DATA MINING
ALGORITMA C4.5

Disertai contoh kasus dan penerapannya dengan program komputer

MUCH AZIZ MUSLIM
BUDI PRASETIYO
EVA LAILY HARUM M
ANISA JULI H
MIRQOTUSSA'ADAH
SITI HARDIYANTI R
ALDI NURZAHPUTRA

Data Mining

Algoritma C4.5

Disertai contoh kasus dan penerapannya dengan program computer

Cetakan: Pertama, 2019

Penulis

Much Aziz Muslim

Budi Prasetiyo

Eva Laily Harum Mawarni

Anisa Juli Herowati

Mirqotussa'adah

Siti Hardiyanti Rukmana

Aldi Nurzahputra

Editor

Eka Listiana

Nova Cahyani

Desain Sampul: Doni Aprilianto

Diterbitkan Oleh:

KATA PENGANTAR

Pertumbuhan yang sangat pesat dalam bidang *data mining*, sebanding dengan kecepatan pertumbuhan data di era *big data* saat ini. *Data mining* memiliki 4 teknik, diantaranya klasifikasi, klasterisasi, regresi, dan asosiasi. diantara keempat teknik *data mining*, klasifikas menjadi teknik yang popular saat ini. algoritmA yang digunakan dalam fungsi klasifikasi pun sangat beragam mulai dari C4.5, SVM, Adaboost, Naïve Bayes, *Multilayer Preercepron*, dll. Namun tidak semuanya akan kita bahas seacara rinci dalam buku ini. Buku ini akan fokus pada algoritma C4.5, sesuai dengan judul buku.

Pada buku ini, kita akan mendiskusikan algoritma C4.5 ke dalam lima bab. Pada bab pertama, kita akan membahas tentang pengantar dan definisi *data mining* serta *dataset*. Pada bab 2 kita akan membahas tentang proses *data mining*, teknik data mining, *knowledge* dan *evaluation*. Selanjutnya, pada bab 3 kita akan membahas lebih jauh tentang algoritma C4.5 disertai dengan perhitungan dan contoh kasus. Pada bab 4 kita kan dikenalkan pada *tool* yang biasa digunakan dalam *data mining* seperti weka dan *RapidMiner*. Pada bab terakhir, akan dijelaskan langkah-langkah atau proses *data mining* dalam beberapa kasus klasifikasi menggunakan *tool* weka dan *RapidMiner*.

Penulis berharap buku ini kn bermanfaat khususnya bagi para pembaca yang sedag menempuh studi dan menekuni bidang *data mining*. Kepada semua pihak yang telah membantu penyelesaian penulisan hingga terbitnya buku ini, penulis mengucapkan terima kasih.

Semarang, Mei 2019

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1

PENGANTAR

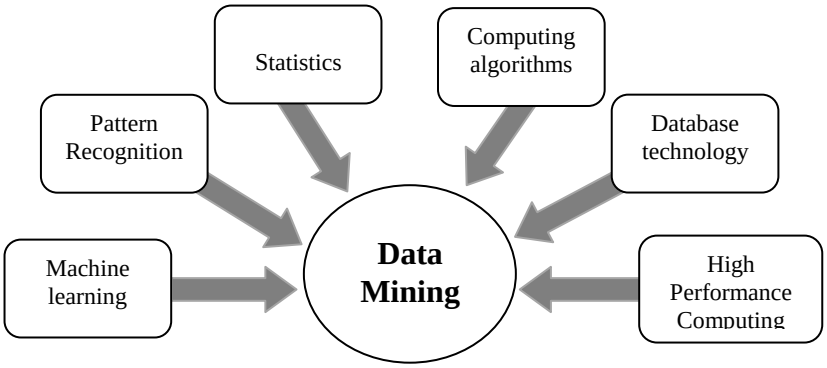
Kebutuhan informasi yang tinggi kadang tidak sebanding dengan penyajian informasi yang memadai. Informasi yang disajikan sering kali masih harus digali dari data dalam jumlah besar. Salah satu contoh yaitu data yang tumbuh dalam bidang kesehatan. Data kesehatan menyimpan banyak sekali data-data yang terkait dalam lingkungan kesehatan seperti data pasien, data obat, data penyakit, yang sangat penting untuk dapat diolah supaya lebih bermanfaat. Metode tradisional yang biasa digunakan untuk menganalisis data, tidak dapat menangani data dalam jumlah besar. Oleh karena itu data tersebut dapat diolah menjadi pengetahuan menggunakan teknik yang disebut *data mining*. Sebagai bidang ilmu yang relatif baru, *data mining* menjadi pusat perhatian para akademisi maupun praktisi. Beragam penelitian dan pengembangan *data mining* banyak diaplikasikan pada bidang kesehatan. Bab ini memberikan pandangan secara singkat mengenai definisi *data mining*, *dataset*, jenis *dataset*, dan jenis atribut.

1.1 Definisi *Data Mining*

Data mining dikenal sejak tahun 1990-an, ketika adanya suatu pekerjaan yang memanfaatkan data menjadi suatu hal yang lebih penting dalam berbagai bidang, seperti marketing dan bisnis, sains dan teknik, serta seni dan hiburan. Sebagian ahli menyatakan bahwa *data mining* merupakan suatu langkah untuk menganalisis pengetahuan dalam basis data atau biasa disebut *Knowledge Discovery in Database* (KDD). *Data mining* merupakan proses untuk

menemukan pola data dan pengetahuan yang menarik dari kumpulan data yang sangat besar. Sumber data dapat mencakup *database*, *data warehouse*, *web*, *repository*, atau data yang dialirkan ke dalam sistem dinamis (Han, 2006).

Data mining, secara sederhana merupakan suatu langkah ekstraksi untuk mendapatkan informasi penting yang sifatnya implisit dan belum diketahui. Selain itu, *data mining* mempunyai hubungan dengan berbagai bidang diantaranya statistik, *machine learning* (pembelajaran mesin), *pattern recognition*, *computing algorithms*, *database technology*, dan *high performance computing*. Diagram hubungan *data mining* disajikan pada Gambar 1.1.



Gambar 1.1. Diagram hubungan *data mining*

Secara sistematis, langkah utama untuk melakukan *data mining* terdiri dari tiga tahap, yaitu sebagai berikut (Gonunescu, 2011);

1. Eksplorasi atau pemrosesan awal data

   Eksplorasi atau pemrosesan awal data terdiri dari pembersihan data, normalisasi data, transformasi data, penanganan *missing value*, reduksi dimensi, pemilihan subset fitur, dan sebagainya.
2. Membangun model dan validasi

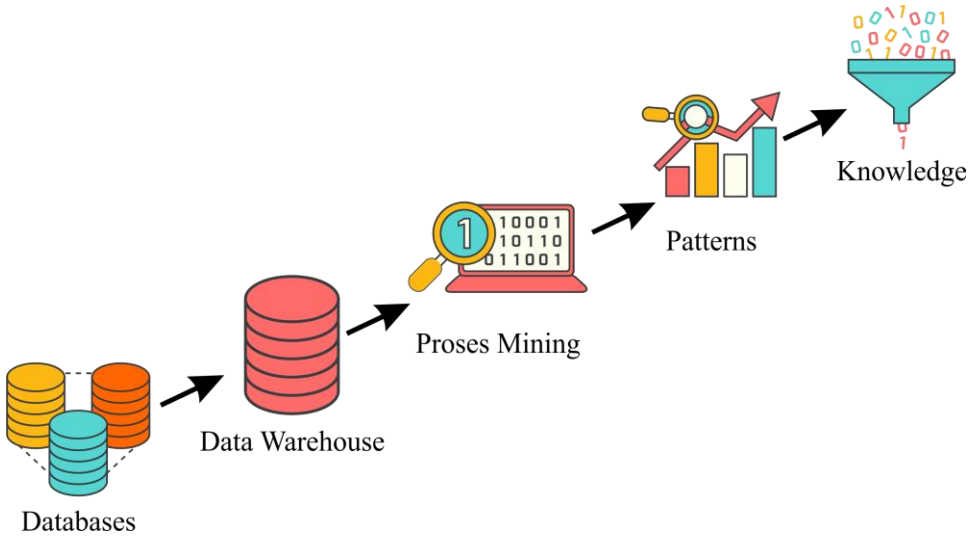
   Membangun model dan validasi, yaitu melakukan analisis dari berbagai model dan memilih model sehingga menghasilkan kinerja yang terbaik.

Pembangunan model dilakukan menggunakan metode-metode seperti klasifikasi, regresi, analisis *cluster*, dan asosiasi.

3. Penerapan

   Penerapan dilakukan dengan menerapkan model yang dipilih pada data yang baru untuk menghasilkan kinerja yang baik pada masalah yang diinvestigasi.

Tahapan proses *data mining* ada beberapa yang sesuai dengan proses KDD sebagaimana seperti yang digambarkan pada Gambar 1.2.



Gambar 1.2. Proses KDD

1. *Cleaning and Integration*

a. Data Cleaning (Pembersihan Data)

Data cleaning (Pembersihan data) adalah proses yang dilakukan untuk menghilangkan *noise* pada data yang tidak konsisten atau bisa disebut tidak relevan. Data yang diperoleh dari *database* suatu perusahaan maupun hasil eksperimen yang sudah ada, tidak semuanya memiliki isian yang sempurna misalnya data yang hilang, data yang tidak valid, atau

bisa juga hanya sekedar salah ketik. Data yang tidak relevan itu dapat ditangani dengan cara dibuang atau sering disebut dengan proses *cleaning*. Proses *cleaning* dapat berpengaruh terhadap performa dari teknik *data mining*.

b. Data Integration (Integrasi data)

Integrasi data merupakan proses penggabungan data dari berbagai *database* sehingga menjadi satu *database* baru. Data yang diperlukan pada proses *data mining* tidak hanya berasal dari satu *database* tetapi juga dapat berasal dari beberapa *database*.

2. *Selection and Transformation*

a. *Data Selection* (Seleksi Data)

Tidak semua data yang terdapat dalam *database* akan dipakai, karena hanya data yang sesuai saja yang akan dianalisis dan diambil dari *database*. Misalnya pada sebuah kasus *market basket analysis* yang akan meneliti faktor kecenderungan pelanggan, maka tidak perlu mengambil nama pelanggan, cukup dengan id pelanggan saja.

b. *Data Transformation* (Transformasi Data)

Transformasi data merupakan proses pengubahan data dan penggabungan data ke dalam format tertentu. *Data mining* membutuhkan format data khusus sebelum diaplikasikan. Misalnya metode standar seperti analisis asosiasi dan *clustering* hanya bisa menerima input data yang bersifat kategorikal. Karenanya data yang berupa angka numerik apabila mempunyai sifat kontinyu perlu dibagi-bagi menjadi beberapa interval. Proses ini sering disebut dengan transformasi data.

3. *Proses Mining*

Proses *mining* dapat disebut juga sebagai proses penambangan data. Proses *mining* merupakan proses utama yang menggunakan metode untuk menemukan pengetahuan berharga yang tersembunyi dari data.

4. *Evaluation and Precentation*

a. Evaluasi Pola (*Pattern Evaluation*)

Evaluasi pola bertugas untuk mengidentifikasi pola-pola yang menarik ke dalam *knowledge based* yang ditemukan. Pada tahap ini dihasilkan pola-pola yang khas dari model klasifikasi yang dievaluasi untuk menilai apakah hipotesa yang ada memang tercapai. Bila ternyata hasil yang diperoleh tidak sesuai dengan hipotesa, terdapat beberapa alternatif yang bisa diambil seperti menjadikannya umpan balik untuk memperbaiki proses *data mining*, atau mencoba metode *data mining* lain yang lebih sesuai.

b. Presentasi Pengetahuan (*Knowledge Presentation*)

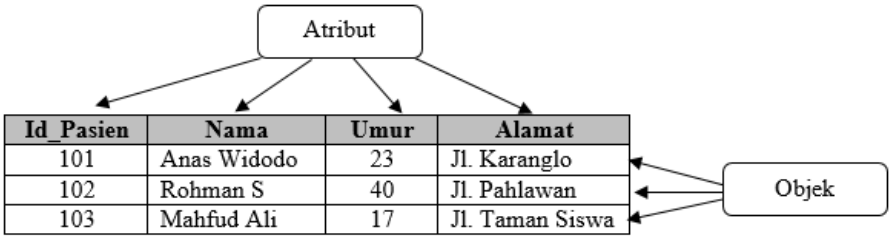
Knowledge presentation merupakan visualisasi dan penyajian pengetahuan mengenai metode yang digunakan untuk memperoleh pengetahuan atau informasi yang telah digali oleh pengguna. Tahap terakhir dari proses *data mining* adalah memformulasikan keputusan dari hasil analisis yang didapat.

1.2 *Dataset*

Data mining tidak pernah lepas dari yang namanya *dataset*, karena dalam pengolahan *data mining*, *dataset* sangat dibutuhkan sebagai objek untuk mendapakan pengetahuan. Dalam terminologi statistik *dataset* adalah kumpulan dari suatu objek yang mempunyai atribut atau variabel tertentu, di mana untuk setiap objek merupakan individu dari data yang mempunyai sejumlah atribut atau

variabel tersebut. Nama lain dari objek yang sering digunakan adalah *record, point, vector, pattern, event, observation*, dan *case*. Sementara itu, baris yang menyatakan objek-objek data dan kolom disebut atribut. Atribut juga dapat disebut dengan variabel, *field*, fitur atau dimensi.

Sebagai contoh seorang pasien merupakan objek, dimana objek pasien tersebut memiliki beberapa atribut seperti id_pasien, nama, umur dan lain-lain. Setiap pasien memiliki nilai atribut yang berbeda dengan pelanggan lainnya. Perbedaan atribut dan objek dapat dilihat pada Gambar 1.3.



Gambar 1.3. Perbedaan atribut dan objek

1.2.1 Jenis-jenis *Dataset*

Karakteristik umum *dataset* yang berpengaruh dalam proses *data mining* ada tiga, diantaranya dimensionalitas, sparsitas, dan resolusi. Sedangkan jenis *dataset* juga ada tiga macam, yaitu sebagai berikut:

1. *Record Data*

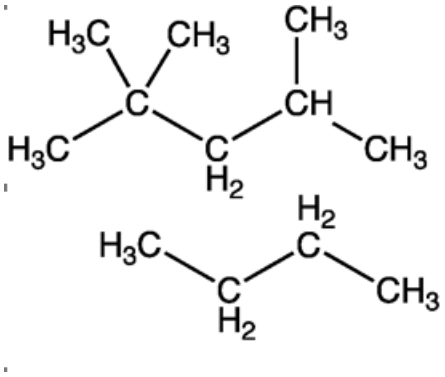
Dataset yang berbentuk *record*, tidak mempunyai hubungan antara baris data yang satu dengan baris data yang lainnya. Setiap baris data berdiri sendiri sebagai sebuah data individu. Jadi, *record data* merupakan data yang terdiri dari sekumpulan *record*, yang masing-masing *record* terdiri dari satu set atribut yang tetap. Contoh *record data* CKD ditunjukkan pada Tabel 1.1.

Tabel 1.1. Contoh *Record Data* CKD

Nama Pasien	Umur	Tekanan Darah	Kepekaan Urine	Kadar Gula	Nanah	Gumpalan Nanah	Kelas
Eka	48	80	1.020	0	Normal	notpresent	ckd
Aldi	7	50	1.020	0	Normal	notpresent	ckd
April	62	80	1.010	3	Normal	notpresent	ckd
Elham	48	70	1.005	0	abnormal	present	not ckd
Hestu	51	80	1.010	0	Normal	notpresent	ckd
Winda	68	70	1.010	0	Normal	notpresent	ckd
Novi	24	?	1.015	4	abnormal	notpresent	notckd
Nerly	50	60	1.010	4	abnormal	present	notckd
Ikhsan	68	70	1.015	1	Normal	present	notckd
Hani	68	80	1.010	2	abnormal	present	ckd
Budi	40	80	1.015	0	Normal	notpresent	ckd
Tiyo	47	70	1.015	0	Normal	notpresent	notckd

2. Data *Graph*

Data *graph* adalah data yang mempunyai bentuk *graph* yang terdiri dari simpul (*node*) dan rusuk (*edge*). Yang termasuk dalam data *graph* diantaranya adalah HTML *links* (dalam WWW), struktur molekul, dan sebagainya. Contoh data *graph* dapat dilihat pada Gambar 1.4.



Gambar 1.4. Contoh data *graph* struktur molekul

3. *Ordered data*

Ordered data merupakan data-data yang memperhatikan urutan nilai-nilainya. Yang termasuk dalam data terurut adalah *genomic sequence data* atau *spatio-temporal data.* Contoh data terurut *genomic sequence data* dapat dilihat pada Gambar 1.5.

GAGGATTAAT	AAATTATAAA	TGTTATTACA
TTACACTGTT	GCACGTCCAC	GTGTTCGTCC
TGATCTTGTT	ATATCATTAT	TATTATTGTT
GTGTACCATA	GTAATCTGAA	AGGAACCGCT
ATAGATTCTA	TTTTCAATTT	CTCAAATCTA
GAACGTGAGT	TATTAAGTTA	ATCTAAATAT

Gambar 1.5. Contoh data terurut *genomic sequence data*

1.2.2 Jenis-jenis atribut

Atribut adalah suatu simbol yang menggambarkan identitas atau karakteristik objek. Sebagai contoh atribut yang menggambarkan objek pasien rumah sakit adalah nama, umur, golongan darah, dan tekanan darah. Berikut penjelasan dari empat macam atribut berdasarkan contohnya.

1. Atribut Nominal

Atribut nominal adalah nilai atribut yang diperoleh dengan cara kategorisasi karena nilainya menggambarkan kategori, kode, atau status yang tidak memiliki urutan. Misalnya, atribut golongan darah yang mempunyai empat kemungkinan nilai yaitu A, B, AB, dan O. Contoh lainnya seperti atribut jenis kelamin yang bisa bernilai pria dan wanita.

2. **Atribut Ordinal**

Atribut ordinal adalah atribut yang memiliki nilai dengan menggambarkan urutan atau peringkat. Namun, ukuran perbedaan antara dua nilai yang berurutan tidak diketahui. Atribut ordinal sangat berguna dalam survei, yaitu untuk penilaian subjektif (kualitatif) yang tidak dapat diukur secara objektif. Misalnya, kepuasan pelanggan yang menghasilkan atribut bernilai ordinal, yaitu 0 (Tidak Puas), 1 (Cukup Puas), 2 (Puas), 3 (Sangat Puas).

3. **Atribut Interval (Jarak)**

Atribut interval adalah atribut numerik yang diperoleh dengan melakukan pengukuran, di mana jarak dua titik pada skala sudah diketahui dan tidak mempunyai titik nol yang absolut. Misalnya, suhu 0°C-100°C atau tanggal 1 sampai tanggal 31.

4. **Atribut Rasio (Mutlak)**

Atribut rasio adalah atribut numerik dengan titik nol absolut. Artinya, jika sistem pengukuran menggunakan rasio, dapat dihitung perkalian atau perbandingan antara suatu nilai dengan nilai yang lain. Misalnya, berat badan Doni 20 kg, berat badan Amanah 40 kg, berat badan Faiz 60 kg dan berat badan Udin 80 kg. Jika diukur dengan skala rasio maka berat badan Udin dua kali berat badan Amanah.

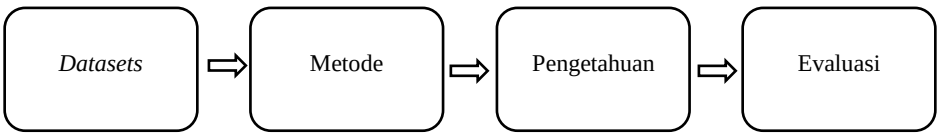
KDD mengalami beberapa proses pengolahan. Sebelum diterapkan pada algoritma *data mining*, *dataset* dapat diolah dengan cepat dan menghasilkan kesimpulan yang tepat. Beberapa proses pengolahan awal adalah proses pengumpulan (*aggregation*), penarikan contoh (*sampling*), pengurangan dimensi (*dimensionality reduction*), pemilihan fitur (*feature selection*), pembuatan fitur (*fitur creation*), pendiskritan dan pembineran (*discretization and binarization*)

dan transformasi atribut (*attribute transformation*). Oleh karena itu, perlu diterapkannya pemrosesan awal pada data sebelum melakukan proses *data mining* yang akan dibahas pada bab selanjutnya.

BAB 2

PROSES DATA MINING

Kinerja yang baik tidak lepas dari proses *data mining* yang baik pula. Oleh karena itu, pada bab ini akan membahas mengenai proses *data mining*, diantaranya ada himpunan data, metode *data mining*, pengetahuan (*knowledge*), dan *evaluation*. Pemilihan algoritma dan teknik yang tepat akan bergantung pada proses *Knowledge Discovery in Database* (KDD) secara keseluruhan. Tahapan untuk menemukan informasi atau pola dari sekumpulan data dengan menggunakan algoritma dan teknik tertentu merupakan pengertian dari *data mining.* Bagan proses *data mining* dapat dilihat pada Gambar 2.1.



Gambar 2.1 Proses *data mining*

2.1. Himpunan Data (Pemahaman dan pengolahan data)

Bukan *data mining* namanya jika tidak ada *dataset* yang diolah di dalamnya. Sedangkan saat ini jumlah data sangat meningkat dari berbagai media seperti *e-commerce*, *e-government*, media elektronik, dan sebagainya. Dari berbagai media tersebut akan menghasilkan data yang sangat besar. Selain itu *dataset* juga dapat diambil dari berbagai organisasi yang akan dijadikan objek penelitian, misalnya Bank, Rumah sakit, Industri, Pabrik, dan sebagainya yang biasa disebut dengan

private dataset. Sedangkan *public dataset* merupakan sekumpulan *dataset* yang diambil dari *public repository* yang sudah disepakati oleh para peneliti *data mining*, misalnya ACM KDD (http://www.sigkdd.org/kddcup/), UCI Repository (http://www.ics.uci.edu/~mlearn/MLRepository.html), dan PredictionIO (http://docs.prediction.io/datacollection/sample/). Pengertian *dataset* sudah dijelaskan pada Bab 1. Untuk mendapatkan hasil dari proses *data mining* yang optimal, maka perlu diadakannya *pre-processing* data. Tujuan dari *pre-processing*, yaitu untuk memudahkan peneliti dalam memahami data yang belum dilakukan proses *data mining*, selain itu bisa juga digunakan untuk meningkatkan kualitas data sehingga hasil *data mining* akan menjadi lebih baik, dan *pre-processing* juga dapat meningkatkan efisiensi serta memudahkan proses penambangan data. Pengolahan data sebelum dilakukan proses *data mining* atau yang sering disebut dengan *pre-processing* data ada empat metode, yaitu pembersihan data, integrasi data, reduksi data, dan transformasi data. Selanjutnya, akan dibahas setiap teknik dari *pre-processing* data secara detail pada subbab berikut ini.

2.1.1. *Data Cleaning* (Pembersihan data)

Data Cleaning merupakan suatu pemrosesan terhadap data untuk penanganan terhadap data yang mempunyai *missing value* pada suatu *record* dan menghilangkan *noise*. *Cleaning* data merupakan langkah awal yang perlu dilakukan, karena dalam *record* data yang digunakan terdapat data yang bernilai karakter "?" atau dapat disebut juga dengan data yang salah (*missing value*) sebagaimana seperti yang ditunjukkan pada Tabel 4.3 merupakan contoh *record dataset crhonic kidney disease* yang mempunyai nilai *missing value*. Oleh karena itu, diberi perlakuan atau penanganan untuk menangani data yang salah (*missing value*).

Jika pada data masih ada yang mempunyai nilai kosong dapat dibersihkan dengan cara sebagai berikut:

1) Abaikan *tuple*

   Mengabaikan *tuple* biasanya dilakukan pada data yang tidak mempunyai label kelas. Cara ini lebih efekif apabila *tuple* tersebut memiliki banyak atribut kosong. Namun, metode ini kurang efektif untuk data yang mempunyai banyak *tuple* dengan sedikit *missing value.*

2) Isi atribut kosong secara manual

   Cara ini dapat digunakan untuk mengatasi kelemahan metode pertama. Namun cara ini tentu saja memerlukan banyak waktu dan seringkali tidak layak untuk himpunan data besar yang mengandung banyak atribut kosong.

3) Menggunakan nilai tendensi sentral rata-rata (*average*)

   Atribut kosong dapat diisi dengan menggunakan model *average* di mana dapat menggantikan *missing value* tersebut dengan nilai rataan berdasarkan nilai yang tersedia pada fitur tersebut, untuk fitur ke-*i*. Contoh data *missing value* dapat dilihat pada Tabel 2.1.

Tabel 2.1. Contoh data yang mempunyai *missing value*

Age	Bp	Bgr	Bu
90	90	121	36
78	?	?	?
?	?	?	?
65	90	423	35
61	80	117	18
60	70	106	53
50	70	74	56

Perhitungan rata-rata untuk mengganti data *missing value* dengan model *average* adalah sebagai berikut.

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} \quad (1)$$

Dimana:

$\bar{x}$: rata-rata

x_i : nilai atribut *i*

n : jumlah data

a. Nilai untuk mengganti *missing value* atribut *Age*:

$\bar{x}(Age) = \frac{\sum_1^7 x(Age)}{6} = \frac{404}{6} = 67{,}3 = 67$

b. Nilai untuk mengganti *missing value* atribut *Bp*:

$\bar{x}(Age) = \frac{\sum_1^7 x(Bp)}{5} = \frac{400}{5} = 80$

c. Nilai untuk mengganti *missing value* atribut *Bgr*:

$\bar{x}(Age) = \frac{\sum_1^7 x(Bgr)}{5} = \frac{841}{5} = 168{,}2 = 168$

d. Nilai untuk mengganti *missing value* atribut *Bu*:

$\bar{x}(Age) = \frac{\sum_1^7 x(Bu)}{5} = \frac{198}{5} = 39{,}6 = 40$

e. Nilai untuk mengganti *missing value* atribut *red blood cells* (rbc):

Jumlah data *record* yang berkategori *normal* adalah 202, sedangkan yang berkategori *abnormal* adalah 47, maka nilai rata-rata yang menggantikan *missing value* adalah "*normal*". Tabel 2.2 menampilkan beberapa data yang telah di-*replace missing value.*

Tabel 2.2. Beberapa contoh data yang telah di *replace missing value*

Age	Bp	Bgr	Bu
90	90	121	36
78	80	168	40
67	80	168	40
65	90	423	35
61	80	117	18
60	70	106	53
50	70	74	56

2.1.2. *Data Integration* (Integrasi Data)

Integrasi data merupakan penggabungan data dari berbagai *database* ke dalam satu *database* baru. Integrasi data yang baik akan menghasilkan data gabungan dengan sedikit redundansi/atau inkonsistensi sehingga meningkatkan akurasi dan kecepatan proses *data mining*. Permasalahan utama dalam integrasi data adalah heterogenitas semantik dan struktur dari semua data yang diintegrasikan.

2.1.3. *Data Reduction* (Reduksi Data)

Tahap seleksi data ini terjadi pengurangan dimensi/ atribut pada *dataset* guna mengoptimalisasi atribut yang akan berpengaruh pada akurasi algoritma dalam me-*mining dataset*. Pengurangan dimensi atau seleksi atribut dapat dilakukan dengan menggunakan beberapa metode, diantaranya sebagai berikut.

1) *Information Gain*

Information Gain merupakan ekspektasi dari pengurangan entropi yang dihasilkan dari partisi objek *dataset* berdasarkan fitur tertentu. Terdapat dua kasus berbeda pada saat perhitungan *information gain*, pertama untuk kasus

perhitungan atribut tanpa *missing value* dan kedua, perhitungan atribut dengan *missing value.*

1. Perhitungan *information gain* tanpa *Missing value*

Menghitung *information gain* tanpa *missing value* digunakan rumus seperti pada persamaan berikut.

$$IG(S, A) = Entropy(S) - \sum_{c \in values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Dimana:

S : himpunan kasus

A : atribut

$|S_i|$: jumlah kasus pada partisi ke-i

$|S|$: jumlah kasus dalam S

Sementara itu, untuk menghitung nilai *entropy* dari koleksi label benda S dan A didefinisikan pada persamaan berikut.

$$Entropy\ (S) = \sum_{i=1}^{c} -p_i log_2 p_i$$

Dimana:

S : himpunan kasus

c : jumlah partisi S

P_i : proporsi dari S_i terhadap S.

Penghapusan atribut dilakukan satu persatu dari atribut yang memiliki nilai *information gain* yang paling kecil lalu akan di-*mining*. Proses pembuangan dan *mining* ini akan berhenti saat hasil akurasi masing-masing algoritma mengalami penurunan.

Contoh:

Kasus perhitungan menggunakan *dataset chronic kidney disease* yang diambil dari UCI *repository of machine learning*. (https://archive.ics.uci.edu/ml/*dataset*s/chronic_kidney_disease). Proses perhitungan nilai *entropy* untuk masing-masing atribut, pertama adalah mencari nilai *entropy* untuk keseluruhan data (*class=”ckd”* dan *class=”notckd”*) terlebih dahulu. Berdasarkan rumus perhitungan *entropy (S)*, diperoleh:

Jumlah *class* target (*class*) = 2 (“*ckd*” dan “*notckd*”)

Jumlah sampel untuk kelas 1 (“*ckd*”) = 250

Jumlah sampel untuk kelas 2 (“*notckd*”) = 150

Total keseluruhan = 400

$$Entropy\ (250{,}150) = -(\left(\frac{250}{400}\right)\log_2(\frac{250}{400})) - (\left(\frac{150}{400}\right)\log_2(\frac{150}{400}))\ = 0{,}954434003$$

Kemudian hitung juga nilai *entropy, info, dan gain* pada setiap atribut, untuk penjelasannya yaitu sebagai berikut.

1. *Information gain* untuk atribut *specific gravity* (sg) dengan tipe data nominal:

   Perhitungan *information gain* pada atribut *specific gravity* (sg) adalah dengan menghitung nilai *entropy* kriteria yang ada pada atribut *specific gravity* ditunjukkan pada Tabel 2.3.

Tabel 2.3. Proses mencari *information gain* atribut *specific gravity*

	v	**hasil +**	**hasil -**	**Jumlah**	***entropi* S_i**	**IG Sg**
S1,005	*a*	7	0	7	0	0,449386996
S1,010	*l*	85	0	85	0	
S1,015	*u*	73	0	73	0	
S1,020	*e*	73	80	153	0,9984895	
S1,025	s	12	70	82	0,6006086	
		250	150	400		

(S(Sg) = "1,005", "1,010", "1,015", "1,020", "1,025"

$$entropy(S_{1,020}) = -(\left(\frac{73}{153}\right)\log_2(\frac{73}{153})) - (\left(\frac{80}{153}\right)\log_2(\frac{80}{153})) = 0{,}9984895$$

$$IG(Class, Sg) = 0{,}954434003 - \left(\left(\frac{7}{400}\right)0\right) - \left(\left(\frac{85}{400}\right)0\right) - \left(\left(\frac{73}{400}\right)0\right) - \left(\left(\frac{153}{400}\right)0{,}9984895\right) - \left(\left(\frac{82}{400}\right)0{,}6006086\right) = 0{,}449386996$$

2. *Information gain* untuk atribut *age* (age), tipe data *numeric:*

   Penghitungan entropi *information gain* pada atribut *age* karena memiliki tipe atribut yang *numeric* atau kontinyu, maka perlu dilakukan dengan menggunakan *Entropy-Based Discretization*. Metode ini menggunakan *entropy* sebagai bagian dari proses pemisahan selang data kontinyu. Untuk menemukan nilai pemisah yang terbaik maka harus dihitung nilai *split poin*, *entropy*, dan nilai *gain* antara 2 *sample*. Metode tersebut dapat disebut dengan cara *data transformation* yang dibahas pada sub bab 2.1.4.

   Berikut ini adalah contoh dari analisis data mengenai langkah-langkah dalam penerapan *Entropy-Based Discretization*:

   a. Mengurutkan data subset dari yang terkecil sampai yang terbesar.
   b. Menghitung rataan nilai per dua data yang bersebelahan yang digunakan untuk *split point.* Setiap nilai rata-rata merupakan titik nilai yang mungkin menjadi titik perpecahan (*split point*) untuk memilih titik terbaik, data akan dipecah menurut titik yang diuji. Untuk mencari *split point* bisa dihitung dengan persamaan berikut:

$$split_{point} = \frac{a_i + a_{i+1}}{2}$$

$$split_{point} = \frac{baris\ ke{-}1 + baris\ ke{-}2}{2} = \frac{2+3}{2} = 2{,}5$$

Lakukan perhitungan *split point* hingga baris terakhir.

c. Hitung nilai *information gain* dari kedua sampel (S_a) yang sebelumnya dicari nilai *entropy* terlebih dahulu. Kemudian T (*split point*) yang memiliki nilai informasi terbesar diambil sebagai batas *node*. Contoh tabel perhitungan diskritisasi atribut *age* sebagai mana ditampilkan pada Tabel 2.4.

Tabel 2.4. Perhitungan diskritisasi atribut *age*

		<=			>						
	Split	Ckd	notckd	jumlah	Ckd	notckd	Jumlah	jumlah total	Entropi <=	Entropi>	Gain
43	43	43	65	108	207	85	292	400	0,969	0,870	0,057
43	43	43	65	108	207	85	292	400	0,969	0,870	0,057
44	43,5	43	65	108	207	85	292	400	0,969	0,870	0,057
44	44	44	70	114	206	80	286	400	0,962	0,855	0,069

2) *Particle Swarm Optimization* (PSO)

Particle swarm optimization merupakan teknik optimasi stokastik berbasis populasi yang dikembangkan oleh Eberhart dan Kennedy pada tahun 1995, yang terinspirasi oleh perilaku sosial kawanan burung atau ikan (Chen *et al.*, 2014: 2). *Particle swarm optimization* banyak digunakan untuk memecahkan masalah optimasi serta sebagai masalah seleksi fitur (Utami, 2017: 104). Berbeda dengan teknik optimasi lainnya, setiap partikel dalam PSO berhubungan dengan suatu *velocity*. Partikel tersebut bergerak melalui penelusuran ruang dengan *velocity* yang dinamis disesuaikan menurut perilaku historisnya. Oleh karena itu, partikel-

partikel mempunyai kecenderungan untuk bergerak ke area penelusuran yang lebih baik setelah melewati proses penelusuran (Saharuna dkk., 2012: 22).

PSO memiliki banyak kesamaan dengan teknik-teknik *evolutionary computation* yang lain, seperti *Evolutionary Programming, Genetic Algorithms (GA), Evolutionary Strategies (ES),* dan sebagainya. PSO ataupun GA dimulai dengan suatu populasi yang terdiri dari sejumlah individu (yang menyatakan solusi) yang dibangkitkan secara acak dan selanjutnya melakukan pencarian solusi optimum melalui perbaikan individu untuk sejumlah generasi tertentu. Namun, PSO telah terbukti lebih kompetitif dalam bidang optimasi dibandingkan dengan algoritma genetika (Muslim M A, dkk, 2018).

Berbeda dengan GA, PSO tidak menggunakan operator-operator evolusi seperti rekombinasi (*crossover*) dan mutasi. Setiap partikel pada PSO tidak pernah mati, sedangkan individu pada GA bisa mati dan digantikan dengan individu baru. PSO memiliki *memory* untuk menyimpan solusi terbaik, sedangkan GA tidak punya. Pada PSO, posisi dan kecepatan terbang partikel di *update* pada setiap iterasi sehingga partikel tersebut bisa menghasilkan solusi baru yang lebih baik (Suyanto, 2010: 195).

Proses dasar dari algoritma PSO yaitu sebagai berikut.

1. Inisialisasi: secara acak menghasilkan partikel awal
2. *Fitness*: ukuran *fitness* setiap partikel dalam populasi
3. *Update*: menghitung kecepatan setiap partikel
   $v_{id}^{t} = w * v_{id}^{t-1} + c_1 r_1 \left(p_{id}^{t} - x_{id}^{t}\right) + c_2 r_2 \left(p_{gd}^{t} - x_{id}^{t}\right), d = 1,2, \dots D$
4. Konstruksi: untuk setiap partikel, bergerak ke posisi berikutnya
   $x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t}, d = 1, 2, \dots D$
5. Penghentian: hentikan algoritma jika kriteria penghentian dipenuhi, dan kembali ke langkah 2 (*fitness*). Iterasi dihentikan jika jumlah iterasi mencapai jumlah maksimum yang telah ditentukan dan memperoleh grafik yang konvergen.

2.1.4. *Data Transformation*

Data diubah atau digabung ke dalam format yang sesuai dalam *data mining*. Beberapa metode *data mining* membutuhkan format data yang khusus sebelum bisa diaplikasikan. Sebagai contoh beberapa metode standar seperti analisis *clustering* dan asosiasi hanya bisa menerima input data kategorikal. Analisis *clustering* dan asosiasi tidak bisa menerima data numerik yang bersifat kontinyu sehingga perlu dibagi-bagi menjadi beberapa interval. Proses ini sering disebut transformasi data. Kasus yang diaplikasikan pada *dataset chronic kidney disease*, maka tahap transformasi data diperlukan *dataset* untuk didiskritisasi menggunakan *Entropy-Based Discretization.* Ini dilakukan untuk mentransformasikan data bertipe *numeric* ke dalam pola yang dapat dikenali guna mendapatkan nilai *entropy* yang bertujuan untuk mencari nilai *information gain* dari atribut bertipe *numeric.* Salah satu contoh tahap transformasi data sudah dibahas pada subbab di atas.

1) *Discretization*

Diskritisasi merupakan salah satu proses transformasi data yang berupa atribut kontinyu menjadi atribut kategoris. Transformasi atribut kontinyu ke atribut kategoris terdiri atas dua langkah, yaitu sebagai berikut:

a. Memutuskan berapa jumlah kategori yang harus digunakan.
b. Memetakan nilai atribut kontinyu ke atribut kategoris. Pada langkah pertama, setelah diurutkan, nilai atribut kontinyu kemudian dibagi menjadi M interval dengan titik pemotongan M-1. Pada langkah kedua, semua nilai yang berada dalam setiap interval dipetakan ke nilai kategoris sesuai dengan intervalnya (Prasetyo, 2012: 35)

Ada beberapa algoritma klasifikasi dan *clustering* yang tidak hanya berurusan dengan atribut nominal dan tidak bisa menangani atribut yang diukur pada skala numerik. Oleh karena itu, secara umum *dataset* yang mempunyai atribut numerik

harus didiskritisasi ke sejumlah rentang kecil yang berbeda. Meskipun sebagian besar pohon keputusan dan aturan pembelajaran keputusan dapat menangani atribut numerik, namun beberapa implementasi bekerja jauh lebih lambat ketika terdapat atribut numerik. Oleh karena itu, diterapkannya *discretization* pada atribut numerik yang bersifat *continuous* (Witten *et al.*, 2011: 314).

a) *Entropy-Based Discretization*

Entropy adalah salah satu langkah diskritisasi yang paling umum digunakan yang pertama kali diperkenalkan oleh Claude Shannon. *Entropy-based discretization* merupakan *supervised*, dengan menggunakan teknik *split top-down*. Terdapat data D yang terdiri dari tupel data yang didefinisikan dengan satu set atribut dan atribut kelas-label (Han *et al.*, 2006: 88). Dasar metode untuk e*ntropy-based discretization* untuk sebuah atribut A adalah sebagai berikut:

1. Setiap nilai A dapat dianggap sebagai batas selang potensial atau *split-point* (dilambangkan *split_point*) untuk partisi range A. Artinya, *split-point* untuk A dapat partisi di D menjadi dua himpunan bagian yang memenuhi kondisi A $\leq$ *split_point* dan A > *split_point*, masing-masing, sehingga menciptakan diskritisasi biner.
2. *Entropy-based discretization*, sebagaimana disebutkan di atas menggunakan informasi mengenai label kelas tupel.
3. Proses penentuan *split-point* rekursif diterapkan untuk setiap partisi yang diperoleh, sampai beberapa kriteria berhenti terpenuhi, seperti ketika kebutuhan informasi minimum pada semua calon *split-point* kurang dari ambang kecil, atau ketika jumlah interval lebih besar dari ambang batas, selang max.

Entropy-based discretization dapat mengurangi ukuran data. Tidak seperti metode lain yang disebutkan di sini sejauh ini, diskritisasi berbasis *entropy* menggunakan informasi kelas. Hal ini memungkinkan bahwa batas-batas selang

(*split-point*) didefinisikan terjadi di tempat-tempat yang dapat membantu meningkatkan akurasi klasifikasi.

Berikut ini adalah contoh dari analisis data pada *dataset pima indian diabetes* yang diambil dari UCI *machine learning repository* dengan menerapkan *discretization*. Langkah-langkahnya adalah sebagai berikut:

1. Mengurutkan data subset dari yang terkecil sampai yang terbesar.
2. Menghitung rataan nilai per dua data yang bersebelahan yang digunakan untuk *split point.* Setiap nilai rata-rata merupakan titik nilai yang mungkin menjadi titik perpecahan (*split point*) untuk memilih titik terbaik, data akan dipecah menurut titik yang diuji. Untuk mencari *split point* bisa dihitung dengan rumus

   $$split_{point} = \frac{a_i + a_{i+1}}{2}$$

   $$split_{point} = \frac{baris\ ke{-}1 + baris\ ke{-}2}{2} = \frac{5+5}{2} = 5{,}5$$

   Lakukan perhitungan *split point* hingga baris terakhir,
3. Hitung nilai *information gain* dari kedua sampel (S_a) yang sebelumnya dicari nilai *entropy* terlebih dahulu. Kemudian T(*split point*) yang memiliki nilai informasi terbesar diambil sebagai batas *node*. Contoh tabel perhitungan diskritisasi atribut *pregnant* sebagai mana ditampilkan pada Tabel 2.5.

Tabel 2.5. Proses mencari *information gain* atribut *pregnant* dengan *entropy-based discretization*

Pregnant	***Class***	***Split***	<=			>			**Jumlah total**	**Entropi <=**		
			Ya	**Tidak**	**Jumlah**	**Ya**	**Tidak**	**Jumlah**			**Entropi >**	**Gain**
5	Tidak	5	157	392	549	111	108	219	768	0,863	0,999	0,0307
5	Tidak	5,5	157	392	549	111	108	219	768	0,863	0,999	0,0307
6	Ya	6	173	426	599	95	74	169	768	0,867	0,989	0,0392
6	Tidak	6	173	426	599	95	74	169	768	0,867	0,989	0,0392
6	Tidak	6	173	426	599	95	74	169	768	0,867	0,989	0,0392

Dari perhitungan diatas, maka diperoleh nilai *information gain* terbesar atribut *pregnant* ialah *split point* pada 6 dengan nilai *information gain* 0,039180261. Kemudian hitung juga *entropy, information gain* untuk masing-masing atribut yang bertipe numerik. Tujuan dari diskritisasi atribut adalah untuk mempermudah pengelompokan nilai berdasarkan kriteria yang telah ditetapkan. Hal tersebut juga bertujuan untuk menyederhanakan permasalahan dan meningkatkan akurasi dalam proses pembelajaran.

