UNIVERSITAS INDONESIA

IMPLEMENTASI ANTAR MUKA APLIKASI *DATA MINING ALGORITHM COLLECTION* DAN MODUL *PREPROCESSING DATA*

SKRIPSI

YOGI KURNIA
0806457930

FAKULTAS ILMU KOMPUTER
PROGRAM STUDI ILMU KOMPUTER
DEPOK
JUNI 2012

UNIVERSITAS INDONESIA

IMPLEMENTASI ANTAR MUKA APLIKASI *DATA MINING ALGORITHM COLLECTION* DAN MODUL *PREPROCESSING DATA*

SKRIPSI

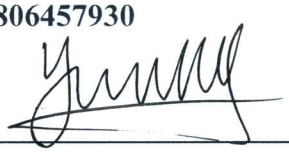
Diajukan sebagai salah satu syarat
untuk memperoleh gelar Sarjana Ilmu Komputer

YOGI KURNIA
0806457930

FAKULTAS ILMU KOMPUTER
PROGRAM STUDI ILMU KOMPUTER
DEPOK
JUNI 2012

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri,
dan semua sumber baik yang dikutip maupun dirujuk
telah saya nyatakan dengan benar.

Nama	**: Yogi Kurnia**
NPM	**: 0806457930**
Tanda Tangan	**:** 
Tanggal	**: 29 Juni 2012**

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh:

Nama	: Yogi Kurnia
NPM	: 0806457930
Program Studi	: Ilmu Komputer
Judul Skripsi	: Implementasi Antar Muka Aplikasi *Data Mining Algorithm Collection* dan Modul *Preprocessing Data*

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana S.Kom pada program Studi Ilmu Komputer, Fakultas Ilmu Komputer, Universitas Indonesia.

DEWAN PENGUJI

Pembimbing	: Yova Ruldeviyani, S.Kom., M.Kom	(.....)
Penguji	: Hisar Maruli Manurung, S.Kom., Ph.D.	(.....)
Penguji	: Samuel Louvan, S.Kom, M.Sc	(.....)

Ditetapkan di : Fakultas Ilmu Komputer
Tanggal : 29 Juni 2012

KATA PENGANTAR

Alhamdulillahirrabbilalamin, segala puji dan syukur hanya kehadirat Allah SWT, karena berkat Rahmat, Hidayah dan Ridho-Nya penulis dapat menyelesaikan skripsi ini dengan tepat waktu. Penulisan skripsi ini dilakukan untuk memenuhi syarat untuk menyelesaikan pendidikan di Program Sarjana Ilmu Komputer. Penulis sangat menyadari bahwa dari awal perkuliahan hingga penulisan skripsi ini dapat diselesaikan, penulis selalu mendapat dukungan, bantuan, dan bimbingan dari berbagai pihak. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. Yova Ruldeviyani S.Kom., M.Kom, selaku dosen pembimbing skripsi penulis yang telah menyediakan waktu, tenaga, dan pikiran untuk mengarahkan penulis selama ini.
2. Asrul dan Marni, orang tua penulis yang luar biasa, yang selalu mendoakan dan memberikan dukungannya dalam bentuk apapun kepada penulis dalam setiap proses hidup yang penulis lalui.
3. Kedua saudari penulis, Atika Putri dan Citra Triana Putri yang selalu memotivasi penulis untuk segera menyelesaikan skripsi ini.
4. Putri Dina Rusdi, sebagai orang terdekat penulis yang selalu memberikan penulis semangat, dukungan, dan bantuan dalam menyelesaikan skripsi ini.
5. Teman seperjuangan penulis sepembimbing dalam mengerjakan skripsi, Tio Pramayudi, Aldrian Yudisthira, Ruth A. Magdalena, Juliana Anita, dan Anas.
6. Sahabat Fasilkom 2008, Angga Aditya Putra dan Rizky Ramadhan yang selalu menemani dan menghibur penulis ketika suntuk.
7. Sahabat geng payung, Toza, Kennet, Reza, Fikri (Babeh), Deny, Herdi, Iwan, Adit, Maya, Adriani (Bonte) yang telah menjadi teman yang baik sejak penulis masuk ke Fasilkom.
8. Seluruh sahabat Kasma 2008, Hadya Utama, Widioseno S. Adimukti, Gusni Rahma, Kiki Yunianti, Della Aptika, Syahzami Putra, SH, Akbar S. S, Arif, Niko, Gabby, Rini, Disa yang selalu membuat hidup penulis lebih ceria.

9. Sahabat Ultra 2008 yang tidak bisa disebutkan satu persatu namanya disini.
10. Semua orang yang telah membantu penulis selama ini, terimakasih atas semuanya

Penulis menyadari bahwa dalam penulisan skripsi ini masih jauh dari kesempurnaan. Oleh karena itu, penulis mengharapkan kritik dan saran yang bersifat membangun. Penulis berharap skripsi ini dapat bermanfaat bagi pembaca pada umumnya dan penulis pada khususnya.

Depok, 29 Juni 2012

Penulis

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertandatangan di bawah ini:

Nama	: Yogi Kurnia
NPM	: 0806457930
Program Studi	: Ilmu Komputer
Fakultas	: Ilmu Komputer
Jenis Karya	: Skripsi

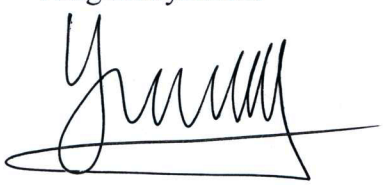
demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universtas Indonesia **Hak Bebas Royalti Noneksklusif (Non-exclusive Royalty-Free Right)** atas karya ilmiah saya yang berjudul:

Implementasi Antar Muka Aplikasi *Data Mining Algorithm Collection* dan Modul *Preprocessing Data*

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (database), merawat, dan memublikasikan tugas akhir saya tanpa meminta izin dari saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok
Pada tanggal : 29 Juni 2012
Yang menyatakan



(Yogi Kurnia)

ABSTRAK

Nama : Yogi Kurnia
Program Studi : Ilmu Komputer
Judul : Implementasi Antar Muka Aplikasi *Data Mining Algorithm Collection* dan Modul *Preprocessing Data*

Algoritma *data mining* membutuhkan sumber data yang berkualitas untuk mendapatkan hasil yang optimal. Kualitas sumber data dapat ditingkatkan kualitasnya dengan menggunakan teknik preprosessing data yang tepat. Kemampuan dalam menampilkan *output* dari proses *data mining* yang mudah dimengerti sangat penting untuk mendapatkan pengetahuan. Penelitian ini bertujuan untuk mengembangkan aplikasi yang bisa menjawab kebutuhan dari algoritma *data mining.* Hasil dari penelitian ini adalah aplikasi yang dapat melakukan keseluruhan proses baik *preprocessing data* dalam hal pemilihan data dan pengolahan data awal, penyediaan metadata, sampai dengan analisis data menggunakan algoritma *data mining*. Sehingga, analisis jumlah data yang besar dapat dilakukan dengan efisien dan efektif, tetapi hasil prediksi yang didapatkan tetap optimal.

Kata Kunci:
Preprosessing Data, Aplikasi *Data Mining*

ABSTRACT

Name : Yogi Kurnia
Study Program : Computer Science
Title : Implementation Interface of Data Mining Algorithm Collection Application and Data Preprocessing Module

Data mining algorithms require high quality data sources to obtain optimal results. Quality of data sources can be enhanced by using appropriate data preprocessing techniques. Ability to display easily understood output of the data mining process is essential to gain knowledge. This study aims to develop applications that can address the needs of data mining algorithms. The results of this study is an application that can do the whole steps from data preprocessing until data analysis using data mining algorithms. Data processing itself includes data and preliminary data processing and provision of metadata.. So, analyzing large amount of data can be done in efficient and effective fashion without disregarding necessary need of optimal prediction result.

Keywords:
Preprocessing Data, Data Mining Application

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR *PSEUDOCODE*

DAFTAR LAMPIRAN

BAB 1
PENDAHULUAN

Pada bab ini akan dibahas bagian pendahuluan yang berisi latar belakang, perumusan masalah, tujuan penelitian, ruang lingkup penelitian, tahapan penelitian dan sistematika penulisan.

1.1 Latar Belakang

Pengetahuan teknologi dan informasi pada saat ini sedang mengalami perkembangan yang pesat. Hal tersebut terjadi karena kebutuhan manusia akan informasi semakin meningkat. Informasi yang cepat dan tepat merupakan aset utama bagi manusia untuk membuat suatu keputusan atau kebijakan. Namun, informasi yang tersedia biasanya tidak dapat langsung digunakan. Pengolahan data mentah secara manual, tidak akan menghasilkan informasi yang lengkap sesuai dengan kebutuhan manusia tersebut. Oleh karena itu, manusia membutuhkan suatu proses ekstraksi dan penggalian informasi agar menghasilkan informasi lengkap. Metode tersebut dikenal dengan istilah *data mining*.

Dalam pelaksanaanya, metode *data mining* membutuhkan algoritma – algoritma yang mampu mencari pola atau prediksi dari sekumpulan data yang besar. Pola atau prediksi yang dihasilkan tersebut dapat menjadi informasi dan mampu memberikan pengetahuan kepada manusia. Beberapa algoritma yang dibutuhkan tersebut telah dikembangkan oleh mahasiswa Fakultas Ilmu Komputer Universitas Indonesia (Fasilkom). Algoritma tersebut telah dievaluasi dan ditingkatkan performanya melalui penelitian mahasiswa. Namun, proses *data mining* tidak dapat berjalan dengan baik jika yang tersedia hanya algoritma saja. Proses ini membutuhkan wadah pencarian data dari berbagai sumber, *preprocessing* masukan agar data terbebas dari kesalahan, dan menampilkan hasil dari proses tersebut.

Pada penelitian ini, penulis mengembangkan suatu aplikasi yang digunakan untuk mewadahi algoritma yang telah dikembangkan tersebut. Aplikasi ini akan

mengakomodasi proses ekstrasi dan penggalian informasi yang terdiri dari beberapa langkah, yaitu pemilihan masukan, *preprocessing* masukan, menampilkan masukan dalam bentuk grafik, memilih dan menjalankan algoritma, dan menampilkan hasil dari proses yang dilakukan algoritma.

1.2 Perumusan Masalah

Berdasarkan latar belakang tersebut, maka penulis merumuskan beberapa masalah terkait dengan proses pengambilan keputusan dengan menggunakan metode *data mining*, yaitu sebagai berikut:

1. Bagaimana membuat aplikasi yang berfungsi sebagai *workspace* untuk mewadahi penggabungan algoritma *data mining*?
2. Bagaimana mendapatkan data dari berbagai sumber untuk digunakan sebagai masukan oleh algoritma?
3. Bagaimana memvisualisasikan dan membuat metadata dari masukan untuk menampilkan karakteristiknya?
4. Bagaimana melakukan *preprocessing input* untuk meningkatkan kualitas dari masukan?
5. Bagaimana pengguna bisa menggunakan algoritma *data mining* untuk mengekstrak informasi?

1.3 Tujuan Penelitian

Tujuan penelitian tugas akhir yang penulis lakukan adalah sebagai berikut.

1. Membuat aplikasi sebagai *workspace* untuk mewadahi penggabungan algoritma *data mining*.
2. Mendapatkan data dari berbagai sumber untuk digunakan sebagai masukan oleh algoritma.
3. Memvisualisasikan dan membuat metadata dari masukan untuk menampilkan karakteristiknya.
4. Melakukan *preprocessing input* untuk meningkatkan kualitas dari masukan.
5. Menggunakan algoritma *data mining* untuk mengekstrak informasi.

1.4 Ruang Lingkup Penelitian

Ruang lingkup pengerjaan dari tugas akhir ini adalah sebagai berikut.

- Masukan dapat bersumber dari berkas dengan format ARFF, CSV, UCI, dan XLS, serta dapat bersumber dari *database* yaitu MySQL dan PostgreSQL.
- Tipe data yang bisa digunakan sebagai masukan yaitu teks dan integer.
- Proses memvisualisasikan data dapat ditampilkan dalam bentuk *bar chart, scatterplot,* dan *linear chart.*
- Algoritma yang digunakan hanya algoritma Adaboost yang dikembangkan oleh Tio Pramayudi.
- Metode yang terdapat pada tahap *preprocessing input* hanya *discretization*.

1.5 Sistematika Penulisan

Sistematika penulisan laporan tugas akhir terdiri dari enam bab sebagai berikut.

Bab 1 Pendahuluan. Bab ini terdiri dari Latar Belakang, Perumusan Masalah, Tujuan Penelitian, Ruang Lingkup Penelitian, dan Tahapan Penelitian yang akan dilakukan oleh penulis.

Bab 2 Landasan Teori. Bab ini berisi landasan teori mengenai hal-hal yang digunakan dalam penelitian, yaitu *data mining*, *preprocessing input,* penjelasan metadata, *binning* yang terdiri atas *discretize,* penjelasan struktur data dari berkas ARFF, CSV, XLS, dan UCI, pembahasan mengenai aplikasi *data* mining yang telah ada seperti Weka dan Rapidminer untuk dijadikan sebagai panduan penulis dalam merancang aplikasi *data mining*.

Bab 3 Perancangan. Bab ini berisi keseluruhan desain rancangan dari sistem yang akan dibangun, yang terdiri dari gambaran umum proses sistem, pemrosesan awal masukan, *discretization,* membangun metadata, pemilihan algoritma, dan menjalankan algoritma.

Bab 4 Implementasi. Bab ini menjelaskan detail implementasi sistem untuk mengerjakan perancangan pada bagian sebelumnya, terdiri dari implementasi pemrosesan awal masukan, implementasi pemilihan data, implementasi membangun metadata, implementasi menampilkan grafik, implementasi penggunaan metode *discretization,* implementasi pemilihan dan konfigurasi algoritma, dan implementasi bagaimana menjalankan algoritma.

Bab 5 Pengujian. Bab ini menyampaikan hasil implementasi yang telah dilakukan oleh penulis dan evaluasi terhadap pengembangan sistem yang sudah dilakukan.

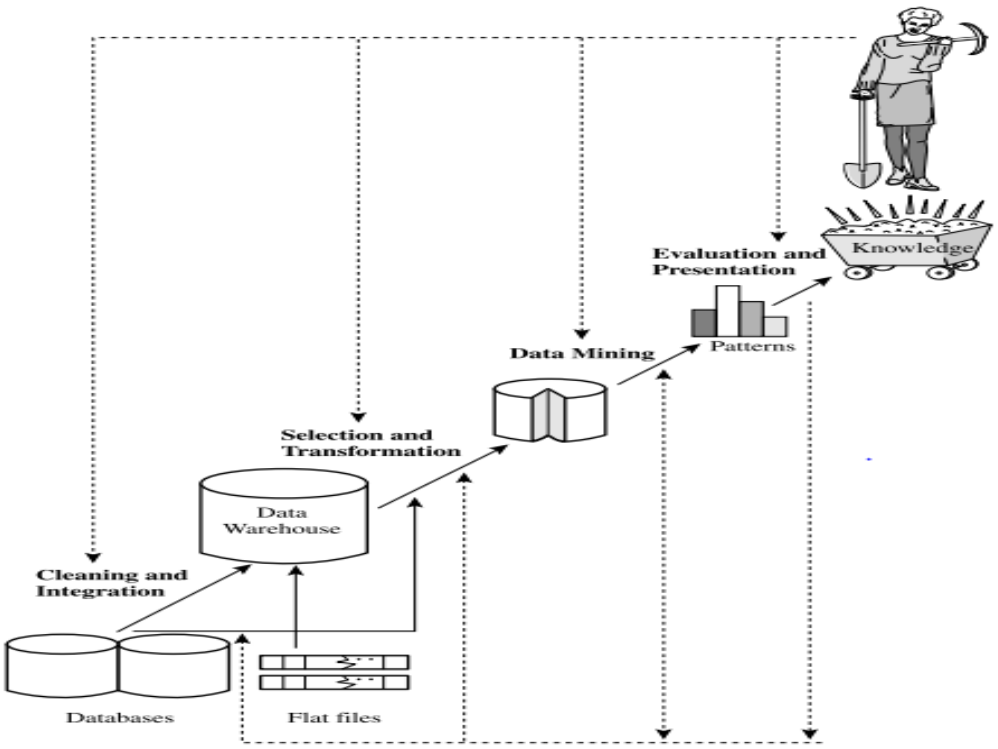
Bab 6 Penutup. Bab ini berisi tentang kesimpulan dari percobaan yang dilakukan, serta beberapa saran untuk pengembangan lebih lanjut terkait sistem ini.

BAB 2
LANDASAN TEORI

Bab ini akan menjabarkan berbagai informasi yang sudah didapatkan dari studi literatur mengenai topik yang terkait. Landasan teori dalam bab ini bertujuan untuk membentuk pemahaman yang sama antara penulis dan pembaca mengenai hal-hal yang akan disampaikan dalam laporan tugas akhir ini. Landasan teori yang akan dibahas antara lain definisi *data mining* yang akan dijelaskan pada sub-bab 2.1, penjelasan mengenai *preprocessing data* pada sub-bab 2.2, penjelasan mengenai metadata pada sub-bab 2.3, penjelasan mengenai data masukan yang akan digunakan pada sub-bab 2.4, penjelasan mengenai metode *discretization* pada sub-bab 2.5, dan penjelasan mengenai aplikasi *data mining* yang telah ada seperti Weka dan Rapid Miner yang masing – masing akan dijelaskan pada sub-bab 2.6 dan sub-bab 2.7.

2.1 Definisi *Data Mining*

Data mining didefinisikan sebagai proses menemukan suatu pola dari sekumpulan data (Witten, 2011). Pola yang ditemukan tersebut harus mempunyai pengetahuan yang nanti digunakan untuk menciptakan keuntungan, khususnya dibidang ekonomi dan bisnis. Pengertian *data mining* yang lain adalah penggalian pengetahuan dari sekumpulan data yang jumlahnya sangat besar (Han, 2006). Pengetahuan yang dihasilkan akan membantu proses pengambilan keputusan. Istilah lain dari *data mining* yang dikemukakan oleh Han adalah *Knowledge Discovery from Data* (KDD).. Istilah tersebut cocok dengan tujuan dari proses *data mining* yaitu untuk melakukan pencarian pengetahuan dari data. Dibawah ini merupakan visualisasi dari tahapan KDD pada Gambar 2.1.



Gambar 2.1 Proses *Knowledge Discovery from Data (KDD)*
Sumber : Han, Data Mining Concepts and Techniques, 2006

Proses penemuan pengetahuan yang dapat dilihat pada gambar 2.1, terdiri atas beberapa tahap yaitu :

1. *Data cleaning*, yaitu tahap pertama dalam proses KDD. Pada tahap ini dilakukan pengisian nilai-nilai yang hilang, mengidentifikasi atau menghapus *outlier*, dan menyelesaikan inkonsistensi data. Tahap ini penting dilakukan karena data yang 'kotor' dapat menyebabkan kebingungan dalam prosedur pencarian pengetahuan sehingga menghasilkan keluaran yang tidak benar atau tidak bisa diandalkan kebenerannya.
2. *Data integration*, yaitu tahap kedua dalam proses KDD. Tahap yang sangat berguna bila pengguna ingin menggabungkan data dari beberapa sumber. Proses penggabungan data tidak mudah karena dalam melakukannya bisa terjadi beberapa masalah. Sebagai contoh, beberapa atribut dapat memiliki nama yang berbeda dalam *database* yang berbeda,

sehingga menyebabkan inkonsistensi dan redundansi. Konsep *data integration* bisa dilakukan untuk menyelesaikan masalah tersebut.

3. *Data selection,* yaitu tahap ketiga dalam proses KDD. Pada tahap ini dilakukan proses pemilihan data yang cocok terhadap proses penemuan pengetahuan yang akan dilakukan.
4. *Data transformation,* yaitu tahap keempat dalam proses KDD. Pada tahap ini dilakukan proses mengubah data sehingga data yang ada sesuai karakteristiknya dengan data yang diperlukan untuk proses penggalian data.
5. *Data mining,* yaitu tahap kelima dalam proses KDD. Pada tahap ini dilakukan proses ekstraksi data sehingga menemukan suatu pola dari data tersebut.
6. *Pattern Evaluation,* yaitu tahap keenam dalam proses KDD. Pada tahap ini dilakukan proses evaluasi terhadap pola – pola yang merepresentasikan pengetahuan.
7. *Knowledge representation,* yaitu tahap ketujuh dalam proses KDD. Pada tahap ini dilakukan proses untuk menampilkan pengetahuan yang telah dihasilkan dalam berbagai bentuk visual. Bentuk visual bertujuan agar manusia yang melihat representasi pengetahuan tersebut bisa lebih memahaminya.

Langkah satu sampai langkah empat merupakan langkah persiapan data sebelum proses *data mining* dilakukan. Proses *data mining* akan menghasilkan suatu pola yang menarik dan disimpan sebagai sebuah pengetahuan baru. Pengetahuan baru tersebut akan dievaluasi lebih lanjut dan ditampilkan dalam bentuk visual. (Han, 2006)

2.2 *Preprocessing Data*

Preprocessing data adalah salah satu proses yang penting dalam kegiatan *data mining*. Proses ini bertujuan untuk meningkatkan kualitas dari data. Jika data yang digunakan untuk proses pencarian pengetahuan kualitasnya rendah, maka pengetahuan yang dihasilkan kualitasnya akan rendah pula (Han, 2006). Ada beberapa teknik yang terdapat pada tahap *preprocessing data*. Masing – masing teknik tersebut digunakan untuk menangani masalah yang berbeda. Masalah yang

terdapat pada data bisa ditemukan melalui karakteristik data dan grafik (Han, 2006).

Dibawah ini akan dijelaskan beberapa teknik yang terdapat pada tahap *preprocessing data.*

- Teknik *data cleaning* dapat diaplikasikan untuk memperbaiki inkonsistensi data dan mengisi nilai yang hilang pada suatu *instance.*
- Teknik *data integration* menggabungkan data yang berasal dari beberapa sumber yang berbeda, seperti *data warehouse*.
- Teknik *data transformations* digunakan untuk melakukan perubahan terhadap data, seperti nilai dan tipe data. Contoh teknik *data transformations* yaitu *discretization,* yang digunakan untuk merubah data dengan tipe *continous* menjadi data dengan tipe diskrit.
- Teknik *data reduction* bisa mereduksi ukuran data dengan cara *aggregating, eliminating redundant features,* atau *clustering.*

Teknik diatas tidak *mutually exlcusive,* yang berarti beberapa teknik bisa digabungkan untuk mencapai hasil yang lebih maksimal. Teknik d*ata processing*, jika diaplikasikan sebelum tahap penggalian data akan meningkatkan rata – rata kualitas dari pola atau prediksi dan waktu yang dibutuhkan dalam proses penggalian data (Han, 2006).

2.3 Metadata

Metadata adalah salah satu solusi untuk mendapatkan informasi yang tersimpan pada data digital. Metadata juga dapat diartikan sebagai data yang menjelaskan suatu data. Selain itu, metadata merupakan bentuk pengindentifikasian, penjelasan suatu atribut dan struktur dari sebuah data atau informasi. Sebuah dokumen metadata mengandung kumpulan informasi mengenai konteks, kualitas, kondisi, maupun karakteristik isi dari suatu data yang dipakai untuk keperluan proses *data mining*. Metadata menjadi sangat berguna pada *preprocessing input* karena sangat penting untuk mengetahui karakteristik dari data sebelum mengimplementasikan metode *preprocessing* yang cocok (Witten, 2011).

2.4 Data Masukan

Data masukan adalah *data training set* yang bersumber dari berkas ataupun *database*. Aplikasi ini mendukung beberapa format berkas sebagai masukan, yaitu ARFF, CSV, UCI, dan XLS. Masukan yang sesuai prosedur harus mengandung relasi, atribut, dan *instance***.** Tiap berkas mempunyai struktur yang berbeda dalam menyimpan masukan. Berikut ini akan dijelaskan cara mendapatkan data masukan berdasarkan tipe berkas yang berbeda.

2.4.1 ARFF

Berkas ARFF (*Attribute-Relation* Berkas *Format*) adalah berkas teks ASCII yang menggambarkan daftar *instance* dan atribut. Berkas ARFF dikembangkan oleh proyek *machine learning* di Departemen Ilmu Komputer dari University of Waikato untuk digunakan oleh perangkat lunak pembelajaran mesin Weka. (Paynter, 2002)

Berkas ARFF memiliki dua bagian yang berbeda. Bagian pertama adalah informasi mengenai nama relasi dan daftar atribut beserta nilainya, sedangkan bagian kedua adalah daftar data. Tiap baris informasi diawali dengan karakter '@', diikuti dengan nilai dari informasi tersebut. Dibawah ini adalah tabel yang berisi penjelasan mengenai struktur penulisan informasi dari berkas ARFF pada Tabel 2.1.

Tabel 2.1 Struktur Penulisan Informasi Berkas ARFF

Informasi	Struktur Penulisan
Relasi	`@relation <nama-relasi>`
Attribut	`@attribute <nama-atribut> {nilai1, nilai2, nilai3}` `@attribute <nama-atribut-real> REAL`

Data	`@data`
	`data1,data2,data3`
	`data1,data2,data3`

Komentar	`%comment`

Berikut contoh dari berkas ARFF :

```
%Weather data
@relation weather

@attribute outlook {sunny, overcast, rainy}
@attribute temperature real
@attribute humidity real
@attribute windy {TRUE, FALSE}
@attribute play {yes, no}

@data
sunny,85,85,FALSE,no
sunny,80,90,TRUE,no
overcast,83,86,FALSE,yes
```

2.4.2 CSV

Berkas CSV (*Comma Separated Value*) adalah berkas teks ASCII yang penulisan nilainya dipisahkan oleh karakter koma. Nama dari berkas csv digunakan untuk mengisi informasi relasi. Pada berkas ini, terdapat n jumlah baris dimana baris pertama adalah daftar nilai atribut, sedangkan baris kedua sampai baris terakhir adalah *instance*. Berikut contoh dari berkas CSV :

```
age,spectacle-prescrip,astigmatism,tear-prod-rate,contact-lenses
young,myope,no,reduced,none
young,hypermetrope,yes,reduced,none
```

2.4.3 UCI

Format UCI adalah format standar yang digunakan oleh *website* penyedia data untuk keperluan proses *data mining*, yaitu *website* UCI *dataset*. UCI terdiri atas

dua berkas, yaitu berkas .data dan berkas .names. Berkas .data ini hanya berisi daftar semua *instance* yang dipisahkan oleh karakter koma. Sedangkan informasi mengenai nama atribut dan nilainya ada di berkas lain dengan format .names. Sehingga, berkas ini dapat digunakan sebagai masukan apabila berkas .names telah tersedia.

Berikut contoh struktur data dari berkas .data.

```
1000025,5,1,1,1,2,1,3,1,1,2

1002945,5,4,4,5,7,10,3,2,1,2

1015425,3,1,1,1,2,2,3,1,1,2

1016277,6,8,8,1,3,4,3,7,1,2

1017023,4,1,1,3,2,1,3,1,1,2

1017122,8,10,10,8,7,10,9,7,1,4
```

Berikut contoh struktur data dari berkas .names

```
the target attribute: Class

Sample_code_number: continuous

Clump_Thickness: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Uniformity_of_Cell_Size: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Uniformity_of_Cell_Shape: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Marginal_Adhesion: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Single_Epithelial_Cell_Size: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Bare_Nuclei: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Bland_Chromatin: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Normal_Nucleoli: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
```

Mitoses: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Class: 2, 4

2.4.4 XLS

XLS adalah berkas yang dihasilkan oleh Microsoft Excel. Struktur dari berkas ini hampir sama dengan CSV. Terdapat sejumlah n baris dimana baris pertama berisi daftar semua atribut, sedangkan baris kedua sampai terakhir berisi *instance*. Untuk membaca semua baris dari berkas XLS diperlukan *library* tambahan yang dikembangkan oleh Lars Vogel (http://www.vogella.com/articles/JavaExcel/article.html). Berikut contoh dari berkas XLS pada Gambar 2.2.

Outlook	Temperature	Humidity	Windy	Play	Play2
Sunny	Hot	High	TRUE	No	No
Overcast	Hot	High	FALSE	Yes	Yes
Rainy	Mild	High	FALSE	Yes	Yes
Rainy	Cool	Normal	FALSE	Yes	Yes
Rainy	Cool	Normal	TRUE	No	No
Overcast	Cool	Normal	TRUE	Yes	Yes
Sunny	Mild	High	FALSE	No	No
Sunny	Cool	Normal	FALSE	Yes	Yes
Rainy	Mild	Normal	FALSE	Yes	Yes
Sunny	Mild	Normal	TRUE	Yes	Yes
Overcast	Mild	High	TRUE	Yes	Yes
Overcast	Hot	Normal	FALSE	Yes	Yes
Rainy	Mild	High	TRUE	No	No
Sunny	Hot	Normal	TRUE	No	No
Sunny	Hot	Normal	FALSE	No	No
Sunny	Mild	High	TRUE	No	No
Sunny	Mild	Normal	FALSE	No	No
Sunny	Cool	High	TRUE	Yes	Yes
Sunny	Cool	High	FALSE	Yes	Yes

Gambar 2.2 Contoh berkas *XLS*

2.5 *Discretization*

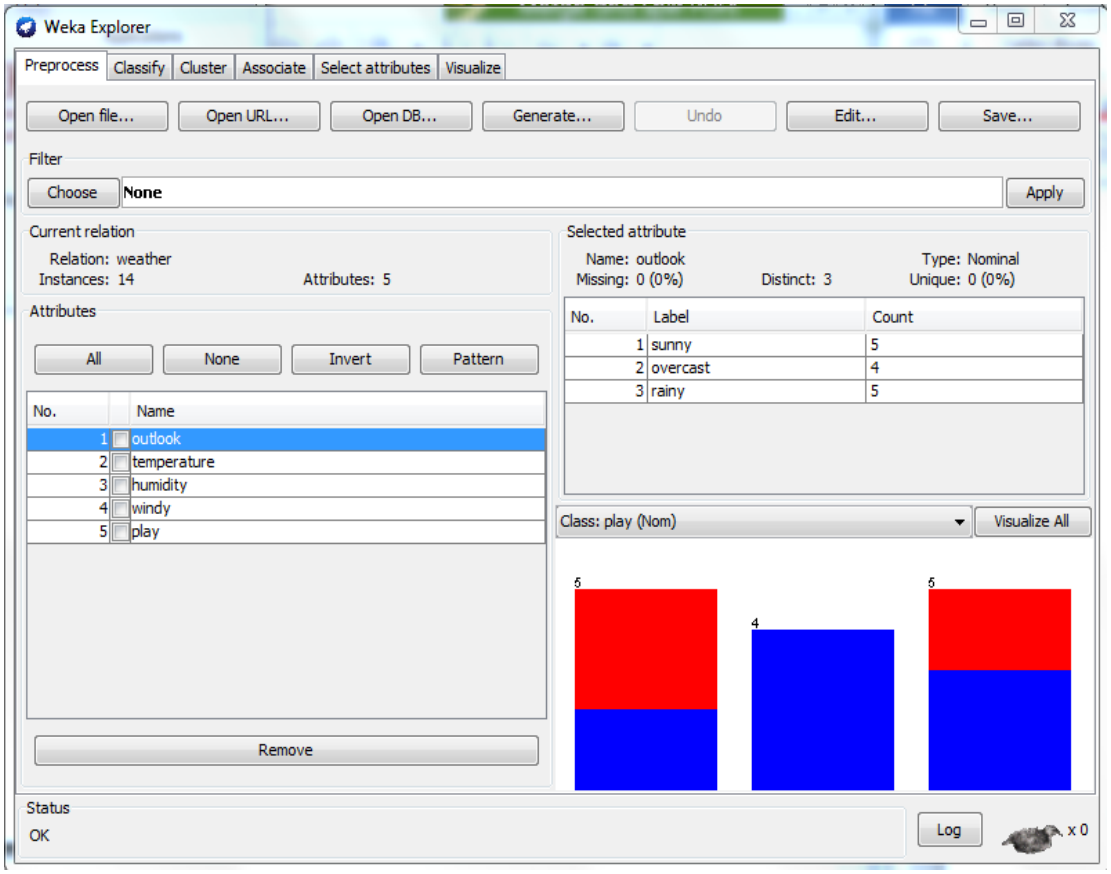
Discretize adalah salah satu metode yang digunakan pada saat *preprocessing input.* Teknik diskritisasi data dapat digunakan untuk mengurangi jumlah nilai atribut *numeric* dengan cara membagi *range* atribut ke dalam interval. Label Interval kemudian dapat digunakan untuk menggantikan nilai-nilai aktual data. Mengganti nilai dalam jumlah yang banyak dari atribut terus menerus dengan sejumlah kecil label interval dapat mengurangi dan menyederhanakan data asli. Sehingga, proses yang dilakukan menjadi singkat dan mudah dalam merepresentasikan tingkat pengetahuan dari hasil proses *data mining*. (Han, 2006)

Teknik diskritisasi dapat dikategorikan berdasarkan apakah menggunakan informasi kelas atau kearah mana proses tersebut dilakukan (yakni, *top-down vs bottom-up*). Jika proses diskritisasi menggunakan informasi kelas, maka dikatakan diskritisasi tersebut adalah *supervised discretization.* Jika tidak, dinamakan un*supervised discretization*. Jika proses dimulai dengan terlebih dahulu menemukan satu atau beberapa titik (disebut titik pembagi atau titik pemotong) untuk membagi seluruh rentang atribut, dan kemudian mengulangi secara rekursif pada interval yang dihasilkan, maka dinamakan *top-down discretization*. Hal ini bertentangan dengan *bottom-up discretization,* yang dimulai dengan mempertimbangkan semua nilai *numeric* sebagai potensi titik pembagi, menghilangkan beberapa nilai dengan menggabungkan nilai-nilai tetangganya untuk membentuk suatu interval, dan kemudian secara rekursif dilakukan proses ini hinggga ke interval yang dihasilkan. Diskritisasi dapat dilakukan secara rekursif pada atribut untuk memberikan hirarkis atau partisi multiresolusi dari nilai atribut, yang dikenal sebagai hirarki konsep. Konsep hirarki berguna untuk *mining* di berbagai tingkat abstraksi. (Han, 2006)

2.6 Weka

Weka merupakan suatu perangkat lunak yang dibangun menggunakan bahasa Java. Algoritma *machine learning* yang ada di Weka sangat banyak. Weka juga mempunyai kemampuan untuk memvisualisasikan hasil analisis data dalam bentuk *scatterplot, bar, chart,* dan *tree.*

Weka dapat menerima input dari berbagai sumber, seperti berkas ARFF, CSV, XLS, bersumber dari *database* (menggunakan JDBC), ataupun dari URL (internet). Setelah masukan didapatkan dari sumber diatas, weka menyediakan fitur untuk melakukan *preprocessing* data seperti terlihat pada Gambar 2.3 (input diambil dari *dataset* yang disediakan oleh Weka).



Gambar 2.3 Tampilan Awal Weka

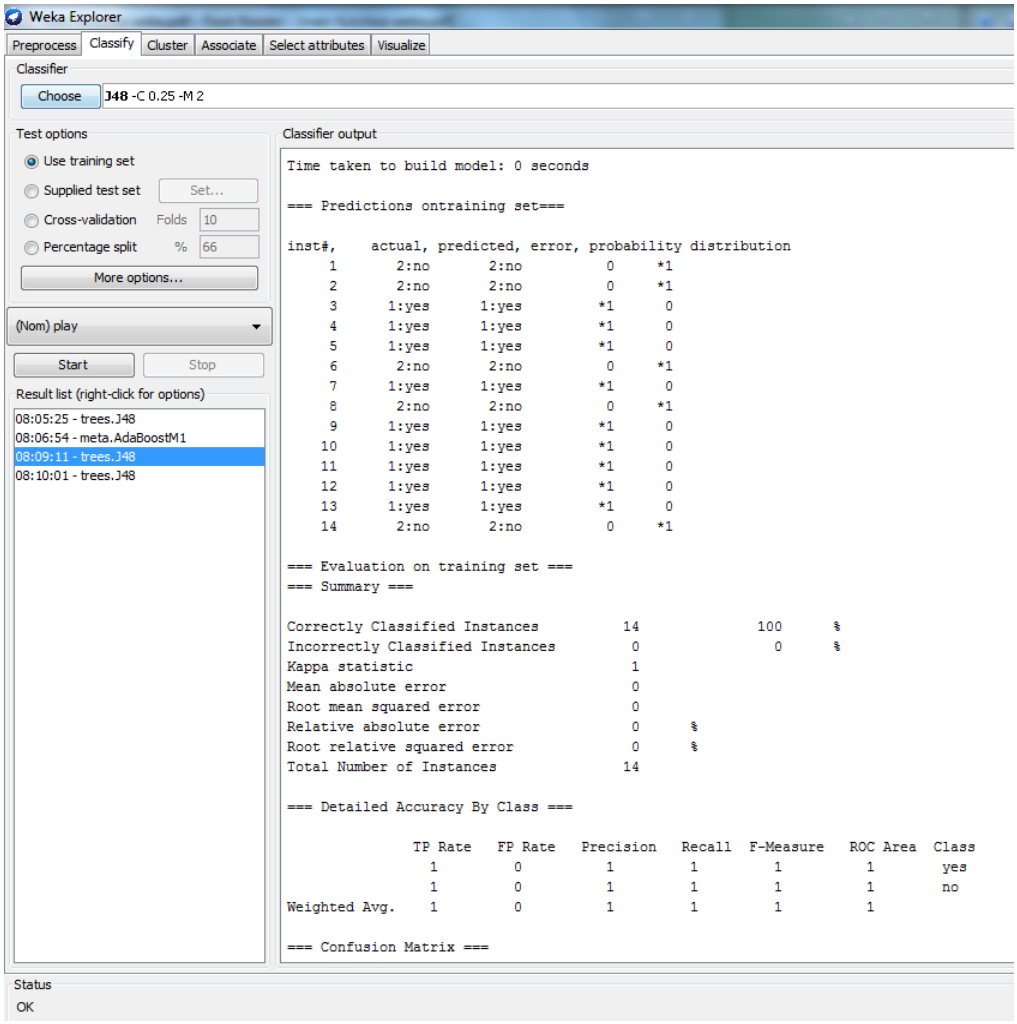
Pada weka, pengguna bisa melakukan berbagai teknik dalam *preprocessing data* untuk menyelesaikan permasalahan. Contohnya menyelesaikan permasalahan *missing value,* mengubah format atribut agar sesuai dengan yang diminta oleh algoritma. Contoh terdapat algoritma *machine* learning yang hanya menerima atribut dengan tipe *numeric,* maka semua atribut yang tidak bertipe *numeric* akan ditransformasi agar algoritma tersebut bisa digunakan. Beberapa teknik *preprocessing data* yang diterapkan oleh Weka adalah :

- *Data Cleaning* : Weka memberikan *report* persentase dari nilai yang hilang. Untuk menyelesaikan masalah tersebut, terdapat filter *ReplaceMissingValues*.

