TESIS

ALGORITMA PERBAIKAN PENENTUAN TITIK PUSAT AWAL BERBASIS HIRARKI UNTUK KLASTERISASI DATA KATEGORIKAL

Oleh:

TITA KARLITA
5104 201 005

PROGRAM STUDI MAGISTER
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2006

TESIS

ALGORITMA PERBAIKAN PENENTUAN TITIK PUSAT AWAL BERBASIS HIRARKI UNTUK KLASTERISASI DATA KATEGORIKAL

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Komputer (M.Kom)
di
Institut Teknologi Sepuluh Nopember Surabaya

Oleh :

TITA KARLITA
5104 201 005

Disetujui oleh Tim Penguji Tesis :

Tanggal Ujian : 7 Agustus 2006
Periode Wisuda : September 2006

Prof. Dr. Ir. Arif Djunaidy, M.Sc.
NIP. 131 633 403

Ir. FX. Arunanto, M.Sc.
NIP. 131 285 253

Febriliyan Samopa, S.Kom., M. Kom.
NIP. 132 206 858

M. M. Irfan Subakti, S.Kom., M.Sc. Eng.
NIP. 132 300 412

Direktur Program Pascasarjana

Prof. Ir. Happy Ratna S.,M.Sc., Ph.D.
NIP. 130 541 829

ABSTRACT

ABSTRACT

HIERARCHICAL INITIAL POINTS REFINEMENT ALGORITHM FOR CATEGORICAL DATA CLUSTERING

Name : Tita Karlita
Number of Student : 5104 201 005
Supervisor : Prof. Dr. Ir. Arif Djunaidy, M.Sc.

The k-means clustering algorithm is designed to work primarily on numeric datasets. Besides this, the k-modes algorithm extends the k-means paradigm to cluster categorical data by using a frequency-based method to update the cluster centroid. However, as in the case with most data clustering algorithms, the k-*modes* algorithm requires a pre-setting or random selection of initial points (modes). Choosing different initial points for each running will produces different results.

In this thesis, an extention to k-modes algorithm; i.e., hierarchical initial-points refinement algorithm for categorical data clustering, that is applied for sub sample data to produce good clustering result is developed. Sub sample data that is produced using *density based multiscale data condensation* algorithm becomes an input to the hierarchical clustering algorithm. Some centroids that are produced by the hierarchical clustering algorithm becomes initial points for *k-modes* algorithm.

Experimental results show that the hierarchical initial-points refinement algorithm produced higher precision and more reliable results than those produced by the random selection method without refinement. In addition to this, experimental results also show that the hierarchical initial points refinements algorithm involving data reduction algorithm produced similar quality results with those produced by the hierarchical initial-points refinement algorithm without data reduction.

Keywords : data clustering, k-modes algorithm, data reduction, categorical data, data mining.

ABSTRAK

ABSTRAK

ALGORITMA PERBAIKAN PENENTUAN TITIK PUSAT AWAL BERBASIS HIRARKI UNTUK KLASTERISASI DATA KATEGORIKAL

Nama : Tita Karlita
NRP : 5104 201 005
Pembimbing : Prof. Dr. Ir. Arif Djunaidy, M.Sc.

Algoritma klasterisasi k-means didesain hanya untuk bekerja pada data berjenis numerik. Dilain pihak algoritma k-modes dikembangkan berdasarkan pada paradigma algoritma k-means agar dapat digunakan untuk mengklaster data kategorikal dengan menggunakan ukuran keserupaan yang didasarkan pada metode frekuensi kemunculan suatu nilai dalam suatu atribut untuk memutakhirkan titik pusat klaster. Kebanyakan algoritma k-modes yang dipublikasikan pada saat ini menentukan titik pusat awal dengan cara random. Pemilihan titik pusat yang berbeda dapat menghasilkan klaster yang berbeda sehingga hasil klaster tidak stabil.

Dalam tesis ini dikembangkan algoritma *k-modes* yang menggunakan algoritma klasterisasi hirarki yang diaplikasikan pada data sub-sampel untuk menghasilkan titik pusat awal yang baik. Data sub-sampel yang dibentuk dengan proses reduksi data menjadi masukan bagi algoritma klasterisasi hirarki. Selanjutnya sejumlah titik pusat yang dihasilkan oleh klasterisasi hirarki dijadikan sebagai masukan titik pusat awal bagi algoritma *k-modes*.

Hasil uji coba menunjukkan bahwa algoritma perbaikan penentuan titik pusat awal berbasis hirarki mampu menghasilkan hasil klaster yang lebih baik dan stabil bila dibandingkan dengan algoritma klasterisasi yang penentuan titik pusat awalnya dilakukan secara random. Selain itu, hasil uji coba juga menunjukkan bahwa hasil klaster dari proses klasterisasi yang melibatkan proses reduksi data mempunyai tingkat akurasi yang sama jika dibandingkan dengan hasil klaster dari proses klasterisasi yang tidak melibatkan proses reduksi data.

Kata Kunci: klasterisasi data, algoritma *k-modes*, reduksi data, data kategorikal, data mining.

Kupersembahkan untuk orang-orang yang sangat kucintai

Suamiku Itok,

Anakku Evan

Serta Mama, Papa, dan Ibuku

KATA PENGANTAR

KATA PENGANTAR

Atas berkat limpahan Rahmat dan Hidayah dari Allah SWT, Alhamdulillah akhirnya tesis yang berjudul:

ALGORITMA PERBAIKAN PENENTUAN TITIK PUSAT AWAL BERBASIS HIRARKI UNTUK KLASTERISASI DATA KATEGORIKAL

dapat diselesaikan dengan baik, dengan tak lupa mengucapkan puja-puji syukur kehadirat-Nya. Tesis ini adalah sebagai salah satu syarat untuk menempuh ujian magister komputer pada Program Pascasarjana Teknik Informatika Fakultas Teknologi Informasi Institut Teknologi Sepuluh Nopember (ITS) Surabaya.

Dengan selesai dan tersusunnya laporan tesis ini, maka penulis mengucapkan banyak terima kasih atas bantuan dan dukungan dari berbagai pihak baik moril atau materiil, kepada:

- Suamiku tercinta Waskitho Wibisono dan buah hatiku Evan Aditia Wibisono, terimakasih banyak atas dukungan semangat dan doa yang tulus agar tugas belajar ini selesai dengan baik. Bersamamu hidup ini terasa lebih indah.
- Papa dan Mama Putut Sunarjo, Bapak (Alm) dan Ibu Karol, Bapak(Alm) dan Ibu Sudarno, terimakasih atas kasih sayang yang tak ternilai. Semoga Allah SWT membalas kebaikan yang telah diberikan.
- My beloved Sister and Brother, I love you so much.
- Om, tante dan sepupu di Malang, terimakasih atas kebaikan dan dukungannya.
- Bapak Prof. Dr. Ir. Arif Djunaidy, M.Sc. selaku dosen pembimbing yang telah banyak memberikan bimbingan dan pengarahan selama penulis mengerjakan tesis.
- Seluruh staf pengajar S2 Teknik Informatika ITS yang telah memberikan bekal ilmu pengetahuan bagi penulis.
- Bapak Dr. Ir. Titon Dutono, MEng., selaku Direktur Politeknik Elektronika Negeri Surabaya, yang telah memberikan kesempatan penulis untuk tugas belajar di program pascasarjana.
- Bapak dan Ibu Pimpinan dan seluruh staf Dosen dan Karyawan Politeknik Elektronik Negeri Surabaya yang telah memberikan dorongan semangat dan bantuan fasilitas selama kuliah dan pembuatan tesis.
- Teman-teman kuliah angkatan 2004 S2 Teknik Informatika ITS, yang telah memberikan dorongan moril serta persahabatan selama perkuliahan.
- Teman-teman sesama mengerjakan tesis Yuliana Setyowati dan Entin Martiana yang menjadi penyemangat mengerjakan tesis, serta Ali Ridho Barakbah terimakasih atas saran-saran dan diskusinya.

- Teman-teman di Jurusan Teknologi Informasi PENS-ITS yang banyak membantu dan menyemangati penulis untuk segera menyelesaikan tesis.
- Semua pihak yang tidak dapat disebutkan satu persatu yang telah banyak membantu penulis dalam menyelesaikan studi S2 Teknik Informatika di ITS.

Semoga Allah S.W.T. membalas semua kebaikan tersebut dengan pahala yang berlimpah, Amin.

Dengan selesainya tesis ini, maka penulis barharap agar buku ini dapat membawa manfaat bagi para pembaca pada umumnya dan juga bagi penulis pada khususnya, serta bagi pihak-pihak yang berkepentingan. Seperti kata pepatah yang mengatakan bahwa "***Tak Ada Gading yang Tak Retak***", begitu pula laporan tesis ini, untuk itu penulis mengharapkan saran dan koreksinya demi sempurnanya tesis ini untuk pengembangan masa mendatang.

DAFTAR ISI

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR TABEL

DAFTAR PSEUDOCODE

DAFTAR PSEUDOCODE

DAFTAR SEGMEN PROGRAM

DAFTAR SEGMEN PROGRAM

BAB 1

PENDAHULUAN

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Salah satu operasi penting dalam data mining adalah melakukan klasterisasi terhadap data set berukuran besar. Klasterisasi merupakan teknik yang bisa digunakan untuk mengelompokkan objek, dimana dalam satu kelompok yang terbentuk memiliki nilai keserupaan objek yang besar dan antar kelompok memiliki nilai keserupaan yang kecil.

Penelitian dibidang klasterisasi yang banyak dikembangkan saaat ini berfokus pada tipe data numerik. Padahal banyak data yang direprentasikan dalam bentuk data kategorikal, dimana nilai-nilai atributnya secara alami tidak bisa diperlakukan sebagai data numerik. Sebagai contoh atribut data kategorikal adalah atribut berdomain bentuk yang nilainya bisa berupa lingkaran, elips, bujur sangkar, dan lain-lain. Dengan sifat atribut data kategorikal khusus seperti itu maka untuk mengklaster akan menjadi sulit dan kompleks daripada mengklaster data numerik.

Beberapa algoritma klasterisasi data kategorikal telah dikembangkan misalnya algoritma CACTUS [Gan99]. Algoritma CACTUS didasarkan pada ringkasan yang cepat. Terdiri dari tiga fase yaitu fase: *summarization, klastering*, dan *validation*. Algoritma yang lain adalah ROCK [Gu99]. ROCK mengadaptasi algoritma klasterisasi hirarki aglomeratif. Algoritma ini dimulai dengan

mengalokasikan tiap objek ke klaster yang terpisah, selanjutnya akan diganbung secara bertahap sesuai dengan jarak yang terdekat. Ukuran terdekat antar klaster dihitung dari jumlah "garis" antar pasangan yang dimiliki oleh suatu objek. Garis antar titik dihitung sebagai jumlah ketetanggaan antar objek. Algoritma Squeezer [He02] membaca objek berulang-ulang satu persatu. Ketika objek pertama dibaca, objek ini akan menjadi satu klaster sendiri. Berdasarkan suatu fungsi keserupaan maka akan ditentukan objek tersebut akan digabung dengan klaster yang lain atau tetap sendiri. Algoritma COOLCAT [Bar02] adalah algoritma heuristik yang didasarkan pada entropi. Dimulai dengan metode heuristik yang meningkatkan nilai rasio "height-to-width" histogram klaster.

Algoritma klasterisasi yang lain adalah algoritma klasterisasi yang dikembangkan oleh Huang [Huang97a], algoritma k-modes, merupakan pengembangan algoritma k-means agar dapat digunakan untuk mengklaster data kategorikal. Kontribusi utama dari algoritma k-modes ini adalah penggunaan ukuran ketidakserupaan berupa kecocokan suatu nilai atribut tiap dimensi terhadap titik pusat klaster, dimana titik pusat *means* diganti dengan *modes*, dan penggunaan metode yang didasarkan pada frekuensi untuk memutakhirkan *modes*.

Oleh karena algoritma k-modes yang dikembangkan oleh Huang menggunakan paradigma algoritma k-means maka algoritma k-modes yang dikembangkan mempunyai karakteristik yang sama dengan algoritma k-means. Dalam konteks ini hasil klaster dipengaruhi oleh nilai pembangkitan titik pusat awal klaster yang dipilih secara random sehingga klaster yang dihasilkan tidak selalu sama. Penelitian yang dilakukan oleh Funderlic (Fun04), menemukan

bahwa algoritma k-modes juga dapat terjebak pada optima lokal. Selain itu, hasil klasterisasi dengan algoritma k-modes juga menghasilkan nilai keserupaan dalam satu klaster yang rendah karena hanya mempertimbangkan penggunaan ukuran ketidakserupaan berupa metode yang didasarkan pada frekuensi untuk memutakhirkan titik pusat klaster [He05].

Algoritma untuk memperbaiki algoritma k-modes telah dikembangkan [Sun02] yaitu algorima perbaikan penentuan titik pusat awal yang dilakukan secara iteratif. Tetapi algoritma yang dikembangkan oleh Sun membutuhkan memori yang besar karena untuk melakukan optimasi titik pusat awal harus menyimpan sejumlah data sub-sampel dan membutuhkan waktu komputasi yang besar karena harus berulang kali mengklaster data sub-sampel tersebut.

1.2 Perumusan Masalah

Dengan kelemahan algoritma k-modes berkaitan dengan penentuan titik pusat awal yang menggunakan cara random sehingga hasil klaster tidak selalu baik [Sun02], maka dibutuhkan suatu metode yang lebih baik untuk melakukan klasterisasi data kategorikal sehingga dapat dihasilkan klaster yang baik dan stabil. Berdasarkan hal tersebut diatas, maka dirasa penting untuk melakukan suatu pengembangan terhadap algoritma k-modes yang dipublikasikan oleh [Huang97a] terutama pada bagian penentuan titik pusat awal supaya tidak ditentukan secara random lagi. Pertanyaan penelitian yang terkait dengan masalah ini adalah bagaimana mendesain algoritma klasterisasi yang bertujuan untuk memperbaiki algoritma k-modes pada penentuan titik pusat awal. Implementasi

algoritma perbaikan ini juga dilakukan untuk melakukan uji coba dan evaluasi kinerja algoritma yang didesain.

Masalah berikutnya adalah bila dalam melakukan perbaikan penentuan titik pusat awal melibatkan keseluruhan data masukan maka ruang pencarian titik pusat awal akan menjadi besar hal ini menyebabkan waktu komputasi akan menjadi lama. Oleh karena itu, sebagai masukan proses penentuan titik pusat awal digunakan data sub-sampel dari data keseluruhan sehingga bisa mereduksi waktu komputasi. Pertanyaan yang muncul dari masalah ini adalah apakah penentuan titik pusat awal yang dikenakan pada data sub-sampel tetap dapat memberikan hasil akurasi yang sama bila dibandingkan bila dikenakan pada data keseluruhan.

1.3 Batasan Penelitian

Data set yang akan digunakan pada penelitian ini adalah data set yang semua atributnya bernilai kategorikal, dimana masing-masing atribut mempunyai domain kategorikal. Struktur data yang digunakan dalam merepresentasikan data set menggunakan dua macam struktur data yaitu matriks data dan matriks ketidakserupaan yang digambarkan pada persamaan 1.1 dan 1.2.

$$\begin{bmatrix} x_{11} & \cdots & x_{ij} & \cdots & x_{1p} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{i1} & \cdots & x_{ij} & \cdots & x_{ip} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{nj} & \cdots & x_{np} \end{bmatrix} \quad (1.1)$$

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \ldots & \ldots & \ldots & 0 & \\ d(n,1) & d(n,2) & \ldots & \ldots & 0 \end{bmatrix} \quad (1.2)$$

Persamaan 1.1 merupakan matriks data yang berisi n objek dan p variabel. Struktur ini berbentuk tabel relasional dalam bentuk matriks berukuran n x p. Persamaan 1.2 merupakan matriks ketidakserupaan. Matriks ini menyimpan sekumpulan kedekatan yang mungkin terjadi diantara objek-objek. Hal tersebut ditampilkan dalam matriks berukuran n x n.

Untuk menentukan titik pusat awal digunakan algoritma barbasis hirarki aglomeratif yang diaplikasikan pada data sub-sampel. Untuk membentuk data sub-sampel digunakan algoritma *Density-Based Multiscale Data Condensation* [Mit02]. Algoritma ini menggunakan metode k-nearest-neighbor untuk menghitung kepadatan pada suatu area tertentu. Sedang algoritma klasterisasi berbasis hirarki aglomeratif dipilih untuk proses penentuan titik pusat awal karena algoritma ini dapat menghasilkan klaster yang baik. Sejumlah titik pusat yang dihasilkan algoritma ini dapat digunakan sebagai titik pusat awal bagi algoritma klasterisasi data kategorikal.

1.4 Tujuan Penelitian

Penelitian yang telah dilakukan menyajikan pengembangan algoritma k-modes yang bertujuan untuk:

a. Mengembangkan algoritma k-modes yang melibatkan penentuan titik pusat awal, sehingga dihasilkan titik pusat awal yang tidak lagi ditentukan secara random yang pada akhirnya akan menghasilkan klaster yang baik dan stabil.
b. Mempercepat proses penentuan titik pusat awal yang melibatkan proses reduksi data untuk membentuk data sub-sampel.

1.5 Kontribusi dan Manfaat Penelitian

Kontribusi dari hasil penelitian ini berupa tersedianya sebuah metode yang bisa digunakan untuk melakukan klasterisasi data kategorikal yang berdimensi banyak. Dalam kebanyakan literatur, metode klasterisasi data kategorikal ini mempunyai metode pengolahan data yang terbatas jika dibandingkan dengan data numerik. Dari hasil penelitian ini penentuan titik pusat awal dioptimalkan dengan suatu metode klasterisasi hirarki algomeratif yang diaplikasikan pada data sub-sampel sehingga menghasilkan klaster yang memiliki rasio kesalahan yang rendah dan membutuhkan waktu komputasi penentuan titik pusat awal yang relatif cepat.

1.6 Sistematika Penulisan

Penulisan yang dilakukan dalam pembahasan penelitian ini dibagi menjadi enam bab. Bab 1 berisi pendahuluan yang menjelaskan latar belakang permasalahan, perumusan masalah, batasan masalah, tujuan dan manfaat penelitian, kontribusi penelitian serta sistematika penulisan

Bab 2 berisi tentang teori dasar data mining khususnya klastering, dimulai dengan pengantar ilmu data mining, konsep dasar klasterisasi, konsep keserupaan, teknik dasar klasterisasi, data kategorikal, konsep reduksi data, algoritma klasterisasi hirarki, dan algoritma klasterisasi partisi, kemudian dilanjutkan membahas penelitian-penelitian yang berkaitan dengan klasterisasi data kategorikal.

Bab 3 berisi tentang algoritma perbaikan titik pusat awal berbasis hirarki untuk mengklaster data kategorikal. Bagian awal bab ini menjelaskan tentang praproses data, pembentukan data sub-sampel, optimasi titik pusat awal, titik pusat konseptual, objek penelitian, praproses data, proses klasterisasi data kategorikal dan diakhiri dengan pengukuran kompleksitas algoritma.

Dalam bab 4 dijelaskan dua tahapan penting yaitu desain aplikasi dan tahap implementasi algoritma untuk algoritma perbaikan penentuan titik pusat berbasis hirarki. Untuk menjelaskan desain aplikasi akan digunakan penggambaran konsep *Unified Modelling Languange* (*UML*) yang dimiliki oleh aplikasi Rational Rose Enterprise Edition 2002, sedangkan tahap implementasi meliputi implementasi kelas dan implementasi antar-muka dari aplikasi.

Bab 5 menjelaskan mengenai uji coba yang telah dilaksanakan. Pertama dibahas tentang lingkungan pelaksanaan uji coba dari aplikasi. Kemudian dijelaskan mengenai beberapa dataset yang digunakan dalam uji coba, skenario uji coba, lingkungan uji coba, dan diakhiri dengan pelaksanaan uji coba serta evaluasi hasil uji coba.

Sebagai penutup, dalam bab 6 dijelaskan simpulan yang dapat diambil dari hasil penelitian dan uji coba. Selain itu bab ini menyajikan saran yang bermanfaat untuk pengembangan lebih lanjut dari penelitian ini.

BAB 2

TINJAUAN KEPUSTAKAAN

BAB 2

TINJAUAN KEPUSTAKAAN

Bab 2 berikut ini menjelaskan topik yang dibahas yaitu pengantar ilmu data mining, konsep dasar klasterisasi, konsep keserupaan, teknik dasar klasterisasi, data kategorikal, konsep reduksi data, algoritma klasterisasi hirarki, dan algoritma klasterisasi partisi, kemudian dilanjutkan membahas penelitian-penelitian yang berkaitan dengan klasterisasi data kategorikal.

2.1 Pengantar Data Mining

Teknologi basis data saat ini memungkinkan untuk menyimpan sejumlah data dalam jumlah yang sangat besar dan terakumulasi. Disinilah awal timbulnya persoalan ledakan data (jumlah data yang tiba-tiba begitu sangat besar). Data memang perlu disimpan, tapi yang lebih penting dari itu adalah proses penemuan pengetahuan *(knowledge)* dari data yang disimpan. Oleh karena itu data yang tersimpan dalam sebuah gudang data (*data warehouses)* perlu dianalisis. Solusi untuk persoalan penemuan pengetahuan dalam database berukuran besar adalah dengan menggunakan *data warehousing* dan *data mining*.

2.1.1 Definisi Data Mining

Ada beberapa definisi dari data mining. Secara umum data mining dapat didefinisikan sebagai berikut[Iman04]:

- Proses penemuan pola yang menarik dari data yang tersimpan dalam jumlah besar. Ini merupakan evolusi alami dari teknologi database, dan merupakan metode yang paling banyak dibutuhkan, dengan aplikasi yang sangat luas.
- Ekstraksi dari suatu informasi yang berguna atau menarik (non-trivial, implisit, sebelumnya belum diketahui, potensial kegunaannya), pola atau pengetahuan dari data yang disimpan dalam jumlah besar.

Data mining memiliki beberapa sebutan atau nama lain yaitu: *Knowledge Discovery in Databases* (KDD), ekstraksi pengetahuan (*knowledge extraction*), Analisis data/pola, kecerdasan bisnis (*business intelligence*), dan lain-lain.

2.2.2 Jenis Data Mining

Pada prakteknya, terdapat dua jenis data mining [Iman04], yaitu *prediktif* dan *deskriptif*. Prediktif melibatkan penggunaan beberapa variabel atau field dalam database untuk memprediksi hal-hal yang tidak/belum diketahui. Sedang deskriptif terfokus pada penemuan pola yang dapat menginterprestasikan data.

Prediktif dan deskriptif dapat diperoleh dengan menggunakan metode data mining yaitu klasifikasi, regresi, klasterisasi, ringkasan, dan pemodelan. Klasifikasi adalah fungsi pembelajaran yang memetakan (mengklasifikasikan) item data ke dalam beberapa kelas yang terdefinisi. Regresi pembelajaran suatu fungsi yang memetakan item data ke dalam variabel prediksi yang bersifat nyata. Klasterisasi adalah deskripsi secara umum untuk mengidentifikasi himpunan kategori atau klaster yang bersifat *finite* untuk mendeskripsikan data. Ringkasan melibatkan metode untuk menemukan deskripsi subhimpunan data. Pemodelan

merupakan penemuan suatu model yang menggambarkan ketergantungan secara signifikan antar berbagai variabel.

Model ketergantungan *(dependency)* terdiri dari dua levelyaitu level struktural dan level kuantitatif. Level *struktural* yang menentukan variabel mana yang tergantung secara lokal satu sama lain. Level kuantitatif yang menentukan kelebihan ketergantungan yang menggunakan beberapa skala numerik.