Dari hasil perhitungan mencari *entropy* pada masing-masing atribut didapatkan *Information Gain* tertinggi yang akan digunakan sebagai batas interval tiap atribut. Ini menyederhanakan data asli dan membuatnya lebih efisien. Pola yang dihasilkan biasanya lebih mudah untuk dipahami (Han *et al*, 2012: 113). Parameter diskritisasi ditunjukan secara lengkap pada Tabel 2.6.

Tabel 2.6. Diskritisasi atribut *dataset Pima Indian Diabetes*

Atribut	Nilai Information Gain	Split	Diskritisasi
Pregnant	0,039180261	6	(≤6), (>6)
Plasma-Glucose	0,13081032	127	(≤127), (>127)
Diastolic BP	0,019661299	68	(≤68), (>68)
Triceps CFT	0,037272124	23	(≤23), (>23)
Serum-Insulin	0,052380722	87	(≤87), (>87)
BMI	0,072547481	27,3	(≤27,3), (>27,3)
DPF	0,020796397	0,527	(≤0,527), (>0,527)
Age	0,07247271	28	(≤28), (>28)
Class	-	-	*Positive* (0), *Negative* (1)

2.2. Teknik *Data mining*

Teknik *data mining* merupakan suatu proses utama yang digunakan saat metode diterapkan untuk menemukan pengetahuan berharga dan tersembunyi dari data. Dengan definisi *data mining*, ada beberapa teknik dan sifat analisa yang dapat digolongkan dalam *data mining* yaitu, sebagai berikut.

2.2.1 *Classification* (Klasifikasi)

Klasifikasi dapat didefinisikan sebagai suatu proses yang melakukan pelatihan/pembelajaran terhadap fungsi target *f* yang memetakan setiap vektor (set fitur) *x* ke dalam satu dari sejumlah label kelas *y* yang tersedia. Di dalam klasifikasi diberikan sejumlah *record* yang dinamakan *training* set, yang terdiri dari beberapa atribut, atribut dapat berupa kontinyu ataupun kategoris, salah satu atribut menunjukkan kelas untuk *record*. Model klasifikasi dapat dilihat pada Gambar 2.2.

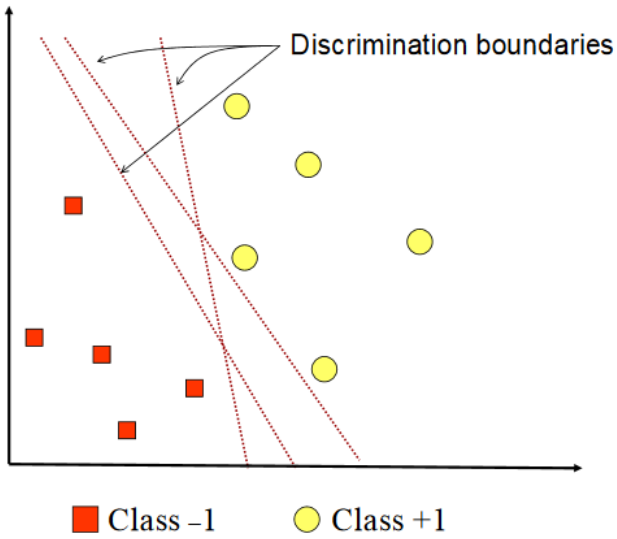
Gambar 2.2. Blok diagram model klasifikasi

Klasifikasi merupakan teknik yang digunakan untuk menemukan model agar dapat menjelaskan atau membedakan konsep atau kelas data, dengan tujuan untuk dapat memperkirakan kelas dari suatu objek yang labelnya tidak diketahui. Metode klasifikasi yang sering digunakan yaitu, *Support Vector Machine, Multilayer Perceptron, Naive bayes, ID3, Ensemble Methode,* dll.

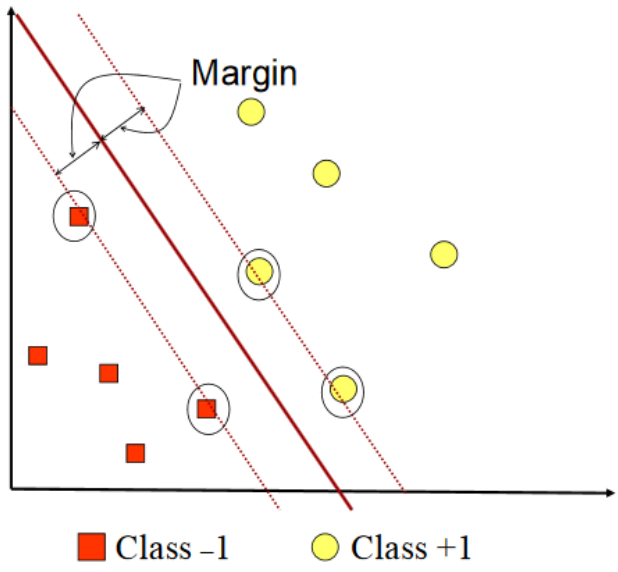
1. *Support Vector Machine (SVM)*

Support Vector Machine (SVM) dikembangkan pada tahun 1992 oleh Vladimir Vapnik bersama dengan kedua rekannya, Bernhard Boser dan Isabelle Guyon. SVM dikembangkan dari teori *structural risk minimalization*. Dengan menggunakan trik kernel untuk memetakan sampel pelatihan dari ruang *input* ke ruang fitur yang berdimensi tinggi (Li *et al.*, 2008: 786). Menurut Han *et al.*, (2012: 408) metode SVM menjadi sebuah metode baru yang menjanjikan untuk pengklasifikasian data, baik data *linear* maupun data *nonlinear*.

Konsep SVM dapat dijelaskan secara sederhana sebagai usaha mencari *hyperplane* terbaik yang berfungsi sebagai pemisah dua buah *class* pada *input space*. Gambar 2.3 memperlihatkan beberapa *pattern* yang merupakan anggota dari dua buah *class*, yaitu: +1 dan –1. *Pattern* yang tergabung pada *class* –1 disimbolkan dengan warna merah (kotak), sedangkan *pattern* pada *class* +1, disimbolkan dengan warna kuning (lingkaran). Problem klasifikasi dapat diterjemahkan dengan usaha menemukan garis *(hyperplane)* yang memisahkan antara kedua kelompok tersebut. Berbagai alternatif garis pemisah *(discrimination boundaries)* ditunjukkan pada Gambar 2.3. *Hyperplane* pemisah terbaik antara kedua *class* dapat ditemukan dengan mengukur *margin hyperplane* tersebut dan mencari titik maksimalnya. *Margin* adalah jarak antara *hyperplane* tersebut dengan *pattern* terdekat dari masing-masing *class*. *Pattern* yang paling dekat ini disebut sebagai *support vector*. Garis solid pada Gambar 2.4 menunjukkan *hyperplane* yang terbaik, yaitu yang terletak tepat pada tengah-tengah kedua *class*, sedangkan titik merah dan kuning yang berada dalam lingkaran hitam adalah *support vector*. Usaha untuk mencari lokasi *hyperplane* ini merupakan inti dari proses pembelajaran pada SVM (Nugroho *et al.*, 2003).



Gambar 2.3. *Discrimination boundaries*



Gambar 2.4. *Hyperplane*

Langkah awal suatu algoritma SVM adalah pendefinisian persamaan suatu *hyperplane* pemisah yang dituliskan dengan persaman berikut.

$$w.X + b = 0$$

Keterangan:

w : bobot vektor, $w = \{w_1, w_2, \dots, w_n\}$;

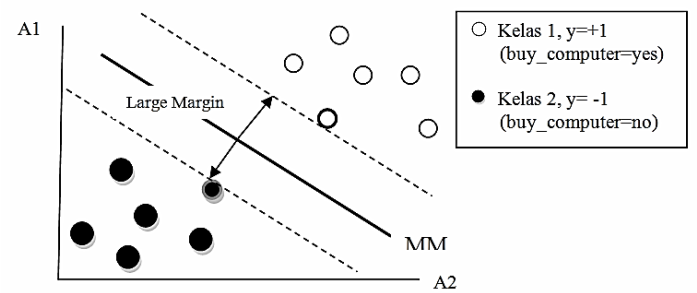
b : skalar yang disebut dengan bias. Jika berdasarkan pada atribut A_1, A_2 dengan permisalan tupel pelatihan

X : $(x_{1,}x_2)$, x_1 dan x_2 merupakan nilai dari atribut A_1 dan A_2, dan jika b dianggap sebagai suatu bobot tambahan w_0,

maka persamaan suatu *hyperplane* pemisah dapat ditulis ulang seperti pada persamaan berikut.

$$w_0 + w_1x_1 + w_2x_2 = 0$$

Setelah persamaan dapat didefinisikan, nilai x_1 dan x_2 dapat dimasukkan ke dalam persamaan untuk mencari bobot w_1, w_2, dan w_0 atau b. Grafik pemisahan dua kelas data dengan margin maksimum dapat dilihat pada Gambar 2.5.



Gambar 2.5. Pemisahan dua kelas data dengan margin maksimum

Pada gambar di atas, dijelaskan bahwa SVM menemukan *hyperplane* pemisah maksimum, yaitu *hyperplane* yang mempunyai jarak maksimum antara tupel pelatihan terdekat. *Support vector* ditunjukkan dengan batasan tebal pada titik

tupel. Dengan demikian, setiap titik yang letaknya di atas *hyperplane* pemisah memenuhi persamaan berikut.

$$w_0 + w_1 x_1 + w_2 x_2 > 0$$

Sedangkan, titik yang letaknya di bawah *hyperplane* pemisah memenuhi rumus seperti pada persamaan berikut.

$$w_0 + w_1 x_1 + w_2 x_2 < 0$$

Jika dilihat dari dua kondisi di atas, maka didapatkan dua persamaan *hyperplane*, seperti pada persamaan yang ada di bawah ini.

$H_1: w_0 + w_1 x_1 + w_2 x_2 \geq 0$ untuk y_i=+1
$H_2: w_0 + w_1 x_1 + w_2 x_2 \leq 0$ untuk y_i=-1

Dengan demikian, setiap *tuple* yang berada di atas H_1 memikiki kelas +1, dan setiap *tuple* yang berada di bawah H_2 memiliki kelas -1. Perumusan model SVM menggunakan trik matematika yaitu *lagrangian formulation*. Berdasarkan *lagrangian formulation, Maksimum Margin Hyperplane* (MMH) dapat ditulis ulang sebagai suatu batas keputusan (*decision boundary*) dituliskan dengan persamaan berikut.

$$d(X^T) = \sum_{i=1}^{l} y_i a_i \; X_i X^T + b_0$$

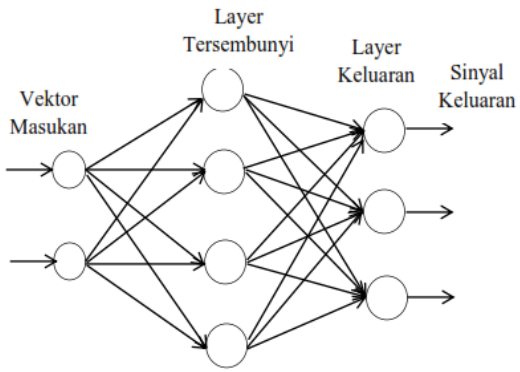
y_i adalah label kelas dari *support vector* X_i, X^T merupakan suatu *tuple test*. a_i dan b_o adalah parameter numerik yang ditentukan secara otomatis oleh optimalisasi algoritma SVM dan l adalah jumlah *support vector*.

2. *Multi Layer Perceptron*

Multi Layer Perceptron (MLP) merupakan turunan algoritma *Artificial Neural Network* (ANN) dari *perceptron*, berupa ANN *feed forward* dengan satu atau lebih dari *hidden layer*. Biasanya, jaringan terdiri dari satu *layer* masukan, setidaknya satu *layer neuron* komputasi di tengah (*hidden layer*) dan sebuah

layer neuron komputasi keluaran. Sinyal masukan dipropagasikan dengan arah maju pada *layer* per *layer*.

Sebenarnya ada satu *layer* lagi dalam MLP, yaitu *layer* masukan berupa vektor masukan. Akan tetapi, karena dalam *layer* ini tidak ada komputasi, yang dilakukan hanya meneruskan sinyal/vektor masukan yang diterima ke *layer* di depannya. Contoh arsitektur MLP diberikan pada Gambar 2.6. Pada gambar tersebut ada satu *hidden layer* dengan empat *neuron*, dan satu *layer* keluaran dengan tiga *neuron*.



Gambar 2.6. Arsitektur *Multilayer Perceptron*

Setiap layer dalam MLP masing-masing mempunyai fungsi khusus. Vektor masukan berfungsi menerima sinyal/vektor masukan dari luar dan mendistribusikannya ke semua *neuron* dalam *hidden layer*. *Layer* keluaran berfungsi untuk menerima sinyal keluaran atau dengan kata lain stimulus pola dari *hidden layer* dan memunculkan sinyal/nilai/kelas keluaran dari keseluruhan jaringan.

Tahapan pada algoritma *multilayer perceptron*, yaitu sebagai berikut.

Langkah 1: Inisialisasi

Inisialisasi semua bobot pada *hidden layer* keluaran, lalu menetapkan fungsi aktivasi yang digunakan untuk setiap *layer*, kemudian tetapkan laju

pembelajaran. Untuk inisialisasi semua bobot bisa menggunakan bilangan acak dalam jangkauan [-1,1] atau [-0.5,0.5] selain itu juga bisa menggunakan inisialisasi distribusi seragam dalam jangkauan kecil (tidak ada aturan baku mengenai interval tersebut). Berikut adalah contoh inisialisasi distribusi seragam.

$\left(-\frac{2.4}{F_i}, +\frac{2.4}{F_i}\right)$, dengan F_i adalah jumlah *neuron* masukan 1 dalam ANN.

Langkah 2: Aktivasi

Mengaktivasi jaringan dengan menerapkan inputan $x_1(p)$, $x_2(p)$, ..., $x_n(p)$ dan keluaran yang diharapkan $y_{d1}(p)$, $y_{d2}(p)$, ..., $y_{dn}(p)$

a) Menghitung keluaran yang didapatkan dari *neuron* dalam *hidden layer*, dengan persamaan berikut:

$$v_j(p) = \sum_{i=1}^{r} \delta_k(p).w_{jk}(p)$$

$$y_j(p) = \frac{1}{1 + e^{-v_j(p)}}$$

r adalah jumlah fitur data inputan pada *neuron j* dalam *hidden layer*.

b) Menghitung keluaran yang didapatkan dari *neuron* dalam *layer* keluaran, menggunakan persamaan berikut ini:

$$v_k(p) = \sum_{j=1}^{m} x_j(p).w_{jk}(p)$$

$$y_k(p) = \frac{1}{1 + e^{-v_k(p)}}$$

m adalah jumlah masukan pada *neuron k* dalam *layer* keluaran.

Langkah 3: Perbarui bobot

Bobot diperbarui pada saat *error* dirambatkan balik dalam ANN, *error* dikembalikan sesuai dengan arah keluarnya sinyal keluaran.

a) Menghitung *gradien error* untuk *neuron* dalam *layer* keluaran, menggunakan persamaan berikut:

$e_k(p) = y_{dk}(p) - y_k(p)$

$\delta_k(p) = y_k(p)x(1 - y_k(p))xe_k(p)$

Kemudian menghitung koreksi bobot:

$\Delta w_{jk}(p) = \pi x y_j(p) + \delta_k(p)$

Memperbarui bobot pada *neuron layer* keluaran:

$w_{jk}(p+1) = w_{jk}(p) + \Delta w_{jk}(p)$

b) Hitung *gradien error* untuk *neuron* dalam *hidden layer* menggunakan persamaan berikut ini:

$$\delta_j(p) = y_j(p)x[1 - y_j(p)] + \sum_{k=1}^{i} \delta_k(p).w_{jk}(p)$$

Menghitung koreksi bobot:

$\Delta w_{ij}(p) = \pi x x_j(p) + \delta_j(p)$

Memperbarui bobot pada *neuron layer* keluaran:

$$w_{ij}(p+1) = w_{ij}(p) + \Delta w_{ij}(p)$$

3. *Naive Bayes*

a) Teorema *Bayes*

Bayes merupakan teknik prediksi berbasis probalistik sederhana yang berdasar pada penerapan teorema *bayes* atau aturan *bayes* dengan asumsi independensi (ketidaktergantungan) yang kuat (*naïve*). Dengan kata lain, *Naïve Bayes* adalah model fitur independen (Prasetyo, 2012: 59). Dalam *Bayes* (terutama *Naïve*

Bayes), maksud independensi yang kuat pada fitur adalah bahwa sebuah fitur pada sebuah data tidak berkaitan dengan ada atau tidaknya fitur lain dalam data yang sama. Prediksi *Bayes* didasarkan pada teorema *Bayes* dengan formula umum dengan persamaan berikut.

$$P(H|E) = \frac{P(E|H)\,P(H)}{P(E)}$$

Penjelasan formula diatas sebagai berikut:

Parameter	Keterangan
P(H|E)	Probabilitas bebas bersyarat (*conditional probability)* suatu hipotesis H jika diberikan bukti *(Evidence)* E terjadi.
P(E|H)	Probabilitas sebuah bukti E terjadi akan mempengaruhi hipotesis H
P(H)	Probabilitas awal (priori) hipotesis H terjadi tanpa memandang bukti apapun
P(E)	Probabilitas awal (priori) bukti E terjadi tanpa memandang hipotesis/bukti yang lain

Ide dasar dari aturan *Bayes* adalah bahwa hasil dari hipotesis atas peristiwa (H) dapat diperkirakan berdasarkan pada beberapa bukti (E) yang diamati. Ada beberapa hal penting dalam aturan *Bayes* tersebut yaitu,

a) Sebuah probabilitas awal/prior H atau P(H) adalah probabilitas dari suatu hipotesis sebelum bukti diamati.
b) Sebuah probabilitas akhir H atau P(H|E) adalah probabilitas dari suatu hipotesis setelah bukti diamati.

Contoh kasus:

Dalam suatu peramalan cuaca untuk memperkirakan terjadinya hujan, ada faktor yang mempengaruhi terjadinya hujan, yaitu mendung. Jika diterapkan dalam *Naïve Bayes*, probabilitas terjadinya hujan, jika bukti mendung sudah diamati, dinyatakan dengan

$$P(\text{Hujan}|\text{Mendung}) = \frac{P(\text{Mendung}|\text{Hujan}) \times P(\text{Hujan})}{P(\text{Mendung})}$$

P(Hujan|Mendung) adalah nilai probabilitas hipotesis hujan terjadi jika bukti mendung sudah diamati. P(Mendung|Hujan) adalah nilai probabilitas bahwa mendung yang diamati akan memengaruhi terjadinya hujan. P(Hujan) adalah probabilitas awal hujan tanpa memandang bukti apapun, sementara P(Mendung) adalah probabilitas terjadinya mendung.

Contoh tersebut dapat dikembangkan dengan menambahkan beberapa observasi yang lain sebagai bukti. Semakin banyak bukti yang dilibatkan, maka semakin baik hasil prediksi yang diberikan. Namun, tentu saja bukti tersebut harus benar-benar berkaitan dan memberi pengaruh pada hipotesis. Dengan kata lain, penambahan bukti yang diamati tidak sembarangan. Bukti gempa bumi tentu saja tidak berkaitan dengan hujan sehingga penambahan bukti gempa bumi dalam prediksi cuaca akan memberikan hasil yang salah. Walaupun ada bukti lain yang mempengaruhi cuaca seperti suhu udara, tetap saja ada nilai probabilitas P(Suhu) yang harus dinilai secara independen dalam teorema *bayes*, yang sulit dilakukan karena suhu udara juga dipengaruhi oleh faktor lain seperti cuaca kemarin, mendung, polusi, dan sebagainya. Jadi, penilaian probabilitas tersebut tidak memandang faktor lain. Inilah sebabnya disebut *Naïve Bayes* (Bayes Naif).

Teorema *bayes* juga bisa menangani beberapa bukti, jika ditambahkan bukti suhu udara dan angin, bentuknya berubah menjadi;

$$P(\text{Hujan}|\text{Mendung}, \text{Suhu}, \text{Angin})$$
$$= \frac{P(\text{Mendung}|\text{Hujan}) \times P(\text{Suhu}|\text{Hujan}) \times P(\text{Hujan}|\text{Hujan}) \times P(\text{Hujan})}{P(\text{Mendung}) \times P(\text{Suhu}) \times P(\text{Angin})}$$

b) *Naïve Bayes* untuk Klasifikasi

Prasetyo (2012: 61) menjelaskan kaitan antara *Naïve Bayes* dengan klasifikasi, korelasi hipotesis dan bukti klasifikasi adalah bahwa hipotesis dalam teorema *bayes* merupakan label kelas yang menjadi target pemetaan dalam klasifikasi,

sedangkan bukti merupakan fitur-fitur yang menjadikan masukan dalam model klasifikasi. Jika X adalah vektor masukkan yang berisi fitur dan Y adalah label kelas, *Naïve Bayes* dituliskan dengan P(X|Y). Notasi tersebut berarti probabilitas label kelas Y didapatkan setelah fitur-fitur X diamati. Notasi ini disebut juga probabilitas akhir (*posterior probability*) untuk Y, sedangkan P(Y) disebut probabilitas awal (*prior probability*) Y.

Selama proses pelatihan harus dilakukan pembelajaran probabilitas akhir P(Y|X) pada model untuk setiap kombinasi X dan Y berdasarkan informasi yang didapat dari data latih. Dengan membangun model tersebut, suatu data uji X' dapat diklasifikasikan dengan mencari nilai Y' dengan memaksimalkan P(Y'|X') yang didapat. Formulasi *Naïve Bayes* untuk klasifikasi yaitu pada persamaan berikut.

$$\mathrm{P(Y|X)} = \frac{\mathrm{P(Y)} \prod_{i=1}^{q} \mathrm{P(X}_i\mathrm{|Y)}}{\mathrm{P(X)}}$$

P(X|Y) adalah probabilitas data dengan vektor X pada kelas Y. P(Y) adalah probabilitas awal kelas Y. $\prod_{i=1}^{q} \mathrm{P(X}_i\mathrm{|Y)}$ adalah probabilitas independen kelas Y dari semua fitur dalam vetor X. Nilai P(X) selalu tepat sehingga dalam perhitungan prediksi nantinya kita tinggal menghitung bagian $\mathrm{P(Y)} \prod_{i=1}^{q} \mathrm{P(X}_i\mathrm{|Y)}$ dengan memilih yang terbesar sebagai kelas yang dipilih sebagai hasil prediksi. Sementara probabilitas independen $\prod_{i=1}^{q} \mathrm{P(X}_i\mathrm{|Y)}$ tersebut merupakan pengaruh semua fitur dari data terhadap setiap kelas Y, yang dinotasikan dengan Persamaan berikut.

$$\mathrm{P(X|Y = y)} = \prod_{i=1}^{q} \mathrm{P(X}_i\mathrm{|Y = y)}$$

Setiap set fitur X= $\{\mathrm{x}_1, \mathrm{x}_2, \mathrm{x}_3, \ldots, \mathrm{x}_\mathrm{q}\}$ terdiri atas q atribut (q dimensi).

Umumnya, *Bayes* mudah dihitung untuk fitur bertipe ketegoris seperti pada kasus klasifikasi hewan dengan fitur "penutup kulit" dengan nilai {bulu, rambut, cangkang} atau kasus fitur "jenis kelamin" dengan nilai {pria, wanita}. Namun

untuk fitur dengan tipe numerik (kontinyu) ada perlakuan khusus sebelum dimasukan dalam *Naïve Bayes*. Caranya yaitu:

a) Melakukan diskretisasi pada setia fitur kontinyu dan mengganti nilai fitur kontinyu tersebut dengan nilai interval diskret. Pendekatan ini dilakukan dengan mentransformasikan fitur kontinyu ke dalam fitur ordinal.
b) Mengasumsi bentuk tertentu dari distribusi probabilitas untuk fitur kontinyu dan memperkirakan parameter distribusi dengan data pelatihan. Distribusi Gaussian biasanya dipilih untuk merepresentasikan probabilitas bersyarat dari fitur kontinyu pada sebuah kelas $P(X_i|Y)$, sedangkan distribusi Gaussian dikarakteristikkan dengan dua parameter, yaitu: *mean* (μ) dan $varian$ (σ^2). Untuk setiap kelas Y_j, probabilitas bersyarat kelas Y_j untuk fitur X_i adalah seperti pada persamaan berikut.

$$P(X_i = x_i | Y_j = y_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}}} \exp^{-\frac{(x_i - \mu_{ij})^2}{2\sigma_{ij}^2}}$$

Parameter μ_{ij} bisa didapat dari *mean* sampel X_i $(\bar{x})$ dari semua data latih yang menjadi milik kelas Y_j, sedangkan σ_{ij}^2 dapat diperkirakan dari *varian* sampel (s^2) dari data latih.

Contoh kasus:

Tabel 2.7. Data latih klasifikasi hewan

Nama Hewan	Penutup Kulit	Melahirkan	Berat	Kelas
Ular	Sisik	Ya	10	Reptil
Tikus	Bulu	Ya	0.8	Mamalia
Kambing	Rambut	Ya	21	Mamalia
Sapi	Rambut	Ya	120	Mamalia
Kadal	Sisik	Tidak	0.4	Reptil
Kucing	Rambut	Ya	1.5	Mamalia

Bekicot	Cangkang	Tidak	0.3	Reptil
Harimau	Rambut	Ya	43	Mamalia
Rusa	Rambut	Ya	45	Mamalia
Kura-kura	Cangkang	Tidak	7	Reptil

Tabel 2.7 adalah data latih untuk klasifikasi jenis hewan. Fitur yang menggunakan tipe numerik adalah berat. *Mean* dan *varian* untuk masing-masing kelas mamalia dan reptil dihitung sebagai berikut:

$$\bar{x}_{mamalia} = \frac{0.8 + 21 + 120 + 1.5 + 43 + 45}{6} = \frac{231.3}{6} = 38.55$$

$$\bar{x}_{reptil} = \frac{10 + 0.4 + 0.3 + 7}{4} = \frac{17.7}{4} = 4.425$$

$$s^2_{mamalia} = \frac{(0.8 - 38.55)^2 + (21 - 38.55)^2 + (120 - 38.55)^2 + (1.5 - 38.55)^2 + (43 - 38.55)^2 + (45 - 38.55)^2}{6 - 1}$$

$$s^2_{mamalia} = \frac{9801.275}{5} = 1960.255$$

$$s_{mamalia} = \sqrt{1960.255} = 44.275$$

$$s^2_{reptil} = \frac{(10 - 4.425)^2 + (0.4 - 4.425)^2 + (0.3 - 4.425)^2 + (7 - 4.425)^2}{4 - 1}$$

$$s^2_{reptil} = \frac{70.9275}{3} = 23.6425$$

$$s_{mamalia} = \sqrt{23.6425} = 4.8625$$

Untuk memperjelas penggunaan fitur bertipe numerik, contoh data uji berupa hewan musang dengan nilai fitur penutup kulit = rambut, melahirkan = iya, berat = 15. Masuk ke kelas manakah hewan musang tersebut?

Untuk menyelesaikannya, pertama kita harus mengetahui nilai probabilitas setiap fitur pada setiap kelasnya atau $P(X_i|Y_j)$, ringkasnya dapat dilihat pada Tabel 2.8. Selanjutnya, untuk data uji di atas, hitung nilai probabilitas untuk fitur dengan tipe numerik, yaitu berat.

$$P(\text{Berat} = 15|\text{Mamalia}) = \frac{1}{\sqrt{2\pi 44.275}} exp^{-\frac{(15-38.55)^2}{2x1960.255}} = 0.0104$$

$$\text{P(Berat} = 15|\text{Reptil)} = \frac{1}{\sqrt{2\pi 4.8625}} exp^{-\frac{(15-4.425)^2}{2x23.6425}} = 0.8733$$

Tabel 2.8. Probabilitas fitur dan kelas

Penutup kulit		**Melahirkan**	
Mamalia	Reptil	Mamalia	Reptil
Sisik = 0 **Bulu = 1** **Rambut = 5** **Cangkang = 0**	Sisik = 2 Bulu = 0 Rambut = 0 Cangkang = 2	Ya = 6 Tidak = 0	Ya = 1 Tidak = 0
Berat		**Kelas**	
Mamalia	Reptil	Mamalia	Reptil
$\bar{x}_{mamalia} = 38.55$ $s^2_{mamalia} = 1960.255$ $s_{mamalia} = 44.275$	$\bar{x}_{reptil} = 4.425$ $s^2_{reptil} = 23.6425$ $s_{reptil} = 4.8624$	Mamalia = 6 P(Mamalia) = $\frac{6}{10}$ = 0.6	Reptil = 4 P(Reptil) = $\frac{4}{10}$ = 0.4

Berulah kemudian menghitung probabilitas akhir untuk setiap kelas:

$$\text{P(X|Mamalia)} = \text{P(Kulit} = \text{Rambut|Mamalia)xP(Lahir} = \text{Ya|Mamalia)xP(Berat} = 15|\text{Mamalia)}$$
$$= \frac{5}{6}\ x\ 1\ x\ 0.0104 = 0.0087$$

$$\text{P(X|Reptil)} = \text{P(Kulit} = \text{Rambut|Reptil)xP(Lahir} = \text{Ya|Reptil)xP(Berat} = 15|\text{Reptil)}$$
$$= 0 \text{ x } 0.25 \text{ x } 0.0104 = 0$$

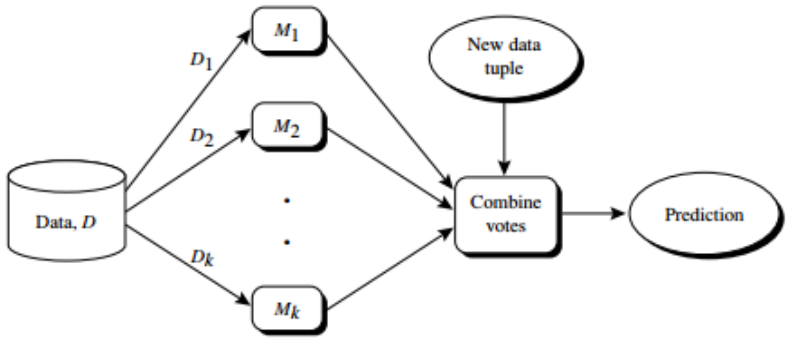
Selanjutnya, nilai tersebut dimasukan untuk mendapatkan probabilitas akhir.

$$\text{P(Mamalia|X)} = \alpha \text{ x } 0.6 \text{ x } 0.0087 = 0.0052\alpha$$
$$\text{P(Reptil|X)} = \alpha \text{ x } 0 \text{ x } 0.4 = 0$$

$\alpha = 1/\text{P(X)}$ nilainya konstan sehingga tidak perlu diketahui karena yang terbesar dari dua kelas tersebut tidak dapat dipengaruhi P(X). Karena nilai probabilitas akhir terbesar ada di kelas mamalia, data uji musang diprediksi sebagai kelas mamalia.

4. *Ensemble Methode*

Ensemble method merupakan salah satu metode yang digunakan untuk meningkatkan akurasi algoritma klasifikasi dengan membangun beberapa *classifier* dari data *training* kemudian pada saat proses klasifikasi metode ini menggunakan *voting/aggregating* dari *classifier-classifer* yang digunakan (Tan, *et al*., 2006: 276). Han *et al.*, (2012: 378) dalam bukunya, menjelaskan bahwa *bagging*, *boosting*, dan *random forest* adalah contoh metode *ensemble*. Metode *ensemble* yaitu menggabungkan serangkaian k model pembelajaran (dasar klasifikasi), seperti: M_1, M_2, ... , M_k. Diberikan sebuah *dataset* D yang digunakan untuk membuat k *data training*, yaitu D_1, D_2, ... , D_k, di mana D_i $(1 \leq i \leq k - 1)$ digunakan untuk menghasilkan *classifier* M_i. Ilustrasi metode *ensemble* ditunjukkan pada Gambar 2.5.



Gambar 2.5. *Ensemble Method*

Teknik ensemble merupakan teknik yang sukses untuk menangani *dataset* yang tidak seimbang meskipun tidak secara khusus dirancang untuk masalah data yang tidak seimbang. Teknik *bagging* merupakan salah satu teknik *ensemble*. Teknik *ensemble* pada klasifikasi digunakan untuk memisahkan data *training* ke dalam beberapa data *training* baru dengan *random sampling* dan membangun model berbasis data *training* baru (Wahono & Suryana, 2013: 157).

a) *Bootstrap Aggregating* (*Bagging*)

Bagging ditemukan oleh Breiman (1996) yang merupakan kepanjangan dari "*bootstrap aggregating*" dan juga salah satu teknik yang dapat digunakan pada beberapa metode klasifikasi dan regresi untuk mereduksi variansi dari suatu prediktor, dan dengan demikian dapat memperbaiki proses prediksi (Sutton, 2005: 176). Han *et al.*, (2012: 379) dalam bukunya menyatakan bahwa *bagging* adalah salah satu teknik dari *ensemble method* dengan cara memanipulasi data *training*, data *training* diduplikasi sebanyak *d* kali dengan pengembalian (*sampling with replacement*), yang akan menghasilkan sebanyak *d* data *training* yang baru, kemudian dari *d* data *training* tersebut akan dibangun *classifier-classifier* yang disebut sebagai *bagged classifier*. Karena data pada teknik *bagging* dilakukan *sampling with replacement*, ukuran data *bagging* sama dengan data aslinya, tetapi distribusi data dari tiap data *bagging* berbeda, beberapa data dari data *training* bisa saja muncul beberapa kali atau mungkin tidak muncul sama sekali (Han & Kamber, 2001: 390). Hal inilah yang menjadi kunci kenapa *bagging* bisa meningkatkan akurasi karena dengan *sampling with replacement* dapat memperkecil *variance* dari *dataset* (Tan, et al., 2006: 283).

Berdasarkan namanya, maka ada dua tahapan utama dalam analisis ini, yaitu *bootstrap* yang tidak lain adalah pengambilan sampel dari data *learning* yang dimiliki (*resampling*) dan *aggregating* yaitu menggabungkan banyak nilai dugaan menjadi satu nilai dugaan. Pada kasus klasifikasi, mengambil suara terbanyak (*majority vote*) merupakan salah satu proses menggabungkan nilai dugaan dari beberapa pohon menjadi satu dugaan akhir, sedangkan untuk kasus regresi menggunakan rata-rata. Dengan demikian proses pembuatan dugaan secara *bagging* menggunakan pohon menurut Sutton (2005: 178) adalah sebagai berikut:

1. Tahapan *bootstrap*
   a. Tarik sampel acak dengan pemulihan berukuran n dari gugus data *learning*.

b. Susun pohon terbaik berdasarkan data tersebut.
c. Ulangi langkah a dan b sebanyak *B* kali sehingga diperoleh *B* buah pohon klasifikasi.

2. Tahapan *aggregating*

   Lakukan pendugaan gabungan berdasarkan *B* buah pohon klasifikasi tersebut menggunakan aturan *majority vote* (suara terbanyak).

Berdasarkan namanya, maka ada dua tahapan utama dalam analisis ini, yaitu *bootstrap* yang tidak lain adalah pengambilan sampel dari data *learning* yang dimiliki (*resampling*) dan *aggregating* yaitu menggabungkan banyak nilai dugaan menjadi satu nilai dugaan. Pada kasus klasifikasi, mengambil suara terbanyak (*majority vote*) merupakan salah satu proses menggabungkan nilai dugaan dari beberapa pohon menjadi satu dugaan akhir, sedangkan untuk kasus regresi menggunakan rata-rata. De demikian proses pembuatan dugaan secara *bagging* menggunakan pohon adalah sebagai berikut;

1. Mempersiapkan *dataset*. *Dataset* bisa diambil dari *UCI repository of machine learning*.
2. Melakukan pembagian *dataset* menjadi data latih dan data uji menggunakan *k-fold cross validation.*
3. Kemudian data *training* dibagi lagi oleh *bagging* menjadi *sub dataset (bootstrap)* sebanyak 10 perulangan *(iterations)*.
4. Masing-masing *bootstrap* data kemudian diproses dengan metode klasifikasi misalnya metode C4.5.
5. Lakukan proses C4.5 pada *sub dataset (bootstrap)* sebanyak 10 perulangan (*iterations*) sehingga diperoleh 10 buah pohon/model acak.
6. Lakukan pendugaan gabungan (*aggregating*) berdasarkan *k* buah pohon tersebut. Untuk memilih kasus klasifikasi menggunakan konsep *majority vote* (suara terbanyak).

7. Hasil pendugaan gabungan (*aggregating*) berdasarkan *majority vote* (suara terbanyak) digunakan untuk menguji ketepatan klasifikasi menggunakan *confusion matrix*.

b) *Adaptive Boosting* (Adaboost)

Algoritma *Adaboost* pertama kali diperkenalkan pada tahun 1995 oleh Freund dan Schapire, telah banyak memecahkan berbagai masalah praktis dari algoritma sebelumnya. *Boosting* merupakan salah satu contoh metode *ensemble* (*ensemble methods*) yang menggabungkan suatu urutan model pembelajaran *k* (atau disebut juga *classifier* dasar), M_1, M_2, ..., M_k, dengan tujuan menciptakan model klasifikasi gabungan yang lebih baik. *Ensemble methods* ini mengembalikan hasil prediksi kelas berdasarkan penilaian dari *classifier* dasarnya (Han, et al., 2012). Adapun algoritma *Adaboost* memiliki pondasi teori yang solid, prediksi yang sangat akurat, tingkat kesederhanaan yang tinggi (cukup hanya dengan 10 baris kode), dan penggunaannya yang luas dan sukses (Wu, et al., 2007).

Adapun penggambaran kerja dari algoritma *Adaboost*, yaitu sebagai berikut: misalkan $\mathcal{X}$ dinotasikan sebagai *instance* dan $\mathcal{Y}$ sebagai *set* label kelas. Diasumsikan $\mathcal{Y} = \{-1, +1\}$. Kemudian diberikan algoritma pembelajaran dasar atau lemah (*weak or base learning algorithm*) dan sebuah *training set* $\{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)$ di mana $x_i \in \mathcal{X}$ dan $y_i \in \mathcal{Y}$. Kemudian algoritma *Adaboost* bekerja seperti berikut: pertama tiap contoh *training* (*training example*) (x_i, y_i) $(i \in \{1, \dots . m\})$ diberikan bobot yang sama. Denotasikan distribusi bobot pada putaran pembelajaran (*learning round*) ke-*t* sebagai D_t. Dari *training set* dan D_t algoritma *Adaboost* ini menghasilkan suatu *weak* atau *base learner* $h_t : \mathcal{X} \to \mathcal{Y}$ dengan memanggil algoritma pembelajaran dasarnya. Kemudian contoh *training* tersebut digunakan untuk menguji h_t, dan bobot-bobot dari contoh klasifikasi yang salah akan meningkat. Dengan demikian, suatu distribusi bobot yang telah diperbarui D_{t+1} diperoleh. Dari *training set* dan D_{t+1} *Adaboost* menghasilkan *weak learner* lain dengan memanggil algoritma pembelajaran

dasarnya lagi. Proses tersebut diulang sebanyak *T* putaran, dan model akhir diperoleh dengan bobot suara terbanyak (*weighted majority voting*) dari kumpulan *T weak learner*, di mana bobot dari *learner* tersebut ditentukan selama proses pelatihan atau *training*. (Wu, et al., 2007).

Pengembangan metode *Adaboost* memiliki banyak varian turunan antara lain: *Adaboost.M1, Adaboost.M1W, Killback-Leibler Boosting* (*KLBoosting*), dan *Jensen-Shannon Boosting* (*JSBoosting*). *Adaboost.M1* merupakan generalisasi langsung dari *Adaboost* untuk dua kelompok dari masalah multikelas. Sedangkan *Adaboost.M1W* merupakan pengembangan dari *Adaboost.M1* dengan meminimalisasi batas atas pengukuran kinerja yang disebut dengan *guessing error*. Kemudian untuk algoritma *KLBoosting* dan *JSBoosting* digunakan untuk pendeteksian pola atau objek gambar. Implementasi *Adaboost* dalam WEKA menggunakan varian *Adaboost.M1*. Metode *ensemble* atau dikenal dengan metode *classifier combination*, merupakan teknik yang dapat digunakan untuk meningkatkan akurasi dari klasifikasi pada *data mining*. Pada *adaboost* ini *training set* yang digunakan untuk setiap *base classifier* dipilih berdasarkan performasi dari *classifier* sebelumnya. Di dalam *boosting*, sampel yang tidak diprediksikan dengan benar oleh *classifier* di dalam rangkaian akan dipilih lebih sering dibandingkan dengan sampel yang telah diprediksikan dengan benar.

Adapun tahapan-tahapan teknik *ensemble adaboost* dalam pengklasifikasian *dataset* menggunakan algoritma klasifikasi. Misalnya, menggunakan algoritma *Support Vector Machine* (SVM), yaitu sebagai berikut.

1) Mempersiapkan data *training* dan memisalkan $\mathcal{X}$ dinotasikan sebagai *instance* dan $\mathcal{Y}$ sebagai *set* label kelas. Diasumsikan $\mathcal{Y} = \{-1, +1\}$.
2) Memberikan inputan berupa suatu kumpulan *sample* penelitian dengan label pada sebuah *training set* $\{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)$ di mana $x_i \in \mathcal{X}$ dan $y_i \in \mathcal{Y} = \{1, \dots, k\}$ dan suatu algoritma pembelajaran dasar atau lemah (*weak or base learning algorithm*) yaitu *support vector machine*, dengan jumlah iterasi atau perulangan *T*.

3) Kemudian algoritma *Adaboost* menginisialisasikan bobot pada contoh *training* (*training example*) $D_i^1 = 1/m$, (x_i, y_i) $(i \in \{1, \ldots . m\})$
4) Denotasikan distribusi bobot pada putaran pembelajaran (*learning round*) ke-*t* sebagai D_t dan t=1, ..., *T*. Di mana *T* adalah banyak putaran atau iterasi.
5) Dari *training set* dan D_t algoritma *Adaboost* ini menghasilkan suatu *weak* atau *base learner* $h_t : \mathcal{X} \to \mathcal{Y}$ dengan memanggil algoritma pembelajaran dasarnya.
6) Selanjutnya menghitung kesalahan pelatihannya dengan persamaan berikut:

$$\varepsilon_t = \sum_{i=1}^{m} D_i^t , y_i \neq sh_t(x_i)$$

Jika $\varepsilon_t \geq 0{,}5$, maka set *T*= $t - 1$, batalkan *loop* dan langsung menuju *output.*

7) Menghitung nilai bobot untuk *base classifier*.