Bila terdapat *Noisy Data*, filter *RemoveMisclassified* atau *MergeTwoValues* dapat digunakan untuk mengatasinya.

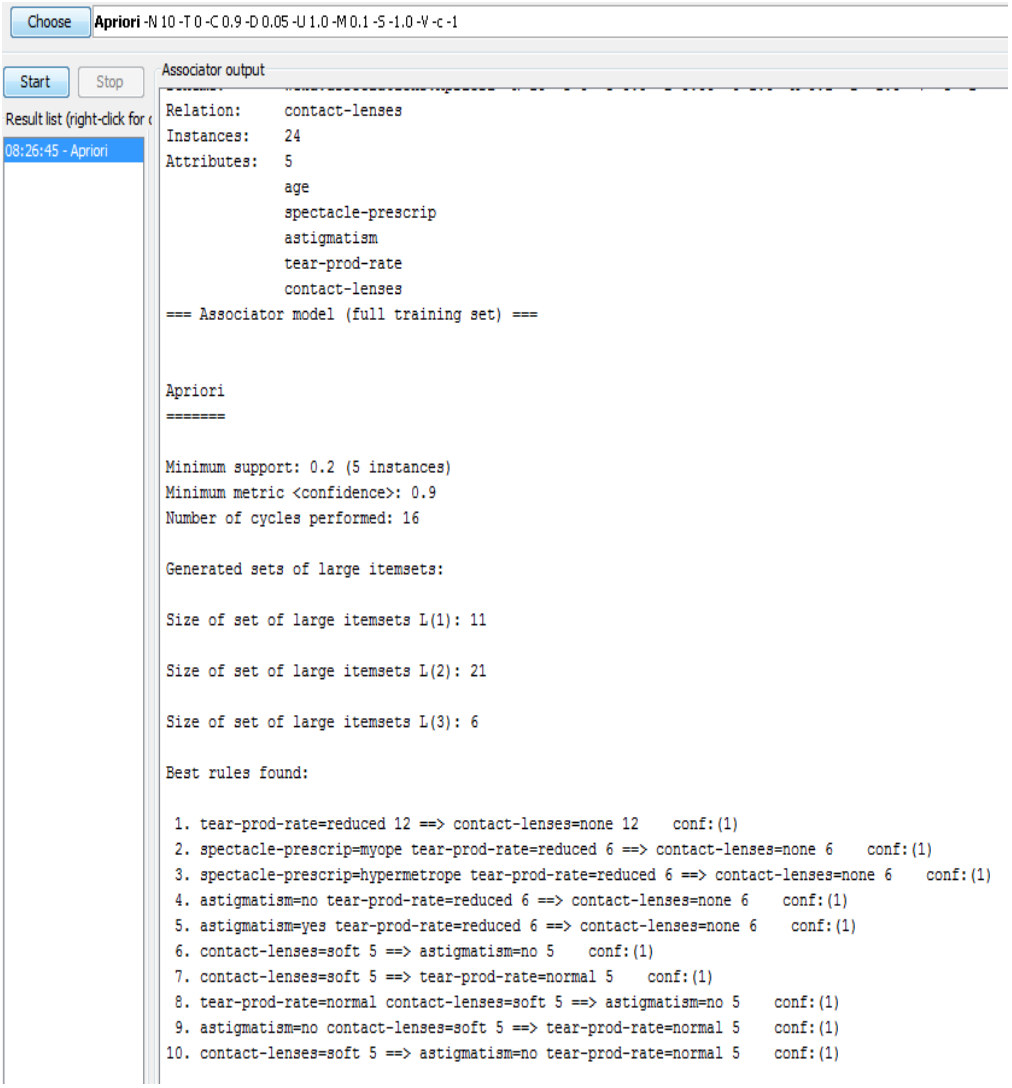
- *Data Transformation* : *data transformation filter* di Weka yaitu *Add, AddExpression, MakeIndicator, NumericTransform, Normalize, Standardize.*

Setelah melakukan *preprocessing data*, langkah selanjutnya adalah memilih *learning scheme*, yaitu *classification, cluster, dan associate*. Pemilihan c*lassifier* dilakukan melalui *classify* panel yang ada di layar. *Classifier* bertujuan untuk memprediksi suatu nilai dari atribut yang didefenisikan sebelumnya. Terdapat banyak algoritma yang bisa digunakan, seperti Naïve Bayes, J48, Multilayer Perceptron, dan lain – lain. *Classifier* bisa dievaluasi menggunakan data sebelumnya (*training set*), memasukkan data baru (*test set*), melakukan *cross validation*, ataupun *percentage split*. Hasil dari proses tersebut akan muncul di panel *classifier output*. Pada jendela *output*, pengguna bisa mengetahui persentase akurasi dari *classifier*, TP (*True Positive*) yaitu persentase dari *instance* dimana nilai yang diprediksi sama dengan nilai yang sebenarnya. FP (*False Positive*) yaitu persentase dari *instance* dimana nilai yang diprediksi berbeda dengan nilai yang sebenarnya. Berikut contoh tampilan dari jendela keluaran pada Gambar 2.4.



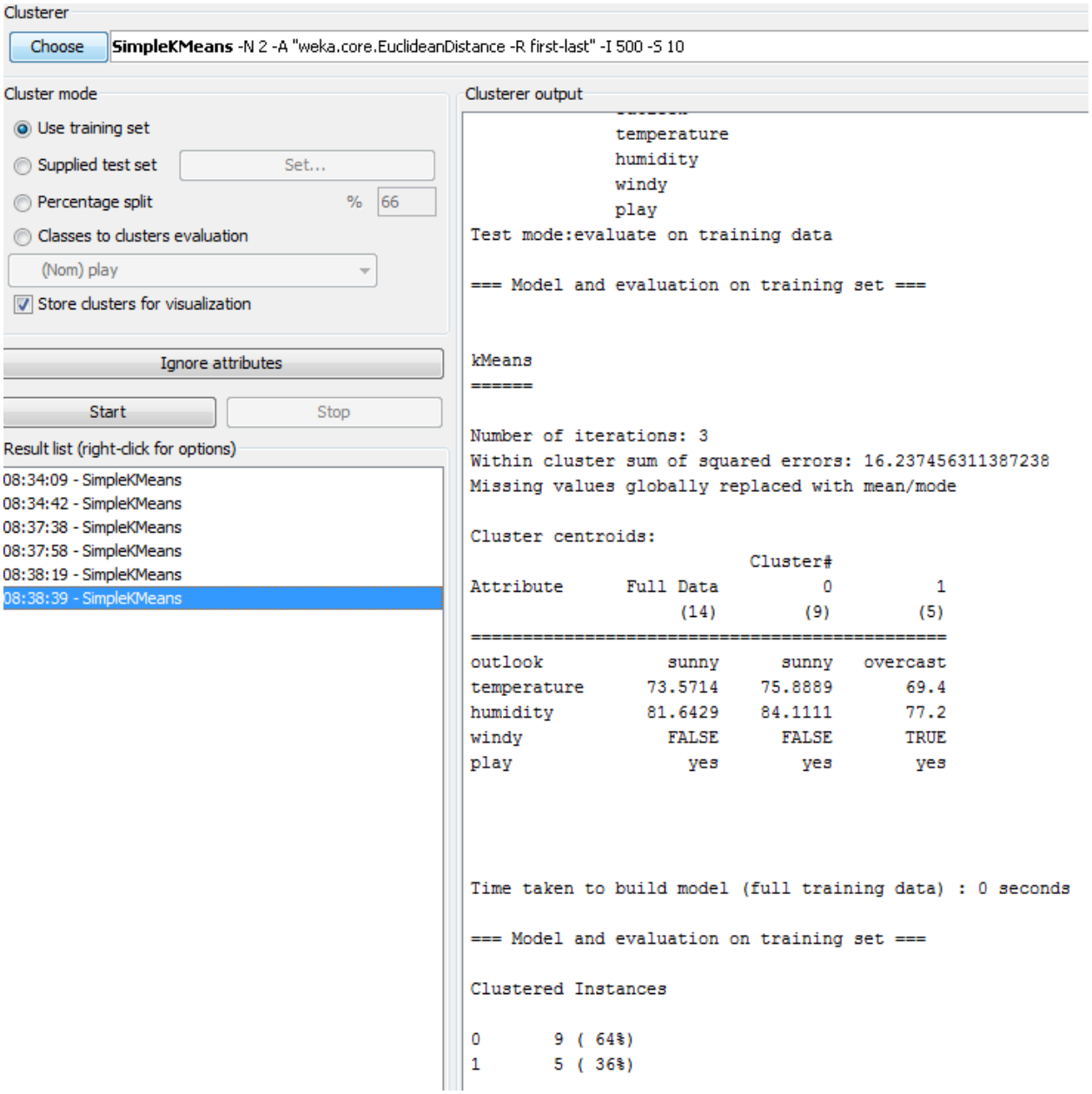
Gambar 2.4 Tampilan Weka *learning by classification*

Untuk *learning by association*, hanya sedikit algoritma yang ada dibandingkan dengan *classifier*. *Association* bertujuan untuk mencari hubungan antar atribut. Tidak ada pilihan untuk memilih *test* atau *training set*. Penulis mencoba kemampuan dari *association* dengan menggunakan algoritma apriori. Hasilnya bisa terlihat pada Gambar 2.5. Terlihat ada sepuluh *rule* yang didapat berdasarkan data yang ada.



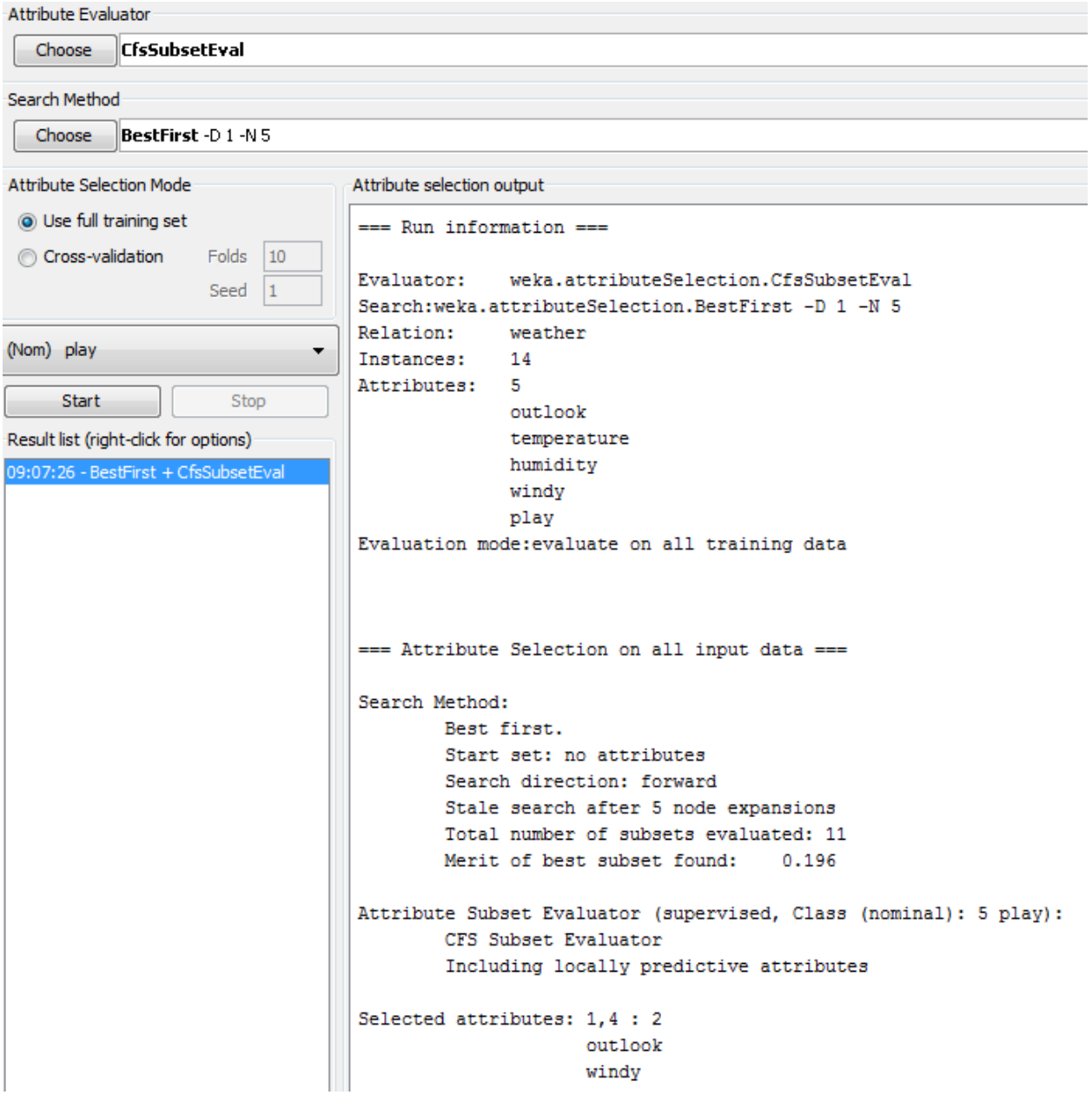
Gambar 2.5 Tampilan Weka *learning by association*

Learning selanjutnya adalah c*lustering*. C*lustering* bertujuan untuk membagi – bagi *training set* menjadi beberapa *cluster*. Terdapat sembilan algoritma yang bisa digunakan, dan juga terdapat pilihan untuk menentukan sumber data dari *test set*. Penulis menggunakan algoritma SimpleKMeans, hasilnya terlihat pada Gambar 2.6.



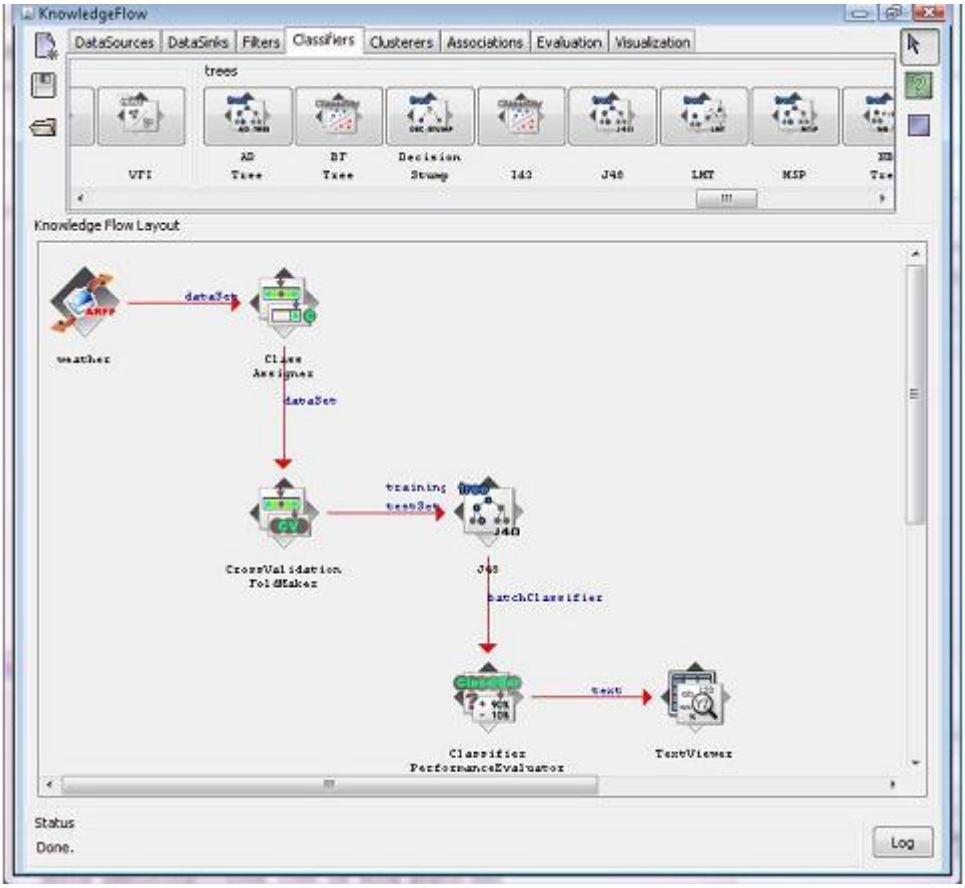
Gambar 2.6 Tampilan Weka *learning by clustering*

Fitur selanjutnya adalah pemilihan atribut. Fitur ini berguna bila *instance* mempunyai banyak atribut, sehingga membuat kinerja algoritma lebih lambat, atau terdapat beberapa atribut yang mungkin nilainya tidak relevan. Ada dua operator yang diperlukan untuk memilih atribut, yaitu *attribute evaluator* dan *search method*. Kemudian, hasilnya akan terlihat di panel *output*. Seperti yang terlihat pada Gambar 2.7, hanya dua atribut yang relevan yaitu *outlook* dan *windy*.



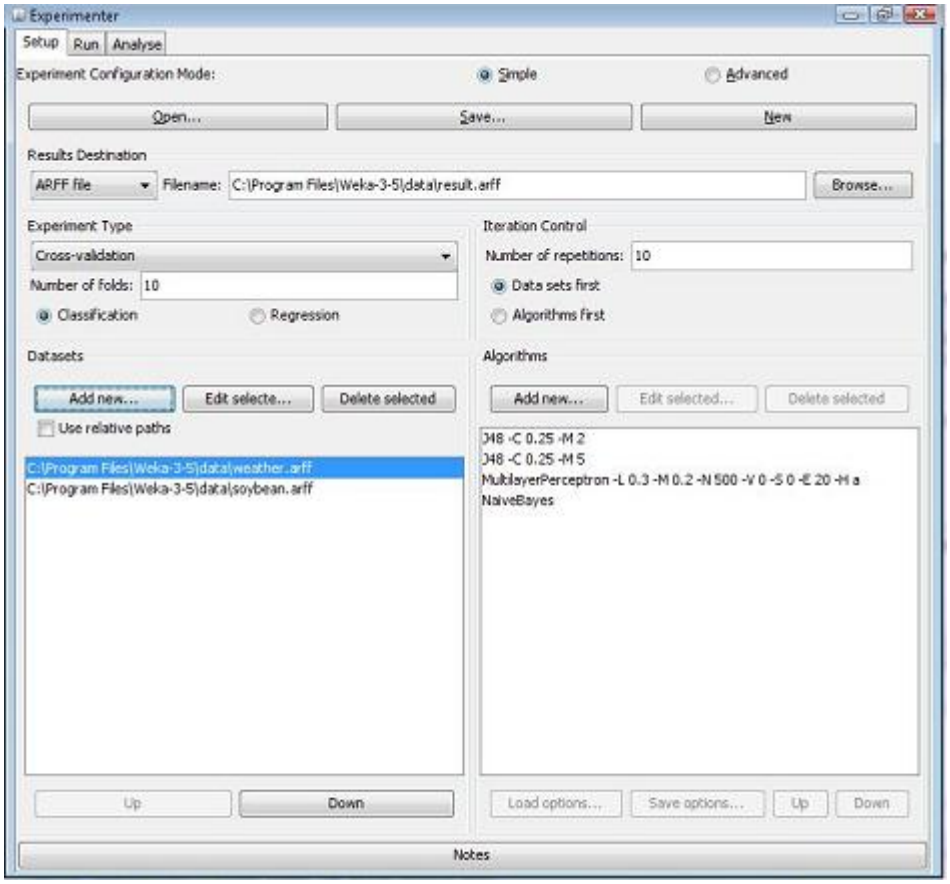
Gambar 2.7 Pemilihan Atribut

Modul yang cara pengerjaannya berbeda tapi dengan tujuan yang sama adalah modul *Knowledge Flow*. Modul ini disajikan dengan konsep *drag and drop style.* Semua komponen seperti *data loader, classifiers, clusterers, attribute selectors,* dan lain – lain, bisa diletakkan di kanvas dan dihubungkan menjadi sebuah *graph*. Pada Gambar 2.8 adalah contoh tampilan modul *knowledge flow.*



Gambar 2.8 Modul *Knowledge Flow*

Modul berikutnya adalah e*xperimenter*. Modul ini dapat digunakan untuk melakukan *multiple experiment*, yang tidak bisa dilakukan oleh dua modul sebelumnya. *Multiple experiment* digunakan untuk menentukan *learning scheme* yang lebih baik dibandingkan yang lainnya. Proses *multiple experiment* harus dicobakan dengan dataset yang berbeda. Tampilan *multiple experiment* seperti yang terlihat pada Gambar 2.9.



Gambar 2.9 Modul E*xperimenter*

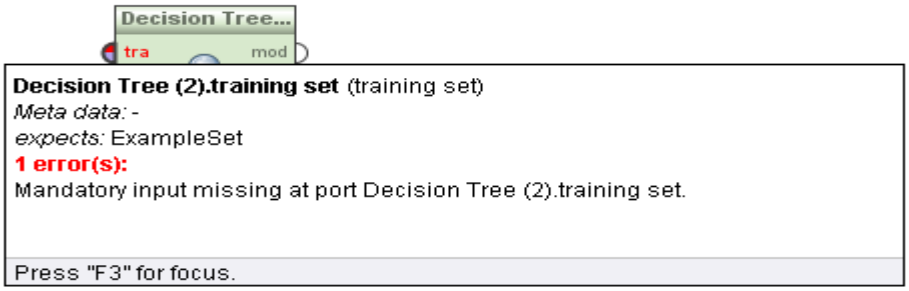
Modul ini dapat melakukan pengujian terhadap dua *dataset* yang berbeda dengan tiga *classifier* yang berbeda. Dimana salah satu dari *classifier* bisa diujikan dengan dua *dataset* sekaligus.

2.7 Rapid Miner

Rapid Miner adalah suatu *software data mining* yang proses penemuan pengetahuannya dimodelkan seperti *tree* dengan visual yang menarik. Rapid Miner ditulis dalam bahasa Java, dan bisa memproses berbagai format berkas, yaitu arff, csv, xls, maupun bersumber dari *database*.

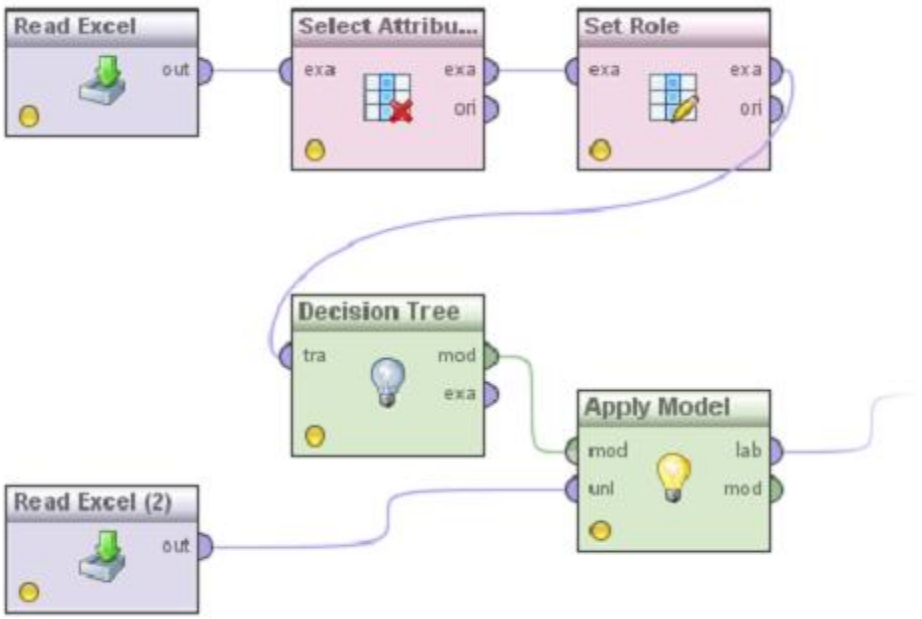
Setiap proses pada Rapid Miner dinamakan operator. Pada rapid miner terdapat *nested operator*, dimana didalamnya terdapat operator lain yang bekerja secara terpisah. Setiap operator mempunyai gerbang *input* dan *output* nilai yang diperlukan, kecuali untuk operator *import* yang hanya mempunyai gerbang *output* saja. Setiap operator dilengkapi deskripsi dari operator tersebut, seperti penjelasan

fungsinya, masukan, dan keluaran. Deskripsi dari operator akan ditampilkan pada suatu panel yang terletak dibagian bawah sebelah kanan dari jendela aplikasi. Bila pengguna memfokuskan *mouse* ke gerbang *input / output* operator tersebut, maka akan muncul keterangan mengenai status dari operator yang bersangkutan. Kesalahan dapat terjadi apabila operator tersebut belum menerima *input* dari operator lain. Contohnya seperti terlihat pada Gambar 2.10.



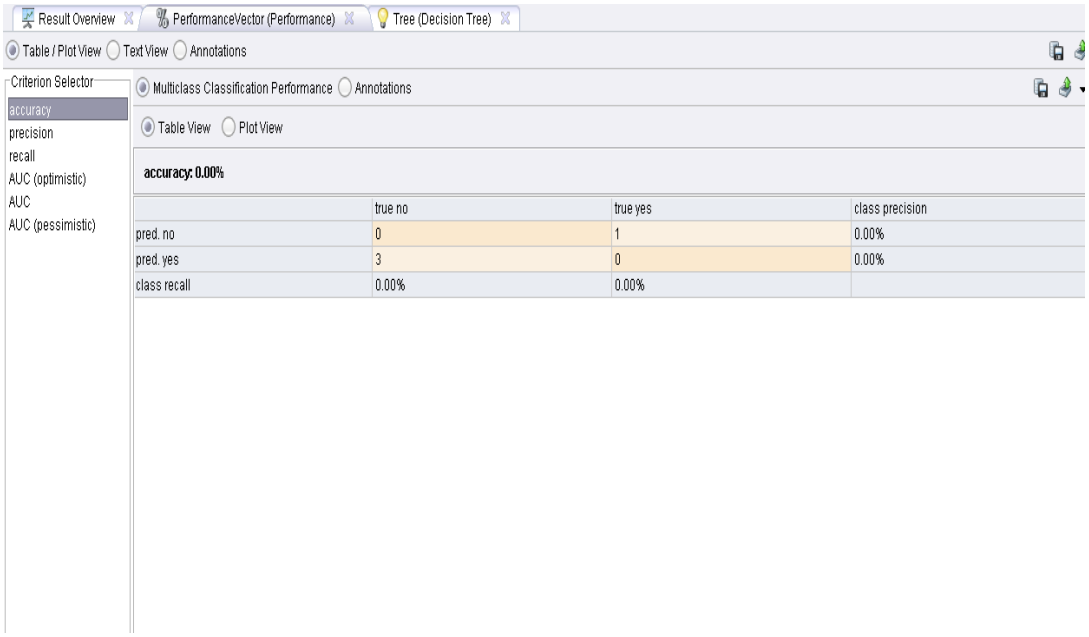
Gambar 2.10 Tampilan *error* Rapid Miner

Keseluruhan proses dari operator akan ditampilkan pada suatu *workspace.* Tiap operator akan dihubungkan oleh suatu garis yang menandakan terjadinya aliran data antar operator. Garis yang menghubungkan tidak selalu berupa garis lurus. Bentuk garis tergantung dari posisi kedua operator yang terhubung oleh garis tersebut. Contoh tampilan dari keseluruhan proses pada Gambar 2.11.



Gambar 2.11 Tampilan *workspace* dari Rapid Miner

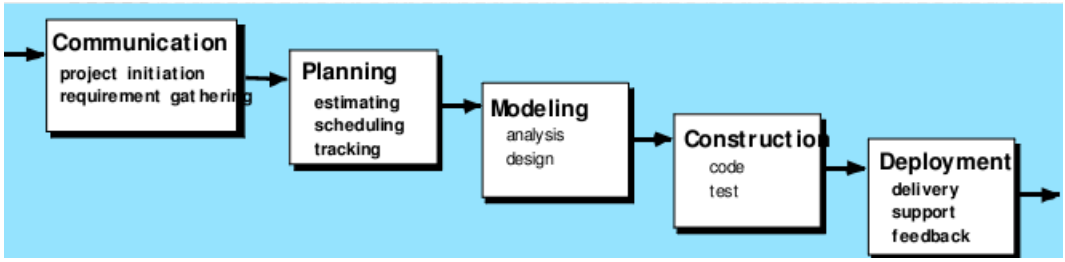
Rapid Miner menampilkan hasil dari proses *data* mining dengan analisis yang lengkap. Jika Weka menampilkan dalam bentuk teks biasa, Rapid Miner menampilkan dengan gaya yang berbeda. Antara informasi akurasi dan model visual (*tree, graph, bar, plot*) dipisahkan oleh panel yang berbeda. Berikut contoh hasilnya pada Gambar 2.12.



Gambar 2.12 Tampilan Hasil Keluaran pada Rapid Miner

2.8 *Waterfall*

Waterfall adalah suatu model pengembangan perangkat lunak yang pertama kali muncul pada tahun 1970. Model ini melakukan pendekatan secara sistematis dan terurut mulai dari level kebutuhan sistem lalu menuju ke tahap analisis, desain, *coding*, *testing / verification, dan maintenance*. Disebut dengan istilah *waterfall* karena tahap demi tahap yang dilalui harus menunggu selesainya tahap sebelumnya. Sebagai contoh, tahap desain harus menunggu selesainya tahap sebelumnya yaitu tahap requirement. Model *waterfall* ini lebih cocok untuk pengembangan *software* yang *requirement* nya telah terdefinisi dengan baik (Pressman, 2005). Secara umum tahapan pada model waterfall dapat dilihat pada Gambar 2.13.



Gambar 2.13 *Waterfall Model*

2.9 Unified Model Language (UML)

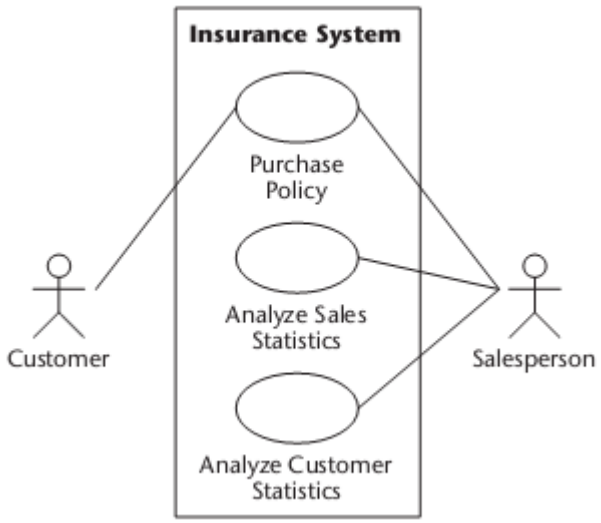
UML merupakan bahasa standar untuk membuat rancangan dari sebuah *software*. Unified Modeling Language (UML) adalah bahasa untuk menentukan, memvisualisasikan, membangun, dan mendokumentasikan artefak dari sistem perangkat lunak, serta untuk pemodelan bisnis dan sistem non-perangkat lunak. Seorang perancang perangkat lunak membuat diagram UML untuk mempermudah kinerja dari pengembang perangkat lunak. UML secara de facto merupakan notasi standar pembuatan diagram yang merepresentasikan gambar yang berkaitan dengan perangkat lunak, terutama untuk perangkat lunak yang berbasis Object Oriented (Larman, 2001).

Terdapat beberapa macam UML *diagram,* yaitu *class, deployment, use case, sequence, communication, activity, dan state diagrams.* Diagram yang bermacam – macam membuat perancang perangkat lunak lebih bisa memodelkan rancangannya dengan lebih jelas dan terarah. UML diagram yang digunakan dalam penelitian ini adalah *use case diagram, activity diagram, sequence diagram,* dan *class diagram.* Penjelasan dan notasi masing – masing diagram akan dijelaskan pada sub-bab berikut.

2.9.1 *Use Case Diagram*

Use case diagram bertujuan untuk menggambarkan fitur - fitur perangkat lunak dari sisi pengguna. Fitur tersebut akan diwakili oleh *use case.* Tiap *use case* akan berhubungan dengan sebuah aktor. Aktor adalah segala hal yang menggunakan sistem tersebut, bisa berupa *user* ataupun sistem lainnya. Sebuah *use case diagram* bisa mempunyai satu atau lebih aktor. *Use case* dan aktor dipisahkan

oleh kotak persegi panjang yang disebut *system boundary*. *System boundary* bertujuan untuk menjelaskan bahwa posisi aktor berada diluar sistem. Pada Gambar 2.14 terdapat contoh dari *use case diagram.*

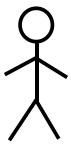


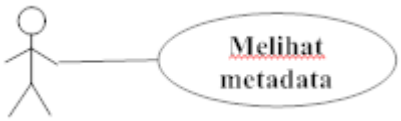
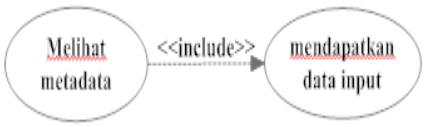
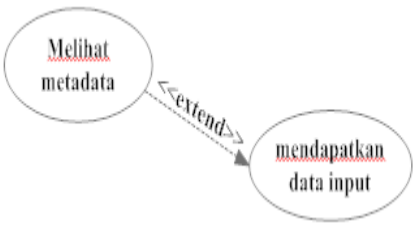
Gambar 2.14 Use-case diagram for an insurance business

Sumber : Eriksson, UML 2 Toolkit, 2004

Tiap diagram mempunyai notasi untuk menjaga keseragaman dari diagram tersebut. Pada Tabel terdapat penjelasan mengenai notasi dari *use case diagram.*

Tabel 2.2 Notasi *Use Case Diagram*

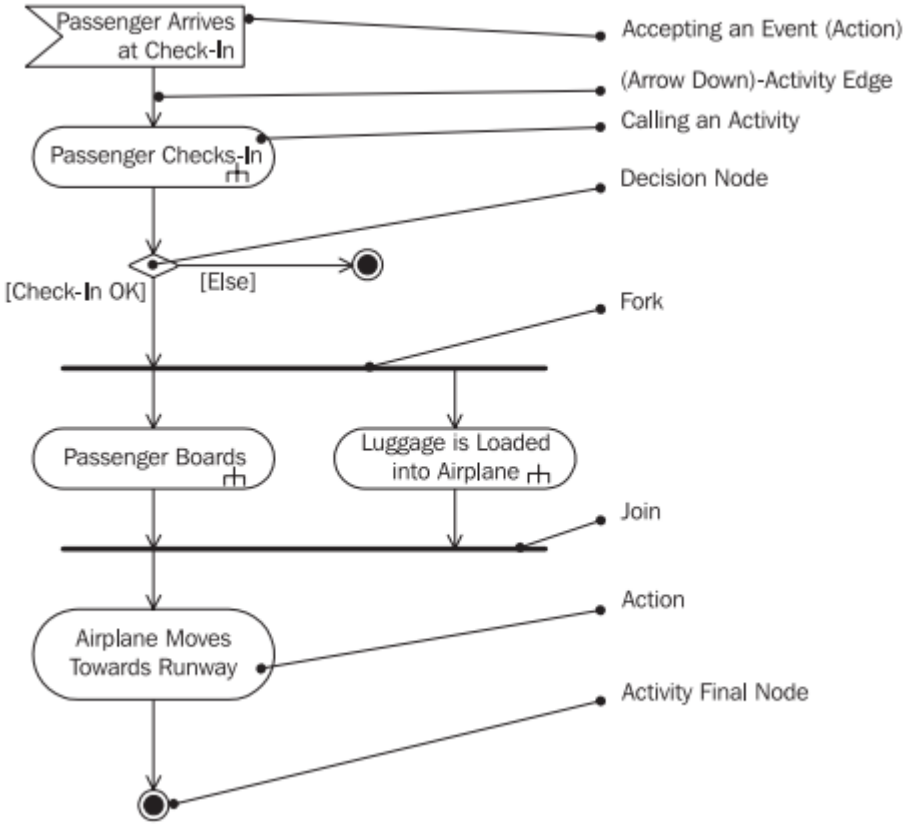
Nama	Notasi	Penjelasan
Aktor		Entitas yang berada diluar sistem dan bisa mengakses seluruh fitur yang tersedia pada sistem
Use case		Sebuah fitur atau fungsi yang bisa dilakukan oleh sistem

Association		Sebuah garis yang menunjukkan keterhubungan antara *use case* dengan satu atau lebih aktor. Hal ini menunjukkan bahwa aktor bisa menggunakan fungsi dari *use case* tersebut
Include		Sebuah garis yang menghubungkan dua *use case* dengan tulisan "include" diatasnya. Makna dari "include" yaitu *use case* yang berada diujung panah diakses fungsinya oleh *use case* yang satunya
Extend		Sebuah garis yang menghubungkan dua *use case* dengan tulisan "extend" diatasnya. Makna dari "extend" yaitu menyisipkan *use case* tambahan dari *use case* dasar. *Use case* tambahan tersebut bersifat opsional

2.9.2 *Activity Diagram*

Activity diagram, digunakan untuk menggambarkan *flow* dari suatu proses yang dilakukan oleh sistem. Dalam *activity diagram*, terlihat jelas apakah pengguna dapat melakukan proses tersebut secara bersama-sama atau independen antara satu

proses dengan lainnya (Grassle, 2005). Pada gambar 2.15 merupakan contoh dan notasi penggambaran dari *activity diagram.*

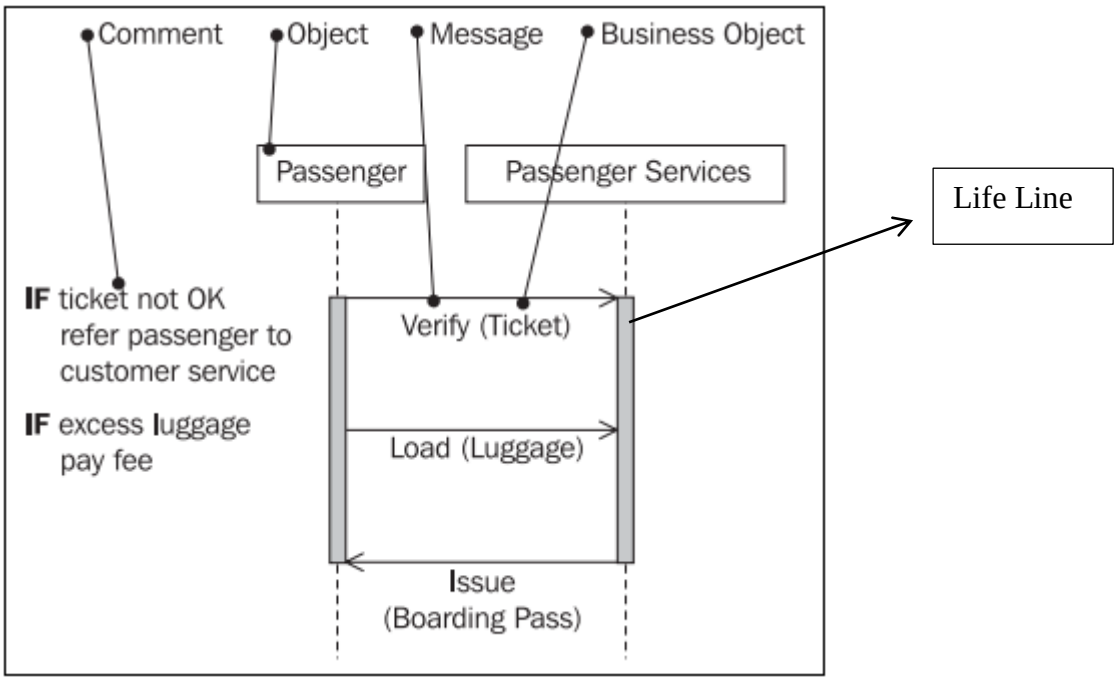


Gambar 2.15 Activity diagram "Passenger Services" with a low level of detail ("High Level")

Sumber : Grassle, UML 2.0 in Action, 2005

2.9.3 *Sequence Diagram*

UML menyediakan jenis diagram untuk representasi interaksi, yaitu *sequence diagram*. Diagram ini bertujuan untuk memvisualisasikan pertukaran informasi. *Sequence diagram* lebih menekankan kepada aspek seberapa lama informasi tersebut terdapat pada sistem. Pada Gambar 2.16 diberikan contoh dari *sequence diagram* beserta notasinya.