2.1.3 Proses Penemuan Pengetahuan pada Data Mining

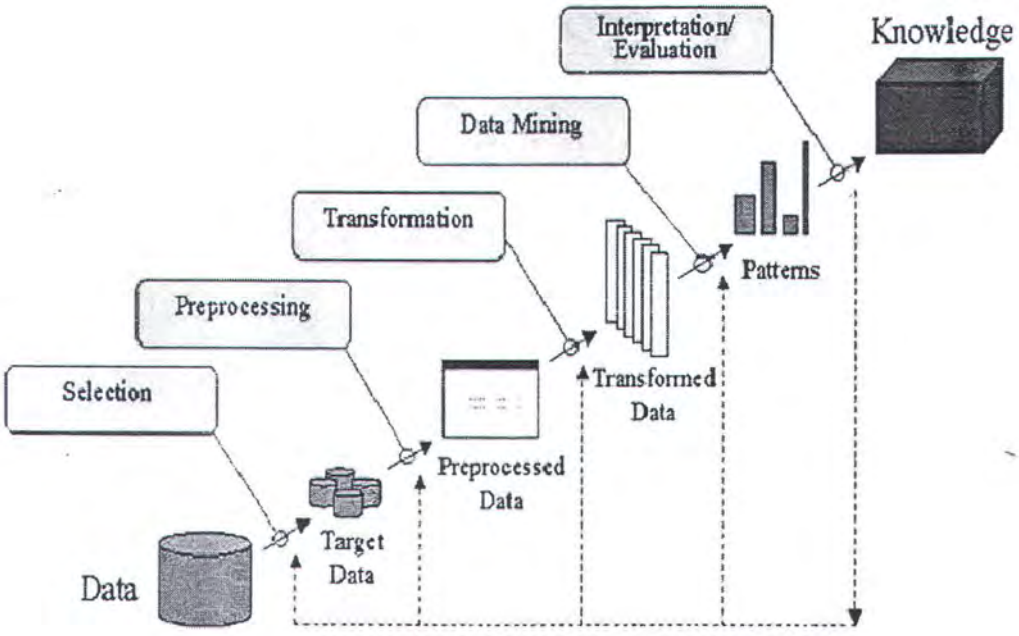
Proses penemuan pengetahuan dalam data mining disebut sebagai *Knowledge Discovery in Database* disingkat KDD. Pada dasarnya proses yang ada pada KDD meliputi pengumpulan data, pengolahan data, pembersihan data, integrasi data, pemilihan data, transformasi, data mining, evaluasi pola/pattern, dan presentasi pengetahuan (*knowledge*). Proses penemuan 'knowledge' pada data mining dijelaskan oleh gambar 2.1.

Langkah-langkah proses KDD secara detil meliputi tahapan berikut [Sya03]:

- Pembelajaran domain aplikasi

  Pengetahuan sebelumnya yang relevan dengan persoalan dan tujuan dari aplikasi yang dibuat

- Membuat himpunan target data: pemilihan data



Gambar 2.1. Proses pencarian 'knowledge' pada data mining

- Pembersihan Data dan preprocessing: (hampir 60% usaha dilakukan pada tahap ini)
- Reduksi Data dan transformasi

  Menemukan fitur yang berguna, dimensionalitas/reduksi variabel, representasi *invariant*.
- Pemilihan Algoritma mining

  Ringkasan (*summary*), klasifikasi, regresi, asosiasi, clustering.
- Implementasi Data mining: pencarian pola yang menarik
- Evaluasi pola dan representasi pengetahuan

  Visualisasi, tranformasi, membuat pola yang redundant, dan lain-lain.
- Menggunakan pengetahuan yang ditemukan.

2.1.4 Pentingnya Suatu Pola dalam Data Mining

Data mining bisa menghasilkan ribuan pola: tapi tidak semuanya penting Suatu pola dikatakan penting jika dimengerti dengan mudah oleh manusia, valid pada data baru dengan beberapa tingkatan kepentingan, kegunaan yang potensial, atau valid terhadap beberapa hipotesis. Pengukuran kepentingan pada data mining dapat bersifat objektif dan subjektif. Pengukuran Objektif: berdasar pada statistik dan struktur pola , dukungan, kepastian, dan lain-lain. Sedangkan pengukuran Subjektif: berdasar pada kepercayaan user pada data yang ada.

Pada data mining, semua pola yang penting diusahakan untuk ditemukan selengkap mungkin. Proses pencarian pola yang penting dapat dilakukan dengan berdasarkan:

- pencarian heuristik: mencari pola dengan menggunakan ilmu kecerdasan buatan.
- pencarian exhaustive: mencari semua kemungkinan yang ada.

2.2 Konsep dan Teknik Dasar Klasterisasi

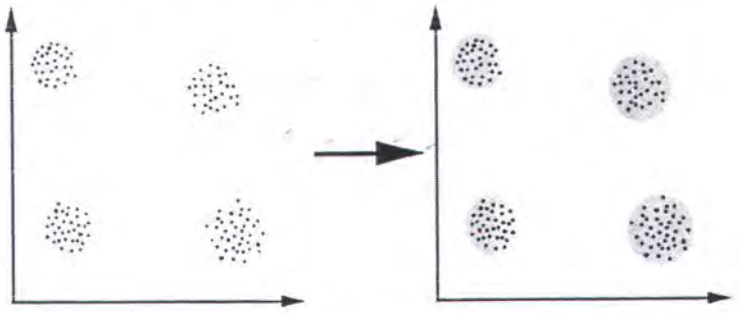
Sebelum mulai membahas algoritma-algoritma klasterisasi maka terlebih dahulu harus memahami konsep dan teknik dasar klasterisasi. Beberapa sub bab berikutnya akan membahas tentang konsep dan teknik dasar klasterisasi.

2.2.1 Konsep Dasar Klasterisasi

Klasterisasi merupakan salah satu bentuk dari proses pembelajaran yang tidak terbimbing, dimana objek yang akan dikelompokkan tidak memiliki label

atau tanda. Klasterisasi berupaya untuk melakukan pengaturan agar objek yang dikelompokkan berada dalam kelompok yang anggotanya serupa atau homogen. Klasterisasi juga berupaya agar kelompok baru yang terbentuk berbeda dengan kelompok lainnya atau heterogen [Huang97a].

Proses klasterisasi dapat ditunjukkan pada gambar 2.2, dimana ada objek yang diklaster menjadi empat kelompok. Kelompok baru yang terbentuk dipengaruhi oleh ukuran keserupaan yang digunakan. Ukuran keserupaan itu biasanya adalah jarak. Sehingga objek yang jaraknya berdekatan, memiliki peluang yang besar untuk menjadi anggota kelompok yang sama. Demikian juga objek yang berjauhan akan menjadi anggota pada kelompok yang berbeda.



Gambar 2.2 Proses klasterisasi yang menghasilkan empat buah klaster

Klasterisasi merupakan metode untuk mengelompokkan objek yang berukuran besar ke dalam kelompok yan berukuran lebih kecil. Kelompok baru yang terbentuk belum tentu merupakan kelompok yang terbaik, karena tidak ada hubungan mutlak antara kriteria terbaik dengan target akhir dari klasterisasi. Sehingga yang dapat dijadikan ukuran adalah seberapa homogen anggota dalam kelompok yang terbentuk. Keduanya dipengaruhi oleh ukuran jarak yang digunakan dan bagaimana metode klasterisasi yang diterapkan.

Klasterisasi telah banyak diterapkan dalam kehidupan sehari-hari, antara lain dalam bidang [Moo01]:

- Pemasaran, untuk menemukan karakteristik kelompok pelanggan berdasarkan perilaku pembelian yang telah mereka lakukan pada masa lalu.
- Biologi, untuk mengklasifikasi binatang dan hewan berdasarkan ciri–ciri yang mereka miliki.
- Perpustakaan, untuk pemesanan buku.
- Asuransi, untuk mengidentifikasi kelompok polis asuransi pemilik kendaraan bermotor berdasarkan rata–rata klaim yang mereka lakukan.
- Perencanaan kota, untuk mengidentifikasi kelompok rumah menurut jenis rumah, harga dan penempatan geografis.

2.2.2 Teknik Dasar Klasterisasi

Terdapat dua teknik klastering, yaitu Partisional dan Hirarkikal dengan definisi sebagai berikut:

Partisional

Terdapat database dengan sejumlah n objek, algoritma klastering partisional membentuk sejumlah k partisi data sedemikian hingga fungsi objektif terpenuhi. Permasalahan yang timbul pada algoritma ini adalah kompleksitas yang tinggi, karena harus melakukan enumerasi pada semua kemungkinan pengelompokan untuk mencapai global optima. Bahkan untuk jumlah objek yang kecil sekalipun, jumlah partisi sangat besar. Oleh karena itu solusi umum dimulai dengan titik

pusat awal yang biasanya dipilih secara random, dipartisi dan selanjutnya dilakukan perbaikan. Pada prakteknya, untuk mendapatkan hasil yang lebih baik adalah dengan cara menjalankan algoritma beberapa kali dengan k titik pusat awal yang berbeda dan memilih hasil yang paling baik.

Hirarkikal

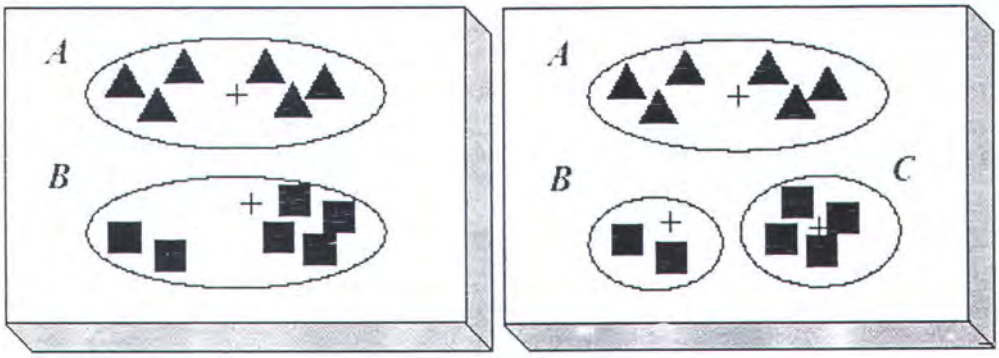
Algoritma hirarki membuat dekomposisi hirarki beberapa objek. Bisa berupa *agglomerative* (*bottom-up*) atau *divisive* (*top-down*):

(a) Algoritma *agglomerative* dimulai dengan tiap objek mewakili dirinya sendiri untuk menjadi suatu klaster. Selanjutnya satu sama lain digabung berdasarkan ukuran jarak tertentu. Proses klaster berhenti bila semua objek menjadi satu kelompok atau beberapa kelompok yang telah ditentukan. Metode ini biasanya mengikuti aturan *greedy bottom-up merging*.

(b) Algoritma *divisive* merupakan lawan dari algoritma *agglomerative*. Dimulai dari satu kelompok yang berisi semua objek, kemudian dipecah menjadi beberapa kelompok yang lebih kecil, sampai tiap objek mewakili dirinya sendiri menjadi suatu klaster, atau sampai didapatkan sejumlah kelompok yang telah ditentukan. Metode ini mirip dengan algoritma devide and conquer.

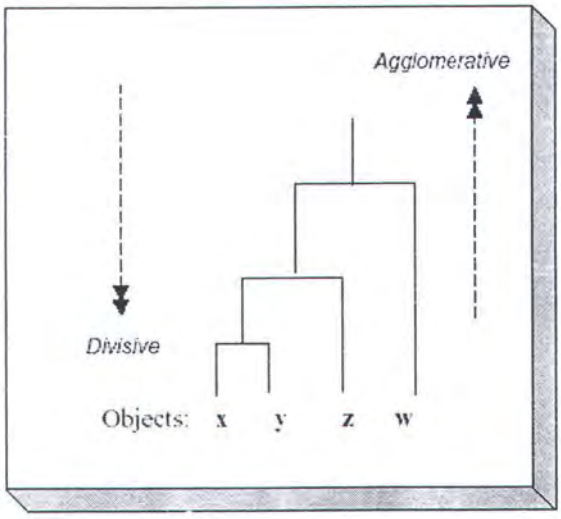
Metode partisi dan hirarki dapat digabungkan, sebagai contoh hasil yang diperoleh dengan menggunakan metode hirarki selanjutnya ditingkatkan dengan menggunakan metode partisi, yang memperbaiki hasil dengan cara iterasi merelokasi titik atau objek.

Gambar 2.3 menggambarkan contoh dua teknik klastering yang dilakukan pada data set yang sama, dengan parameter awal yang berbeda. Tanda "+" menandakan pusat klaster, dimana pada kasus ini didefinisikan sebagai nilai rata-rata objek yang berada dalam klaster tertentu. Gambar 2.4 menggambarkan *dendrogram* yang dihasilkan baik dengan menggunakan algoritma klastering divisive atau *agglomerative*. Suatu *dendrogram* adalah struktur tree yang menggambarkan urutan penggabungan selama proses klastering terjadi berdasarkan nilai jarak atau keserupaan.



Klasterisasi dengan k=2 *Klasterisasi dengan k=2*

Gambar 2.3 Algoritma klastering partisi.



Gambar 2.4 *Dendrogram* algoritma klastering hirarki

2.2.3 Konsep Keserupaan

Konsep keserupaan ini digunakan untuk menjawab seberapa dekat, atau seberapa jauh suatu objek terhadap objek lainnya. Secara umum semakin mirip dua objek maka semakin besar keserupaan diantara keduanya dan semakin besar ketidakserupaan diantara keduanya. Keserupaan dapat diukur dengan berbagai cara, salah satu diantaranya adalah jarak. Lebih jauh, jarak dapat diukur dengan menggunakan menggunakan salah satu variasi pengukuran jarak. Semua pengukuran bergantung pada tipe atribut yang sedang dianalisisi Misalnya, untuk data kategorikal, tidak bisa menggunakan pengukuran jarak yang membutuhkan orientasi geometris data, karena data kategorikal tidak mempunyai orientasi geometris.

Untuk semua jenis pengukuran, terdapat beberapa panduan yang disebut dengan *metrik* (disebut juga dengan *metrik jarak*). Untuk tiga titik data $\hat{x}, \hat{y}$ dan $\hat{z}$ dimana semua titik tersebut berada dalam D maka metrik jarak d harus memenuhi syarat sebagai berikut [And04] :

1. $d(\hat{x}, \hat{y}) \geq 0$: *non-negativity*;
2. $d(\hat{x}, \hat{y}) = 0$ hanya dan hanya jika $\hat{x} = \hat{y}$: *identity*;
3. $d(\hat{x}, \hat{y}) = d(\hat{y}, \hat{x})$: *symmetry*;
4. $d(\hat{x}, \hat{z}) \leq d(\hat{x}, \hat{y}) + d(\hat{y}, \hat{z})$: *triangle inequality*;

Beberapa metrik digunakan untuk menghitung jarak objek $\hat{x}$ dan $\hat{y}$, jika objek $\hat{x}$ dan $\hat{y}$ didefinisikan oleh atribut numerik, maka bisa digunakan perhitungan *Minkowski Distance* yang didefinisikan sebagai berikut:

$$d(\hat{x},\hat{y}) = (\sum_{i=1}^{m} | x_i - y_i |^{q})^{1/q} \quad (2.1)$$

dimana q adalah bilangan integer positif. Dari *Minkowski Distance* tersebut bisa diturunkan beberapa rumus sebagai berikut:

- untuk $q = 2$, disebut dengan *Euclidean Distance*, yang didefinisikan:

$$d(\hat{x},\hat{y}) = \sqrt{\sum_{i=1}^{m}(x_i - y_i)^2} \quad (2.2)$$

- untuk $q = 1$, disebut dengan *Manhattan Distance*, yang didefinisikan:

$$d(\hat{x},\hat{y}) = \sum_{i=1}^{m} | x_i - y_i |^2 \quad (2.3)$$

- untuk $q = \infty$, disebut dengan *Manhattan Distance*, yang didefinisikan:

$$d(\hat{x},\hat{y}) = \max_{i=1}^{m} | x_i - y_i | \quad (2.4)$$

Harus diperhatikan bahwa metrik diatas tidak bisa digunakan untuk objek yang didefinisikan oleh kombinasi data bernilai numerikal dan kategorikal.

Pengukuran keserupaan untuk data binari bisa dilakukan dengan bantuan tabel kontingensi sebagaimana digambarkan pada tabel 2.1.

Tabel 2.1 Tabel kontingensi untuk objek binari $\hat{x}$ dan $\hat{y}$, dengan $\tau = \alpha + \beta + \gamma + \delta$

	$\hat{y}:1$	$\hat{y}:0$	
$\hat{x}:1$	α	β	$\alpha+\beta$
$\hat{x}:0$	γ	δ	$\gamma+\delta$
	$\alpha+\gamma$	$\beta+\delta$	τ

Untuk objek $\hat{x}$ dan $\hat{y}$ dengan hanya nilai binari, maka nilai yang mungkin adalah 0 atau 1. Maksud simbol pada tabel kontingensi 2.1 adalah sebagai berikut:

- α adalah jumlah atribut, i, dimana $x_i = y_i = 1$;
- β adalah jumlah atribut, i, dimana $x_i = 1$ dan $y_i = 0$;
- γ adalah jumlah atribut, i, dimana $x_i = 0$ dan $y_i = 1$;
- δ adalah jumlah atribut, i, dimana $x_i = y_i = 0$;

Kemudian metrik untuk menghitung keserupaan adalah:

- *Simple Matching Coefficient*, didefinisikan sebagai:

$$d(\hat{x}, \hat{y}) = \frac{\alpha + \delta}{\tau} \tag{2.5}$$

- *Jaccard Coefficient*, didefinisikan sebagai:

$$d(\hat{x}, \hat{y}) = \frac{\alpha}{\alpha + \beta + \gamma} \tag{2.6}$$

Perhatikan bahwa koefisien ini mengabaikan angka kesamaan 0 dengan 0.

2.3 Data Kategorikal

Data kategorikal adalah data yang menggambarkan objek yang hanya mempunyai atribut kategorikal. Objek, yang disebut dengan objek kategorikal, adalah versi penyederhanaan dari objek simbolis. Kita anggap semua atribut numerik (kuantitatif) adalah atribut terkategorisasi dan tidak menganggap atribut kategorikal yang mempunyai nilai kombinasi, misalnya Languagesspoken (Chinese, English).

2.3.1 Domain dan Atribut Kategorikal

Misal A1, A2,..., Am adalah sejumlah m atribut yang berada pada space Ω dan DOM(A1), DOM(A2), ..., DOM(Am) adalah domain dari atribute. Domain DOM(Aj) didefinisikan sebagai kategorikal jika bersifat terbatas dan tidak terurut, misalnya untuk a, b $\in$ DOM(Aj), baik a = b atau a $\neq$ b. Aj disebut dengan atribut kategorikal. Ω disebut dengan space kategorikal jika semua A1, A2, ..., Am adalah kategorikal.

Domain kategorikal yang didefinisikan pada paper bersifat *singleton*. Nilai kombinasional seperti ini tidak diperbolehkan. Nilai khusus, yang dinotasikan oleh $\in$, didefinisikan pada semua domain kategorikal dan digunakan untuk merepresentasikan nilai yang hilang. Untuk menyederhanakan ukuran ketidakserupaan, kita tidak mempertimbangkan hubungan konseptual antara nilai dalam domain kategorikal misalnya car dan vehicle adalah dua nilai kategorikal dalam domain dan secara konseptual car adalah juga merupakan vehicle. Meskipun hubungan ini memang ada dalam dunia nyata.

2.3.2 Objek Kategorikal

Seperti pada objek kategorikal $X \in \Omega$ secara logik direpresentasikan sebagai konjungsi nilai pasangan atribut [A1 = x1] $\wedge$[A2 = x2] $\wedge$...$\wedge$ [Am = xm], dimana xj $\in$ DOM(Aj) untuk $1 \leq j \leq m$. Nilai pasangan atribut $A_j = x_j$ disebut dengan *selector*. Tanpa ambigu direpresentasikan X sebagai a vector [x1, x2,

...,xm]. Kita anggap tiap objek dalam Ω mempunyai tepat m atribut values. Jika nilai atribut Aj tidak tersedia untuk objek X, maka Aj = $\in$.

Anggap X = {X1, X2, ..., Xn} adalah himpunan n kategorikal objek dan $X \subseteq \Omega$. Objek Xi direpresentasikan sebagai [xi,1, xi,2,..., xi,m]. Ditulis Xi = Xk jika xi,j = xk,j untuk $1 \leq j \leq m$. Relasi Xi = Xk tidak berarti bahwa Xi, Xk adalah objek yang sama dalam dunia nyata. Hal ini berarti bahwa dua objek mempunyai nilai kategorikal yang sama pada atribut A1, A2, ..., Am. Sebagai contoh, dua pasien dalam data set mungkin saja mempunyai nilai yang sama pada atribut Sex, Disease dan Treatment. Namun dalam database rumah sakit, hal itu dibedakan oleh atribut lain misalnya ID dan Address dimana kedua atribut tersebut tidak dipertimbangkan dalam proses klastering.

Asumsikan X terdiri dari n objek dimana terdapat sejumlah p objek yang berbeda. Anggap N adalah kardinalitas dari produk kartesian DOM(A1) x DOM(A2) x ... x DOM(Am). Didapat p ≤ N. Namun , n mugkin lebih besar dari N, yang berarti bahwa ada duplikat dalam X.

2.4 Konsep Reduksi Data

Permasalahan yang populer dari data mining dan data warehousing adalah biaya untuk penyimpanan data, dimana data yang harus disimpan bisa mencapai ukuran terabyte. Proses penggalian data dengan ukuran gigabyte yang kecil ketika diproses dengan sebuah metode *machine learning* seringkali sudah membutuhkan perangkat keras dan algoritma secara paralel. Penyelesaian untuk permasalahan ini adalah dengan pemilihan sub himpunan yang kecil dari data untuk proses

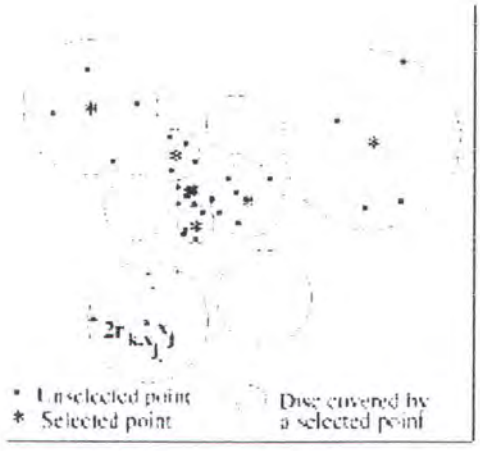
learning. Di satu sisi yang lain data seringkali berisi data yang redundan Dengan demikian akan lebih baik data yang besar diwakili oleh sub himpunan kecil dengan pola dari data asli direpresentasikan dalam data reduksi.

Penerapan sederhana dari reduksi data adalah teknik random, yaitu pemilihan data reduksi dilakukan secara random. Terdapat beberapa pemilihan data reduksi secara statistik, dimana setiap anggota data mempunyai peluang untuk dipilih sebagai sampel, yaitu random *sampling, stratified sampling*, dan *peepholing*. Metode reduksi yang sederhana tidak dapat diterapkan pada data nyata dengan noise. Penerapan algoritma random menghilangkan informasi dari data-data yang tidak terpilih dalam proses reduksinya. Algoritma reduksi harus menyertakan informasi dari semua data dalam proses reduksinya.