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1 - \varepsilon_t}{\varepsilon_t}\right)$$

8) *Update* distribusi bobot *sample* pelatihan

$$D_i^{t+1} = \frac{D_i^t \exp\{-\alpha_t y_i h_t(x_i)\}}{Z_t}$$

Lakukan perulangan atau iterasi sebanyak t.

9) Dengan demikian, suatu distribusi bobot yang telah diperbarui D_{t+1} diperoleh. Dari *training set* dan D_{t+1} *Adaboost* menghasilkan *weak learner* lain dengan memanggil algoritma pembelajaran dasarnya lagi. Proses tersebut diulang sebanyak *T* putaran, dan model akhir diperoleh dengan bobot suara terbanyak (*weighted majority voting*) dari kumpulan *T weak learner*, dimana bobot dari *learner* tersebut ditentukan selama proses pelatihan atau *training*. *Output* yang didapatkan dari teknik *ensemble adaboost* ini berupa:

$$f(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

2.2.2 *Clustering* (Klasterisasi)

Teknik klasterisasi berbeda dengan teknik klasifikasi yang kelas data telah ditentukan sebelumnya. *Clustering* merupakan teknik dengan cara mengelompokkan data secara otomatis tanpa diberitahukan label kelasnya. *Clustering* dapat digunakan untuk memberikan label pada kelas data yang belum diketahui, karena *clustering* sering digolongkan sebagai metode *unsupervised learning*. Metode *clustering* yang sering digunakan yaitu, *K-Medoids, K-Means, Fuzzy C-Means, Self-Organizing Map* (SOM), dll.

2.2.3 *Association Rule*

Association rule mining adalah teknik *data mining* yang berfokus untuk menemukan aturan kesamaan dalam suatu kejadian. Contoh aturan asosiasi yang sering dijumpai adalah proses pembelian barang dagangan pada pusat perbelanjaan. Dari data historis diketahui bahwa pembelian susu sebagian besar diikuti dengan pembelian roti dengan demikian pemilik toko dapat meningkatkan keuntungannya dengan cara mengatur letak susu dan roti agar berdekatan serta memberikan diskon yang menarik pembeli. Metode asosiasi yang umum digunakan adalah *FP-Growth, Coefficient of Correlation, Chi Square, A Priori,* dll.

2.2.4 *Regression*

Teknik *regression* tidak jauh berbeda dengan teknik klasifikasi, perbedaanya teknik *regression* tidak bisa mencari pola yang dijelaskan pada tabel. Teknik *regression* digunakan untuk mencari pola dan menentukan nilai numerik. Metode

yang sering digunakan adalah *linear regression, logistic regression, support vector regression,* dll.

2.3. Pengetahuan (*Knowledge*)

Presentasi pengetahuan merupakan proses visualisasi dan penyampaian informasi serta pengetahuan hasil penggalian kepada pengguna. Pengetahuan dari *data mining* adalah bagaimana cara mengatur pembuatan keputusan tertentu atau aksi selanjutnya yang akan dilakukan sesuai dari hasil analisis yang didapat. Proses penyampaian informasi hasil pengolahan *data mining* kadang kala dibutuhkan orang-orang yang tidak memahami *data mining.* Karena informasi hasil pengolahan *data mining* yang dapat dipahami oleh orang lain adalah salah satu tahapan dalam proses *data mining.* Visualisasi dapat digunakan untuk memudahkan komunikasi hasil *data mining* kepada orang lain.

2.4. *Evaluation*

Evaluasi pola merupakan suatu proses identifikasi pola-pola menarik untuk dimasukkan ke dalam *knowledge based* yang ditemukan. Evaluasi adalah proses penilaian pola-pola menarik atau model prediksi apakah memenuhi hipotesa awal atau belum. Apabila hasil yang didapatkan tidak sesuai dengan hipotesa maka dapat diambil beberapa pilihan alternatif untuk memperbaiki proses *data mining* dengan cara mencoba metode *data mining* lain yang sesuai atau menerima hasil ini sebagai suatu hasil diluar dugaan yang mungkin bermanfaat.

Evaluasi tipe *data mining* klasifikasi dilakukan pengujian untuk proses prediksi kebenaran objek. Proses pengujian memanfaatkan *confusion matrix* yang menempatkan kelas prediksi dibagian atas matriks kemudian yang diamati ditempatkan pada bagian kiri matriks. Setiap sel matriks berisi angka yang menampilkan jumlah kasus aktual dari kelas yang diamati.

Nilai kinerja klasifikasi pada *dataset* tidak semua hasilnya bisa mencapai 100% benar sehingga diperlukan proses pengukuran pada model klasifikasi. Model klasifikasi pada umumnya diukur menggunakan *confusion matrix* (Prasetyo, 2012: 47). *Confussion matrix* adalah salah satu cara yang sering digunakan pada proses evaluasi model *data mining* klasifikasi dengan memprediksi kebenaran objek.

Proses evaluasi kinerja model *data mining* yang dibangun diperlukan pengukuran akurasi (*accuracy*) atau tingkat kesalahan *(error rate)*. Pada table 2.9 dijelaskan contoh *confussion matrix* proses klasifikasi dua kelas, dimana nilai dua kelas tersebut yaitu 0 dan 1. Kolom f_{ij} menotasikan jumlah *record* dari kelas *i* yang hasil prediksinya masuk ke kelas *j* pada saat pengujian.

Tabel 2.9. *Confussion matrix* untuk klasifikasi dua kelas

f_{ij}		Kelas hasil prediksi (j)	
		Kelas =1	**Kelas =0**
Kelas asli (*i*)	Kelas = 1	f_{11}	f_{10}
	Kelas = 0	f_{01}	f_{00}

Keterangan:

a. TP (*True Positif*)= $f_{11,}$ *Observed Class* benar dengan hasil *predicted class* benar.
b. TN (*True Negatif*)= f_{10}, *Observed Class* benar dengan hasil *predicted class* salah.
c. FP (*False Positif*)= f_{01}, *Observed Class* salah dengan hasil *predicted class* benar.
d. FN (*False Negatif*)= f_{00}, *Observed Class* salah dengan hasil *predicted class* salah.

Dari nilai $f_{11,}$ $f_{10,}$ $f_{01,}$ dan f_{00} dapat dihitung nilai *accuracy*, *precision*, dan *recall*-nya (Aziz A, 2016: 118). Berdasarkan isi tabel *confussion matrix*, dapat diambil jumlah data dari tiap kelas yang diprediksi secara benar, yaitu (f_{11}+f_{00}), dan data yang diprediksi secara salah, yaitu (f_{10} + f_{01}). Jumlah total *confussion matrix* dapat dirangkum menjadi dua nilai, yaitu akurasi (*accuracy)* dan laju error *(error rate).* Hasil prediksi akurasi diperoleh dari jumlah data yang klasifikasikan secara benar kemudian hasil laju error diperoleh dari jumlah data yang diklasifikasikan secara salah. Oleh karena itu, pengukuran akurasi (*accuracy*) dapat dituliskan dengan persamaan berikut.

$$Accuracy = \frac{jumlah\_prediksi\_yg\_benar}{jumlah\_prediksi\_keseluruhan} = \frac{f11+f00}{f11+f10+f01+f00}$$

Sedangkan untuk menghitung tingkat kesalahan (*error rate*) dapat didefinisikan pada persamaan berikut.

$$Error\ rate = \frac{jumlah\_prediksi\_yg\_salah}{jumlah\_prediksi\_keseluruhan} = \frac{f01+f10}{f11+f10+f01+f00}$$

BAB 3

ALGORITMA C4.5

Algoritma C4.5 diperkenalkan oleh Quinlan sebagai versi perbaikan dari ID3. Dalam ID3, induksi *decision tree* hanya bisa dilakukan pada fitur bertipe kategorikal (nominal/ordinal), sedangkan tipe numerik (internal/rasio) tidak dapat digunakan. Perbaikan yang membedakan algoritma C4.5 dari ID3 adalah dapat menangani fitur dengan tipe numerik, melakukan pemotongan (*pruning*) *decision tree*, dan penurunan (*deriving*) *rule set.* Algoritma C4.5 juga menggunakan kriteria *gain* dalam menentukan fitur yang menjadi pemecah *node* pada pohon yang diinduksi (Prasetyo, 2014: 65).

Dalam algoritma C4.5 untuk membangun pohon keputusan hal pertama yang dilakukan yaitu memilih atribut sebagai akar. Kemudian dibuat cabang untuk tiap-tiap nilai di dalam akar tersebut. Langkah berikutnya yaitu membagi kasus dalam cabang. Kemudian ulangi proses untuk setiap cabang sampai semua kasus pada cabang memiliki kelas yang sama (Kusrini, 2009: 15).

3.1. *Entropy*

Entropi digunakan untuk menentukan yang manakah *node* yang akan menjadi pemecah data latih berikutnya. Nilai entropi yang lebih tinggi akan meningkatkan potensi klasifikasi. Yang perlu diperhatikan adalah jika entropi untuk *node* bernilai 0 berarti semua data vektor berada pada label kelas yang sama dan *node* tersebut menjadi daun yang berisi keputusan (label kelas). Yang juga perlu diperhatikan dalam perhitungan entropi adalah jika salah satu dari elemen w_i jumlahnya 0 maka entropi dipastikan 0 juga. Jika proporsi semua

elemen w_i sama jumlahnya maka dipastikan entropi bernilai 1 (Prasetyo, 2014: 60).

Sementara itu, penghitungan nilai entropi dapat dilihat pada persamaan berikut (Kusrini, 2009: 16):

$$Entropy(S) = \sum_{i=1}^{n} - p_i * log_2 p_i$$

Keterangan:

S : Himpunan Kasus

n : jumlah partisi S

p_i : Proporsi dari S_i terhadap S

Di mana $log_2 pi$ dapat dihitung dengan cara seperti persamaan berikut:

$$log(X) = \frac{ln(X)}{ln(2)}$$

3.2. *Gain Ratio*

Kriteria yang paling banyak digunakan untuk memilih fitur sebagai pemecah dalam algoritma C4.5 adalah *gain ratio*, diformulasikan oleh persamaan berikut (Prasetyo, 2014: 67):

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo(A)}$$

Untuk menghitung *gain* digunakan persamaan berikut (Kusrini, 2009: 16):

$$Gain(S, A) = Entropy(S) - \sum_{i=1}^{n} \frac{|S_i|}{|S|} * Entropy(S_i)$$

Keterangan:

S : Himpunan Kasus

A : Atribut

n : jumlah partisi atribut A

$|S_i|$: jumlah kasus pada partisi ke-i

$|S|$: jumlah kasus dalam S

Sedangkan untuk menghitung $SplitEntropy$ digunakan persamaan berikut:

$$Split_{info}(S) = -\sum_{i=1}^{n} \frac{|Si|}{|S|} * log_2 \frac{|Si|}{|S|}$$

Keterangan:

S : Himpunan Kasus

A : Atribut

n : jumlah partisi atribut A

$|S_i|$: jumlah kasus pada partisi ke-i

$|S|$: jumlah kasus dalam S

3.3. Contoh Kasus

Contoh kasus berikut adalah perhitungan algoritma C4.5 menggunakan *dataset diabetes mellitus* yang didapat dari *UCI of machine learning repository*. Sebelum melakukan proses klasifikasi perlu dilakukan tahap pengolahan data seperti yang dijelaskan di Bab 2. Setelah dilakukan tahap pengolahan data yang diantaranya meliputi tahap data *cleaning* dan data *transformation* sehingga menghasilkan data baru untuk diproses ke tahap selanjutnya, yaitu dengan menerapkan algoritma C4.5. Kelebihan algoritma C4.5 yaitu dapat menangani fitur dengan tipe numerik. Langkah awal setelah menyiapkan data yang telah

diolah, selanjutnya melakukan pembagian data *training* untuk proses pembentukan pohon dan data *testing* untuk mengukur kemampuan dari pohon klasifikasi yang terbentuk. Pembagian data dilakukan dengan menggunakan *k-fold cross validation* dengan *default* nilai $k = 10$.

Berikut ini adalah contoh dari analisis data mengenai langkah-langkah dalam pembentukan *decision tree* dengan menggunakan algoritma C4.5:

a. Menyiapkan data *training*. Data *training* biasanya diambil dari data histori yang pernah terjadi sebelumnya dan sudah dikelompokkan ke dalam kelas tertentu.
b. Menghitung jumlah kasus, jumlah kasus untuk keputusan *tested_positive*, jumlah kasus untuk keputusan *tested_negative* dari tiap atribut yang bisa dilihat pada Tabel 3.1.

Tabel 3.1. Jumlah kasus dari tiap atribut

No	Atribut	Nilai Atribut	Jumlah Kasus Total	Tested_ Positive	Tested_ Negative
1	*Pregnant*	≤6	599	173	426
		>6	169	95	74
2	*Plasma*	≤127	485	94	391
		>127	283	174	109
3	*Pressure*	≤68	283	70	213
		>68	485	198	287
4	*Skin*	≤23	172	27	145
		>23	596	241	355
5	*Insulin*	≤87	118	9	109
		>87	650	259	391
6	*BMI*	≤27,3	183	18	165
		>27,3	585	250	335
7	*Pedigree*	≤0,527	509	148	361
		>0,527	259	120	139
8	*Age*	≤28	367	71	296
		>28	401	197	204

c. Menghitung nilai *entropy, information gain, split info* dan *gain ratio* dari semua kasus dan kasus yang dibagi berdasarkan masing-masing atribut. Setelah itu lakukan perhitungan mencari nilai *gain ratio* untuk masing-masing atribut. Nilai *gain ratio* yang paling tinggi yang akan menjadi akar pertama. Perhitungan selengkapnya akan ditampilkan pada Tabel 3.2.

Tabel 3.2. Hasil perhitungan *gain*

No	Atribut	Jumlah Kasus Total	*Tested_ Positive*	*Tested_ Negative*	*Entropy*	*Info Gain*	*Split Info*	*Gain Ratio*
1	*Class*	768	268	500	0,9331			
2	*Pregnant*					0,0391	0,7602	0,0515
	≤6	599	173	426	0,8671			
	>6	169	95	74	0,9888			
3	*Plasma*					0,1308	0,9495	0,1377
	≤127	485	94	391	0,7093			
	>127	283	174	109	0,9616			
4	*Pressure*					0,0196	0,9495	0,0207
	≤68	283	70	213	0,807			
	>68	485	198	287	0,9755			
5	Skin					0,0372	0,7673	0,0485
	≤23	172	27	145	0,627			
	>23	596	241	355	0,9734			
6	*Insulin*					0,0523	0,6188	0,0846
	≤87	118	9	109	0,3889			
	>87	650	259	391	0,97			
7	*BMI*					0,0725	0,7921	0,0915
	≤27,3	183	18	165	0,4637			
	>27,3	585	250	335	0,9847			
8	*Pedigree*					0,0207	0,9221	0,0225
	≤0,527	509	148	361	0,8697			
	>0,527	259	120	139	0,9961			

9	*Age*					0,0724	0,9985	0,0725
	≤28	367	71	296	0,7086			
	>28	401	197	204	0,9997			

d. Proses mendapatkan nilai *entropy* untuk masing-masing atribut, pertama adalah mencari nilai *entropy* untuk keseluruhan data (*class=”tested_positive”* dan *class=”tested_negative”*) terlebih dahulu.

Jumlah *class* target (*class*) = 2 (“*tested_positive*” dan “*tested_negative*”)

Jumlah kasus *tested_positive* = 268

Jumlah kasus *tested_negative*= 500

Total keseluruhan = 768

Dengan perhitungan yang sama dilakukan terhadap tiap atribut dengan berdasar pengelompokan jumlah kasus pada tiap atribut dan *subset* atribut di dalamnya.

Berdasarkan rumus perhitungan *entropy* yang diperoleh dari:

$$Entropy\ (Class) = I(268{,}500) = -((\frac{268}{768})\log_2(\frac{268}{768})) - ((\frac{500}{768})\log_2(\frac{500}{768}))$$

$$= 0{,}933134317$$

e. Kemudian hitung nilai *gain* pada tiap atribut. Nilai *gain* pada atribut *pregnant* dapat dihitung sebagai berikut:

Gain(Class, Pregnant) = *Entropy(Class)*$-\sum_{i=1}^{n}\frac{|Jumlah|}{|Total|}$ * *Entropy(Pregnant)*

Gain(Class, Pregnant) = 0,9331 – ($\frac{599}{768}$) * 0,8671)−($\frac{169}{768}$) * 0,9888)

Gain(Class, Pregnant) = 0,039180261

f. Untuk menghitung *gain ratio* dibutuhkan nilai *split info*. Untuk menghitung *split info* sebagai berikut:

$$Split_{info}(Pregnant) = -\sum_{i=1}^{n} \frac{|Jumlah|}{|Total|} * log_2 \frac{|Jumlah|}{|Total|}$$

$$Split_{info}(Pregnant) = \left(\left(\frac{599}{768}\right) * \log_2\left(\frac{599}{768}\right)\right) + \left(\left(\frac{599}{768}\right) * \log_2\left(\frac{599}{768}\right)\right)$$

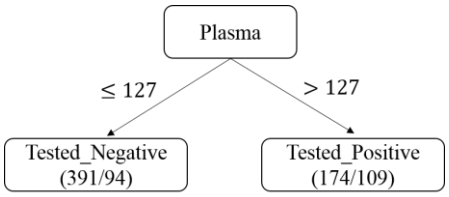
$$Split_{info}(Pregnant) = 0{,}760262594$$

g. Untuk menghitung *gain ratio* digunakan persamaan berikut:

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo(A)}$$

$$GainRatio(A) = \frac{0{,}039180261}{0{,}760262594} = 0{,}051535169$$

Dari hasil Tabel 3.4 dapat diketahui bahwa atribut dengan *gain ratio* tertinggi adalah *plasma*, yaitu sebesar 0,1377. Dengan demikian, *plasma* dapat dijadikan *node* akar. Dari hasil tersebut dapat digambarkan pohon keputusan sementara seperti Gambar 3.1.



Gambar 3.1. Gambar *tree* hasil perhitungan *node* 1

h. Menghitung jumlah kasus, jumlah kasus untuk keputusan *tested_positive*, jumlah kasus untuk keputusan *tested_negative*, *entropy, information gain, split info* dan *gain ratio* dari tiap atribut dengan menghilangkan atribut yang

telah dipilih. Setelah itu lakukan perhitungan mencari nilai *gain ratio* untuk masing-masing atribut. Setelah semua atribut masuk pohon (memiliki kelas), maka tidak perlu dilakukan perhitungan lebih lanjut.

BAB 4

TOOLS DATA MINING

4.1. *Waikato Environment for Knowledge Analysis* (Weka)

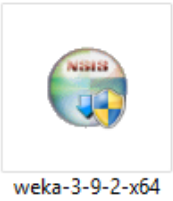
Weka merupakan sebuah *tools data mining open source* yang berbasis java dan berisi kumpulan algoritma *machine learning* dan data *pre-processing*. *Waikato Environment for Knowledge Analysis* adalah kepanjangan dari Weka yang dibuat di Universitas Waikato, New Zealand yang digunakan untuk pendidikan, penelitian, dan aplikasi eksperimen *data mining*. Weka mampu menyelesaikan masalah-masalah *data mining* yang ada di dunia, khususnya klasifikasi yang mendasari pendekatan *machine learning*. Weka dapat digunakan dan diterapkan pada beberapa tingkatan yang berbeda. Weka menyediakan pengimplementasian algoritma pembelajaran *state of the art* yang dapat diterapkan pada *dataset* dari *command line*.

Dalam Weka terdapat *tools* untuk *preprocessing data*, *clustering*, aturan asosiasi, klasifikasi, regresi, dan visualisasi. Adanya *tools* yang sudah tersedia pengguna dapat melakukan *preprocess* pada data, kemudian memasukkannya dalam sebuah skema pembelajaran, dan menganalisis *classifier* yang dihasilkan serta performanya. Semua itu dilakukan tanpa menulis kode program sama sekali. Contoh penggunaan weka adalah menerapkan sebuah metode pembelajaran ke *dataset* dan menganalisis hasilnya untuk memperoleh informasi tentang data, atau menerapkan beberapa metode dan membandingkan performa untuk dipilih.

4.1.1. Instalasi Weka

Instalasi weka pada buku ini menggunakan versi 3.9.2. Langkah-langkah instalasi weka, yaitu sebagai berikut.

1. Klik dua kali file *executable* dari weka 3.9.2 (.exe) seperti pada Gambar 4.1

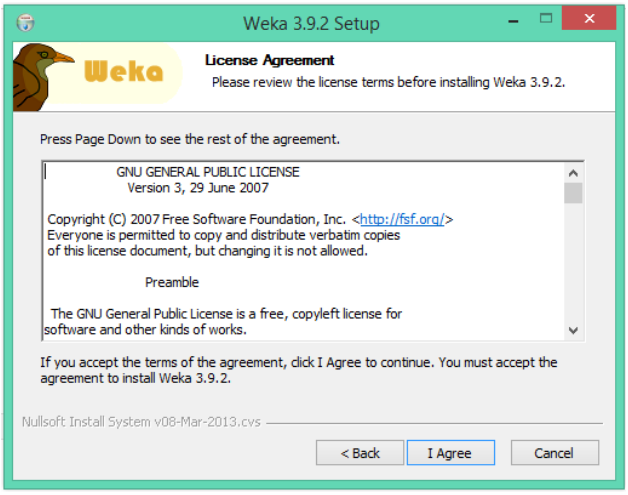


Gambar 4.1. *Installer* weka 3.9.2

2. Kemudian akan muncul jendela seperti pada Gambar 4.2, selanjutnya pilih *next.*

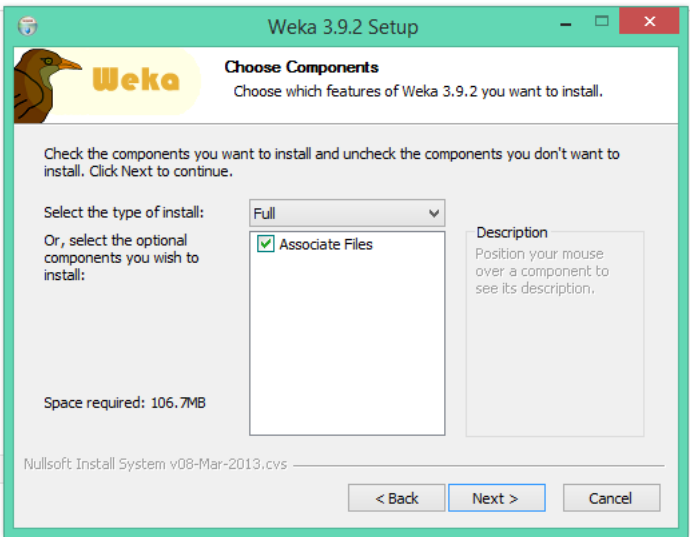
Gambar 4.2. Jendela awal *setup* weka 3.9.2

3. Kemudian akan muncul jendela *Licene Agreement* seperti pada Gambar 4.3, lalu pilih *I Agree.*



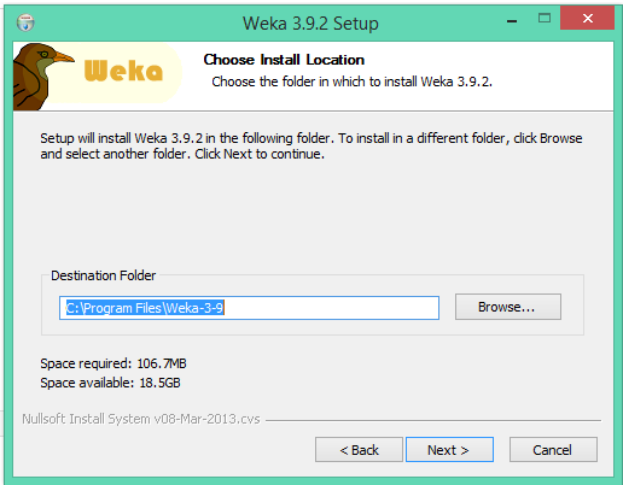
Gambar 4.3. Jendela *License Agreement* proses instalasi weka 3.9.2

4. Selanjutnya akan muncul jendela seperti pada Gambar 4.4, pada bagian *select type of install* pilih *full*, untuk menginstall seluruh komponen yang diperlukan untuk menjalankan apilkasi, kemudian pilih *next.*



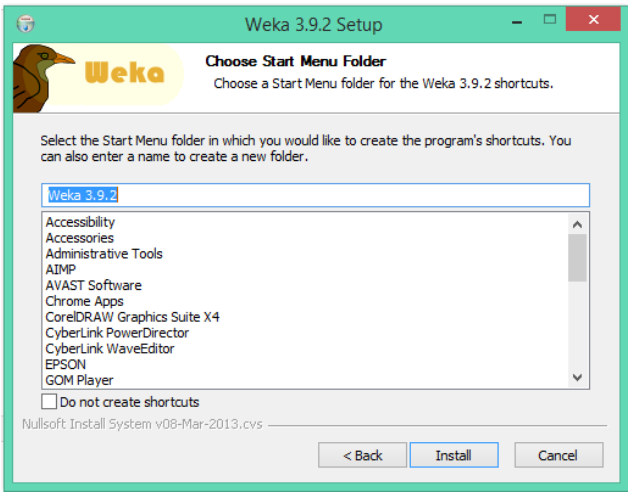
Gambar 4.4. Jendela *Choose Components*

5. Selanjutnya seperti pada Gambar 4.5, tentukan dimana ingin menyimpan *file* hasil proses instalasinya, setelah selesai menentukan direktori kemudian pilih *next.*



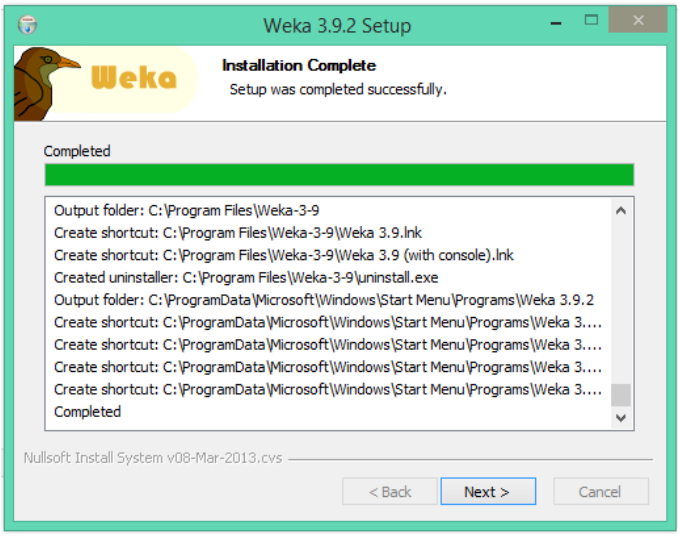
Gambar 4.5. Jendela *Choose Install location*

6. Kemudian tentukan apakah ingin membuat *shortcut* untuk menjalankan aplikasinya pada *start* menu atau tidak dan tentukan nama dari *shortcut*nya, selanjutnya pilih *install* seperti pada Gambar 4.6 berikut ini.



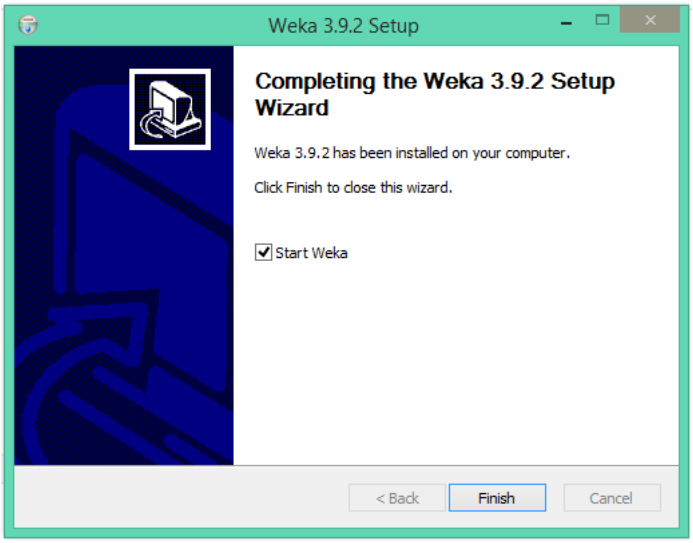
Gambar 4.6. Jendela *Choose Start Menu Folder*

7. Maka proses instalasi akan dilakukan. Setelah proses instalasi selesai kemudian klik *next* seperti pada Gambar 4.7.



Gambar 4.7. Jendela Instalasi weka 3.9.2 (Instalasi selesai)

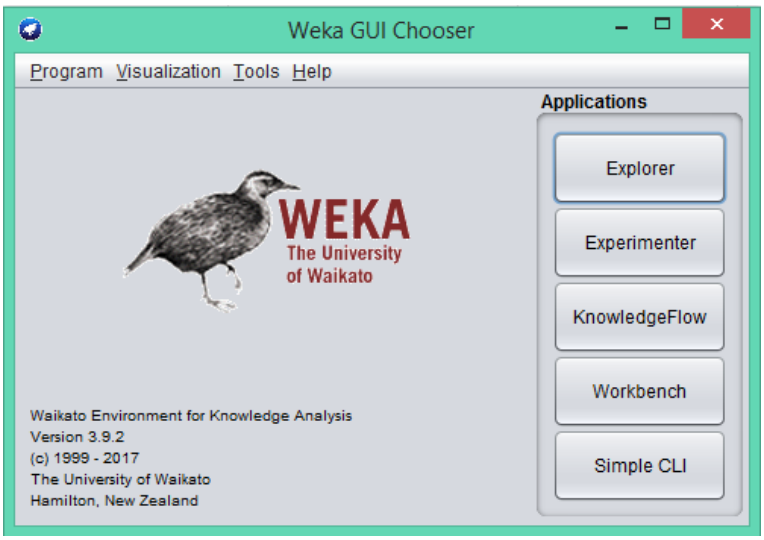
8. Selanjutnya klik *finish*, apabila ingin menjalankan aplikasi weka maka tandai pilih *Start Weka* seperti yang ditunjukkan pada Gambar 4.8.



Gambar 4.8. Jendela pemberitahuan instalasi weka 3.9.2 selesai

4.1.2. Menjalankan Weka

Setelah proses instalasi selesai, pada subbab ini akan membahas tentang GUI Weka**.** Gambar 4.9 merupakan halaman awal dari weka 3.9.2.



Gambar 4.9. Tampilan Awal Weka 3.9.2

Tampilan awal ketika apliaksi weka dijalankan maka terlihat seperti pada Gambar 4.9, pada tampilan awal weka terdapat empat menu utama diantaranya *program, visualisation, tools,* dan *help* serta lima tombol diantaranya *explorer, experimenter, knowledgeflow, woekbench,* dan *simple CLI.*

1. Program

   Menu program mempunyai empat sub menu, diantaranya:

   a. *LogWindow* (*Shortcut* CTRL+L)

      Sub menu *LogWindow* digunakan untuk menampilkan log yang merekap semua yang tercetak untuk stdout dan stderr.

b. *Memory usage* (*Shortcut* CTRL+M)

   Sub *menu memory usage* digunakan untuk menampilkan penggunaan memori pada saat aplikasi weka digunakan.

c. *Setting*

   Sub menu *setting* digunkaan untuk mengatur tampilan pada *user interface.*

d. *Exit* (*Shortcut* CTRL+E)

   Submenu *exit* digunakan untuk keluar dari aplikasi weka.

2. *Visualisation*

   Menu *visualisation* merupakan sarana yang digunakan untuk memvisualisasikan data dengan aplikasi weka. Menu ini mempunyai lima submenu, diantaranya:

   a. *Plot* (*Shortcut* CTRL+P)

      Sub menu plot digunakan untuk menampilkan plot 2D dari sebuah *dataset*.

   b. ROC (*Shortcut* CTRL+R)

      Sub menu ROC digunakan untuk menampilkan kurva ROC yang telah disimpan sebelumnya.

   c. *TreeVisualizer* (*Shortcut* CTRL+T)

      Sub menu *TreeVisualizer* digunakan untuk menampilkan graf berarah, contohnya: *decision tree.*

   d. *Graph Visualizer* (*Shortcut* CTRL+G)

      Sub menu *graph visualizer* digunakan untuk memvisualisasikan format grafik XML, BIF, atau DOT, contohnay sebuah jaringan bayesian.

   e. *Boundary Visualizer* (Shortcut CTRL+B)

      Sub menu *boundary visualizer* bertugas untuk mengizinkan visualisasi dari batas keputusan *classifier* dalam plot 2D.

3. *Tools*

   Menu *tools* menampilkan aplikasi lainnya yang berguna bagi pengguna. Pada menu ini terdapat empat sub menu, yaitu:

   a. *Package Manager* (*Shortcut* CTRL+U)
   b. *ArffViewer* (*Shortcut* CTRL+A)

      Sebuah aplikasi MDI yang menampilkan *file* Arff dalam format *spreadsheet.*
   c. *SqlViewer* (*Shortcut* CTRL+S)

      Merepresentasikan sebuah lembar kerja SQL, untuk melakukan *query database* via JDBC.
   d. *Bayes net editor* (*Shortcut* CTRL+N)

      Sebuah apliksi untuk mengedit, memvisualisasikan dan mempelajari *bayes net.*

4. *Help*

   Menu *help* mempunyai empat sub menu, diantaranya:

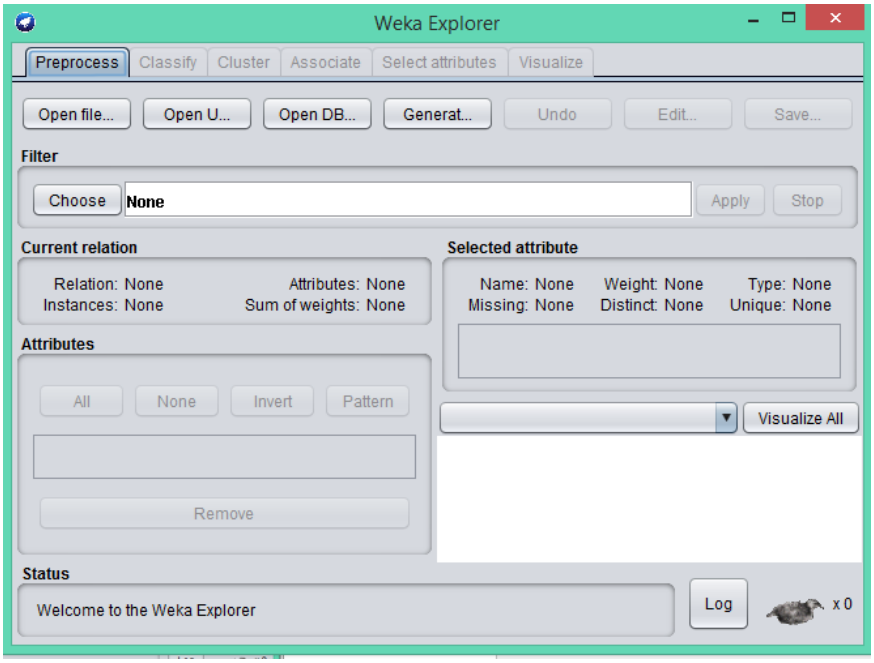
   a. *Weka Homepage* (*Shortcut* CTRL+H)
   b. *HOWTOs, code snippets, etc.* (*Shortcut* CTRL+W)
   c. *Weka on sourcefouge* (*Shortcut* CTRL+F)
   d. *System info* (*Shortcut* CTRL+I)

Tools yang dapat digunakan untuk *preprocessing dataset* membuat pengguna dapat berfokus pada algoritma yang digunakan tanpa terlalu memperhatikan detail seperti pembacaan data dari *file-file*, penyediaan kode untuk evaluasi hasil, dan implementasi algoritma *filtering*. Untuk melakukan pengujian maka perlu memahami beberapa tombol GUI yang ada di weka, diantaranya:

a. *GUI Explorer*

   GUI Explorer merupakan GUI yang menyediakan semua fitur weka dalam bentuk tombol dan tampilan visualisasi yang menarik dan lengkap. *GUI Explorer* adalah GUI yang paling mudah digunakan. *Preprocess*, asosiasi,

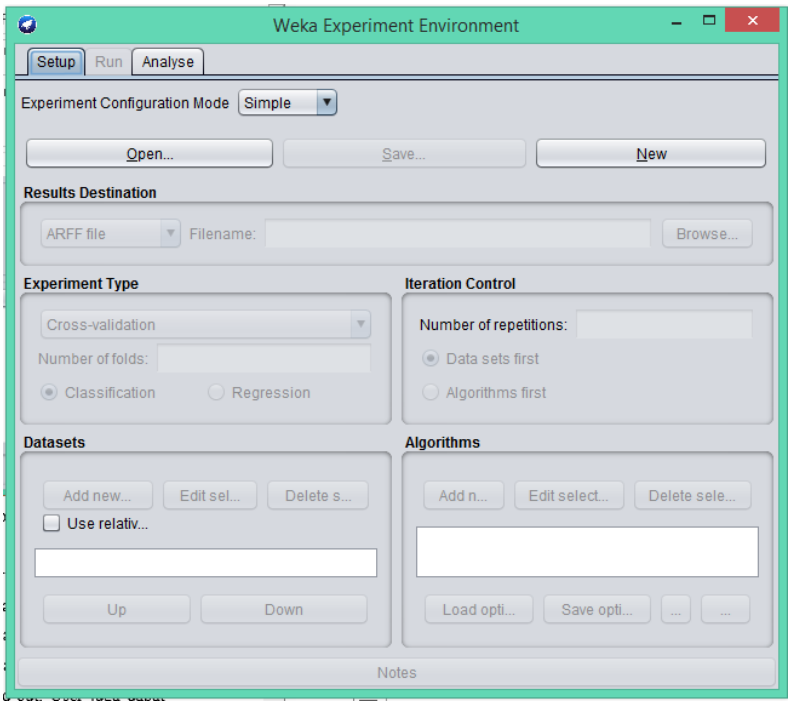
klasifikasi, *clustering*, *select atribut*, dan *visualize* dapat dilakukan dengan mudah. Tampilan *weka expoler* dapat dilihat pada Gambar 4.10.



Gambar 4.10. Tampilan *weka explorer*

b. *GUI Experimenter*

GUI experimenter biasanya digunakan untuk klasifikasi dan regresi. *GUI experimenter* dapat memudahkan perbandingan performansi skema-skema pembelajaran yang berbeda. Hasil dari perbandingan tersebut dapat dituliskan dalam *database* atau file. Dalam Weka tersedia pilihan evaluasi yaitu *learning curve*, *cross-validation*, *hold-out*. Pengguna juga dapat melakukan iterasi menurut beberapa *setting parameter* yang berbeda. *GUI experimenter* dapat dilihat pada Gambar 4.11.



Gambar 4.11. Tampilan *Weka Experiment Environment*

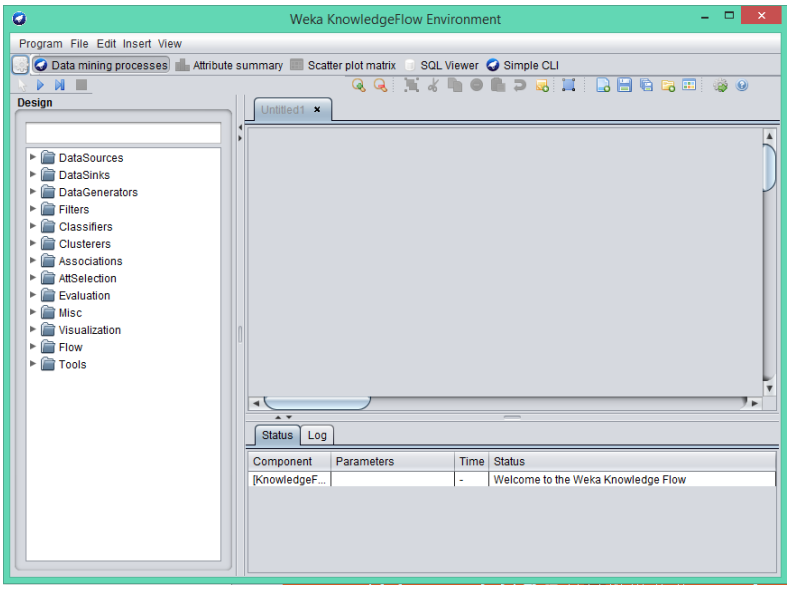
Tab setup yang muncul saat dibuka *experimenter* memungkinkan *user* memilih dan mengkonfigurasi eksperimen yang dilakukan. Setelah menyimpan definisi eksperimen yang dilakukan pengguna dapat memulai eksperimen dari *tab run* dan mengklik tombol *start*. Hasilnya akan disimpan dalam format CSV dan dapat dibuka dalam bentuk *spreadsheet*.

Tab Analize, dapat digunakan untuk menganalisa hasil eksperimen yang dikirim ke weka, Jumlah baris hasil ditunjukkan pada *panel source*. Hasilnya dapat di-*load* dalam format .arff maupun dari basis data

c. *GUI Knowledge Flow*

GUI knowledge flow merupakan GUI yang ada dalam weka yang merupakan antarmuka *Java-Beans-Based* untuk melakukan *setting* dan menjalankan

percbaan-percobaan *macine learning*. Tampilan GUI *Weka knoledge flow* dapat dilihat pada Gambar 4.12.



Gambar 4.12. *Weka Knowledge Flow*

Knowledge flow dapat menangani data secara *incremental* maupun dalam *batches* (*Explorer* hanya menangani data *batch*). Tentunya diperlukan sebuah *classifier* yang dapat *diupdate instance per instance* untuk pembelajaran dari data secara *incremental*.