Gambar 2.16 The elements of the sequence diagram

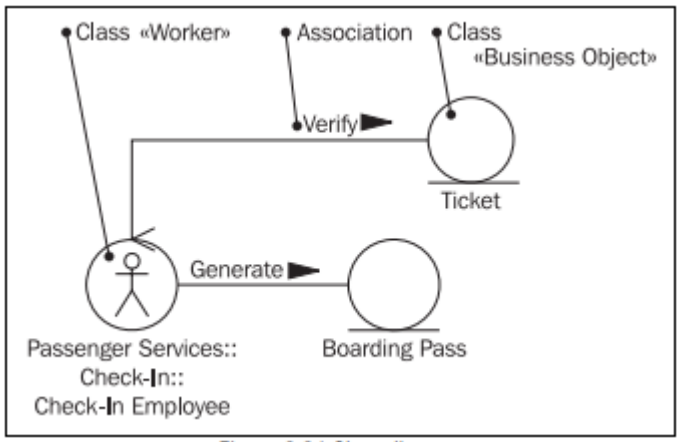
Sumber : Grassle, UML 2.0 in Action, 2005

2.9.4 *Class Diagram*

Class diagram menunjukkan struktur statis kelas dalam sistem. *Class* mewakili sesuatu yang ditangani dalam sistem. Setiap *class* dapat berhubungan satu sama lain dengan beberapa cara, yaitu :

1. *Association* : saling terhubung satu sama lain.
2. *Dependent* : suatu *class* menggunakan *class* lain.
3. *Specialized* : satu *class* merupakan spesialisasi dari *class* lainnya.
4. *Packaged* : beberapa *class* yang mempunyai fungsi hampir sama digabungkan dalam satu unit

Semua hubungan ini akan ditampilkan dalam *class diagram* bersama dengan struktur internal kelas dalam yaitu atribut dan *method*. Pada Gambar 2.17 merupakan contoh dari *class diagram.*



Gambar 2.17 Class Diagram

Sumber : Grassle, UML 2.0 in Action, 2005

BAB 3
METODOLOGI PENELITIAN

Bab ini membahas mengenai tahapan penelitian yang dilakukan penulis, yaitu perencanaan aplikasi yang akan dikembangkan, analisis, desain atau perancangan, implementasi dan evaluasi.

3.1 Tahapan Penelitian

Pengembangan aplikasi ini menggunakan suatu metode agar pengembangan yang dilakukan lebih terarah, yaitu *waterfall*. Metode *waterfall* cocok digunakan untuk penelitian ini karena *requirement* dari aplikasi telah terdefenisi dengan baik. Berikut langkah – langkah mengembangkan aplikasi ini dengan menggunakan metode *waterfall.*

3.1.1 *Requirement Gathering*

Pada tahap ini, penulis mengidentifikasi seluruh kebutuhan dari aplikasi. Tahap ini dilakukan dengan cara melakukan studi literatur terkait dengan proses yang diperlukan dalam suatu aplikasi *data mining.* Penulis juga menganalisis aplikasi *data mining* yang telah ada, seperti Weka dan Rapid Miner. Analisis ini bertujuan untuk mengetahui fitur yang dibutuhkan pada aplikasi yang akan dikembangkan.

3.1.2 *Planning*

Tahap perencanaan bertujuan untuk mengidentifikasi seluruh konsep aplikasi, penjadwalan, dan batasan dari aplikasi. Hasil dari tahapan ini merupakan sebuah rencana kerja dengan beberapa target serta batasan tertentu untuk melanjutkan ke tahapan berikutnya.

3.1.3 *Analysis*

Setelah seluruh kebutuhan dari aplikasi telah terdefenisi, selanjutnya akan dipaparkan mengenai bahasa pemrograman yang digunakan, format data masukan yang akan dipakai, teknik *preprocessing data*, algoritma *data mining* yang akan digunakan, dan bentuk grafik yang ditampilkan.

3.1.4 *Design*

Perancangan aplikasi ini terdiri dari beberapa tahap. Hubungan antara tahap tersebut terdapat ketergantungan dimana suatu tahap bisa dilaksanakan jika tahap sebelumnya telah selesai. Berikut penjelasan mengenai tahap tersebut.

3.1.4.1 Identifikasi *Use Case* dan Aktor

Kebutuhan dari aplikasi yang telah diidentifikasi sebelumnya akan dipetakan ke *use case*. *Use case* yang didapat menggambarkan proses – proses yang relatif belum terkait dengan teknologi dan detil implementasinya, serta belum ditentukan hubungan proses dengan *user interface.* Sedangkan aktor adalah pengguna dari aplikasi ini yang berpartisipasi dalam *use case.* Aktor akan berinteraksi dengan sistem seperti memberi *input* atau menerima *output.*

3.1.4.2 Analisis dan Perancangan *Input* dan *Output* Sistem

Pada tahap ini dilakukan analisis terhadap *input dan output* dari sistem. Sistem ini dimulai dari penataan data masukan mulai dari pemrosesan data dari sumber sampai siap dipakai oleh algoritma. Algoritma akan memproses data tersebut, lalu keluaran akan ditampilkan pada aplikasi.

3.1.4.3 Analisis *User Interface*

Pembuatan *user interface* dilakukan setelah *use case* didefinisikan. *User interface* digunakan untuk menampilkan dan mendapatkan informasi yang dibutuhkan secara efisien. *User interface* sangat mempengaruhi cara pengguna berinteraksi dengan aplikasi.

3.1.4.4 Analisis dan Perancangan Proses

Pada tahap ini dijelaskan lebih detil tentang *use case* yang akan digunakan pada aplikasi dalam bentuk *use case specification*. Sedangkan, alur dari tiap *use case* akan dijelaskan pada *activity diagrams.* Selanjutnya, akan dijelaskan juga perancangan penggabungan algoritma *data mining* kedalam aplikasi ini.

3.1.4.5 *Sequence Diagram*

Sequence diagram memberikan penjelasan lebih lanjut mengenai *use case* dengan memperlihatkan urutan pertukaran informasi yang terjadi antara *method*.

3.1.4.6 *Class Diagram*

Class diagram memberikan rancangan dari *class* yang akan diimplementasikan pada penelitian ini. Sumber dari *class, method,* dan atribut pada *class diagram* salah satunya adalah *sequence diagram*. Tiap aliran informasi menjadi atribut, fungsi yang dipanggil menjadi *method,* dan objek dari *sequence diagram* menjadi *class*.

3.1.5 *Implementation*

Tahap implementasi dapat dilakukan setelah semua modul telah didefenisikan. Implementasi yang dilakukan mengacu kepada *use case diagram.* Sedangkan tampilan dari aplikasi berdasarkan disain tampilan yang telah dibuat. Hasil dari tahapan implementasi ini berupa sebuah aplikasi *data mining* yang mewadahi algoritma yang telah dikembangkan sebelumnya.

3.1.6 *Evaluation*

Evaluasi terhadap tiap proses yang ada pada aplikasi dilakukan setelah proses implementasi selesai. Evaluasi dilakukan untuk mengecek apakah proses berjalan sebagaimana mestinya dan tidak ada kesalahan yang mungkin terjadi.

BAB 4
ANALISIS DAN PERANCANGAN

4.1 *Requirement Gathering*

Pada tahap ini, penulis mengidentifikasi seluruh kebutuhan dari aplikasi. Tahap ini dilakukan dengan cara melakukan studi literatur terkait dengan proses yang diperlukan dalam suatu aplikasi *data mining.* Setelah membaca literature, penulis mendapatkan gambaran mengenai tahap dari proses *data* mining. Proses tersebut seperti telah dijelaskan pada sub-bab 2.1 mengenai definisi *data mining.* Tiap proses tersebut akan menjadi *requirement* bagi aplikasi ini.

Penulis juga menganalisis aplikasi *data mining* yang telah ada, seperti Weka dan Rapid Miner. Penjelasan mengenai aplikasi Weka dan Rapid Miner dijelaskan pada sub-bab 2.5 dan 2.6. Analisis terhadap dua aplikasi diatas bertujuan untuk mengetahui fitur yang dibutuhkan pada penelitian ini. Berikut ini adalah beberapa hasil dari analisis yang penulis lakukan terhadap aplikasi Rapid Miner :

- *Interface* yang sangat menarik dan mudah dimengerti pengguna. Hal ini menjadi sangat penting karena *user interface* yang baik akan mengurangi kesalahan dalam menjalankan aplikasi dan memudahkan pengguna dalam mencapai tujuannya (Pressman, 2005).
- Mempunyai suatu *workspace* untuk menampung seluruh kegiatan pengguna sehingga memudahkan pengguna dalam melihat alur dari seluruh proses *data mining.*
- Deskripsi dari algoritma yang sangat lengkap, yaitu gambaran umum dan *input*/*output* dari algoritma tersebut.
- Keterangan *error* yang terjadi pada suatu proses dan saran untuk menyelesaikannya.
- Mempunyai teknik partisi terhadap data untuk dibagi menjadi data pelatihan dan data percobaan.
- Algoritma bisa dikonfigurasi sesuai dengan parameter yang tersedia.

Selanjutnya, dibawah ini adalah beberapa hasil dari analisis yang penulis lakukan terhadap aplikasi Weka :

- Weka mempunyai *user interface* yang kurang menarik.
- Alur proses *data mining* pada Weka tidak terlalu terlihat.
- Mempunyai teknik partisi terhadap data untuk dibagi menjadi data pelatihan dan data percobaan.
- Tidak mempunyai proses konfigurasi terhadap parameter algoritma.
- Deskripsi algoritma hanya berdasarkan cara pengambilan pengetahuan saja. Tidak ada deskripsi lebih lanjut mengenai *input/output* dari algoritma tersebut

Dari analisis yang dilakukan terhadap kedua aplikasi tersebut, penulis mengambi beberapa fitur dari kedua aplikasi diatas dan menggabungkannya kedalam aplikasi yang penulis kembangkan. Fitur yang ada di weka dan rapid miner yang penulis implementasikan terdapat pada Tabel 4.1.

Tabel 4.1 Fitur yang diambil dari Weka dan Rapid Miner

Fitur Aplikasi	Sumber Aplikasi	Alasan Diimplementasikan
Metadata dari atribut	Weka dan Rapid Miner	bisa memberikan informasi mengenai karakteristik dari data
Workspace dari proses *data mining*	Rapid Miner	alur dari seluruh proses *data mining* bisa terlihat dengan jelas
Keterangan *log*, *warning*, *error*	Rapid Miner	memberikan informasi kepada pengguna
Keterangan deskripsi algoritma	Rapid Miner	memberikan informasi kepada pengguna

Preprocessing input	Weka dan Rapid Miner	teknik yang digunakan untuk memperbaiki data masukan
Menampilkan data dalam bentuk grafik	Weka dan Rapid Miner	bisa memberikan informasi mengenai karakteristik dari data
Sumber data berasal dari *file* dan *database*	Weka dan Rapid Miner	data yang berasal dari berbagai sumber memudahkan pengguna dalam mencari sumber data

Setelah melakukan kedua proses diatas, penulis merumuskan beberapa *requirement* terkait dengan aplikasi, yaitu :

1. *Workspace* dari proses *data mining*
2. Sumber data berasal dari *file* dan *database*
3. *Preprocessing input*
4. Pembangunan Metadata
5. Pemilihan dan Konfigurasi Algoritma
6. Menjalankan Algoritma

4.2 *Planning*

Tahap perencanaan bertujuan untuk mengidentifikasi seluruh konsep aplikasi, penjadwalan, dan batasan dari aplikasi. Hasil dari tahapan ini merupakan sebuah rencana kerja dengan beberapa target serta batasan tertentu untuk melanjutkan ke tahapan berikutnya.

4.3 *Analysis*

Hasil akhir dari penelitian ini adalah aplikasi *data mining* yang mewadahi kebutuhan yang telah diidentifikasi. Kebutuhan tersebut perlu dianalisis lebih lanjut untuk mengetahui beberapa hal terkait yaitu :

1. Bahasa pemrograman yang digunakan adalah Java. Bahasa ini digunakan karena penulis telah menemukan beberapa *library* dalam bahasa Java yang akan membantu penulis dalam penelitian ini. *Library* yang digunakan adalah *library* untuk membaca *file* XLS dan ARFF, *library* untuk membuat koneksi ke *database*, serta *library* untuk menampilkan data dalam bentuk grafik.
2. Format *file* yang dapat digunakan. Format *file* yang dapat digunakan terkait dengan ketersediaan sumber data yang ada diinternet, format *file* yang biasa digunakan oleh aplikasi *data mining*, dan pengguna biasa menyimpan datanya dalam format tertentu. Format ARFF digunakan karena format ini merupakan format *file* standar yang digunakan oleh Weka. Sedangkan, format XLS digunakan karena struktur penulisan dari format ini berbentuk tabel sehingga memudahkan dalam melakukan proses membaca *file.* Format XLS adalah format standar Microsoft Excel yang banyak digunakan oleh pengguna. Format CSV adalah format yang juga dihasilkan oleh Microsoft Excel. Terakhir, adalah format UCI. UCI *dataset* dikelola dengan baik oleh *website* penyedia format tersebut. Sehingga, format UCI sangat mudah didapatkan oleh pengguna dan tersedia dalam jumlah yang banyak.
3. *Database* yang dapat digunakan. Pada penelitian ini, *database* yang dapat digunakan adalah MySql dan PostgreSql. Dalam proses *connection to database*, penulis menggunakan *library* yang disediakan oleh Netbeans.
4. Teknologi *preprocessing input.* Seperti yang telah dijelaskan pada bab 2, terdapat beberapa teknik *preprocessing input.* Teknik yang dipakai pada penelitian ini adalah *data transformation* yaitu *discretization.*
5. Bentuk grafik yang akan ditampilkan berdasarkan kepada tipe data dari masukan. Aplikasi ini hanya menerima dua tipe data, yaitu teks dan *integer.* Sehingga, bentuk grafik yang dipilih harus bisa menampilkan kedua tipe data tersebut. Bentuk grafik yang dipilih adalah *bar chart* untuk menampilkan frekuensi dari data dengan tipe teks dan *scatterplot* serta *linear chart* untuk menampilkan data dengan tipe *integer*.

4.4 *Design*

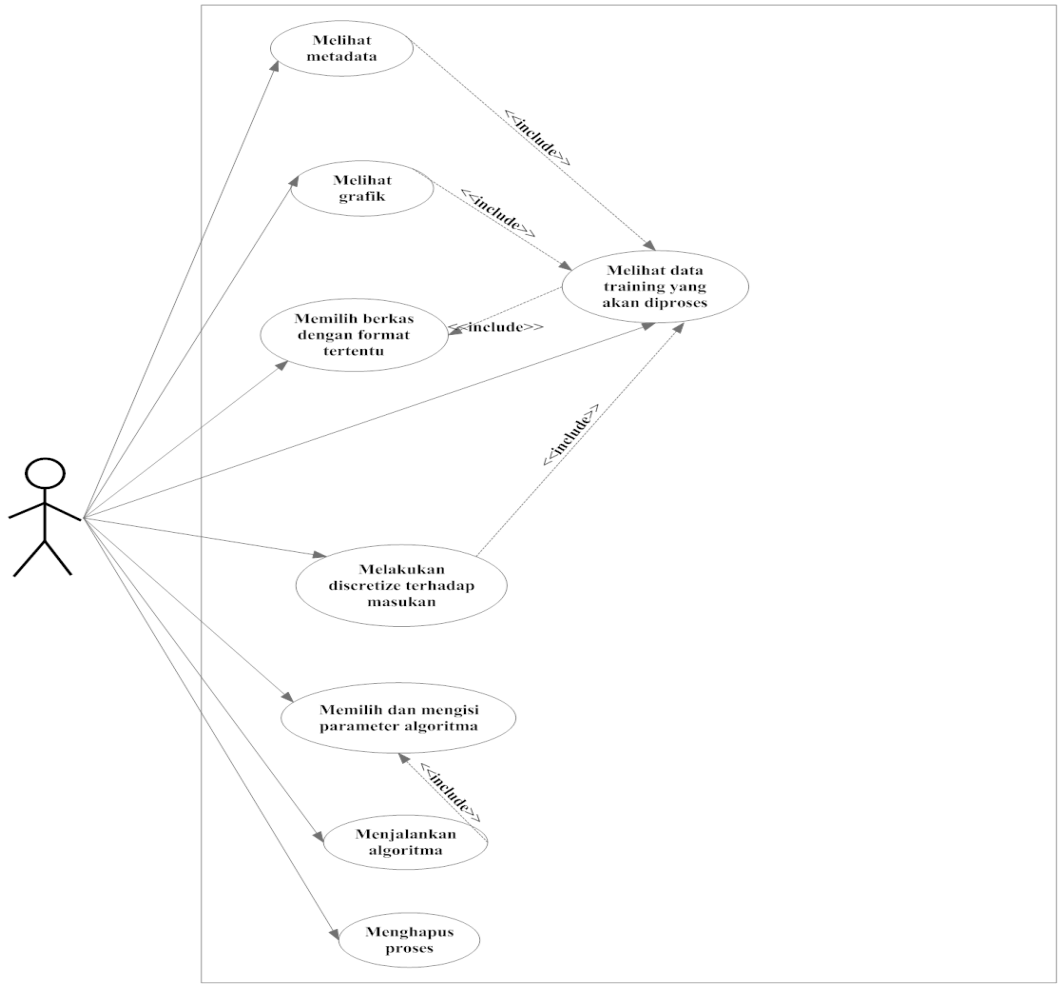
Perancangan aplikasi ini terdiri dari beberapa tahap pengerjaan. Hubungan antara tahap tersebut terdapat ketergantungan dimana suatu tahap bisa dilaksanakan jika tahap sebelumnya telah selesai. Berikut penjelasan mengenai tahap - tahap tersebut.

4.4.1 Identifikasi *Use Case* dan Aktor

Perancangan dimulai dengan mengidentifikasi actor dan *use case. Use case* digunakan untuk memberikan gambaran secaran umum dari sistem yang akan dirancang. Beberapa *use case* yang teridentifikasi seperti dijelaskan dibawah ini.

1. Melakukan pemrosesan data masukan
   Data masukan yang didapat dari berbagai sumber diproses terlebih dahulu agar bisa digunakan oleh algoritma
2. Melihat metadata dari data masukanMetadata bertujuan untuk memberikan informasi mengenai karakteristik dari data masukan. Informasi yang ditampilkan berupa nama atribut nilai, beserta jumlah dari nilai atribut, jumlah data *instance,* tipe data
3. Melakukan proses *binning*
   Pengguna bisa melakukan proses *binning* terhadap data masukan. *Binning* bertujuan untuk mentransformasi tipe data dari data masukan.
4. Memilih algoritma
   Pengguna bisa memilih algoritma dari daftar algoritma yang tersedia
5. Mengkonfigurasi algoritma
   Algoritma mempunyai beberapa parameter yang harus ditentukan nilainya. Pengguna dapat melakukan pengaturan terhadap nilai dari parameter.
6. Menjalankan algoritma
   Pengguna bisa menjalankan algoritma yang telah dikonfigurasi tersebut.

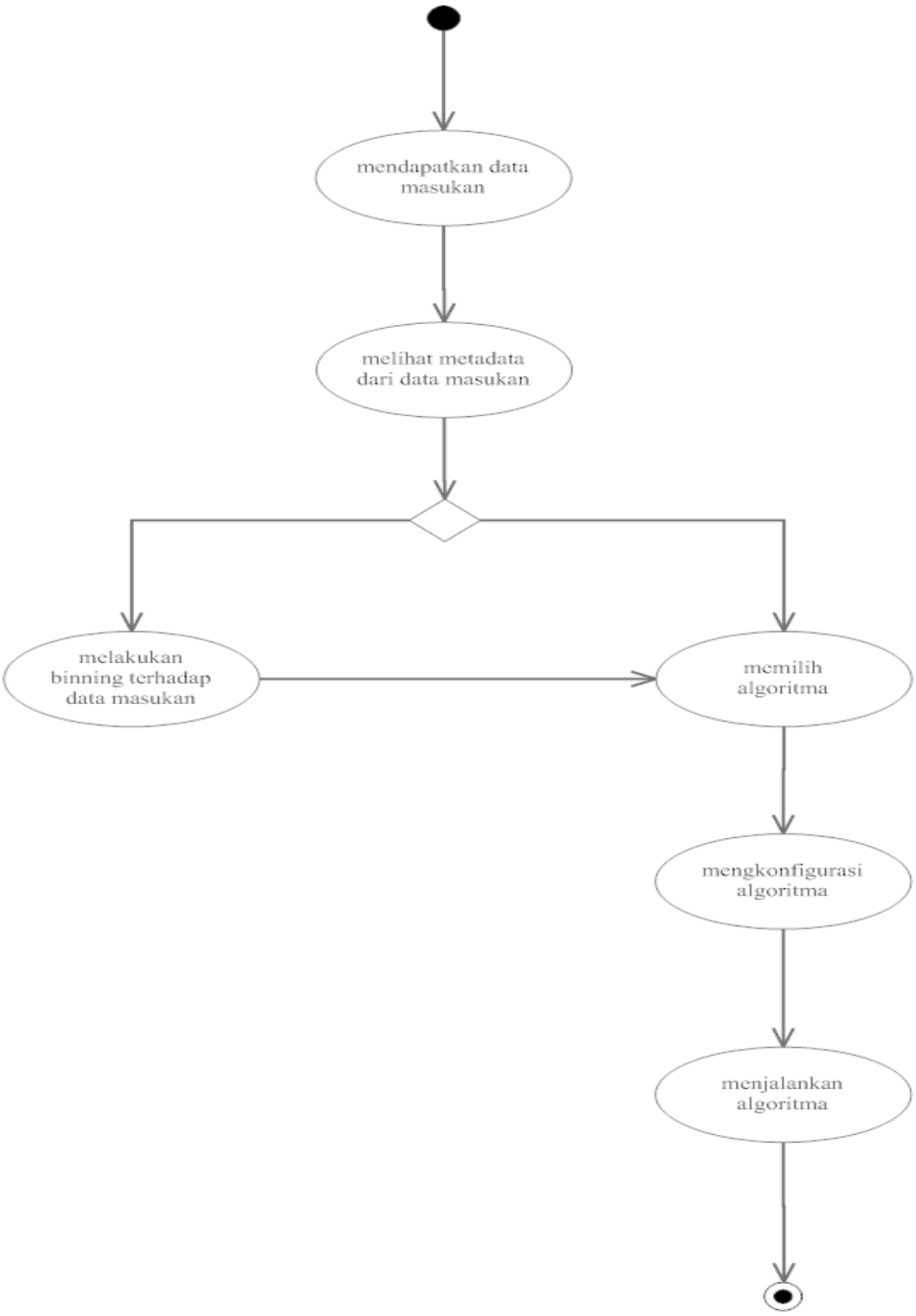
Aktor yang terdapat pada sistem ini hanya satu yaitu pengguna dari aplikasi. Gambaran keseluruhan interaksi antara aktor dengan *use case* terdapat pada Gambar 4.1.



Gambar 4.1 *Main use case diagram*

4.4.2 Analisis dan Perancangan *Input* dan *Output* Sistem

Analisis dan perancangan terhadap *input* dan o*utput* dari sistem bisa dilakukan jika *use case* telah terdefenisi karena o*utput* dari suatu *use case* bisa menjadi *input* bagi *use case* yang lain. Hubungan antar *use case* tersebut akan menciptkan rancangan alur dari sistem. Gambaran umum mengenai alur dari sistem secara keseluruhan terdapat pada Gambar 4.2.



Gambar 4.2 *Activity Diagram* Alur Proses Sistem

Alur dari sistem dimulai dengan proses mendapatkan data masukan dari sumber data. Sumber data yang berbeda – beda menciptakan tindakan yang berbeda pula dalam memproses data masukan. Tindakan yang dilakukan tersebut tergantung dari format penulisan yang dimiliki oleh masing – masing *file* sumber seperti yang dijelaskan pada sub-bab 2.4 tentang data masukan. Sedangkan bila sumber data

berasal dari *database,* sistem harus membuat koneksi terlebih dahulu dengan *database*. Membuat koneksi dilakukan dengan cara memasukkan beberapa parameter yang dibutuhkan, yaitu URL, *username,* dan *password.* URL berfungsi untuk menunjukkan lokasi dari *database,* sedangkan *username* dan *password* digunakan untuk *login* ke *database.* Setelah berhasil terhubung ke *database,* langkah selanjutnya adalah mendapatkan data dari tabel yang ada. Data bisa didapatkan dengan cara memasukkan *query*.

Dalam menjalankan prosesnya, *discretization* dan algoritma *data mining* membaca data dari beberapa *file.* Oleh karena itu, data masukan yang telah didapat sebelumnya harus dituliskan kedalam *file* yang dibutuhkan. Format penulisan pun berbeda tergantung dari proses apa yang dilakukan. Pada Tabel 4.2 berisi daftar *file* yang dibutuhkan oleh proses *discretization.*

Tabel 4.2 Format dan fungsi tiap *file*

Extensions	Format penulisan data	Fungsi
.ALL	Tiap baris mewakili satu *instance*. Tiap nilai dipisahkan oleh koma. Nilai terakhir merupakan label dari instance tersebut	Menyimpan semua *instance* yang terdapat pada data tersebut
.DATA	Tiap baris mewakili satu *instance*. Tiap nilai dipisahkan oleh koma. Nilai terakhir merupakan label dari instance tersebut	Berisi 2/3 dari total *instance* yang ada.
.TEST	Tiap baris mewakili satu *instance*. Tiap nilai dipisahkan oleh koma. Nilai terakhir merupakan label dari instance tersebut	Berisi 1/3 dari total *instance* yang ada.
.NAMES	Baris pertama berisi semua nilai	Berisi informasi tentang nama dan nilai dari tiap atribut, dan

	dari label. Baris 2 – n berisi nama diikuti nilai tiap atribut. Khusus untuk data dengan tipe *numeric*, nilai atributnya adalah *continuous*	nilai dari label

Algoritma membutuhkan dua *file* agar prosesnya bisa berjalan, yaitu *file* .data dan .names. *File* .data berisi daftar semua *instance*, sedangkan *file* .names berisi informasi mengenai nama atribut beserta nilainya dan nilai dari label. Namun, terdapat pengecualian jika data yang diterima berasal dari hasil proses *discretization*. Pada proses tersebut, *file* .names yang dihasilkan mempunyai beberapa karakter yang tidak bisa dibaca oleh algoritma. Sehingga, perlu dilakukan sedikit perubahan terhadap penulisan pada *file* .names agar algoritma bisa memprosesnya. Format penulisan dari *file* .names terdapat pada Tabel 4.3.

Tabel 4.3 Format penulisan *file* .names

//nilai awal atribut sebelum transformasi
Iris-setosa, Iris-versicolor, Iris-virginica, Iris-setosa, Iris-versicolor, Iris-virginica.
sepallength: - 5\.55, 5\.55-6\.15, 6\.15+.
sepalwidth: - 2\.95, 2\.95-3\.35, 3\.35+.
petallength: - 2\.45, 2\.45-4\.75, 4\.75+.
petalwidth: - 0\.8, 0\.8-1\.75, 1\.75+.
//nilai akhir atribut setelah transformasi
the target attribute: class
sepallength : bin0, bin1, bin2

sepalwidth : bin0, bin1, bin2

petallength : bin0, bin1, bin2

petalwidth : bin0, bin1, bin2

class: Iris-setosa, Iris-versicolor, Iris-virginica, Iris-setosa, Iris-versicolor, Iris-virginica

Sedangkan, format penulisan dari *file* .data terdapat pada Tabel 4.4.

Tabel 4.4 Format penulisan *file* .data

//berkas .data sebelum dilakukan perubahan

- 5\.55, 3\.35+, - 2\.45, - 0\.8, Iris-setosa.

- 5\.55, 2\.95-3\.35, - 2\.45, - 0\.8, Iris-setosa.

- 5\.55, 2\.95-3\.35, - 2\.45, - 0\.8, Iris-setosa.

- 5\.55, 2\.95-3\.35, - 2\.45, - 0\.8, Iris-setosa.

- 5\.55, 3\.35+, - 2\.45, - 0\.8, Iris-setosa.

- 5\.55, 3\.35+, - 2\.45, - 0\.8, Iris-setosa.

- 5\.55, 3\.35+, - 2\.45, - 0\.8, Iris-setosa.

- 5\.55, 3\.35+, - 2\.45, - 0\.8, Iris-setosa.

- 5\.55, - 2\.95, - 2\.45, - 0\.8, Iris-setosa.

- 5\.55, 2\.95-3\.35, - 2\.45, - 0\.8, Iris-setosa.

//Berkas .data setelah dilakukan perubahan

bin0,bin2,bin0,bin0,Iris-setosa

```
bin0,bin1,bin0,bin0,Iris-setosa

bin0,bin1,bin0,bin0,Iris-setosa

bin0,bin1,bin0,bin0,Iris-setosa

bin0,bin2,bin0,bin0,Iris-setosa

bin0,bin2,bin0,bin0,Iris-setosa

bin0,bin2,bin0,bin0,Iris-setosa

bin0,bin2,bin0,bin0,Iris-setosa

bin0,bin0,bin0,bin0,Iris-setosa

bin0,bin1,bin0,bin0,Iris-setosa
```

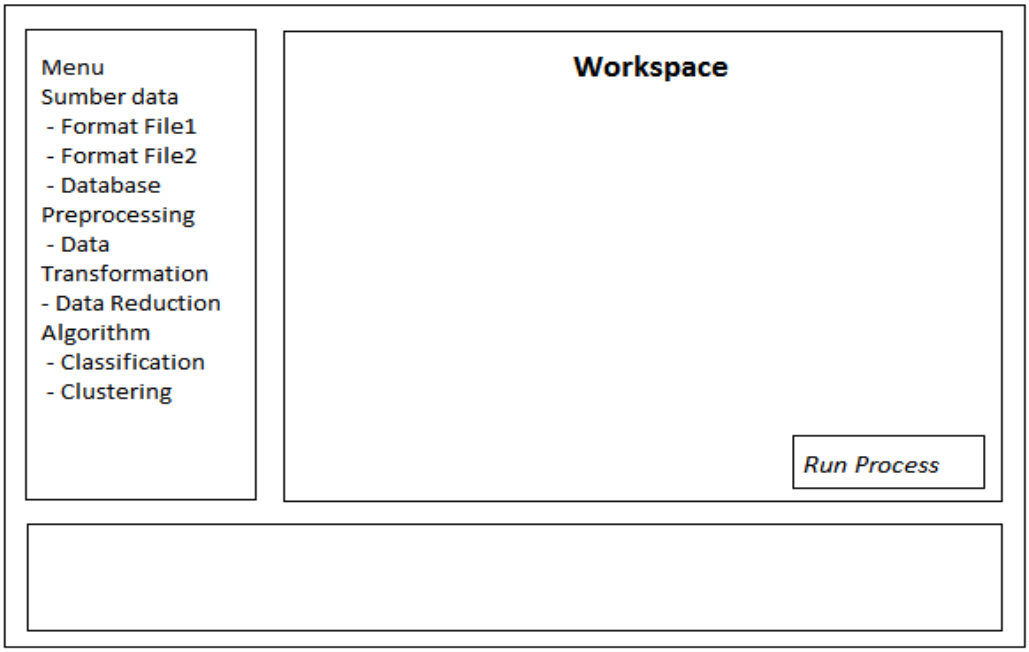
Data telah tersedia dan siap diproses oleh algoritma. Proses yang dilakukan oleh algoritma akan menghasilkan suatu keluaran. Keluaran tersebut berisi beberapa informasi, yaitu model, akurasi data, dan jumlah *error*. Format penulisan dari keluaran diserahkan sepenuhnya oleh pengembang algoritma. Pada penelitian ini, penulis menyediakan suatu area berisi teks untuk menampung keluaran tersebut.

4.4.3 Analisis dan Perancangan *User Interface*

Dari *use case* yang teridentifikasi dan alur dari sistem diperoleh gambaran secara umum mengenai sistem yang dirancang. *Use case* akan memberikan keterangan mengenai jendela konfigurasi dan jendela informasi yang perlu dibuat. Sebagai contoh, *use case* mengkonfigurasi algoritma membutuhkan satu jendela agar dapat melakukan konfigurasi tersebut. Sedangkan alur dari sistem akan memberikan informasi mengenai jendela mana yang perlu ditampilkan terlebih dahulu, serta beberapa komponen yang tidak bisa diakses pada suatu *state* tertentu.

4.4.3.1 Gambaran Umum *User Interface* dan Jendela Awal Aplikasi

Semua tahap yang ada pada aplikasi ini dilakukan melalui *graphical user interface*. Tampilan awal dari proses aplikasi ini disebut dengan jendela utama yang berfungsi sebagai pusat dari kegiatan pengguna dalam menjalankan aplikasi. Secara umum, setiap proses utama pada aplikasi ini menggunakan konsep *click and drop*. Setiap proses yang dilakukan tersebut bersumber dari daftar berbentuk *tree* yang berada disebelah kiri jendela. Konsep *click and drop* ini adalah suatu metode untuk menampilkan operator pada *workspace*. Metode ini diharapkan dapat memberikan gambaran kepada pengguna mengenai alur dari tiap proses yang telah dilakukan. Pengguna bisa memilih proses yang diinginkan dengan cara mengklik satu kali pada label yang tertera pada daftar berbentuk *tree*. Kemudian, suatu *operator* muncul di dalam *workspace*. Operator tersebut mewakili proses yang sedang dilakukan pengguna. Selain itu, operator yang berada di dalam *workspace* juga berfungsi sebagai penghubung dengan suatu jendela konfigurasi. Untuk membuka jendela konfigurasi, pengguna harus mengklik dua kali operator tersebut. Ketika pengguna ingin mengganti proses yang telah dipilih dengan proses yang lain, pengguna harus menghapus *operator* yang ada di *workspace* terlebih dahulu dengan cara mengklik kanan *operator* tersebut. Sebuah menu *delete* akan muncul dan bila diklik akan menghapus *operator* yang ada di *workspace.* Dibagian bawah jendela utama terdapat tiga *tab* yang masing – masing berisi *text area* dengan fungsi yang berbeda. *Tab* tersebut berisi *text area* untuk menampung catatan dari proses yang telah atau sedang dilakukan pengguna, deskripsi dari algoritma yang digunakan pengguna, dan deskripsi kesalahan yang mungkin terjadi ketika pengguna melakukan suatu proses. Dibawah ini adalah rancangan jendela awal antar muka aplikasi pada Gambar 4.3.