Beberapa skema penelitian untuk reduksi data dibangun dari penerapan klasifikasi secara umum dan aturan k-NN secara khusus. Keefektifan dari data reduksi terukur dalam keakuratan hasil klasifikasi. Metode yang pertama kali adalah penyingkatan data dengan aturan k-NN adalah CNN diinspirasikan oleh Hart. Beberapa algoritma lain yang termasuk dalam reduksi NN dan algoritma penyingkatan secara iteratif. Matriks pembobotan kesamaan secara asimetrik lokal (LASM) diterapkan pada reduksi data dan menunjukkan kinerja yang bagus dibanding metode berbanding k-NN. Selain itu terdapat algoritma reduksi data dan model bervariasi berbasiskan *neural network* didiskusikan dalam. Bagaimanapun juga metode reduksi yang berbasiskan pada klasifikasi sangat spesifik terhadap model dan masalah klasifikasi yang digunakan. Reduksi data

yang ditampilkan dengan metode vektor kuantisasi klasik menggunakan sebuah himpunan yang berisi vektor kode yang meminimumkan error kuantisasi.

Kelompok lain dari metode reduksi data adalah metode yang berbasiskan pada ukuran kepadatan data dengan mempertimbangkan fungsi kepadatan dari data lebih bertujuan untuk penyingkatan daripada meminimumkan error kuantisasi. Metode ini tidak menyertakan learning dalam prosesnya dan deterministik (dengan input yang sama akan memberikan output yang tetap) Metode *Density Based Multiscale Data Consensation* (DBMDC) [Mit02] dikembangkan dari kelompok ini. Algoritma ini bekerja mengurangi data dengan melakukan pada skala yang berbeda. Dengan skala yang berbeda diharapkan algoritma efisien dalam memperkirakan error kepadatan dan sesuai dengan distribusi data. Selain penggunaan skala yang berbeda, dalam algoritma ini digunakan beberapa skala dengan harapan informasi yang ada akan diwakili dalam data reduksi. Gambar 2.5 menunjukkan ilustrasi reduksi data dengan algoritma *Density Based Multiscale Data Consensation.*



Gambar 2.5. Ilustrasi reduksi data dengan DBMDC

Algoritma reduksi data meliputi perkiraan kepadatan pada suatu titik, mengurutkan titik-titik berdasarkan kriteria kepadatan, memilih suatu titik berdasarkan daftar urutan, dan pemangkasan semua titik yang berada dalam lingkaran dengan radius tertentu yang berbading terbalik dengan kepadatan pada titik tersebut. Sebuah metode non parametrik yang berfungsi memperkirakan probabilitas kepadatan dalam algoritma ini digunakan k-NN. Algoritma ini bekerja dengan langkah-langkah sebagai berikut:

1. Set himpunan $B_N = \{x_1, x_2, ..., x_N\}$ sebagai data set inputan. Pilih nilai integer positif k.
2. Untuk tiap titik $x_i \in B_N$, hitung jarak k^{th} nearest neighbor dari x_i pada B_N. Tandai dengan r_{k,x_i}.
3. Pilih titik $x_j \in B_N$ yang mempunyai nilai r_{k,x_j} terkecil dan letakkan dalam himpunan E.
4. Hapus semua titik dari B_N yang berada dalam lingkaran radius $2r_{k,x_j}$ yang berpusat di x_j dan titik-titik yang tersisa di set sebagai B_N.

Pseudocode 2.1 Algoritma ***Density Based Multiscale Data Consensation***

2.5 Algoritma Klasterisasi Hirarki

Metode klasterisasi hirarki merupakan salah satu metode yang terkenal dari banyak metode-metode klasterisasi. Metode-metode hirarki banyak dipakai karena kemudahannya dalam cara membentuk klaster dengan memperhatikan

unsur-unsur keserupaan antar data yang akan diklaster. Jika proses pembentukan klasternya dibentangkan, maka akan terlihat hirarki pembentukan klaster dari tahap demi tahap. Oleh karena itu, metode ini dinamakan metode hirarki. Metode-metode hirarki ada beberapa macam [Moo01], diantaranya adalah *Single Linkage, Complete Linkage, Centroid Linkage*, dan *Average Linkage*.

2.5.1 Algoritma *Single Linkage*

Metode ini merupakan metode yang paling mudah di antara metode-metode hirarki yang lain. Metode ini menggabungkan 2 klaster melalui jarak terdekat di antara anggota-anggota klaster. Metode ini sangat cocok untuk dipakai pada kasus *shape independent clustering*, karena kemampuannya untuk membentuk pola tertentu dari klaster. Untuk kasus *condensed clustering*, metode ini tidak bagus. P*seudocode* algoritma *Single linkage* selengkapnya adalah sebagai berikut:

1. Diasumsikan tiap titik n mewakili dirinya sendiri dengan menjadi klaster c_i, dimana $i = 1..n$.
2. Hitung jarak terdekat antara $m(c_r)$ dan $m(c_u)$, dimana $r \neq u$ dan $m(c_j)$ merupakan anggota klaster c_j.
3. Gabung c_r dan c_u menjadi satu klaster baru c_a dimana $m(c_a)$ adalah anggota gabungan dari c_r dan c_u.
4. Ulangi langkah 2 sampai mencapai kondisi optimum.

Pseudocode 2.2 Algoritma ***Single linkage***

2.5.2 Algoritma *Complete Linkage*

Metode ini merupakan kebalikan dari *Single Linkage*. Metode ini menggabungkan 2 klaster melalui jarak terjauh di antara anggota-anggota klaster. Metode ini baik untuk kasus klasterisasi dengan normal data set distribution. Akan tetapi, metode ini tidak cocok untuk data yang mengandung outlier. *Pseudocode* algoritma *Complete linkage* selengkapnya adalah sebagai berikut:

```
1. Diasumsikan tiap titik n mewakili dirinya sendiri dengan
   menjadi klaster c_i, dimana i = 1..n.
2. Hitung jarak terjauh antara m(c_r) dan m(c_u), dimana r ≠ u
   dan m(c_j) merupakan anggota klaster c_j.
3. Gabung c_r dan c_u menjadi satu klaster baru c_a dimana m(c_a)
   adalah anggota gabungan dari c_r dan c_u.
4. Ulangi langkah 2 sampai mencapai kondisi optimum.
```

Pseudocode 2.3 Algoritma *complete linkage*

2.5.3 Algoritma *Centroid Linkage*

Algoritma *centroid linkage* merupakan algoritma klasterisasi hirarki. Penggabungan klaster pada centroid linkage didasarkan pada lokasi titik pusat yang terbentuk pada tahap sebelumnya. Metode ini dibangun dengan memperhatikan pengecilan nilai standar deviasi klaster sekecil-kecilnya. Metode ini menggabungkan dua klaster melalui jarak terdekat diantara titik pusat antar klaster. Metode ini sangat ampuh untuk memperkecil *variance within cluster* karena melibatkan titik pusat pada saat penggabungan antar klaster. Metode ini

juga baik untuk data yang mengandung outlier. Berikut ini adalah *pseudocode* algoritma *centroid linkage*:

1. Diasumsikan tiap titik n mewakili dirinya sendiri dengan menjadi klaster c_i, dimana $i = 1..n$.
2. Hitung jarak terdekat antara $m(c_r)$ dan $m(c_u)$, dimana $r \neq u$ dan $m(c_j)$ merupakan anggota klaster c_j.
3. Gabung c_r dan c_u menjadi satu klaster baru c_a dimana $m(c_a)$ adalah anggota gabungan dari c_r dan c_u.
4. Ulangi langkah 2 dengan $m(c_a)$ diwakili oleh titik pusat dari semua anggotanya.

Pseudocode 2.4 Algoritma *centroid linkage*

2.5.4 Algoritma *Average Linkage*

Metode ini menggabungkan 2 klaster melalui rata-rata jarak terdekat di antara masing-masing anggota antar klaster. Metode ini relatif yang terbaik dari metode-metode hirarki. Namun, ini harus dibayar dengan waktu komputasi yang paling tinggi dibandingkan dengan metode-metode hirarki yang lain. Berikut ini adalah *pseudocode* algoritma *average linkage*:

1. Diasumsikan tiap titik n mewakili dirinya sendiri dengan menjadi klaster c_i, dimana $i = 1..n$.
2. Hitung jarak terdekat antara $m(c_r)$ dan $m(c_u)$, dimana $r \neq u$

Pseudocode 2.5 Algoritma *average linkage*

dan $m(c_j)$ merupakan anggota klaster c_j.

3. Gabung c_r dan c_u menjadi satu klaster baru c_a dimana $m(c_a)$ adalah anggota gabungan dari c_r dan c_u.

4. Ulangi langkah 2 dengan $m(c_a)$ diwakili oleh rata-rata jarak terdekat di antara masing-masing anggota antar klaster.

Pseudocode 2.5 Algoritma *average linkage* (lanjutan)

Selain keempat metode hirarki diatas, masih ada lagi beberapa metode yang lain, seperti *Ward, Weighted*, dan *Median Linkage*. Dari keempat metode klasterisasi diatas *Single Linkage* membutuhkan waktu komputasi paling cepat, diikuti oleh *Complete Linkage, Centroid Linkage* dan yang paling lambat adalah *Average Linkage*. Tetapi dari segi kepresisian hasil, terbaik adalah *Average Linkage*, diikuti oleh *Centroid Linkage*, *Complete Linkage* dan yang terakhir adalah *Single Linkage*.

2.6 Algoritma Klasterisasi Partisi

Dalam bagian ini dijelaskan konsep dan teori dasar tentang algoritma k-modes sebagai algoritma klasterisasi partisi untuk mengklaster data kategorikal. Algoritma k-modes merupakan pengembangan algoritma k-means dengan sedikit modifikasi sehingga bisa digunakan untuk mengklaster data kategorikal.

Pada algoritma ini dilakukan tiga modifikasi terhadap algoritma k-means yaitu [Huang97a]:

- Menggunakan ukuran ketidakserupaan yang berbeda.

- K-means diganti dengan k-modes.
- Menggunakan *frequency based method* untuk meng-update modes

2.6.1. Ukuran Keserupaan

Bila diasumsikan X, Y adalah dua objek kategorikal yang dideskripsikan oleh m atribut kategorikal. Ukuran ketidakserupaan antara X dan Y dapat didefinisikan sebagai total ketidakcocokan pada atribut kategori yang berkorespondensi pada dua objek. Semakin kecil jumlah ketidakserupaan, semakin mirip dua objek tersebut. Berikut ini adalah rumus ukuran keserupaan

$$d(X,Y) = \sum_{j=1}^{m} \delta(x_j, y_j) \tag{2.7}$$

dimana

$$\delta(x_j, y_j) = \begin{cases} 0 & x_j = y_j \\ 1 & x_j \neq y_j \end{cases} \tag{2.8}$$

Dalam persamaan (2.7) $d(X,Y)$ menggambarkan keserupaan untuk tiap kategori atribut. Jika dibawa ke basis frekuensi dari kategori dalam data set, ukuran ketidakserupaan dapat dirumuskan sebagai

$$d_{x^2}(X,Y) = \sum_{j=1}^{m} \frac{n_{x_j} + n_{y_j}}{n_{x_j} n_{y_j}} \delta(x_j, y_j) \tag{2.9}$$

dimana n_{x_j}, n_{y_j} adalah jumlah objek dalam data set yang mempunyai kategori x_j dan y_j untuk atribut j. Karena $d_{x^2}(X,Y)$ mirip dengan jarak chi-square dalam, maka $d_{x^2}(X,Y)$ bisa kita sebut sebagai jarak chi-square. Ukuran ketidakserupaan

ini digunakan jika frekuensi kemunculan suatu kelas sangat sedikit bila dibandingkan dengan frekuensi kemunculan kelas yang lainnya. Persamaan (2.9) sangat berguna bila digunakan untuk mencari *under-represented object clusters* seperti *fraudulent claims* dalam database asuransi.

2.6.2. Himpunan Mode

Bila dianggap X adalah himpunan objek kategorikal yang dideskripsikan oleh atribut kategorikal A1, A2, ..., Am. Maka bisa didefinisikan bahwa mode dari X adalah vektor Q = [q1, q2, ..., qm] $\in \Omega$ yang meminimalkan

$$D(Q,X) = \sum_{i=1}^{n} d(X_i, Q) \tag{2.10}$$

Dimana X = {X1, X2, ..., Xn} dan d bisa merupakan (2.7) atau persamaan (2.9). Disini, Q tidak harus elemen X.

2.6.3. Menemukan Himpunan Mode

Bila diasumsikan $n_{c_{k,j}}$ adalah jumlah objek yang mempunyai kategori $c_{k,j}$ pada atribut A_j dan $fr(A_j = c_{k,j} \mid X) = \frac{n_{c_{k,j}}}{n}$ adalah frekuensi relative kategori $c_{k,j}$ dalam X. Selanjutnya terdapat teori bahwa fungsi $D(Q,X)$ akan minimal jika $fr(A_j = q_j \mid X) \geq fr(A_j = c_{k,j} \mid X)$ untuk $q_j \neq c_{k,j}$ untuk semua j = 1...m.

2.6.4. Algoritma K-Modes

Anggap {S1, S2, ..., Sk} adalah pembagian dari X, dimana $S_l \neq \phi$ untuk $1 \leq l \leq k$, dan {Q1,Q2,...,Qk} adalah modes untuk {S1, S2, ..., Sk}. Total biaya untuk membagi didefinisikan dengan

$$P(W,Q) = \sum_{l=1}^{k} \sum_{i=1}^{n} \sum_{j}^{m} w_{i,l} d(X_i, Q_l) \qquad (2.11)$$

dimana w adalah matrik mxk dari $Q = \{Q_1, Q_2, ..., Q_k\}$ dan d bisa merupakan persamaan (2.7) atau persamaan (2.9). Mirip dengan algoritma k-means, tujuan utama mengklaster X adalah untuk menemukan himpunan *{Q1, Q2, ..., Qk}* yang dapat meminimalkan P. Persamaan (2.11) dapat diminimalkan dengan algoritma k-modes. Algoritma k-modes terdiri dari beberapa langkah [Huang97a]:

1. Pilih k inisial modes, satu untuk tiap klaster.
2. Alokasikan suatu objek ke klaster yang mempunyai mode terdekat berdasarkan d. Selanjutnya update mode klaster tersebut setiap kali terjadi alokasi.
3. Setelah tiap objek sudah dialokasikan ke klaster, cek kembali ketidakserupaan objek dengan mode saat ini. Jika ternyata mode terdekat objek tersebut milik klaster lain maka realokasikan objek tersebut ke klaster lain tersebut dan update kembali mode kedua klaster yang bersangkutan tadi.
4. Ulangi langkah 3 sampai tidak ada objek yang berubah letak klasternya.

Pseudocode 2.6 Algoritma K-Modes

Seperti algoritma k-means, algoritma k-modes juga menghasilkan lokal optima tergantung pada inisial modes dan urutan objek dalam data set.

Pada implementasi dengan menggunakan algoritma k-modes digunakan dua macam metode pemilihan mode awal. Metode pertama memilih *k* buah record yang berbeda yang muncul pertama kali dari data set sebagai inisial *k* modes. Metode kedua adalah dengan menggunakan matrik frekuensi yang diimplementasikan dengan cara berikut:

a. Hitung frekuensi dari semua kategori untuk semua atribut dan simpan dalam aray kategori dengan urutan frekuensi secara menurun sebagaimana ditunjukkan pada gambar 2.6. Disini, c_{ij} menyatakan kategori *i* dari atribut *j* dan $f(ci,j) \geq f(ci+1,j)$ dimana $f(ci,j)$ adalah frekuensi dari kategori c_{ij}.

$$\begin{Bmatrix} c_{1,1} & c_{1,2} & c_{1,3} & c_{1,4} \\ c_{2,1} & c_{2,2} & c_{2,3} & c_{2,4} \\ c_{3,1} & & c_{3,3} & c_{3,4} \\ c_{4,1} & & c_{4,3} & \\ & & c_{5,3} & \end{Bmatrix}$$

Gambar 2.6. Kategori aray data set dengan 4 atribut yang masing-masing mempunyai 4,2,5,3 kategori.

b. Menentukan *k* initial modes berdasarkan larik pada gambar 2.6:

Contoh: nilai $k = 3$

$Q1 = [q1,1=c1,1, q1,2=c2,2, q1,3=c3,3, q1,4=c1,4]$

$Q2 = [q2,1=c2,1, q2,2=c1,2, q2,3=c4,3, q2,4=c2,4]$

c. Q3 = [q3,1=c3,1, q3,2=c2,2, q3,3=c1,3, q3,4=c3,4]

d. Dimulai dari Q1, pilih record yang paling mirip dengan *Q1* dan subtitusikan *Q1* dengan record tersebut sebagai inisial mode yang pertama. Lakukan proses ini sampai pada *Qk*. Pada pemilihan ini $Ql \neq Qt$ untuk $l \neq t$.

Langkah 3 dilakukan untuk menghindari timbulnya klaster kosong. Tujuan dari pemilihan metode ini adalah supaya inisial mode berbeda-beda, sehingga bisa menghasilkan hasil klaster yang baik.

2.7 Penelitian yang Terkait

Akhir-akhir ini terdapat beberapa penelitian dibidang klasterisasi data kategorikal. Salah satu diantaranya adalah STIRR [Gib98]. STIRR merupakan algoritma klasterisasi yang digunakan untuk mengklaster transaksi konsumen dalam *market database*. STIRR merupakan algoritma iteratif yang didasarkan pada sistem non-linear dinamis. Pendekatan yang digunakan dapat diaplikasikan pada beberapa tipe sistem non-linear. Jika sistem dinamis konvergen, maka data kategorikal bisa diklaster. Tetapi terdapat penelitian lain yang menyebutkan bahwa sistem dinamis tidak dijamin akan mencapai kondisi konvergen, dan mengajukan sistem dinamis yang diperbaiki sehingga kondisi konvergen bisa dicapai.

Pengklasteran dengan algoritma K-Modes [Huang97a] memberikan hasil yang memuaskan. Hasil pengklasteran mencapai kondisi yang konvergen, tetapi tidak optimal [Sun02]. Pengklasteran dengan metode K-Modes dipengaruhi oleh nilai pembangkitan awal pada pusat klasternya sehingga dapat menghasilkan klaster yang berbeda tergantung pada pemilihan titik pusat awal. Dari penelitian

yang telah dilakukan, juga menunjukkan kondisi adanya hasil klaster yang tidak memiliki anggota [Fun04], sehingga mengakibatkan jumlah jarak dalam kelompok menjadi sangat besar. Hal ini menyebabkan solusi yang diperoleh tidak global, tetapi terjebak pada lokal minimum.

Beberapa peneliti, [San04] dan [Sun02], telah mencoba mengembangkan algoritma K-Modes pada penentuan titik pusat awal. San menentukan titik pusat awal dengan memanfaatkan himpunan fuzzy. Sun menentukan titik pusat awal dengan cara melakukan langkah iteratif terhadap algoritma K-Modes yang diaplikasikan pada beberapa data sub-sampel, setelah titik pusat awal dihasilkan selanjutnya algoritma K-Modes dijalankan sekali lagi terhadap data set keseluruhan. Karena harus menjalankan algoritma k-modes secara iteratif maka algoritma yang dikembangkan oleh Sun membutuhkan waktu proses yang lebih lama daripada algoritma K-Modes asli, juga membutuhkan media penyimpanan yang lebih karena harus membentuk dan menyimpan beberapa sub-sampel data. Pada algoritma yang dikembangkan oleh [Sun02] dan [San04] masih terdapat kelemahan yaitu memiliki nilai variasi dalam satu klaster yang relatif tinggi karena tidak mempertimbangkan modus nilai yang ada pada klaster yang sudah terbentuk pada tahap sebelumnya. Penelitian lain yang dilakukan oleh [He05] mencoba untuk mengembangkan algoritma K-Modes dengan tujuan menghasilkan klaster yang lebih presisi dengan cara mengaplikasikan rumus perhitungan jarak yang mempertimbangkan nilai modus dalam klaster yang sudah terbentuk pada tahap sebelumnya. Tetapi dalam penelitian yang dilakukan oleh [He05] masih

menggunakan metode random, sehingga masih mungkin mendapatkan hasil klaster yang jelek.

BAB 3

PERBAIKAN PENENTUAN TITIK PUSAT AWAL BERBASIS HIRARKI

BAB 3

PERBAIKAN PENENTUAN TITIK PUSAT AWAL BERBASIS HIRARKI

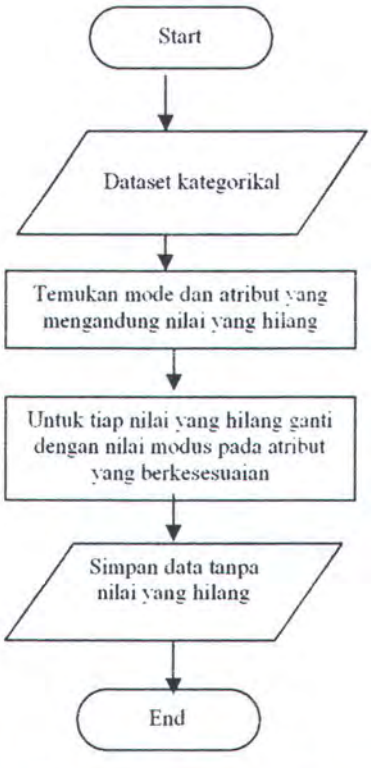
Bab 3 ini membahas tentang algoritma perbaikan titik pusat awal berbasis hirarki untuk mengklaster data kategorikal. Bagian awal bab ini menjelaskan tentang praproses data pembentukan data sub-sampel, optimasi titik pusat awal, titik pusat konseptual, obyek penelitian, praproses data, proses klasterisasi data kategorikal dan diakhiri dengan pengukuran kompleksitas algoritma.

3.1 Praproses Data

Tidak semua data bisa langsung dipakai sebagai data masukan. Semua data atau objek yang dijadikan sebagai data masukan sistem perbaikan titik pusat awal berbasis hirarki ini harus direpresentasikan dalam domain kategorikal. Sehingga, data yang mengandung bilangan integer atau *real* yang merupakan bilangan numerik tidak bisa digunakan sebagai data masukan.

Data masukan sistem ini dapat mempunyai jumlah atribut yang besar, untuk menyingkat penulisan nilai atribut datanya, nilai atribut bisa dikonversikan dalam bentuk nilai numerik integer. Meskipun dikonversikan dalam bentuk numerik integer, dalam pembentukan klaster, data hasil konversi ini akan tetap diperlakukan sebagai data kategorikal.

Data masukan harus mempunyai atribut yang lengkap, jika terdapat nilai atribut yang hilang, maka proses yang dilakukan adalah mengganti nilai yang hilang tersebut dengan nilai modus atau nilai yang sering muncul pada atribut dan pada klaster tersebut. Sebelum diproses, data masukan harus melalui proses penggantian nilai atribut yang hilang. Dalam kaitan ini, tahap yang harus dilalui pada proses ini adalah mencari nilai modus dari atribut yang hilang tersebut. Selanjutnya gantilah nilai yang hilang dengan nilai atribut modus yang sudah ditemukan. Langkah-langkah praproses data bisa dilihat pada gambar 3.1.



Gambar 3.1 Flowchart praproses data

3.2 Pembentukan Data Sub-sampel

Setelah data masukan siap diproses tahap selanjutnya yang harus dilakukan dalam penelitian ini adalah pembentukan data sub-sampel. Pembentukan data sub-sampel dilakukan dengan cara mereduksi data. Untuk mereduksi data digunakan algoritma *Density Based Multiscale Data Consensation* (DBMDC) [Mit02]. Prinsip dasar algoritma ini adalah mengurutkan titik-titik berdasarkan estima*ted densities*, memilih titik-titik yang padat, dan menghapus titik lain yang berada dalam jarak tertentu dari titik yang dipilih sebagai bagian dari sampel data. Metode non-parametrik dalam memperkirakan *probability density function* adalah metode *k-nearest-neighbor.*

Pada metode k-NN kepadatan titik dihitung berdasarkan area pada suatu lingkaran yang berisi sejumlah k titik yang betetanggaan. Sedemikian hingga area yang padat mempunyai radius lingkaran yang kecil dari pada area yang jarang. Area dalam lingkaran berbading terbalik dengan probability density function pada titik pusat lingkaran. Algoritma DBMDC selengkapnya adalah sebagai berikut.