4.2. *RapidMiner*

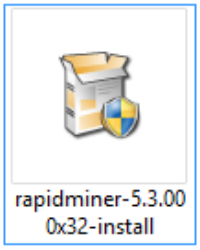
RapidMiner merupakan perangkat lunak yang bersifat *open source* untuk melakukan analisis *data mining, text mining*, dan analisis prediksi. Operator data mining yang terdapat pada *RapidMiner* diantaranya operator untuk *input, output*, visualisasi, dan data *preprocessing*. *RapidMiner* ditulis menggunakan bahasa java sehingga bisa bekerja di semua sistem operasi. Sebelumnya *RapidMiner* bernama *Yet Another Learning Environment (YALE)*, dimana versi awalnya mulai dikembangkan pada tahun 2001 oleh RalfKlinkenberg, Ingo Mierswa, dan

Simon Fischer di Artificial Intelligence Unit dari University of Dortmund. *RapidMiner* didistribusikan di bawah lisensi GNU *Affero General Public License (AGPL)* versi 3. Hingga saat ini ribuan aplikasi telah dikembangkan menggunakan *RapidMiner* di lebih dari 40 negara. Sebagai *software open source* untuk *data mining, RapidMiner* tidak perlu diragukan lagi karena *software* ini sudah terkemuka di dunia. Peringkat pertama *software data mining* pada *polling* oleh KDnuggets, sebuah portal *data mining* pada 2010-2011 ditempati oleh *RapidMiner.*

4.2.1. Instalasi *RapidMiner*

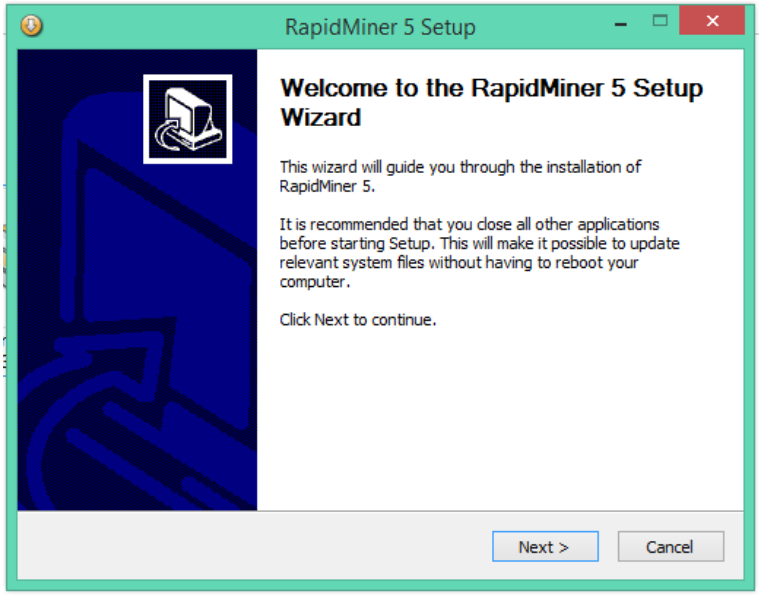
Instalasi *RapidMiner* pada buku ini menggunakan versi 5.3. Langkh-lanagkah instalasi *rapid miner*, yaitu sebagai berikut.

1. Klik dua kali file *executable* dari *RapidMiner* 5.3.000x32-install(.exe) seperti pada Gambar 4.13.



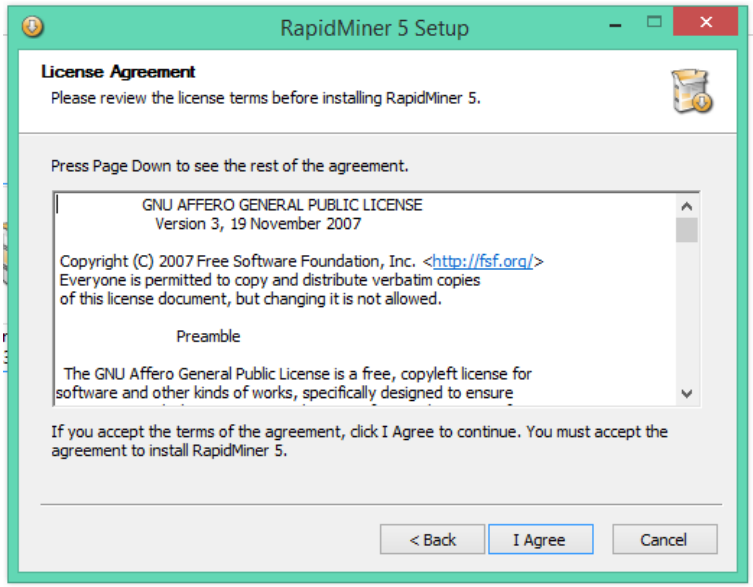
Gambar 4.13. *Installer RapidMiner* 5.3.000x32-install.exe

2. Jendela yang muncul pada Gambar 4.14, pilih *next*



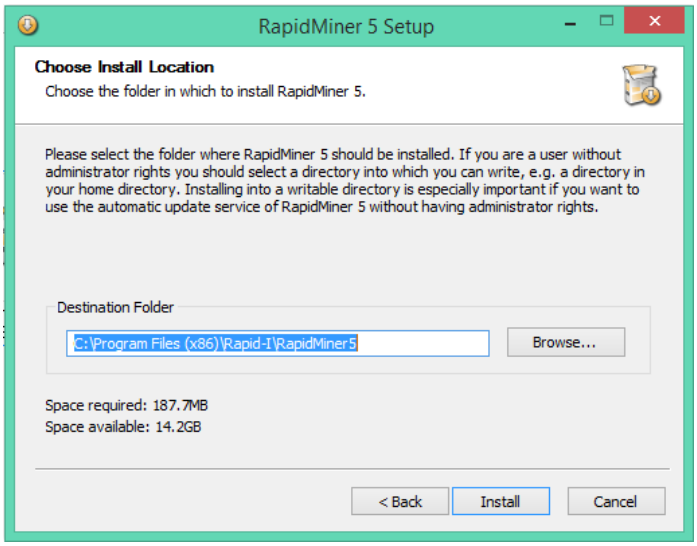
Gambar 4.14. Jendela awal *setup RapidMiner* 5.3

3. Jendela *Licene Agreement* pada Gambar 4.15, pilih *I Agree*



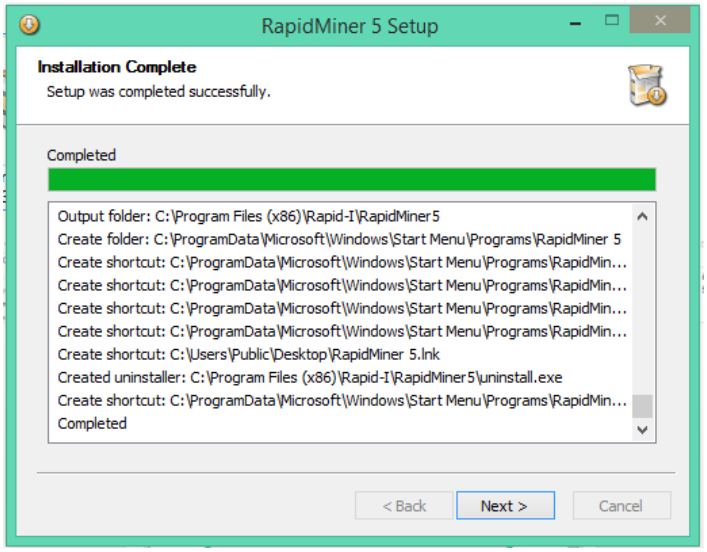
Gambar 4.15. Jendela *License Agreement* proses instalasi *RapidMiner* versi 5.3

4. Selanjutnya akan menampilkan *form* seperti pada Gambar 4.15 untuk menentukan tempat penyimpanan *file* hasil proses instalasi, setelah selesai menentukan direktori kemudian pilih *install*



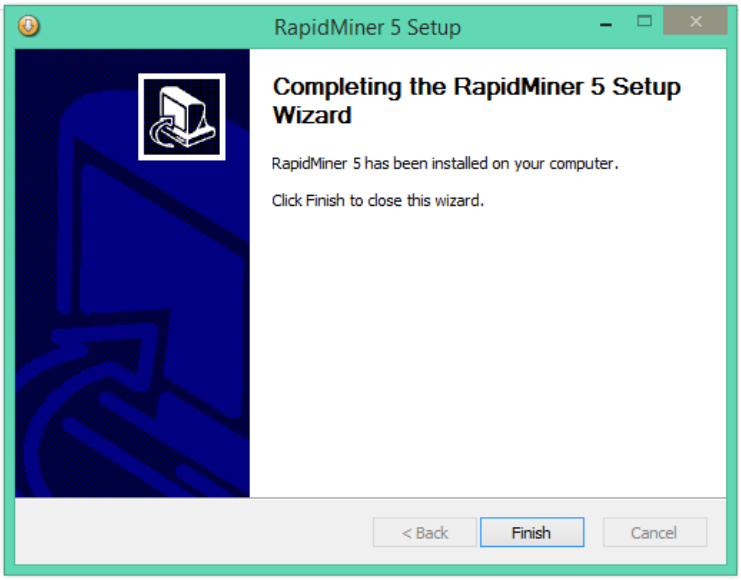
Gambar 4.15. Jendela *Choose Install location*

5. Setelah proses instalasi selesai kemudian klik *next* seperti pada Gambar 4.16.



Gambar 4.16. Jendela Instalasi *RapidMiner* 5.3 (Instalasi selesai)

6. Selanjutnya klik *finish* pada Gambar 4.17 untuk mengakhiri proses instalasi.



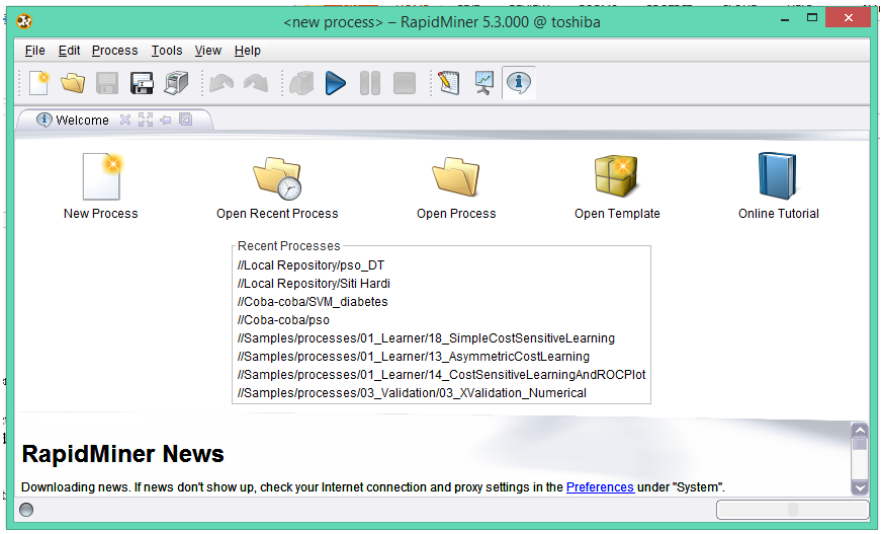
Gambar 4.17. Jendela pemberitahuan instalasi *RapidMiner* 5.3 selesai

4.2.2. Pengenalan *Interface RapidMiner*

RapidMiner menyediakan tampilan yang *user friendly* untuk memudahkan penggunanya ketika menjalankan aplikasi. Tampilan pada *RapidMiner* dikenal dengan istilah *perspective* dan terdapat 3 *perspective* dalam *RapidMiner* diantaranya *welcome perspective*, *design perspective*, dan *result perspective*.

1. ***Welcome Perspective***

Ketika membuka aplikasi maka akan tampil seperti pada Gambar 4.18.



Gambar 4.18. Tampilan *Welcome Perspective*

Pada gambar di atas menampilkan beberapa daftar aksi, diantaranya *New, Open Recent Process, Open Process, Open Template, dan Online Tutorial*. Berikut ini rincian lengkap daftar aksi tersebut:

a. *New*

New digunakan untuk memulai proses analisis baru. Untuk memulai proses analisis, pertama-tama harus menentukan nama dan lokasi proses serta data *repository*.

b. *Open Recent Process*

Open recent process digunakan untuk membuka proses yang baru saja ditutup. Selain itu, dapat digunakan untuk membuka proses yang baru ditutup dengan mengeklik dua kali dari salah satu daftar yang ada pada *recent process* maka tampilan *welcome persective* akan otomatis beralih ke *design perspective*.

c. *Open Process*

Open process digunakan untuk membuka *repository browser* yang berisi daftar proses.

d. *Open Template*

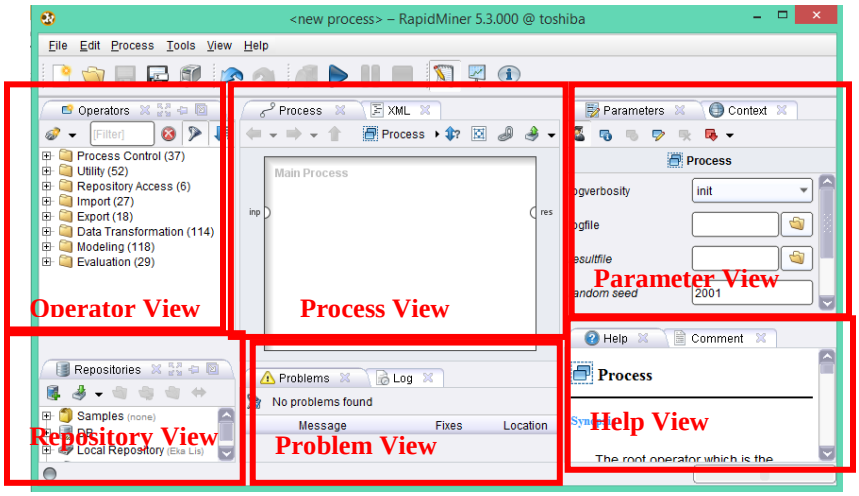
Open template digunakan untuk menunjukkan pilihan lain yang sudah ditentukan oleh proses analisis.

e. *Online Tutorial*

Online tutorial digunakan untuk memulai tutorial secara *online* dengan catatan komputer harus terhubung dengan internet. Tutorial yang didapat secara langsung dari rapid miner berupa perkenalan tentang konsep *data mining*.

2. *Design Perspective*

Design perspective merupakan lembar kerja pada *RapidMiner* yang digunakan untuk membuat dan mengelola proses analisis dari konsep *data mining*. Seperti yang ditunjukkan pada Gambar 4.19 yang mempunyai beberapa *view*.



Gambar 4.19 Tampilan *Design Perspective*

Berikut rincian dari beberapa *view* yang ditampilkan pada *design perspective*:

a. *Operator View*

   Operator view merupakan salah satu *view* yang paling penting karena semua operator dari *RapidMiner* disajikan dalam bentuk hierarki sehingga operator-operator tersebut dapat digunakan pada proses analisis *data mining*.

b. *Process View*

   Process view merupakan halaman yang digunakan untuk menunjukkan langkah-langkah dalam proses analisis *data mining* dengan menggunakan komponen-komponen yang ada pada *operator view*.

c. *Parameter View*

   Beberapa operator dalam *RapidMiner* membutuhkan satu atau lebih parameter agar dapat didefinisikan sebagai fungsionalitas yang benar. Namun terkadang parameter tidak mutlak dibutuhkan, meskipun eksekusi operator dapat dikendalikan dengan menunjukkan nilai parameter tertentu.

d. *Repository View*

   Repository view merupakan komponen utama dalam *design perspective*. *Repository view* digunakan untuk mengelola dan menata proses analisis *data mining*.

e. *Problem View*

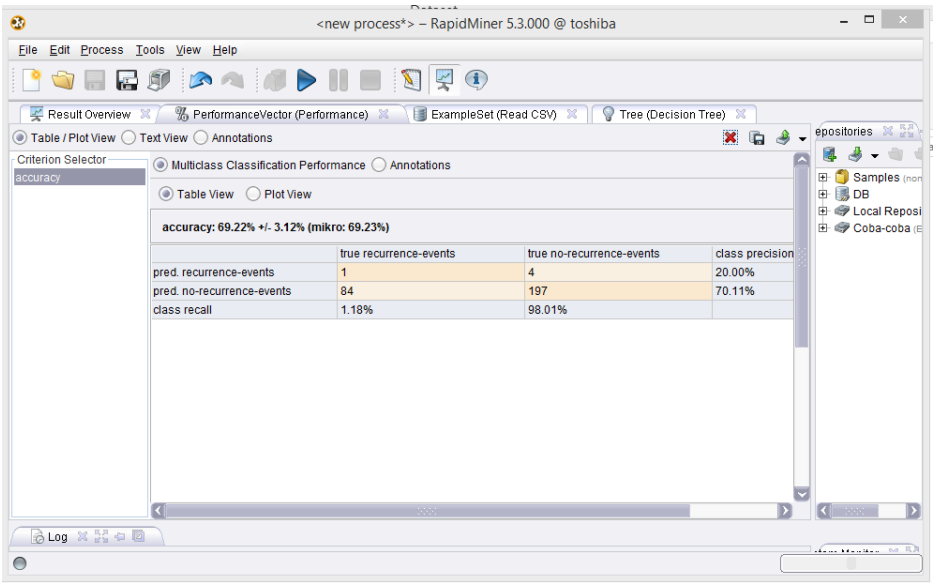
   Problem view merupakan komponen yang sangat berharga dan merupkan sumber bantuan bagi pengguna selama merancang proses analisis, karena apabila ada kesalahan dalam proses analisis maka akan ada pemberitahuan pada halaman *problem view*.

f. *Help dan Comment View*

Help view digunakan untuk memberi penjelasan singkat mengenai fungsi operator dalam satu atau beberapa kalimat. Sedangkan *comment view* merupakan area bagi pengguna menuliskan komentar pada proses analisis *data mining*.

3. *Result Perspective*

Result perspective merupakan tampilan yang digunakan untuk menampilkan hasil dari proses analisis data mining, seperti yang ditunjukkan pada Gambar 4.20.



Gambar 4.20. Tampilan *Result Perspective*

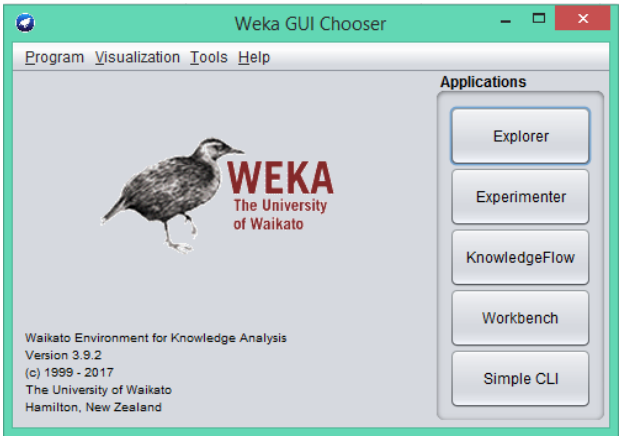
BAB 5

KASUS KLASIFIKASI PADA *TOOLS*

Setelah dijelaskan beberapa proses penerapan klasifikasi pada bab sebelumnya, maka pada bab ini akan dijelaskan beberapa proses *data mining* menggunakan *software* atau *tools data mining* berupa weka dan *RapidMiner*. Seperti yang telah dijelaskan pada bab sebelumnya weka merupakan aplikasi *data mining open source* berbasis java yang berisi kumpulan algoritma *machine learning* dan data *preprocessing*. Sedangkan *RapidMiner* merupakan *tools* atau perangkat lunak yang bersifat *open source* untuk melakukan analisis *data mining, text mining*, dan analisis prediksi.

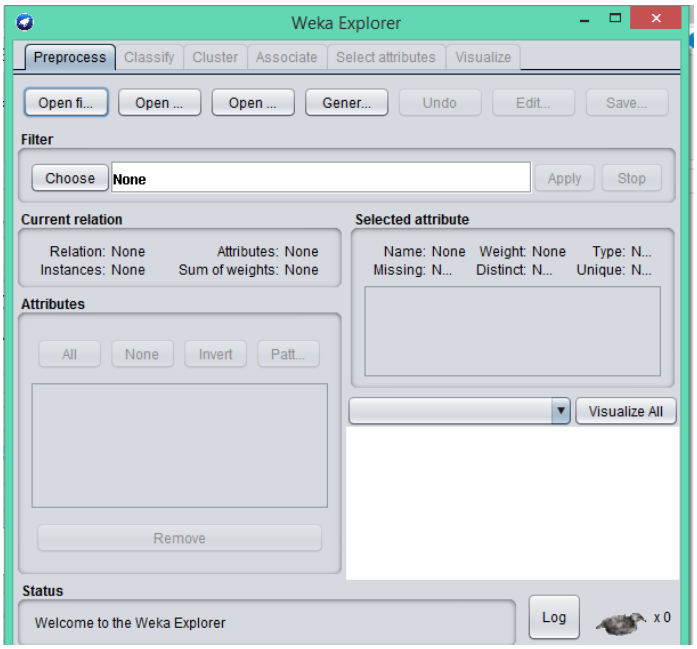
5.1 Algoritma C4.5 dan *Multilayer Perceptron Backpropagation*

Penerapan kasus yang dilakukan di *tools* weka dapat dilakukan dengan cara berikut. Tampilan awal dari *software* weka ditunjukkan pada Gambar 5.1.



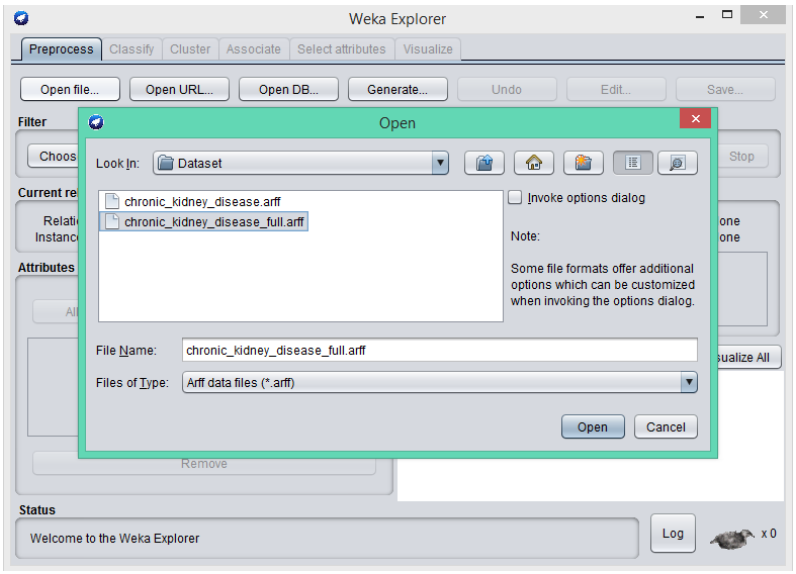
Gambar 5.1. Tampilan awal *software* weka

Ada lima *interface* dalam *software weka* sesuai dengan yang dijelaskan pada bab sebelumnya, tetapi *interface* yang akan digunakan dalam kasus ini yaitu *interface weka explorer*, Gambar 5.2 merupakan tampilan *interface weka explorer*.



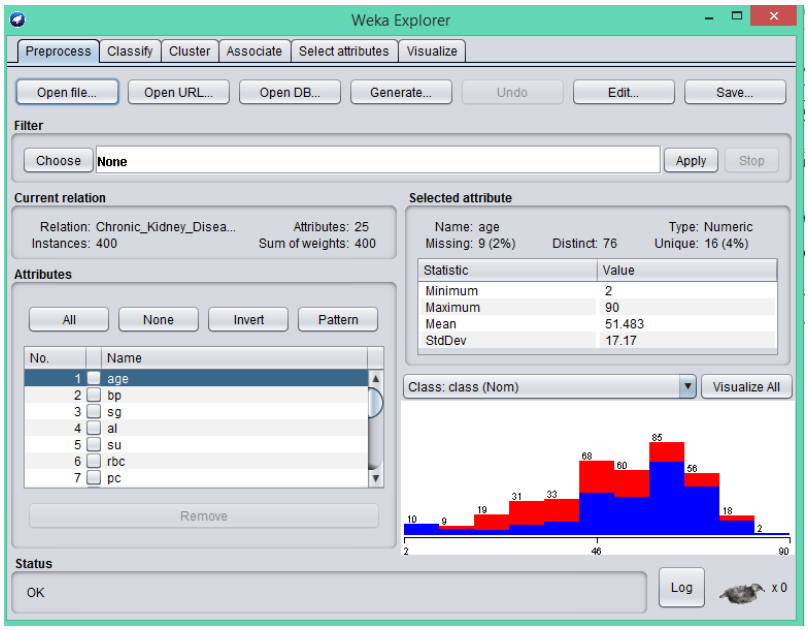
Gambar 5.2. Tampilan dari *interface explorer*

Bagian awal tampilan *weka explorer* terdapat menu *Preprocess*, yang digunakan untuk memproses data sebelum masuk ke proses klasifikasi. Pada menu *Preprocess* terdapat menu *open file* yang digunakan untuk membuka atau memilih data yang akan diproses dalam *tools* weka. Menu tersebut merupakan salah satu langkah awal yang dilakukan setelah membuka *tools* weka. Data yang digunakan dalam kasus ini yaitu *dataset chronic kidney disease* yang diperoleh dari UCI *of machine learning repository*. Setelah klik menu *open file* kemudian memasukkan *dataset chronic kidney disease* seperti pada Gambar 5.3.



Gambar 5.3. Memasukkan *dataset chronic kidney disease*.arff

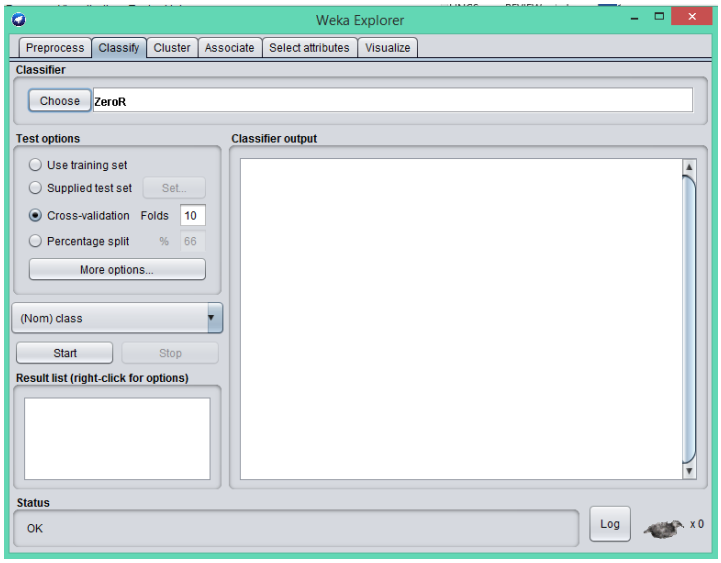
Setelah *dataset chronic kidney disease* dimasukkan ke weka, maka grafik data dapat ditampilkan pada Gambar 5.4.



Gambar 5.4. Hasil grafik isi *dataset chronic kidney disease* setelah di-*input*-kan

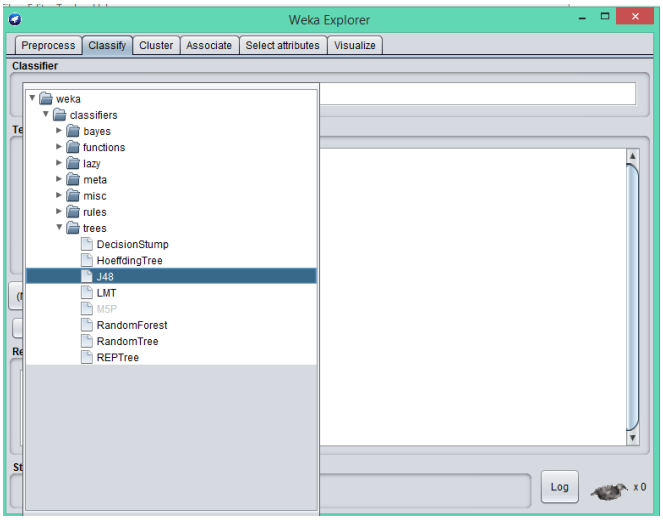
5.1.1 Pengujian dan hasil klasifikasi *decision tree* dengan algoritma C4.5

Proses klasifikasi algoritma C4.5 di weka dapat dilakukan setelah *dataset chronic kidney disease* di-*input*-kan melalui menu *Preprocess*, maka langkah selanjutnya yaitu menuju ke menu *classify*. Tampilan menu *classify* ditunjukkan pada Gambar 5.5.



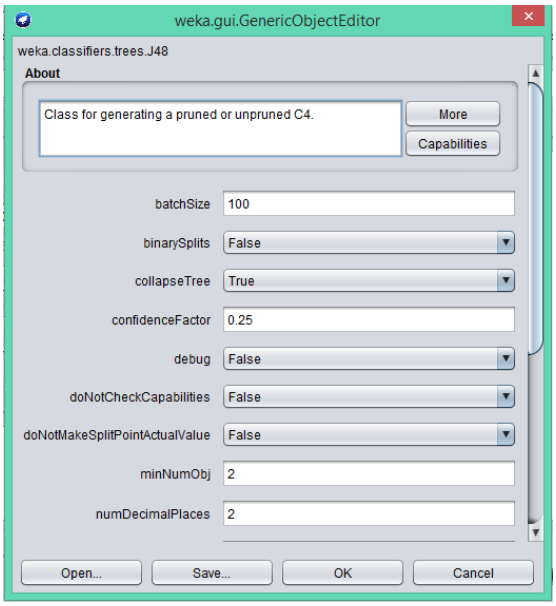
Gambar 5.5. Tampilan menu *classify*

Pada menu *classify* terdapat beberapa bagian diantaranya *classifier*, *test options*, dan *classifier output*. Bagian *classifier* digunakan untuk memilih algoritma yang digunakan, *text options* digunakan untuk memilih metode pengujian evaluasi, dan *classifier output* digunakan untuk menampilkan hasil dari klasifikasi. Bagian *classifier* pada menu *choose* untuk memilih algoritma C4.5 atau dalam weka biasa disebut dengan J48 seperti yang ditampilkan pada Gambar 5.6.



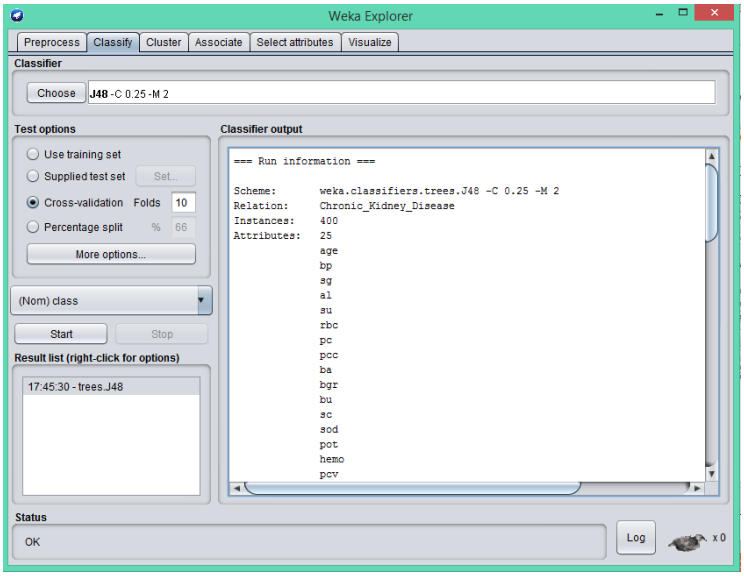
Gambar 5.6. Memilih klasifikasi algoritma C4.5

Langkah selanjutnya, yaitu mengatur klasifikasi *decision tree* C4.5 untuk proses pemangkasan *(prunning)* pada *dataset* yang digunakan pada *interface weka gui generic object editor*. Tampilan weka *gui generic objec editor* C4.5 dapat dilihat pada Gambar 5.7.



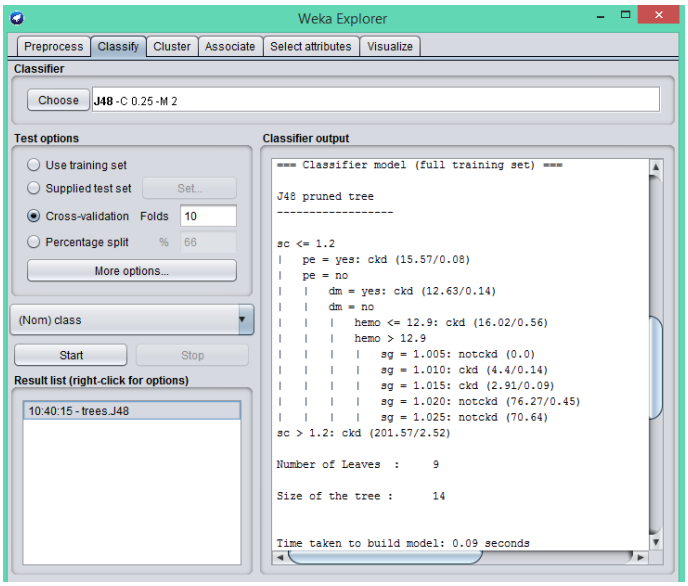
Gambar 5.7. Tampilan *weka gui generic objec editor* C4.5

Penjelasan dari Gambar 5.8, yaitu digunakan untuk mengatur nilai *confidence factor* atau nilai pangkas yang akan digunakan pada proses klasifikasi. Kemudian, mengatur nilai *minNumObj* yang digunakan untuk mengatur nilai *minimun instance* per *leaf*. Selanjutnya, pada pilihan *unpruned* dipilih *false* apabila ingin dilakukan pemangkasan begitu juga sebaliknya. Setelah mengatur *gui generic objec editor* C4.5 maka pilih menu OK untuk melakukan proses klasifikasi selanjutnya. Proses selanjutnya yaitu memilih proses pengujian dengan *cross validation* kemudian untuk memulai proses klasifikasi pilih menu *start* maka hasil akan muncul di jendela *classifier output* seperti pada Gambar 5.8.



Gambar 5.8. Hasil dari klasifikasi algoritma C4.5

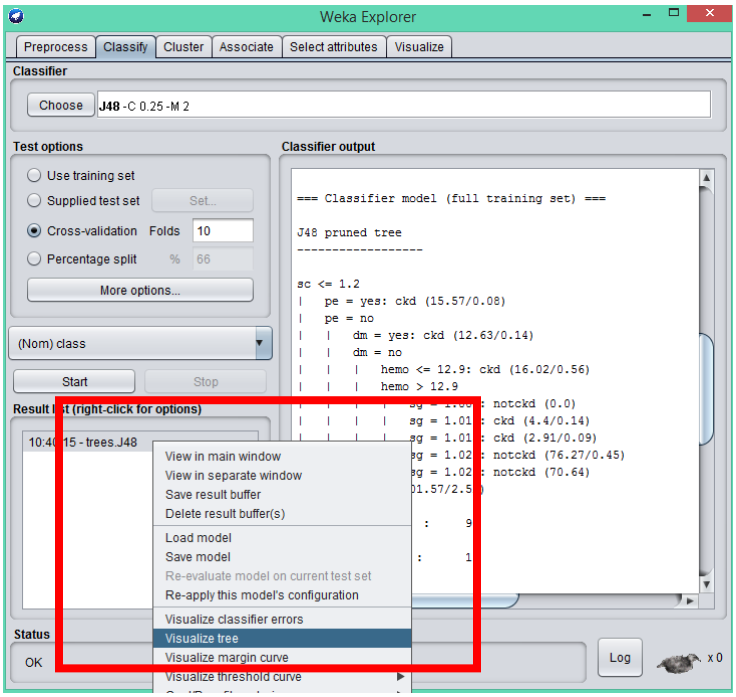
Berdasarkan Gambar 5.8 dapat dilihat pada jendela *classifier output* bahwa data yang digunakan yaitu *chronic kidney disease* yang mempunyai 400 *instance* dan 25 atribut. Selanjutnya, hasil dari pemangkasan algoritma C4.5 dan waktu untuk membangun model (kompleksitas waktu) dapat dilihat pada Gambar 5.9.



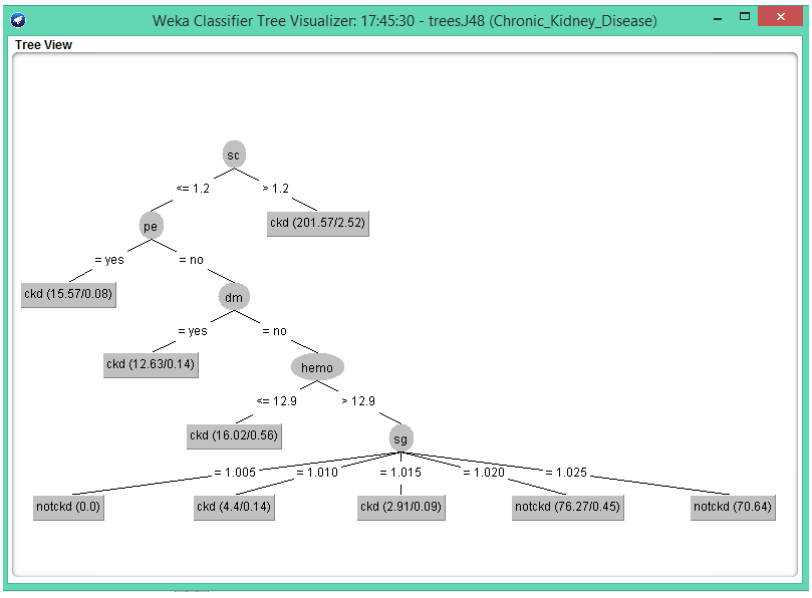
Gambar 5.9. Hasil dari pemangkasan algoritma C4.5 dan waktu untuk membangun model

Pemangkasan algoritma C4.5 menghasilkan 9 *leaf* dan 14 *tree*, dimana definisi *leaf* merupakan simpul yang tidak mempunyai cabang lagi, sedangkan *tree* merupakan simpul yang mempunyai cabang. *Time to build model* digunakan untuk mengetahui waktu yang dibutuhkan untuk membangun model klasifikasi dari algoritma C4.5. Pada kasus ini waktu yang dibutuhkan yaitu 0,09 *seconds*. Hasil pohon keputusan dari pemangkasan algoritma C4.5 dapat ditampilkan pada jendela *weka classifier tree visualizer*.

Langkah untuk melihat pohon keputusan dapat dilakukan dengan cara klik kanan pada jendela *result list trees*. J48 → *visualize tree* seperti yang terlihat pada Gambar 5.10, maka pohon keputusan yang dihasilkan dapat dilihat pada Gambar 5.11.



Gambar 5.10. Tampilan jendela *result list* (*right-click for options*) *trees J48*

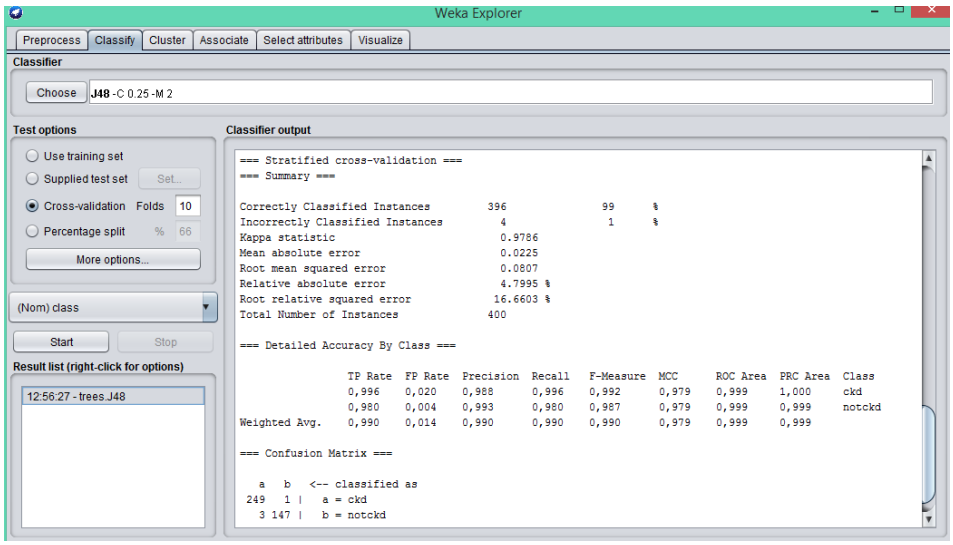


Gambar 5.11. Hasil pohon keputusan dari algoritma C4.5

Gambar 5.11 mempunyai bentuk aturan *If then* untuk *decision tree* C4.5, yaitu sebagai berikut;

a. If sc > 1,2 then ckd
b. If sc ≤ 1,2 and pe = yes then ckd
c. If sc ≤ 1,2 and pe =no and dm = yes then ckd
d. If sc ≤ 1,2 and pe = no and dm = no and hemo ≤ 12,9 then ckd
e. If sc ≤ 1,2 and pe = no and dm = no and hemo > 12,9 and sg = 1,005 then notckd
f. If sc ≤ 1,2 and pe = no and dm = no and hemo > 12,9 and sg = 1,010 then ckd
g. If sc ≤ 1,2 and pe=no and dm=no and hemo > 12,9 and sg=1,015 then ckd
h. If sc ≤ 1,2 and pe = no and dm = no and hemo > 12,9 and sg = 1,020 then notckd
i. If sc ≤ 1,2 and pe=no and dm=no and hemo > 12,9 and sg=1,025 then notckd

Selain waktu eksekusi, hasil dari klasifikasi algoritma C4.5 berupa *confusion matrix* dan akurasi seperti yang dapat dilihat pada Gambar 5.12.



Gambar 5.12. Hasil *confusion matrix* dan akurasi algoritma C4.5 menggunakan *dataset chronic kidney disease*

Hasil evaluasi dari *dataset chronic kidney disease* yang menggunakan *k-fold cross validation* dengan *default* k=10 dapat dilihat pada Tabel 5.1.

Tabel 5.1. Hasil evaluasi algoritma C4.5 menggunakan 10-*fold cross validation*

No.	Spesifikasi Pengukuran	Nilai
1	*Corectly classified instance*	396 atau 99%
2	*Incorectly classified instance*	4 atau 1%
3	*Kappa atatistic*	0,9786
4	*Mean absolute error*	0,0225
5	*Root mean squared error*	0,0807
6	*Relative absolute error*	4,7995%
7	*Root relative squared error*	16,6603%
8	*Total Number of instance*	400

Correctly classified instances merupakan banyak baris data yang terklasifikasikan dengan benar atau sering disebut dengan akurasi dimana akurasi yang diperoleh adalah 99% dengan *record* klasifikasi yang benar sejumlah 396

record. Sedangkan *incorectly classified instances* merupakam baris data yang terklasifikasikan secara tidak benar yaitu sejumlah 4 *record* atau 1%. Algoritma C4.5 menghasilkan nilai *detailed accuracy by class* seperti yang disajikan pada Tabel 5.2. Sedangkan hasil *confusion matrix* disajikan pada Tabel 5.3.