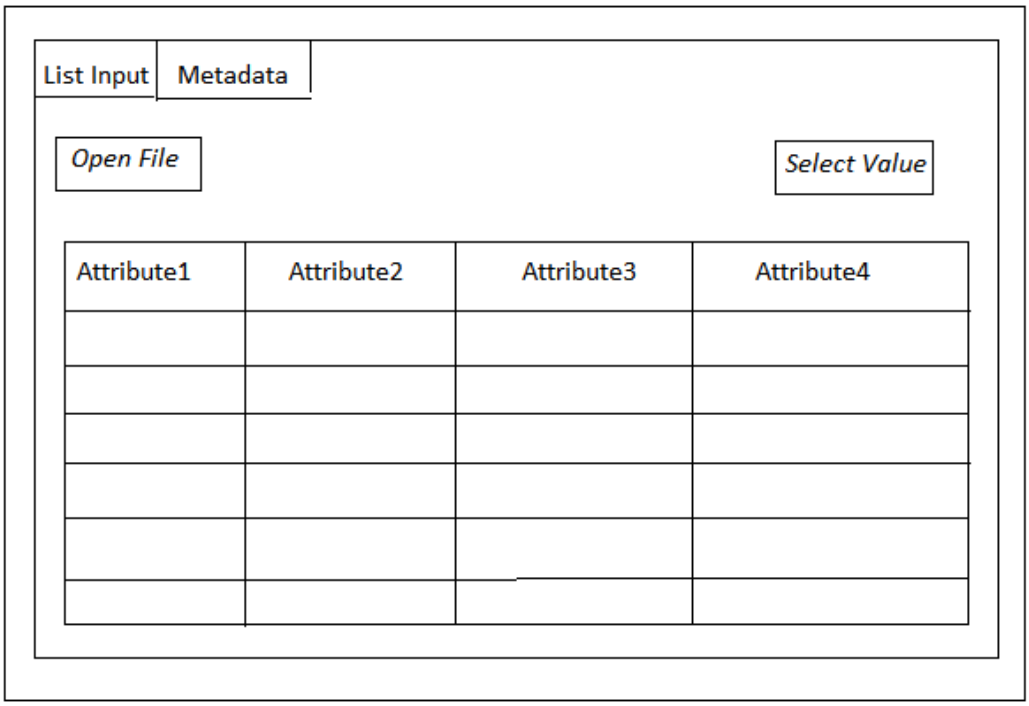


Gambar 4.3 Rancangan Jendela Awal Aplikasi

4.4.3.2 Jendela Konfigurasi Masukan dari *File*

Jendela konfigurasi masukan dari *file* berfungsi sebagai antar muka bagi pengguna dalam proses mendapatkan data masukan dan juga menampilkan informasi dari metadata kepada pengguna.

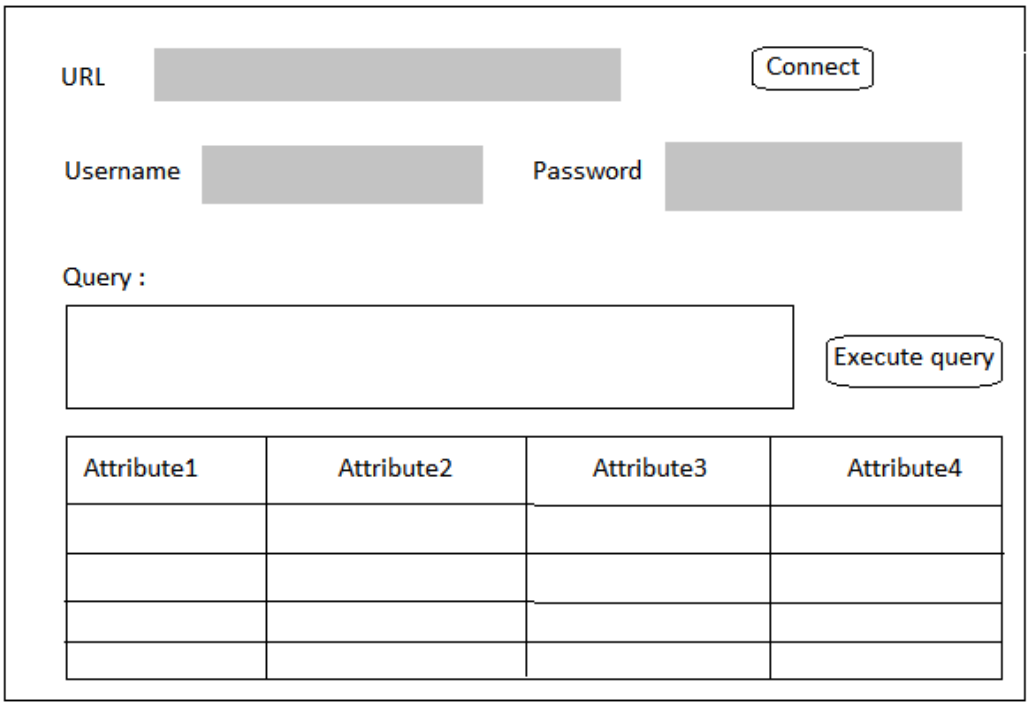
Konfigurasi masukan terdapat pada salah satu *tab* pada jendela ini. Lokasi dari *file input* diakses melalui suatu tombol. Setelah *file* didapatkan, maka data masukan akan ditampilkan ke suatu tabel. Sedangkan untuk menampilkan informasi metadata, terdapat *tab* lain untuk menampilkannya. Pada *tab* tersebut terdapat tabel untuk menampilkan atribut dan panel untuk menampilkan informasi yang lain. Pada Gambar 4.4 merupakan rancangan jendela konfigurasi masukan.



Gambar 4.4 Rancangan Jendela Konfigurasi Masukan

4.4.3.3 Jendela Konfigurasi Masukan dari *Database*

Terdapat dua langkah yang harus dilakukan untuk mendapatkan data dari *database*, yaitu membuka koneksi ke *database* dan memasukkan *query*. Maka, jendela konfigurasi masukan dari *database* harus mempunyai komponen yang bisa melayani kedua hal diatas. Pada Gambar 4.5 merupakan rancangan tampilan dari jendela konfigurasi masukan dari *database*.



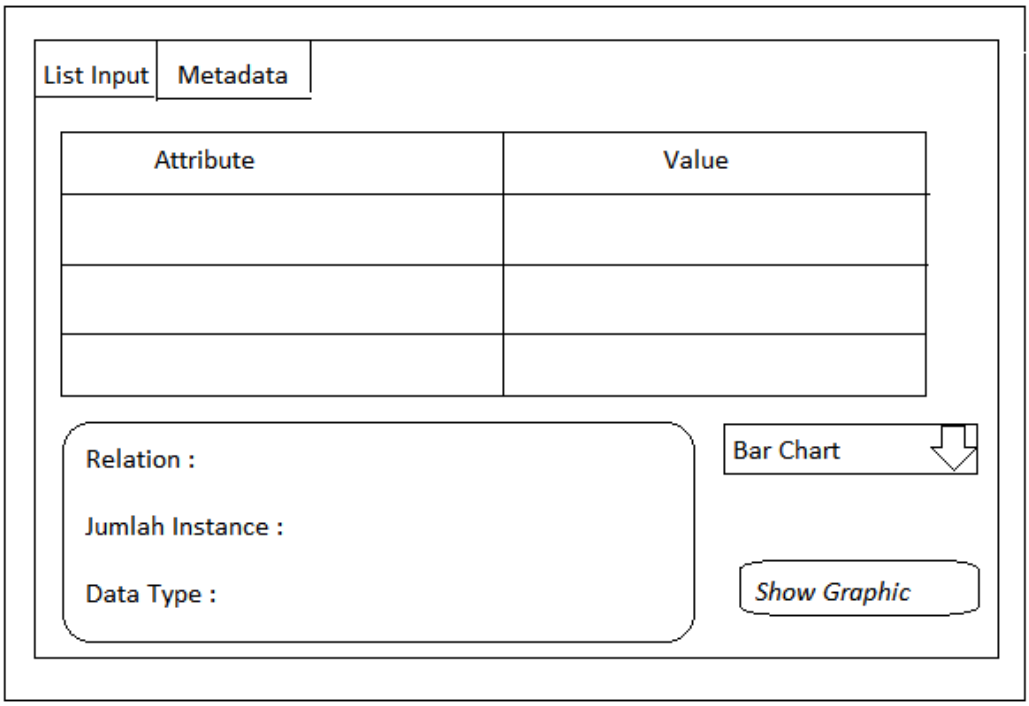
Gambar 4.5 Rancangan Jendela Konfigurasi *Database*

4.4.3.4 Jendela Informasi Metadata

Metadata perlu ditampilkan karena kebutuhan untuk mengetahui karakteristik dari data, apakah data yang tersedia cocok dipakai untuk algoritma yang akan digunakan. Metadata yang disediakan yaitu :

- Daftar atribut dan nilainya
- Jumlah nilai tiap atribut
- Grafik yang menampilkan frekuensi dari nilai tiap atribut
- Jumlah *instance* yang tersedia
- Nama relasi
- Tipe data

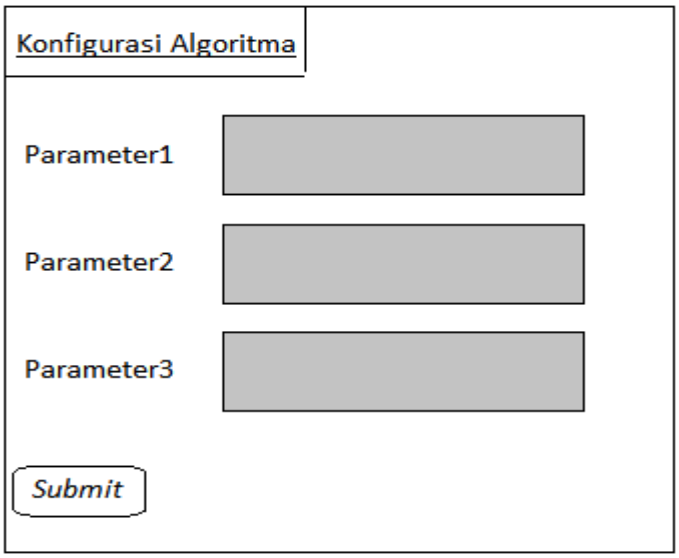
Pada Gambar 4.6 merupakan rancangan dari tampilan jendela metadata.



Gambar 4.6 Rancangan Jendela Metadata

4.4.3.5 Jendela Konfigurasi Algoritma

Jendela konfigurasi algoritma berisi parameter yang dibutuhkan oleh algoritma. Parameter tersebut diatur nilainya sesuai dengan yang dikehendaki oleh pengguna. Pada Gambar 4.7 merupakan rancangan jendela konfigurasi algoritma.



Gambar 4.7 Rancangan Jendela Konfigurasi Algoritma

4.4.4 Analisis dan Perancangan Proses

Pada tahap ini akan dijelaskan perancangan dari proses aplikasi. Penjabaran lebih detil dari *use case* terdapat pada *use case specification.* Pada *use case specification,* terdapat informasi tentang interaksi antara aktor dengan sistem melalui *user interface.* Sedangkan, untuk alur proses dari tiap *use case* akan tergambar pada *activity diagram.*

Beberapa algoritma *data mining* yang telah dikembangkan akan digabungkan kedalam aplikasi ini. Tiap algoritma mempunyai jumlah parameter yang unik, sehingga membutuhkan usaha yang lebih untuk membuat tiap jendela konfigurasi algoritma. Oleh karena itu, penulis merancang agar jendela konfigurasi dibuat ketika aplikasi berjalan. Selanjutnya, para pengembang harus menetapkan jumlah, nama, dan tipe data dari parameter didalam algoritma agar pembuatan jendela saat aplikasi berjalan bisa dilakukan. Sedangkan, untuk berkas masukan bagi algoritma berasal dari dua berkas yaitu berkas .data dan .names. Berkas .data akan berisi semua *instance* sedangkan berkas .names akan berisi informasi atribut. Format penulisan dari berkas masukan dibakukan agar semua pengembang harus mengikuti format yang diberikan. Hasil keluaran dari proses yang dilakukan oleh algoritma akan ditampilkan pada suatu *text area.* Sehingga, para pengembang

algoritma harus menyediakan fungsi untuk menampilkan hasil keluaran kedalam *source code*nya. Selanjutnya, akan dijabarkan u*se case specification* untuk tiap *use case* yang ada diaplikasi ini.

Use case specification mendapatkan *input* dari *file* terdapat pada tabel 4.5.

Tabel 4.5 *Use case specification* mendapatkan *input* dari *file*

Use-Case Name	Mendapatkan *Input* dari *File*	
Basic Flow	1. Pengguna memilih format *file*	
		2. Sistem menampilkan operator pada workspace
	3. Pengguna mengklik dua kali pada operator	
		4. sistem menampilkan jendela konfigurasi *input*
	5. Pengguna memilih lokasi *file*	
		6. Sistem memisahkan nilai lalu disimpan pada struktur data relasi, atribut, *instance*
	7. Sistem menampilkan *instance* ke tabel	
		8. Pengguna memilih semua atau beberapa nilai *instance*
	9. Sistem menampilkan *instance* yang terpilih	
Alternative Flow	4a. Jika pengguna ingin mengganti format *file* atau menggunakan *database*, maka pengguna bisa menutup jendela konfigurasi *input* dan memilih format *file* yang lain atau *database*	
	7a. Pengguna bisa melihat metadata dari *input* pada *tab* metadata	
Special Requirement		
Preconditions	Aplikasi telah dijalankan	
Postconditions	Sistem telah mempunyai data *input* yang siap digunakan	

	untuk proses selanjutnya
Extension Points	

Use case specification mendapatkan *input* dari *database* terdapat pada Tabel 4.6.

Tabel 4.6 *Use case specification* mendapatkan *input* dari *database*

Use-Case Name	Mendapatkan *Input* dari *Database*	
Basic Flow	1. Pengguna memilih label *database*	
		2. Sistem menampilkan operator pada workspace
	3. Pengguna mengklik dua kali pada operator	
		4. sistem menampilkan jendela konfigurasi *database*
	5. Pengguna memasukkan URL, *username, password*	
		6. Sistem membuka koneksi ke *database*
	7. Pengguna memasukkan *query*	
		8. Sistem menampilkan *instance* ketabel
	9. Sistem menampilkan *instance* yang terpilih	
Alternative Flow	4a. Jika pengguna ingin menggunakan *file* sebagai *input*, maka pengguna bisa menutup jendela konfigurasi *database* dan memilih format *file*	
	6a. Pengguna bisa melihat metadata dari *database* pada *tab* metadata	
Special Requirement		
Preconditions	Aplikasi telah dijalankan	
Postconditions	Sistem telah mempunyai data *input* yang siap digunakan untuk proses selanjutnya	

Extension Points	

Use case specification melihat metadata terdapat pada tabel 4.7.

Tabel 4.7 *Use case specification* melihat metadata

Use-Case Name	Melihat Metadata	
Basic Flow	1. Pengguna memilih *tab* metadata	
		2. Sistem menampilkan informasi metadata
Alternative Flow		
Special Requirement		
Preconditions	Sistem sedang berada di jendela konfigurasi *input* dan data *input* telah didapatkan	
Postconditions		
Extension Points		

Use case specification menghapus operator terdapat pada tabel 4.8.

Tabel 4.8 *Use case specification* menghapus operator

Use-Case Name	Menghapus Operator	
Basic Flow	1. Pengguna mengklik kanan pada operator	
		2. Sistem akan

		menampilkan menu *delete*
	3. Pengguna mengklik satu kali pada menu *delete*	
		4. Sistem akan menghilangkan operator dari *workspace*
Alternative Flow		
Special Requirement		
Preconditions	Terdapat operator pada *workspace*	
Postconditions	Informasi yang terkandung pada operator tersebut akan hilang dari sistem	
Extension Points		

Use case specification menggunakan teknik *discretization* terdapat pada tabel 4.9.

Tabel 4.9 *Use case specification* menggunakan teknik *discretization*

Use-Case Name	Menggunakan Teknik *Discretization*	
Basic Flow	1. Pengguna memilih label *discretization*	
		2. . Sistem menampilkan operator pada workspace
	3. Pengguna mengklik dua kali pada operator	

		4. Sistem akan menampilkan aplikasi *discretization*
	5. Pengguna memasukkan nama berkas yang akan diproses dan nama berkas keluaran	
		6. Sistem melakukan proses *discretization* terhadap *input*
	7. Pengguna mengklik tombol *data binning*	
		8. Sistem menampilkan *instance* yang telah di *binning*
Alternative Flow		
Special Requirement		
Preconditions	Sistem telah mempunyai data *input*	
Postconditions	Data *input* akan ditransformasi kedalam bentuk *binning*	
Extension Points		

Use case specification memilih algoritma terdapat pada tabel 4.10

Tabel 4.10 *Use case specification* memilih algoritma

Use-Case Name	Memilih Algoritma

Basic Flow	1. Pengguna memilih algoritma pada menu	
		2.Sistem akan menampilkan operator pada *workspace*
Alternative Flow		
Special Requirement		
Preconditions	Sistem telah mempunyai data *input*	
Postconditions		
Extension Points		

Use case specification mengkonfigurasi algoritma terdapat pada Tabel 4.11.

Tabel 4.11 *Use case specification* mengkonfigurasi algoritma

Use-Case Name	Mengkonfigurasi Algoritma	
Basic Flow	1. Pengguna mengklik dua kali operator algoritma pada *workspace*	
		2. Sistem akan menampilkan jendela konfigurasi algoritma
	3. Pengguna memasukkan nilai pada parameter algoritma	
		4. Pengguna mengklik

		tombol *Submit*
Alternative Flow		
Special Requirement		
Preconditions	Telah terdapat operator algoritma pada *workspace*	
Postconditions	Sistem telah mempunyai informasi mengenai nilai dari parameter algoritma	
Extension Points		

Use case specification menjalankan algoritma terdapat pada Tabel 4.12.

Tabel 4.12 *Use case specification* menjalankan algoritma

Use-Case Name	Menjalankan Algoritma	
Basic Flow	1. Pengguna mengklik tombol run process pada jendela utama	
		2. Sistem akan menjalankan algoritma dan menampilkan keluaran pada jendela keluaran
Alternative Flow		
Special Requirement		
Preconditions	Parameter algoritma telah terkonfigurasi	

Postconditions	
Extension Points	

Use case specification menampilkan *bar chart* terdapat pada Tabel 4.13.

Tabel 4.13 *Use case specification* menampilkan *bar chart*

Use-Case Name	Menampilkan *Bar Chart*	
Basic Flow	1. Pengguna memilih *tab* metadata	
		2. . Sistem menampilkan jendela informasi metadata
	3. Pengguna memilih bentuk *bar chart*	
		4. Sistem akan membaca data *instance* dan menampilkan dalam bentuk *bar chart*
Alternative Flow		
Special Requirement		
Preconditions	Sistem telah mempunyai data *input* dan sedang berada di *tab* metadata	
Postconditions		
Extension Points		

Use case specification menampilkan s*catterplot* terdapat pada Tabel 4.14.

Tabel 4.14 *Use case specification* menampilkan s*catterplot*

Use-Case Name	Menampilkan *Scatterplot*	
Basic Flow	1. Pengguna memilih *tab* metadata	
		2. . Sistem menampilkan jendela informasi metadata
	3. Pengguna memilih bentuk *scatterplot*	
		4. Sistem menampilkan jendela pemilihan atribut
	5. Pengguna memilih satu atribut untuk ditampilkan	
		6. Sistem akan membaca data pada atribut yang bersangkutan dan menampilkan dalam bentuk *scatterplot*
Alternative Flow		
Special Requirement		
Preconditions	Sistem telah mempunyai data *input* dan sedang berada di *tab* metadata	
Postconditions		

Extension Points	

Use case specification menampilkan *line chart* terdapat pada Tabel 4.15.

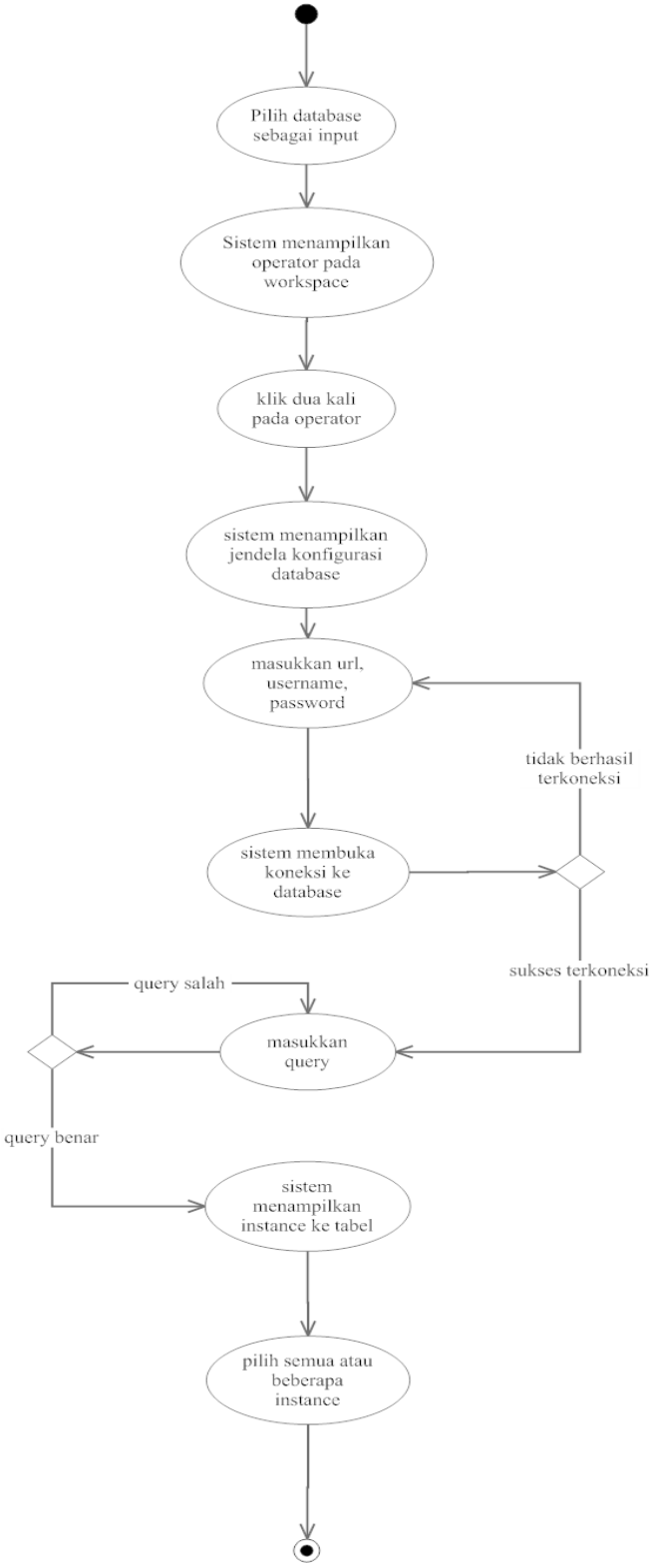
Tabel 4.15 *Use case specification* menampilkan l*ine chart*

Use-Case Name	Menampilkan *Line Chart*	
Basic Flow	1. Pengguna memilih *tab* metadata	
		2. . Sistem menampilkan jendela informasi metadata
	3. Pengguna memilih bentuk *line chart*	
		4. Sistem menampilkan jendela pemilihan atribut
	5. Pengguna memilih dua atribut untuk ditampilkan	
		6. Sistem akan membaca data pada atribut yang bersangkutan dan menampilkan dalam bentuk *line chart*
Alternative Flow		
Special Requirement		
Preconditions	Sistem telah mempunyai data *input* dan sedang berada di	

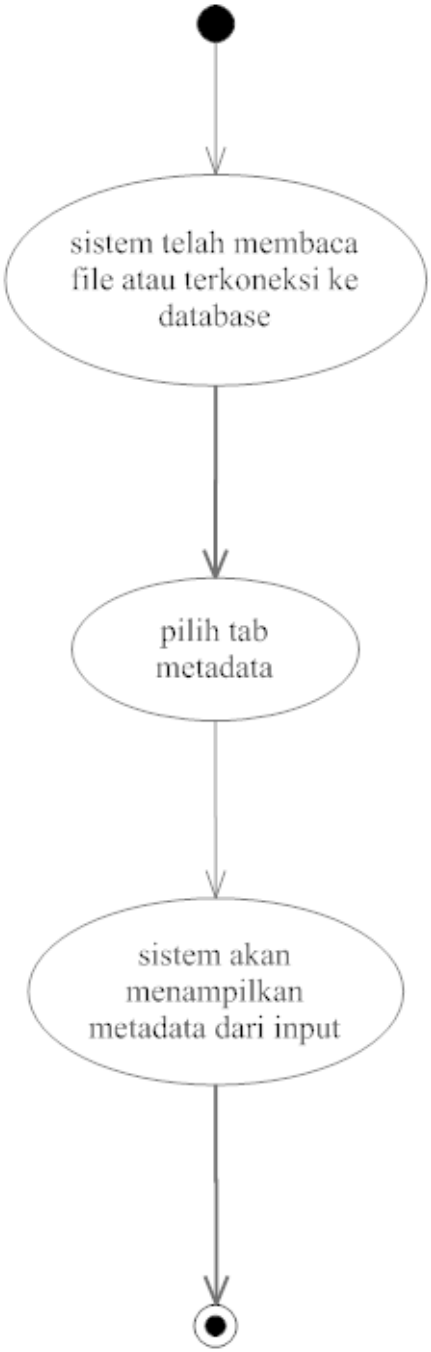
	tab metadata
Postconditions	
Extension Points	

Dibawah ini adalah gambar dari *activity diagram* untuk tiap *use case* yang ada pada aplikasi ini.

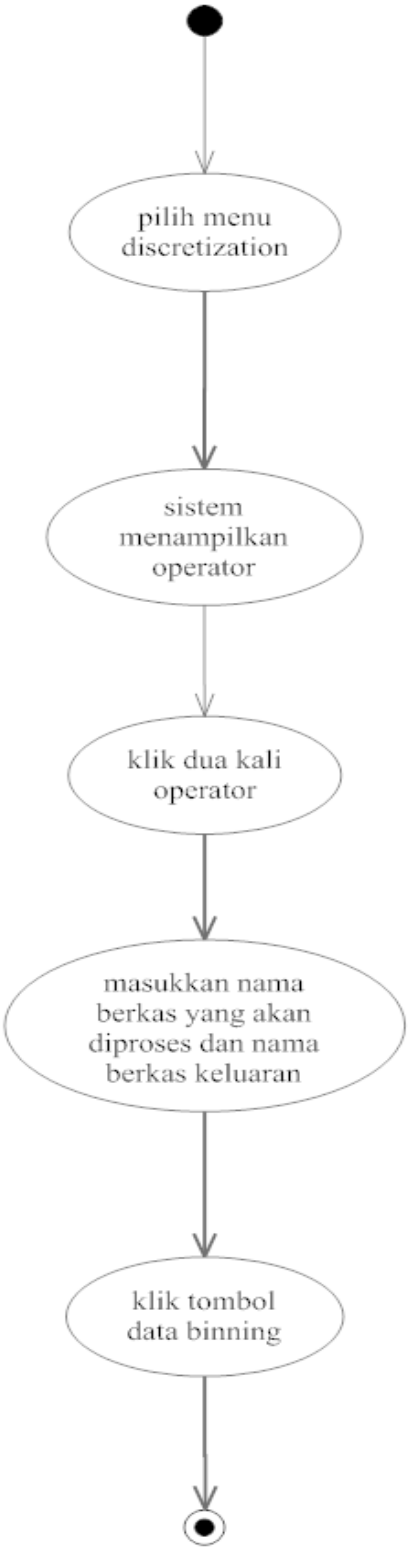
Gambar 4.8 *Activity diagram* mendapatkan data dari *file*



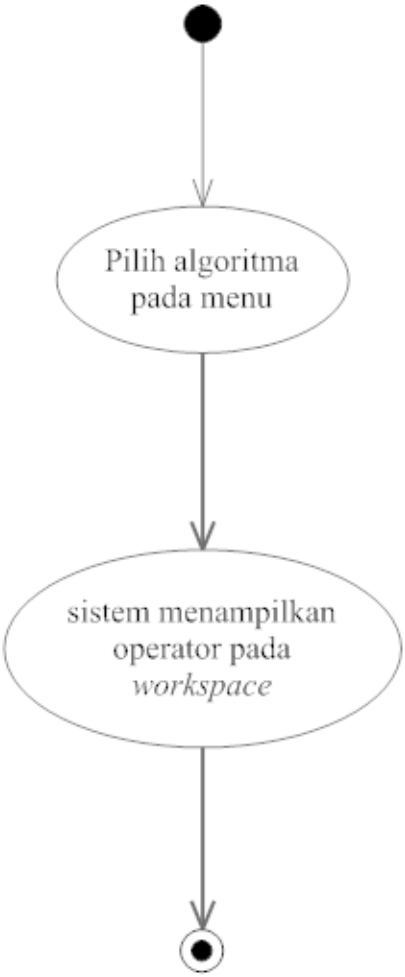
Gambar 4.9 *Activity diagram* mendapatkan data dari *database*



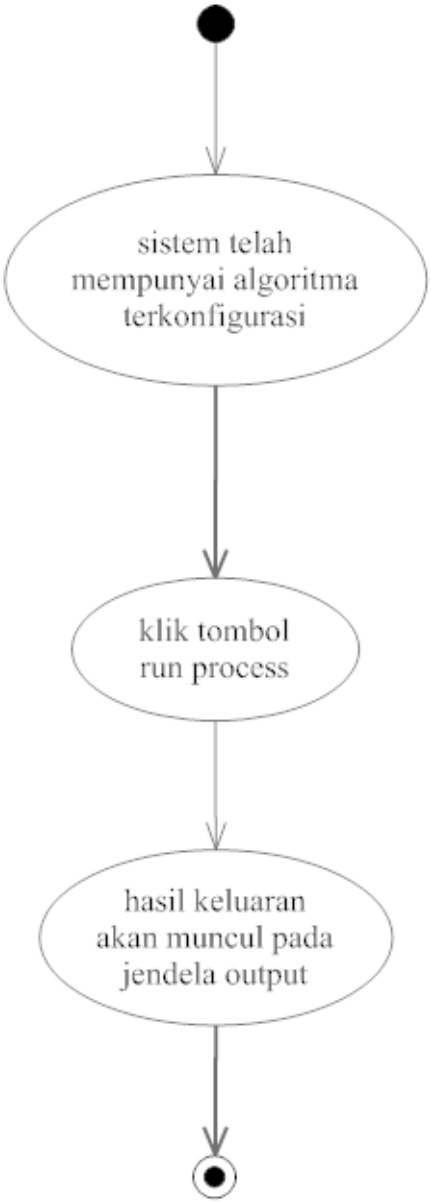
Gambar 4.10 *Activity diagram* melihat metadata



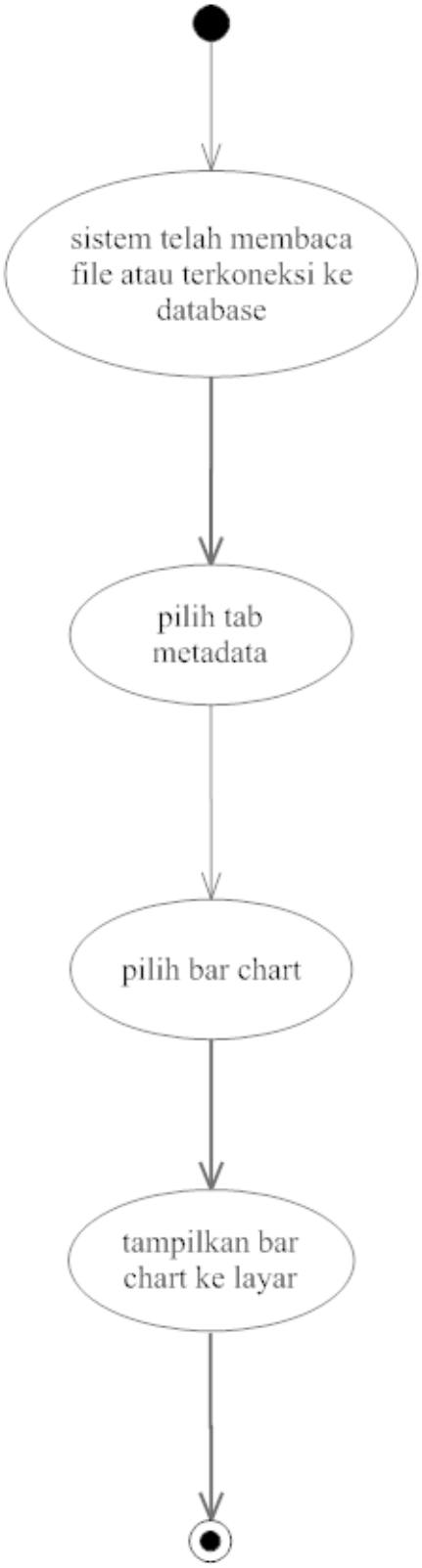
Gambar 4.11 *Activity diagram* melakukan proses *discretization*



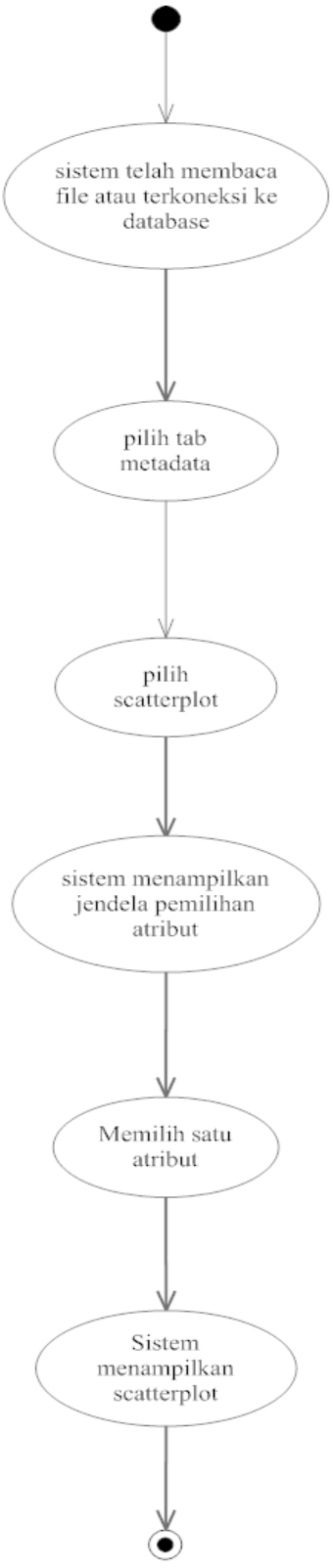
Gambar 4.12 *Activity diagram* memilih algoritma



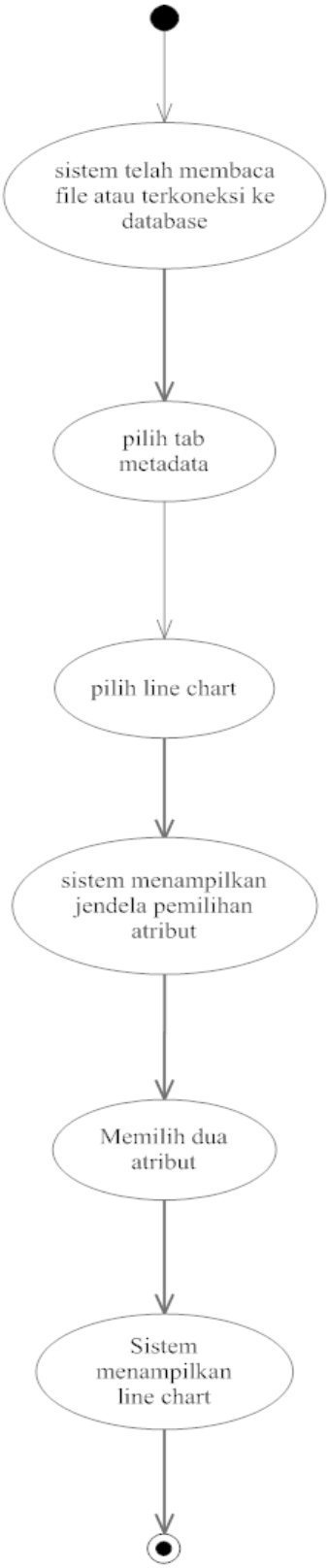
Gambar 4.13 *Activity diagram* menjalankan algoritma