1. Set $B_N = \{x_1, x_2, \ldots, x_n\}$ sebagai data set inputan. Pilih nilai integer positif k_{red}.
2. Untuk tiap titik $x_i \in B_N$, hitung jarak k^{th} nearest neighbor dari x_i pada B_N. Tandai dengan r_{k,x_i}.

Pseudocode 3.1. Algoritma *Density Based Multiscale Data Condensation*

3. Pilih titik $x_j \in B_N$ yang mempunyai nilai r_{k,x_j} terkecil dan letakkan dalam himpunan E.
4. Hapus semua titik dari B_N yang berada dalam lingkaran radius $2r_{k,x_j}$ yang berpusat di x_j .
5. Ulangi langkah 3 pada B_N sampai B_N menjadi himpunan kosong.

Pseudocode 3.1. Algoritma *Density Based Multiscale Data Condensation*

(lanjutan)

Cara pengukuran jarak untuk mengukur keserupaan antara dua obyek atau titik untuk data kategorikal yang digunakan pada algoritma DBMDC adalah ditunjukkan pada persamaan (3.1) dan (3.2), anggap X, Y adalah dua obyek kategorikal yang dideskripsikan oleh m atribut kategorikal. Ukuran ketidakserupaan antara X dan Y dapat didefinisikan sebagai total ketidakcocokan pada atribut kategori yang berkorespondensi pada dua obyek. Semakin kecil jumlah ketidakserupaan, semakin mirip dua obyek tersebut.

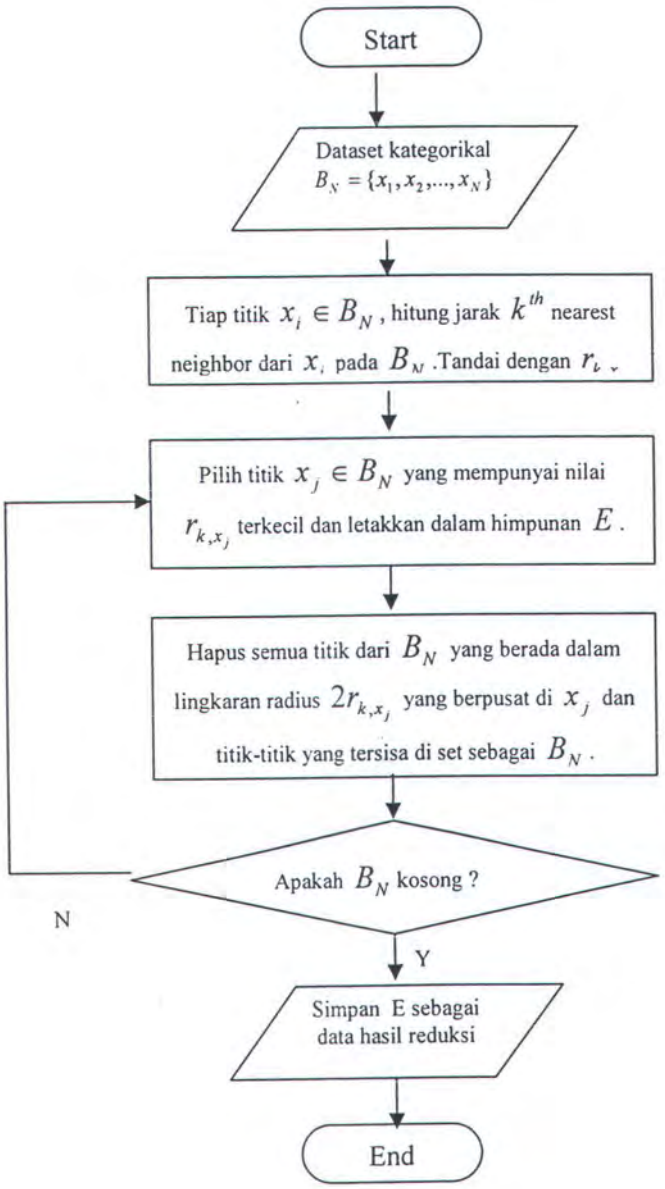
$$d_1(X,Y) = \sum_{j=1}^{m} \delta(x_j, y_j) \tag{3.1}$$

dimana

$$\delta(x_j, y_j) = \begin{cases} 0 & x_j = y_j \\ 1 & x_j \neq y_j \end{cases} \tag{3.2}$$

$d_1(X,Y)$ menggambarkan keserupaan untuk tiap kategori atribut.

Langkah-langkah pembentukan data sub-sampel bisa dilihat pada gambar 3.2.



Gambar 3.2 Flowchart pembentukan data sub-sampel

Kompleksitas komputasi merupakan hal yang dipertimbangkan dalam memilih suatu algoritma. Pada algoritma reduksi data ini, algoritma dapat dibagi menjadi tiga tahap. Tahap pertama adalah untuk tiap titik dihitung k^{th} nearest neighbor. Tahap kedua, titik yang mempunyai nilai terdekat dipilih, dan pada tahap ketiga semua titik yang berada pada radius $2r_{k,x_j}$ dihapus. Tampak bahwa pada tahap kedua dan ketiga kecepatan semakin meningkat karena ukuran data semakin berkurang tergantung pada nilai k reduksi dan distribusi data. Tahap pertama adalah tahap yang paling banyak membutuhkan waktu komputasi yaitu sebesar $(O(k_{red}n^2))$, dimana n adalah jumlah data.

3.3 Optimasi Titik Pusat Awal

Tahap kedua adalah menentukan beberapa titik pusat awal yang baik dari data yang telah direduksi dengan menggunakan algoritma klasterisasi hirarki *aglomerative*. Untuk menetukan titik pusat awal, dirancang suatu algoritma klasterisasi hirarki yang bekerja mengikuti paradigma algoritma *centroid linkage*. Berikut ini adalah pseudocode algoritma penentuan titik pusat awal berbasis hirarki:

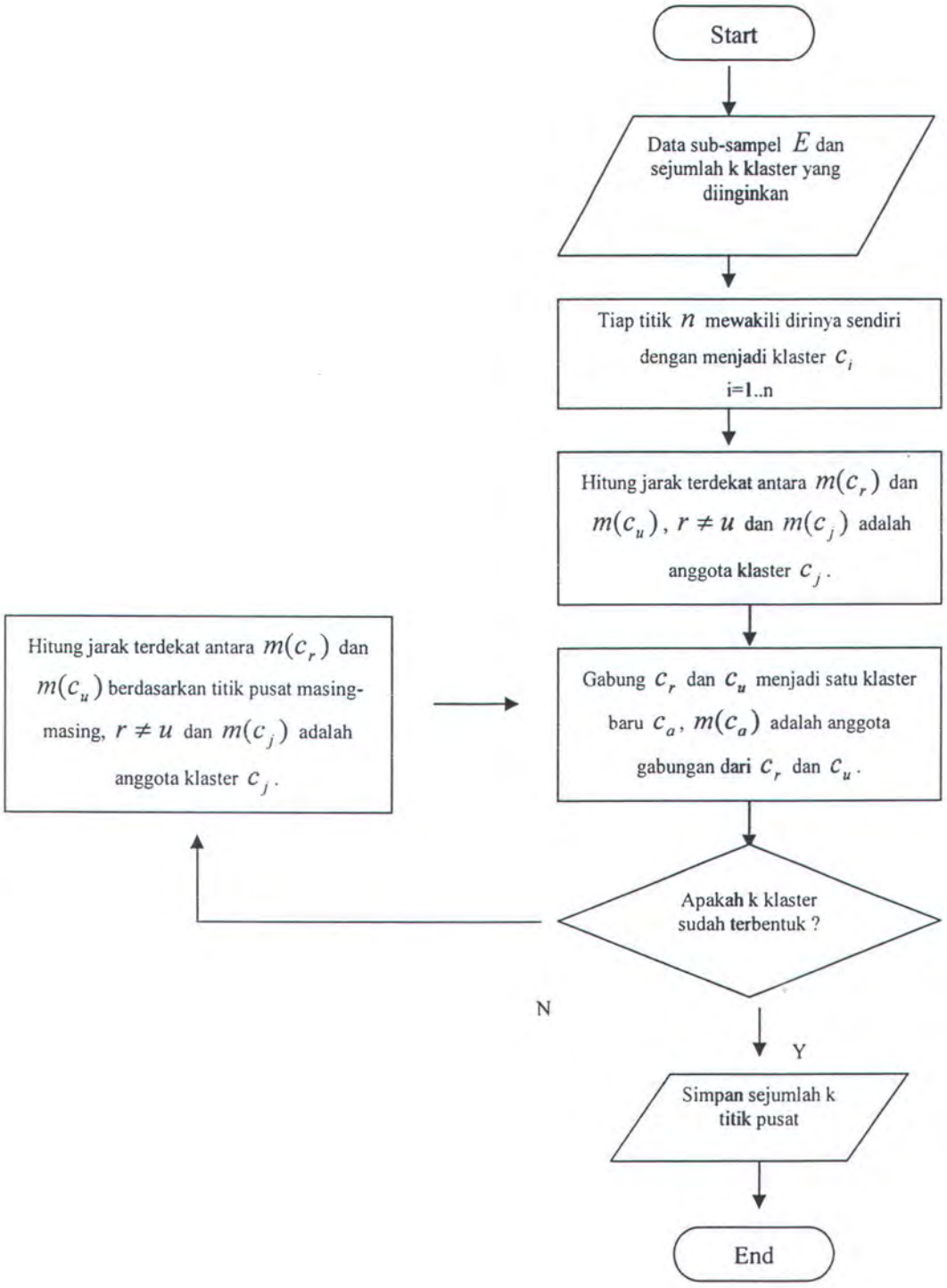
1. Diasumsikan tiap titik n_{red} mewakili dirinya sendiri dengan menjadi klaster c_i, dimana $i = 1..n_{red}$.
2. Hitung jarak terdekat antara $m(c_r)$ dan $m(c_u)$, dimana $r \neq u$ dan $m(c_j)$ merupakan anggota klaster c_j.

Pseudocode 3.2. Algoritma Penentuan Titik Pusat Awal

3. Gabung c_r dan c_u menjadi satu klaster baru c_a dimana $m(c_a)$ adalah anggota gabungan dari c_r dan c_u.
4. Ulangi langkah 2 dengan $m(c_a)$ diwakili oleh titik pusat dari semua anggotanya.

Pseudocode 3.2. Algoritma Penentuan Titik Pusat Awal (lanjutan)

Dengan algoritma seperti diatas maka proses utama dan yang paling sering dilakukan adalah mengukur jarak antara suatu klaster dengan klaster yang lain. Pada tahap awal jarak yang harus diukur melibatkan kombinasi sejumlah n_{red} titik yang ada dalam data masukan. Dan pada tahap selanjutnya proses pengukuran jarak akan berangsur-angsur berkurang karena pada tiap tahap terjadi proses penggabungan klaster Dengan uraian tersebut maka kompleksitas komputasionalnya adalah $O(n_{red}{}^2) + O(n^{1+2k_{klast}})$ dimana k_{klast} adalah jumlah titik pusat awal yang harus dihasilkan. Untuk menghitung jarak digunakan persamaan (3.1) dan (3.2). Langkah-langkah optimasi titik pusat awal bisa dilihat pada gambar 3.3



Gambar 3.3. Flowchart penentuan titik pusat awal

3.4 Klasterisasi Data Kategorikal

Tahap terakhir adalah proses pembentukan klaster dengan masukan dataset seluruhnya dan beberapa titik pusat awal hasil optimasi. Proses pembetukan klaster menggunakan algoritma klasterisasi partisi ditunjukkan secara lengkap seperti berikut:

1. Sejumlah k modes hasil klasterisasi hirarki gunakan sebagai k inisial modes, satu untuk tiap klaster.
2. Alokasikan suatu obyek ke klaster yang mempunyai mode terdekat berdasarkan persamaan (3.1). Selanjutnya update mode klaster tersebut setiap kali terjadi alokasi.
3. Setelah tiap obyek sudah dialokasikan ke klaster, cek kembali ketidakserupaan obyek dengan mode saat ini dengan menggunakan persamaan (3.3). Jika ternyata mode terdekat obyek tersebut milik klaster lain maka realokasikan obyek tersebut ke klaster lain tersebut dan update kembali mode kedua klaster yang bersangkutan tadi.
4. Ulangi langkah 3 sampai tidak ada obyek yang berubah letak klasternya.

Pseudocode 3.3 Algoritma klasterisasi partisional

Seperti ditunjukkan pada pseudocode 3.3, pada intinya ketika iterasi pertama digunakan perhitungan jarak (3.1) dan untuk iterasi selanjutnya, pada saat klaster sudah terbentuk pada iterasi berikutnya, digunakan perhitungan jarak pada persamaan (3.3) sebagai berikut:

$$d_2(X_i, Q_i) = \sum_{j=i}^{m} \phi(x_{i,j}, q_{l,j}) \qquad (3.3)$$

dimana

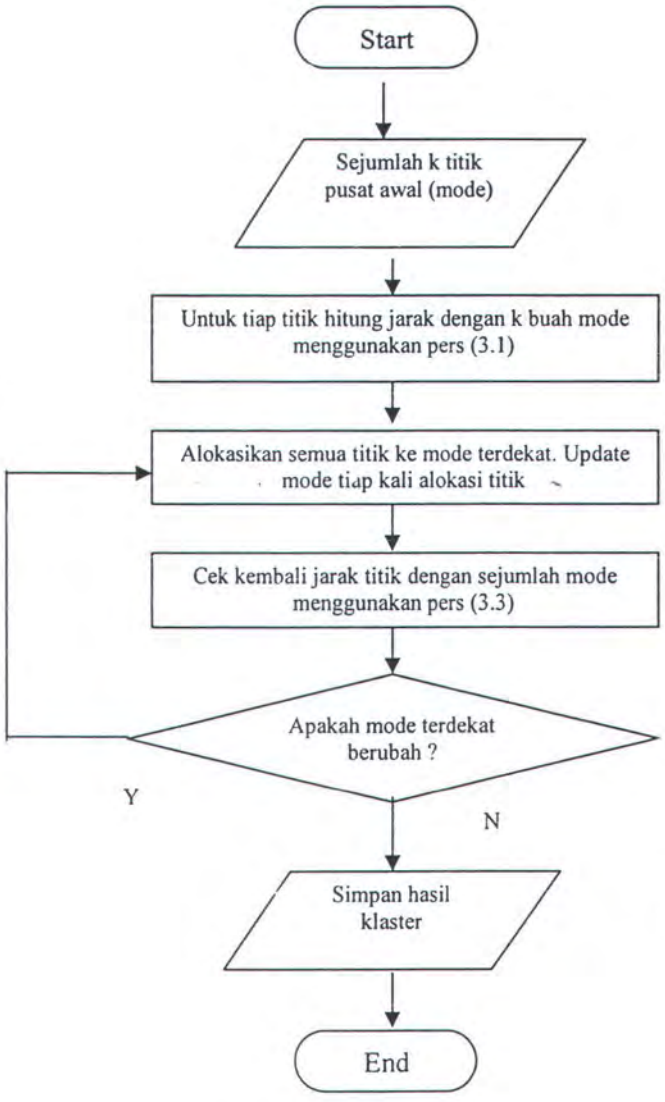
$$\phi(x_{i,j}, q_{l,j}) = \begin{cases} 1 - fr(A_j) = q_{l,j} \mid X_l & (x_{i,j} = q_{l,j}) \\ 1 & (x_{i,j} \neq q_{l,j}) \end{cases} \quad (3.4)$$

Pada persamaan (3.4), $fr(A_j) = q_{l,j} \mid X_l$ menyatakan frekuensi $q_{l,j}$ pada klaster X_l. Proses pembentukan klaster secara hirarki bisa dilihat pada gambar 3.4.

Total biaya untuk membentuk klaster setelah titik pusat awal didapatkan adalah sebagai berikut:

$$E = \sum_{l=1}^{k} \sum_{i=1}^{n} y_{i,j} d(X_i, Q_l) \quad (3.5)$$

dengan $y_{i,l}$ adalah elemen partisi dari matrik Y_{nxl} dan d bisa persamaan (3.1) atau (3.3). Waktu komputasi algoritma ini adalah $O(T * k_{klast} * n)$. Dengan T adalah jumlah iterasi yang dibutuhkan sampai mencapai kondisi konvergen, n jumlah data masukan, dan k_{klast} adalah jumlah klaster yang dibentuk. Dalam hal ini jumlah klaster yang dibentuk sama dengan jumlah titik pusat awal yang telah dihasilkan.



Gambar 3.4 Flowchart pembentukan klaster

3.5 Titik Pusat Konseptual

Baik data kategorikal maupun data numerik, dalam bidang klasterisasi tiap klaster pasti mempunyai titik pusat. Yang dimaksud dengan titik pusat konseptual klaster pada penelitian ini adalah suatu titik yang direpresentasikan dengan

sejumlah n atribut. Untuk data kategorikal maka titik pusat klaster dalam penelitian ini disebut dengan mode. Nilai-nilai sejumlah n atribut ditentukan sebagai modus, atau nilai yang sering muncul dalam atribut tersebut.

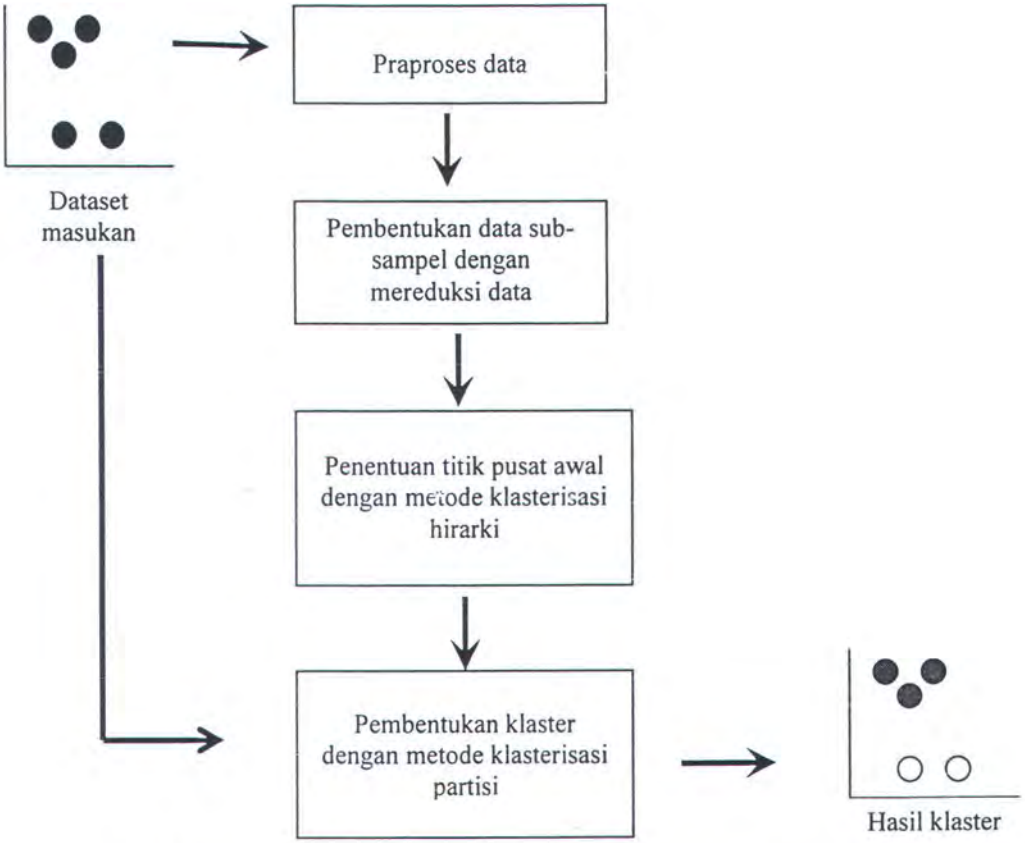
Sebagai ilustrasi, gambar 3.5 merupakan kumpulan objek dalam sebuah klaster. Objek pada klaster tersebut mempunyai tiga buah atribut yaitu A1, A2, dan A3, dan mempunyai empat objek anggota. Sekarang mode konseptual tersebut akan dihitung. Pertama kali yang dihitung adalah nilai yang mewakili atribut pertama yaitu A1. Atribut A1 memiliki dua buah nilai yaitu a dan b. Frekuensi kemunculan a sebanyak 3 dan b sebanyak 1. Oleh karena itu untuk atribut A1 mode diwakili oleh nilai a. Tahap kedua adalah menentuka mode atribut A2. Nilai pada atribut dua ada satu macam yaitu p yang muncul sebayak empat kali. Sehingga mode atribut A2 adalah p. Atribut terakhir yaitu atribut A3 memiliki tiga nilai atribut yaitu r yang muncul sebanyak dua kali, nilai t sebanyak satu kali dan s sebanyak satu kali. Sehingga mode atribut A3 adalah r karena frekuensi r muncul paling banyak. Sehingga mode konseptual klaster pada gambar 3.5 adalah {a, p, r}.

A1	A2	A3
a	p	r
b	p	r
a	p	t
a	p	s

Gambar 3.5 Representasi sebuah klaster

3.6 Desain Algoritma Penentuan Titik Pusat Awal Berbasis Hirarki

Dalam penelitian ini, sebelum melalu proses reduksi data, data masukan harus melalui praproses untuk memastikan data masukan adalah data kategorikal dan tidak mengandung nilai yang hilang. Tahap selanjutnya adalah membentuk data sub-sampel. Pembentukan data sub-sampel ini dilakukan dari proses mereduksi data. Yang menjadi alasan kenapa dataset masukan harus direduksi adalah supaya proses optimasi penentuan titik pusat awal tidak memproses dataset secara keseluruhan, tapi cukup menggunakan data sub-sampel dimana data sub-sampel ini sudah mewakili dataset secara keseluruhan. Bila dataset direduksi maka sama dengan mereduksi ruang pencarian titik pusat awal yang pada akhirnya akan mempercepat waktu komputasi optimasi penentuan titik pusat awal. Tahap selanjutnya adalah optimasi penentuan titik pusat awal dilakukan dengan menggunakan algoritma klasterisasi secara hirarki aglomeratif. Yang menjadi masukan bagi algoritma klasterisasi secara hirarki aglomeratif ini adalah data sub-sampel dari suatu dataset masukan. Hasil optimasi penentuan titik pusat awal akan menghasilkan beberapa titik pusat awal. Beberapa titik pusat awal yang dihasilkan ini dan dataset keseluruhan menjadi masukan bagi algoritma klasterisasi partisi sehingga akan dihasilkan beberapa klaster yang diinginkan. Desain algoritma secara keseluruhan sebagaimana ditunjukkan pada gambar 3.6.



Gambar 3.6. Desain algoritma perbaikan penentuan titik pusat awal berbasis hirarki

3.7 Kompleksitas Algoritma

Pada sub-bab sebelumnya telah diuraikan kompleksitas masing-masing algoritma. Karena algoritma penentuan titik pusat awal yang dibuat menjalankan proses pembentukan data sub-sampel, penentuan titik pusat awal, dan membentuk klaster secara sekuensial maka kompleksitas algortima bisa dihitung dengan cara menjumlahkan kompleksitas masing-masing algoritma tersebut.