Tabel 5.2. Hasil *detailed accuracy by class*

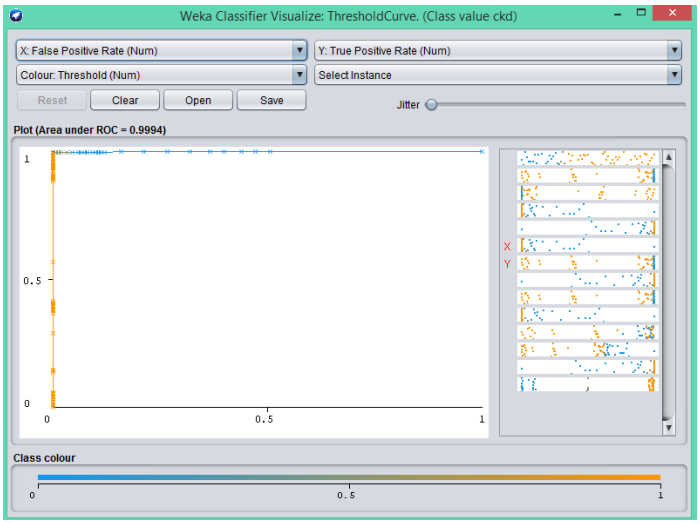
	TP rate	FP rate	Precision	Racall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,996	0,020	0,988	0,996	0,992	0,979	0,999	1,000	ckd
	0,980	0,004	0,993	0,980	0,980	0,979	0,999	0,999	notckd
Weighted Avg.	0,990	0,14	0,990	0,990	0,990	0,979	0,999	0,999	

Tabel 5.3. Hasil *confusion matrix* algoritma C4.5

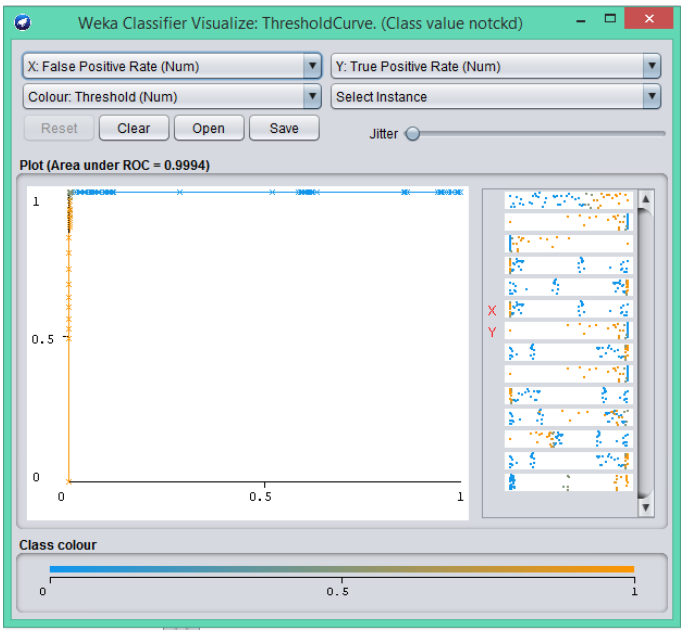
a	b	Classified as
249	1	a=ckd
3	147	b=notckd

Diketahui dari 400 *record*, terdiri dari 250 *record* data yang diklasifikasikan sebagai *class* ckd dengan 249 *record* data yang benar diklasifikasikan sebagai class ckd dan 1 *record* data benar diklasifikasikan sebagai *class* notckd. Sedangkan, 150 *record* data diklasifikasikan sebagai *class notckd* dengan 147 *record* data benar diklasifikasikan sebagai class not ckd dan 3 *record* data salah yang diklasifikasikan sebagai class ckd.

Hasil perhitungan divisualisasikan dengan kurva ROC (*Receiver Operating Characteristic*) atau AUC (*Area Under Curve*). Kurva ROC algoritma C4.5 untuk *class* ckd dapat dilihat pada Gambar 5.13 sedangkan kurva ROC algoritma C4.5 untuk *class* notckd dapat dilihat pada Gambar 5.14.



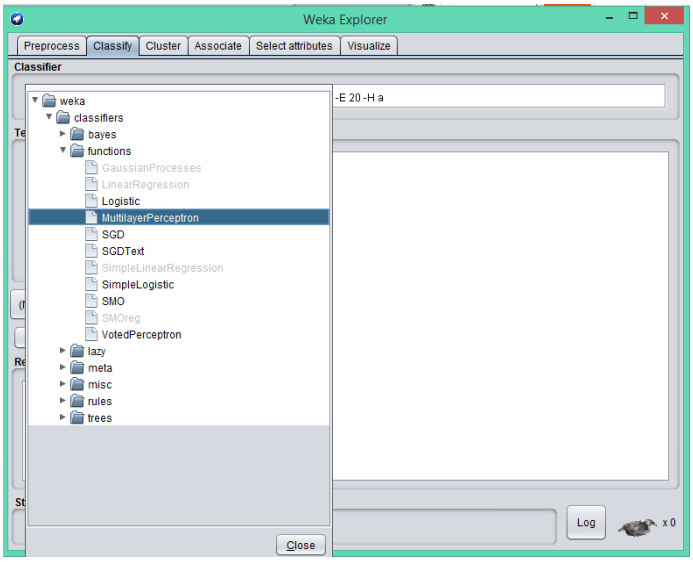
Gambar 5.13. Kurva ROC algoritma C4.5 untuk class ckd



Gambar 5.14. Kurva ROC algoritma C4.5 untuk *class* notckd

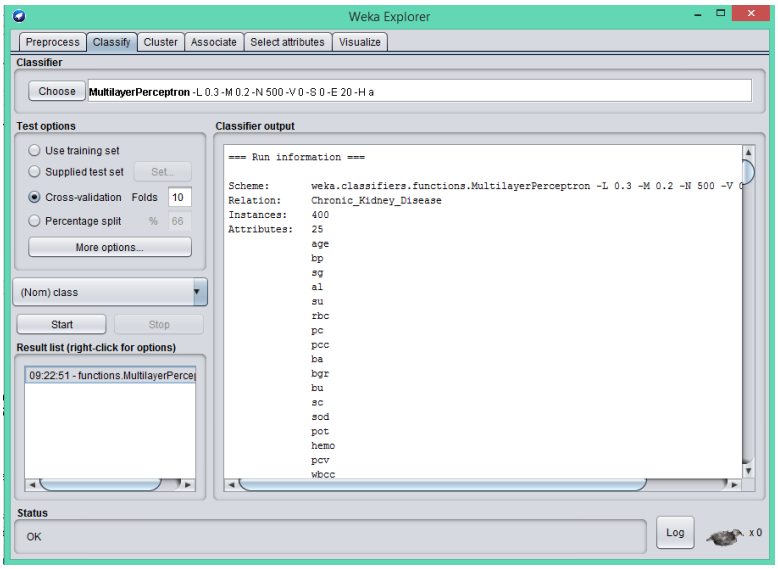
5.1.2 Pengujian dan hasil klasifikasi Algoritma *Multi Layer Perceptron Backpropagation*

Proses klasifikasi algoritma *multilayer perceptron backpropagation* di weka dapat dilakukan setelah *dataset chronic kidney disease* di-*input*-kan melalui menu *Preprocess*, maka langkah selanjutnya yaitu menuju ke menu *classify* kemudian pilih menu *choose* untuk memilih algoritma *multilayer perceptron backpropagation*. Tampilan menu *classify* algoritma *multilayer perceptron backpropagation* ditunjukkan pada Gambar 5.15.



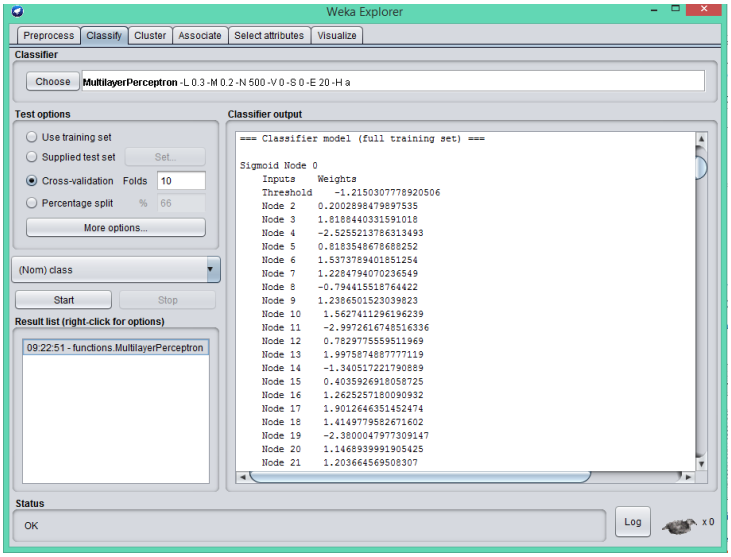
Gambar 5.15. Tampilan menu *classify* algoritma *multilayer perceptron backpropagation*

Eksperimen yang dilakukan pada kasus ini menggunakan *test options cross validation* dengan *folds* 10, setelah itu langsung klik *start* untuk mendapatkan hasil *run information* seperti yang ditampilkan pada Gambar 5.16.



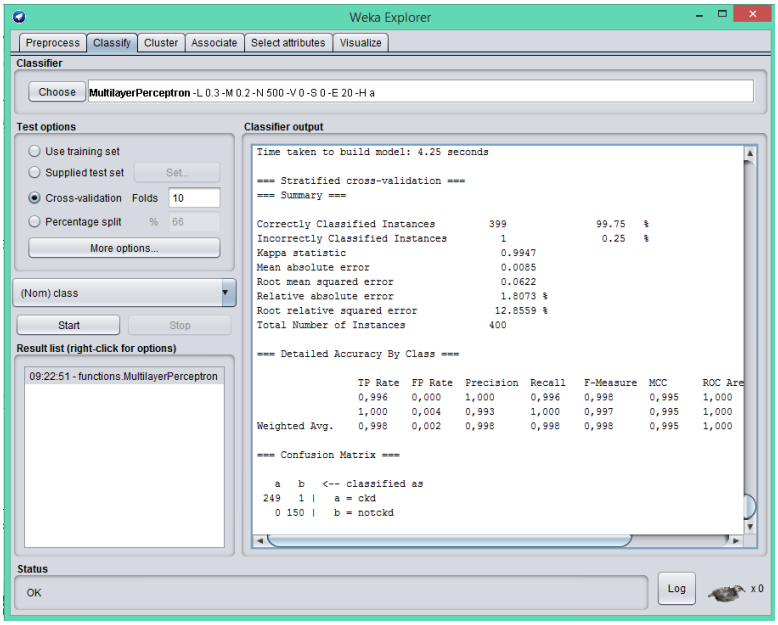
Gambar 5.16. Hasil *run information* algoritma *multilayer perceptron backpropagation*

Hasil dari klasifikasi algoritma *multilayer perceptron backpropagation* ditampilkan pada jendela *classifier output*. *Classifier output* menampilkan *run information* yang menjelaskan algoritma yang digunakan adalah algoritma *multilayer perceptron backpropagation* atau di dalam weka disebut dengan *multilayer perceptron*. Hasil model klasifikasi dari algoritma *multilayer perceptron backpropagation* ditunjukkan pada Gambar 5.17. Model klasifikasi menampilkan *sigmoid node* 0 sampai *sigmoid node* 21 yang berisi *input threshold* dan *weights*.



Gambar 5.17. Hasil model klasifikasi dari algoritma *multilayer perceptron backpropagation*

Hasil akurasi dari klasifikasi algoritma *multilayer perceptron backpropagation* ditampilkan pada Gambar 5.18.



Gambar 5.18. Hasil akurasi dari klasifikasi algoritma *multilayer perceptron backpropagation*

Hasil evaluasi dari *dataset chronic kidney disease* yang menggunakan *k-fold cross validation* dengan *default* k=10 dapat dilihat pada Tabel 5.4.

Tabel 5.4. Hasil evaluasi algoritma *multilayer perceptron backpropagation* menggunakan *10-fold cross validation*

No.	Spesifikasi Pengukuran	Nilai
1	*Corectly classified instance*	399 atau 99,75%
2	*Incorectly classified instance*	1 atau 0,25%
3	*Kappa atatistic*	0,9947
4	*Mean absolute error*	0,0085
5	*Root mean squared error*	0,0622
6	*Relative absolute error*	1,8073%
7	*Root relative squared error*	12,8559%
8	*Total Number of instance*	400

Correctly classified instances merupakan banyak baris data yang terklasifikasikan dengan benar atau sering disebut dengan akurasi dimana akurasi yang diperoleh adalah 99,75% dengan *record* klasifikasi yang benar sejumlah 399 record. Sedangkan *incorectly classified instances* merupakam baris data yang terklasifikasikan secara tidak benar yaitu sejumlah 1 *record* atau 0,25%.

Algoritma *multilayer perceptron* mempunyai nilai *detailed accuracy by class* seperti yang disajikan pada Tabel 5.5. Sedangkan hasil *confusion matrix* disajikan pada Tabel 5.6.

Tabel 5.5. Nilai *detailed accuracy by class*

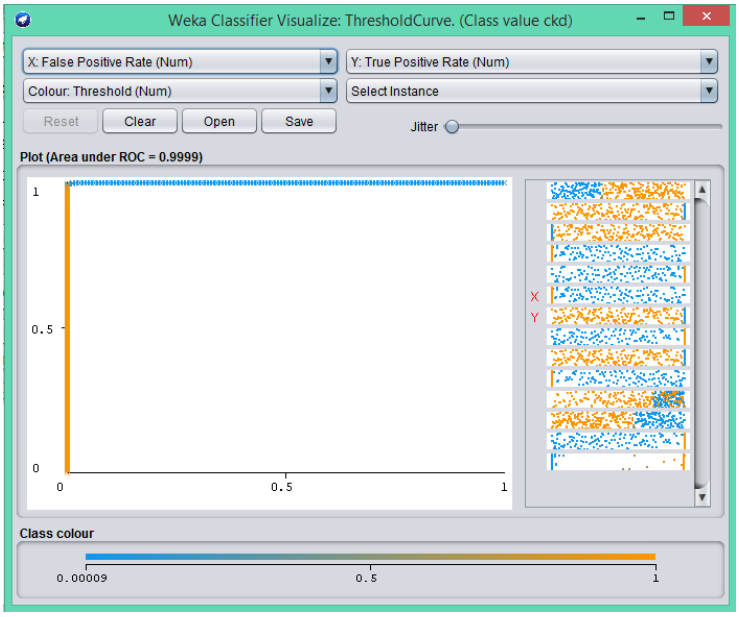
	TP rate	FP rate	Precision	Racall	F-Measure	ROC Area	Class
	0,996	0	1	0,996	0,998	1	ckd
	1	0,004	0,993	1	0,997	1	notckd
Weighted Avg.	0,998	0,002	0,998	0,998	0,998	1	

Tabel 5.6. *Hasil confusion matrix* algoritma *multilayer perceptron backpropagation*

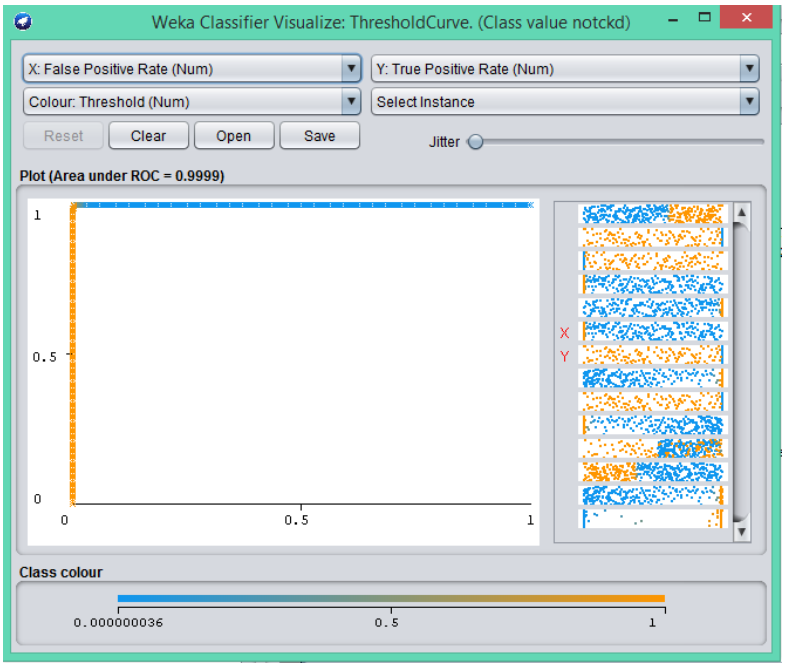
a	b	Classified as
249	1	a=ckd
0	150	b=notckd

Diketahui dari 400 *record*, terdiri dari 250 *record* data yang diklasifikasikan *class* ckd dengan 249 *record* data benar diklasifikasikan sebagai *class* ckd dan 1 *record* data benar yang diklasifikasikan sebagai *class* notckd. Sedangkan, 150 *record* data diklasifikasikan benar sebagai *class* not ckd.

Hasil perhitungan divisualisasikan dengan kurva ROC atau AUC. Kurva ROC algoritma *multilayer perceptron backpropagation* untuk *class* ckd dapat dilihat pada Gambar 5.19 sedangkan kurva ROC algoritma *multilayer perceptron backpropagation* untuk class notckd dapat dilihat pada Gambar 5.20.



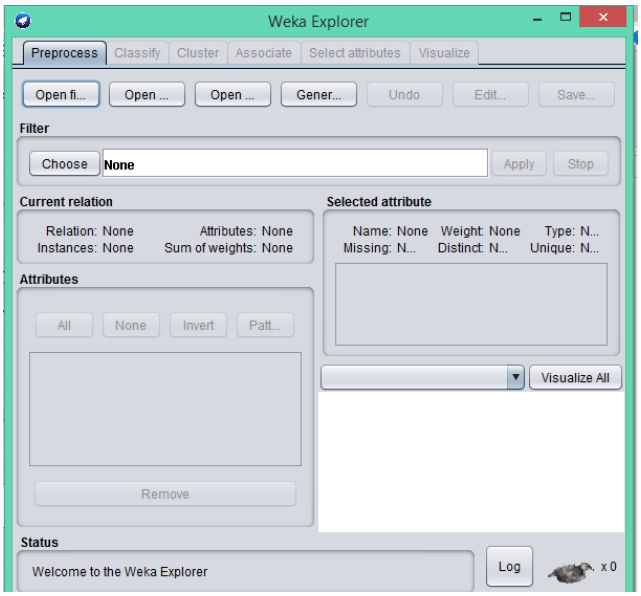
Gambar 5.19. Kurva ROC algoritma *multilayer perceptron backpropagation* untuk *class* ckd



Gambar 5.20. Kurva ROC algoritma C4.5 untuk class notckd

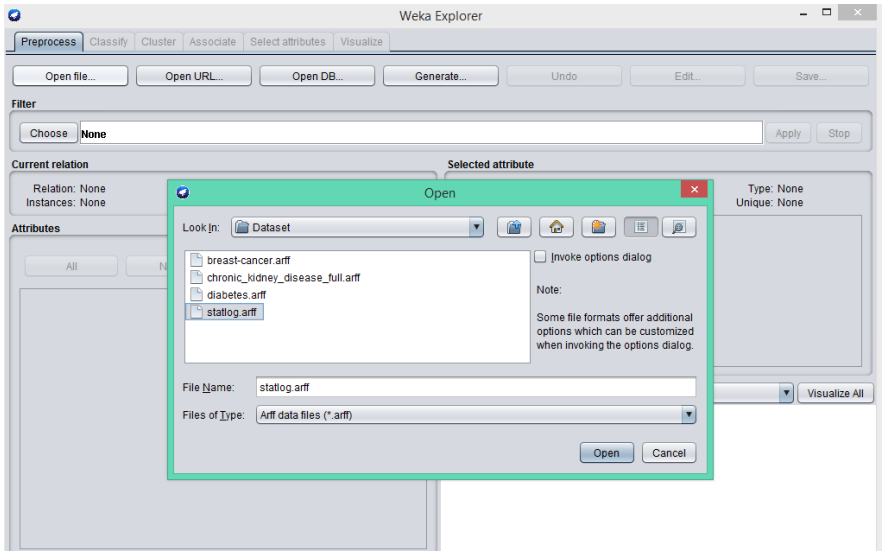
5.2 *Pessimistic Pruning* untuk Meningkatkan Akurasi Algoritma C4.5

Ada lima *interface* dalam *software weka* sesuai dengan yang dijelaskan pada bab sebelumnya, tetapi *interface* yang akan digunakan dalam kasus ini yaitu *interface weka explorer*, Gambar 5.21 merupakan tampilan *interface weka explorer*.



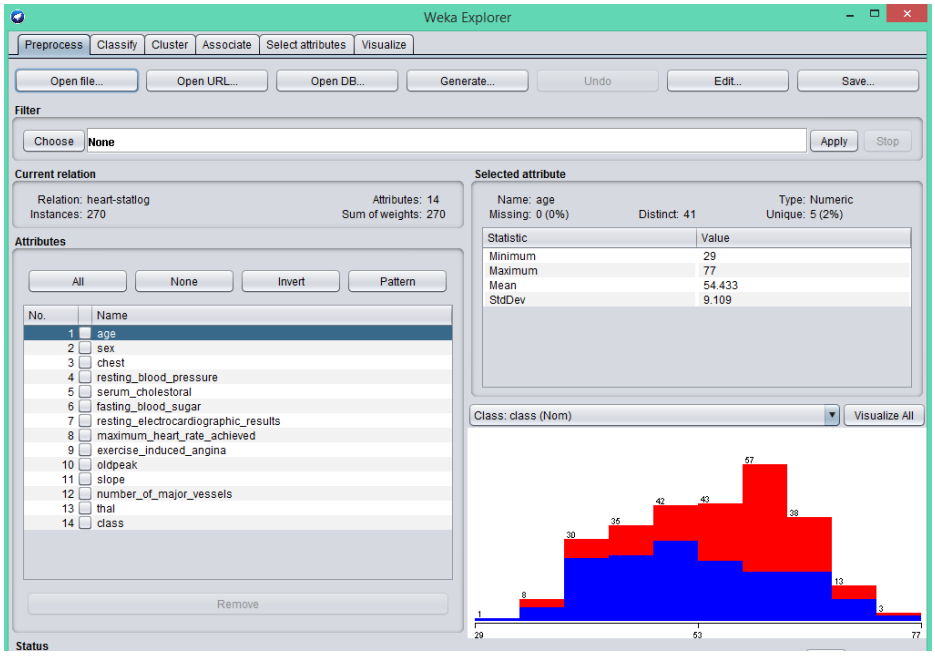
Gambar 5.21. Tampilan dari *interface explorer*

Bagian awal tampilan *weka explorer* terdapat menu *Preprocess*, yang digunakan untuk memproses data sebelum masuk ke proses klasifikasi. Pada menu *Preprocess* terdapat menu *open file* yang digunakan untuk membuka atau memilih data yang akan diproses dalam *tools* weka. Menu tersebut merupakan salah satu langkah awal yang dilakukan setelah membuka *tools* weka. Data yang digunakan dalam kasus ini yaitu *dataset* penyakit jantung yang diperoleh dari UCI *of machine learning repository*. Setelah klik menu *open file* kemudian memasukkan *dataset* penyakit jantung seperti pada Gambar 5.22.



Gambar 5.22. Memasukkan *dataset statlog*.arff

Setelah *dataset* penyakit jantung dimasukkan ke weka. Maka grafik data dapat ditampilkan pada Gambar 5.23.

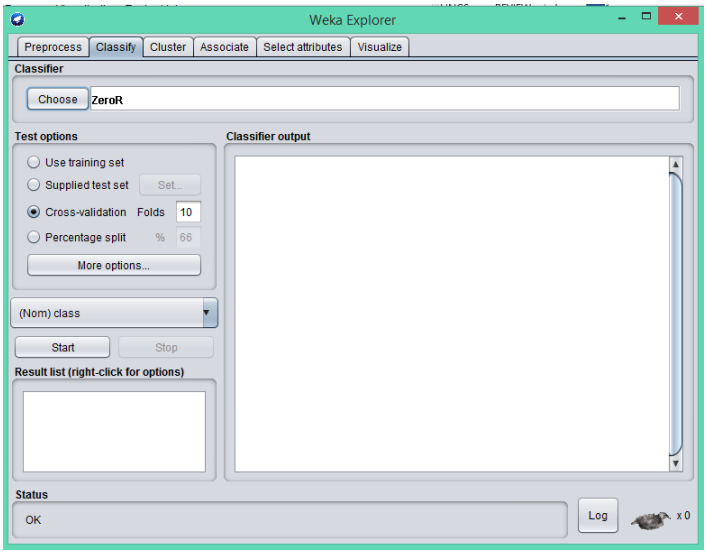


Gambar 5.23. Hasil grafik isi *dataset* penyakit jantung setelah di-*input*-kan

Pada Gambar 5.23 menjelaskan rincian dari dataset penyakit jantung, yaitu terdiri dari 14 atribut dan 270 *instance*.

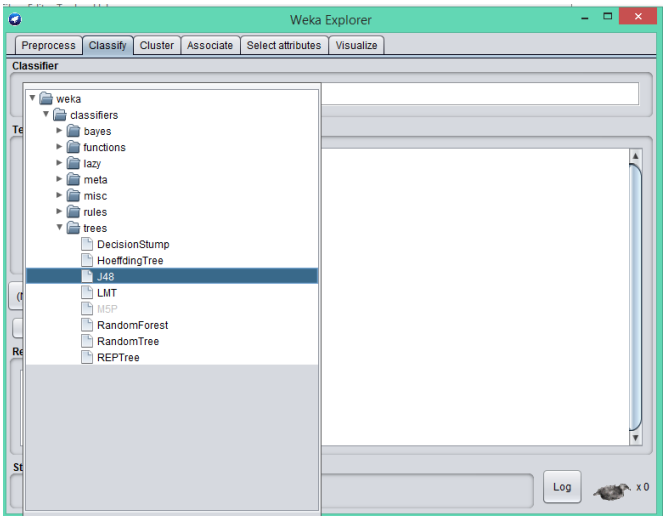
5.2.1 Pengujian dan hasil klasifikasi *decision tree* dengan algoritma C4.5 tanpa *pessimistic prunning*

Proses klasifikasi algoritma C4.5 di weka dapat dilakukan setelah *dataset* penyakit jantung di-*input*-kan melalui menu *Preprocess*, maka langkah selanjutnya yaitu menuju ke menu *classify*. Tampilan menu *classify* ditunjukkan pada Gambar 5.24.



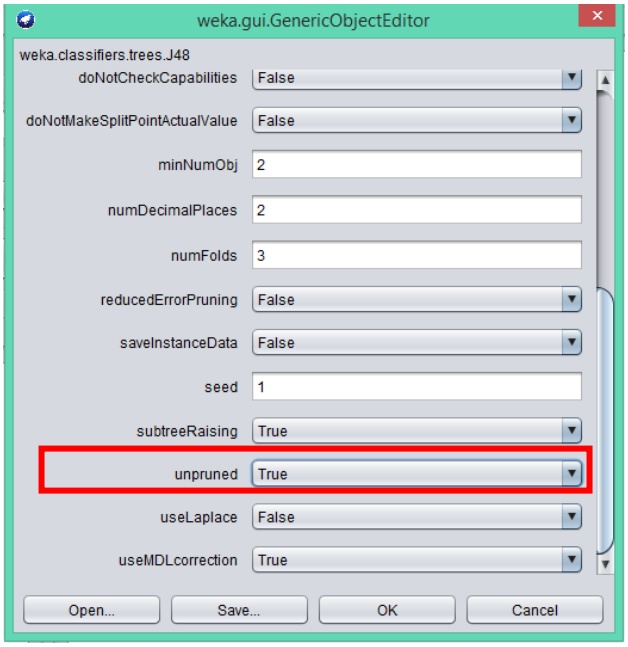
Gambar 5.24. Tampilan menu *classify*

Menu *classify* terdapat beberapa bagian diantaranya *classifier*, *test options*, dan *classifier output*. Bagian *classifier* digunakan untuk memilih algoritma yang digunakan, *text options* digunakan untuk memilih metode pengujian evaluasi, dan *classifier output* digunakan untuk menampilkan hasil dari klasifikasi. Bagian *classifier* pada menu *choose* untuk memilih algoritma C4.5 atau dalam weka biasa disebut dengan J48 seperti yang ditampilkan pada Gambar 5.25.



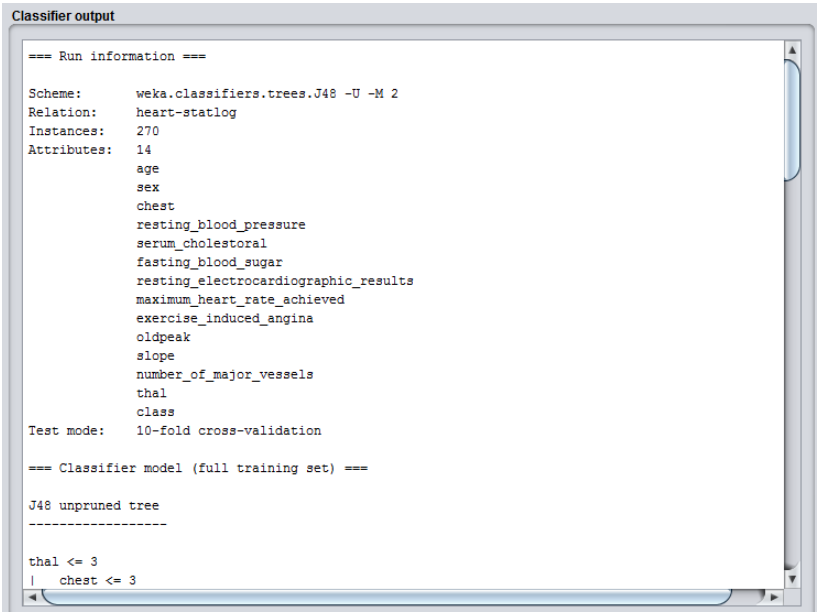
Gambar 5.25. Memilih klasifikasi algoritma C4.5

Langkah selanjutnya, yaitu mengatur klasifikasi *decision tree* C4.5 untuk proses pemangkasan *(prunning)* pada *dataset* yang digunakan pada *interface weka gui generic object editor*. Tampilan weka *gui generic objec editor* C4.5 dapat dilihat pada Gambar 5.26.



Gambar 5.26. Tampilan *weka gui generic object editor* C4.5

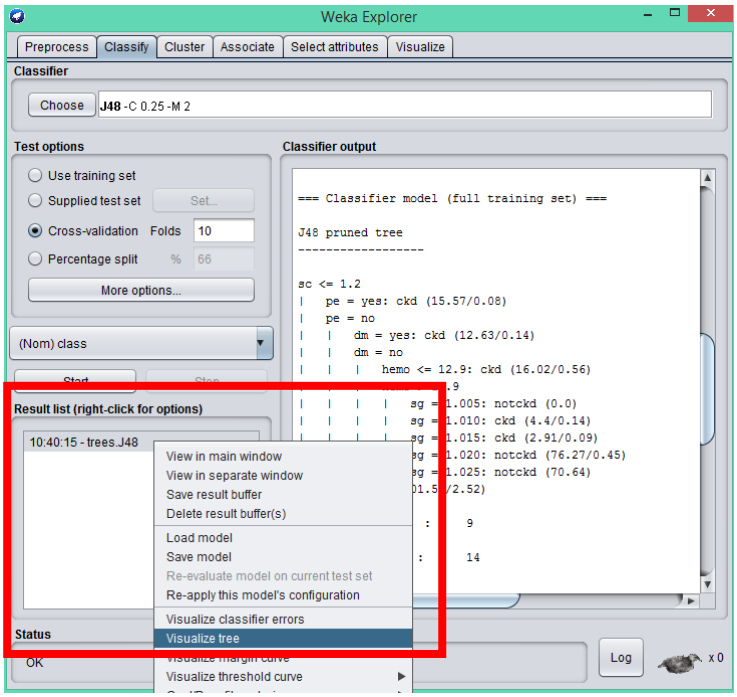
Penjelasan dari Gambar 5.26, yaitu digunakan untuk mengatur nilai *confidence factor* atau nilai pangkas yang akan digunakan pada proses klasifikasi. Pada gambar tersebut menjelaskan bahwa proses yang dilakukan tanpa menggunakan *prunning*. Setelah mengatur selanjutnya klik OK. Proses selanjutnya yaitu memilih proses pengujian dengan *cross validation* kemudian untuk memulai proses klasifikasi pilih menu *start* maka hasil akan muncul di jendela *classifier output* seperti pada Gambar 5.27.



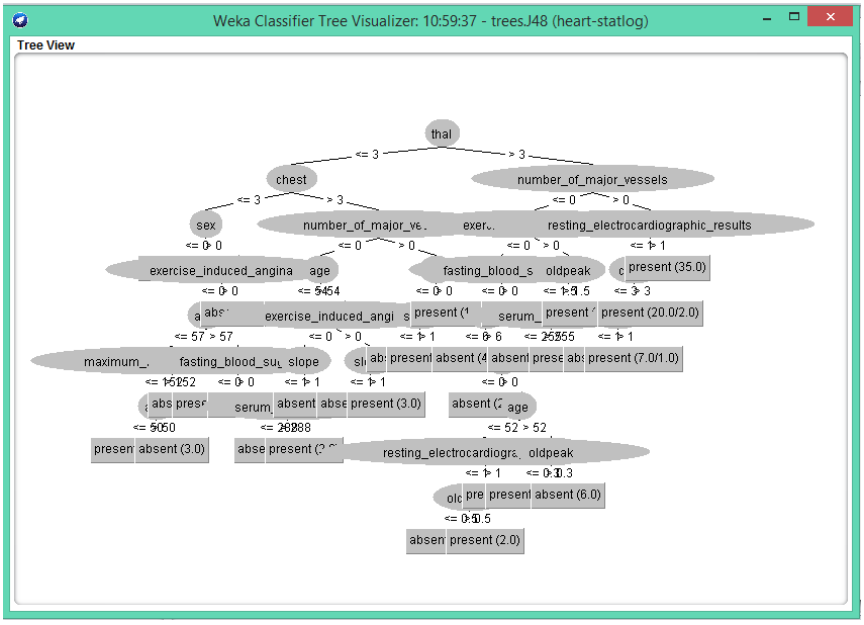
Gambar 5.27. Hasil dari klasifikasi algoritma C4.5

Berdasarkan Gambar 5.27 dapat dilihat pada jendela *classifier output* bahwa data yang digunakan yaitu penyakit jantung yang mempunyai 270 *instance* dan 14 atribut tanpa menggunakan metode *prunning*.

Langkah untuk melihat pohon keputusan dapat dilakukan dengan cara klik kanan pada jendela *result list trees*. J48 → *visualize tree* seperti yang terlihat pada Gambar 5.28, maka pohon keputusan yang dihasilkan dapat dilihat pada Gambar 5.29.

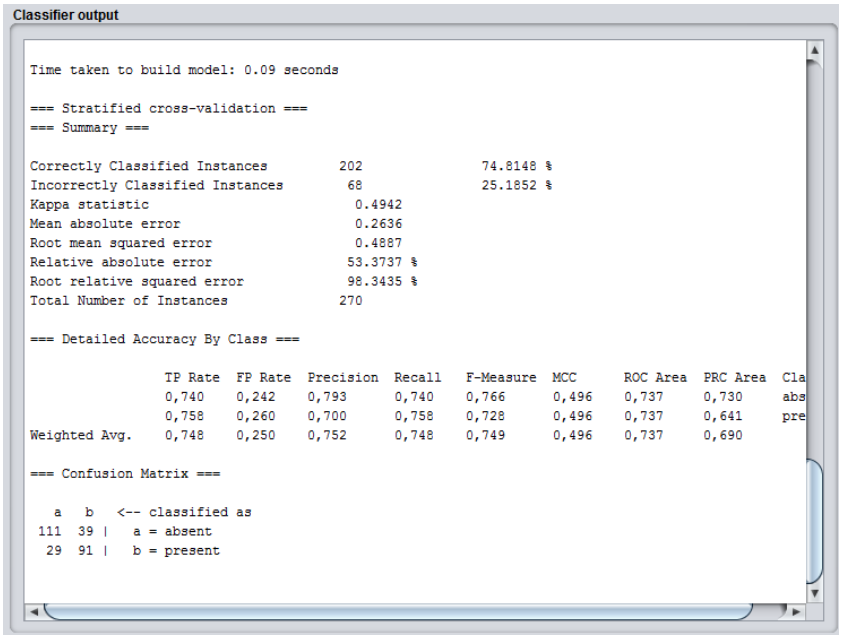


Gambar 5.28. Tampilan jendela *result list* (*right-click for options*) *trees* C4.5



Gambar 5.29. Hasil pohon keputusan dari algoritma C4.5

Selain waktu eksekusi, hasil dari klasifikasi algoritma C4.5 berupa *confusion matrix* dan akurasi dapat dilihat pada Gambar 5.30.



Gambar 5.30. Hasil *confusion matrix* dan akurasi algoritma C4.5 menggunakan *dataset* penyakit jantung

Hasil evaluasi dari *dataset* penyakit jantung yang menggunakan *k-fold cross validation* dengan *default* k=10 dapat dilihat pada Tabel 5.7.

Tabel 5.7. Hasil evaluasi algoritma C4.5 menggunakan 10-*fold cross validation*

No.	Spesifikasi Pengukuran	Nilai
1	*Corectly classified instance*	202 atau 74,81%
2	*Incorectly classified instance*	68 atau 25,18%
3	*Kappa atatistic*	0,4942
4	*Mean absolute error*	0,2636
5	*Root mean squared error*	0,4887
6	*Relative absolute error*	53,3737%
7	*Root relative squared error*	98,3435%
8	*Total Number of instance*	270

Algoritma C4.5 menghasilkan nilai *detailed accuracy by class* seperti yang disajikan pada Tabel 5.8. Sedangkan hasil *confusion matrix* disajikan pada Tabel 5.9.

Tabel 5.8. Hasil *detailed accuracy by class*

	TP rate	FP rate	Precision	Racall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,740	0,242	0,793	0,740	0,766	0,496	0,737	0,730	absent
	0,758	0,260	0,700	0,758	0,728	0,496	0,737	0,641	present
Weighted Avg.	0,748	0,250	0,752	0,748	0,749	0,496	0,737	0,690	

Tabel 5.9. Hasil *confusion matrix* algoritma C4.5

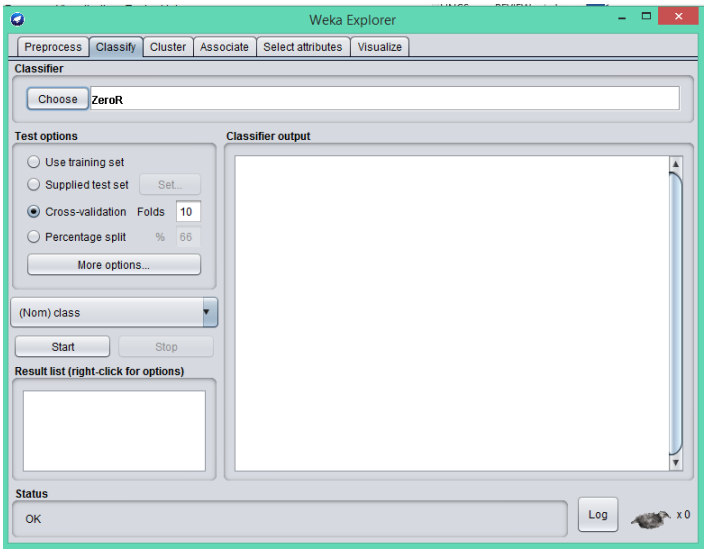
a	b	Classified as
111	39	a=absent
29	91	b=present

Diketahui dari 270 *record*, terdiri dari 150 *record* data yang diklasifikasikan sebagai *class absent* dengan 111 *record* data yang benar diklasifikasikan sebagai *class absent* dan 39 *record* data benar diklasifikasikan sebagai *class present*. Sedangkan, 120 *record* data diklasifikasikan sebagai *class present* dengan 29 *record* data benar diklasifikasikan sebagai *class present* dan 91 *record* data salah yang diklasifikasikan sebagai *class absent*.

5.2.2 Pengujian dan hasil klasifikasi Algoritma *decision tree* C4.5 menggunakan *pessimistic prunning*

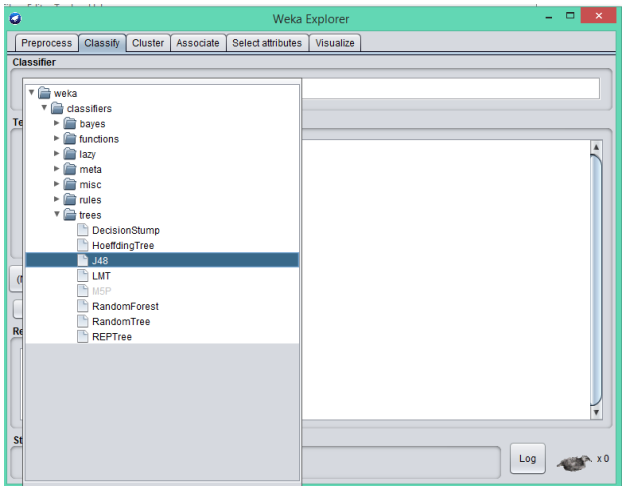
Penerapan *pessimistic prunning* dapat dilakukan pada halaman *weka gui generic object editor* C4.5. Sebelum menuju ke halaman *weka gui generic object editor* C4.5 maka melakukan proses klasifikasi algoritma C4.5 di weka dengan

memasukkan *dataset* penyakit jantung yang di-*input*-kan melalui menu *preprocess* seperti langkah pada Gambar 5.3, maka langkah selanjutnya yaitu menuju ke menu *classify*. Tampilan menu *classify* ditunjukkan pada Gambar 5.31.



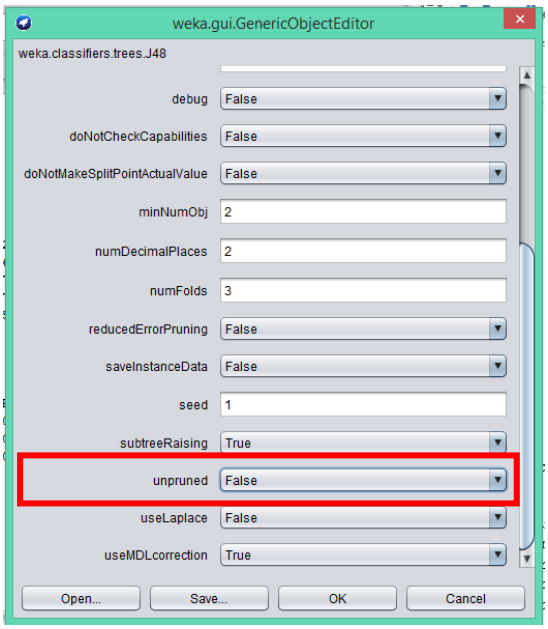
Gambar 5.31. Tampilan menu *classify*

Pada menu *classify* terdapat beberapa bagian diantaranya *classifier*, *test options*, dan *classifier output*. Bagian *classifier* digunakan untuk memilih algoritma yang digunakan, *text options* digunakan untuk memilih metode pengujian evaluasi, dan *classifier output* digunakan untuk menampilkan hasil dari klasifikasi. Bagian *classifier* pada menu *choose* untuk memilih algoritma C4.5 atau dalam weka biasa disebut dengan J48 seperti yang ditampilkan pada Gambar 5.32.



Gambar 5.32. Memilih klasifikasi algoritma C4.5

Langkah selanjutnya, yaitu mengatur klasifikasi *decision tree* C4.5 untuk proses pemangkasan *(prunning)* pada *dataset* yang digunakan pada *interface weka gui generic object editor*. Tampilan weka *gui generic object editor* C4.5 dapat dilihat pada Gambar 5.33.