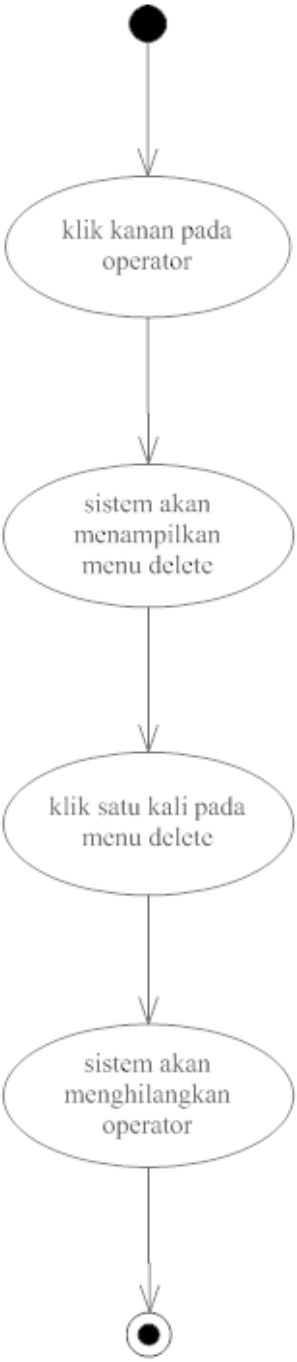
Gambar 4.14 *Activity diagram* menampilkan *bar chart*



Gambar 4.15 *Activity diagram* menampilkan *scatterplot*



Gambar 4.16 *Activity diagram* menampilkan *line chart*

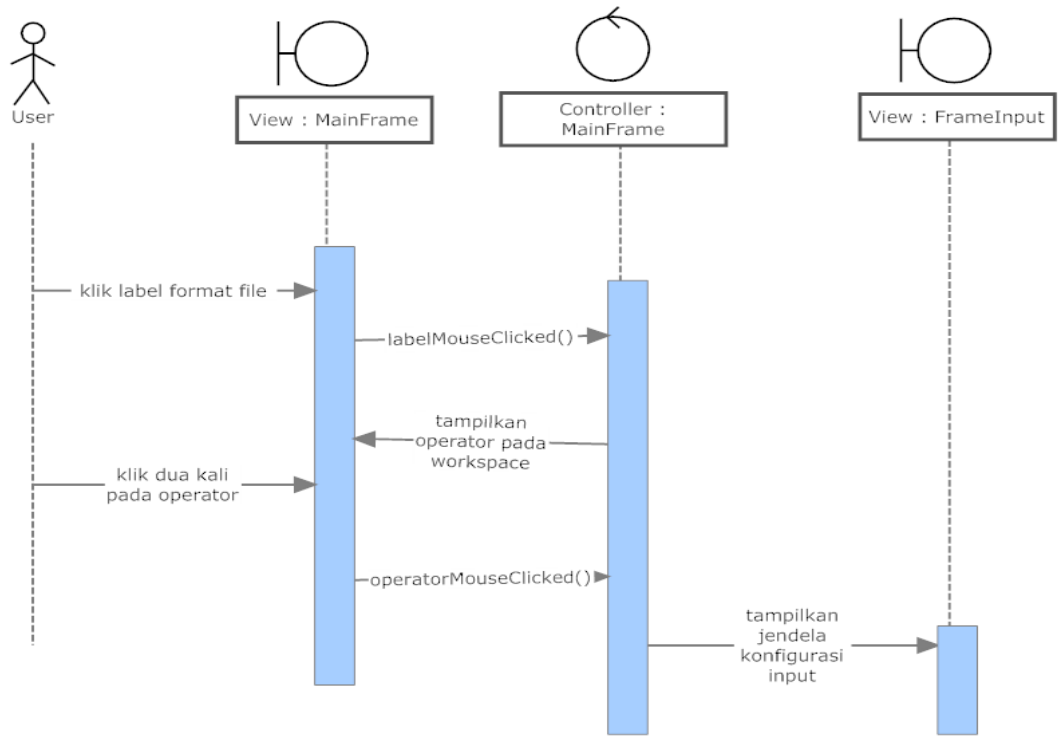


Gambar 4.17 *Activity diagram* menghapus operator

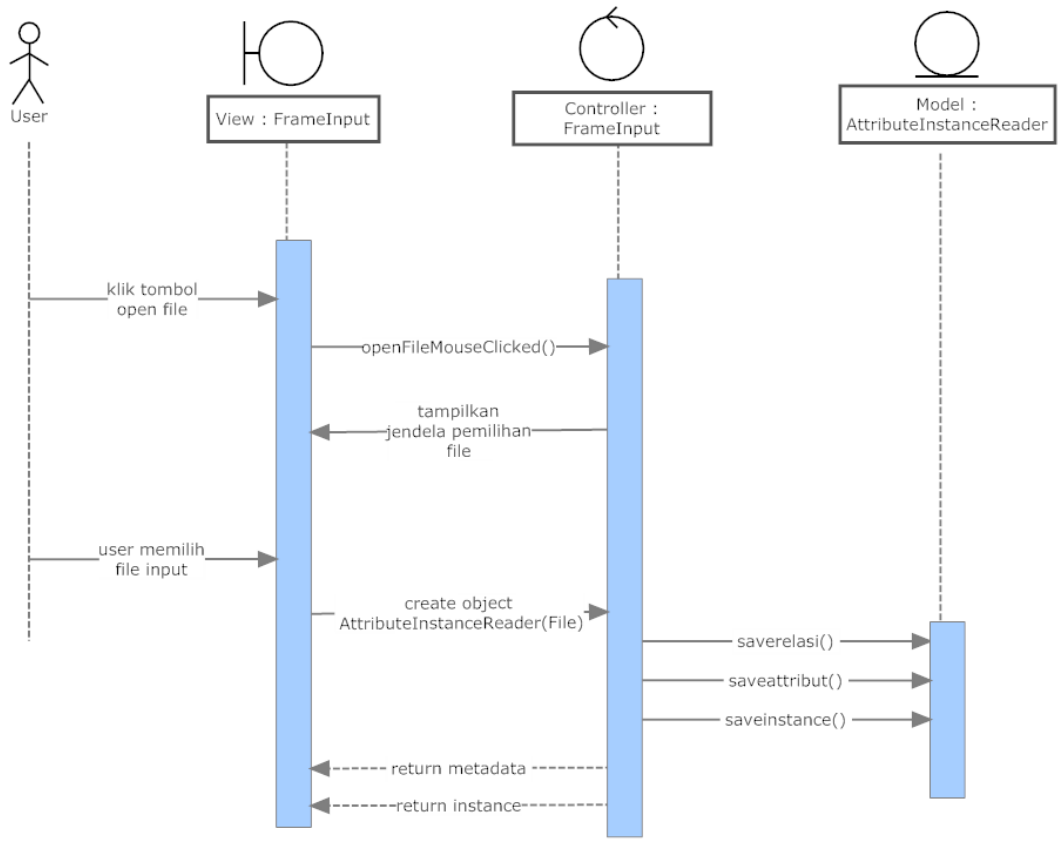
4.4.5 Sequence Diagram

Sequence diagram berguna untuk memodelkan aspek dinamis dari sistem. Pada diagram ini akan tergambar interaksi antar objek melalui pertukaran *message*.

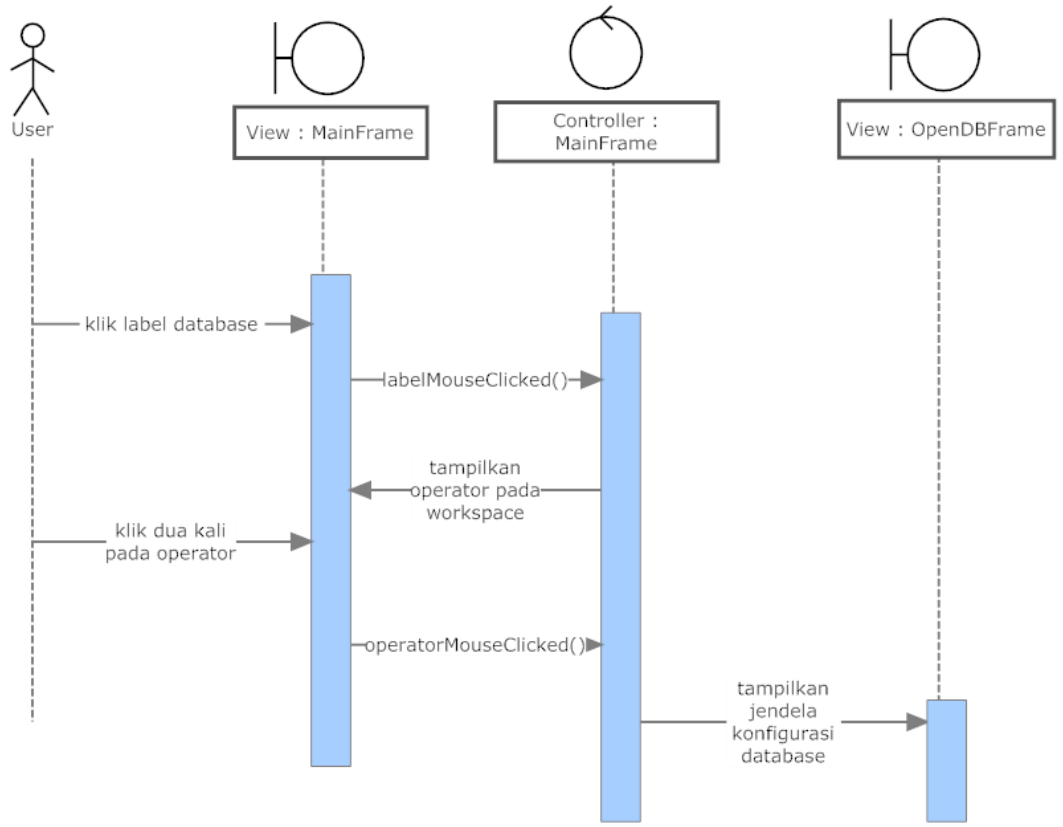
Sequence diagram akan menjelaskan skenario yang terdapat pada *use case* dalam aspek yang dinamis. Dibawah ini adalah *sequence diagram* dari penelitian ini.



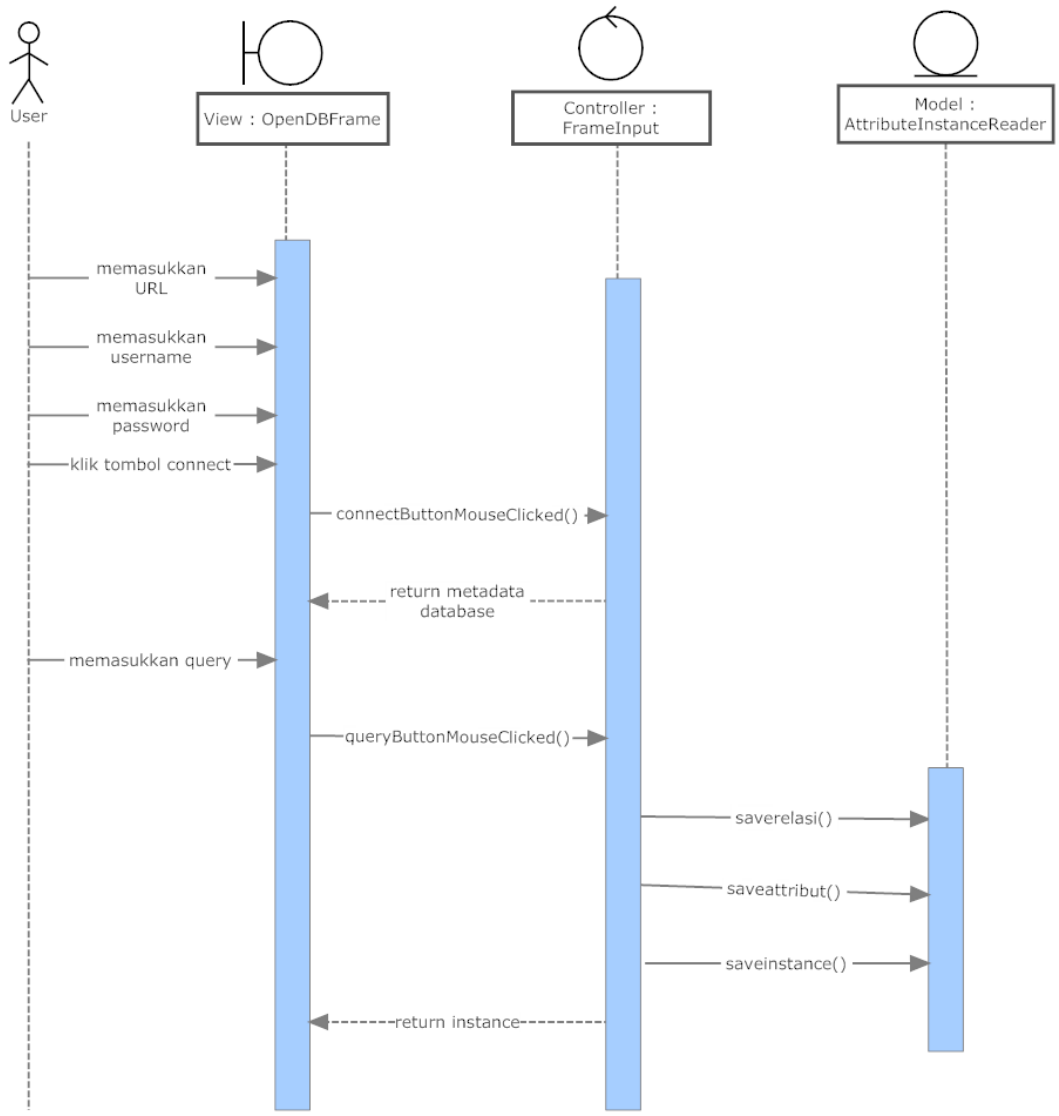
Gambar 4.18 *Sequence Diagram* menampilkan jendela konfigurasi *input*



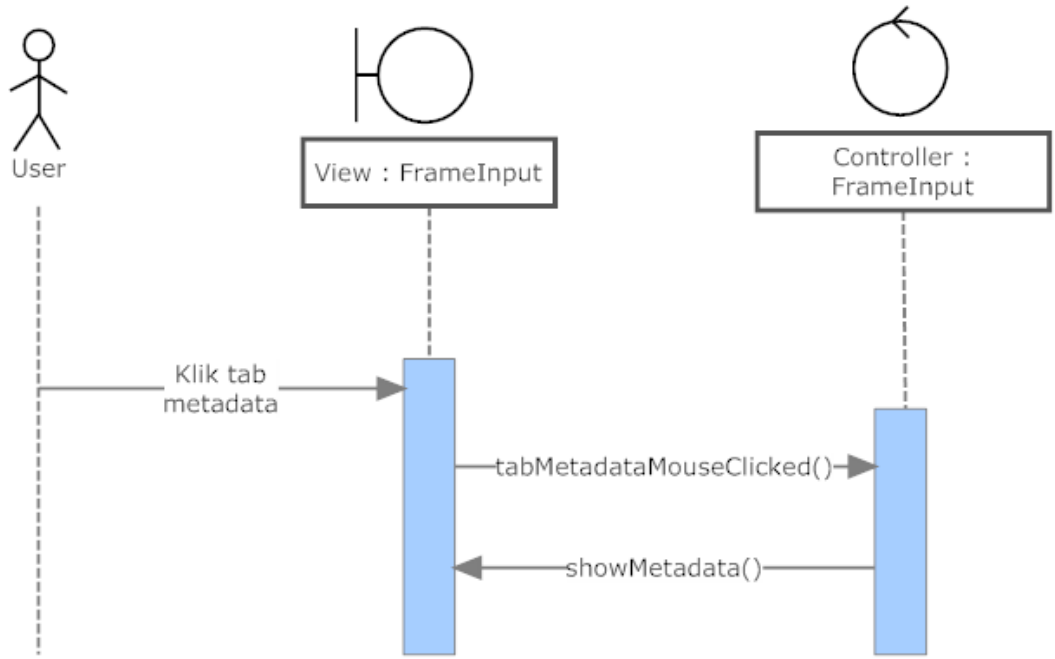
Gambar 4.19 *Sequence diagram* mendapatkan data *instance* dari *file*



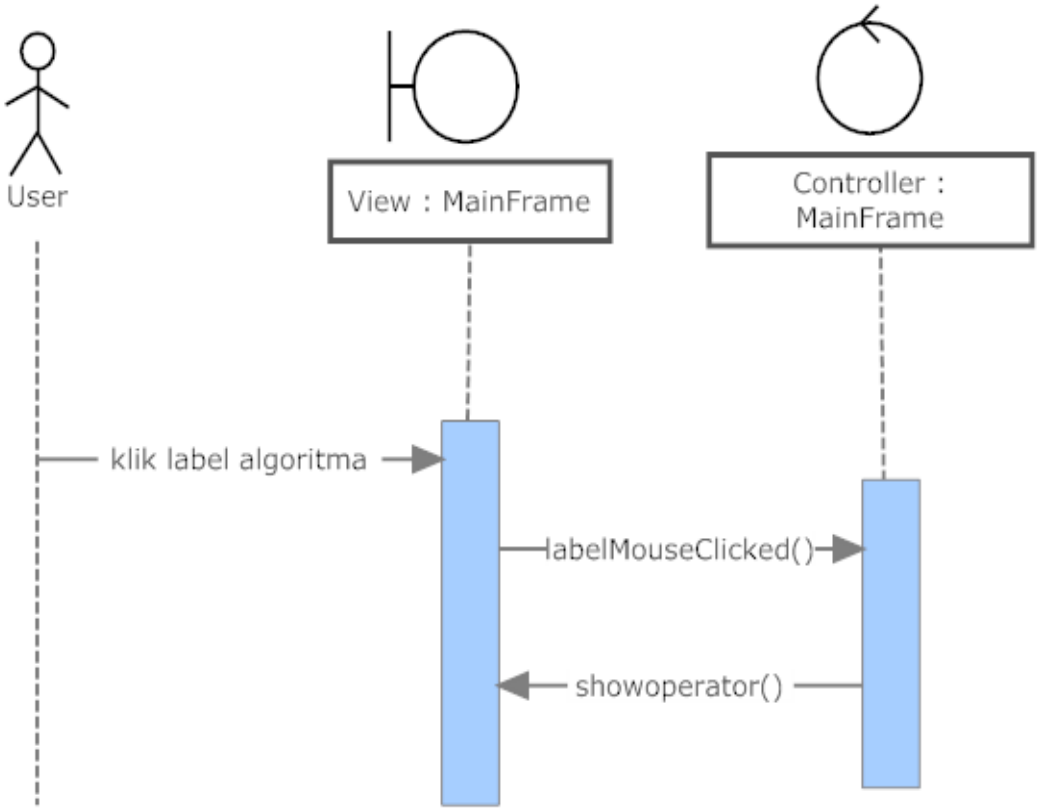
Gambar 4.20 *Sequence diagram* membuka jendela konfigurasi *database*



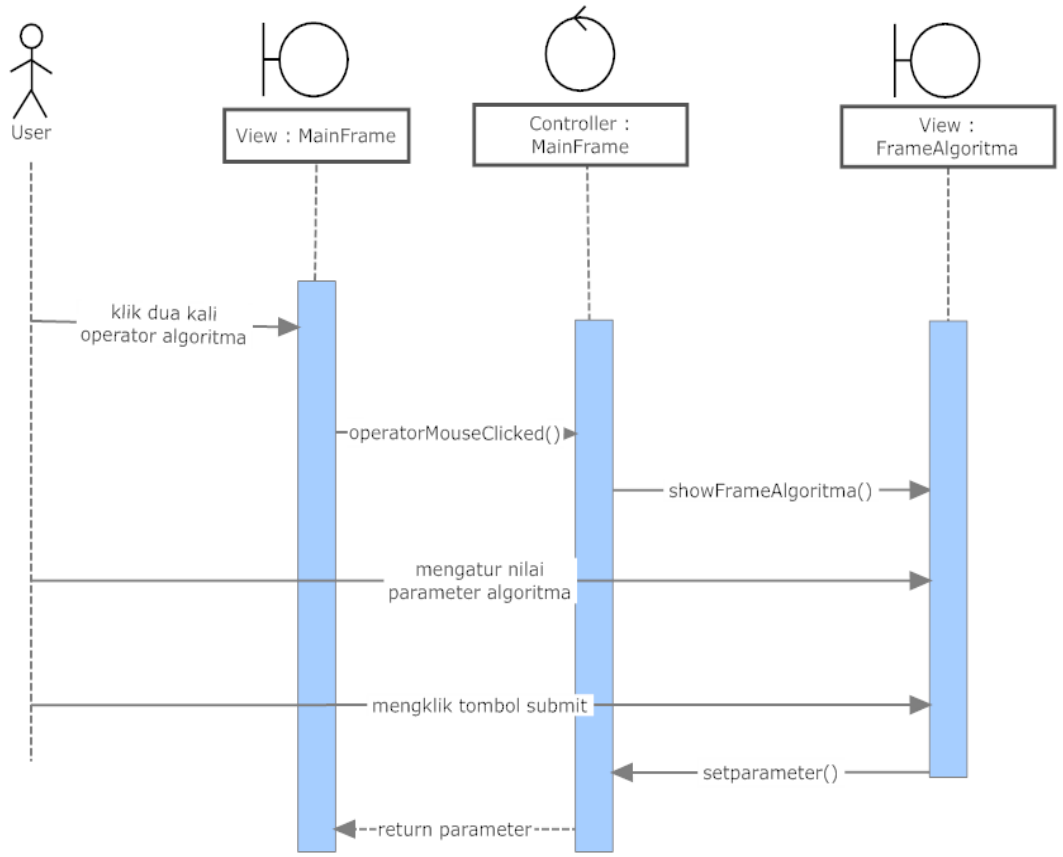
Gambar 4.21 *Sequence diagram* mendapatkan data *instance* dari *database*



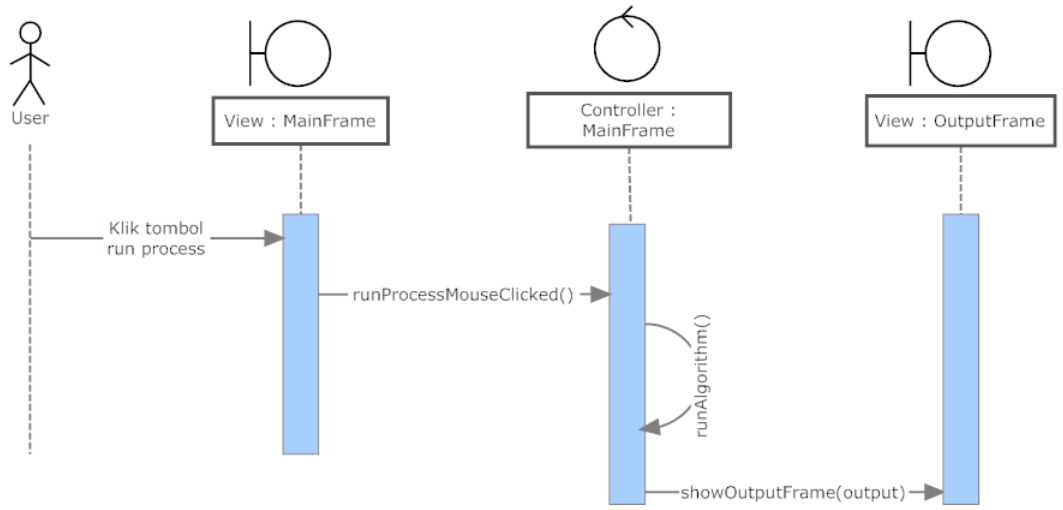
Gambar 4.22 *Sequence diagram* Melihat metadata



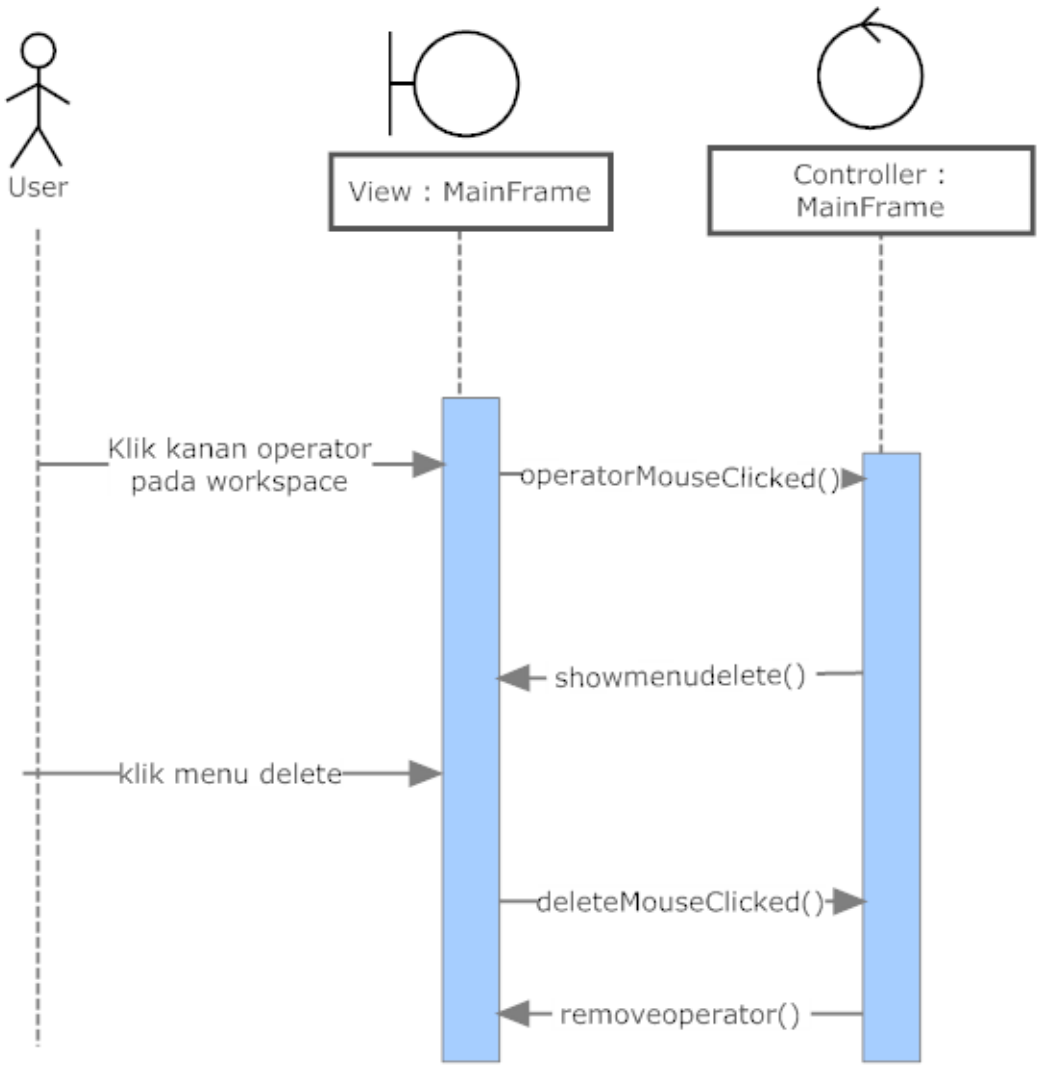
Gambar 4.23 *Sequence diagram* memilih algoritma



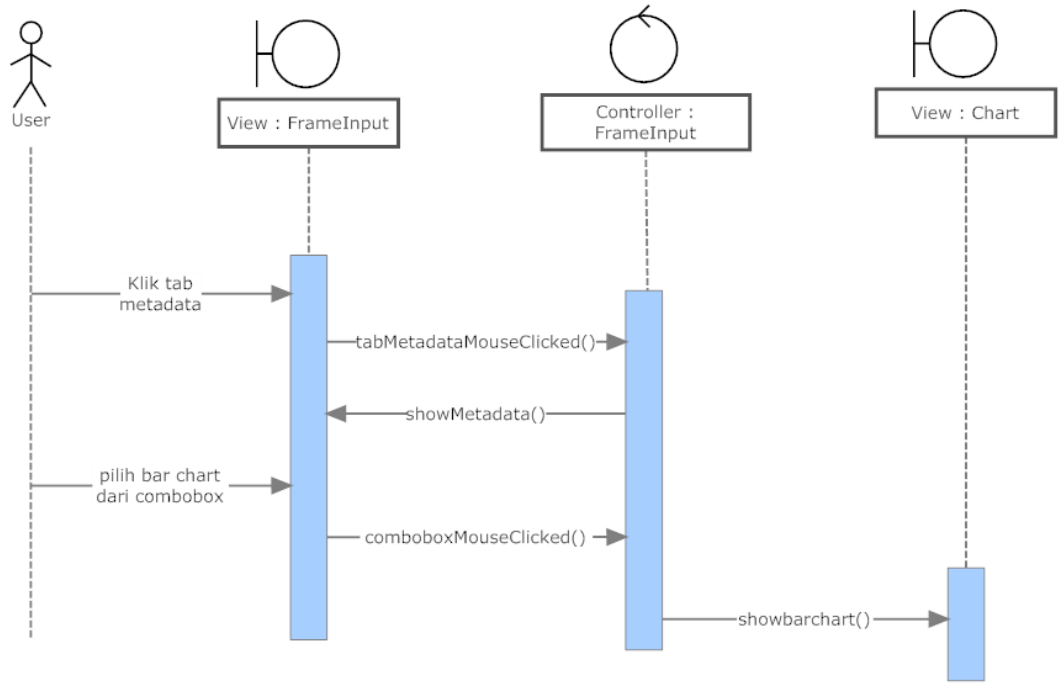
Gambar 4.24 *Sequence diagram* Mengkonfigurasi algoritma



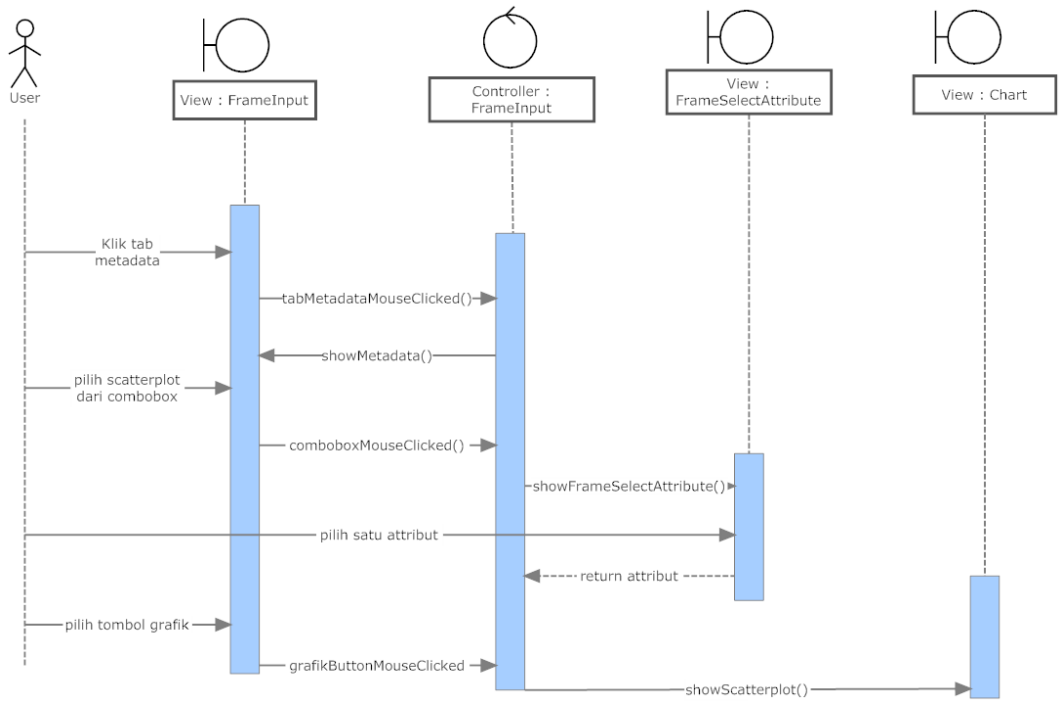
Gambar 4.25 *Sequence diagram* Menjalankan Algoritma



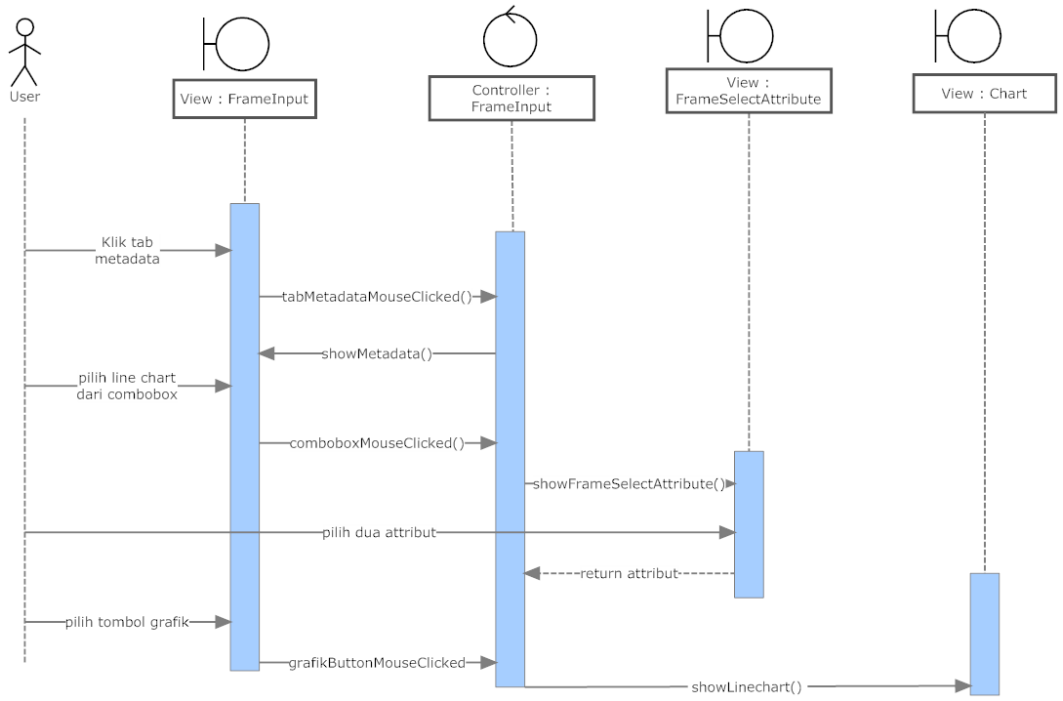
Gambar 4.26 *Sequence diagram* menghapus operator



Gambar 4.27 *Sequence diagram* menampilkan bar chart



Gambar 4.28 *Sequence diagram* menampilkan scatterplot



Gambar 4.29 *Sequence diagram* menampilkan line chart

4.4.6 Class Diagram

Hasil implementasi dari aplikasi ini berupa berkas – berkas Java berikut.

- MainFrame

  Berkas yang berfungsi sebagai pengatur jalannya aplikasi. Setiap tahap pemrosesan aplikasi dimulai dan berakhir pada jendela ini. *Workspace* yang terdapat pada jendela ini berfungsi untuk membantu pengguna dalam memonitor setiap tahap yang telah dilakukan. Dibagian bawah juga terdapat *text area* untuk menampilkan log, deskripsi algoritma, dan keterangan *error.*

- AttributeInstanceReader

  Berkas yang berfungsi mengambil masukan dari berkas ARFF, XLS, UCI, dan CSV, lalu menyimpannya sebagai *attribute*, relasi, atau *instance*. Proses implementasi awal masukan dijelaskan pada bab 4.1.

- OpenDBFrame

Berkas ini berfungsi untuk melakukan koneksi ke *database* dan menampilkan data dengan menulis *query*. Bila *query* yang dimasukkan benar, maka pada jendela OpenDBFrame akan muncul semua data.

- FrameInput

  Berkas yang berfungsi menampilkan semua *instance* dan metadata dari masukan.

- Chart

  Berkas ini berfungsi untuk menampilkan semua data dari masukan ke dalam bentuk grafik. Dalam pengoperasiannya, berkas ini memafaatkan *library* JFreeChart yang tersedia di internet. Grafik yang bisa ditampilkan dalam bentuk *bar chart*, *scatterplot*, dan *line chart*.

- FrameChosenInstance

  Berkas ini berfungsi untuk menampilkan daftar *instance* yang dipilih oleh pengguna.

- FrameParameterAlgoritma

  Berkas ini berfungsi untuk menampilkan parameter yang diperlukan oleh algoritma yang dipilih. Pengguna bisa menentukan nilai dari parameter tersebut.

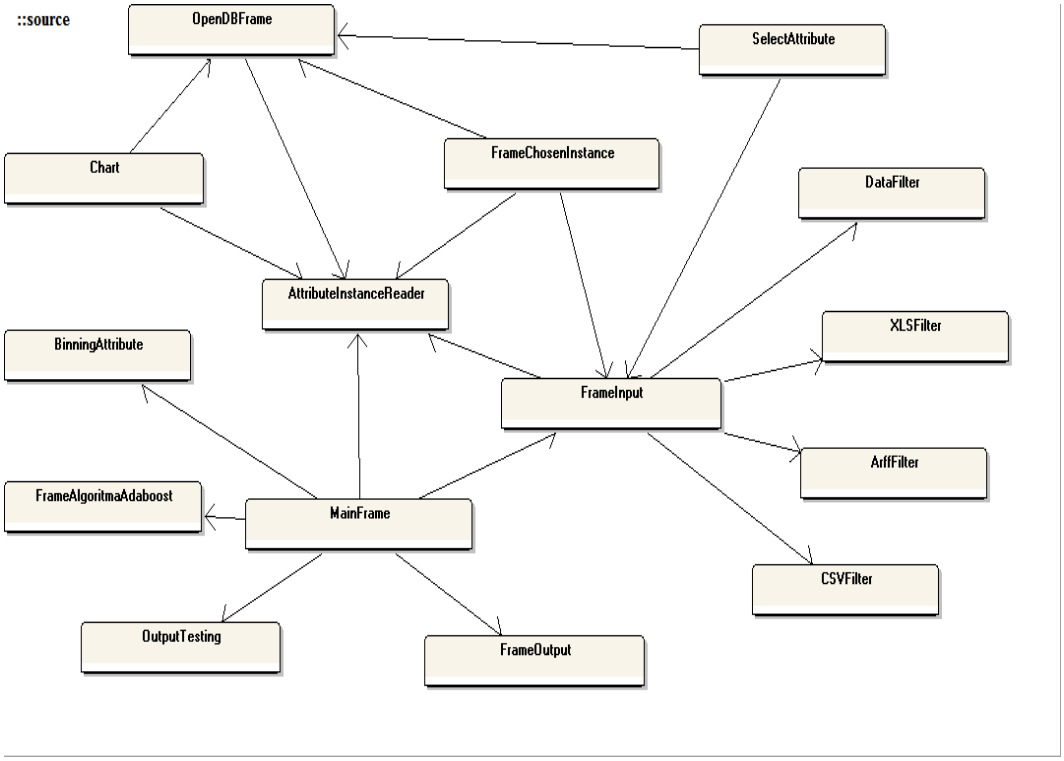
- Output

  Berkas ini berfungsi untuk menampilkan hasil keluaran dari proses yang telah dilakukan oleh algoritma. Hasil keluaran akan ditampilkan didalam *text area*.

