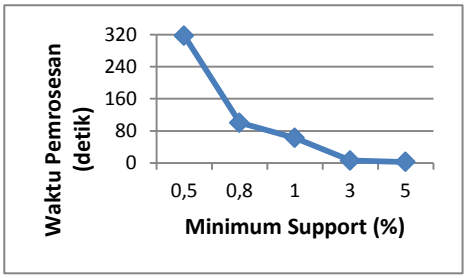
3.1 Pengujian Waktu Pemrosesan Dan Jumlah Ruleitem Yang Dihasilkan

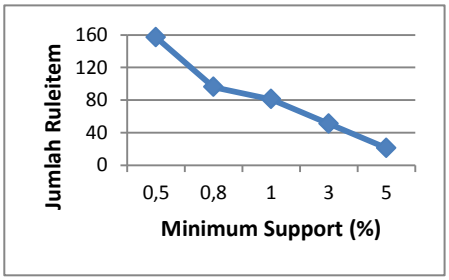
Perangkat keras yang digunakan untuk pengujian adalah processor Intel Core i3-2310M CPU @ 2.10GHz dengan RAM 4 GB.Pengujian menggunakan nilai parameter *minimum confidence* dan *minimum support*. Nilai *minimum confidence* yang digunakan sebesar 50%, sesuai dengan nilai yang digunakan oleh [7].Nilai *minimum confidence* ini, lalu diproses dengan 5 buah nilai *minimum support*,yaitu 0,5%, 0,8%, 1%, 3%, dan 5%. Hasil pengujian ditunjukkan pada Tabel 4 dan Gambar 7.

Tabel 4 Hasil pengujian waktu pemrosesan dan jumlah *ruleitem* (*minimum confidence* 50%)

No	*Minimum Support* (%)	Waktu Pemrosesan (detik)	Jumlah *Ruleitem*
1	0,5	316,78	157
2	0,8	100,35	96
3	1	62,62	81
4	3	6,15	51
5	5	2,57	21

(a)





(b)

Gambar 7 Grafik waktu pemrosesan dan jumlah *ruleitem*(*minimum confidence* 50%)

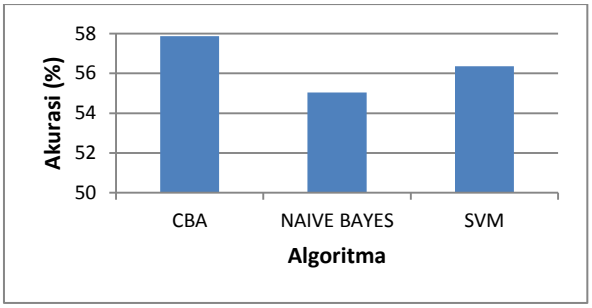
Tampak pada Tabel 4 dan Gambar 7 bagian (a) bahwa lama waktu pemrosesan berlawanan dengan besar nilai *minimum support*, semakin besar nilai *minimum support* maka semakin cepat waktu pemrosesan yang dibutuhkan, sebaliknya semakin kecil nilai *minimum support* maka semakin lama waktu pemrosesan. Gambar 7 bagian (b) menunjukkan bahwa jumlah *ruleitem* berlawanan dengan besar nilai *minimum support*, semakin besar nilai *minimum support* maka semakin sedikit jumlah *ruleitem* yang dihasilkan, sebaliknya semakin kecil nilai *minimum support* maka semakin banyak jumlah *ruleitem* yang dihasilkan.

3.2 Pengujian Akurasi

Pengujian dilakukan pada aplikasi yang dibangunmenggunakan algoritma CBA, dan pada perangkat lunak lain, yaitu aplikasi Rapid Miner 5.3 yang menggunakan algoritma *Naive Bayes* dan SVM.Hasilnya pengujian ditunjukkan pada Tabel 5 dan Gambar 8.

Tabel 5 Perbandingan rata-rata akurasi model dari beberapa algoritma

No	Algoritma	Rata-Rata Akurasi (%)
1	CBA	57,86
2	Naive Bayes	55,03
3	SVM	56,35



Gambar 8 Grafik rata-rata akurasi model dari beberapa algoritma

Tampak pada Tabel 5 dan Gambar 8 bahwa rata-rata nilai akurasi model menggunakan algoritma CBA paling tinggi, diikuti algoritma SVM dan *Naive Bayes* dengan nilai akurasi model berturut-turut adalah 57,86%, 56,35%, dan 55,03%.