Gambar 5.33. Tampilan *weka gui generic objec editor* C4.5

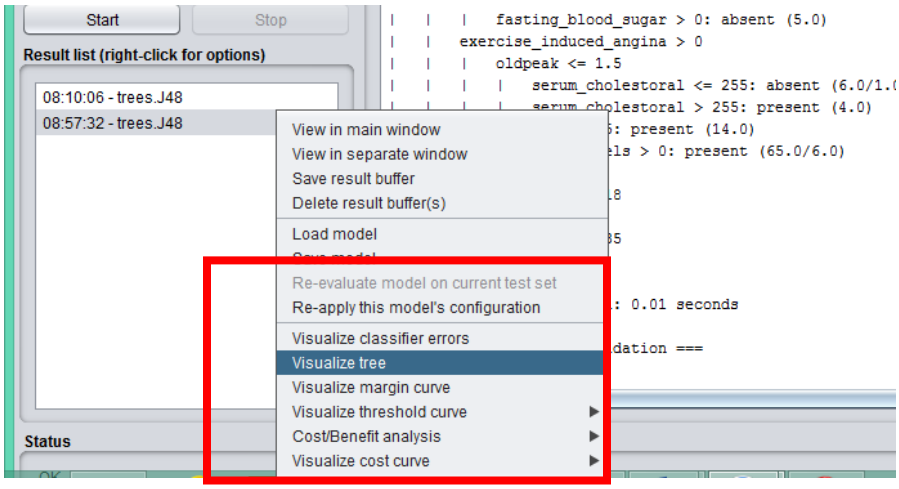
Penjelasan dari Gambar 5.34, yaitu digunakan untuk mengatur nilai *confidence factor* atau nilai pangkas yang akan digunakan pada proses klasifikasi. Kemudian, mengatur nilai *minNumObj* yang digunakan untuk mengatur nilai *minimun instance* per *leaf*. Selanjutnya, pada pilihan *unpruned* dipilih *false* apabila ingin dilakukan pemangkasan begitu juga sebaliknya. Setelah mengatur *gui generic object editor* C4.5 maka pilih menu OK untuk melakukan proses klasifikasi selanjutnya. Proses selanjutnya yaitu memilih proses pengujian dengan *cross validation* kemudian untuk memulai proses klasifikasi pilih menu start maka hasil akan muncul di jendela *classifier output* dari pemangkasan algoritma C4.5 dan waktu untuk membangun model (kompleksitas waktu) dapat dilihat pada Gambar 5.34.

```
=== Classifier model (full training set) ===
J48 pruned tree
------------------
       thal <= 3
       |  chest <= 3: absent (101.0/10.0)
       |  chest > 3
       |  |  number_of_major_vessels <= 0
       |  |  |  age <= 54: absent (17.0)
       |  |  |  age > 54
       |  |  |  |  exercise_induced_angina <= 0
       |  |  |  |  |  slope <= 1
       |  |  |  |  |  |  serum_cholestoral <= 288: absent (2.0)
       |  |  |  |  |  |  serum_cholestoral > 288: present (2.0)
       |  |  |  |  |  slope > 1: absent (5.0/1.0)
       |  |  |  |  exercise_induced_angina > 0
       |  |  |  |  |  slope <= 1: absent (2.0)
       |  |  |  |  |  slope > 1: present (3.0)
       |  |  number_of_major_vessels > 0
       |  |  |  sex <= 0
       |  |  |  |  slope <= 1: absent (2.0)
       |  |  |  |  slope > 1: present (4.0/1.0)
       |  |  |  sex > 0: present (14.0)
       thal > 3
       |  number_of_major_vessels <= 0
       |  |  exercise_induced_angina <= 0
       |  |  |  fasting_blood_sugar <= 0
       |  |  |  |  thal <= 6: absent (4.0)
       |  |  |  |  thal > 6
       |  |  |  |  |  age <= 52: present (9.0/2.0)
       |  |  |  |  |  age > 52: absent (11.0/2.0)
       |  |  |  fasting_blood_sugar > 0: absent (5.0)
       |  |  exercise_induced_angina > 0
       |  |  |  oldpeak <= 1.5
       |  |  |  |  serum_cholestoral <= 255: absent (6.0/1.0)
       |  |  |  |  serum_cholestoral > 255: present (4.0)
       |  |  |  oldpeak > 1.5: present (14.0)
       |  number_of_major_vessels > 0: present (65.0/6.0)
Number of Leaves  :     18
Size of the tree :      35
```

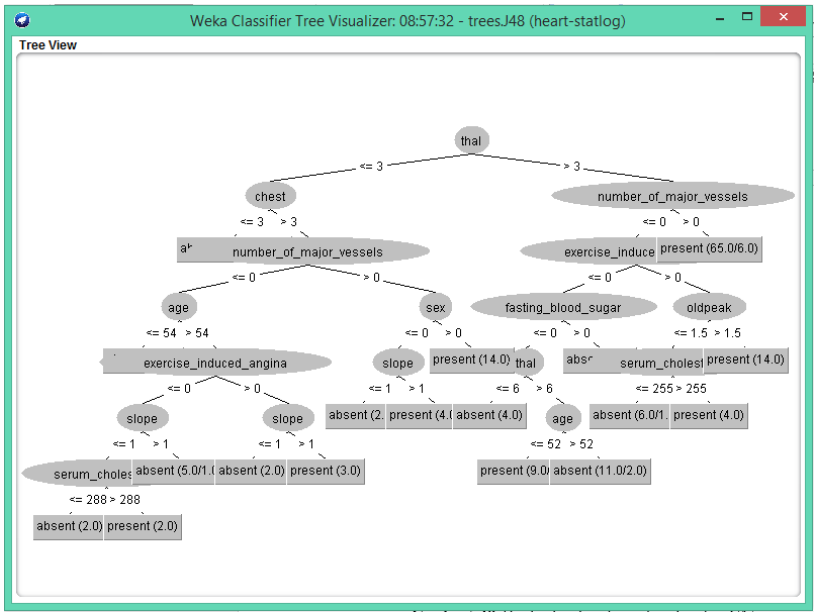
Gambar 5.34. Hasil *pessimistic prunning* pada weka

Pemangkasan algoritma C4.5 menghasilkan 18 *leaf* dan 35 *tree*, dimana definisi *leaf* merupakan simpul yang tidak mempunyai cabang lagi, sedangkan *tree* merupakan simpul yang mempunyai cabang. *Time to build model* digunakan untuk mengetahui waktu yang dibutuhkan untuk membangun model klasifikasi dari algoritma C4.5. Pada kasus ini waktu yang dibutuhkan yaitu 0,01 *seconds*. Hasil pohon keputusan dari pemangkasan algoritma C4.5 dapat ditampilkan pada jendela *weka classifier tree visualizer*.

Langkah untuk melihat pohon keputusan dapat dilakukan dengan cara klik kanan pada jendela *result list trees*. J48 → *visualize tree* seperti yang terlihat pada Gambar 5.35, maka pohon keputusan yang dihasilkan dapat dilihat pada Gambar 5.36.

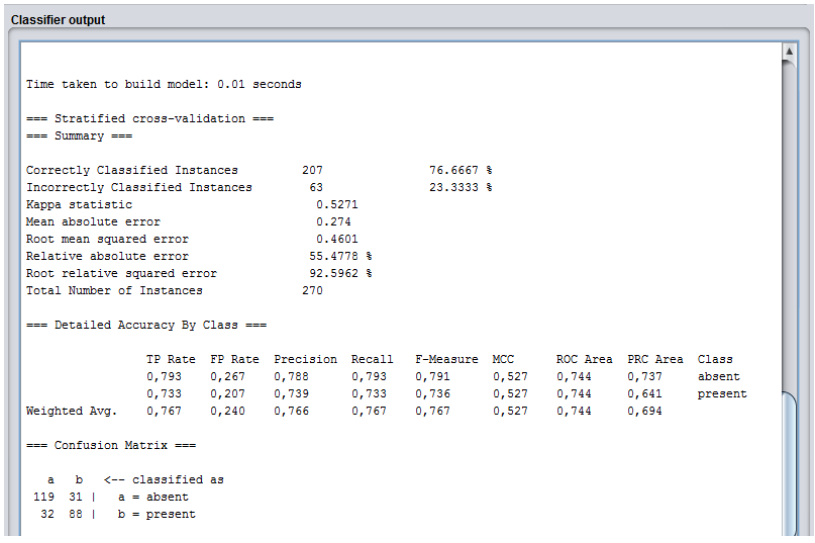


Gambar 5.35. Tampilan jendela *result list* (*right-click for options*) *trees J48*



Gambar 5.36. Hasil pohon keputusan dari algoritma C4.5

Selain waktu eksekusi, hasil dari klasifikasi algoritma C4.5 berupa *confusion matrix* dan akurasi seperti yang dapat dilihat pada Gambar 5.37.



Classifier output

```
Time taken to build model: 0.01 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         207               76.6667 %
Incorrectly Classified Instances        63               23.3333 %
Kappa statistic                          0.5271
Mean absolute error                      0.274
Root mean squared error                  0.4601
Relative absolute error                 55.4778 %
Root relative squared error             92.5962 %
Total Number of Instances              270

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,793    0,267    0,788      0,793    0,791      0,527    0,744     0,737     absent
                 0,733    0,207    0,739      0,733    0,736      0,527    0,744     0,641     present
Weighted Avg.    0,767    0,240    0,766      0,767    0,767      0,527    0,744     0,694

=== Confusion Matrix ===

   a   b   <-- classified as
 119  31 |   a = absent
  32  88 |   b = present
```

Gambar 5.37. Hasil *confusion matrix* dan akurasi algoritma C4.5 menggunakan *dataset* penyakit jantung

Hasil evaluasi dari *dataset* penyakit jantung yang menggunakan *k-fold cross validation* dengan *default* k=10 dapat dilihat pada Tabel 5.10.

Tabel 5.10. Hasil evaluasi algoritma C4.5 menggunakan 10-*fold cross validation*

No.	Spesifikasi Pengukuran	Nilai
1	*Corectly classified instance*	207 atau 76,67%
2	*Incorectly classified instance*	63 atau 23,33%
3	*Kappa atatistic*	0,5271
4	*Mean absolute error*	0,274
5	*Root mean squared error*	0,4601
6	*Relative absolute error*	55,4778%
7	*Root relative squared error*	92,5962%
8	*Total Number of instance*	270

Algoritma C4.5 menghasilkan nilai *detailed accuracy by class* seperti yang disajikan pada Tabel 5.11**.** Sedangkan hasil *confusion matrix* disajikan pada Tabel 5.12.

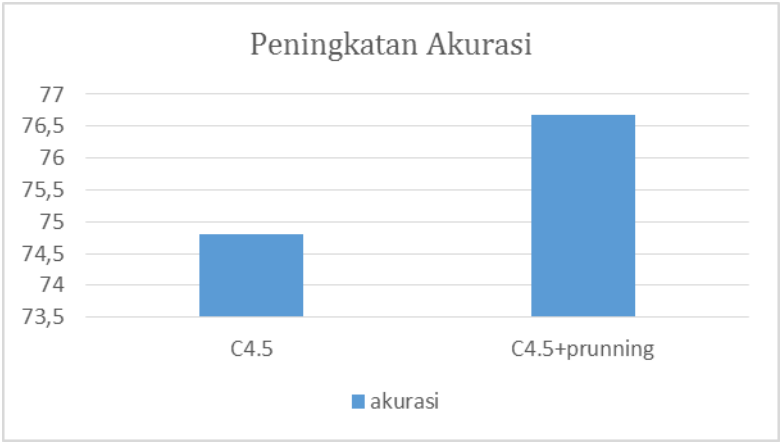
Tabel 5.11. Hasil *detailed accuracy by class*

	TP rate	FP rate	Precision	Racall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,793	0,267	0,788	0,793	0,791	0,527	0,744	0,737	absent
	0,733	0,207	0,739	0,733	0,736	0,527	0,744	0,641	present
Weighted Avg.	0,767	0,240	0,766	0,767	0,767	0,527	0,744	0,694	

Tabel 5.12. Hasil *confusion matrix* algoritma C4.5

A	b	Classified as
119	31	a=absent
32	88	b=present

Penggunaan *pessimistic prunning* mempengaruhi pada akurasi algoritma C4.5 dalam mendiagnosis penyakit jantung. Peningkatan akurasi yang dialami yaitu sebesar 1,86%. Grafik peningkatan akurasi dapat dilihat pada Gambar 5.38.

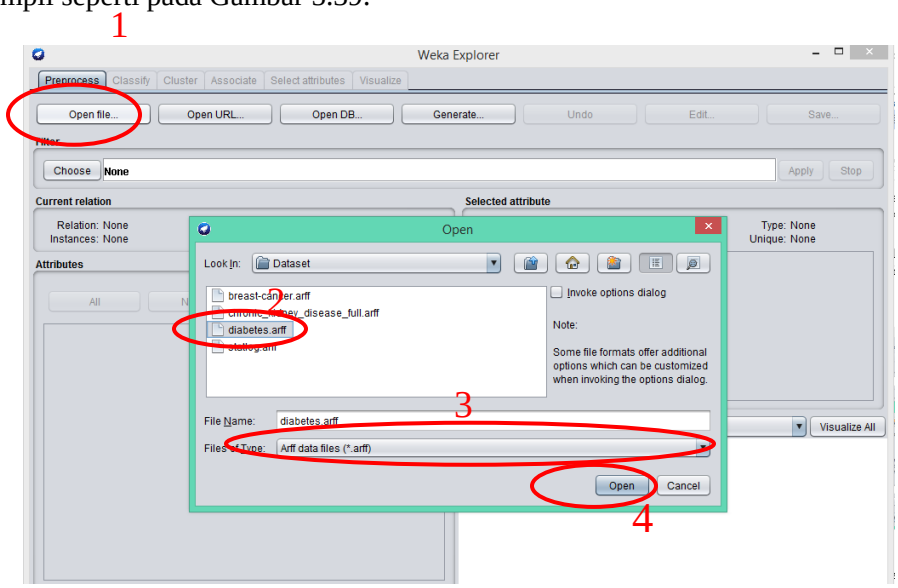


Gambar 5.38. Grafik perbandingan akurasi C4.5 dan C4.5 menggunakan *pessimistic prunning*

5.3 *Dizcretization* dan Teknik *Bagging* pada Algoritma C4.5

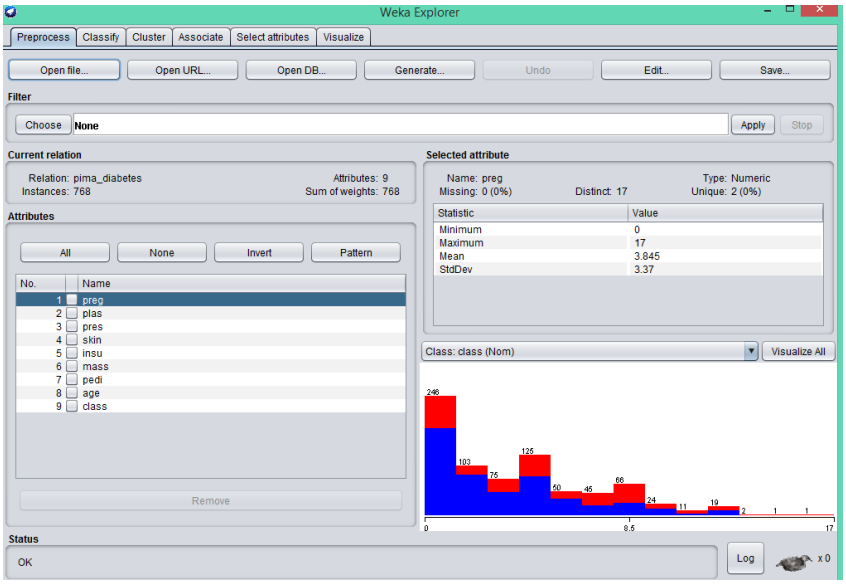
Penerapan *discretization* dan teknik *bagging* untuk meningkatkan akurasi klasifikasi berbasis *ensemble* pada algoritma C4.5 dalam mendiagnosis diabetes di proses pada *tools* weka. Langkah-langkah proses klasifikasi algoritma C4.5 menggunakan weka, yaitu sebagai berikut.

Siapkan dataset.arff yang akan digunakan untuk diproses klasifikasi, pada kasus ini akan menggunakan *dataset* diabetes. Setelah *dataset* sudah siap klik menu *open file* pada menu *Preprocess* yang ada di tampilan *weka explorer*, maka akan tampil seperti pada Gambar 5.39.



Gambar 5.39. Memasukkan *dataset diabetes.arff*

Setelah *dataset* diabetes dimasukkan ke weka. Maka grafik data dapat ditampilkan pada Gambar 5.40.

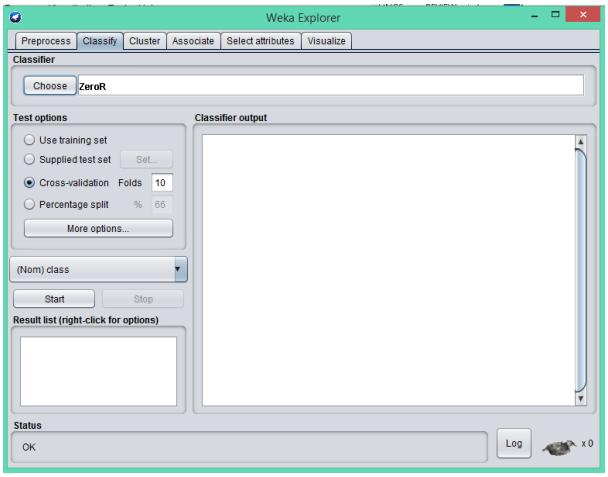


Gambar 5.40. Hasil grafik isi *dataset* diabetes setelah diinputkan

Pada Gambar 5.40. menjelaskan rincian dari *dataset* penyakit jantung, yaitu terdiri dari 9 atribut dan 768 *instance*.

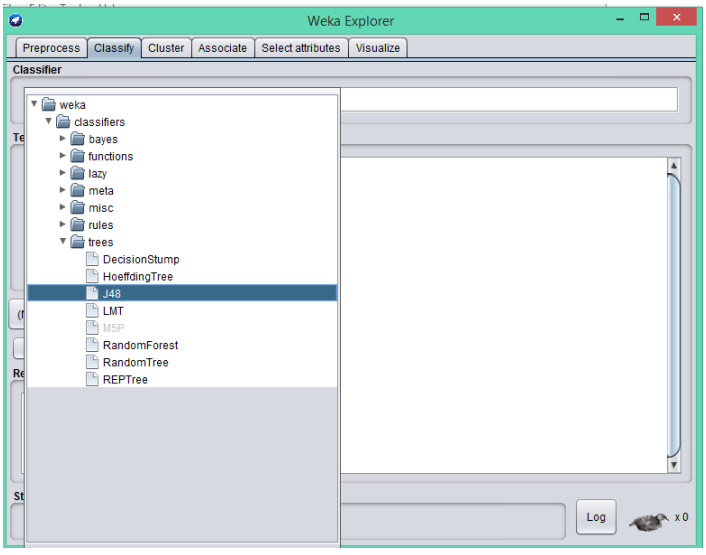
5.3.1 Pengujian dan hasil klasifikasi *decision tree* dengan algoritma C4.5 untuk mendiagnosis *diabetes*

Proses klasifikasi algoritma C4.5 di weka dapat dilakukan setelah *dataset* diabetes diinputkan melalui menu *Preprocess*, maka langkah selanjutnya yaitu menuju ke menu *classify*. Tampilan menu *classify* ditunjukkan pada Gambar 5.41.



Gambar 5.41. Tampilan menu *classify*

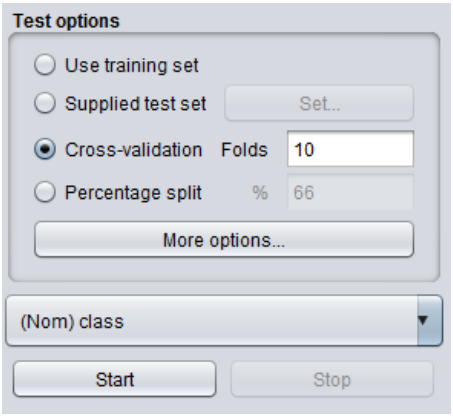
Pada menu *classify* kemudian pilih menu *choose* untuk memilih algoritma klasifikasi *decision tree* C4.5. Algoritma C4.5 dalam weka nama lainnya yaitu J48 seperti yang terlihat pada Gambar 5.42.



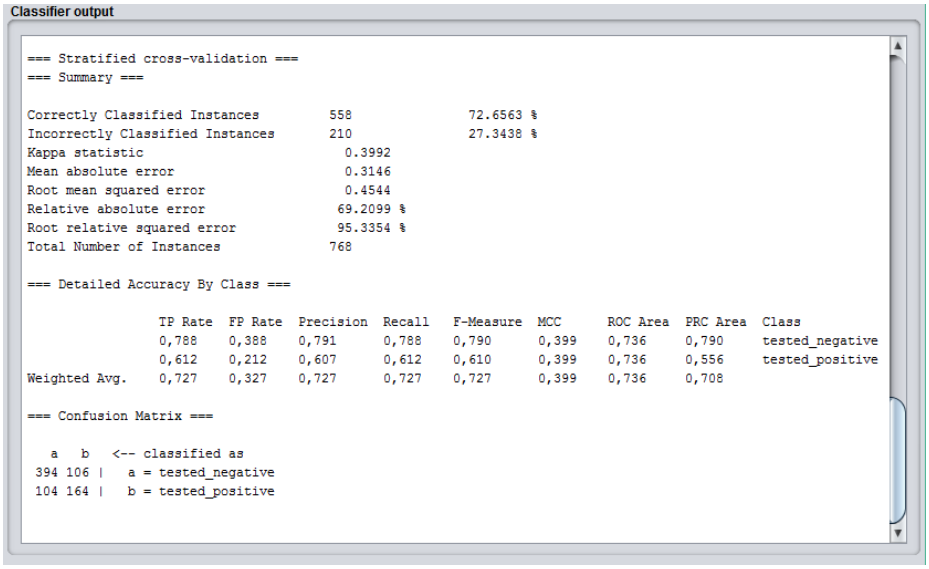
Gambar 5.42. Memilih klasifikasi algoritma C4.5

Langkah selanjutnya yaitu memilih proses pengujian 10-*fold cross validation* kemudian untuk memulai proses klasifikasi pilih menu *start* seperti yang

ditunjukkan pada Gambar 5.43. Dan hasil klasifikasi akan muncul di jendela *classifier output* seperti pada Gambar 5.44.

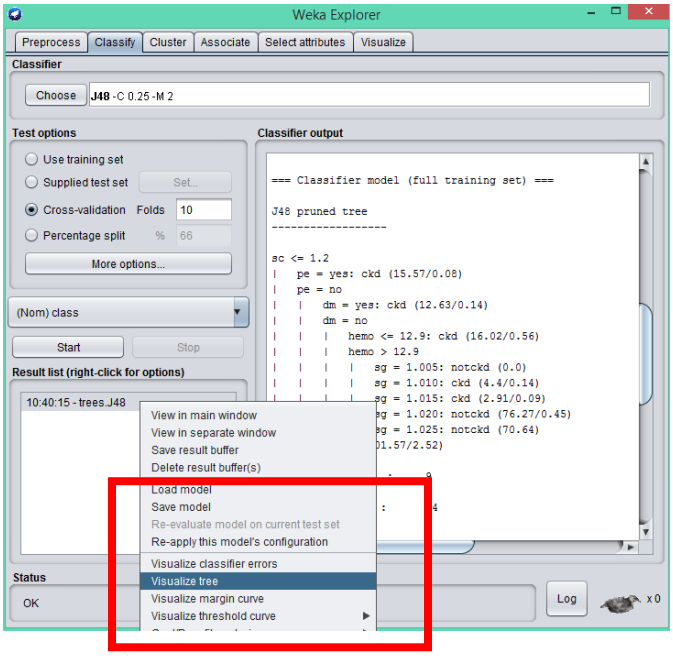


Gambar 5.43. Proses evaluasi klasifikasi

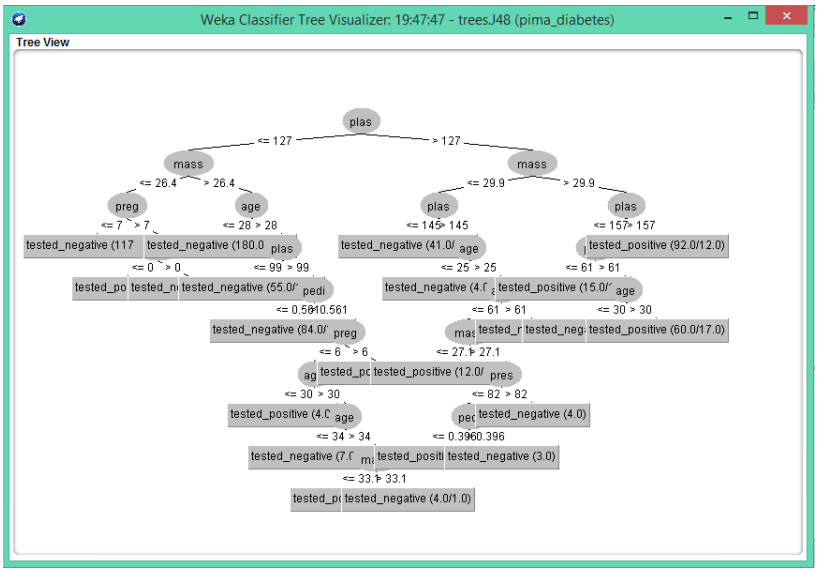


Gambar 5.44. Hasil dari klasifikasi algoritma C4.5 pada diagnosis diabetes

Langkah untuk melihat pohon keputusan dapat dilakukan dengan cara klik kanan pada jendela *result list trees*. J48 → *visualize tree* seperti yang terlihat pada Gambar 5.45, maka pohon keputusan yang dihasilkan dapat dilihat pada Gambar 5.46.

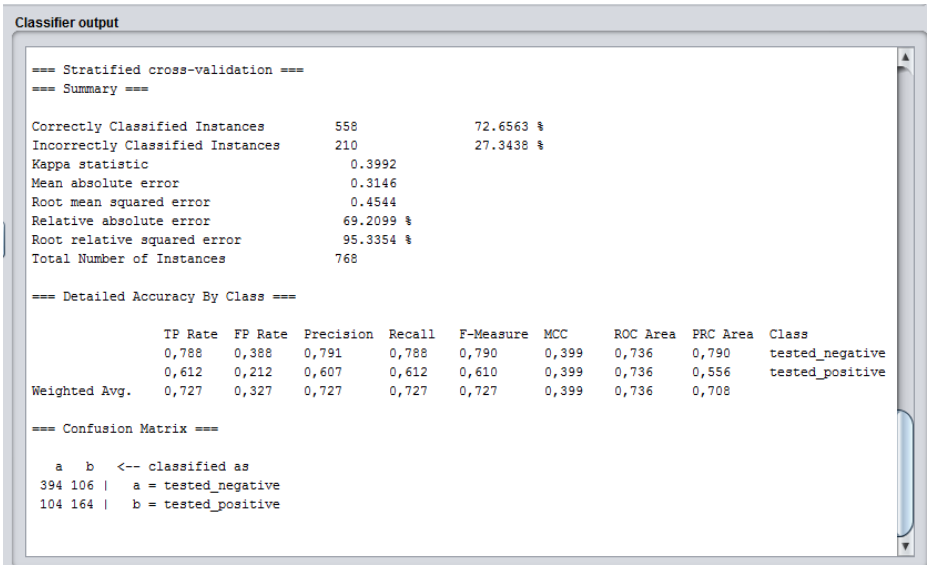


Gambar 5.45. Tampilan jendela *result list* (*right-click for options*) *trees* C4.5



Gambar 5.46. Hasil pohon keputusan dari algoritma C4.5 pada diagnosis diabetes

Untuk waktu eksekusi, hasil dari klasifikasi algoritma C4.5 untuk mendiagnosis diabetes mempunyai hasil akurasi dengan berupa *confusion matrix* dan akurasi dapat dilihat pada Gambar 5.47.



Gambar 5.47. Hasil *confusion matrix* dan akurasi algoritma C4.5 menggunakan *dataset* diabetes

Hasil evaluasi dari *dataset* penyakit jantung yang menggunakan *k-fold cross validation* dengan *default* k=10 dapat dilihat pada Tabel 5.13.

Tabel 5.13. Hasil evaluasi algoritma C4.5 menggunakan 10-*fold cross validation*

No.	Spesifikasi Pengukuran	Nilai
1	*Corectly classified instance*	558 atau 72,66%
2	*Incorectly classified instance*	210 atau 27,34%
3	*Kappa atatistic*	0,3992
4	*Mean absolute error*	0,3146
5	*Root mean squared error*	0,4544
6	*Relative absolute error*	69,2099%
7	*Root relative squared error*	95,3354%
8	*Total Number of instance*	768

Algoritma C4.5 menghasilkan nilai *detailed accuracy by class* seperti yang disajikan pada Tabel 5.14**.** Sedangkan hasil *confusion matrix* disajikan pada Tabel 5.15.

Tabel 5.14. Hasil *detailed accuracy by class*

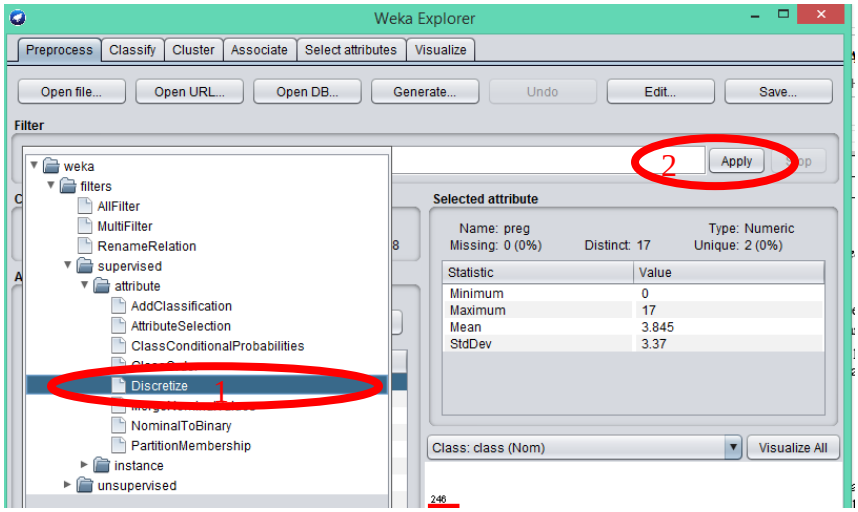
	TP rate	FP rate	Precision	Racall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,788	0,388	0,791	0,788	0,790	0,399	0,736	0,790	tested_neg
	0,612	0,212	0,607	0,612	0,610	0,399	0,736	0,556	tested_pos
Weighted Avg.	0,727	0,327	0,727	0,727	0,727	0,399	0,736	0,708	

Tabel 5.15. Hasil *confusion matrix* algoritma C4.5

a	b	Classified as
394	106	a=tested_neg
104	164	b=tested_pos

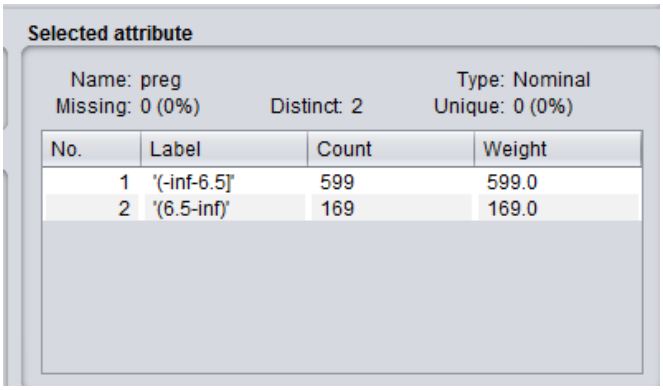
5.3.2 Pengujian dan hasil klasifikasi Algoritma *decision tree* C4.5 menggunakan *dizcretization*

Penerapan *dizcretization* dapat dilakukan pada *Preprocess* sebelum melakukan klasifikasi *decision tree* C4.5. Pada menu *Preprocess* bagian filter pilih menu *choose* untuk memilih *dizcretization* untuk penanganan pada atribut yang mempunyai sifat kontinyu kemudian klik *apply*. Proses *dizcretization* dapat dilihat pada Gambar 5.48.



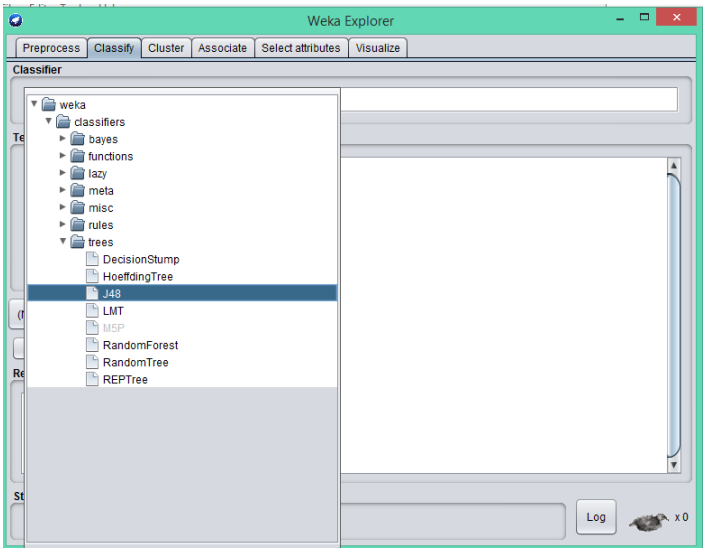
Gambar 5.48. Proses *discretization* pada algoritma C4.5

Hasil dari *discretization* dapat dilihat pada Gambar 5.49.



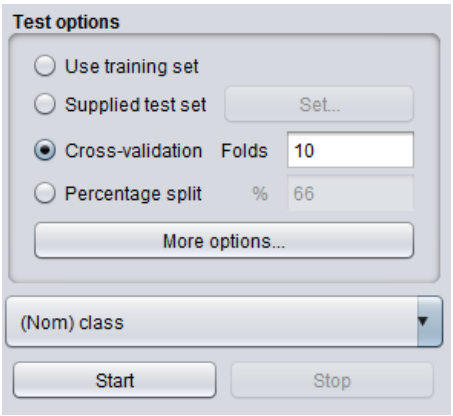
Gambar 5.49. Hasil *discretization dataset* diabetes

Pada Gambar 5.49 menjelaskan bahawa pada *dataset* diabetes mempunyai 2 atribut yang bersifat kontinyu. Setelah melakukan proses *discretization* langkah selanjutnya yaitu proses klasifikasi dengan cara klik menu *classify* kemudian *choose* dan pilih algoritma J48 sebagai pengganti algoritma C4.5 yang ada pada weka. Proses klasifikasi C4.5 dapat dilihat pada Gambar 5.50.



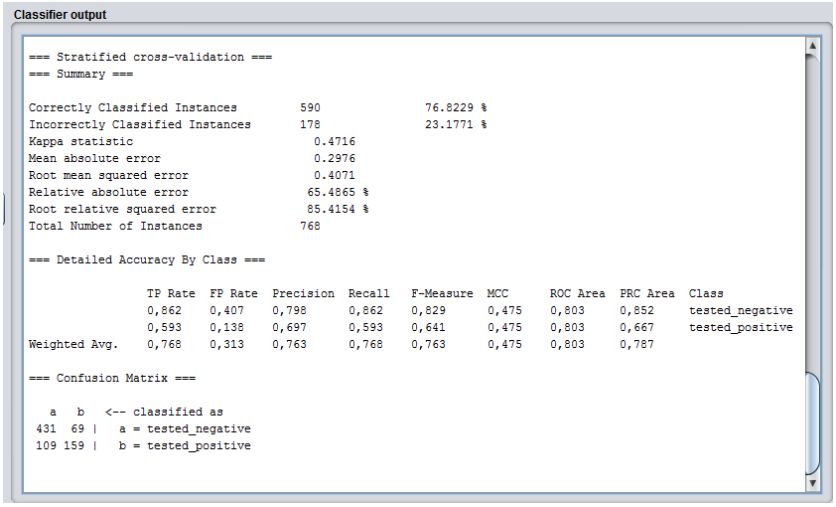
Gambar 5.50. Memilih klasifikasi algoritma C4.5

Langkah selanjutnya yaitu memilih proses pengujian *10-fold cross validation* kemudian untuk memulai proses klasifikasi pilih menu *start* seperti yang ditunjukkan pada Gambar 5.51.



Gambar 5.51. Proses evaluasi klasifikasi algoritma C4.5 dan diskrit

Hasil yang didapat setelah algoritma C4.5 didiskrit yaitu ditampilkan pada Gambar 5.52.



Gambar 5.52. Hasil klasifikasi algoritma C4.5 menggunakan *discretization*

Hasil evaluasi dari *dataset* diabetes yang menggunakan *k-fold cross validation* dengan *default* k=10 dapat dilihat pada Tabel 5.16.

Tabel 5.16. Hasil evaluasi algoritma C4.5 menggunakan 10-*fold cross validation*

No.	Spesifikasi Pengukuran	Nilai
1	*Corectly classified instance*	590 atau 76,82%
2	*Incorectly classified instance*	178 atau 23,18%
3	*Kappa atatistic*	0,4716
4	*Mean absolute error*	0,2976
5	*Root mean squared error*	0,4071
6	*Relative absolute error*	65,4865%
7	*Root relative squared error*	85,4154%
8	*Total Number of instance*	768

Algoritma C4.5 menghasilkan nilai *detailed accuracy by class* seperti yang disajikan pada Tabel 5.17**.** Sedangkan hasil *confusion matrix* disajikan pada Tabel 5.18.

Tabel 5.17. Hasil *detailed accuracy by class*

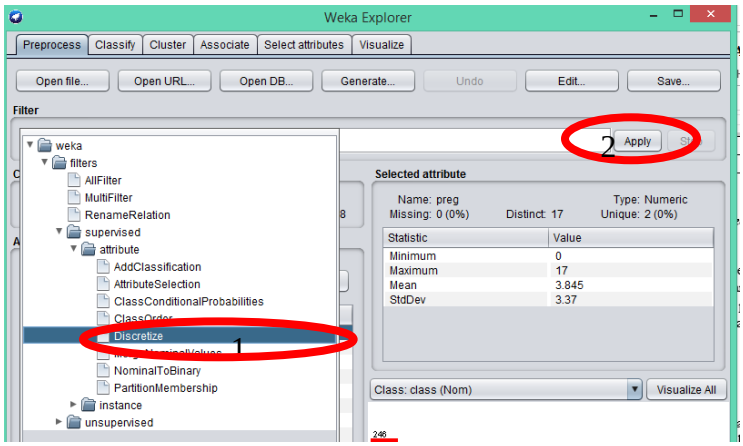
	TP rate	FP rate	Precision	Racall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,862	0,407	0,798	0,862	0,829	0,475	0,803	0,852	test_pos
	0,593	0,138	0,697	0,593	0,641	0,475	0,803	0,667	test_neg
Weighted Avg.	0,768	0,313	0,763	0,768	0,763	0,475	0,803	0,787	

Tabel 5.18. Hasil *confusion matrix* algoritma C4.5

A	b	Classified as
431	69	a=test_pos
109	159	b=test_neg

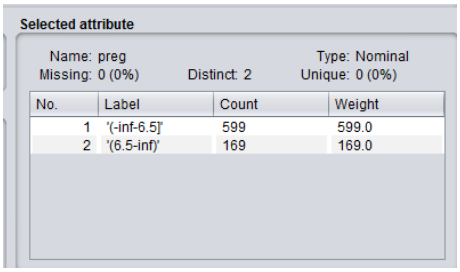
5.3.3 Pengujian dan hasil klasifikasi algoritma *decision tree* C4.5 menggunakan *bagging* dan *dizcretization* pada *dataset diabetes*

Penerapan *dizcretization* dapat dilakukan pada *Preprocess* sebelum melakukan klasifikasi *decision tree* C4.5. Pada menu *Preprocess* bagian filter pilih menu *choose* untuk memilih *dizcretization* untuk penanganan pada atribut yang mempunyai sifat kontinyu kemudian klik *apply*. Proses *dizcretization* dapat dilihat pada Gambar 5.53.



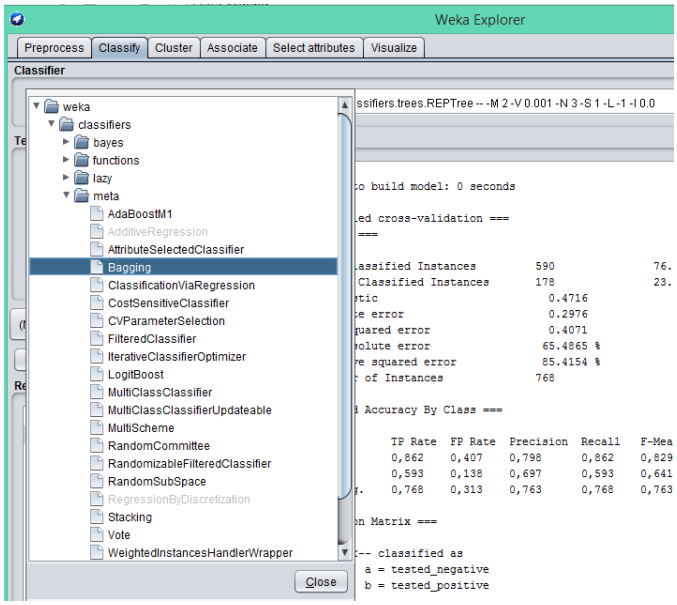
Gambar 5.53. Proses *discretization* pada algoritma C4.5

Hasil dari *discretization* dapat dilihat pada Gambar 5.54.