- SelectAttribute

  Berkas ini berfungsi untuk memilih data dari atribut yang akan ditampilkan dalam bentuk grafik.

Berikut gambar *class diagram* dari aplikasi yang penulis implementasikan.



Gambar 4.30 *Class Diagram* Aplikasi

BAB 5
IMPLEMENTASI

Pada bab ini akan dibahas implementasi terkait dengan perancangan implementasi yang sudah dibahas pada bab sebelumnya. Pembahasannya terdiri dari pembahasan implementasi pengolahan input pada sub-bab 4.1, pembahasan implementasi metadata pada sub-bab 4.2, pembahasan implementasi *binning* pada sub-bab 4.3, pembahasan implementasi grafik pada sub-bab 4.4, pembahasan implementasi konfigurasi algoritma pada sub-bab 4.5, pembahasan implementasi menjalankan algoritma pada sub-bab 4.6, dan pembahasan implementasi menghapus proses pada sub-bab 4.7. Aplikasi ini dikembangkan dalam bahasa pemrograman Java. Daftar *attribute* dan *method* dari semua *class* terdapat pada Lampiran 4.

Adapun cara menjalankan aplikasi ini adalah :

- Aplikasi ini dimulai dengan mengklik dua kali pada file TA.jar yang tersedia. Kemudian, jendela utama dari aplikasi ini akan muncul.

- Kemudian, pengguna harus memilih sumber masukan yang dibutuhkan. Sumber masukan dapat berasal dari berkas dan *database.*

    1. Bila pengguna ingin menggunakan berkas sebagai sumber masukan, maka :
        - Pengguna harus memilih format berkas dari daftar berbentuk *tree* yang terletak disebelah kiri menu utama.
        - Kemudian, operator masukan akan muncul pada *workspace.* Operator masukan diklik dua kali, kemudian jendela konfigurasi masukan akan muncul. Berkas masukan yang akan diproses dipilih dengan mengklik tombol *open file.* Setelah berkas didapatkan, semua data *instance* akan ditampilkan ke tabel daftar *instance.*

2. Bila pengguna ingin menggunakan *database* sebagai sumber masukan, maka:
    - Pengguna harus memilih label OpenDB dari daftar berbentuk *tree* yang terletak disebelah kiri menu utama. Kemudian, operator OpenDB akan muncul di *workspace.*
    - Operator OpenDB diklik dua kali, maka jendela konfigurasi *database* akan muncul. Namun, pengguna harus menghubungkan aplikasi terlebih dahulu dengan DBMS, dengan cara memasukkan URL, *username,* dan *password* ke dalam kolom yang tersedia, lalu tombol *Connect* diklik.
    - Setelah pengguna berhasil membuat koneksi, tahap selanjutnya adalah memasukkan *query* untuk mendapatkan *data instance*. Semua data *instance* akan ditampilkan pada tabel daftar *instance.*

- Masing – masing dari masukan mempunyai metadata yang dapat diakses melalui *tab* metadata.

- Pada *tab* metadata tersedia daftar grafik yang akan ditampilkan. Pengguna harus memilih bentuk grafik yang diinginkan, yaitu *bar chart, scatterplot*, dan *line chart.* Bila pengguna memilih bentuk *bar chart,* maka pengguna dapat langsung menekan tombol *graphic* untuk menampilkan data dalam bentuk *bar chart.* Bila pengguna memilih bentuk *scatterplot,* maka pengguna akan diminta memilih satu atribut yang akan ditampilkan, kemudian pengguna menekan tombol *graphic*. Sedangkan bila pengguna memilih bentuk *line chart,* maka pengguna akan diminta memilih dua atribut yang akan ditampilkan, kemudian pengguna menekan tombol *graphic*.

- Bila pengguna ingin melakukan proses *discretization* terhadap data masukan, maka pengguna memilih label *discretization* yang ada di menu sebelah kiri jendela utama. Kemudian, operator *discretization* akan muncul

di *workspace*. Dengan mengklik dua kali operator *discretization*, maka jendela konfigurasi akan muncul dalam bentuk *command prompt*. Kemudian, pengguna memasukkan nama berkas yang akan diproses dan memasukkan nama berkas keluaran.

- Pada menu sebelah kiri jendela utama, pengguna dapat memilih algoritma yang dibutuhkan. Kemudian, operator algoritma akan muncul di *workspace*. Pengguna harus mengklik dua kali operator tersebut, sehingga jendela konfigurasi algoritma akan muncul. Penentuan nilai parameter harus dilakukan, dan akhiri proses dengan mengklik tombol *Submit*.

- Pengguna dapat menekan tombol *Run Process* yang ada di menu utama untuk menjalankan algoritma. Kemudian, hasil keluaran dari algoritma akan muncul pada suatu jendela keluaran.

5.1 Implementasi Pemrosesan Awal Masukan

Seperti yang telah dijelaskan pada bab sebelumnya, berkas masukan dapat berasal dari format ARFF, XLS, CSV, dan UCI. Berikut dijabarkan kode pengolahan masukan yang berasal dari masing – masing berkas, yaitu pada *Pseudocode* 5.1, *Pseudocode* 5.2, *Pseudocode* 5.3, dan *Pseudocode* 5.4

1. ARFF

```
If berkas type is ARFF
      While the line is not empty
            If the line contains 'relation', add to relation
            Else if the line contains 'attribute', add to name
attribute and value attribute
            Else if the line contains 'data', add instance to
dataInstanceList
      Count the sum of each value in attribute
```

***Pseudocode* 5.1** *Method* untuk membaca berkas ARFF pada *class* AttributeInstanceReader

2. XLS

```
If berkas type is XLS
      Get the berkas name and add to relation
      Read first row, add each column to name attribute
```

```
      Read per column except the last and add value attribute to
each attribute
      Read last column and add as classes.
      Read the second line to last line, add as instances
```

***Pseudocode* 5.2** *Method* untuk membaca berkas XLS pada *class* AttributeInstanceReader

3. CSV

```
If berkas type is CSV
      Get the berkas name and add to relation
      Read first row, split by comma then add each value to name
attribute
      Read per column except the last and add value attribute to
each attribute
      Read last value in first line and add as classes.
         Read the second line to last line, add as instances.
```

***Pseudocode* 5.3** *Method* untuk membaca berkas CSV pada *class* AttributeInstanceReader

4. UCI

```
If berkas type is data
      Get the berkas name and add to relation
      Read berkas names
            Split line by ':'
            For t <- 0 to line splitted
                  Remove the whitespace
            Add to attribute list
      Read last value in line as classes

      Read berkas data
      Split line by ','
      Add as instances
```

***Pseudocode* 5.4** *Method* untuk membaca berkas .data pada *class* AttributeInstanceReader

Pengolahan berkas masukan dibuat agar data yang ada didalam berkas bisa digunakan oleh algoritma.

Selain berasal dari berkas, sumber masukan dapat berasal dari *database*. Untuk membuat koneksi ke *database*, penulis menggunakan *library* yang telah disediakan oleh Netbeans. Berikut merupakan *pseudocode* untuk melakukan koneksi ke *database* yang terdapat pada *Pseudocode* 5.5.

```
Make object Connection
```

```
Get URL string from textfield URL
Get Username string from textfield Username
Get Password string from textfield Password
Make a connection to database using driver manager

If connection success
      Assign to object Connection
      Set text in text area <- "connection successfull"
Else
      Set text in text area <- "connection failed"
```

***Pseudocode* 5.5** *Method* untuk membuka koneksi ke *database* pada *class* OpenDBFrame

Setelah aplikasi terhubung ke *database*, data dapat diambil dari *database* menggunakan *query*. Berikut *pseudocode* untuk mendapatkan data dengan menggunakan *query* yang terdapat pada *Pseudocode* 5.6.

```
If connection success
      Get query string from textfield query
      Create object statement
      Execute query using object statement
      If query true
            Set text in text area <- "query true"
      Else
            Set text in text area <- "query false"
```

***Pseudocode* 5.6** *Method* untuk mendapatkan *data instance* menggunakan *query* pada *class* OpenDBFrame

Setelah mendapatkan *data instance*, maka langkah selanjutnya memasukkan semua *instance* tersebut kedalam tabel pada jendela masukan. Berikut *pseudocode* dari tahap memasukkan data ke dalam tabel pada jendela masukan yang terdapat pada *Pseudocode* 5.7.

```
Create model of table instance
For i <- 0 to length attributeArray
      Add name attribute to column modelTableInstance

For i <- 0 to length dataInstanceList
      Add instance list to row modelTableInstance

Set model tableInstance to modelTableInstance
```

***Pseudocode* 5.7** *Method* untuk menampilkan *data instance* ke tabel pada *class* OpenDBFrame

5.2 Implementasi Pemilihan Data

Data masukan telah ditampilkan pada tabel yang ada di jendela masukan. Bila pengguna ingin menggunakan seluruh data yang ada, pengguna diharuskan mengklik tombol *select all*. Tetapi, pengguna bisa memilih data tertentu yang akan digunakan untuk proses selanjutnya. Langkah yang harus dilakukan yaitu menyorot beberapa baris yang berisi data pilihan pengguna. Setelah data yang diinginkan telah disorot, pengguna mengklik tombol *select value*. Data yang terpilih tersebut akan ditampilkan pada jendela data *instance* yang terpilih. Berikut *pseudocode* dari langkah pemilihan data yang terdapat pada *Pseudocode* 5.8.

```
Create chosenValue[]
Create List of array Chosen Instance
selectedRow[] <- get array of selected rows
sumColumn <- total table column

for z <- 0 to selectedRow[]
      chosenValue <- new String[column]
      for j <- 0 to sumColumn
            chosenValue[j] <- get value from table that selected
      add chosenValue[] to ChosenInstance
```

***Pseudocode* 5.8** *Method* untuk memilih *data instance* pada *class* FrameInput

5.3 Implementasi Metadata

Metadata baru dapat dibuat jika semua data yang diperlukan telah tersedia. Tiap elemen metadata dibuat dengan cara yang berbeda. Berikut penjelasan implementasi dan *pseudocode* dari tiap elemen metadata.

- Daftar atribut dan nilainya

  Untuk data yang diambil dari berkas ARFF, daftar atribut beserta nilainya disediakan oleh berkas tersebut. Sedangkan untuk berkas CSV, .data dan XLS, daftar atribut tidak disediakan secara eksplisit. Sehingga, semua nilai pada tiap kolom dimasukkan kedalam struktur data *TreeSet*. Struktur data ini hanya menerima nilai yang unik, bila ada nilai yang sama maka *TreeSet* akan membuang nilai secara otomatis. Diakhir proses pemasukan nilai ke *TreeSet*, akan didapatkan nilai yang unik dimana nilai tersebut adalah nilai pada tiap

atribut. Berikut *pseudocode* pengambilan nilai atribut untuk berkas CSV / XLS yang terdapat pada *Pseudocode* 5.9.

```
Create Objet AttributeList
Create object TreeSet
Loop through each column
   Loop through each value in the column
      Add to TreeSet
      Convert treeset to array
      Add array to attributelist
      Clear the treeset
```

***Pseudocode* 5.9** *Method* untuk mengambil nilai atribut dari berkas CSV dan XLS pada *class* AttributeInstanceReader

Untuk menampilkan data ke tabel atribut, caranya sama untuk semua berkas. Berikut *pseudocode* untuk menampilkan nilai atribut yang terdapat pada *Pseudocode* 5.10.

```
Count <- 0
Loop through attribute
   attributeList[] <- get array of attribute list
   for i <- 0 to length attributeList
         if attributeList[i] equals "numeric"
               add row of table to "numeric"
         else
               add row of table to list of attribute
   Increment count
```

***Pseudocode* 5.10** *Method* untuk menampilkan nilai atribut ke tabel pada *class* FrameInput

- Jumlah nilai tiap atribut

  Jumlah nilai tiap atribut akan ditampilkan di kolom nilai atribut. Format penulisannya yaitu NilaiAtribut (jumlahNilaiAtribut). Berikut *pseudocode* untuk menampilkan jumlah nilai atribut yang terdapat pada *Pseudocode* 5.11.

```
Count <- 0
Loop through attribute
   attributeCount[] <- get array of attribute count
   attributeList[] <- get array of attribute list
   for i <- 0 to length attributeList
         if attributeList[i] equals "numeric"
               add row of table to "numeric"
         else
```

```
                add row of table attribute count
   Increment count
```

***Pseudocode* 5.11** *Method* untuk menampilkan jumlah nilai atribut ke tabel pada c*lass* AttributeInstanceReader

- Jumlah *instance* yang tersedia

  Jumlah instance yang tersedia didapat dari *method size()* yang ada di bahasa pemrograman Java. Selanjutnya, nilai keluaran dari *method* tersebut ditampilkan di *tab* metadata pada jendela masukan.

- Nama relasi

  Nama relasi dari data didapat dari nama berkas yang sedang diproses. Selanjutnya, nama relasi tersebut ditampilkan di *tab* metadata pada jendela utama.

- Tipe data

  Tiap atribut mempunyai tipe data yang berbeda. Tipe data dari atribut akan muncul ketika baris dari atribut pada tabel yang menampilkan daftar atribut diklik. Tipe data muncul pada panel yang berada dibawah tab metadata pada jendela masukan. Berikut kode dari proses menampilkan tipe data tiap atribut yang terdapat pada *Pseudocode* 5.12.

```
Pos <- position of selected row
Get attribute list
If attributeList[pos] equals "numeric"
   Set label type of data to "numeric"
Else
   Set label type of data to "String"
```

***Pseudocode* 5.12** *Method* untuk menampilkan tipe data atribut pada *class* FrameInput

5.4 Implementasi Menampilkan Grafik

Grafik ditampilkan menggunakan *library* JFreeChart yang dikembangkan oleh David Gilbert dan Thomas Morgner. Berikut *pseudocode* dari impelementasi *library* untuk menampilkan grafik dalam *bar chart* yang terdapat pada *Pseudocode* 5.13.

```
Create object JFreeChart
Create object ChartFrame
```

```
Create object XYSeriesCollection
If graphic equals "bar chart"
      attributeNameList<> <- get attribute name list
      attributeList<> <- get attribute value list
      countAttribute<> <- get sum of each attribute value

      while attributeNameList<> not null
            aString <- next value of attributeNameList
            aValue[] <- next value of countAttribute
            aList[] <- next value of attributeList

            for i <- 0 to aList[]
                  set value of dataset

      chart <- create bar chart
      chartFrame <- chart
      show the chartFrame
```

***Pseudocode* 5.13** *Method* untuk menampilkan *Bar Chart* pada *class* Chart

Bentuk grafik selanjutnya adalah *scatterplot. Scatterplot* dapat menampilkan data dengan tipe *integer*. Pengguna memilih data dari satu atribut untuk ditampilkan dalam bentuk *scatterplot*. Berikut *pseudocode* untuk menampilkan data dalam bentuk *scatterplot* yang terdapat pada *Pseudocode* 5.14.

```
Create object JFreeChart
Create object ChartFrame
Create object XYSeriesCollection
If graphic equals "scatter plot"
   Create object XYSeries
   Loop until size of data intstance
       Get data from attribute1
       Add data to series
   Add series to XYSeriesCollection

   chart <- create XYLineChart
   chartFrame <- chart
   show the chartFrame
```

***Pseudocode* 5.14** *Method* untuk menampilkan *Scatterplot* pada *class* Chart

Bentuk grafik selanjutnya adalah *line chart. Line chart* dapat menampilkan data dengan tipe *integer*. Pengguna memilih data dari dua atribut untuk ditampilkan

bentuk *line chart*. Berikut *pseudocode* untuk menampilkan data dalam bentuk *line chart* yang terdapat pada *Pseudocode* 5.15.

```
Create object JFreeChart
Create object ChartFrame
Create object XYSeriesCollection
If graphic equals "line chart"
   Create object XYSeries
   Loop until size of data intstance
       Get data from attribute1
       Get data from attribute2
       Add data to series

   Add series to XYSeriesCollection

   chart <- create XYLineChart
   chartFrame <- chart
   show the chartFrame
```

***Pseudocode* 5.15** *Method* untuk menampilkan *line chart* pada *class* Chart

5.5 Implementasi Menggunakan *Discretization*

Dalam implementasi ini, penulis tidak membuat metode *discretization* sendiri, melainkan penulis menggunakan aplikasi *discretization* yang telah dikembangkan oleh Ronny Kohavi. Untuk menggunakan aplikasi ini, dibutuhkan beberapa berkas yaitu berkas dengan format .all, .data, .names, dan .test. Bila pengguna menggunakan format ARFF, CSV, .data, atau XLS sebagai masukan, maka aplikasi akan membuat keempat berkas tersebut. Berikut *pseudocode* dari proses pembuatan keempat berkas yang diperlukan dalam proses *discretization* yang terdapat pada *Pseudocode* 5.16.

```
Create new berkas relation.all
Create new berkas relation.data
Create new berkas relation.test
Create new berkas relation.names
Create string temp

//write instance
While dataInstanceList not null
      dataInstance[] <- element dataInstanceList
      for i <- 0 to dataInstance[]
            temp <- temp + dataInstance[] + ‘,’
```

```
      write to berkas <- temp

//write attribute
Create string temp2

While listClass not null
      temp2 <- temp2 + element listClass

while attributeNameList not null
      if attributeNameList.size – 1
            break
      else
            if element attributeNameList equals 'numeric'
                  temp2 <- temp2 + ': continous'
            else
                  attributeList[] <- get attribute list
                  string name <- element attributeNameList
                  temp2 <- temp2 + name + ': "

                  for j <- 0 to attributeList[]
                        temp2 <- temp2 + attributeList[j] + ','

      write to berkas <- temp2
```

***Pseudocode* 5.16** *Method* untuk menggunakan aplikasi *discretization* pada *class* FrameChosenInstance

Langkah selanjutnya adalah menjalankan aplikasi *discretization* dengan mengklik dua kali operator *discretization.* Pengguna akan diminta untuk mengisikan nama berkas yang akan diproses serta nama berkas sebagai hasil keluaran. Aplikasi ini akan menghasilkan dua berkas keluaran, yaitu berkas dengan format .data yang berisi *data instance* yang baru dan berkas dengan format .names yang berisi informasi mengenai atribut dan label.

Penulis melakukan beberapa perubahan terhadap berkas yang dihasilkan oleh proses *discretization* agar algoritma bisa menggunakan berkas tersebut. Perubahan dilakukan dibagian format penulisan berkas .names dan .data seperti yang dijelaskan pada bab 3.

5.6 Implementasi Memilih dan Mengkonfigurasi Algoritma

Tiap algoritma mempunyai konfigurasi yang berbeda, tergantung kepada parameter yang diminta oleh algoritma tersebut. Algoritma bisa dipilih dari *tree* yang ada disebelah kiri jendela utama. Kemudian, operator akan muncul pada *workspace* yang mewakili keberadaan algoritma tersebut. Berikut *pseudocode* dari pemilihan algoritma yang terdapat pada *Pseudocode* 5.17.

```
Path[] <- get tree path
For i <- 0 to path length
      If path[i] equals algorithm
            Set operator location and size
            Drop operator in workspace
            Add mouse listener to operator
```

***Pseudocode* 5.17** *Method* untuk memilih algoritma pada *class* MainFrame

Setelah memilih algoritma dan operator telah muncul di *workspace*, pengguna bisa mengkonfigurasi algoritma tersebut sebelum menjalankannya. Konfigurasi yang dilakukan pada jendela konfigurasi algoritma yang dibuat ketika aplikasi berjalan. Jumlah dan nama parameter ditentukan oleh pengembang algoritma. Berikut *pseudocode* dari tahap konfigurasi algoritma yang terdapat pada *Pseudocode* 5.18.

```
If the operator clicked twice
      Button run process enabled
      Get number of parameter and name of parameter
      Create frame algorithm contain number of parameter
      Set the parameter
      Set parameter in MainFrame
      Dispose the frame algorithma window
```

***Pseudocode* 5.18** *Method* untuk melakukan konfigurasi terhadap algoritma pada *class* MainFrame

5.7 Implementasi Menjalankan Algoritma

Algoritma dapat dijalankan setelah tahap konfigurasi selesai, dengan mengklik tombol *Run Process* yang ada di jendela utama. Keluaran dari proses tersebut akan muncul pada jendela keluaran, yaitu pada *text area.* Berikut *pseudocode* dari tahap menjalankan algoritma yang terdapat pada *Pseudocode* 5.19.

```
Check what algorithm being chosen
Get the parameter that algorithm needed
Send the parameter to the algorithm
If success
      Create Output Frame
      Set text in text area to the output of algorithm
```

***Pseudocode* 5.19** *Method* untuk menjalankan algoritma pada *class* MainFrame

5.8 Implementasi Menghapus Operator

Pengguna bisa menghapus operator yang muncul pada *workspace* untuk menggantinya dengan operator yang lain. Langkah ini dilakukan jika pengguna melakukan kesalahan dalam memilih proses yang akan dilakukannya. Berikut *pseudocode* dari tahap menghapus *operator* yang terdapat pada *Pseudocode* 5.20.

```
If the operator right clicked
      Create Popup Menu
      Create Menu Item, assign the text to "delete"
      If menu item clicked
            Remove the operator
```

***Pseudocode* 5.20** *Method* untuk menghapus operator yang ada di *workspace* pada *class* MainFrame

BAB 6
PENGUJIAN APLIKASI

Pada bab ini akan dibahas mengenai pengujian hasil implementasi yang telah dilakukan penulis. Tujuan dari pengujian ini adalah untuk melihat harapan keluaran dari aplikasi untuk tiap *use case* yang diujikan.

6.1 Pengujian Mendapatkan Data Masukan dan Melihat Metadata

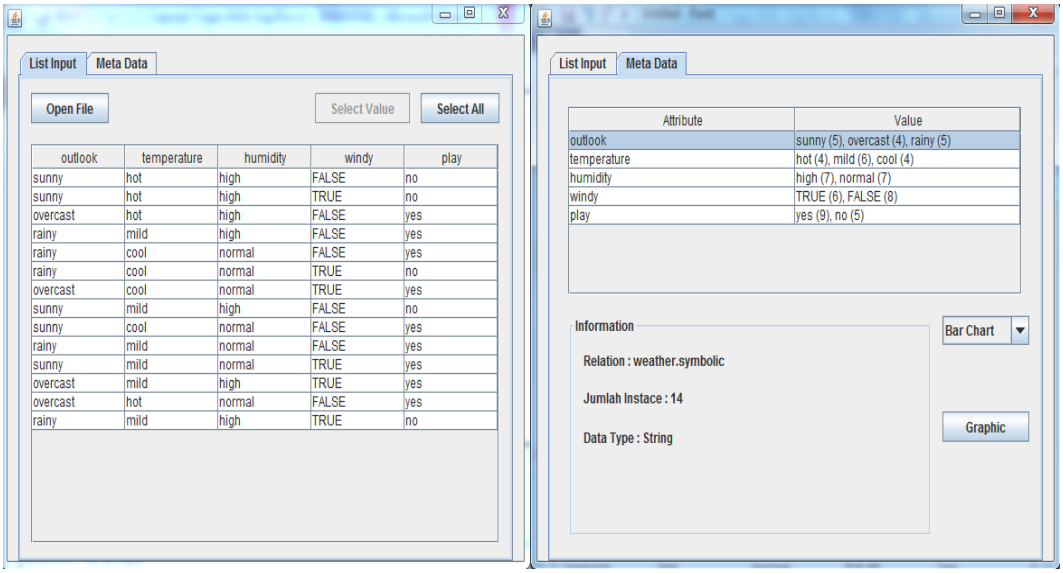
Aplikasi ini mendapatkan data masukan dari dua sumber, yaitu berkas dan *database.* Pengujian dibawah ini dilakukan terhadap kedua sumber tersebut.

6.1.1 Masukan yang Bersumber dari Berkas

Aplikasi ini dapat menerima masukan dari berkas dengan format ARFF, XLS, CSV, dan UCI. Berikut ini adalah pengujian terhadap data masukan untuk tiap format berkas diatas beserta melihat informasi metadata.

- ARFF

  Isi dari berkas ARFF yang diujikan terdapat pada Lampiran 1. Hasil yang diharapkan yaitu semua *instance* yang ada akan ditampilkan pada tabel, sedangkan informasi atribut akan ditampilkan pada *tab* metadata. Pada Gambar 6.1 adalah hasil sebenarnya dari keluaran aplikasi ketika mendapatkan data masukan dari berkas ARFF.

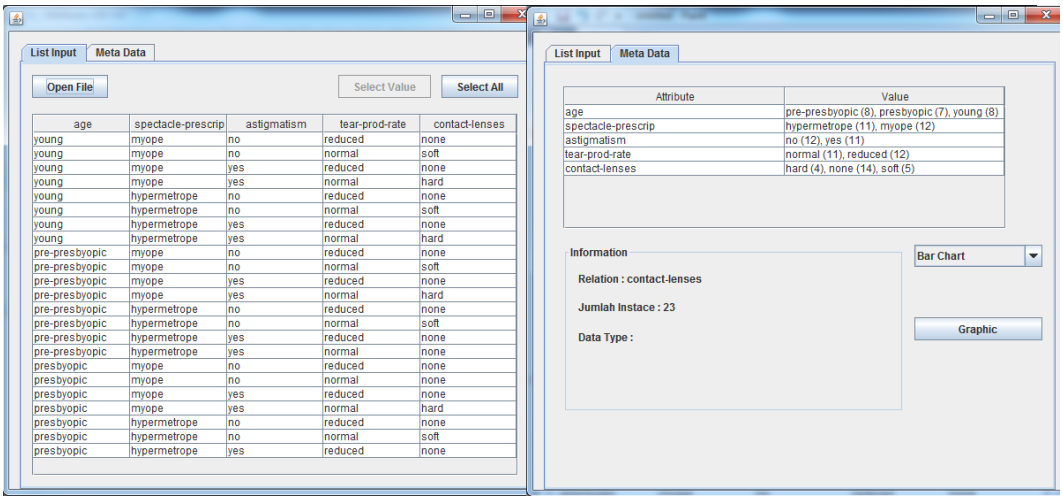


Gambar 6.1 Hasil pengujian mendapatkan data masukan dari berkas ARFF

Terlihat bahwa hasil keluaran dari aplikasi sesuai dengan yang diharapkan. Semua *instance* ditampilkan pada tabel *instance*, sedangkan informasi mengenai atribut ditampilkan pada tab metadata.

- CSV

Isi dari berkas CSV yang diujikan terdapat pada Lampiran 4. Hasil yang diharapkan yaitu semua *instance* yang ada akan ditampilkan pada tabel, sedangkan informasi atribut akan ditampilkan pada *tab* metadata. Pada Gambar 6.2 adalah hasil sebenarnya dari keluaran aplikasi ketika mendapatkan data masukan dari berkas CSV.

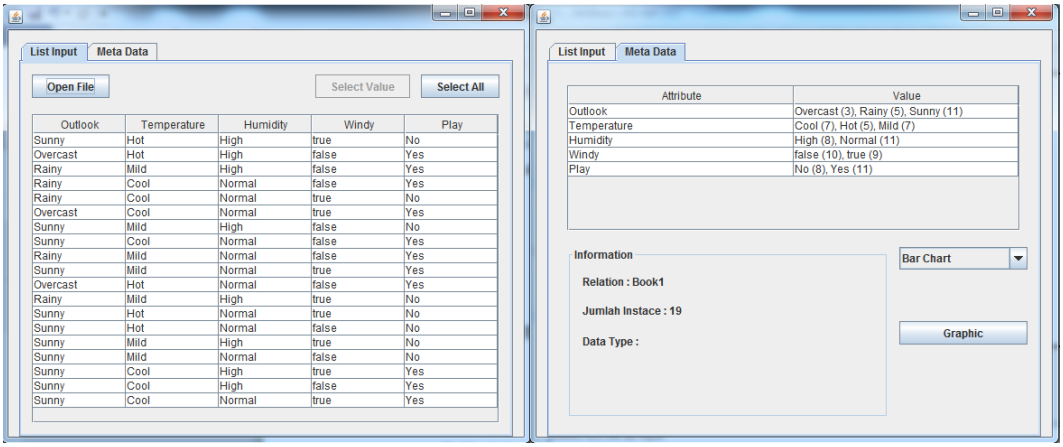


Gambar 6.2 Hasil pengujian mendapatkan data masukan dari berkas CSV

Terlihat bahwa hasil keluaran dari aplikasi sesuai dengan yang diharapkan. Semua *instance* ditampilkan pada tabel *instance*, sedangkan informasi mengenai atribut ditampilkan pada tab metadata.

- XLS

Isi dari berkas XLS yang diujikan terdapat pada Lampiran 5. Hasil yang diharapkan yaitu semua *instance* yang ada akan ditampilkan pada tabel, sedangkan informasi atribut akan ditampilkan pada *tab* metadata. Pada Gambar 6.3 adalah hasil sebenarnya dari keluaran aplikasi ketika mendapatkan data masukan dari berkas XLS.

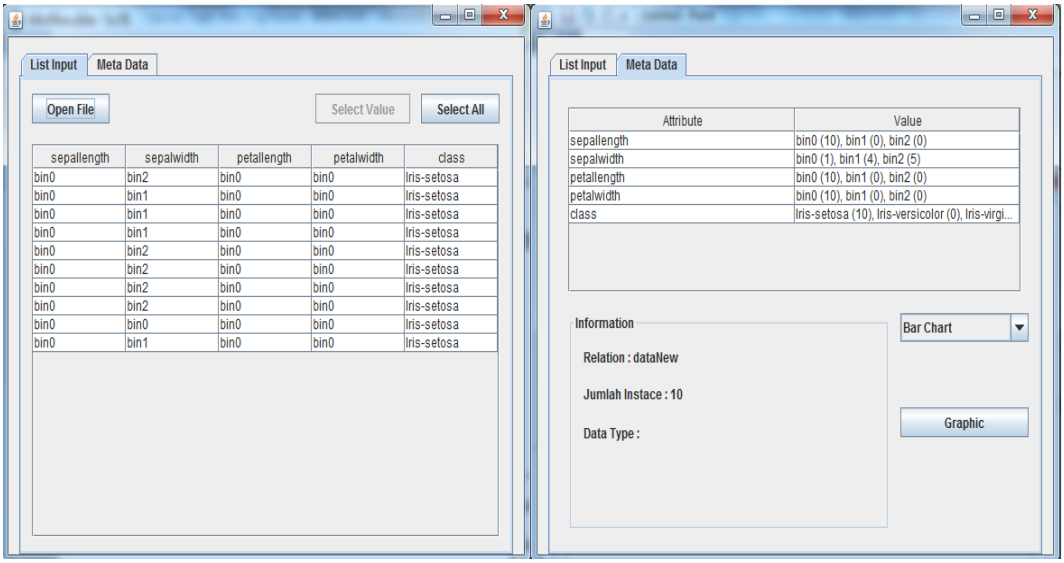


Gambar 6.3 Hasil pengujian mendapatkan data masukan dari berkas XLS

Terlihat bahwa hasil keluaran dari aplikasi sesuai dengan yang diharapkan. Semua *instance* ditampilkan pada tabel *instance*, sedangkan informasi mengenai atribut ditampilkan pada tab metadata.

- UCI

Isi dari berkas .data dan .names yang diujikan terdapat pada Lampiran 6. Hasil yang diharapkan yaitu semua *instance* yang ada akan ditampilkan pada tabel, sedangkan informasi atribut akan ditampilkan pada *tab* metadata. Pada Gambar 6.4 adalah hasil sebenarnya dari keluaran aplikasi ketika mendapatkan data masukan dari berkas .data dan .names.



Gambar 6.4 Hasil pengujian mendapatkan data masukan dari berkas XLS

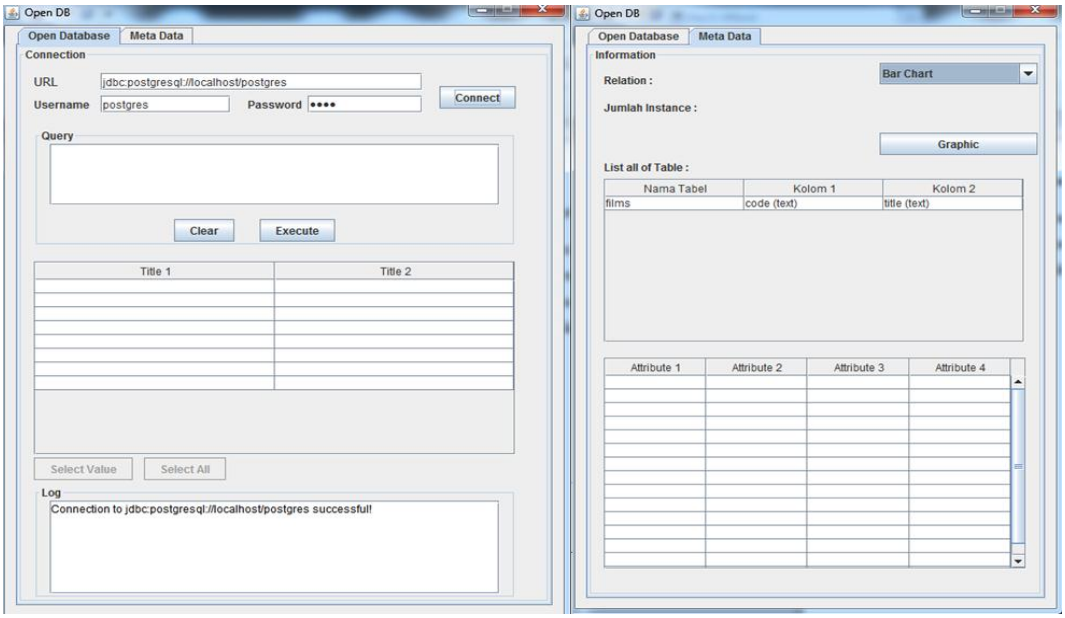
Terlihat bahwa hasil keluaran dari aplikasi sesuai dengan yang diharapkan. Semua *instance* ditampilkan pada tabel *instance*, sedangkan informasi mengenai atribut ditampilkan pada tab metadata.

6.1.2 Masukan yang Bersumber dari *Database*

Data masukan juga bisa bersumber dari *database*. *Database* yang bisa digunakan adalah PostgreSql dan Mysql. Pengujian dilakukan terhadap kedua sumber diatas. Tahap pertama yang dilakukan untuk mendapatkan data masukan dari *database* adalah membuka koneksi ke *database* tersebut. Sebelum melakukan proses ini, diasumsikan pengguna telah memiliki minimal satu tabel di *database* yang akan diakses. Selanjutnya, tahap kedua yaitu mendapatkan data dengan memasukkan *query* kedalam area teks yang tersedia.