Kompleksitas dari perangkat lunak perbaikan titik pusat awal berbasis hirarki dapat diketahui dengan mempelajari tahap-tahap pembentukan klaster

sebagai berikut. Untuk menghasilkan klaster, maka tahap-tahap yang harus dilalui adalah seperti berikut. Tahap pertama adalah pembentukan data sub-sample yang mempunyai kompleksitas waktu $(O(k_{red} n_{red}^{\ 2}))$. Tahap kedua adalah penentuan titik pusat awal dari sejumlah data sub-sample yang membutuhkan kompleksitas waktu sebesar $O(n_{red}^{\ 2}) + O(n^{1+2k_{red}})$. Dan tahap yang terakhir adalah pembentukan klaster yang membutuhkan kompleksitas waktu sebesar $O(T * k_{klast} * n)$. Dengan demikian, maka total waktu komputasi algoritma perbaikan penentuan titik pusat awal untuk klasterisasi data kategorikal adalah $(O(k_{red} n_{red}^{\ 2})) + O(n_{red}^{\ 2}) + O(n^{1+2k_{klast}}) + O(T * k_{klast} * n)$.

BAB 4

DESAIN DAN IMPLEMENTASI APLIKASI

BAB 4

DESAIN DAN IMPLEMENTASI APLIKASI

Dalam bab ini dijelaskan dua tahapan penting yaitu desain aplikasi dan tahap implementasi algoritma untuk algoritma perbaikan penentuan titik pusat berbasis hirarki. Untuk menjelaskan desain aplikasi akan digunakan penggambaran konsep *Unified Modelling Languange* (*UML*) yang dimiliki oleh aplikasi Rational Rose Enterprise Edition 2002, sedangkan tahap implementasi meliputi implementasi kelas dan implementasi antar-muka dari aplikasi.

4.1 Desain Aplikasi

Pada bagian ini dijelaskan desain aplikasi untuk algoritma perbaikan penentuan titik pusat berbasis hirarki yang meliputi tiga aspek penting yaitu desain *use case*, diagram kelas dan desain antar-muka. Untuk tahap desain aplikasi digunakan empat macam diagram yang mereprentasikan setiap proses dalam aplikasi. Diagram–diagram tersebut antara lain adalah :

- Diagram *use case*: untuk menjelaskan aktor yang ada dalam sistem dan kegiatan yang bisa dilakukan oleh aktor tersebut
- Diagram aktifitas (*activity diagram*): untuk menggambarkan urutan aktifitas atau proses yang ada dalam tiap kegiatan yang dilakukan oleh aktor sistem aplikasi ini.

- Diagram sekuensi (*sequence diagram*): untuk menggambarkan urutan kejadian antar objek yang berinteraksi dalam sistem aplikasi ini.
- Diagram kelas (*class diagram*): diagram kelas menggambarkan keseluruhan kelas dari aplikasi yang mengimplementasikan desain diagram *use case*.

4.1.1 Analisis *Use Case*

Suatu *use case* dapat berhubungan dengan *use case* yang lain. Hubungan antar *use case* menunjukan keterkaitan proses pada kedua *use case* tersebut. Secara umum, hubungan antara dua *use case* dapat dibedakan menjadi dua jenis, yaitu:

- Hubungan *include*

  Hubungan *include* terjadi bila suatu *use case* membutuhkan *use case* yang lainnya agar *use case* tersebut dapat berjalan.

- Hubungan *extend*

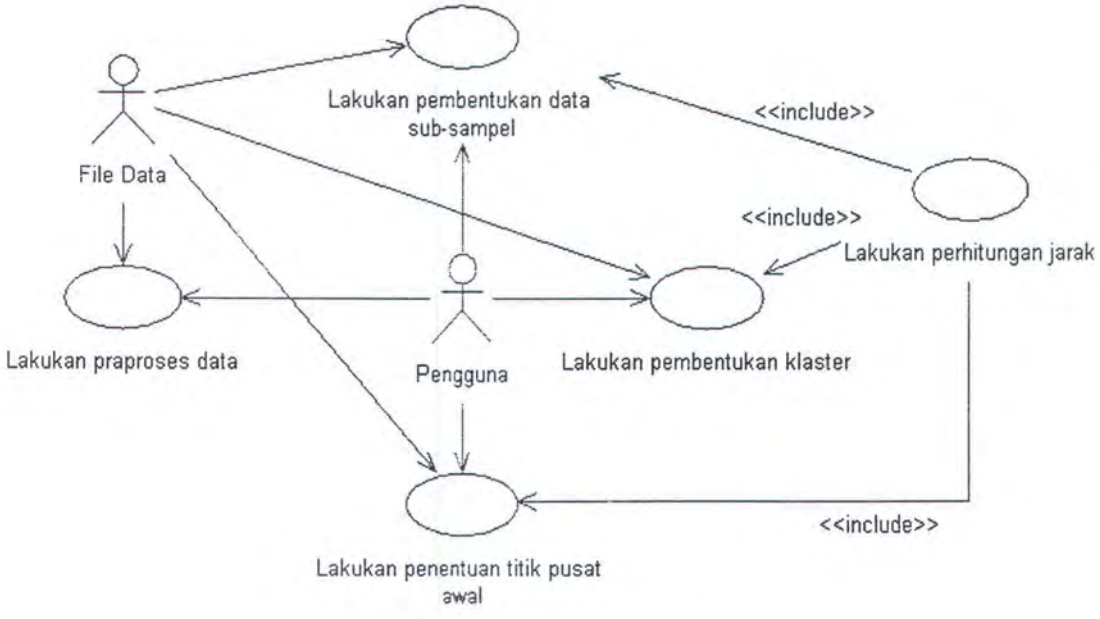
  Hubungan *extend* terjadi bila suatu *use case* dapat disertai dengan suatu *use case* lain atau tidak disertai dengan suatu *use case* lain tersebut.

Untuk membuat diagram *use case*, proses-proses yang terdapat pada aplikasi harus diidentifikasi terlebih dahulu. Detil dari keterangan proses-proses yang terdapat dalam aplikasi dijelaskan dalam tabel 4.1.

Tabel 4.1 Daftar proses

No	Nama proses	Keterangan
1	Lakukan praproses data	Proses ini merupakan proses menyiapkan data supaya siap menjadi masukan bagi proses klasterisasi.
2	Lakukan pembentukan data sub-sampel	Proses ini untuk membentuk data sub-sampel yang menjadi masukan proses penentuan sejumlah titik pusat awal.
3	Lakukan penentuan titik pusat awal	Proses ini mencari sejumlah titik pusat awal yang menjadi masukan bagi aplikasi klasterisasi.
4	Lakukan pembentukan klaster	Proses ini membentuk sejumlah klaster.

Berdasarkan proses–proses tersebut, maka dapat ditentukan diagram *use case* yang dapat ditunjukkan dalam gambar 4.1. Pada diagram *use case* tersebut, terdapat dua aktor yaitu aktor pengguna dan aktor sistem basis data, serta memiliki enam *use case* berdasarkan proses pada table 4.1



Gambar 4.1 Diagram Use Case

Pada diagram *use case* tersebut, terdapat satu aktor yaitu aktor pengguna. Aktor Pengguna merupakan pengguna yang memakai aplikasi untuk melakukan proses–proses seperti yang dijelaskan pada tabel 4.1. Untuk memperjelas detail desain proses setiap *use case* digunakan tabel dokumentasi naratif *use case*. Berikut ini merupakan detail proses setiap *use case* pada aplikasi yang diimplementasikan dalam tesis ini.

(a) Lakukan praproses data

Pada sub-bab ini dijelaskan mengenai proses yang terjadi didalam *use case* lakukan praproses data. Tabel dokumentasi naratif *use case* lakukan praproses data ditunjukkan pada tabel 4.2.

Tabel 4.2 Dokumentasi naratif *use case* lakukan praproses data

Nama *Use case*	lakukan praproses data	**Desain Sistem**
Pelaku Utama	- Aktor pengguna - Aktor file	
Pelaku Partisipan Lain	Tidak ada	
Deskripsi	*Use case* ini digunakan oleh aktor pengguna untuk melakukan penyiapan data supaya siap menjadi masukan bagi proses klasterisasi.	
Prakondisi	Aplikasi dapat dijalankan.	
Pemicu	*Use case* ini diawali saat aktor pengguna meminta praproses data	
Aliran Kejadian	**Kegiatan Aktor Pengguna**	**Respon Aplikasi**
Normal	**Langkah 1**: Aktor pengguna memilih menu untuk menam-pilkan antar-muka praproses data.	**Langkah 2**: Aplikasi merespon dengan menampilkan antar-muka praproses data.

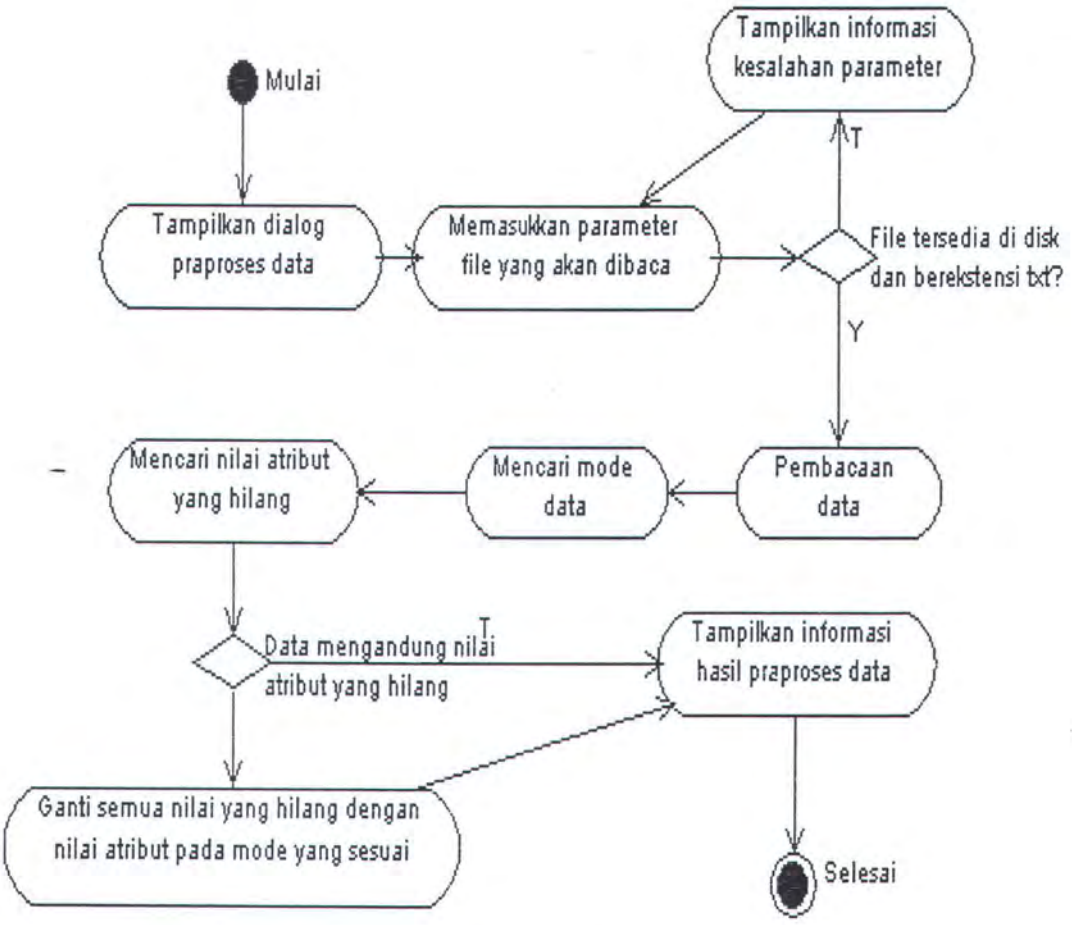
Tabel 4.2 Dokumentasi naratif *use case* lakukan praproses data (lanjutan)

	Langkah 3: Aktor pengguna memasukkan data file yang akan dibaca dan mengeksekusi praproses data..	**Langkah 4**: Aplikasi melakukan praproses data. **Langkah 5**: Aplikasi menampilkan status bahwa praproses data telah selesai.
		Sub Aliran Kejadian Normal
		Data dibaca dari file dan dilakukan praproses data..
Aliran Alternatif	**Alt-Langkah 4**: Jika parameter file yang akan dibaca salah, aplikasi menampilkan informasi kesalahan parameter	
	Alt-Langkah 5: Jika format file data salah, maka ditampilkan informasi kegagalan membaca file.	
Pascakondisi	Aplikasi menampilkan status bahwa praproses data telah selesai dilakukan	
Aturan *Use case*	Parameter yang digunakan untuk uji koneksi basis data adalah: (1) tipe basis data dan (2) nama pengguna dan (3) kata sandi pengguna dan (4) komputer *server* dan (5) *port* dan (6) nama basis data	
Batasan dan Spesifikasi Implementasi	1. *Use case* ini hanya dapat digunakan untuk file berekstensi txt. 2. Aplikasi hanya dapat melakukan koneksi untuk satu file data	
Asumsi	Terdapat file yang berisi data masukan.	

Pengguna dapat menjalankan proses ini dengan memilih menu "Praproses Data" kemudian aplikasi merespon dengan menampilkan dialog antar-muka praproses data, sehingga aktor pengguna dapat memasukkan parameter praproses data . Proses praproses data dimulai saat pengguna menentukan parameter koneksi terhadap file data. Parameter yang dimasukkan nama file yang akan dibaca.

Proses ini sangat penting karena memungkinkan aplikasi untuk mendapatkan data masukan berupa nama file data untuk proses penghilangan nilai yang hilang. Setiap informasi pembacaan file akan bergantung pada tipe file data yang ditentukan oleh pengguna. Aplikasi akan menyediakan informasi keberhasilan maupun kegagalan pembacaan file.

Pada proses selanjutnya, aplikasi secara otomatis akan melakukan pencarian mode data, mencari nilai yang hilang dan mengganti nilai yang hilang dengan mode pada tribut yang telah ditemukan. Diagram aktifitas lakukan praproses data seperti yang ditunjukkan pada gambar 4.2.



Gambar 4.2 Diagram aktifitas lakukan praproses data

(b) Lakukan pembentukan data sub-sampel

Pada sub-bab ini dijelaskan mengenai proses yang terjadi didalam *use case* lakukan pembentukan data sub-sampel. Tabel dokumentasi naratif *use case* lakukan pembentukan data sub-sampel ditunjukkan pada tabel 4.3.

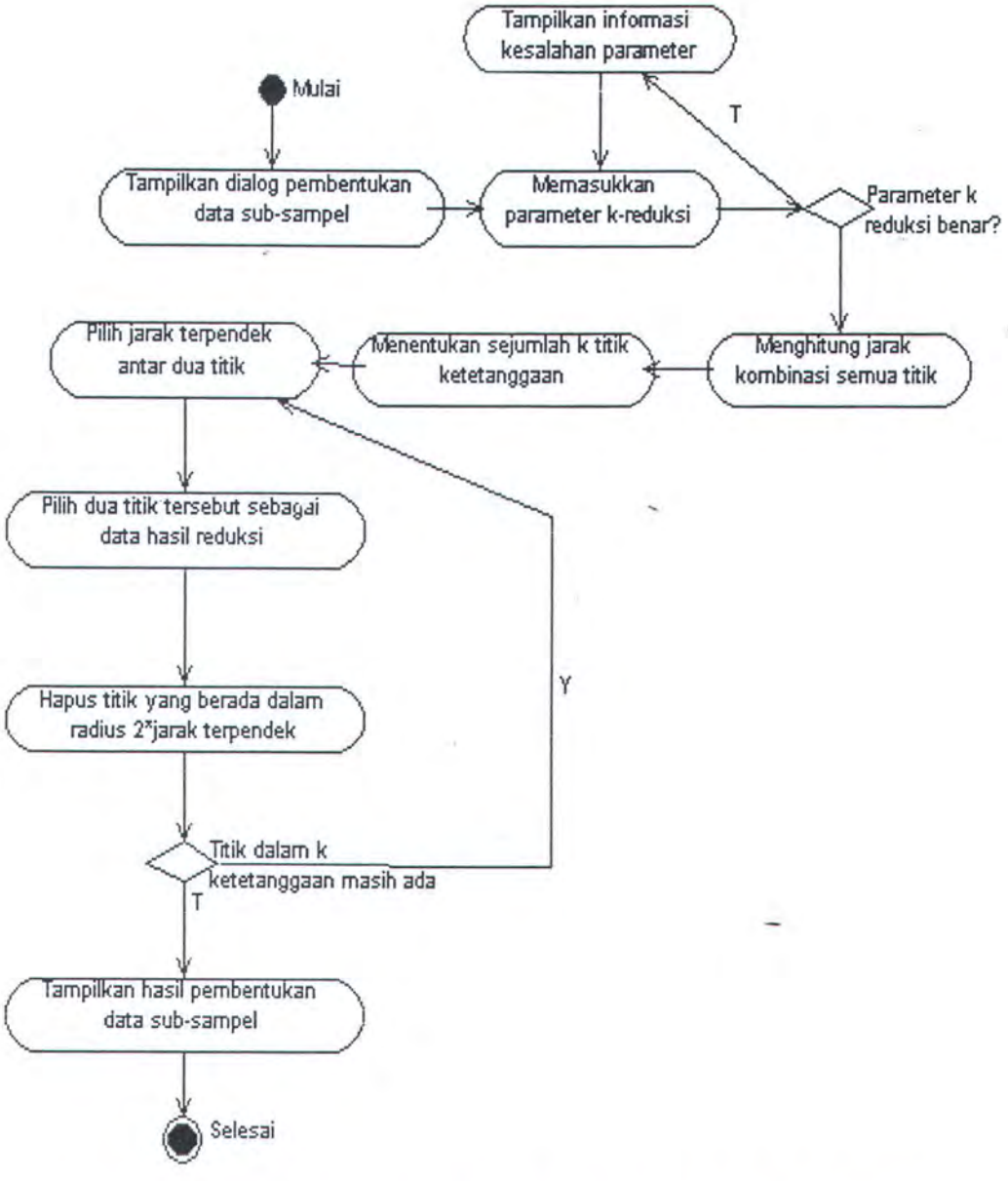
Tabel 4.3 Dokumentasi naratif *use case* lakukan pembentukan data sub-sampel

Nama *Use case*	lakukan pembentukan data sub-sampel	**Desain Sistem**
Pelaku Utama	- Aktor pengguna	
Pelaku Partisipan Lain	Tidak ada	
Deskripsi	*Use case* ini digunakan aktor pengguna untuk membentuk data sub-sampel sesuai dengan parameter yang ditentukan.	
Prakondisi	Data masukan tidak mengandung nilai atribut yang hilang.	
Pemicu	*Use case* ini diawali saat aktor pengguna meminta pembentukan data sub-sampel dari keseluruhan data masukan.	
Aliran Kejadian Normal	**Kegiatan Aktor Pengguna**	**Respon Aplikasi**
	Langkah 1: Aktor pengguna memilih menu untuk menampilkan antar-muka pembentukan data sub-sampel **Langkah 3:** Aktor pengguna memasukkan parameter dan eksekusi pembentukan data sub-sampel.	**Langkah 2**: Aplikasi merespon dengan menampilkan antar-muka pembentukan data sub-sampel. **Langkah 4**: Aplikasi melakukan validasi parameter pembentukan data sub-sampel **Langkah 5**: Eksekusi pembentuka.. data sub-sampel **Langkah 6**: Aplikasi menampilkan sejumlah data hasil reduksi.
Aliran Alternatif	**Alt-Langkah 4**: Jika parameter pembentukan data sub-sampel, aplikasi menampilkan informasi kesalahan parameter.	
Pascakondisi	Aplikasi menampilkan sejumlah data hasil reduksi yang menjadi data sub-sampel.	
Aturan *Use case*	Parameter yang digunakan pada pembentukan data sub-sampel adalah nilai k-reduksi data.	
Batasan dan Spesifikasi Implementasi	*Use case* ini dilakukan untuk data yang disimpan dengan atribut yang telah urut	
Asumsi	Data masukan telah dibaca dari file.	

Pengguna dapat menjalankan proses ini dengan memilih menu "Proses Pembentukan Data Sub-sampel". Kemudian aplikasi akan menampilkan dialog untuk proses pembentukan data sub-sampel. Disini pengguna terlebih dahulu mengisi parameter k-reduksi data. Pembentukan data sub-sampel dimulai dengan menghitung jarak k ketetanggaan, memilih titik dengan nilai jarak terpendek, menghapus titik pada jarak dua kali jarak terpendek yang telah ditemukan. Bila nilai k tetetanggan masih ada maka ulangi langkah memilih jarak terpendek, bila nilai k tetetanggan sudah habis maka tampilkan informa si jumlah data sub-sampel beserta titik-titik hasil reduksi dan proses selesai Untuk lebih jelasnya mengenai proses pada *use case* ini, maka diagram aktifitas lakukan pembentukan data sub-sampel ditunjukkan pada gambar 4.3.

(c) Lakukan penentuan titik pusat awal

Pada sub-bab ini dijelaskan mengenai proses yang terjadi didalam *use case* lakukan penentuan titik pusat awal. Tabel dokumentasi naratif *use case* lakukan penentuan titik pusat awal ditunjukkan pada tabel 4.4.



Gambar 4.3 Diagram aktifitas lakukan pembentukan data sub-sampel

Tabel 4.4 Dokumentasi naratif *use case* lakukan penentuan titik pusat awal

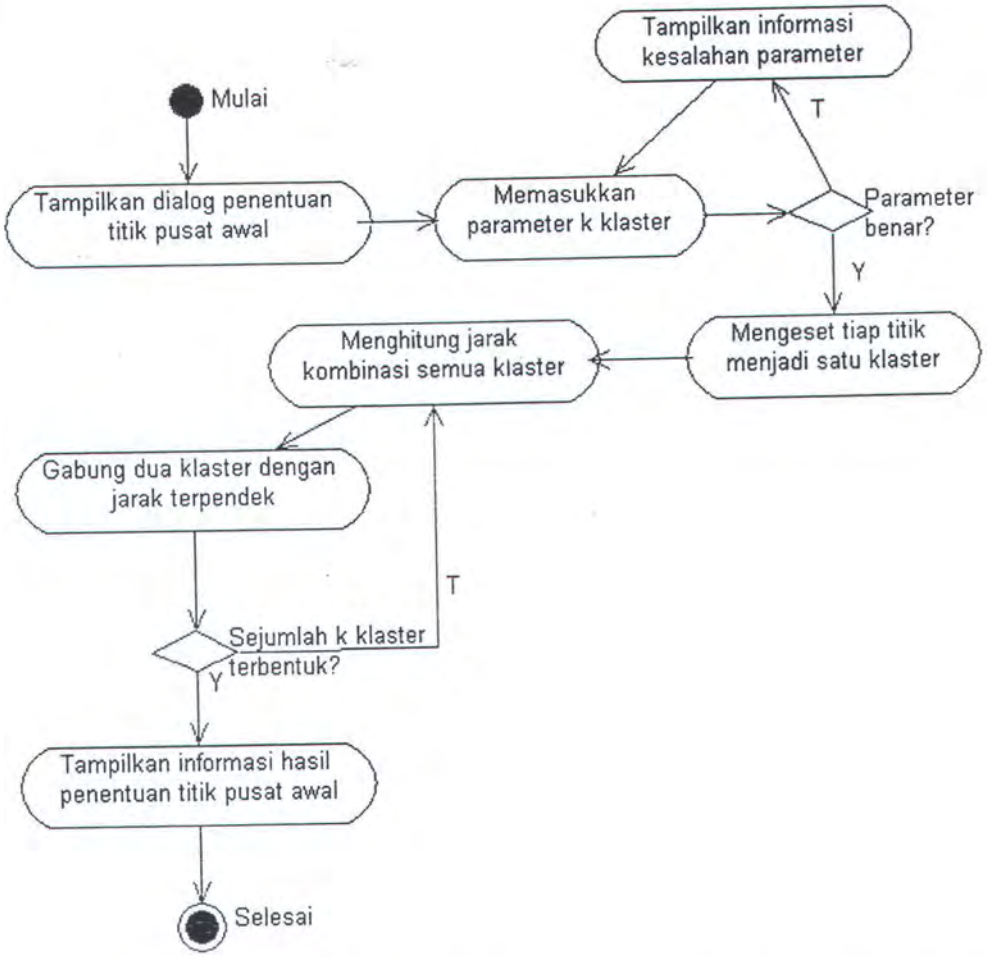
Nama *Use case*	lakukan penentuan titik pusat awal	**Desain Sistem**
Pelaku Utama	Aktor pengguna	
Pelaku Partisipan Lain	Tidak ada	
Deskripsi	*Use case* ini digunakan aktor pengguna untuk menentukan sejumlah titik pusat awal sesuai dengan parameter yang ditentukan.	
Prakondisi	Data sub-sampel telah terbentuk.	