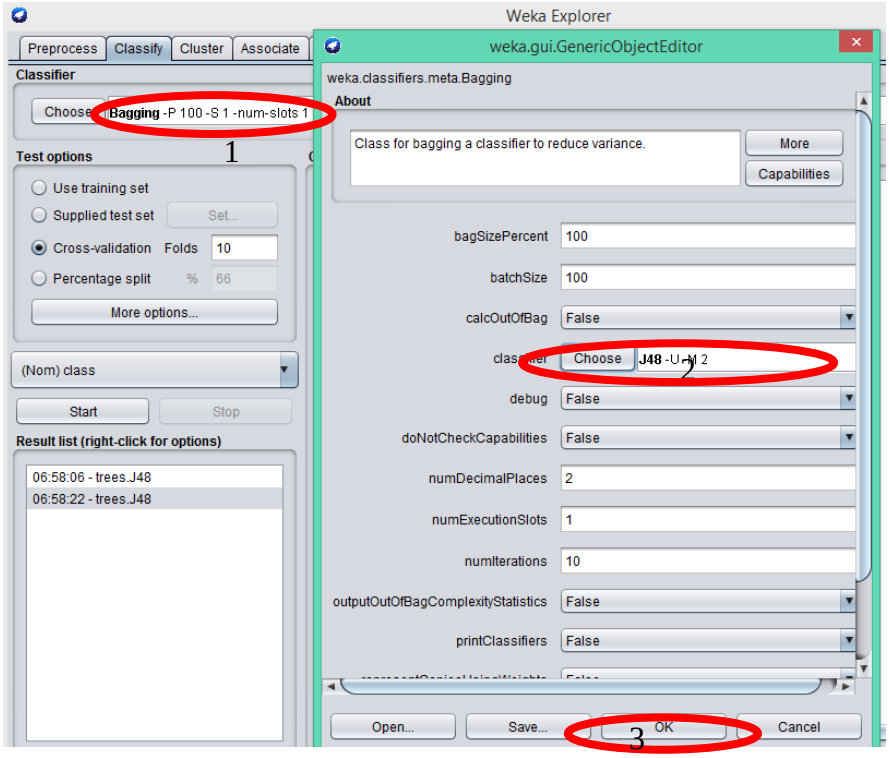
Gambar 5.54. Hasil *discretization* dataset diabetes

Pada Gambar 5.54 menjelaskan bahawa pada *dataset* diabetes mempunyai 2 atribut yang bersifat kontinyu. Setelah melakukan proses *discretization* langkah selanjutnya yaitu proses klasifikasi dengan cara klik menu *classify* kemudian *choose* dan pilih meta untuk menerapkan *ensemble learning bagging* yang ditunjukkan pada Gambar 5.55.



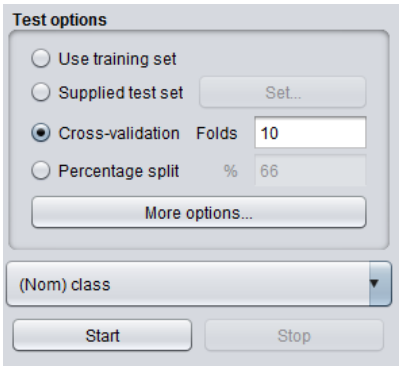
Gambar 5.55. Memilih klasifikasi algoritma C4.5

Kemudian pada kolom *choose* pilih *classifier* algoritma J48 sebagai pengganti algoritma C4.5 yang ada pada weka lalu klik ok. Proses klasifikasi c4.5 menggunakan *bagging* dapat dilihat pada Gambar 5.56.



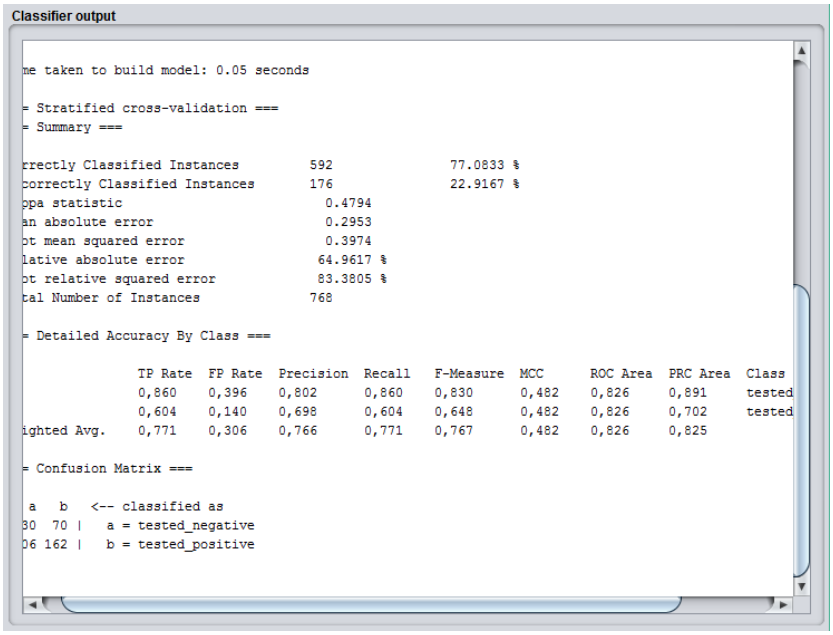
Gambar 5.56. Proses klasifikasi C4.5 menggunakan *bagging*

Langkah selanjutnya yaitu memilih proses pengujian 10-*fold cross validation* kemudian untuk memulai proses klasifikasi pilih menu *start* seperti yang ditunjukkan pada Gambar 5.57.



Gambar 5.57. Proses evaluasi klasifikasi algoritma C4.5 menggunakan bagging dan diskrit

Hasil yang didapat setelah algoritma C4.5 menggunakan *bagging* dan diskrit dapat dilihat pada Gambar 5.58.



Gambar 5.58. Hasil klasifikasi algoritma C4.5 menggunakan *bagging* dan *discretization*

Hasil evaluasi dari klasifikasi algoritma C4.5 menggunakan *bagging* dan *discretization* pada *dataset* diabetes dengan *k-fold cross validation default* k=10 dapat dilihat pada Tabel 5.19.

Tabel 5.19. Hasil evaluasi algoritma C4.5 menggunakan 10-*fold cross validation*

No.	Spesifikasi Pengukuran	Nilai
1	*Corectly classified instance*	592 atau 77,08%
2	*Incorectly classified instance*	176 atau 22,92%
3	*Kappa atatistic*	0,4794
4	*Mean absolute error*	0,2953
5	*Root mean squared error*	0,3974
6	*Relative absolute error*	64,9617%

7	*Root relative squared error*	83,3805%
8	*Total Number of instance*	768

Algoritma C4.5 menghasilkan nilai *detailed accuracy by class* seperti yang disajikan pada Tabel 5.20. Sedangkan hasil *confusion matrix* disajikan pada Tabel 5.21.

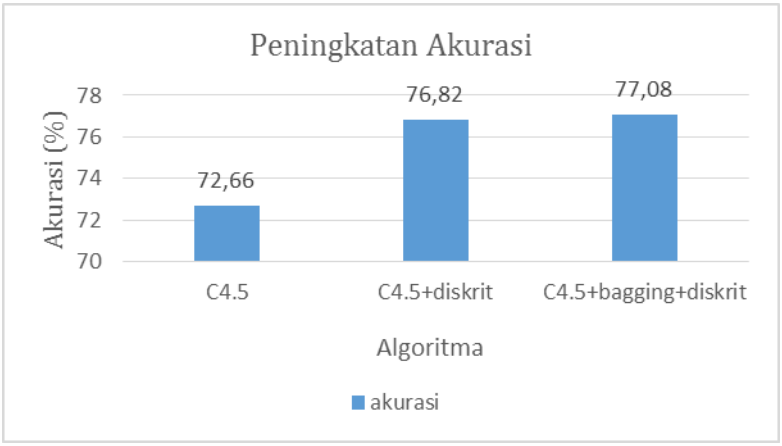
Tabel 5.20. Hasil *detailed accuracy by class*

	TP rate	FP rate	Precision	Racall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,860	0,396	0,802	0,860	0,830	0,482	0,826	0,891	test_pos
	0,604	0,140	0,698	0,604	0,648	0,482	0,826	0,702	test_neg
Weighted Avg.	0,771	0,306	0,766	0,771	0,767	0,482	0,826	0,825	

Tabel 5.21. Hasil *confusion matrix* algoritma C4.5

A	b	Classified as
430	70	a=test_pos
106	162	b=test_neg

Penggunaan bagging dan *discretization* mempengaruhi peningkatan akurasi pada algoritma C4.5 dalam mendiagnosis penyakit diabetes. Peningkatan akurasi yang dialami yaitu sebesar 4,42% dari 72,66% menjadi 77,08%, Grafik peningkatan akurasi dapat dilihat pada Gambar 5.59.

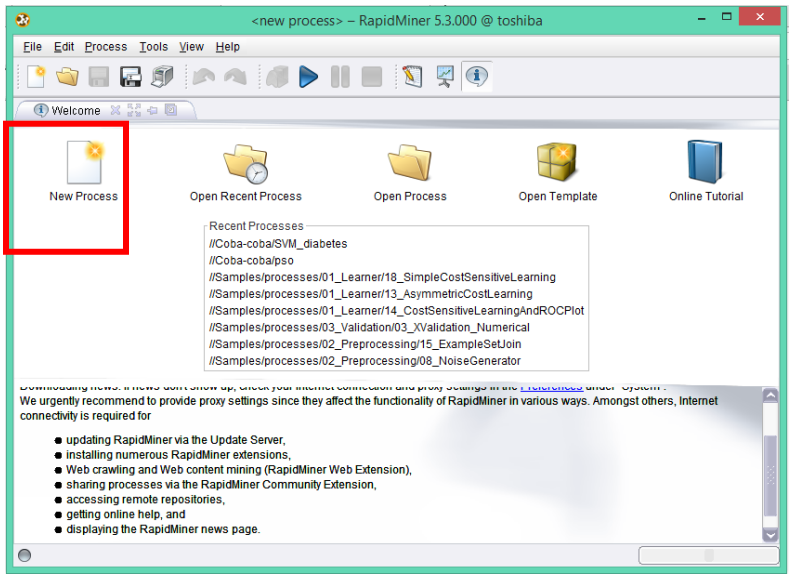


Gambar 5.59. Grafik peningkatan akurasi algoritma C4.5 menggunakan *discretization* dan *bagging*

5.4 Algoritma C4.5 Berbasis *Particle Swarm Optimization*

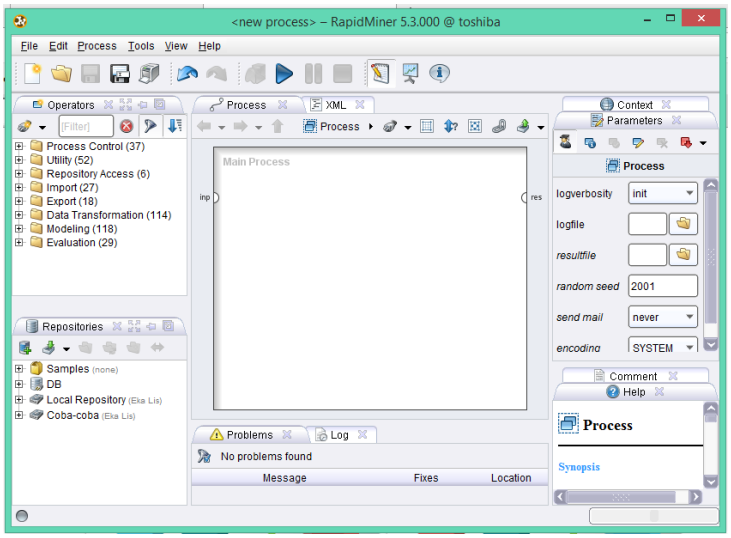
Penerapan *particle swarm optimization* pada algoritma C4.5 untuk mendiagnosis kanker payudara menggunakan *tools RapidMiner*. Langkah-langkah proses klasifikasi *particle swarm optimization* pada algoritma C4.5 menggunakan *RapidMiner*, yaitu sebagai berikut.

Langkah awal untuk menjalankan *process RapidMiner* pada halaman *welcome perspective* untuk membuat lembar kerja baru maka klik *new process* seperti yang dilihat pada Gamabr 5.60.



Gambar 5.60. Halaman *welcome perspective*

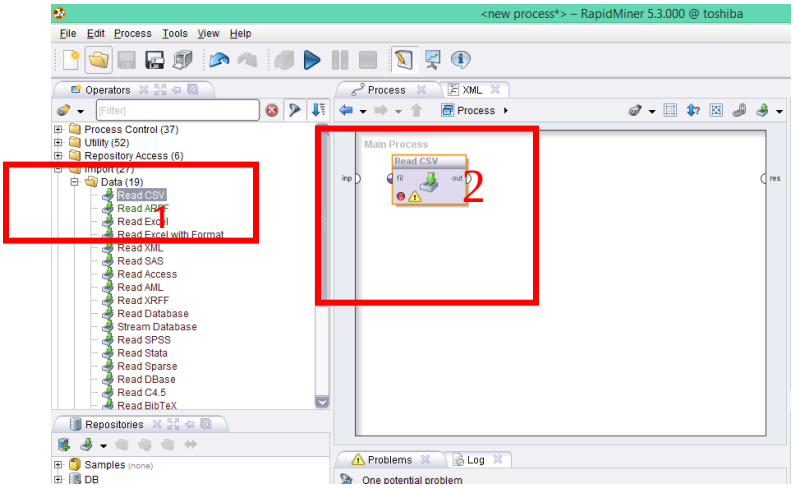
Setelah itu maka akan tampil halaman proses seperti yang terlihat pada Gambar 5.61.



Gambar 5.61. Halaman *new process*

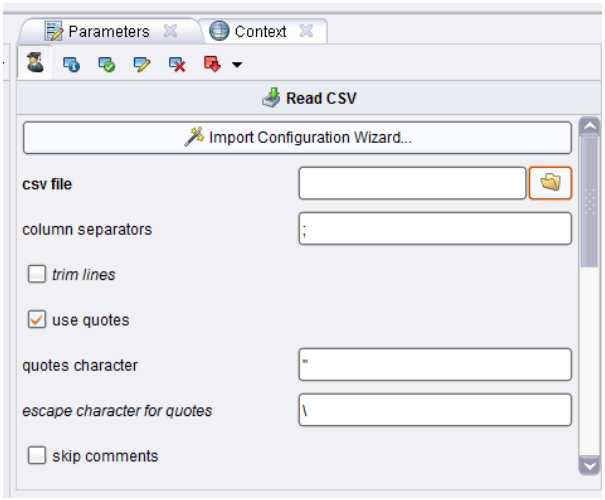
Sebelum meng-*import* data maka siapkan *dataset.csv* terlebih dahulu untuk proses klasifikasi, pada kasus ini akan menggunakan *dataset breast cancer* atau kanker payudara.

Langkah selanjutnya yaitu *import* data dalam *main process* dengan cara pada *operator view* pilih *import* → Data → *Read CSV*, apabila data yang digunakan berekstensi .csv. Pada kasus ini penulis menggunakan data yang berekstensi .csv. Kemudian *drag Read CSV* ke *process view* maka akan terlihat seperti Gambar 5.62.



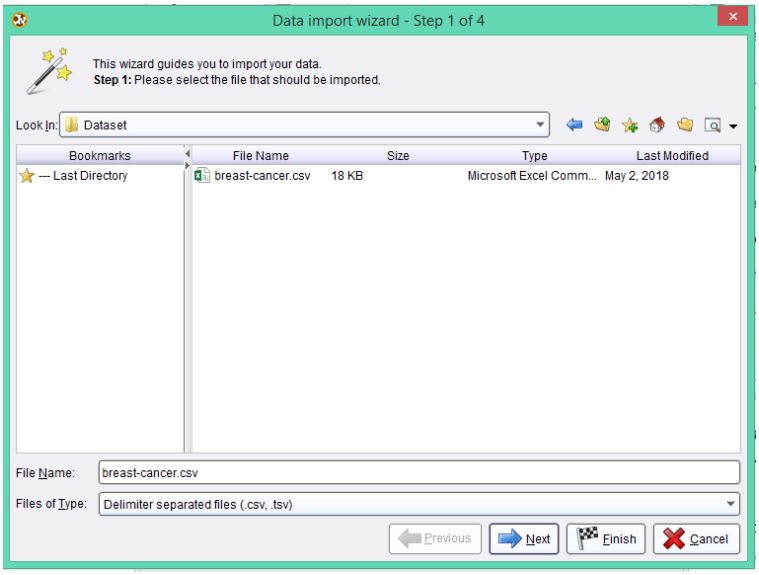
Gambar 5.62. Tampilan *process read* data

Langkah selanjutnya yaitu pada *process view Read CSV* klik dua kali maka pada bagian *parameter perspective* akan muncul seperti pada Gambar 5.63.



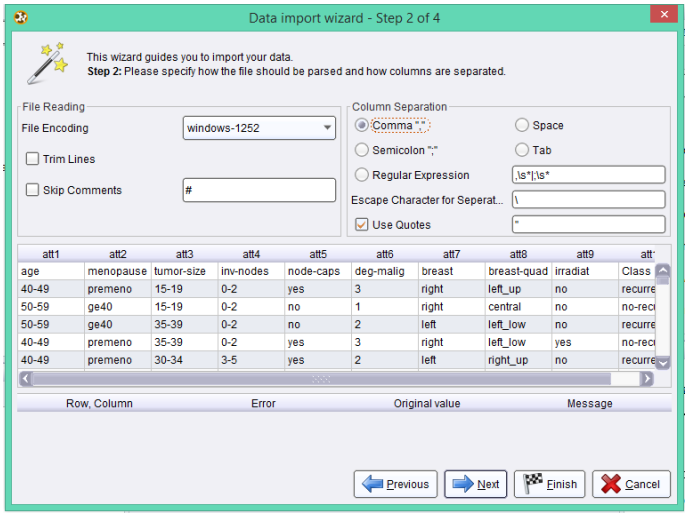
Gambar 5.63. Tampilan *context Read CSV*

Setelah itu, maka klik *import configuration wizard* untuk memasukkan data yang sudah disiapkan dan mengatur pelabelan pada data. Setelah diklik maka akan muncul jendela baru yang terdiri dari *step* 1 dari 4 *step data import wizard* seperti yang terlihat pada Gambar 5.64.



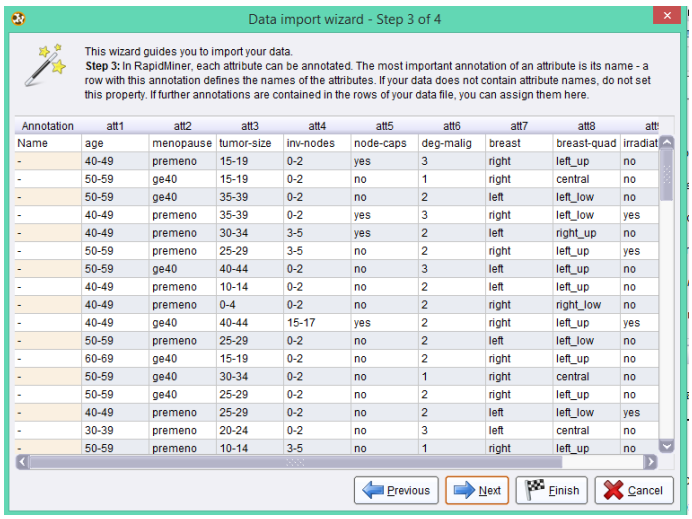
Gambar 5.64. *Step 1 of 4 import wizard*

Pada *step* 1 cari file .csv dengan cara klik pada *look in*. Setelah menemukan file .csv yang dibutuhkan maka selanjutnya klik *next* maka akan tampil *step 2 of 4 import wizard* seperti yang ditampilkan pada Gambar 5.65.



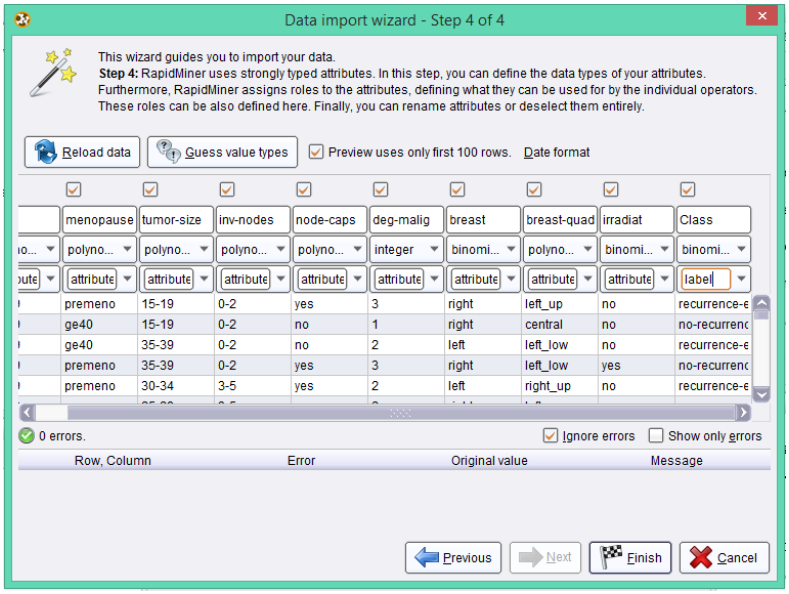
Gambar 5.65. *Step 2 of 4 import wizard*

Klik *next*, maka selanjutnya akan menuju ke *step* 3 *of* 4 yang memberikan informasi data yang digunakan seperti yang terlihat pada Gambar 5.66.



Gambar 5.66. *Step 3 of 4 import wizard*

Klik *next* untuk melanjutkan *step* 4 yaitu memberikan tipe label data pada *class* tabel data yang digunakan. Seperti yang terlihat pada Gambar 5.67.

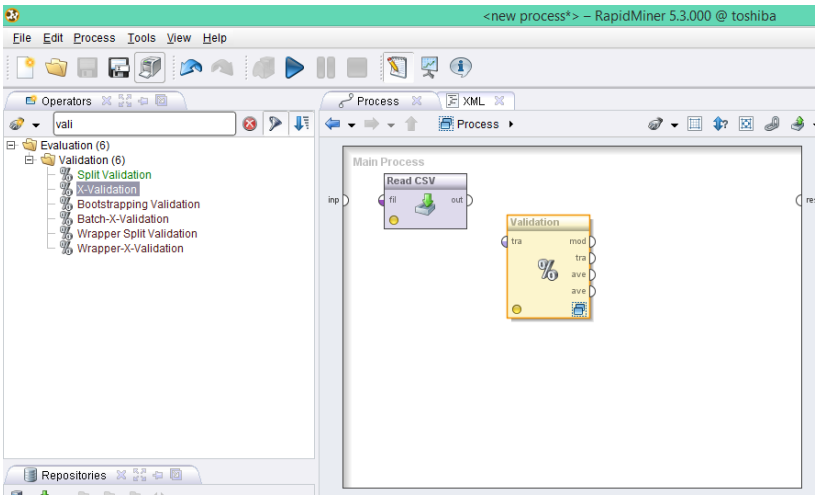


Gambar 5.67. *Step 4 of 4 import wizard*

Kemudian klik *finish*. Setelah itu melakukan proses klasifikasi antara algoritma C4.5 dengan penerapan *particle swarm optimization* pada algoritma C4.5.

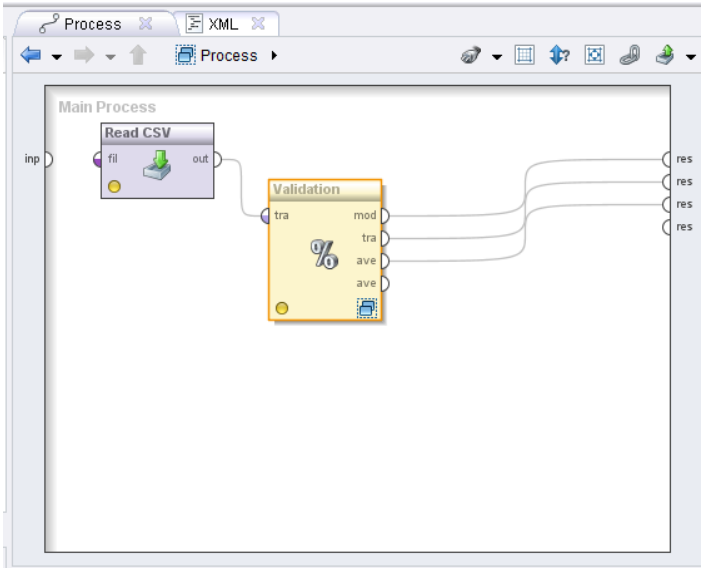
5.4.1 Pengujian dan hasil klasifikasi *decision tree* dengan algoritma C4.5 untuk mendiagnosis *breast cancer*

Proses klasifikasi algoritma C4.5 di *RapidMiner* dapat dilakukan setelah *dataset breast cancer* di*import* sesuai dengan langkah-langkah *import data* yang sudah dijelaskan, maka langkah selanjutnya yaitu pada *main process drag validation* dari *operator view* untuk memvalidasi proses klasifikasi seperti yang ditampilkan pada Gambar 5.68.



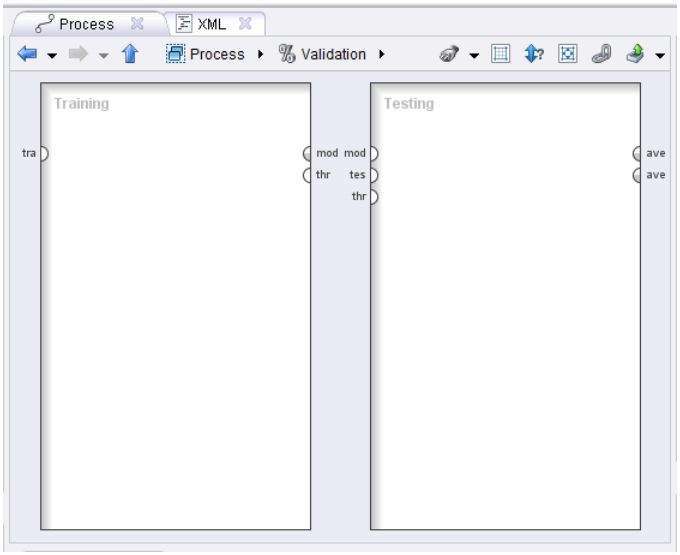
Gambar 5.68. Tampilan *main process validation*

Langkah selanjutnya hubungkan setiap *node* seperti yang ditampilkan pada Gambar 5.69.



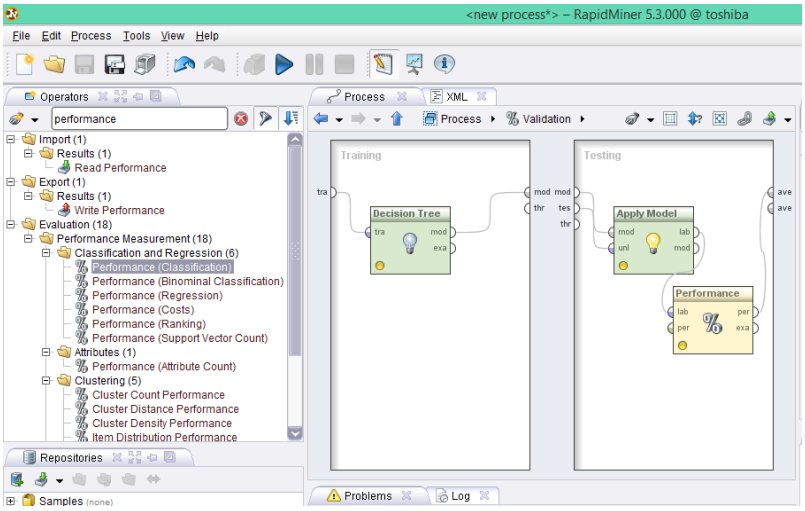
Gambar 5.69. Tampilan hubungan *node validation* pada *main process*

Kemudian klik dua kali pada *validation* maka akan tampil pembagian dua halaman yaitu *training* dan *testing* seperti yang ditunjukkan pada Gambar 5.70.



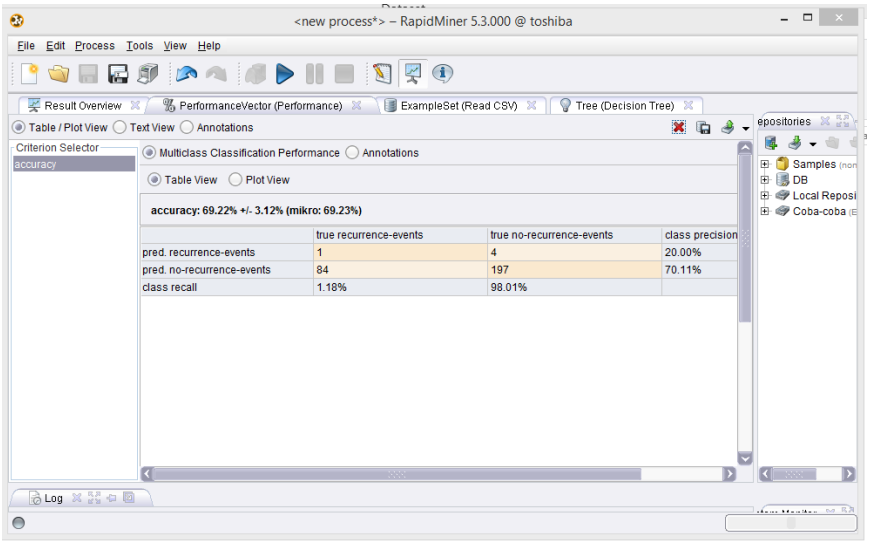
Gambar 5.70. Tampilan halaman *validation process*

Langkah selanjutnya yaitu proses klasifikasi *decision tree* pada bagian *training view* dan pengujian model pada *testing view* dengan cara *drag decision tree* dari *operator view* menuju *training view*. Kemudian *drag apply model* dan *performance* menuju *testing view* untuk menunjukkan hasil dari proses klasifikasi *desicion tree*. Setelah itu hubungkan setiap *node* seperti pada Gambar 5.71.



Gambar 5.71. Tampilan *proses validation desicion tree*

Kemudian klik *run* untuk memproses klasifikasi pada *desicion tree*, maka hasil dapat dilihat pada Gambar 5.72.



Gambar 5.72. Hasil proses klasifikasi C4.5 menggunakan *dataset* kanker payudara

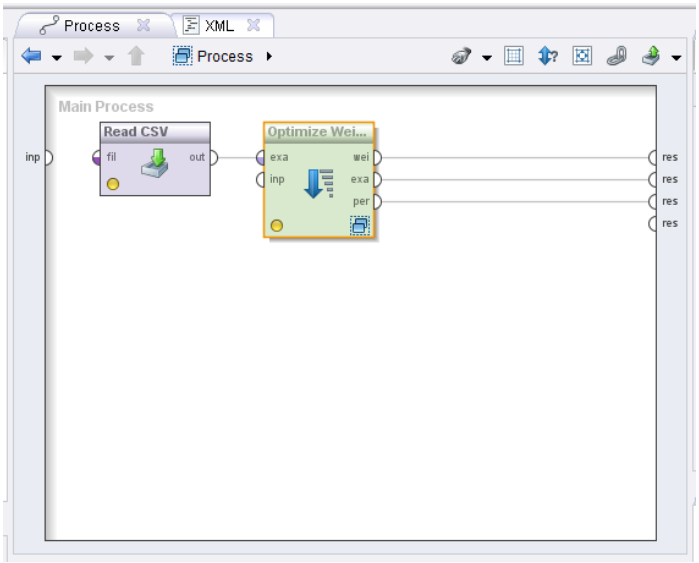
Akurasi yang didapat pada algoritma C4.5 menggunakan *dataset* kanker payudara sebesar 69,22%. Hasil evaluasi dari *dataset* kanker payudara yang menggunakan algoritma C4.5 pada *RapidMiner* dapat dilihat pada Tabel 5.22.

Tabel 5.22. Hasil evaluasi dari *dataset* kanker payudara yang menggunakan algoritma C4.5 pada *RapidMiner*

	True recure	True no-recure	Classified precission
Pred-recure	1	4	20,00%
Pred-no recure	84	197	70,11%
Class recall	1,18%	98,01%	

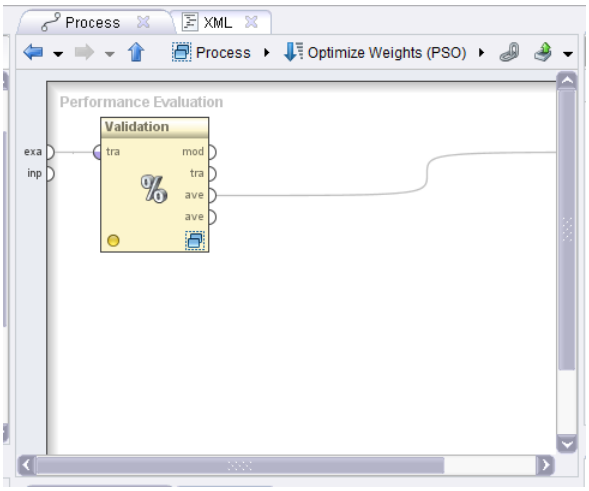
4.4.2 Pengujian dan hasil klasifikasi Algoritma *decision tree* C4.5 menggunakan *particle swarm optimization* untuk mendiagnosis kanker payudara

Proses klasifikasi algoritma C4.5 menggunakan *particle swarm optimization* untuk mendiagnosis kanker payudara di *RapidMiner* dapat dilakukan setelah *dataset* kanker payudara di-*import* sesuai dengan langkah-langkah *import data* yang sudah dijelaskan, Setelah data di*import* maka langkah selanjutnya yaitu *drag particle swarm optimization* dari *operator view* menuju *main process* dan hubungkan *node-node* seperti yang ditampilkan pada Gambar 5.73.



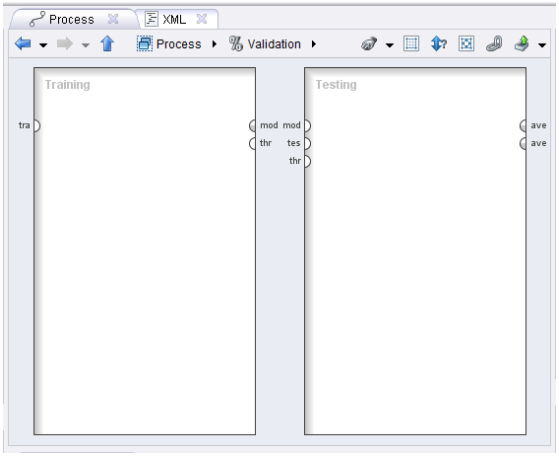
Gambar 5.73. Tampilan *main process*

Langkah selanjutnya yaitu melakukan proses validasi pada *particle swarm optimization* dengan cara klik dua kali pada *optimize weighted* maka akan tampil *performance evaluation*, kemudain *drag validation* dari *operator view* serta hubungkan *node* seperti pada Gambar 5.74.



Gambar 5.74. Tampilan *performance evaluation*

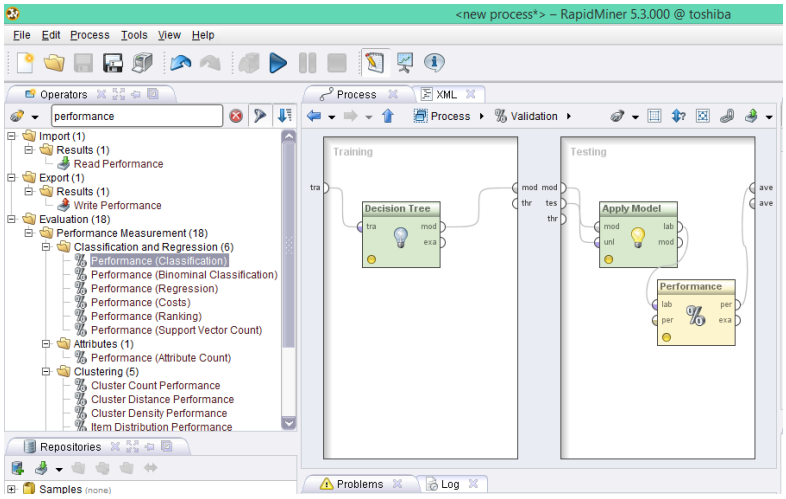
Kemudian klik dua kali pada *validation* maka akan tampil pembagian dua halaman yaitu *training* dan *testing* seperti yang ditunjukkan pada Gambar 5.75.



Gambar 5.75. Tampilan halaman *validation process*

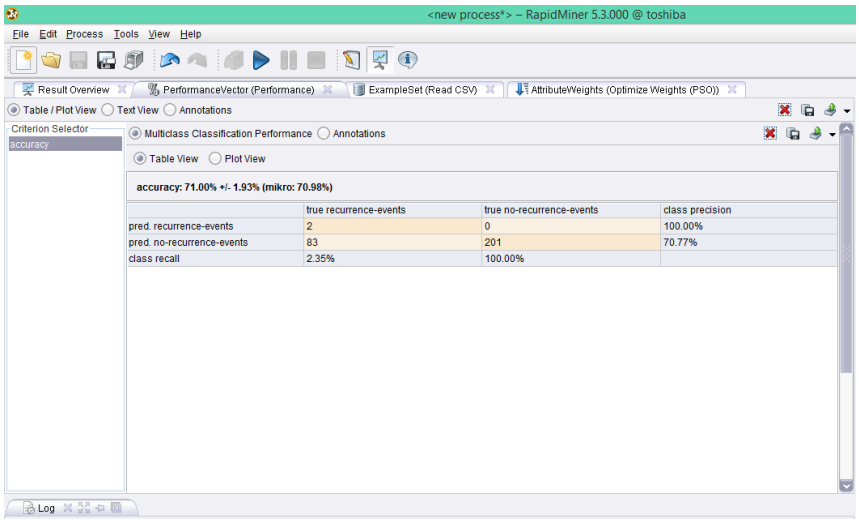
Langkah selanjutnya yaitu proses klasifikasi *decision tree* pada bagian *training view* dan pengujian model pada *testing view* dengan cara *drag decision tree* dari *operator view* menuju *training view*. Kemudian *drag apply model* dan *performance* menuju *testing view* untuk menunjukkan hasil dari proses klasifikasi

desicion tree menggunakan *particle swarm optimization*. Setelah itu hubungkan titik-titik seperti yang terlihat pada Gambar 5.76.



Gambar 5.76. Tampilan proses *validation* pada pso

Kemudian klik *run* untuk memproses klasifikasi pada *desicion tree* menggunakan *particle swarm optimization*, maka hasil dapat dilihat pada Gambar 5.77.



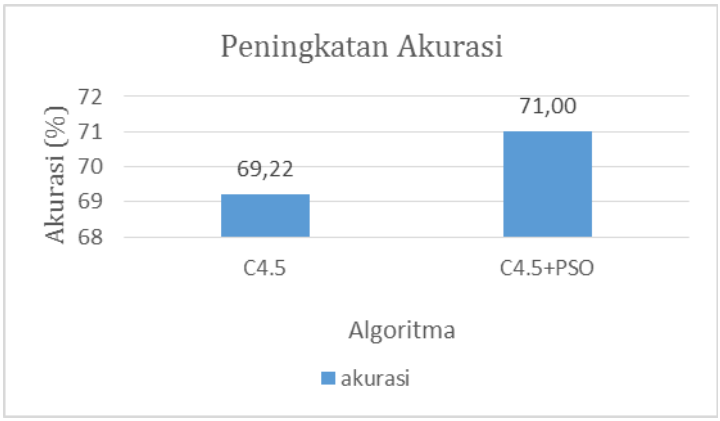
Gambar 5.77. Hasil proses klasifikasi C4.5 dan *particle swarm optimization* menggunakan *dataset* kanker payudara

Akurasi yang didapat pada algoritma C4.5 menggunakan dataset kanker payudara sebesar 71,00%. Hasil evaluasi dari *dataset* kanker payudara yang menggunakan algoritma C4.5 dan *particle swarm optimization* pada *rapid miner* dapat dilihat pada Tabel 5.23.

Tabel 5.23. Hasil evaluasi dari *dataset* kanker payudara yang menggunakan algoritma C4.5 dan *particle swarm optimization* pada *RapidMiner*

	True recure	True no-recure	Classified precission
Pred-recure	2	0	100,00%
Pred-no recure	83	201	70,77%
Class recall	2,35%	100,00%	

Penggunaan *particle swarm optimization* mempengaruhi peningkatan akurasi pada algoritma C4.5 dalam mendiagnosis kanker payudara. Peningkatan akurasi yang dialami yaitu sebesar 1,78% dari 69,22% menjadi 71,00%, Grafik peningkatan akurasi dapat dilihat pada Gambar 5.78.

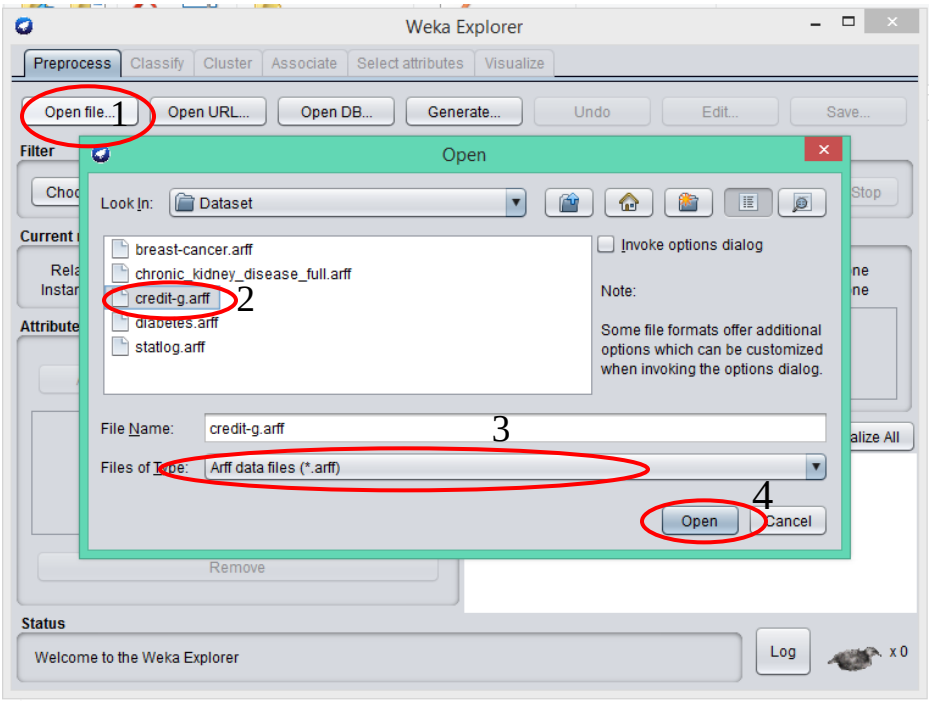


Gambar 5.78. Grafik peningkatan akurasi algoritma C4.5 menggunakan *discretization* dan *bagging*

5.5 Algoritma C4.5 Menggunakan *Split Feature Reduction Model* dan *Bagging Ensemble*

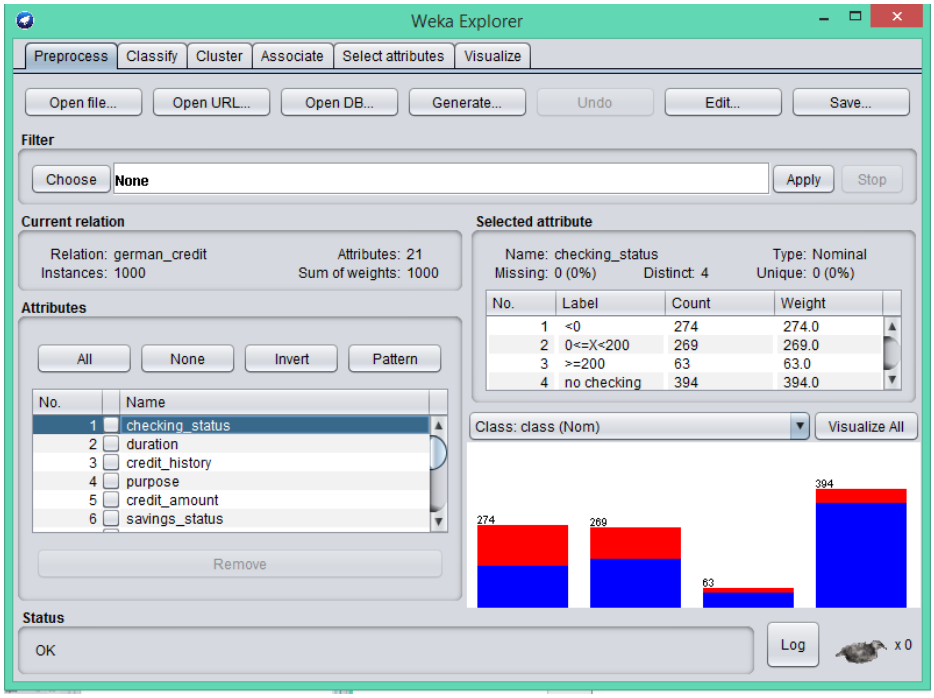
Peningkatan akurasi pada algoritma C4.5 menggunakan *split feature reduction model* dan *bagging ensemble* untuk prediksi risiko kartu kredit di proses pada *tools weka*. Langkah-langkah proses klasifikasi algoritma C4.5 pada *tools* weka, yaitu sebagai berikut.