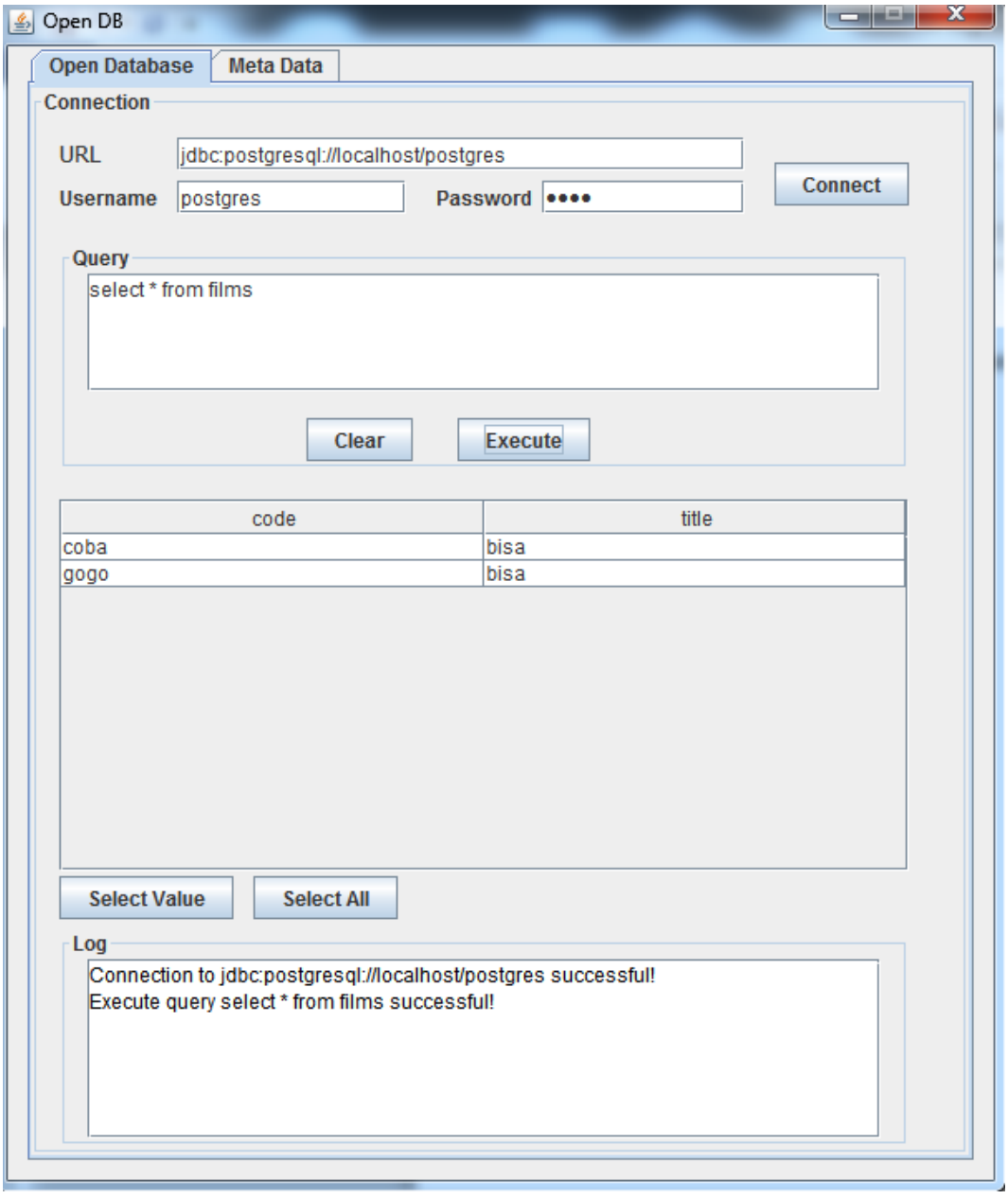
- PostgreSql

  Untuk membuka koneksi digunakan URL "jdbc:postgresql://localhost/postgres", *username* postgres dan *password* yogi. Hasil yang diharapkan yaitu aplikasi telah terhubung dengan *database*, dengan dibuktikan munculnya teks "*Connection Successful*" pada jendela konfigurasi *database*. Selanjutnya, metadata dari *database* akan muncul pada tab metadata. Pada Gambar 6.5 adalah hasil sebenarnya dari keluaran aplikasi.



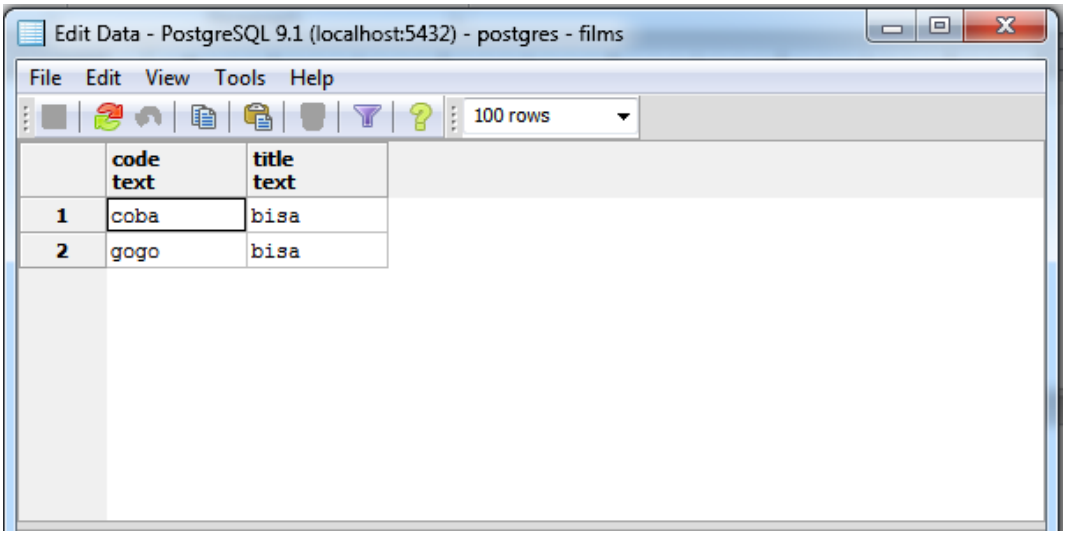
Gambar 6.5 Hasil pengujian aplikasi telah terhubung dengan PostgreSql

Terlihat aplikasi telah terhubung dan informasi metadata dari *database* telah muncul pada tab metadata. Langkah selanjutnya adalah mendapatkan data dengan memasukkan *query*. Pada jendela metadata, terdapat informasi mengenai nama tabel berikut kolomnya. *Query* yang dimasukkan adalah "Select * from films". Hasil yang diharapkan yaitu semua baris yang ada di tabel "films" akan muncul pada tabel. Pada Gambar 6.6 merupakan hasil sebenarnya dari keluaran aplikasi.



Gambar 6.6 Hasil penguji aplikasi setelah memasukkan query

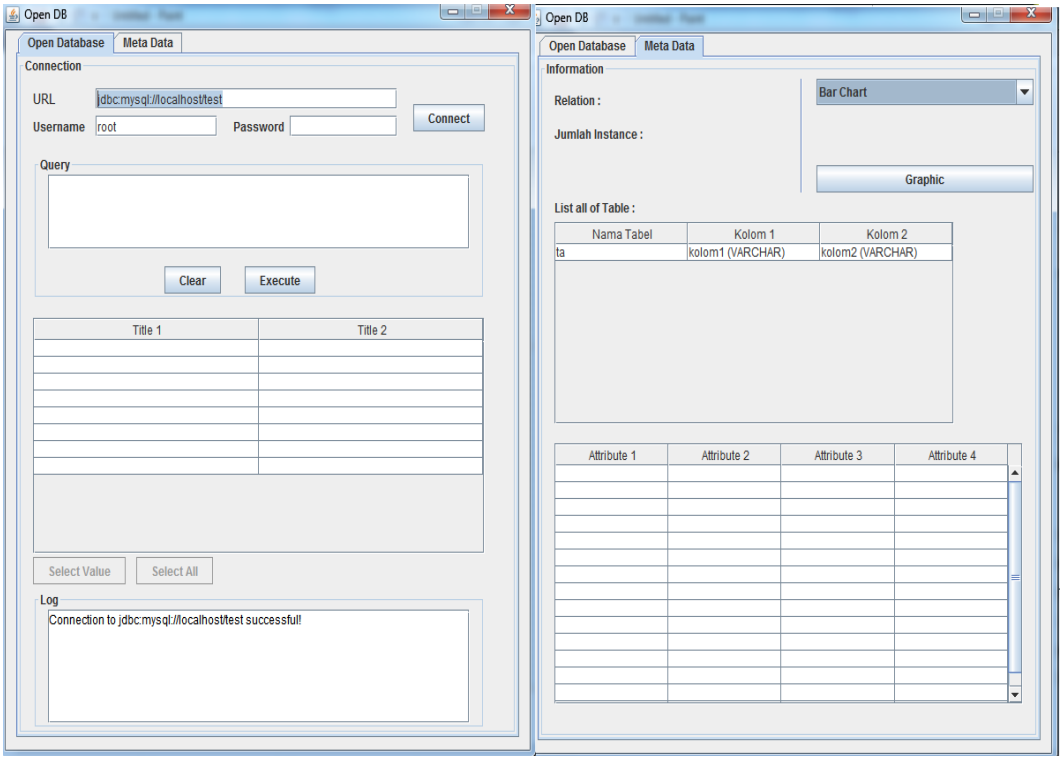
Data yang ditampilkan pada tabel dan informasi metadata dari tabel films sesuai dengan yang ada di tabel films. Pada Gambar 6.7 merupakan tampilan dari tabel films.



Gambar 6.7 Tampilan tabel films pada PostgreSql

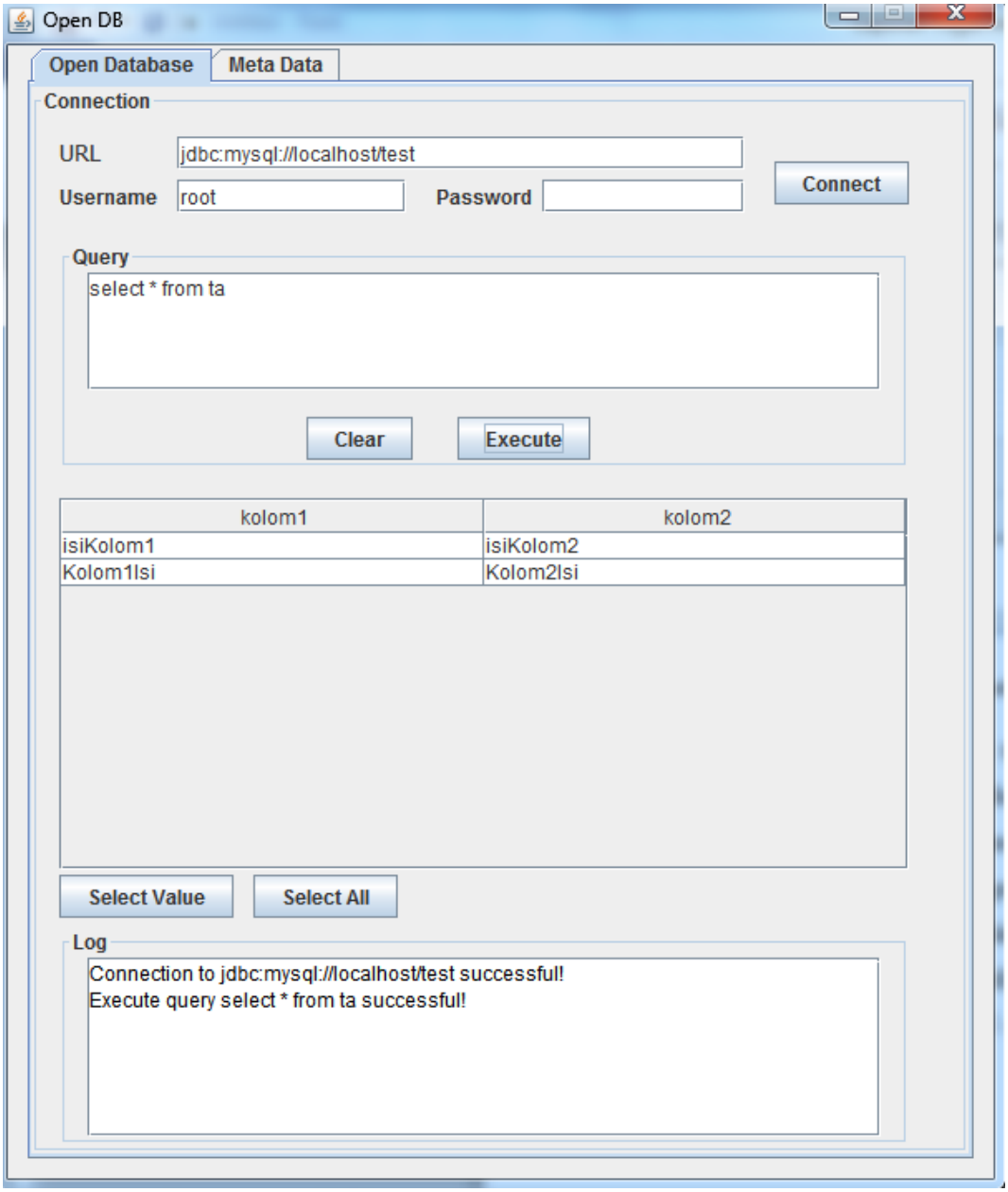
- MySql

  Untuk membuka koneksi digunakan URL "jdbc:mysql://localhost/test", *username* root, sedangkan *password* dikosongkan. Hasil yang diharapkan yaitu aplikasi telah terhubung dengan *database*, dengan dibuktikan munculnya teks "*Connection Successful*" pada jendela konfigurasi *database* dan metadata dari *database* akan muncul pada tab metadata. Pada Gambar 6.8 adalah hasil sebenarnya dari keluaran aplikasi.



Gambar **6.8** Hasil pengujian aplikasi telah terhubung dengan MySql

Terlihat aplikasi telah terhubung dan informasi metadata dari *database* telah muncul pada tab metadata. Langkah selanjutnya adalah mendapatkan data dengan memasukkan *query*. Pada jendela metadata, terdapat informasi mengenai nama tabel berikut kolomnya. *Query* yang dimasukkan adalah "Select * from ta". Hasil yang diharapkan yaitu semua baris yang ada di tabel "ta" akan muncul pada tabel. Pada Gambar 6.9 merupakan hasil sebenarnya dari keluaran aplikasi setelah memasukkan *query*.



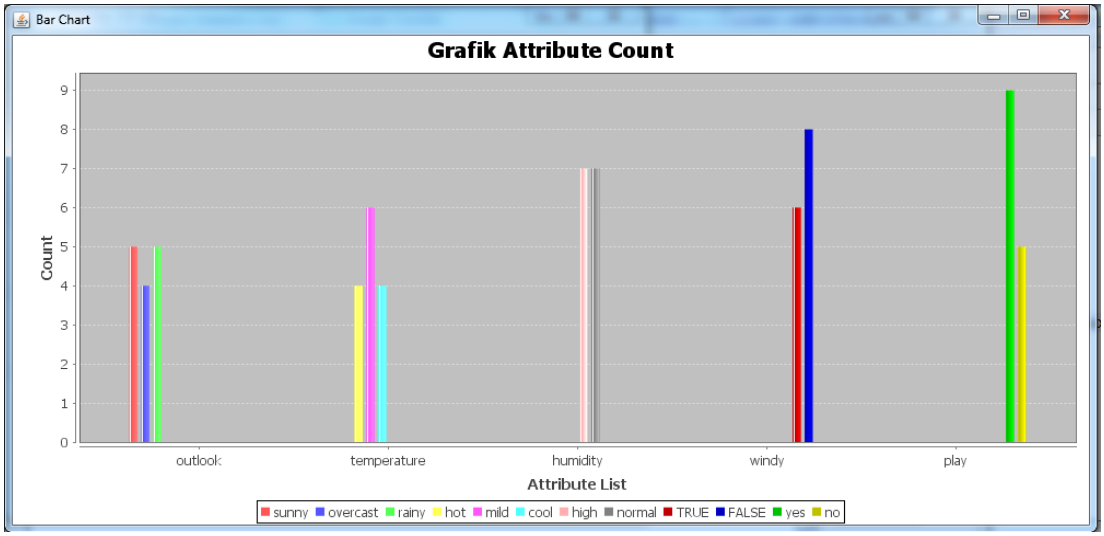
Gambar 6.9 Hasil pengujian aplikasi telah memasukkan query

Data yang ditampilkan pada tabel dan informasi metadata dari tabel ta sesuai dengan yang ada di tabel ta. Pada Gambar 6.10 merupakan tampilan dari tabel ta.

Gambar 6.10 Tampilan tabel ta pada MySql

6.2 Pengujian Melihat Grafik

Terdapat tiga pilihan untuk memvisualisasikan data masukan, yaitu *bar chart, scatterplot,* dan *linear chart. Bar chart* digunakan untuk menampilkan data dengan tipe nominal atau *discrete.* Data pengujian untuk menampilkan *bar chart* menggunakan berkas pada Lampiran 1. Hasil yang diharapkan yaitu frekuensi tiap nilai atribut akan ditampilkan dalam bentuk *bar*. Pada Gambar 6.11 merupakan hasil sebenarnya dari *bar chart* yang ditampilkan pada aplikasi.



Gambar 6.11 Hasil Pengujian Menampilan *Bar Chart*

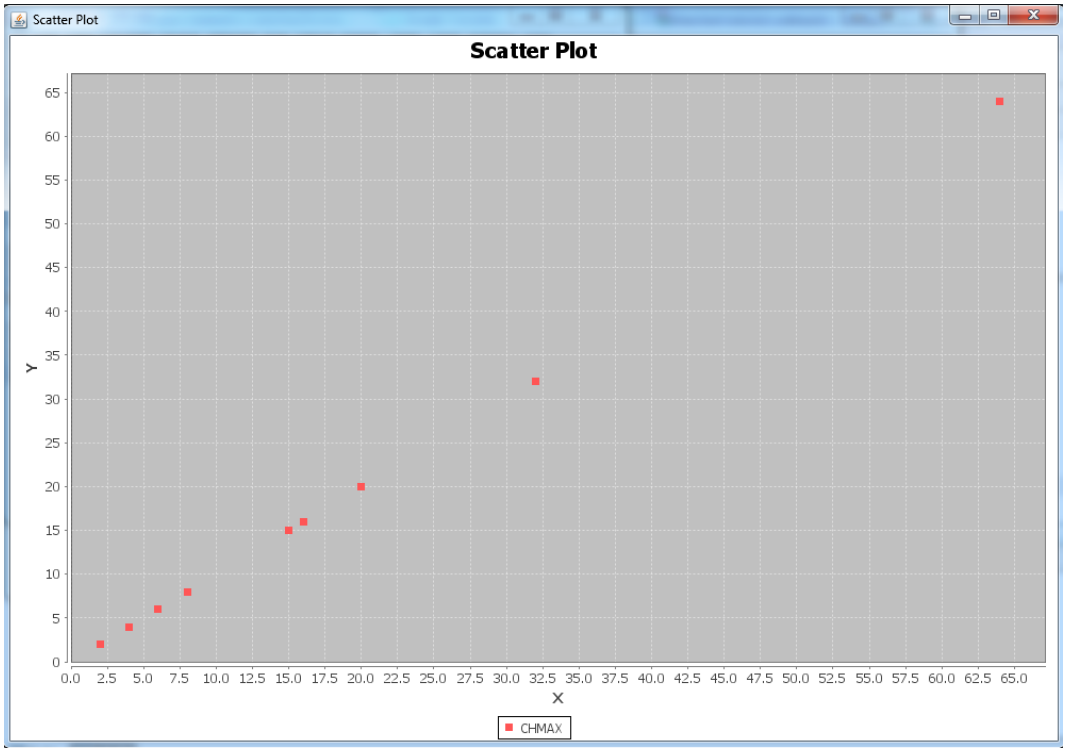
Frekuensi yang ditampilkan pada *bar chart* sesuai dengan jumlah dari nilai tiap atribut. Pada Gambar 6.12 adalah tampilan jumlah dari nilai tiap atribut pada aplikasi.

Attribute	Value
outlook	sunny (5), overcast (4), rainy (5)
temperature	hot (4), mild (6), cool (4)
humidity	high (7), normal (7)
windy	TRUE (6), FALSE (8)
play	yes (9), no (5)

Gambar 6.12 Tampilan jumlah dari nilai tiap atribut pada aplikasi

Untuk tampilan visualisasi berikutnya yaitu *line chart* dan *scatterplot,* lebih cocok digunakan untuk data dengan tipe *numeric*. Untuk melakukan pengujian menampilkan bentuk *line chart* dan *scatterplot,* penulis menggunakan data yang terdapat pada Lampiran 2. Data yang ditampilkan berasal dari atribut CHMAX.

Hasil yang diharapkan yaitu koodinat tiap data atribut akan ditampilkan dalam bentuk *plot*. Pada Gambar 6.13 merupakan hasil sebenarnya dari *scatterplot* yang ditampilkan pada aplikasi.



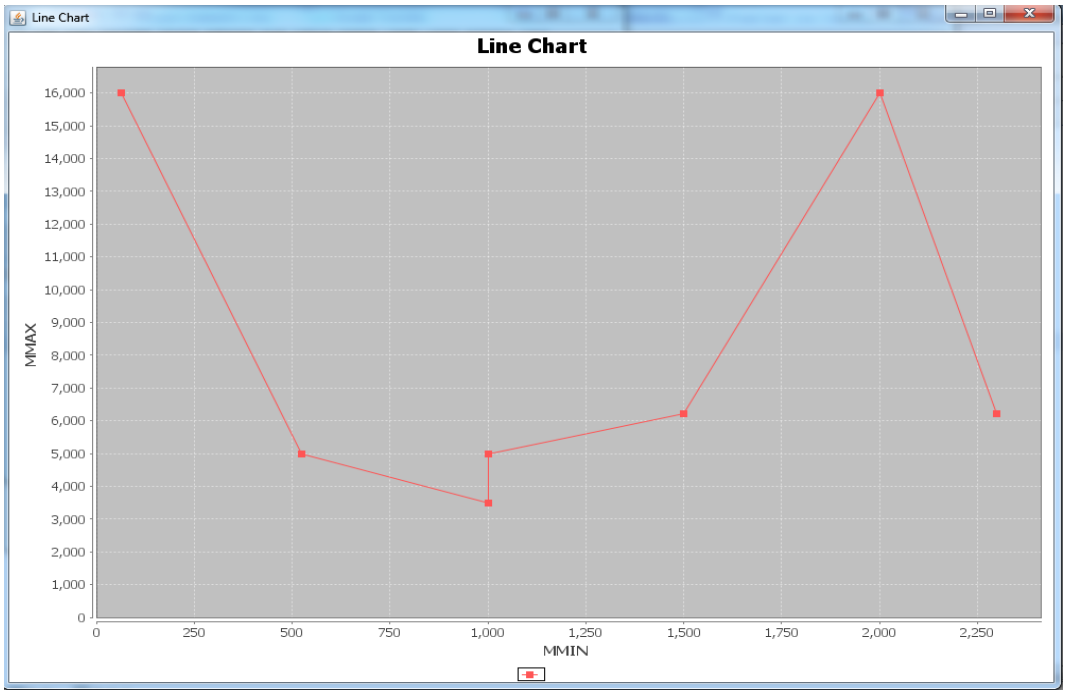
Gambar 6.13 Hasil Pengujian Menampilan *Scatterplot* pada aplikasi

Titik yang ditampilkan pada *scatterplot* sesuai dengan data tiap atribut. Pada Gambar 6.14 adalah tampilan data atribut CHMAX pada aplikasi.

chmax
2
6
8
8
4
32
15
32
16
64
32
20
64
64

Gambar 6.14 Tampilan data chmax dari aplikasi

Bentuk visualisasi selanjutnya adalah *line chart.* Pada pengujian ini penulis menggunakan data yang terdapat pada Lampiran 2 untuk menampilkan *line chart*, dan memilih dua atribut yaitu MMIN sebagai X dan MMAX sebagai Y. Hasil yang diharapkan yaitu koodinat tiap data atribut akan ditampilkan dalam bentuk *plot* dan dihubungkan oleh garis. Pada Gambar 6.15 merupakan hasil sebenarnya dari *line chart* yang ditampilkan pada aplikasi.



Gambar 6.15 Hasil Pengujian Menampilan *Line Chart* pada aplikasi

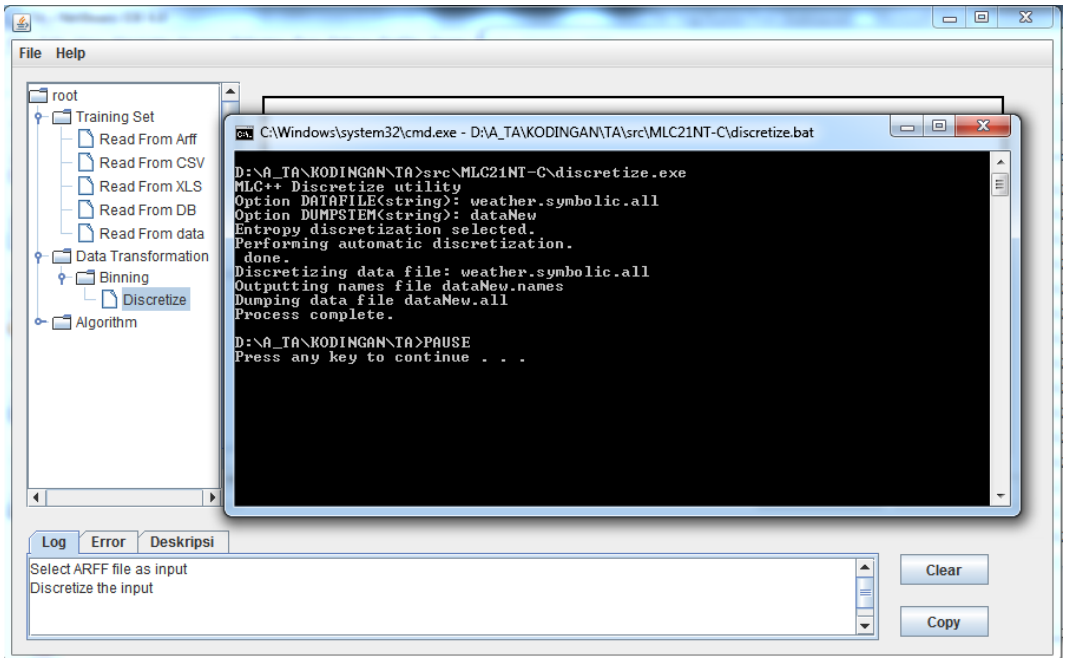
Titik yang ditampilkan pada *line chart* sesuai dengan data tiap atribut. Pada Gambar 6.16 adalah tampilan data atribut MMIN dan MMAX pada aplikasi.

mmin	mmax
1000	3000
512	3500
2000	8000
4000	16000
64	64
512	16000
524	2000
512	5000
1000	2000
5000	5000
1500	6300
3100	6200
2300	6200
3100	6200

Gambar 6.16 Tampilan data dari atribut MMIN dan MMAX

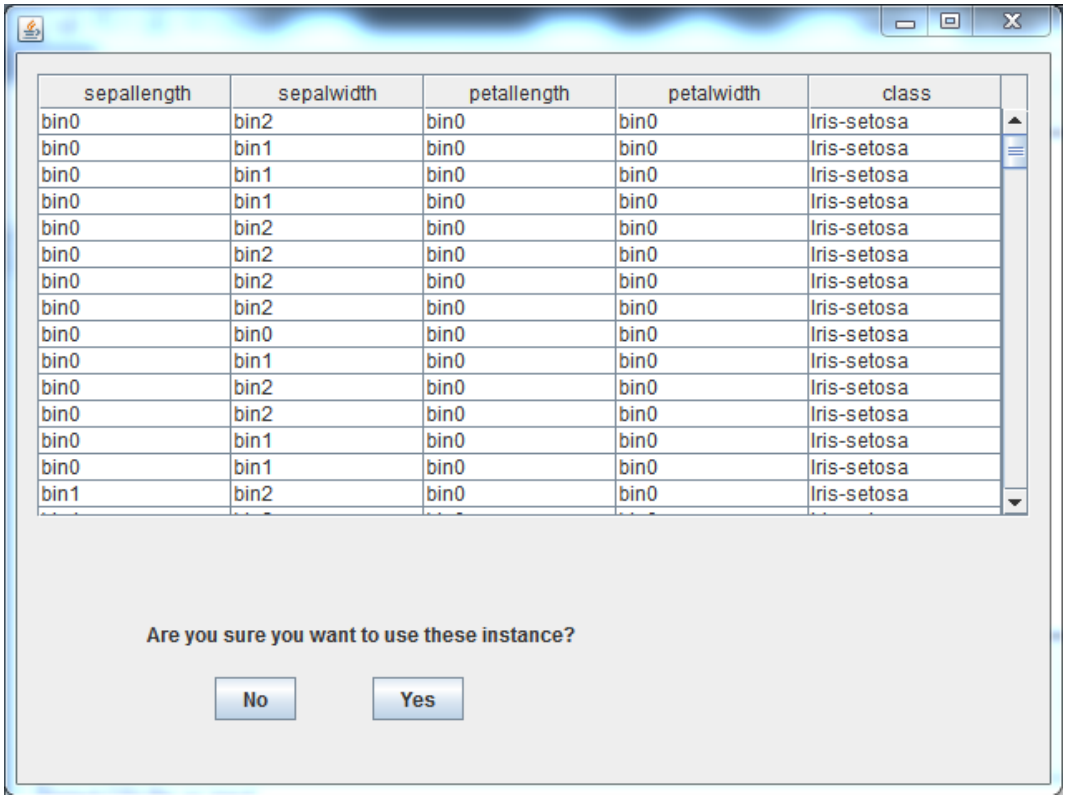
6.3 Pengujian Penggunaan *Discretization* Terhadap Data Masukan

Pengujian penggunaan teknik *discretization* dilakukan terhadap data yang terdapat pada Lampiran 3. Ada beberapa langkah yang harus dipatuhi, yaitu berkas masukan yang akan diproses tidak dipilih dari direktori, tetapi diambil dari data masukan yang telah diproses sebelumnya. Pengguna harus mengingat nama berkas masukan tersebut karena aplikasi *discretization* akan meminta nama berkas yang akan diproses. Selanjutnya, pengguna akan diminta untuk memasukkan nama berkas keluaran dari proses *discretization.* Hasil yang diharapkan adalah aplikasi dapat mengubah data dari tipe *integer* menjadi tipe *discrete* dan ditampilkan pada sebuah tabel. Pada Gambar 6.17 merupakan tampilan tahap awal dari proses *discretization* yaitu memasukkan nama berkas masukan dan keluaran.



Gambar 6.17 Hasil Pengujian dari Proses *Discretization* pada aplikasi

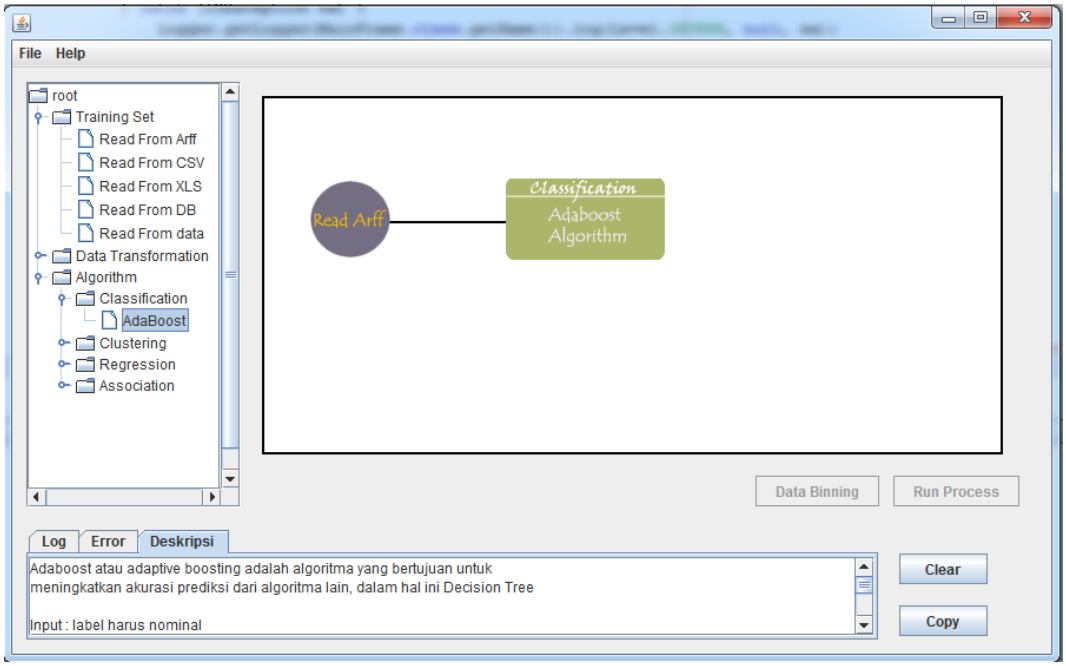
Pada Gambar 6.18 merupakan hasil sebenarnya berupa data yang berubah tipenya menjadi *discrete* dari proses *discretization.*



Gambar 6.18 Hasil Pengujian Data Hasil Diskretisasi pada aplikasi

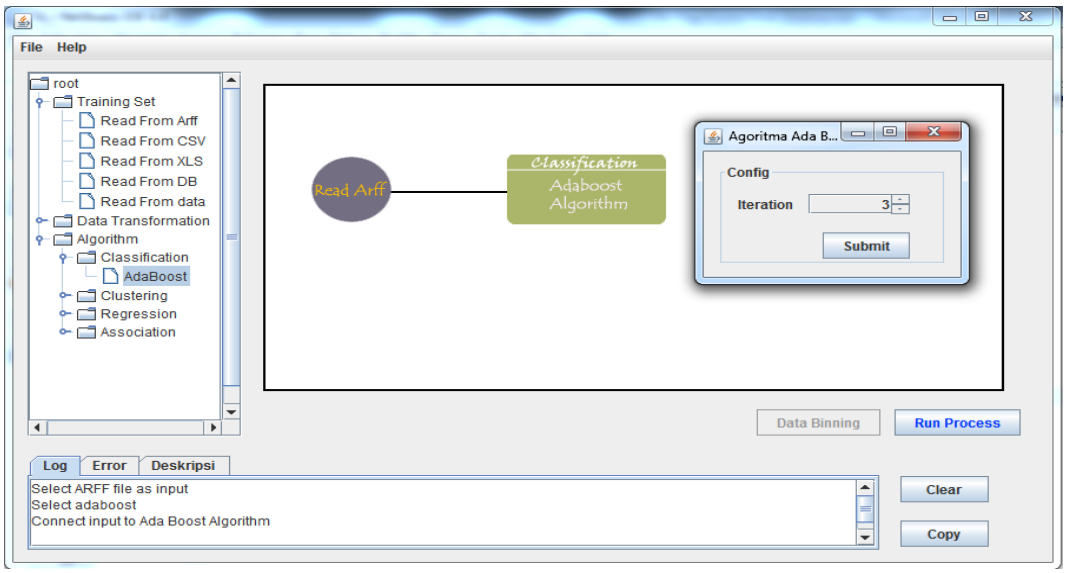
6.4 Pengujian Pemilihan dan Konfigurasi Algoritma

Pengujian dilakukan dengan memilih algoritma Adaboost yang terdapat pada menu yang terletak disebelah kiri jendela utama. Hasil yang diharapkan adalah operator algoritma muncul didalam *workspace* dan deskripsi mengenai algoritma Adaboost akan muncul pada *text area* deskripsi algoritma. Pada Gambar 6.19 merupakan hasil sebenarnya dari tampilan aplikasi setelah memilih algoritma.



Gambar 6.19 Hasil Pengujian Pemilihan Algoritma

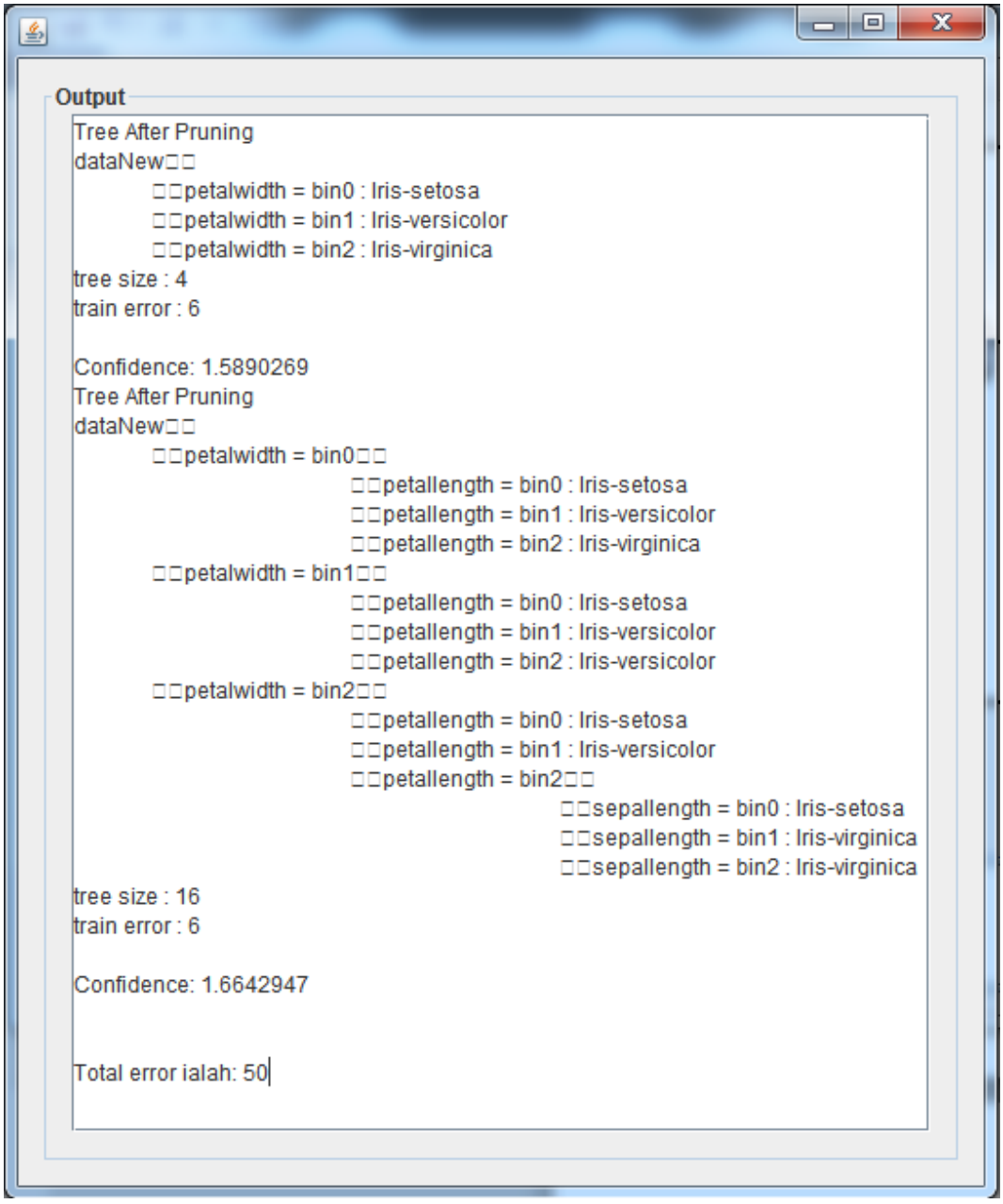
Operator algoritma telah muncul dan data masukan telah tersedia untuk diproses. Langkah selanjutnya adalah melakukan konfigurasi terhadap algoritma. Jendela konfigurasi algoritma bisa diakses dengan mengklik dua kali operator algoritma. Pada jendela tersebut terdapat parameter yang diperlukan oleh algoritma. Hasil yamg diharapkan adalah aplikasi menyimpan nilai dari parameter algoritma yang telah dikonfigurasi. Pada Gambar 6.20 merupakan hasil sebenarnya dari aplikasi.