Tabel 4.4 Dokumentasi naratif *use case* lakukan penentuan titik pusat awal (lanjutan)

Pemicu	*Use case* ini diawali saat aktor pengguna meminta penentuan titik pusat awal.	
Aliran Kejadian Normal	**Kegiatan Aktor Pengguna**	**Respon Aplikasi**
	Langkah 1: Aktor pengguna memilih menu untuk menam-pilkan antar-muka penentuan titik pusat awal. **Langkah 3:** Aktor pengguna memasukkan parameter dan eksekusi penentuan titik pusat awal.	**Langkah 2**: Aplikasi merespon dengan menampilkan antar-muka penentuan titik pusat awal. **Langkah 4**: Aplikasi melakukan validasi parameter penentuan titik pusat awal. **Langkah 5**: Eksekusi penentuan titik pusat awal. **Langkah 6**: Aplikasi menam-pilkan sejumlah titik pusat awal.
Aliran Alternatif	**Alt-Langkah 4**: Jika parameter penentuan titik pusat awal, aplikasi menampilkan informasi kesalahan parameter.	
Pascakondisi	Aplikasi menampilkan sejumlah titik pusat awal	
Aturan *Use case*	Parameter yang digunakan pada penentuan titik pusat awal adalah k-klaster yang harus dibentuk.	
Batasan dan Spesifikasi Implementasi	*Use case* ini dilakukan untuk data yang disimpan dengan atribut yang telah urut	
Asumsi	Data masukan telah dibaca dari file.	

Pengguna dapat menjalankan proses ini dengan memilih menu "Proses Penentuan Titik Pusat Awal". Kemudian aplikasi akan menampilkan dialog untuk proses penentuan titik pusat awal. Disini pengguna terlebih dahulu mengisi parameter k-klaster yang artinya akan dihasilkan k buah titik pusat awal. Penentuan titik pusat awal dimulai dengan mengeset satu titik adalah satu klaster kemudian menghitung jarak antar klaster, memilih dua kalster yang memiliki nilai jarak terpendek, gabung dua klaster terpendek tersebut. Bila sejumlah k klaster belum terbentuk maka ulangi langkah memilih jarak terpendek antar dua klaster, jika k klaster telah maka tampilkan informasi sejumlah titik pusat awal yang terbentuk dan proses selesai Untuk lebih jelasnya mengenai proses pada *use case*

ini, maka diagram aktifitas lakukan penentuan titik pusat awal ditunjukkan pada gambar 4.4.



Gambar 4.4 Diagram aktifitas lakukan penentuan titik pusat awal

(d) Lakukan Pembentukan Klaster

Pada sub-bab ini dijelaskan mengenai proses yang terjadi didalam *use case* lakukan pembentukan klaster. Tabel dokumentasi naratif *use case* lakukan pembentukan klaster ditunjukkan pada tabel 4.5.

Tabel 4.5 dokumentasi naratif *use case* lakukan pembentukan klaster

Nama *Use case*	lakukan pembentukan klaster	**Desain Sistem**
Pelaku Utama	Aktor pengguna	
Pelaku Partisipan Lain	Tidak ada	
Deskripsi	*Use case* ini digunakan aktor pengguna untuk melakukan pembentukan klaster.	
Prakondisi	- Sejumlah titik pusat awal telah terbentuk. - Data yang akan diklaster telah tesedia	
Pemicu	*Use case* ini diawali saat aktor pengguna meminta pembentukan klaster.	
Aliran Kejadian Normal	**Kegiatan Aktor Pengguna**	**Respon Aplikasi**
	Langkah 1: Aktor pengguna memilih menu untuk menam-pilkan antar-muka pembentukan klaster. **Langkah 3:** Aktor pengguna memasukkan parameter dan eksekusi pembentukan klaster.	**Langkah 2**: Aplikasi merespon dengan menampilkan antar-muka pembentukan klaster. **Langkah 4**: Aplikasi melakukan validasi parameter jumlah klaster yang akan dibentuk. **Langkah 5**: Eksekusi pembentukan klaster. **Langkah 6**: Aplikasi menam-pilkan sejumlah klaster yang telah terbentuk.
Aliran Alternatif	**Alt-Langkah 4**: Jika parameter jumlah klaster yang harus dibentuk salah, aplikasi menampilkan informasi kesalahan parameter.	
Pascakondisi	Aplikasi menampilkan sejumlah klaster yang terbentuk.	
Aturan *Use case*	Parameter yang digunakan pada pembentukan klaster adalah k-klaster yang harus dibentuk.	
Batasan dan Spesifikasi Implementasi	*Use case* ini dilakukan untuk data yang disimpan dengan atribut yang telah urut	
Asumsi	Data masukan telah dibaca dari file.	

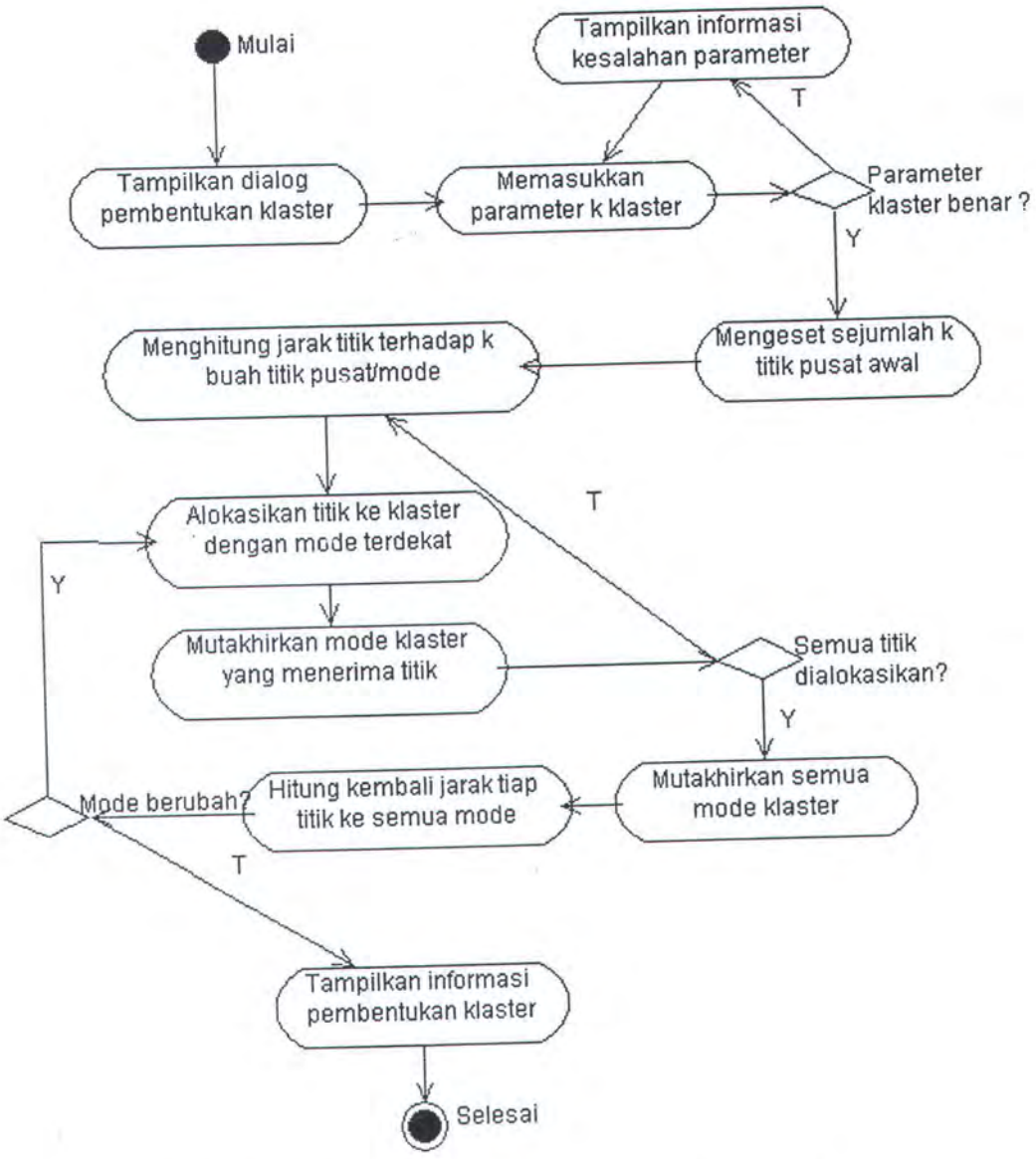
Pengguna dapat menjalankan proses ini dengan memilih menu "Proses Pembentukan Klaster". Kemudian aplikasi akan menampilkan dialog untuk proses pembentukan klaster. Disini pengguna terlebih dahulu mengisi parameter k-klaster yang artinya akan dihasilkan k buah klaster. Pembentukan klaster dimulai dengan

mengeset sejumlah k klaster menjadi titik pusat awal bagi k klaster yang harus dibentuk. Selanjutnya lakukan perhitungan jarak semua titik terhadap k buah titik pusat, alokasikan tiap titik ke titik pusat terdekat. Update titik pusat masing-masing klaster dan lakukan perhitungan jarak ulang, alokasikan kembali tiap titik ke titik pusat terdekat. Bila titik pusat sudah tidak berubah dan anggota klaster tidak berubah maka tampilkan informasi sejumlah sejumlah klaster yang terbentuk dan proses selesai Untuk lebih jelasnya mengenai proses pada *use case* ini, maka diagram aktifitas lakukan pembentukan klaster ditunjukkan pada gambar 4.5.

4.1.2 Realisasi *Use Case*

Pada sub-bab ini diuraikan detail urutan kejadian antar objek yang berinteraksi dalam setiap *use case* dengan menggunakan diagram sekuensi. Diagram sekuensi menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) dalam sebuah pesan (*message*) terhadap waktu. Diagram sekuensi terdiri antar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

Diagram sekuensi biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah kejadian (*event*) untuk menghasilkan keluaran tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan keluaran apa yang dihasilkan.



Gambar 4.5 Diagram aktifitas lakukan pembentukan klaster

Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. Pesan digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah pesan. Objek tersebut memiliki sifat khusus yang dapat didefinisikan menjadi objek *boundary, controller* dan *persistent entity*

4.1.2.1 Kelas Utilitis

Kelas utilitis merupakan kelas yang digunakan berulang kali pada setiap proses. Detail setiap kelas utilitis aplikasi ditunjukkan pada tabel 4.6.

Tabel 4.6 Kelas utiiitis beserta fungsinya

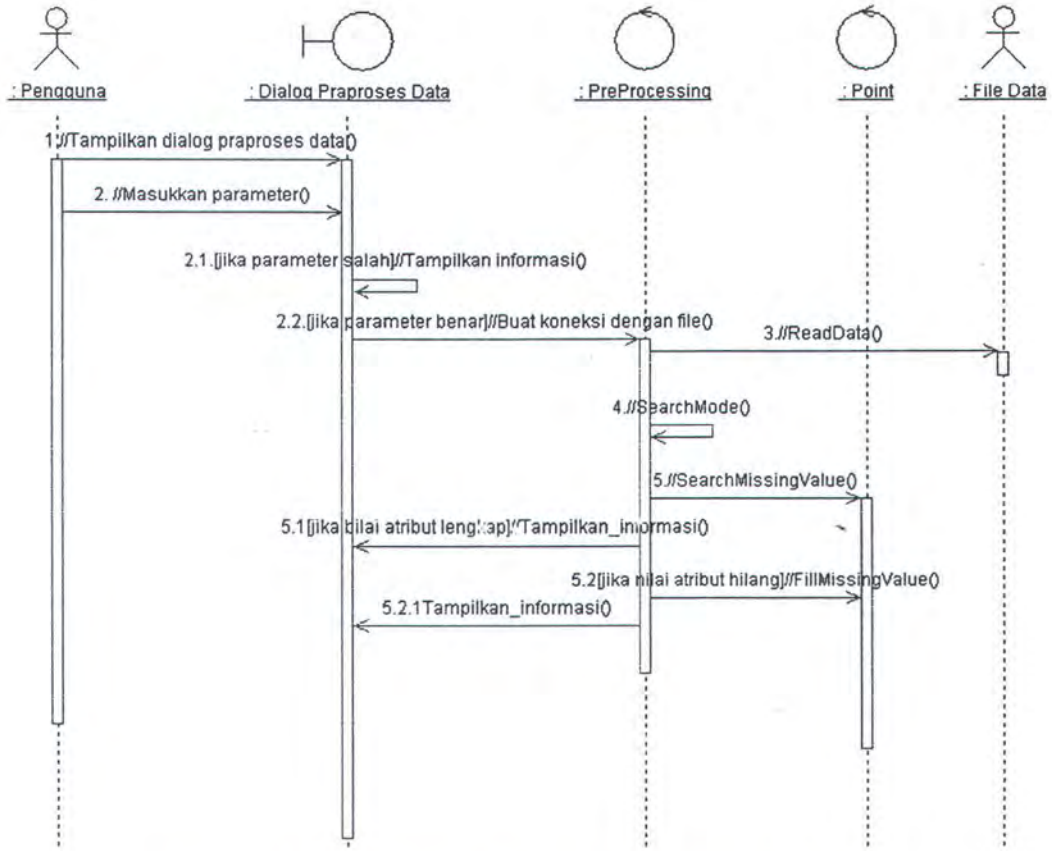
Kelas utilitis	Fungsi
Kontrol File	Kelas boundary untuk membaca data dari file
Point	Kelas kontrol untuk menghitung jarak antar titik
Cluster	Kelas entity untuk menangani informasi klaster

4.1.2.2 Diagram Sekuensi

Berikut ini merupakan diagram sekuensi pada aplikasi yang diimplementasikan dalam tesis ini.

(e) Lakukan praproses data

Proses dimulai saat aktor pengguna memilih menu "lakukan praproses data" untuk menampilkan antar-muka praproses data. Ketika parameter praproses dieksekusi, maka aplikasi akan melakukan validasi parameter melalui antar-muka praproses data. Jika parameter valid, maka aplikasi akan membaca data sesuai dengan nama file yang ditentukan. Apabila pembacaan berhasil maka dipanggil method untuk melakukan praproses data. Jika parameter salah maka aplikasi akan menampilkan informasi tersebut. Diagram sekuensi lakukan praproses data ditunjukkan pada gambar 4.6.

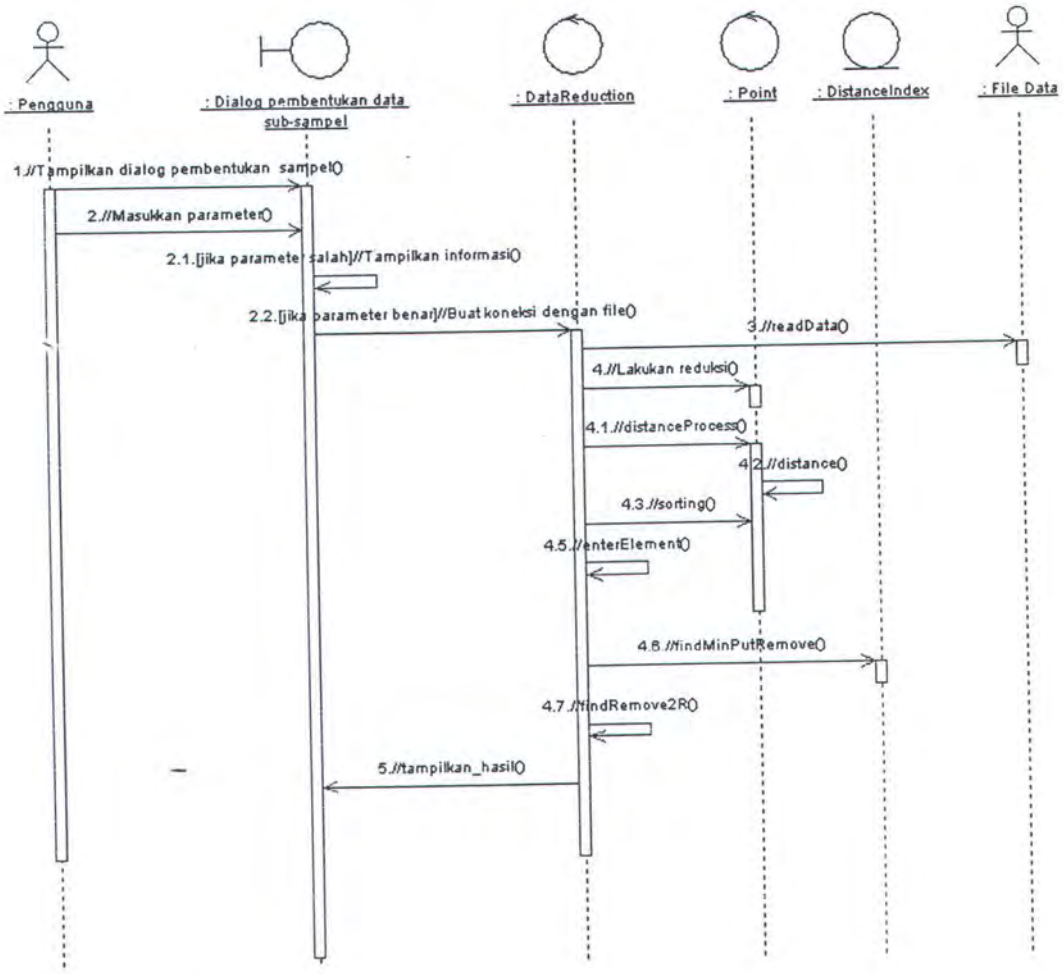


Gambar 4.6. Diagram sekuensi lakukan praproses data

(f) Lakukan pembentukan data sub-sampel

Proses dimulai saat aktor pengguna memilih menu "lakukan pembentukan data sub-sampel" untuk menampilkan antar-muka pembentukan data sub-sampel. Ketika parameter pembentukan data sub-sampel dieksekusi, maka aplikasi akan melakukan validasi parameter melalui antar-muka pembentukan data sub-sampel. Jika parameter valid, maka aplikasi akan membaca data sesuai dengan nama file yang ditentukan. Apabila pembacaan berhasil maka dipanggil method untuk melakukan pembentukan data sub-sampel. Jika parameter salah maka aplikasi

akan menampilkan informasi tersebut. Diagram sekuensi lakukan pembentukan data sub-sampel ditunjukkan pada gambar 4.7.

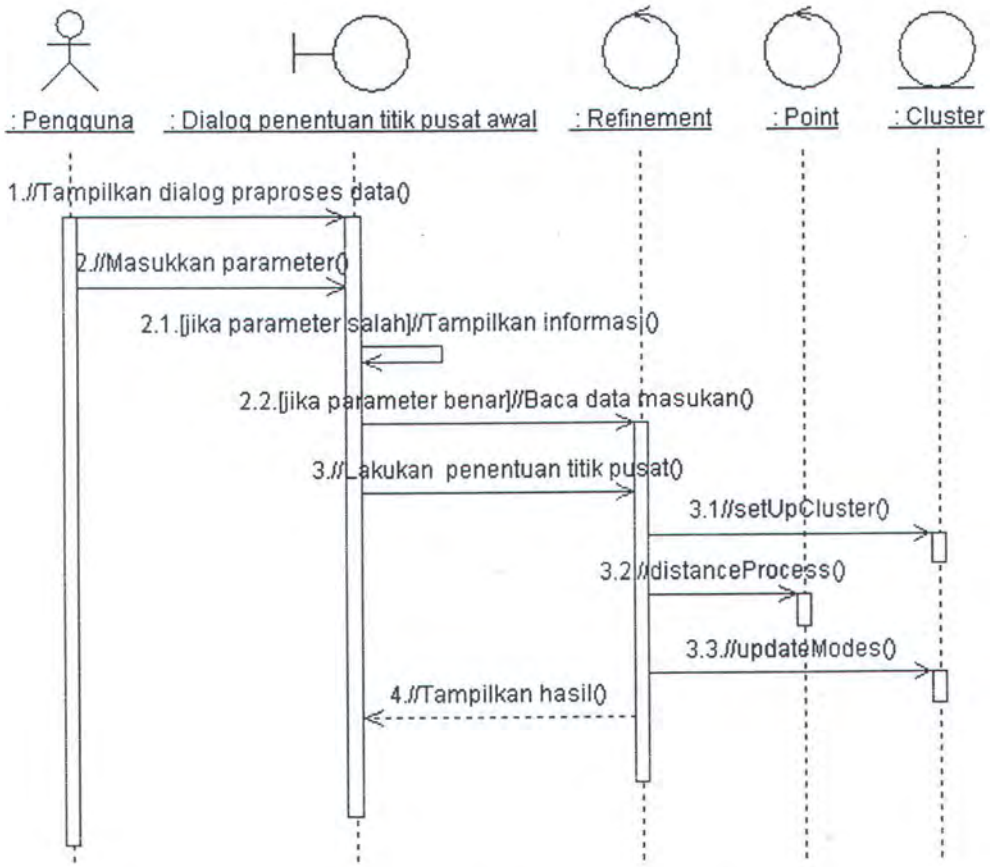


Gambar 4.7. Diagram sekuensi lakukan pembentukan data sub-sampel

(g) Lakukan penentuan titik pusat awal

Proses dimulai saat aktor pengguna memilih menu "lakukan penentuan titik pusat awal" untuk menampilkan antar-muka pembentukan data sub-sampel. Ketika parameter pembentukan data sub-sampel dieksekusi, maka aplikasi akan melakukan validasi parameter melalui penentuan titik pusat awal. Jika parameter valid, maka aplikasi akan membaca data masukan dan melakukan proses

pembentukan titik pusat awal dengan jumlah sesuai dengan masukan pengguna. Jika parameter salah maka aplikasi akan menampilkan informasi tersebut. Diagram sekuensi lakukan penentuan titik pusat awal ditunjukkan pada gambar 4.8

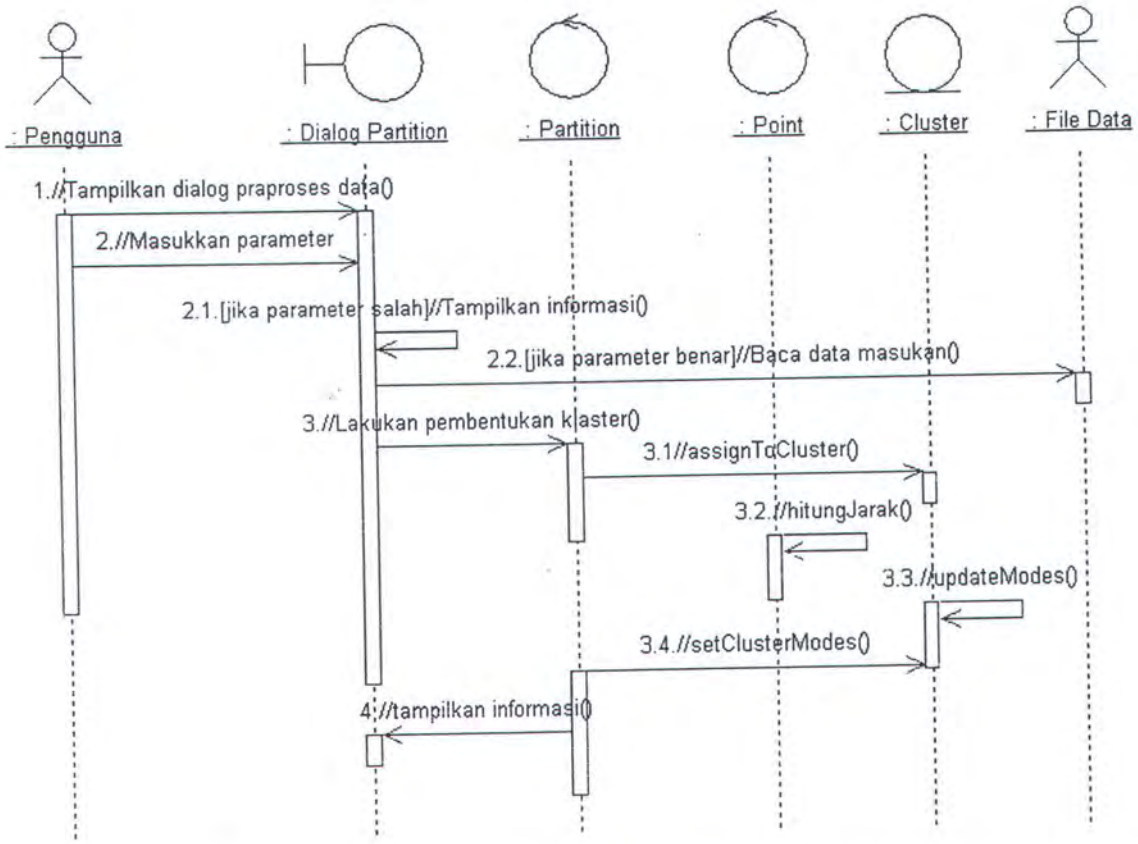


Gambar 4.8. Diagram sekuensi lakukan penentuan titik pusat awal

(h) Lakukan penentuan titik pusat awal

Proses dimulai saat aktor pengguna memilih menu "lakukan pembentukan klaster" untuk menampilkan antar-muka pembentukan klaster. Ketika parameter pembentukan klaster dieksekusi, maka aplikasi akan melakukan validasi parameter pembentukan klaster. Jika parameter valid, maka aplikasi akan

membaca data masukan dan melakukan proses pembentukan klaster. Jika parameter salah maka aplikasi akan menampilkan informasi tersebut. Diagram sekuensi lakukan pembentukan klaster ditunjukkan pada gambar 4.9.