KESIMPULAN

Berdasarkan hasil pengujian yang telah diperoleh, dapat ditarik kesimpulan:

1. Hasil pengujian waktu pemrosesan menggunakan nilai *minimum confidence* 50%, menunjukkan bahwa semakin besar nilai *minimum support* yang digunakan, maka semakin cepat waktu pemrosesan yang dibutuhkan.
2. Hasil pengujian jumlah *ruleitem* yang dihasilkan menggunakan nilai *minimum confidence* 50%, menunjukkan bahwa semakin besar nilai *minimum support* yang digunakan, maka semakin sedikit jumlah *ruleitem* yang dihasilkan.
3. Hasil pengujian akurasi model menggunakan aplikasi yang dibuat dan perangkat lunak Rapid Miner 5.3, menunjukkan bahwa rata-rata nilai akurasi pada aplikasi yang dibuat dengan algoritma CBA (57,86%), sedikit lebih tinggi dibandingkan rata-rata nilai akurasi menggunakan algoritma *Support Vector Machine* dan *Naive Bayes* dari perangkat lunak Rapid Miner 5.3 (56,35% dan 55,03%).

SARAN

Pengembangan perangkat lunak ini masih memiliki keterbatasan yang dapat dijadikan acuan untuk pengembangan dimasa yang akan datang, sehingga disarankan beberapa hal:

1. Aplikasi yang dibuat dengan algoritma CBA, selanjutnya dapat dibandingkan dengan algoritma asosiasi klasifikasi lainnya MCAR (*Multi-class Classification based on Association Rules*) [13] dari aspek kecepatan waktu pemrosesan dan akurasi model yang diperoleh.
2. Kecepatan waktu pemrosesan algoritma CBA, dapat ditingkatkan dengan algoritma Apriori-TFP (*Total from Partial*) [14], yang memiliki kecepatan pemrosesan lebih baik dibandingkan algoritma apriori.

DAFTAR PUSTAKA

[1] Han, J., dan Kamber, M., 2006, *Data Mining: Concepts and Techniques*, Second Edition, Morgan Kaufman, San Francisco.

[2] Antonakis, A.C., dan Sfakianakis, M.E., 2009, Assessing naive Bayes as a method for screening credit applicants, *J. of Applied Statistics*, Vol.36, No.5 May 2009, hal.537-545.

[3] Chen, W.-H., dan Shih, J.-Y., 2006, A study of Taiwan's issuer credit rating systems using support vector machines, *J. Expert System with Applications 30 (2006)*, hal.427-435.

[4] Shin, K.-S., Lee, T.S., dan Kim, H.-j., 2005, An application of support vector machines in bankruptcy prediction model, *J. Expert Systems with Applications 28 (2005)*, hal.127-135.

[5] Xu, W., Zhou, S., Duan, D., dan Chen, Y., 2010, A Support Vector Machine Based Method For Credit Risk Assessment, *IEEE Int. Conf. on E-Business Engineering*, hal.50-55.

[6] Liu, B., 2011.*Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*, Second Edition, Springer, New York.

[7] Liu, B., Hsu, W., dan Ma, Y., 1998, Integrating Classification and Association Rule Mining, *Proc. of the Fourth Int. conf. on Knowledge Discovery and Data Mining*, 27 - 31 Agustus 1998, hal.80-86.

[8] Liu, B., Ma, Y., dan Wong, C.-K., 2001, *Classification Using Association Rules: Weaknesses and Enhancements*, Grossman, R.L., Kamath, C., Kegelmeyer, P., Kumar, V., dan Namburu, R.R., *Data Mining for Scientific and Engineering Applications*, Kluwer Academic Publishers, New York.

[9] Thabtah, F., dan Cowling, P., 2008, Mining the data from a hyperheuristic approach using associative classification, *J. Expert Systems with Applications 34 (2008)*, hal.1093-1101.

[10] Srisawat, A., dan Kijsirikul, B., 2004, Using Associative Classification for Predicting HIV-1 Drug Resistance, *Proc. of the Fourth Int. Conf. on Hybrid Intelligent Systems (HIS'04)*, 5-8 Desember 2004, hal.280-284.

[11] Larose, D. T., 2005, *DISCOVERING KNOWLEDGE IN DATA: An Introduction to Data Mining*, John Wiley & Sons Inc, New York.

[12] Coenen, F., 2005, The LUCS-KDD Implementations of the CBA Algorithm, *http://cgi.csc.liv.ac.uk/~frans/KDD/Software/CBA/cba.html*, diakses tanggal 15 April 2013.

[13] Thabtah, F., Cowling, P., dan Hammoud, S., 2005, Improving rule sorting, predictive accuracy and training time in associative classification, *J. Expert Systems with Applications31 (2006)*, hal.414-426.

[14] Coenen, F., Leng, P., dan Ahmed, S., 2004, Data Structure for Association Rule Mining: T-Trees and P-Trees, *J. IEEE Transactions on Knowledge and Data Engineering*, Vol. 16, No. 6, Juni 2004, hal.1-5.