Gambar 6.20 Hasil Pengujian Jendela Konfigurasi Algoritma

6.5 Pengujian Menjalankan Algoritma

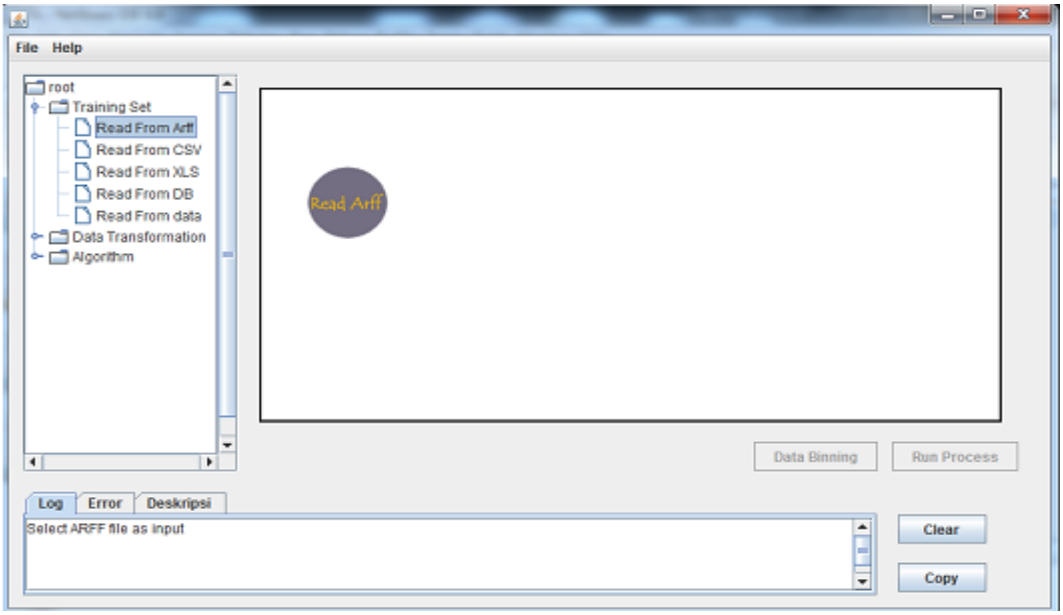
Pengujian menjalankan algoritma dilakukan terhadap algoritma Adaboost. Algoritma dijalankan dengan mengklik satu kali tombol *run process* yang ada di menu utama. Dalam pengujian ini, penulis memakai nilai parameter iterasinya adalah tiga. Hasil yang diharapkan adalah hasil dari proses yang dilakukan algoritma tersebut ditampilkan pada *text area* jendela keluaran. Pada Gambar 6.21 merupakan hasil sebenarnya dari aplikasi.



Gambar 6.21 Hasil pengujian Menjalankan Algoritma

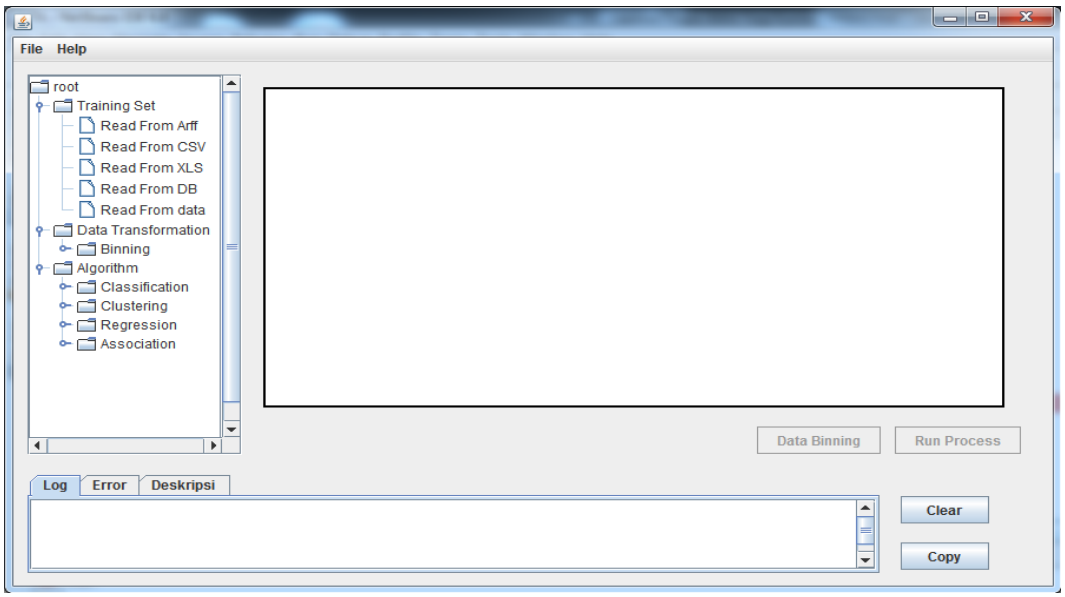
6.6 Pengujian Menghapus Operator

Pengujian menghapus operator dilakukan terhadap operator membaca berkas arff. Pada Gambar 6.22 adalah tampilan operator membaca berkas arff pada aplikasi.



Gambar 6.22 Tampilan operator membaca berkas arff pada aplikasi

Hasil yang diharapkan adalah operator yang terdapat pada *workspace* akan dihapuskan. Pada Gambar 6.23 merupakan hasil sebenarnya dari aplikasi. Terlihat operator tersebut telah dihilangkan dari *workspace.*



Gambar 6.23 Hasil pengujian menghapus operator

BAB 7
PENUTUP

Bab ini merupakan bab terakhir yang memberikan kesimpulan dari pembuatan tugas akhir ini dan saran untuk pengembangan lebih lanjut dalam topik yang berkaitan dengan tugas akhir, sehingga dapat menyempurnakan hasil penelitian.

7.1 Kesimpulan

Hasil yang telah dicapai melalui pelaksanaan penelitian tugas akhir ini adalah sebagai berikut.

1. Aplikasi *data mining* yang berbasiskan GUI pada penelitian ini dapat berfungsi sebagai *workspace* untuk mewadahi penggabungan algoritma *data mining*.
2. Tahap pemrosesan masukan telah diimplementasikan pada penelitian ini. Tahap tersebut dilakukan untuk mendapatkan data dari berbagai sumber yang akan digunakan sebagai masukan oleh algoritma.
3. Tahap membangun metadata dan memvisualisasikan masukan dalam bentuk grafik telah diterapkan pada penelitian ini. Metadata merupakan informasi penting untuk menampilkan karakteristik dari masukan tersebut. Grafik juga dapat membantu pengguna dalam memahami sebaran data masukan.
4. Tahap *preprocessing input* telah diterapkan pada penelitian ini. Permasahalahan data yang kualitasnya kurang baik bisa diselesaikan menggunakan tahap tersebut.

7.2 Saran

Berikut ini adalah saran yang penulis ajukan untuk pengembangan sistem selanjutnya :

1. Belum adanya pengimplementasian terhadap *preprocessing data* yang lengkap seperti *data cleaning*, dan *data reduction.*
2. Pemilihan *library* grafik yang lebih bagus tampilannya dan lebih presisi yang bertujuan untuk meningkatkan *user experience.*

3. Pilihan format *file* yang dapat digunakan sebaiknya diperbanyak agar pengguna lebih leluasa dalam mencari sumber *input.*
4. Fitur *generate input* sebaiknya difasilitasi bila pengguna ingin mencoba aplikasi ini dan tidak mempunyai sumber data.

DAFTAR PUSTAKA

Eriksson, H.-E., Penker, M., Lyons, B., & Fado, D. (2004). *UML 2 Toolkit.* Indianapolis: Wiley Publishing.

Gilbert, D., & Morgner, T. (t.thn.). *JFreeChart*. Dipetik April 2012, dari JFreeChart: http://www.jfree.org/jfreechart/

GmbH, R.-I. (2010). *RapidMiner 5.0 Manual.* Dipetik April 2012, dari http://www.rapidminer.com/

Grässle, P., Baumann, H., & Baumann, P. (2005). *UML 2.0 In Action.* Birmingham: Galileo Press.

Han, J., & Kamber, M. (2006). *Data Mining Concepts and Techniques 2nd edition.* San Francisco: Morgan Kaufmann Publishers.

Ian H. Witten, E. F. (t.thn.). *Weka: Practical Machine Learning Tools and Techniques.* Dipetik Mei 2012, dari University of Waikato Website: http://www.cs.waikato.ac.nz

Kohavi, R. (t.thn.). *SGI - MLC++*. Dipetik Mei 2012, dari Silicon Graphics International: http://www.sgi.com/tech/mlc/source.html

Kohavi, R., John, G., Long, R., Manley, D., & Pfleger, K. (1994). MLC++ : A Machine Learning Library in C++. 1-4.

Paynter, G. (t.thn.). *Attribute-Relation File Format (ARFF)*. Dipetik Juni 2012, dari Department of Computer Science University of Waikato: www.cs.waikato.ac.nz/ml/weka/arff.html

Pressman, R. S. (2005). *Software Engineering : A Practitioner's Aproach Sixth Edition.* New York: McGraw-Hill.

Sumathi, S., & Sivanandam, S. (2006). *Introduction to Data Mining and its Application.* New York: Springer.

Vogel, L. (2011, July 9). *Excel and Java - Read and Write Excel with Java* . Dipetik April 2012, dari Vogella: http://www.vogella.com/articles/JavaExcel/article.html

Witten, I. H., Frank, E., & Hall, M. A. (2011). *Data Mining Practical Machine Learning Tools and Techniques.* Burlington: Morgan Kaufmann Publishers.

Lampiran 1 : Berkas weather.arff

```
@relation weather

@attribute outlook {sunny, overcast, rainy}
@attribute temperature real
@attribute humidity real
@attribute windy {TRUE, FALSE}
@attribute play {yes, no}

@data
sunny,85,85,FALSE,no
sunny,80,90,TRUE,no
overcast,83,86,FALSE,yes
rainy,70,96,FALSE,yes
rainy,68,80,FALSE,yes
rainy,65,70,TRUE,no
overcast,64,65,TRUE,yes
sunny,72,95,FALSE,no
sunny,69,70,FALSE,yes
rainy,75,80,FALSE,yes
sunny,75,70,TRUE,yes
overcast,72,90,TRUE,yes
overcast,81,75,FALSE,yes
rainy,71,91,TRUE,no
```

Lampiran 2 : Berkas cpu.arff

```
@relation ‘cpu’

@attribute MYCT real
@attribute MMIN real
@attribute MMAX real
@attribute CACH real
@attribute CHMIN real
@attribute CHMAX real
@attribute class real
@data
400,1000,3000,0,1,2,38
400,512,3500,4,1,6,40
60,2000,8000,65,1,8,92
50,4000,16000,65,1,8,138
350,64,64,0,1,4,10
200,512,16000,0,4,32,35
167,524,2000,8,4,15,19
143,512,5000,0,7,32,28
143,1000,2000,0,5,16,31
110,5000,5000,142,8,64,120
```

```
143,1500,6300,0,5,32,30

143,3100,6200,0,5,20,33

143,2300,6200,0,6,64,61

110,3100,6200,0,6,64,76
```

Lampiran 3 : Berkas iris.arff

```
@RELATION iris

@ATTRIBUTE sepallength   REAL

@ATTRIBUTE sepalwidth    REAL

@ATTRIBUTE petallength   REAL

@ATTRIBUTE petalwidth    REAL

@ATTRIBUTE class {Iris-setosa,Iris-versicolor,Iris-virginica}

@DATA

5.1,3.5,1.4,0.2,Iris-setosa

4.9,3.0,1.4,0.2,Iris-setosa

4.7,3.2,1.3,0.2,Iris-setosa

4.6,3.1,1.5,0.2,Iris-setosa

5.0,3.6,1.4,0.2,Iris-setosa

5.4,3.9,1.7,0.4,Iris-setosa

4.6,3.4,1.4,0.3,Iris-setosa

5.0,3.4,1.5,0.2,Iris-setosa

4.4,2.9,1.4,0.2,Iris-setosa

4.9,3.1,1.5,0.1,Iris-setosa

5.4,3.7,1.5,0.2,Iris-setosa

4.8,3.4,1.6,0.2,Iris-setosa

4.8,3.0,1.4,0.1,Iris-setosa

4.3,3.0,1.1,0.1,Iris-setosa

5.8,4.0,1.2,0.2,Iris-setosa

5.7,4.4,1.5,0.4,Iris-setosa

5.4,3.9,1.3,0.4,Iris-setosa
```

5.1,3.5,1.4,0.3,Iris-setosa

5.7,3.8,1.7,0.3,Iris-setosa

5.1,3.8,1.5,0.3,Iris-setosa

5.4,3.4,1.7,0.2,Iris-setosa

5.1,3.7,1.5,0.4,Iris-setosa

4.6,3.6,1.0,0.2,Iris-setosa

5.1,3.3,1.7,0.5,Iris-setosa

4.8,3.4,1.9,0.2,Iris-setosa

5.0,3.0,1.6,0.2,Iris-setosa

5.0,3.4,1.6,0.4,Iris-setosa

5.2,3.5,1.5,0.2,Iris-setosa

5.2,3.4,1.4,0.2,Iris-setosa

4.7,3.2,1.6,0.2,Iris-setosa

4.8,3.1,1.6,0.2,Iris-setosa

5.4,3.4,1.5,0.4,Iris-setosa

5.2,4.1,1.5,0.1,Iris-setosa

5.5,4.2,1.4,0.2,Iris-setosa

4.9,3.1,1.5,0.1,Iris-setosa

5.0,3.2,1.2,0.2,Iris-setosa

5.5,3.5,1.3,0.2,Iris-setosa

4.9,3.1,1.5,0.1,Iris-setosa

4.4,3.0,1.3,0.2,Iris-setosa

5.1,3.4,1.5,0.2,Iris-setosa

5.0,3.5,1.3,0.3,Iris-setosa

4.5,2.3,1.3,0.3,Iris-setosa

```
4.4,3.2,1.3,0.2,Iris-setosa
5.0,3.5,1.6,0.6,Iris-setosa
5.1,3.8,1.9,0.4,Iris-setosa
4.8,3.0,1.4,0.3,Iris-setosa
5.1,3.8,1.6,0.2,Iris-setosa
4.6,3.2,1.4,0.2,Iris-setosa
5.3,3.7,1.5,0.2,Iris-setosa
5.0,3.3,1.4,0.2,Iris-setosa
7.0,3.2,4.7,1.4,Iris-versicolor
6.4,3.2,4.5,1.5,Iris-versicolor
6.9,3.1,4.9,1.5,Iris-versicolor
5.5,2.3,4.0,1.3,Iris-versicolor
6.5,2.8,4.6,1.5,Iris-versicolor
5.7,2.8,4.5,1.3,Iris-versicolor
6.3,3.3,4.7,1.6,Iris-versicolor
4.9,2.4,3.3,1.0,Iris-versicolor
6.6,2.9,4.6,1.3,Iris-versicolor
5.2,2.7,3.9,1.4,Iris-versicolor
5.0,2.0,3.5,1.0,Iris-versicolor
5.9,3.0,4.2,1.5,Iris-versicolor
6.0,2.2,4.0,1.0,Iris-versicolor
6.1,2.9,4.7,1.4,Iris-versicolor
5.6,2.9,3.6,1.3,Iris-versicolor
6.7,3.1,4.4,1.4,Iris-versicolor
5.6,3.0,4.5,1.5,Iris-versicolor
```

5.8,2.7,4.1,1.0,Iris-versicolor

6.2,2.2,4.5,1.5,Iris-versicolor

5.6,2.5,3.9,1.1,Iris-versicolor

5.9,3.2,4.8,1.8,Iris-versicolor

6.1,2.8,4.0,1.3,Iris-versicolor

6.3,2.5,4.9,1.5,Iris-versicolor

6.1,2.8,4.7,1.2,Iris-versicolor

6.4,2.9,4.3,1.3,Iris-versicolor

6.6,3.0,4.4,1.4,Iris-versicolor

6.8,2.8,4.8,1.4,Iris-versicolor

6.7,3.0,5.0,1.7,Iris-versicolor

6.0,2.9,4.5,1.5,Iris-versicolor

5.7,2.6,3.5,1.0,Iris-versicolor

5.5,2.4,3.8,1.1,Iris-versicolor

5.5,2.4,3.7,1.0,Iris-versicolor

5.8,2.7,3.9,1.2,Iris-versicolor

6.0,2.7,5.1,1.6,Iris-versicolor

5.4,3.0,4.5,1.5,Iris-versicolor

6.0,3.4,4.5,1.6,Iris-versicolor

6.7,3.1,4.7,1.5,Iris-versicolor

6.3,2.3,4.4,1.3,Iris-versicolor

5.6,3.0,4.1,1.3,Iris-versicolor

5.5,2.5,4.0,1.3,Iris-versicolor

5.5,2.6,4.4,1.2,Iris-versicolor

6.1,3.0,4.6,1.4,Iris-versicolor

5.8,2.6,4.0,1.2,Iris-versicolor

5.0,2.3,3.3,1.0,Iris-versicolor

5.6,2.7,4.2,1.3,Iris-versicolor

5.7,3.0,4.2,1.2,Iris-versicolor

5.7,2.9,4.2,1.3,Iris-versicolor

6.2,2.9,4.3,1.3,Iris-versicolor

5.1,2.5,3.0,1.1,Iris-versicolor

5.7,2.8,4.1,1.3,Iris-versicolor

6.3,3.3,6.0,2.5,Iris-virginica

5.8,2.7,5.1,1.9,Iris-virginica

7.1,3.0,5.9,2.1,Iris-virginica

6.3,2.9,5.6,1.8,Iris-virginica

6.5,3.0,5.8,2.2,Iris-virginica

7.6,3.0,6.6,2.1,Iris-virginica

4.9,2.5,4.5,1.7,Iris-virginica

7.3,2.9,6.3,1.8,Iris-virginica

6.7,2.5,5.8,1.8,Iris-virginica

7.2,3.6,6.1,2.5,Iris-virginica

6.5,3.2,5.1,2.0,Iris-virginica

6.4,2.7,5.3,1.9,Iris-virginica

6.8,3.0,5.5,2.1,Iris-virginica

5.7,2.5,5.0,2.0,Iris-virginica

5.8,2.8,5.1,2.4,Iris-virginica

6.4,3.2,5.3,2.3,Iris-virginica

6.5,3.0,5.5,1.8,Iris-virginica

7.7,3.8,6.7,2.2,Iris-virginica

7.7,2.6,6.9,2.3,Iris-virginica

6.0,2.2,5.0,1.5,Iris-virginica

6.9,3.2,5.7,2.3,Iris-virginica

5.6,2.8,4.9,2.0,Iris-virginica

7.7,2.8,6.7,2.0,Iris-virginica

6.3,2.7,4.9,1.8,Iris-virginica

6.7,3.3,5.7,2.1,Iris-virginica

7.2,3.2,6.0,1.8,Iris-virginica

6.2,2.8,4.8,1.8,Iris-virginica

6.1,3.0,4.9,1.8,Iris-virginica

6.4,2.8,5.6,2.1,Iris-virginica

7.2,3.0,5.8,1.6,Iris-virginica

7.4,2.8,6.1,1.9,Iris-virginica

7.9,3.8,6.4,2.0,Iris-virginica

6.4,2.8,5.6,2.2,Iris-virginica

6.3,2.8,5.1,1.5,Iris-virginica

6.1,2.6,5.6,1.4,Iris-virginica

7.7,3.0,6.1,2.3,Iris-virginica

6.3,3.4,5.6,2.4,Iris-virginica

6.4,3.1,5.5,1.8,Iris-virginica

6.0,3.0,4.8,1.8,Iris-virginica

6.9,3.1,5.4,2.1,Iris-virginica

6.7,3.1,5.6,2.4,Iris-virginica

6.9,3.1,5.1,2.3,Iris-virginica

```
5.8,2.7,5.1,1.9,Iris-virginica

6.8,3.2,5.9,2.3,Iris-virginica

6.7,3.3,5.7,2.5,Iris-virginica

6.7,3.0,5.2,2.3,Iris-virginica

6.3,2.5,5.0,1.9,Iris-virginica

6.5,3.0,5.2,2.0,Iris-virginica

6.2,3.4,5.4,2.3,Iris-virginica

5.9,3.0,5.1,1.8,Iris-virginica
```

Lampiran 4 : Berkas contact-lenses.csv

```
age,spectacle-prescrip,astigmatism,tear-prod-rate,contact-lenses
young,myope,no,reduced,none
young,myope,no,normal,soft
young,myope,yes,reduced,none
young,myope,yes,normal,hard
young,hypermetrope,no,reduced,none
young,hypermetrope,no,normal,soft
young,hypermetrope,yes,reduced,none
young,hypermetrope,yes,normal,hard
pre-presbyopic,myope,no,reduced,none
pre-presbyopic,myope,no,normal,soft
pre-presbyopic,myope,yes,reduced,none
pre-presbyopic,myope,yes,normal,hard
pre-presbyopic,hypermetrope,no,reduced,none
pre-presbyopic,hypermetrope,no,normal,soft
pre-presbyopic,hypermetrope,yes,reduced,none
pre-presbyopic,hypermetrope,yes,normal,none
presbyopic,myope,no,reduced,none
presbyopic,myope,no,normal,none
presbyopic,myope,yes,reduced,none
presbyopic,myope,yes,normal,hard
presbyopic,hypermetrope,no,reduced,none
presbyopic,hypermetrope,no,normal,soft
presbyopic,hypermetrope,yes,reduced,none
```

Lampiran 5 : book1.xls

	A	B	C	D	E
1	Outlook	Temperature	Humidity	Windy	Play
2	Sunny	Hot	High	TRUE	No
3	Overcast	Hot	High	FALSE	Yes
4	Rainy	Mild	High	FALSE	Yes
5	Rainy	Cool	Normal	FALSE	Yes
6	Rainy	Cool	Normal	TRUE	No
7	Overcast	Cool	Normal	TRUE	Yes
8	Sunny	Mild	High	FALSE	No
9	Sunny	Cool	Normal	FALSE	Yes
10	Rainy	Mild	Normal	FALSE	Yes
11	Sunny	Mild	Normal	TRUE	Yes
12	Overcast	Hot	Normal	FALSE	Yes
13	Rainy	Mild	High	TRUE	No
14	Sunny	Hot	Normal	TRUE	No
15	Sunny	Hot	Normal	FALSE	No
16	Sunny	Mild	High	TRUE	No
17	Sunny	Mild	Normal	FALSE	No
18	Sunny	Cool	High	TRUE	Yes
19	Sunny	Cool	High	FALSE	Yes
20	Sunny	Cool	Normal	TRUE	Yes

Lampiran 6 : dataNew.data dan dataNew.names

dataNew.data

bin0,bin2,bin0,bin0,Iris-setosa

bin0,bin1,bin0,bin0,Iris-setosa

bin0,bin1,bin0,bin0,Iris-setosa

bin0,bin1,bin0,bin0,Iris-setosa

bin0,bin2,bin0,bin0,Iris-setosa

bin0,bin2,bin0,bin0,Iris-setosa

bin0,bin2,bin0,bin0,Iris-setosa

bin0,bin2,bin0,bin0,Iris-setosa

bin0,bin0,bin0,bin0,Iris-setosa

bin0,bin1,bin0,bin0,Iris-setosa

dataNew.names

sepallength : bin0, bin1, bin2

sepalwidth : bin0, bin1, bin2

petallength : bin0, bin1, bin2

petalwidth : bin0, bin1, bin2

class: Iris-setosa, Iris-versicolor, Iris-virginica, Iris-setosa, Iris-versicolor, Iris-virginica

Lampiran 7 : Daftar *attribute* dan *method* dari tiap *class*

Class ArffFilter

Method

Name	Parameters	Returns	Visibility
accept	File	boolean	public
getDescription	-	String	public
getExtension	File	String	public

Class AttributeInstanceReader

Attributes

Name	Type	Visibility
relation	String	private
dataInstancesArray	DataInstance[]	private
attributesNameList	List<String>	private
attributesArray	Attribute[]	private
attributesList	List<String[]>	private
dataInstancesList	List<int[]>	private
realPosition	List<Integer>	private
countAttributeValue	List< int[]>	private

dataInstancesList2	List<String[]>	private
columnReal	List<double[]>	private
tree	TreeSet<String>	private
classes	List<String>	private
label	String	private
fileIsOk	boolean	private
isReal	boolean	private
instanceList	String[]	public
row	int	public
col	int	public
binningAttributes	List<BinningAttribute[]>	public

Methods

Name	Parameters	Returns	Visibility
AttributeInstanceReader			public
AttributeInstanceReader File file			public

Class BinningAttribute

Attributes

Name	Type	Visibility
bin	String	private
value	String	private

Methods

Name	Parameters	Returns	Visibility
BinningAttribute	String, String	void	public
getBin	-	String	public
getValue	-	String	public

Class Chart

Attributes

Name	Type	Visibility
chartPanel	JPanel	private
panel	ChartPanel	public
attributeList	List<String[]>	public
attributeNameList	List<String>	public

dataInstanceList	List<String[]>	public
countAttributeValue	List<int[]>	public
myFileReader	AttributeInstanceReader	public
db	OpenDBFrame	public
dataBar	DefaultCategoryDataset	public
dataset	XYSeriesCollection	public
xyDataSet	XYDataset	public
chart	JFreeChart	public
chartFrame	ChartFrame	public
graph	String	public
postAttribute1	int	public
postAttribute2	int	public

Methods

Name	Parameters	Returns	Visibility
createDataSet	-	void	private
createChart	-	void	private
initComponents	-	void	private

Chart	AttributeInstanceReader, int, int, String	void	public
Chart	OpenDBFrame, int, int, String	void	public

Class CSVFilter

Methods

Name	Parameters	Returns	Visibility
accept	File	boolean	public
getDescription	-	String	public
getExtension	File	String	public

Class DataFilter

Operations

Name	Parameters	Returns	Visibility
accept	File	boolean	public
getDescription	-	String	public
getExtension	File	String	public

Class FrameParameterAlgoritma

Attributes

Name	Type	Visibility
iterationSpinner	JSpinner	private
jLabel1	JLabel	private
jPanel1	JPanel	private
saveModelButton	JButton	private
submitButton	JButton	private

Methods

Name	Parameters	Returns	Visibility
saveModelButtonActionPerformed	java.awt.event.ActionEvent	void	private
submitButtonActionPerformed	java.awt.event.ActionEvent	void	private
FrameAlgoritmaAdaboost	-	void	public
FrameAlgoritmaAdaboost	AttributeInstanceReader	void	public

Class FrameChosenInstance

Attributes

Name	Type	Visibility
attributesArray	Attribute[]	private
dataInstancesArray	DataInstance[]	private
dataInstancesList	List<String[]>	private
classes	List<String>	private
attributeNameList	List<String>	private
attributeList	List<String[]>	private
binningAttributes	List<BinningAttribute[]>	private
myFileReader	AttributeInstanceReader	private
chosenInstanceTable	JTable	private
jLabel1	JLabel	private
jScrollPane1	JScrollPane	private
noButton	JButton	private
yesButton	JButton	private
chosenInstanceModel	DefaultTableModel	public
openDBFrame	OpenDBFrame	public
frameInput	FrameInput	public

Methods

Name	Parameters	Returns	Visibility
noButtonActionPerformed	java.awt.event.ActionEvent	void	private
yesButtonActionPerformed	java.awt.event.ActionEvent	void	private
writeAllAndNamesFile	-	void	private
FrameChosenInstance	-	void	public
FrameChosenInstance	AttributeInstanceReader, OpenDBFrame, FrameInput	void	public
getMyFileReader	-	AttributeInstanceReader	public
setMyFileReader	AttributeInstanceReader	void	public

Class FrameInput

Attributes

Name	Type	Visibility
myFileReader	AttributeInstanceReader	private
attributesArray	Attribute[]	private

dataInstancesArray	DataInstance[]	private
choosenValue	String[]	private
choosenInstance	List<String[]>	private
countAttributeValue	List<int[]>	private
attributeTable	javax.swing.JTable	private
graphicButton	javax.swing.JButton	private
graphicComboBox	javax.swing.JComboBox	private
instanceTable	javax.swing.JTable	private
jPanel1	javax.swing.JPanel	private
jPanel3	javax.swing.JPanel	private
jPanel4	javax.swing.JPanel	private
jScrollPane1	javax.swing.JScrollPane	private
jScrollPane2	javax.swing.JScrollPane	private
jScrollPane3	javax.swing.JScrollPane	private
jTabbedPane1	javax.swing.JTabbedPane	private
metadataInputInstance	javax.swing.JLabel	private
metadataInputRelation	javax.swing.JLabel	private
metadataInputType	javax.swing.JLabel	private

openFileButton	javax.swing.JButton	private
selectAllButton	javax.swing.JButton	private
selectValueButton	javax.swing.JButton	private
tableValueAttribute1	javax.swing.JTable	private
box2	JCheckBox	public
dataInstancesList	List<String[]>	public
attributeName	List<String>	public
attributesList	List<String[]>	public
fileType	String	public
postAttribute1	int	public
postAttribute2	int	public

Methods

Name	Parameters	Returns	Visibility
attributeTableMouseClicked	java.awt.event.MouseEvent	void	private
selectAllButtonActionPerformed	java.awt.event.ActionEvent	void	private

instanceTableMouseClicked	java.awt.event.MouseEvent	void	private
selectValueButtonActionPerformed	java.awt.event.MouseEvent	void	private
openFileButtonActionPerformed	java.awt.event.ActionEvent	void	private
graphicButtonActionPerformed	java.awt.event.ActionEvent	void	private
graphicComboBoxActionPerformed	java.awt.event.ActionEvent	void	private
FrameInput	String	void	public
FrameInput	-	void	public
getGraphicComboBox	-	JComboBox	public

Class FrameOutput

Attributes

Name	Type	Visibility
graphicComboBox	JComboBox	private

jPanel1	JPanel	private
jPanel2	JPanel	private
jPanel3	JPanel	private
jPanel4	JPanel	private
jTabbedPane1	JTabbedPane	private

Methods

Name	Parameters	Returns	Visibility
initComponents	-	void	private
FrameOutput	-	void	public

Class MainFrame

Attributes

Name	Type	Visibility
buttonRunProcess	javax.swing.JButton	private
clearButton	javax.swing.JButton	private
copyButton	javax.swing.JButton	private
dataBinningButton	javax.swing.JButton	private
deskripsiTextArea	javax.swing.JTextArea	private

errorTextArea	javax.swing.JTextArea	private
jDialog1	javax.swing.JDialog	private
jMenu1	javax.swing.JMenu	private
jMenu2	javax.swing.JMenu	private
jMenuBar1	javax.swing.JMenuBar	private
jMenuItem1	javax.swing.JMenuItem	private
jMenuItem2	javax.swing.JMenuItem	private
jPanel1	javax.swing.JPanel	private
jScrollPane1	javax.swing.JScrollPane	private
jScrollPane2	javax.swing.JScrollPane	private
jScrollPane3	javax.swing.JScrollPane	private
jScrollPane4	javax.swing.JScrollPane	private
jScrollPane5	javax.swing.JScrollPane	private
jTabbedPane1	javax.swing.JTabbedPane	private
logTextArea	javax.swing.JTextArea	private
treeAlgorithm	javax.swing.JTree	private
workspacePanel	javax.swing.JPanel	private
readArffInput	javax.swing.JLabel	public

readCsvInput	javax.swing.JLabel	public
readXlsInput	javax.swing.JLabel	public
readDbInput	javax.swing.JLabel	public
readDataInput	javax.swing.JLabel	public
discretize	javax.swing.JLabel	public
categorize	javax.swing.JLabel	public
adaBoostAlgorithm	javax.swing.JLabel	public
cartAlgorithm	javax.swing.JLabel	public
testing	javax.swing.JLabel	public
lineInputToAlgorithm	javax.swing.JLabel	public
lineInputToDiscretize	javax.swing.JLabel	public
lineDiscretizeToAlgorithm	javax.swing.JLabel	public
algorithmUsed	String	public
deskripsiAlgoritma	String	public
fi	FrameInput	public
faa	FrameAlgoritmaAdaboost	public
fo	FrameOutput	public
myFileReader	AttributeInstanceReader	public

inputArffClicked	boolean	public
inputCsvClicked	boolean	public
inputXlsClicked	boolean	public
inputDbClicked	boolean	public
inputDataClicked	boolean	public
adaboostClicked	boolean	public
cartClicked	boolean	public
discretizeClicked	boolean	public
iteration	int	public
resample	int	public
ot	OutputTesting	public
binningAttributes	List<BinningAttribute[]>	public
ba	BinningAttribute	public
binningInstance	String[][]	public

Methods

Name	Parameters	Returns	Visibility
treeListener	javax.swing.event.TreeSelectionEvent	void	private

clearButtonActionPerformed	java.awt.event.ActionEvent	void	private
copyButtonActionPerformed java.awt.event.ActionEvent evt	java.awt.event.ActionEvent	void	private
buttonRunProcessActionPerformed	java.awt.event.ActionEvent	void	private
dataBinningButtonActionPerformed	java.awt.event.ActionEvent	void	private
binningDiscretizeMouseClicked	java.awt.event.MouseEvent	void	private
binningCategorizeMouseClicked	java.awt.event.MouseEvent	void	private
readArffInputMouseClicked	java.awt.event.MouseEvent	void	private
readCsvInputMouseClicked	java.awt.event.MouseEvent	void	private
readXlsInputMouseClicked	java.awt.event.MouseEvent	void	private
readDbInputMouseClicked	java.awt.event.MouseEvent	void	private
readDataInputMouseClicked	java.awt.event.MouseEvent	void	private

adaBoostAlgorithmMouseClicked	java.awt.event.MouseEvent	void	private
cartAlgorithmMouseClicked	java.awt.event.MouseEvent	void	private
setBinningToInstance	-	void	private
setFileNamesBinning	-	void	private
setFileDataBinning	-	void	private
MainFrame	-	void	public
getMyFileReader	-	AttributeInstanceReader	public
setMyFileReader	AttributeInstanceReader	void	public
setParamAlgorithm	int[] param	void	public
main	String args[]	void	public

Class OpenDBFrame

Attributes

Name	Type	Visibility
connection	Connection	private
myFileReader	AttributeInstanceReader	private
tree	TreeSet<String>	private

instanceList	List<String[]>	private
attributeValueList	List<String[]>	private
attributeNameList	ArrayList<String>	private
countAttributeValue	List<int[]>	private
res	ResultSetMetaData	public
rs	ResultSet	public
stmt	Statement	public
sql	String	public
url	String	public
log	String	public
modelTableInstance	DefaultTableModel	public
modelTableAttribute	DefaultTableModel	public
postAttribute1	int	public
postAttribute2	int	public

Methods

Name	Parameters	Returns	Visibility
initComponents	-	void	private

graphicButtonActionPerformed	java.awt.event.ActionEvent	void	private
graphicComboBoxActionPerformed	java.awt.event.ActionEvent	void	private
useDBButtonActionPerformed	java.awt.event.ActionEvent	void	private
instanceTableMouseClicked	java.awt.event.MouseEvent	void	private
clearDBButtonActionPerformed	java.awt.event.ActionEvent	void	private
executeQueryButtonActionPerformed	java.awt.event.ActionEvent	void	private
connectURLButtonActionPerformed	java.awt.event.ActionEvent	void	private
selectAllDBButtonActionPerformed	java.awt.event.ActionEvent	void	private
OpenDBFrame	-	void	public
getInstanceTable	-	javax.swing.JTable	public
getattributeDBTable	-	javax.swing.JTab	public

getInstanceList	-	le List<String[]>	public

Class OutputTesting

Attributes

Name	Type	Visibility
jPanel1	JPanel	private
jScrollPane2	JScrollPane	private
outputTextArea	JTextArea	private

Operations

Name	Parameters	Returns	Visibility
initComponents	-	void	private
OutputTesting	-	void	public
OutputTesting	String	void	public
main	String[]	void	public

Class SelectAttribute

Attributes

Name	Type	Visibility

jButton1	javax.swing.JButton	private
jLabel1	javax.swing.JLabel	private
jLabel2	javax.swing.JLabel	private
jPanel1	javax.swing.JPanel	private
cb	javax.swing.JComboBox	public
cb2	javax.swing.JComboBox	public
temp	String[]	public
f	FrameInput	public
db	OpenDBFrame	public

Methods

Name	Parameters	Returns	Visibility
initComponents	-	void	private
jButton1ActionPerformed	java.awt.event.ActionEvent	void	private
SelectAttribute	-	void	public
SelectAttribute	List<String>, FrameInput	void	public
SelectAttribute	List<String>, OpenDBFrame	void	public

main	String[]	void	public

Class XLSFilter

Methods

Name	Parameters	Returns	Visibility
accept	File	boolean	public
getDescription	-	String	public
getExtension	File	String	public