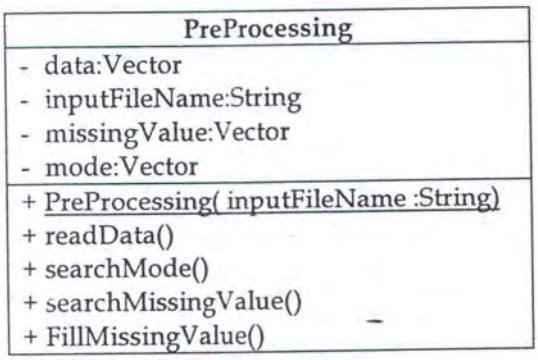
Gambar 4.9. Diagram sekuensi lakukan pembentukan klaster

4.1.3 Diagram Kelas

Sub bab ini menjelaskan tentang kelas-kelas yang akan dibangun sesuai dengan desain diagram *use case*. Terdapat delapan kelas utama untuk aplikasi yang dikembangkan sesuai desain diagram use case. Sub bab berikutnya akan menjelaskan semua kelas yang akan dibuat.

4.1.3.2 Kelas Praproses Data

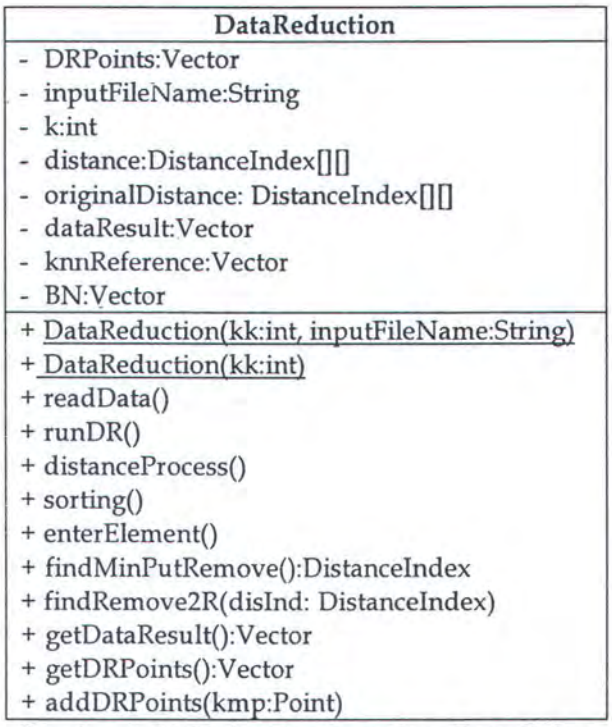
Kelas ini digunakan untuk memastikan bahwa data masukan tidak mengandung nilai yang hilang dan digunakan untuk menentukan mode konseptual tiap kelas dalam suatu data masukan. Pertama kali mode konseptual akan ditentukan selanjutnya bila data mengandung nilai yang hilang maka nilai yang hilang untuk tiap-tiap atribut akan diganti dengan nilai mode konseptual yang telah ditemukan. Kelas diagram yang digunakan untuk menangani praproses data dapat dilihat pada gambar 4.10.



Gambar 4.10 Kelas Praproses Data

4.1.3.3 Kelas Pembentukan Data Sub-sampel

Kelas yang digunakan untuk mereduksi data dalam rangka membentuk data sub-sampel ditunjukkan pada gambar 4.11. Kelas ini membaca dataset masukan dari data yang berupa teks. Dari kelas ini akan dihasilkan data sub-sampel.



Gambar 4.11 Kelas Pembentukan Data Sub-sampel

4.1.3.4 Kelas Representasi Titik

Pada kelas pembentukan sub-sampel terdapat kelas bantuan, kelas yang digunakan untuk mewakili obyek data, yaitu kelas KModesPoint. Kelas KmodesPoint mewakili titik atau data yang selanjutnya data ini akan diukur jaraknya dengan titik yang lain. Kelas Point ditunjukkan pada gambar 4.12. Kelas ini mewakili titik yang akan dihitung jaraknya terhadap titik-titik yang lain. Pada kelas ini diimplementasikan dua rumus pengukuran jarak dengan cara *overloading method.* Yaitu pengukuran jarak tanpa mempertimbangkan modus anggota klaster yang telah terbentuk pada tahap sebelumnya dan pengukuran jarak yang mempertimbangkan modus anggota klaster yang telah terbentuk pada tahap sebelumnya. Pengukuran jarak tanpa mempertimbangkan modus anggota klaster

yang telah terbentuk pada tahap sebelumnya digunakan pada proses pembentukan data sub-sampel, optimasi titik pusat awal dan proses pembentukan klaster pada iterasi pertama. Sedang pengukuran jarak dengan mempertimbangkan modus anggota klaster yang telah terbentuk pada tahap sebelumnya digunakan oleh proses pembentukan klaster pada iterasi selain pertama dengan tujuan untuk meminimalkan jarak antara obyek data dengan titik pusat dan anggota klaster yang lainnya. Selain dengan kegunaan diatas, kegunaan kelas ini juga untuk menyimpan informasi nomor klaster titik, representasi atau atribut-atribut titik dan indek titik dalam dataset.

Point
- point:Vector - index:int - clusterNumber:int
+ KModesPoint(p:Vector) + KModesPoint(p:Point) + assignToCluster(clusterNumber:int) + equals(obj:Object):boolean + getClusterNumber():int + getPoint():Vector + getIndex():int + distance(dp1: Point, dp2: Point, clusterMember:Vector) :double + distance(dp1: Point, dp2: Point) :double + toString():String

Gambar 4.12 Kelas Representasi Titik

4.1.3.5 Kelas Posisi Titik

Selama proses reduksi data untuk membentuk data sub-sampel, titik-titik data akan mengalami proses sorting dan proses penggunaan titik-titik tersebut menjadi kandidat data yang tidak direduksi. Selama proses tersebut maka indeks titik-titik tersebut harus terus dijaga supaya tidak kehilangan informasi

sebelumnya. Untuk menangani hal tersebut maka diperlukan suatu kelas sebagaimana ditunjukkan pada gambar 4.13.

DistanceIndex
- distance:double - index:int
+ DistanceIndex(idx :int, dst:double) + setDistance(dist: double) + getDistance():double + setIndex(id: int) + getIndex():int

Gambar 4.13 Kelas Posisi Titik

4.1.3.6 Kelas Optimasi Titik Pusat Awal

Data sub-sampel hasil mereduksi data akan menjadi masukan bagi proses optimasi titik pusat. Kelas yang digunakan dalam mengoptimasi titik pusat ditunjukkan pada gambar 4.14.

Refinement
- k:int - clPoints:Vector - inputFileName:String - clustersCL:Vector
+ Refinement(k:int, subsampleData:Vector) + readData() + runOpt() + setUpCluster() + updateModes() + distanceProcess() + getClPoints():Vector + getClClusters():Vector + getK():int + toString():String

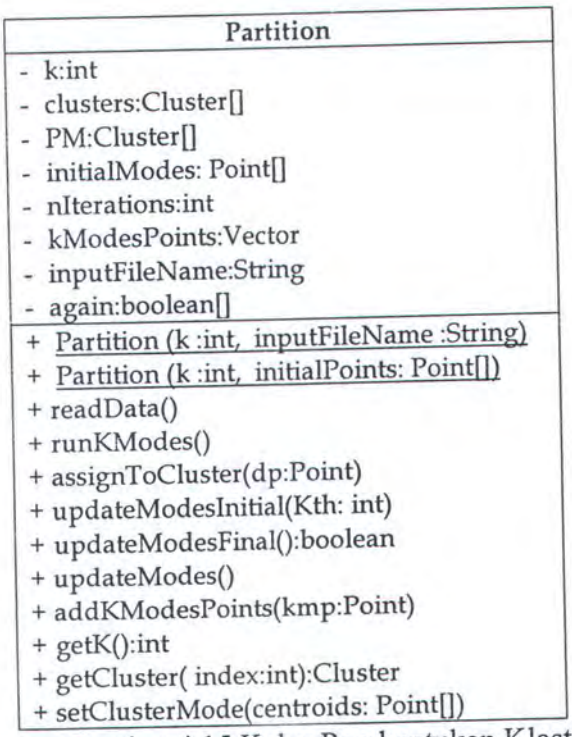
Gambar 4.14 Kelas Optimasi Titik Pusat Awal

4.1.3.7 Kelas Klasterisasi Data Kategorikal

Data masukan beserta sejumlah titik pusat hasil optimasi akan menjadi masukan titik pusat awal bagi klasterisasi data kategorikal. Kelas yang digunakan untuk membentuk klaster ditunjukkan pada gambar 4.15.

4.1.3.8 Kelas Informasi Klaster

Selama proses pembentukan klaster, informasi klaster yang terbentuk harus disimpan. Informasi klaster ini bisa berupa posisi titik pusat klaster dan titik-titik anggota klaster. Kelas dengan kegunaan tersebut ditunjukkan pada gambar 4.16.



Gambar 4.15 Kelas Pembentukan Klaster

Cluster
- mode: Point - clusterMember:Vector
+ Cluster(clusterNumber :int) + Cluster() + setMode(modeDataPoint: Point) + getMode():KmodesPoint + setClusterNumber(no:int) + getClusterNumber():int + addClusterMember(point: Point) + addClusterMember(point:Vector) + getClusterMember():Vector + toString():String

Gambar 4.16 Kelas Informasi Klaster

4.1.3.9 Kelas Perhitungan Kesalahan

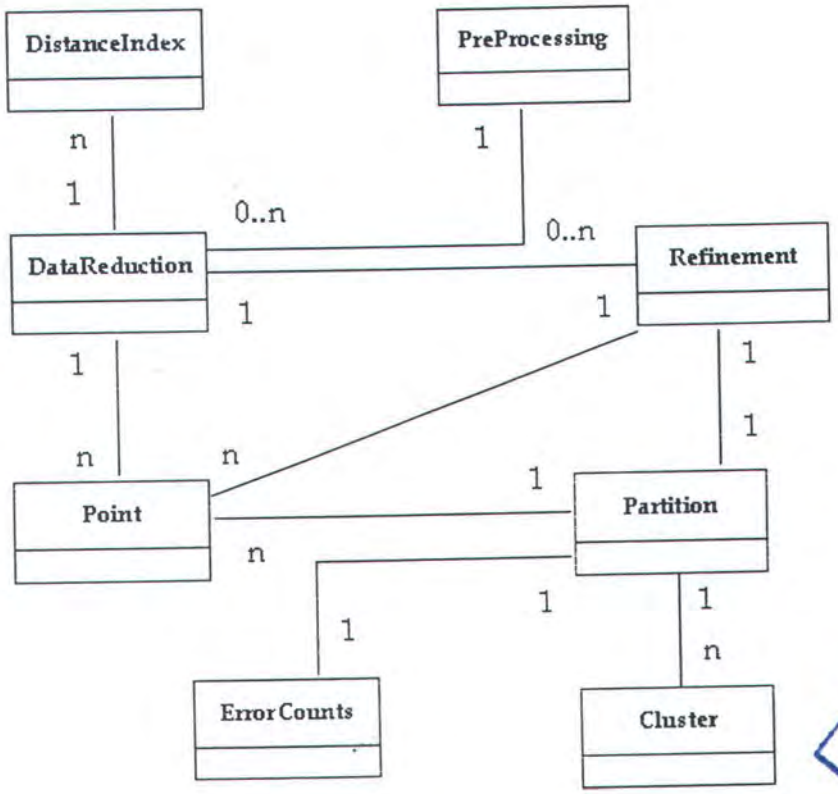
Jika klaster sudah terbentuk maka untuk validasi hasil diperlukan informasi tingkat keakuratan hasil klaster. Perhitungan keakuratan hasil klaster ditangani oleh kelas yang ditunjukkan pada gambar 4.17

ErrorCounts
- k:int - jmlPerClass: int[] - result:Vector
+ ErrorCounts(rslt:Vector, kValue:int, jmlDataPerKlas:int[]) + missClassCounts():Hashtable

Gambar 4.17 Kelas Perhitungan Kesalahan

4.1.3.10 Diagram Hubungan Antar Kelas

Diagram kelas menggambarkan keseluruhan kelas dari aplikasi yang mengimplementasikan desain diagram *use case*. Terdapat delapan kelas untuk aplikasi yang dikembangkan seperti yang ditunjukkan pada gambar 4.18. Detail dari kelas tersebut yaitu state dan method nya bisa dilihat pada pembahasan sub bab sebelumnya.



Gambar 4.18 Diagram hubungan antar kelas

Dari beberapa kelas tersebut maka dibuat kelas-kelas utama yaitu kelas yang mengimplementasikan algoritma perbaikan titik pusat awal untuk klasterisasi data kategorikal. Kelas-kelas utama ini antara lain:

a. Kelas HirarkiPartisiReduksi

   Kelas yang mengimplementasikan algoritma perbaikan titik pusat awal untuk klasterisasi data kategorikal. Pada algoritma ini proses reduksi termasuk dalam proses klasterisasi. Diagram kelas ini ditunjukkan pada gambar 4.19.

b. Kelas HirarkiPartisiTanpaReduksi

   Kelas yang mengimplementasikan algoritma perbaikan titik pusat awal untuk klasterisasi data kategorikal tanpa mereduksi data. Pada algoritma ini proses reduksi tidak terjadi dalam proses klasterisasi. Diagram kelas ini ditunjukkan pada gambar 4.20.

HirarkiPartisiReduksi
- kRed:int - kKlaster: int - dataReductionResult :Vector - initialPoints: Point[] - objects:Vector - inputFileName:String - startRed:Date - finishRed:Date - runtimeRed:String - startOpt:Date - finishOpt:Date - runtimeOpt:String - startKlast:Date - finishKlast:Date - runtimeKlast:String
+ HirarkiPartisiReduksi(kRed:int, kKlast:int, fileName:String + readData() + runDataReduction():Vector + fromDR2CL(v:Vector) + runOptimation() + setInitialPoints(v:Vector) + runPartition() + time(start:Date, finish:Date):String

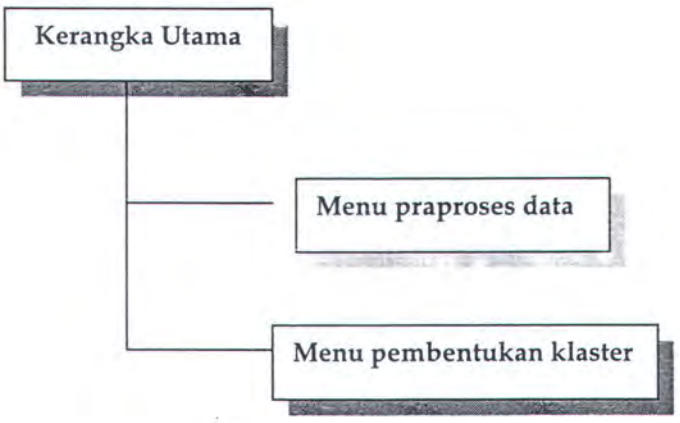
Gambar 4.19 Kelas Hirarki Partisi Reduksi

HirarkiPartisiTanpaReduksi
- kKlaster: int - initialPoints: Point[] - objects:Vector - inputFileName:String - startOpt:Date - finishOpt:Date - runtimeOpt:String - startKlast:Date - finishKlast:Date - runtimeKlast:String
+ HirarkiPartisiTanpaReduksi(kKlast::int, fileName:String) + readData() + runOptimation() + setInitialPoints(v:Vector) + runPartition() + time(start:Date, finish:Date):String

Gambar 4.20 Kelas Hirarki Partisi Tanpa Reduksi

4.1.4 Desain Antar-muka

Antar-muka merupakan komponen yang menghubungkan aplikasi dengan aktor pengguna. Desain antar-muka aplikasi ditunjukkan pada gambar 4.21.



Gambar 4.21 Desain antar-muka

Berdasarkan desain antar-muka pada gambar 4.21terdapat empat antar-muka. Berikut detail keempat antar-muka tersebut:

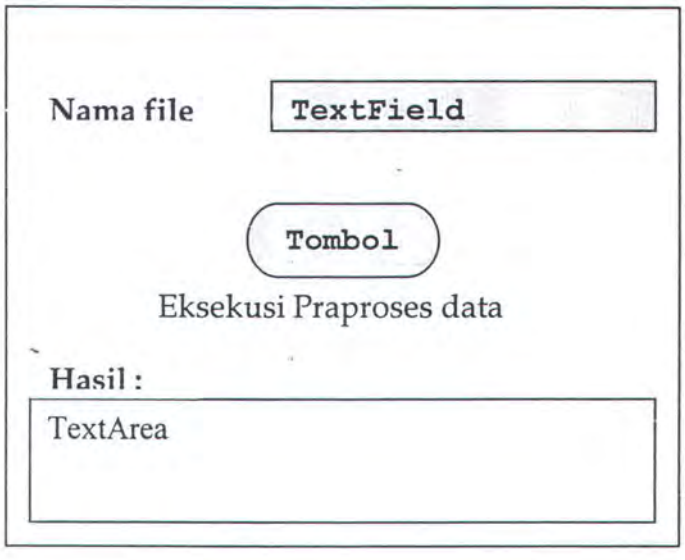
a) Antar-muka kerangka utama

Antar-muka kerangka utama merupakan antar-muka awal aplikasi dijalankan. Antar-muka kerangka utama memiliki menu untuk menampilkan dialog antar-muka setiap proses.

b) Antar-muka praproses data

Antar-muka praproses data merupakan dialog antar-muka *use case* lakukan praproses data. Antar-muka ini menangani data masukkan berupa nama file yang berisi data yang akan dilakukan praproses. Tipe data masukan adalah String.

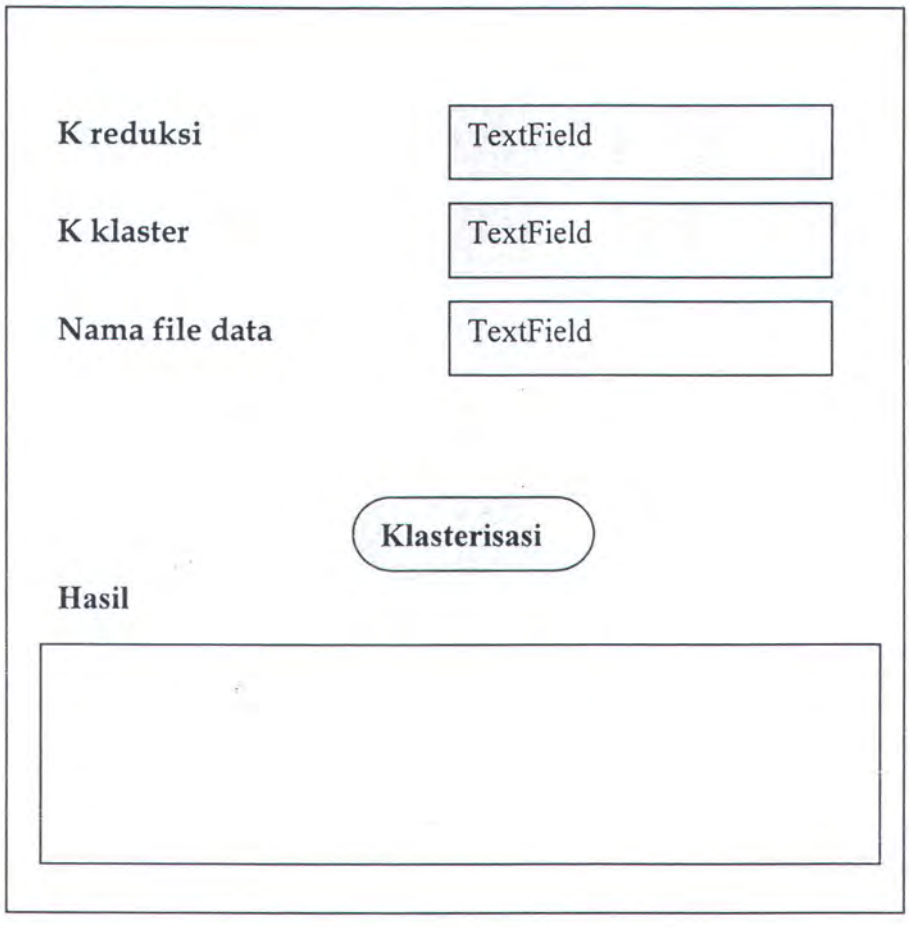
Berdasarkan kebutuhan data masukkan *use case* lakukan praproses data, maka desain antar-muka praproses data ditunjukkan pada gambar 4.22.



Gambar 4.22 Desain antar-muka praproses data

c) Antar-muka Pembentukan Klaster

Antar-muka pembentukan klaster merupakan dialog antar-muka *use case* lakukan pembentukan data sub-sampel, penentuan titik pusat awal dan pembentukan klaster. Antar-muka ini menangani data masukkan berupa jumlah ketetanggan ketika membentuk klaster dan jumlah titik pusat yang harus dihasilkan. Jumlah titik pusat yang harus dihasilkan sama dengan jumlah klaster yang akan dihasilkan sehingga cukup membuat satu masukan. Tipe data masukan jumlah ketetanggaan dan jumlah klaster yang dihasilkan adalah integer. Berdasarkan kebutuhan data antar-muka lakukan pembentukan klaster, maka desain antar-muka pembentukan klaster ditunjukkan pada gambar 4.23.