Siapkan *dataset* .arff yang akan digunakan untuk diproses klasifikasi, pada kasus ini akan menggunakan *dataset German Credit Card* (GCD) yang diambil dari *UCI repository of machine learning*. Setelah *dataset* sudah siap klik menu *open file* pada menu *Preprocess* yang ada di tampilan *weka explorer*, maka akan tampil sepertipada Gambar 5.79.



Gambar 5.79. Memasukkan *dataset german credit card.arff*

Setelah *dataset german credit card* dimasukkan ke weka. Maka grafik data dapat ditampilkan pada Gambar 5.80.

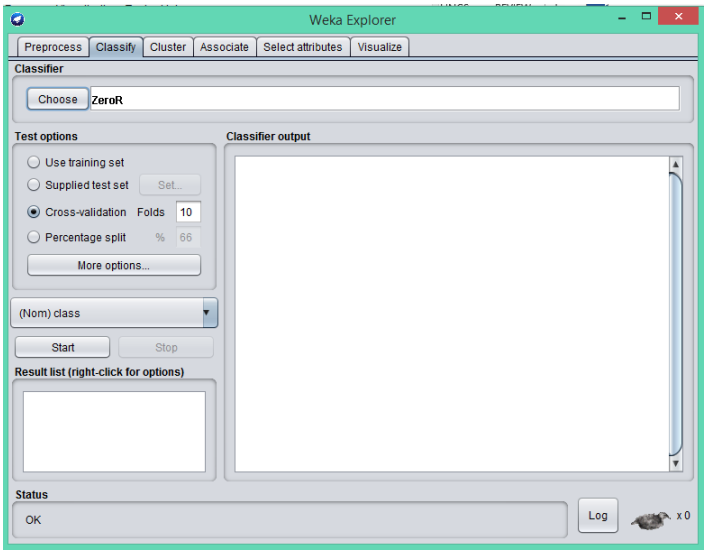


Gambar 5.80. Hasil grafik isi *dataset german credit card* setelah diinputkan

Pada Gambar 5.80. menjelaskan rincian dari *dataset german credit card*, yaitu terdiri dari 20 atribut dan 1 atribut *class* dan 1000 *instance*.

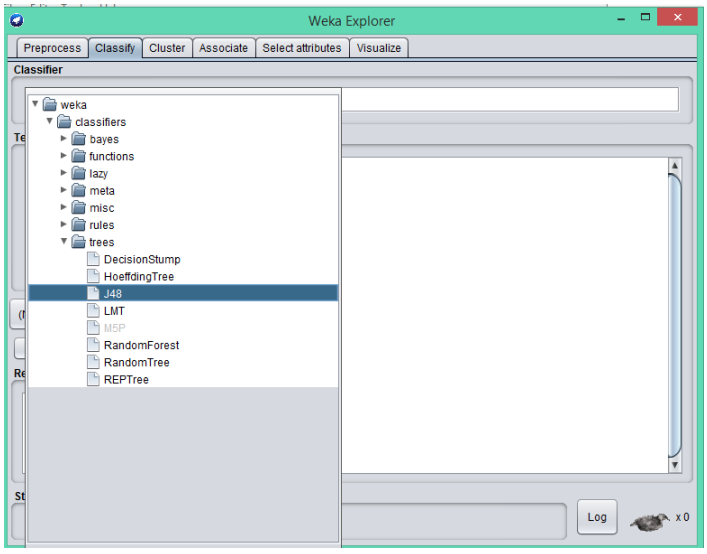
5.5.1 Pengujian dan hasil klasifikasi *decision tree* dengan algoritma C4.5 untuk prediksi resiko kartu kredit

Proses klasifikasi algoritma C4.5 di weka dapat dilakukan setelah *dataset german kredit card* di-*input*-kan melalui menu *Preprocess*, maka langkah selanjutnya yaitu menuju ke menu *classify*. Tampilan menu *classify* ditunjukkan pada Gambar 5.81.



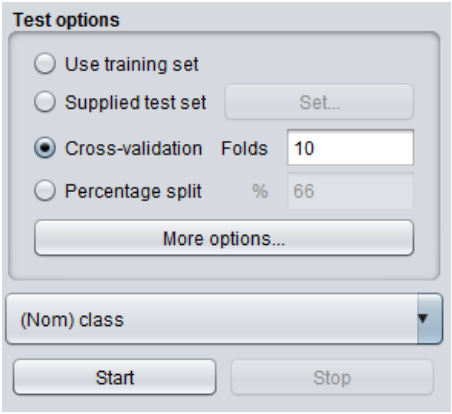
Gambar 5.81. Tampilan menu *classify*

Pada menu *classify* kemudian pilih menu *choose* untuk memilih algoritma klasifikasi *decision tree* C4.5. Algoritma C4.5 dalam weka nama lainnya yaitu J48 seperti yang terlihat pada Gambar 5.82.

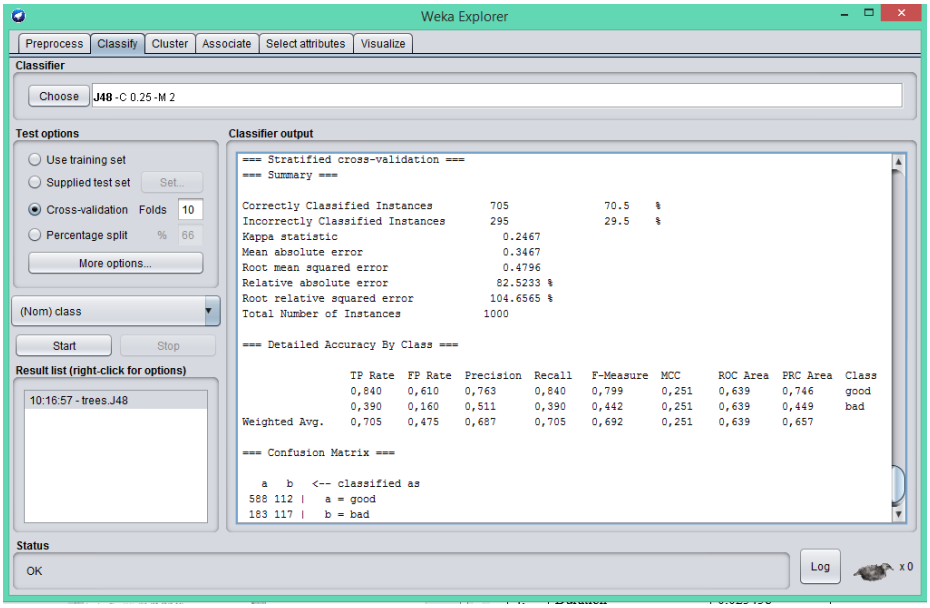


Gambar 5.82. Memilih klasifikasi algoritma C4.5

Langkah selanjutnya yaitu memilih proses pengujian *10-fold cross validation* kemudian untuk memulai proses klasifikasi pilih menu *start* seperti yang ditunjukkan pada Gambar 5.83. Dan hasil klasifikasi akan muncul di jendela *classifier output* seperti pada Gambar 5.84.



Gambar 5.83. Proses evaluasi klasifikasi



Gambar 5.84. Hasil dari klasifikasi algoritma C4.5 pada prediksi risiko kartu kredit

Hasil evaluasi dari *dataset german credit card* yang menggunakan *k-fold cross validation* dengan *default* k=10 dapat dilihat pada Tabel 5.24.

Tabel 5.24. Hasil evaluasi algoritma C4.5 menggunakan 10-*fold cross validation*

No.	Spesifikasi Pengukuran	Nilai
1	*Corectly classified instance*	705 atau 70,5%
2	*Incorectly classified instance*	295 atau 29,5%
3	*Kappa atatistic*	0,2467
4	*Mean absolute error*	0,3467
5	*Root mean squared error*	0,4796
6	*Relative absolute error*	82,5233%
7	*Root relative squared error*	104,6565%
8	*Total Number of instance*	1000

Algoritma C4.5 menghasilkan nilai *detailed accuracy by class* seperti yang disajikan pada Tabel 5.25**.** Sedangkan hasil *confusion matrix* disajikan pada Tabel 5.26.

Tabel 5.25. Hasil *detailed accuracy by class*

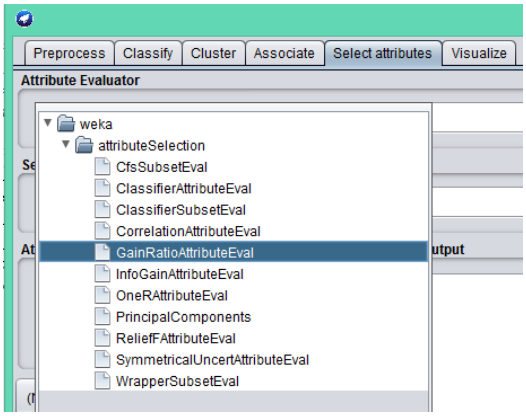
	TP rate	FP rate	Precision	Racall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,840	0,610	0,763	0,840	0,799	0,251	0,639	0,746	good
	0,390	0,160	0,511	0,390	0,442	0,251	0,639	0,449	bad
Weighted Avg.	0,705	0,475	0,687	0,705	0,692	0,251	0,639	0,657	

Tabel 5.26. Hasil *confusion matrix* algoritma C4.5

a	b	Classified as
394	106	a=tested_neg
104	164	b=tested_pos

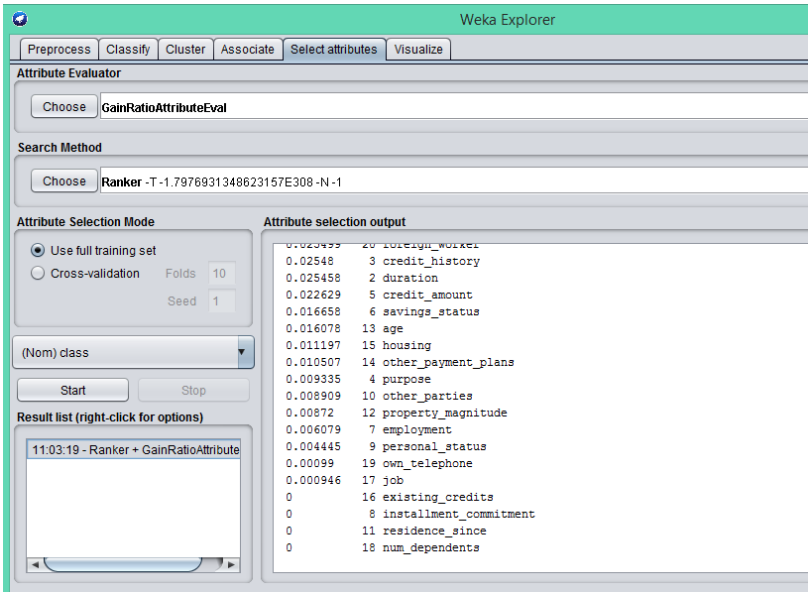
5.5.2 Pengujian dan hasil klasifikasi Algoritma *decision tree* C4.5 menggunakan *gain ratio* pada *split feature reduction model*

Apabila melihat *dataset* GCD, *dataset* tersebut termasuk *dataset* yang besar. Oleh karena itu perlu menerapkan *data reduction* (pengurangan data yang tidak perlu). Salah satu metode yang dapat digunakan adalah *feature selection*. Banyak algoritma *feature selection* yang dapat diterapkan. Salah satunya adalah dengan *gain ratio*. Penggunaan *gain ratio* pada *german credit card* dikarenakan pada dataset terdapat beberapa atribut yang memiliki lebih dari dua *values*. Semua Atribut akan dihitung nilai *gain*-nya. *Gain ratio* berkerja tidak langsung mengurangi atribut/menghapus atribut namun mengurutkan dari nilai *gain* terbesar hingga terkecil. Langkah mencari nilai *gain ratio* pada weka dapat dilakukan pada menu *select attributes* → *choose* → *attributesSelection* pilih *GainRatioAttributeEval* seperti yang terlihat pada Gambar 5.85.



Gambar 5.85. Mencari nilai *gain ratio* pada weka

Setelah memilih *selection attribute* selanjutnya klik *start* untuk memulai perhitungan nilai *gain ratio* seperti yang terlihat pada Gambar 5.86. Hasil *gain ratio* pada GCD menggunakan weka dapat dilihat pada Tabel 5.27.



Gambar 5.86. Hasil perhitungan *gain ratio* pada weka

Tabel 5.27. Nilai *gain ratio* yang sudah diurutkan

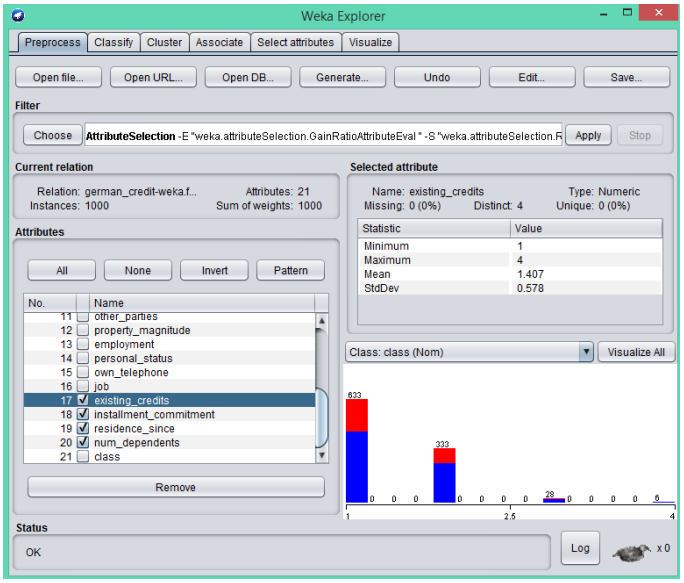
No.	Atribut	Gain Ratio
1.	Checking status	0.052573
2.	Foreign Worker	0.025499
3.	Credit History	0.02548
4.	Duration	0.025458
5.	Credit Amount	0.022629
6.	Saving Status	0.016658
7.	Age	0.016078

8.	Housing	0.011197
9.	Other Payment Plans	0.010507
10.	Purpose	0.009335
11	Other Parties	0.008909
12.	Property Magnitude	0.00872
13.	Employment	0.006079
14.	Personal Status	0.004445
15.	Own Telephone	0.00099
16.	Job	0.000946
17.	Installment Commitment	0
18.	Rendence Since	0
19.	Existing Credit	0
20.	Num Dependents	0

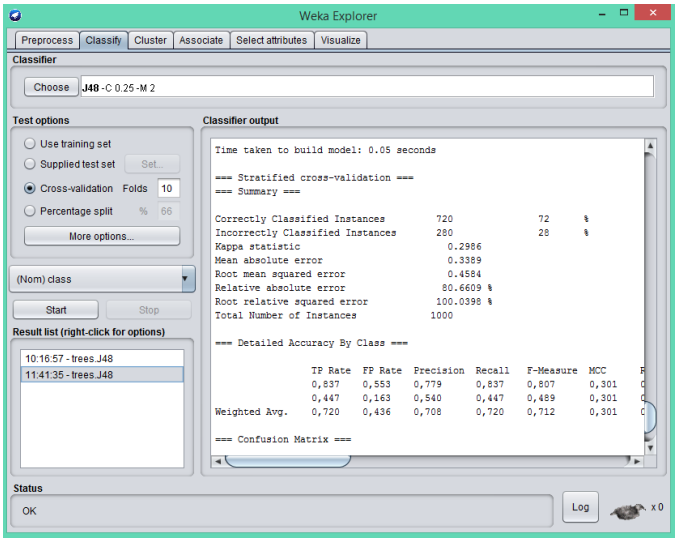
Setelah dihitung nilai *gain* dan *ranker*. Maka, pada kasus ini akan menerapkan reduksi pada atribut GCD. Pertama, atribut dengan nilai *gain* 0, dihapus karena atribut tersebut memang sangat kurang relevan dengan peningkatan akurasi. Sehingga mendapatkan 16 atribut. Pada *data reduction*, tidak semata-mata membuang satu per satu aribut namun pada kasus ini menerapkan pembagian dari semua atribut berdasarkan nilai *gain*-nya. Sama hal nya yang dilakukan pada penelitian lain, dapat dibagi berdasarkan *level*/n. Dari ke-16 atribut tersebut akan dibagi 4 dengan sama disini pembagian tersebut dapat dinamakan dengan ***Split Feature Reduction Model*.**

Proses *split feature reduction model* diterapkan di weka hampir sama dengan proses *selection* atribut sebelumnya. Namun pada *split feature selection model* ini dilakukan berdasarkan pembagian dari atribut *dataset*. Langkah yang digunakan pada weka dapat dilakukan sebagai berikut.

Atribut yang mempunyai nilai *gain* "0" dihapus untuk dilanjutkan proses klasifikasi seperti yang terlihat pada Gambar 5.87 . Maka hasilnya dapat dilihat pada Gambar 5.88.



Gambar 5.87. Proses *split feature reduction*



Gambar 5.88. Hasil *split festure reduction model* pertama

Langkah selanjutnya melakukan proses *split feature reduction* secara berulang sebanyak pembagian atribut data yang sudah digunakan. Hasil dari *split feature reduction model* pada algoritma C4.5 dapat dilihat pada Tabel 5.28.

Tabel 5.28. Hasil *split feature reduction model* pada algoritma C4.5

Feature Select	Gain Ratio pada **Split Feature Reduction Model**			
Classifier	C4.5			
Atribut	16 atribut	12 atribut	8 atribut	4 atribut
Akurasi	72%	71.7%	**73.1%**	71.5%

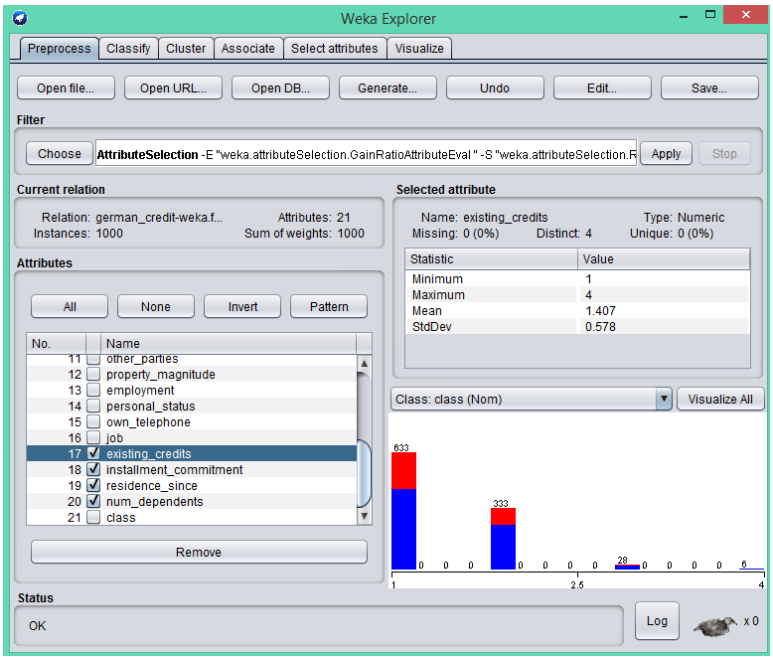
Dari hasil tersebut dapat diambil akurasi yang terbesar yaitu dengan menggunakan 8 atribut dengan akurasi sebesar 73,10%.

4.5.3 Pengujian dan hasil klasifikasi algoritma *decision tree* C4.5 menggunakan *bagging* dan *gain ratio* pada *split feature reduction model*

Penerapan *gain ratio* pada *split feature reduction model* dapat dilakukan pada *Preprocess* sebelum melakukan klasifikasi *decision tree* C4.5 dengan *bagging*. Perhitungan nilai *gain* pada *gain ratio* dilakukan pada menu *Preprocess*. Langkah-langkah pemrosesan dapat dilihat pada sub bab 4.7.2. Setelah dihitung nilai *gain* dan *ranker*. Maka, pada kasus ini akan menerapkan reduksi pada atribut GCD yang diklasifikasi menggunakan algoritma *decision tree* C4.5 dan *ensemble learning bagging*. Pertama, atribut dengan nilai *gain* 0 dihapus, sehingga mendapatkan 16 atribut yang digunakan untuk klasifikasi C4.5 dengan *bagging*. Pada *data reduction*, tidak semata-mata membuang satu per satu aribut namun pada kasus ini menerapkan pembagian dari semua atribut berdasarkan nilai *gain*-nya. Sama hal nya yang dilakukan pada penelitian lain, dapat dibagi

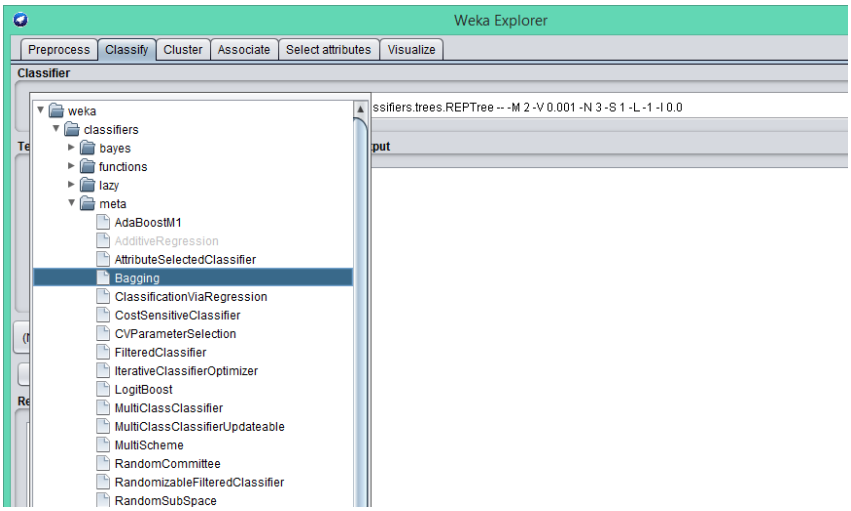
berdasarkan *level*/n. Dari ke-16 atribut tersebut akan dibagi 4 dengan sama disini pembagian tersebut dapat dinamakan dengan ***Split Feature Reduction Model.***

Atribut yang mempunyai nilai *gain* "0" dihapus untuk dilanjutkan proses kalsifikasi C4.5 dan *bagging* seperti yang terlihat pada Gambar 5.89.



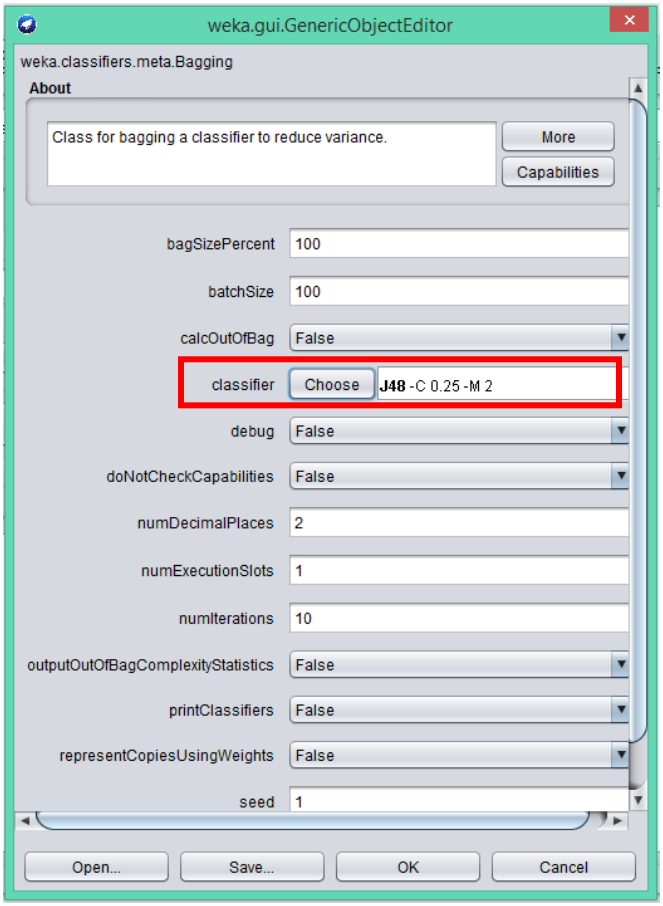
Gambar 5.89. Proses *split feature reduction*

Selanjutnya menuju proses klasifikasi dengan menerapkan *bagging* pada algoritma *desicion tree* C4.5. Langkah menggunakan *bagging* pada weka yaitu, klik menu *classify* → *choose* →meta (*bagging*) seperti yang terlihat pada Gambar 5.90.



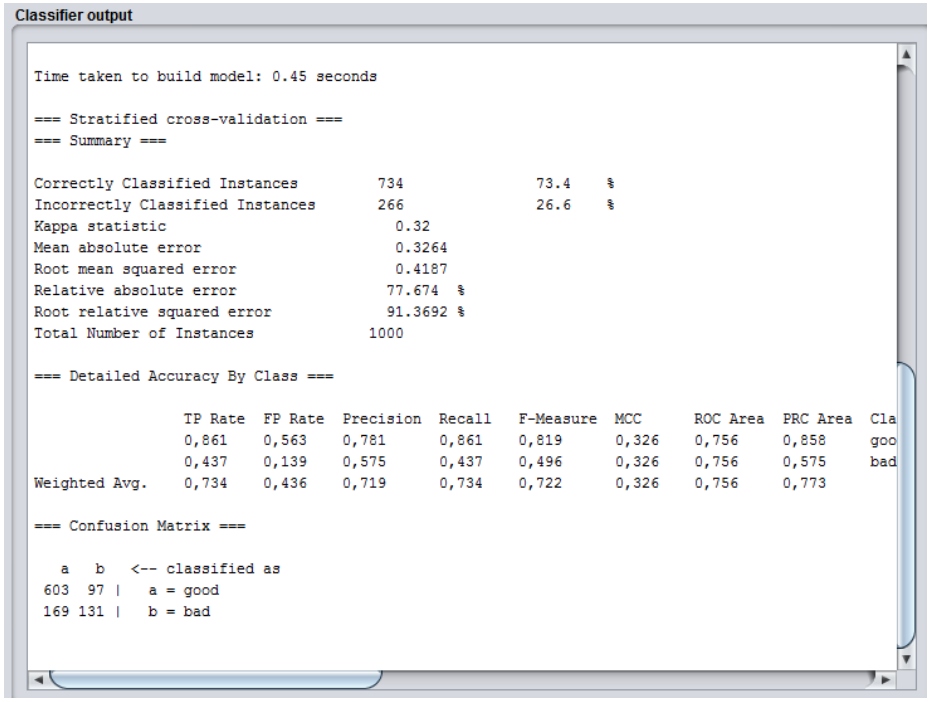
Gambar 5.90. Proses *bagging* pada weka

Setelah menerapkan *bagging* pada weka selanjutnya menerapkan algoritma C4.5 yaitu langkahnya klik pada kolom *choose* maka akan muncul tampilan seperti pada Gambar 5.91. Maka pada bagian *classifier* pilih J4.8 sebagai pengganti C4.5 pada *tools* weka.



Gambar 5.91. Tampilan weka.*Gui.GenericObjectEditor*

Setelah itu, klik oke kemudian pilih model pengujian. Dalam kasus ini menggunakan *cross validation* dengan *default* 10 setelah itu klik start untuk memulai klasifikasi algoritma *decision tree* C4.5 menggunakan *bagging* dan *gain ratio* pada *split feature reduction model* dalam prediksi resiko kartu kredit dengan 16 atribut. Hasil *split feature reduction model* pertama menggunakan algoritma C4.5 dan *bagging* dapat dilihat pada Gambar 5.92.



Gambar 5.92. Hasil *split feature reduction model* pertama menggunakan algoritma C4.5 dan *bagging*

Langkah selanjutnya melakukan proses *split feature reduction* secara berulang sebanyak pembagian atribut data yang sudah digunakan. Hasil dari *split feature reduction model* pada algoritma C4.5 dan *bagging* dapat dilihat pada Tabel 5.29.

Tabel 5.29. Hasil *split feature reduction model* pada algoritma C4.5

Feature Select	Gain Ratio pada **Split Feature Reduction Model**			
Classifier	C4.5 + Bagging			
Atribut	16 atribut	12 atribut	8 atribut	4 atribut
Akurasi	73,4%	73.7%	**75.1%**	73%

Dari hasil tersebut dapat diambil akurasi yang terbesar yaitu dengan menggunakan 8 atribut dengan akurasi sebesar 75,10%.

Hasil evaluasi dari *gain ratio* pada *split feature reduction model* pada klasifikasi algoritma C4.5 menggunakan *bagging* untuk prediksi resiko kartu kredit dengan *k-fold cross validation default* k=10 dapat dilihat pada Tabel 5.30.

Tabel 5.30. Hasil evaluasi algoritma C4.5 menggunakan *bagging* dan *gain ratio* pada *split feature reduction model*

No.	Spesifikasi Pengukuran	Nilai
1	*Corectly classified instance*	751 atau 75,10%
2	*Incorectly classified instance*	249 atau 24,90%
3	*Kappa atatistic*	0,3615
4	*Mean absolute error*	0,3296
5	*Root mean squared error*	0,4264
6	*Relative absolute error*	78,4438%
7	*Root relative squared error*	93,0414%
8	*Total Number of instance*	1000

Algoritma C4.5 dan *bagging* menggunakan *gain ratio* pada *split feature reduction model* menghasilkan nilai *detailed accuracy by class* seperti yang disajikan pada Tabel 5.31. Sedangkan hasil *confusion matrix* disajikan pada Tabel 5.32.

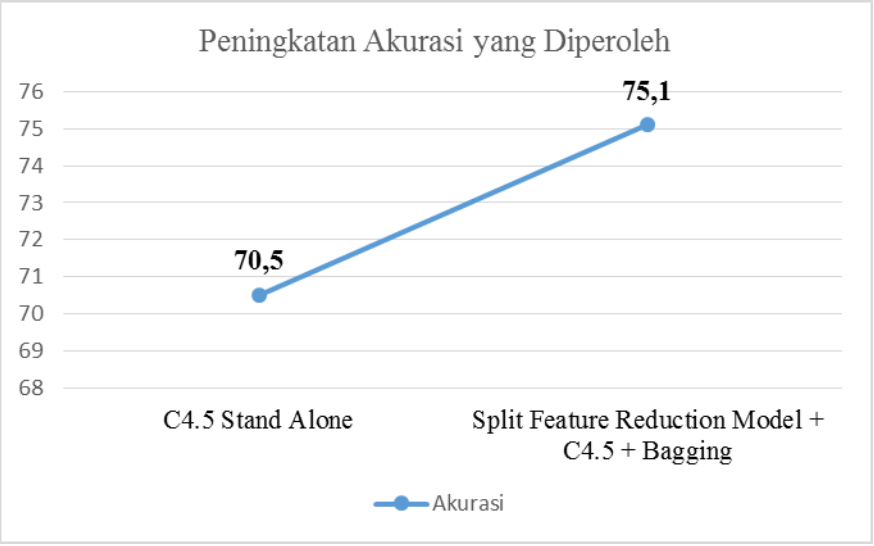
Tabel 5.31. Hasil *detailed accuracy by class*

	TP rate	FP rate	Precision	Racall	F-Measure	MCC	ROC Area	PRC Area	Class
	0,876	0,540	0,791	0,876	0,831	0,368	0,732	0,833	good
	0,460	0,124	0,613	0,460	0,526	0,368	0,732	0,546	bad
Weighted Avg.	0,751	0,415	0,738	0,751	0,740	0,368	0,732	0,747	

Tabel 5.32. Hasil *confusion matrix* algoritma C4.5 dan *bagging* menggunakan *gain ratio* pada *split feature reduction model*

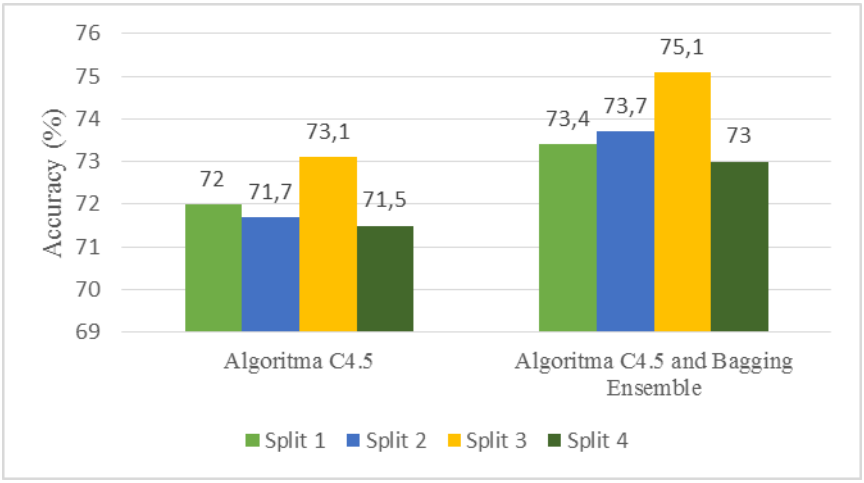
a	b	Classified as
613	87	a=good
162	138	b=bad

Apabila dibandingkan dengan hanya C4.5 saja maka akurasi untuk semua pembagian atribut meningkat. Sehingga tingkat akurasi tertinggi dengan menerapkan *Split Feature Reduction Model* dan C4.5 *Bagging* ada pada 8 atrbut dengan akurasi sebesar 75.10%. Sehingga peningkatan yang diperoleh bila dibandingkan dengan GCD dan C4.5 saja tanpa *Split Feature Reduction Model* dan *Bagging* dengan GCD dan *Split Feature Reduction Model*, C4.5, dan *Bagging* meningkat 4.6%. Peningkatan akurasi yang diperoleh dari C4.5 dengan *split feature reduction model* pada C4.5 dan *bagging* dapat dilihat pada Gambar 5.93.

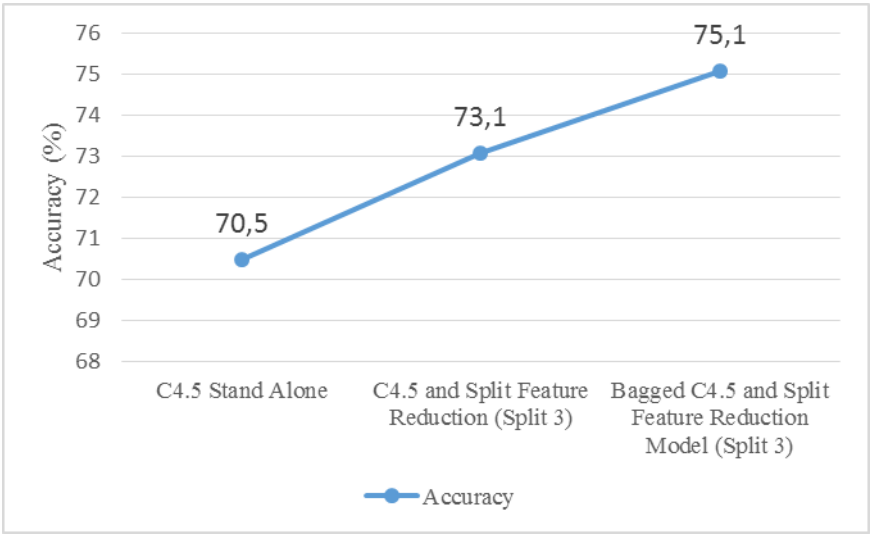


Gambar 5.93. Peningkatan akurasi yang diperoleh dari C4.5 dengan *split feature reduction model* pada C4.5 dan *bagging*

Sehingga dapat disimpulkan model pembagian atribut pada *dataset* dapat meningkatkan akurasi dengan *classifer* dan juga *ensemble learning.* Pada GCD diperoleh akurasi tertinggi dengan *split feature reduction model* hanya menggunakan 8 atribut dari 20 atribut seperti yang terlihat pada Gambar 5.94. Penerapan *bagging ensemble* juga dapat meningkat akurasi algoritma C4.5 bila dibandingkan dengan C4.5 *stand alone* seperti yang terlihat pada Gambar 5.95.



Gambar 5.94. Hasil akurasi dengan *split feature reduction model*



Gambar 5.95. Peningkatan akurasi algoritma C4.5, C4.5+*gain ratio*, C4.5+*Gain ratio*+*bagging*

DAFTAR PUSTAKA

Aziz, A., R. Saptono, & K. P. Suryajaya. 2016. Implementasi Vector Space Model dalam Pembangkitan Frequently Asked Questions Otomatis dan Solusi yang Relevan untuk Keluhan Pelanggan. *Scientific Journal of Informatics,* 2(2): 111-121.

Gorunescu, Florin. 2011. *Data mining Concept, Model and Technique*. Berlin: Springer.

Han, J. & M. Kamber. 2006. *Data mining : Concept and Techniques Second Edition.* San Fransisco: Morgan Kaufmann Publishers.

Han, J., K. Micheline, & P. Jian. 2012. *Data mining: Concepts and Techniques* (3th ed.). Waltham MA: Elsevier/Morgan Kaufmann.

Kusrini & Emha, T. L. 2009. *Algoritma Data Mining*. Yogyakarta: Andi Offset.

Listiana, E., & Muslim, M. A. (2017). Penerapan Adaboost untuk Klasifikasi Support Vector Machine Guna Meningkatkan Akurasi pada Diagnosa Chronic Kidney Disease. *Prosiding SNATIF*, 875-881.

Li, X., L. Wang, & E. Sung. 2008. AdaBoost with SVM-based component classifiers. *Engineering Applications of Artificial Intelligence*. 21 (5): 785-795.

Muslim, M. A., Herowati, A. J., Sugiharti, E., & Prasetiyo, B. (2018, March). Application of the pessimistic pruning to increase the accuracy of C4. 5 algorithm in diagnosing chronic kidney disease. In *Journal of Physics: Conference Series* (Vol. 983, No. 1, p. 012062). IOP Publishing.

Muslim, M. A., Nurzahputra, A., & Prasetiyo, B. Improving Accuracy of C4. 5 Algorithm Using Split Feature Reduction Model and Bagging Ensemble for Credit Card Risk Prediction.

Muslim, M. A., Rukmana, S. H., Sugiharti, E., Prasetiyo, B., & Alimah, S. (2018, March). Optimization of C4. 5 algorithm-based particle swarm optimization for breast cancer diagnosis. In *Journal of Physics: Conference Series* (Vol. 983, No. 1, p. 012063). IOP Publishing.

Muslim, M. A., Sugiharti, E., Prasetiyo, B., & Alimah, S. Penerapan Dizcretization dan Teknik Bagging Untuk Meningkatkan Akurasi Klasifikasi Berbasis Ensemble pada Algoritma C4. 5 dalam Mendiagnosa Diabetes. *Lontar Komputer: Jurnal Ilmiah Teknologi Informasi*, 135-143.

Nurgoho, A.S., Witarto, A.B., & Handoko, D. 2003. Support Vector Machine: Teori dan Aplikasinya dalam Bioinformatika. Proceeding of Indonesian Scientific Meeting in Central Japan. Gifu: Japan

Nurzahputra, A., & Muslim, M. A. (2017). Peningkatan Akurasi Pada Algoritma C4. 5 Menggunakan Adaboost Untuk Meminimalkan Resiko Kredit. *Prosiding SNATIF*, 243-247.

Prasetyo, E. 2012. *Data Mining: Konsep dan Aplikasi Menggunakan Matlab.* Yogyakarta: CV. Andi Offset.

Prasetyo, E. 2014. *Data Mining: Mengolah Data Menjadi Informasi Menggunakan Matlab*. Yogyakarta: CV. Andi Offset.

Saharuna, Z., Widyawan, W., & Sumaryonjo, S. (2012). Simulasi Deployment Jaringan Sensor Nirkabel Berdasarkan Algoritma Particle Swarm Optimization. *Jurnal Nasional Teknik Elektro dan Teknologi Informasi (JNTETI)*, *1*(3).

Sutton, C. D. 2005. Classification and Regression Trees, Bagging, and Boosting. *Handbook of Statistics: Data Mining and Data Visualization*, 24: 303-329.

Tan, Pang, N., Michael, S. & Vipin, K. 2006. *Introduction to Datamining.* Boston: Pearson Addison Wesley.

Wahono, R. S. & Suryana, N., 2013. Combining Particle Swarm Optimization based Feature Selection and Bagging Technique for Software Defect. *International Journal of Software Engineering and Its Applications* (*IJSEIA*), 7(5): 153-166.

Wijaya, K. P. & M. A. Muslim. 2016. Peningkatan Akurasi Pada Algoritma Support Vector Machine Dengan Penerapan Information Gain Untuk Mendiagnosa Chronic Kidney Disease. *Prosiding 3rd Seminar Nasional Ilmu Komputer*. Semarang: Universitas Negeri Semarang.

Witten, H. I., Eibe, F. & Hall, A. M., 2011. *Data Mining Machine Learning Tools and Techniques.* Burlington: Morgan Kauffman Publisher.

DATA MINING
ALGORITMA C4.5

Disertai contoh kasus dan penerapannya dengan program komputer

Dengan data mining, data yang awalnya tidak berguna dan terdapat derau dapat diolah menjadi sebuah informasi yang berguna untuk masa depan. Dalam buku ini dijelaskan beberapa teknik klasifikasi dalam data mining. Penjelasan diberikan secara meyeluruh dari konsep dasar hingga langkah-langkah bagaimana sebuah data dapat menjadi informasi. Dalam buku ini Anda akan dituntun untuk memahami cara kerja sebuah algoritma disertai dengan contoh-contoh kasus yang relevan.