Gambar 4.23 Desain antar-muka pembentukan klaster

4.2 Implementasi Aplikasi

Sebagaimana desain dilakukan, implementasi aplikasi ini juga dibagi menjadi dua bagian, yakni implementasi kelas dan implementasi antar-muka. Bahasa pemrograman yang digunakan adalah Java dengan kompiler jdk1.4. Implementasi aplikasi ini disarankan dijalankan pada lingkungan yang telah diujicobakan seperti dijelaskan dalam bab 5 (sub-bab 5.1 lingkungan uji coba)

4.2.1 Implementasi Kelas

Diagram kelas menggambarkan keseluruhan kelas dari aplikasi yang mengimplementasikan desain diagram *use case*. Terdapat sepuluh kelas untuk aplikasi yang dikembangkan, antara lain:

a) Kelas praproses data

Kelas ini berfungsi untuk menyiapkan data supaya siap menjadi data masukan. Sesuai dengan fungsinya maka kelas ini pada intinya akan mengganti nilai atribut yang hilang dengan nilai atribut yang bersesuaian. Pada implementasi aplikasi kelas ini disebut sebagai kelas `PreProcessing`. Selanjutnya setiap method yang pada kelas `PreProcessing` ditunjukkan tabel 4.7.

Tabel 4.7 Penjelasan method kelas `PreProcessing`.

Method	Keterangan
`PreProcessing()`	Konstruktor pada kelas `PreProcessing`
`readData()`	Membaca data masukan berupa file
`searchMode()`	Mencari mode data masukan, data masukan dianggap satu klaster, jadi yang ditemukan hanya satu buah mode
`searchMissingValue()`	Mencari atribut-atribut yang memiliki nilai yang hilang
`fillMissingValue()`	Mengisi nilai atribut yang hilang dengan nilai mode yang ditemukan pada atribut tersebut

Proses pembacaan data dilakukan terhadap data yang tersimpan dalan file berekstensi .txt. Kode sumber metode `readData()` ditunjukkan pada segmen program 4.1. Proses mencari mode seperti ditunjukkan pada segmen program 4.2.

```
(1)  public void readData() throws IOException{
(2)          BufferedReader in = new BufferedReader(new
(3)                                  FileReader(this.inputFileName));
(4)                String line = "";
(5)                Vector dimension = new Vector();
(6)          int index = 0;
(7)
(8)                while ((line = in.readLine()) != null ){
(9)                StringTokenizer st = new StringTokenizer(line, "
(10)                                                  \t\n\r\f,");
(11)               dimension.addElement(new Integer(index));
(12)               while(st.hasMoreTokens()){
(13)                     dimension.addElement(st.nextToken());
(14)               }
(15)               KModesPoint dp = new KModesPoint(dimension);
(16)               dp.assignToCluster(-2);
(17)               dp.setIndex(index);
(18)               this.data.add(dp);
(19)               dimension = new Vector();
(20)               index++;
(21)         }
(22)         in.close();
(23) } // end of readData()
```

Segmen program 4.1 Method `readData()`

```
(1)  KModesPoint searchMode(){
(2)              Vector newMode = new Vector();
(3)     Vector dim = new Vector();
(4)     Hashtable countFreq = new Hashtable();
(5)     String modus = null;
(6)     int coloumn = 1;
(7)     KModesPoint m = null;
(8)     newMode.addElement("updateModus");
(9)     do{
(10)            for (int j=0; j<this.data.size(); j++){
(11)               KModesPoint kp = (KModesPoint)
    this.data.elementAt(j);
(12)               dim = kp.getPoint();
(13)               String dimValue = (String) dim.elementAt(coloumn);
```

Segmen program 4.2 Method `searchMode()`

```
(14)        if (countFreq.containsKey(dimValue)){
(15)                Integer counter = (Integer)
    countFreq.get(dimValue);
(16)                int primCounter = counter.intValue()+1;
(17)            countFreq.put(dimValue, new Integer(primCounter));
(18)              }else
(19)                countFreq.put(dimValue, new Integer(1));
(20)        }  // end of for j
(21)          int pembanding = 0;
(22)
(23)          for (Enumeration e =
    countFreq.keys();e.hasMoreElements();){
(24)           String stringModus = (String) e.nextElement();
(25)           Integer count = (Integer) countFreq.get(stringModus);
(26)           int kandidatModus = count.intValue();
(27)           if (kandidatModus>pembanding){
(28)              pembanding = kandidatModus;
(29)               modus = stringModus;
(30)           }
(31)        }
(32)        countFreq.clear();
(33)        newMode.addElement(modus);
(34)        if (coloumn == (dim.size()-1)){
```

Segmen program 4.2 Method `searhMode()` (lanjutan)

b) Kelas pembentukan data sub-sampel

Kelas ini berfungsi untuk membentuk data sub-sampel. Sesuai dengan fungsinya maka kelas ini pada intinya akan mereduksi data sehingga ukuran data menjadi lebih kecil tetapi tidak menghilangkan informasi data keseluruhan. Pada implementasi aplikasi kelas ini disebut sebagai kelas `DataReduction`. Selanjutnya setiap method yang pada kelas `DataReduction` ditunjukkan tabel 4.8.

Tabel 4.8 Penjelasan method kelas `DataReduction`.

Method	Keterangan
`DataReduction()`	Konstruktor pada kelas `DataReduction`
`readData()`	Membaca data masukan berupa file
`runDR()`	Menjalankan proses pembentukan data sub-sampel

Tabel 4.8 Penjelasan method kelas `DataReduction`. (lanjutan)

`distanceProses()`	Menghitung jarak antar semua kombinasi titik
`sorting()`	Mengurutkan jarak antar titik
`enterElement()`	Memasukkan k buah titik menjadi kandidat data hasil reduksi
`findMinPutRemove()`	Memilih titik dengan jarak terpendek dan menjadikan titik tersebut menjadi bagian dari hasil reduksi
`findRemove2R()`	Menghapus titik yang berada dalam radius 2*jarak terpendek yang baru ditemukan
`getDataResult()`	Mendapatkan data hasil reduksi
`addDRPoints()`	Menambahkan titik sebagai masukan

Proses pembacaan data dilakukan terhadap data yang tersimpan dalan file berekstensi .txt. Method yang digunakan untuk proses mengurutkan jarak seperti ditunjukkan pada segmen program 4.3. Kode sumber metode `distanceProses()` ditunjukkan pada segmen program 4.4.

```
(1)  public void sorting(){
(2)       DistanceIndex temp;
(3)       for(int k=0; k<this.DRPoints.size(); k++){
(4)          for(int i=0; i<this.DRPoints.size()-1; i++)
(5)             for(int j=i+1; j<this.DRPoints.size(); j++){
(6)                double xx =
     ((DistanceIndex)distance[k][i]).getDistance();
(7)                double xxx =
     ((DistanceIndex)distance[k][j]).getDistance();
(8)                if(xx > xxx){
(9)                   temp = distance[k][i];
(10)                  distance[k][i] = distance[k][j];
(11)                  distance[k][j] = temp;
(12)               }
(13)            }
(14)        }
(15) }
```

Segmen program 4.3 Method `sorting()`

```
(1)  public void distanceProcess(){
(2)     this.distance = new
     DistanceIndex[this.DRPoints.size()][this.DRPoints.size()];
(3)     this.originalDistance = new
     DistanceIndex[this.DRPoints.size()][this.DRPoints.size()];
(4)     double tempDistance = -1.0;
(5)     for(int i=0; i<(this.DRPoints.size()); i++){
(6)          KModesPoint kmp = (KModesPoint)
(7)          this.DRPoints.elementAt(i);
(8)             for (int k=0; k<this.DRPoints.size(); k++){
(9)                if (i==k){
(10)                     tempDistance = -1;
(11)                     distance[i][k]= new DistanceIndex(k,
     tempDistance);
(12)               }else{
(13)                     KModesPoint kmpNext = (KModesPoint)
(14)                     this.DRPoints.elementAt(k);
(15)                     tempDistance = KModesPoint.distance(kmp,
     kmpNext);
(16)                     distance[i][k] = new DistanceIndex(k,
     tempDistance);
(17)                     originalDistance[i][k] = new
     DistanceIndex(k, tempDistance);
(18)               } //end else
(19)          }  }
(20) }  // end of distanceProcess
```

Segmen program 4.4 Method `distanceProcess()`

Method `findRemove2R()`yang digunakan untuk menghapus titik yang berada dalam radius 2*jarak terpendek yang baru ditemukan ditunjukkan pada segmen program 4.5. Method `findMinPutRemove()` yang digunakan untuk memilih titik dengan jarak terpendek dan menjadikan titik tersebut menjadi bagian dari hasil reduksi ditunjukkan pada segmen program 4.6.

```
(1)  public void findRemove2R(DistanceIndex disInd){
(2)     double _2R = 2*disInd.getDistance();
(3)
(4)     for(int i=0; i<this.DRPoints.size(); i++){
(5)     if (this.originalDistance[i][disInd.getIndex()] == null)
(6)          continue;
(7)     else if
     (((DistanceIndex)(this.originalDistance[i][disInd.getIndex()])).g
     etDistance() <= _2R &
     ((DistanceIndex)(this.originalDistance[i][disInd.getIndex()])).ge
     tDistance() >=0.0){
(8)
(9)     for(int j=0; j<this.knnReference.size(); j++){
(10)         if
     ( (((DistanceIndex)this.knnReference.elementAt(j)).getIndex()) ==
     i ){
(11)              this.knnReference.remove(j);
(12)              j= --i;
(13)         }
(14)    }
(15)
(16)    for(int k=0; k<this.DRPoints.size(); k++){
(17)         this.originalDistance[i][k] = null;
(18)    }
(19)    // Searching in the radius 2R by row_in the distance
(20)    for(int k=0; k<this.DRPoints.size(); k++){
(21)         this.originalDistance[k][i] = null;
(22)    }
(23)
(24)    this.BN.remove(new Integer(i));
(25)
(26)   }
(27)  }
(28)    // Making null for index that entered E
(29)    for(int k=0; k<this.DRPoints.size(); k++){
(30)         this.originalDistance[disInd.getIndex()][k] = null;
(31)    }
(32)
(33) // Searching in the radius 2R by row in the distance
(34)    for(int k=0; k<this.DRPoints.size(); k++){
(35)         this.originalDistance[k][disInd.getIndex()] = null;
(36)    }
(37) }
```

Segmen program 4.5 Method `findRemove2R()`

```
(1)  public DistanceIndex findMinPutRemove(){
(2)     double min = 1000.0;
(3)     int index = -1;
(4)     for(int i=0; i<this.knnReference.size(); i++){
(5)        if
     ( (((DistanceIndex)this.knnReference.elementAt(i)).getDistance())
     <= min ){
(6)           min =
     ((DistanceIndex)this.knnReference.elementAt(i)).getDistance();
(7)           index =
     ((DistanceIndex)this.knnReference.elementAt(i)).getIndex();
(8)        }
(9)     }
(10)    this.dataResult.addElement(this.DRPoints.elementAt(index));
(11)    this.knnReference.size());
(12)    for(int i=0; i<this.knnReference.size(); i++){
(13)       if
     ((((DistanceIndex)this.knnReference.elementAt(i)).getIndex()) ==
     index ){ this.knnReference.remove(i);i= --i;}
(14)    }
(15)    this.BN.remove(new Integer(index));
(16)    findMinPutRemove: "+index);
(17)    DistanceIndex di = new DistanceIndex(index, min);
(18)    return di;
(19) }
```

Segmen program 4.6 Method `findMinPutRemove()`

c) Kelas Representasi Titik

Kelas ini berfungsi untuk merepresentasikan sebuah titik. Sesuai dengan fungsinya maka kelas ini merepresentasikan operasi-operasi sebuah titik antara lain mengukur jarak dengan titik lain, menampilkan atribut-atributnya, menampilkan nomor klasternya, membandingkan posisi titik tersebut dengan titik yang lain, dan mengalokasikan titik ke klaster tertentu. Pada implementasi aplikasi kelas ini disebut sebagai kelas `Point`. Selanjutnya setiap method yang pada kelas `Point` ditunjukkan tabel 4.9.

Tabel 4.9 Penjelasan method kelas `Point`.

Method	Keterangan
`Point()`	Konstruktor pada kelas `Point`
`assignToCluster()`	Mengalokasikan titik ke klaster tertentu
`equals()`	Membandingkan posisi antar titik
`getClusterNumber()`	Mendapatkan informasi titik tersebut anggota klaster yang mana
`getPoint()`	Mendapatkan informasi nilai atribu-atribut
`getIndex()`	Mendapatkan informasi index titik
`distance()`	Mengukur jarak dengan titik yang lain

Method `distance()` yang digunakan untuk Mengukur jarak dengan titik yang lain dibagi menjadi dua (overloading method). Segmen program 4.7 untuk mengukur jarak antar dua titik tanpa mempertimbangkan anggota klaster yang lain. Segmen program 4.8 untuk mengukur jarak antar dua titik dengan mempertimbangkan anggota klaster yang lain. Method `equals()` yang digunakan untuk membandingkan posisi titik tersebut dengan titik yang lain ditunjukkan pada segmen program 4.9.

```
(1) public static double distance(KModesPoint dp1, KModesPoint dp2,
    Vector clusterMember){
(2)     double result = 0;
(3)     double totalDistance = 0;
(4)     int [] counter = new int[dp1.getPoint().size()];
(5)     for(int j=1; j<dp1.getPoint().size(); j++){
(6)     if
    (((String)(dp1.getPoint().elementAt(j))).equals((String)(dp2.get
    Point().elementAt(j)))){
(7)         Iterator i = clusterMember.iterator();
(8)            while (i.hasNext()) {
(9)            KModesPoint dp = (KModesPoint)(i.next());
(10)                        If
```

Segmen program 4.7 Method `distance()`

```
(11)                counter[j]++;
(12)                }
(13)                double div = (double)
        (counter[j])/(double)(clusterMember.size());
(14)                result = 1 - div;
(15)           }
(16)           else{                        result =  1;
(17)         }
(18)         totalDistance += result;
(19)         }
(20)             return totalDistance;
(21)     } // end of distance()
```

Segmen program 4.7 Method `distance()` (lanjutan)

```
(1)  public boolean equals(Object obj){
(2)     if (obj instanceof KModesPoint){
(3)     KModesPoint temp = (KModesPoint) obj;
(4)     Vector tt = temp.getPoint();
(5)     Vector ttt = new Vector();
(6)     Vector thisTemp = this.getPoint();
(7)     Vector thisT = new Vector();
(8)     for(int i=1; i<tt.size(); i++){
(9)         ttt.addElement(tt.elementAt(i));
(10)    }
(11)    for(int i=1; i<thisTemp.size(); i++){
(12)        thisT.addElement(thisTemp.elementAt(i));
(13)    }
(14)    if (thisT.equals(ttt))          return true;
(15)    else                            return false;
(16)    }//else instance of
(17)    else                            return false;
(18) }// end of equals
```

Segmen program 4.9 Method `equals()`

d) Kelas Posisi Titik

Kelas ini berfungsi untuk mengeset dan menyimpan informasi indeks titik-titik selama proses pembentukan klaster supaya tidak kehilangan informasi sebelumnya. Untuk menangani hal tersebut maka diimplementasikan kelas DistanceIndex dengan method seperti ditunjukkan pada tabel 4.10.

Tabel 4.10 Penjelasan method kelas `DistanceIndex`.

Method	Keterangan
`DistanceIndex()`	Konstruktor pada kelas `Point`
`setDistance()`	Mengeset jarak titik dalam k ketetanggaan
`getDistance()`	Mendapatkan informasi jarak titik dalam k ketetanggaan
`setIndex()`	Mengeset indek titik dalam k ketetanggaan
`getIndex`	Mendapatkan informasi indek titik dalam k ketetanggaan

e) Kelas penentuan titik pusat awal

Data sub-sampel hasil mereduksi data akan menjadi masukan bagi proses penentuan titik pusat. Kelas yang digunakan untuk keperluan tersebut bernama kelas `Refinement`. Selanjutnya setiap method yang pada kelas `Refinement` ditunjukkan tabel 4.11. Method yang digunakan untuk mengukur jarak ditunjukkan pada segmen program 4.10. Kode sumber metode `updateModes()` ditunjukkan pada segmen program 4.10.

Tabel 4.11 Penjelasan method kelas `Refinement`.

Method	Keterangan
`Refinement()`	Konstruktor pada kelas `Refinement`
`readData()`	Membaca data masukan berupa file
`runOpt()`	Menjalankan proses penentuan titik pusat awal
`setUpCluster()`	Mempersiapkan kelas supaya siap melakukan penentuan titik pusat awal
`updateModes()`	Memutakhirkan mode setiap terjadi alokasi
`distanceProcess()`	Mengukur jarak antar klaster
`getClPoints()`	Mendapatkan informasi perubahan pada setiap titik
`getK()`	Mendapatkan informasi jumlah titik pusat yang harus ditentukan
`toString()`	Menampilkan informasi titik di layar

```
(1)  public void distanceProcess(){
(2)     double distance [][] = new
     double[this.clustersCL.size()][this.clustersCL.size()];
(3)     double min = 1000.0;
(4)     for(int i=0; i<(clustersCL.size()-1); i++){
(5)        Cluster clr = (Cluster) clustersCL.elementAt(i);
(6)     KModesPoint kmp = clr.getMode();
(7)     for (int k=i+1; k<clustersCL.size(); k++){
(8)          Cluster clrNext = (Cluster) clustersCL.elementAt(k);
(9)          KModesPoint kmpNext = clrNext.getMode();
(10)         distance[i][k] = KModesPoint.distance(kmp, kmpNext,
     clr.getClusterMember());
(11)         if (distance[i][k]<min)
(12)               min = distance[i][k];
(13)         }
(14)    }
(15)    int flag = 0;
(16)    for(int i=0; i<(clustersCL.size()-1); i++){
(17)     if (flag==1) break;
(18)         for (int k=i+1; k<clustersCL.size(); k++){
(19)            if (distance[i][k] == min){
(20)               Cluster cTo = (Cluster) clustersCL.elementAt(i);
(21)               Cluster cFrom = (Cluster) clustersCL.elementAt(k);
(22)               Vector v = cFrom.getClusterMember();
(23)               Vector temp = new Vector
(24)               for (int j=0; j<v.size(); j++){
(25)                  KModesPoint kmp = (KModesPoint) v.elementAt(j);
(26)                  kmp.assignToCluster(i);
(27)                  temp.addElement(kmp);
(28)               }
(29)               cTo.addClusterMember(temp);
(30)               clustersCL.remove(k);
(31)               for(int a=k; a<(clustersCL.size()); a++){
(32)                  Cluster cl = (Cluster) clustersCL.elementAt(a);
(33)                  Vector vec = cl.getClusterMember();
(34)                     for (int b=0; b<vec.size(); b++){
(35)                        KModesPoint kmp = (KModesPoint)
     vec.elementAt(b);
(36)                        kmp.assignToCluster(a);
(37)                     }
(38)               }
(39)               flag = 1;
(40)               break;
(41)            }
(42)         }
(43)    }
(44) }  // end of distanceProcess
```

Segmen program 4.10 Method `distanceProcess()`

```
(1)  private void updateModes() {
(2)     Vector newMode = new Vector();
(3)     Vector dim = new Vector();
(4)     Hashtable countFreq = new Hashtable();
(5)     String modus = null;
(6)     int i=0;
(7)     int coloumn = 1;
(8)     newMode.addElement("updateModus");
(9)
(10)    for (i=0; i<this.clustersCL.size(); i++) {
(11)       for (int j=0; j<this.clPoints.size(); j++){
(12)       KModesPoint kp = (KModesPoint) this.clPoints.elementAt(j);
(13)       if (kp.getClusterNumber() == i){
(14)          dim = kp.getPoint();
(15)          String dimValue = (String) dim.elementAt(coloumn
(16)       if (countFreq.containsKey(dimValue)){
(17)          Integer counter = (Integer) countFreq.get(dimValue);
(18)          int primCounter = counter.intValue()+1;
(19)          countFreq.put(dimValue, new Integer(primCounter));
(20)       }else
(21)          countFreq.put(dimValue, new Integer(1));
(22)       }
(23)       } // end of for j
(24)    int pembanding = 0; //temp untuk pembanding
(25)    for (Enumeration e = countFreq.keys();e.hasMoreElements();){
(26)       String stringModus = (String) e.nextElement();
(27)       Integer count = (Integer) countFreq.get(stringModus);
(28)       int kandidatModus = count.intValue();
(29)       if (kandidatModus>pembanding){
(30)          pembanding = kandidatModus;
(31)          modus = stringModus;
(32)       }
(33)    }
(34)    countFreq.clear();
(35)    newMode.addElement(modus);
(36)    if (coloumn == (dim.size()-1)){
(37)       coloumn = 1;
(38)       KModesPoint m = new KModesPoint(newMode);
(39)       m.assignToCluster(i);
(40)       ((Cluster)(clustersCL.elementAt(i))).setMode(m);
(41)       newMode = new Vector();
(42)       newMode.addElement("updateModus");
(43)    }else{
(44)       coloumn++;
(45)       i = --i;
(46)    }
(47)    }  // end of i
(48) } // end of updateModes()
```

Segmen program 4.11 Method `updateModes()`

f) Kelas pembentukan klaster

Data masukan beserta sejumlah titik pusat hasil penentuan titik pusat awal akan menjadi masukan titik pusat awal bagi klasterisasi data kategorikal. Kelas yang digunakan untuk membentuk klaster disebut dengan kelas `Partition`. Selanjutnya setiap method yang pada kelas `Partition` ditunjukkan tabel 4.12. Method yang digunakan untuk menjalankan kelas Partition ditunjukkan pada segmen program 4.12. Method yang digunakan untuk mengalokasikan titik ke klaster dengan mode terdekat ditunjukkan pada segmen program 4.13. Kode sumber metode `updateModesInitial()` ditunjukkan pada segmen program, 4.14 sedang segmen program 4.15 adalah kode sumber method `updateModesFinal()`.

Tabel 4.12 Penjelasan method kelas `Partition()`.

Method	Keterangan
`Partition()`	Konstruktor pada kelas `Refinement`
`readData()`	Membaca data masukan berupa file
`runKModes()`	Menjalankan proses pembentukan klaster
`assignToCluster()`	Mengalokasikan titik ke klaster yang memiliki jarak dengan mode terdekat
`updateModesFinal()`	Memutakhirkan mode setelah semua titik teralokasi
`updateModesInitial()`	Memutakhirkan mode, dilakukan pada saat iterasi pertama saja.

```
(1)  public void runKModes() {
(2)     boolean loop = true;
(3)     do {
(4)     for (int a=0; a<this.k; a++) {
(5)        this.PM[a] = new Cluster();
(6)        KModesPoint temp = new
   KModesPoint(this.clusters[a].getMode());
(7)        temp.assignToCluster(a);
(8)        this.PM[a].setMode(temp);
(9)        this.PM[a].addClusterMember(this.clusters[a].getClusterMe
   mber());
(10)    }
(11)    this.nIterations++;
(12)    Iterator i = this.kModesPoints.iterator();
(13)       while (i.hasNext()){
(14)       this.assignToCluster((KModesPoint)(i.next()));
(15)    }
(16)    this.updateModesFinal();
(17)    for (int a=0; a<this.again.length; a++) {
(18)         if(this.again[a]==false)
(19)         loop = false;
(20)         }
(21)    }while (loop);
(22) } // end of runKModes()
```

Segmen program 4.12 Method `runKModes()`

```
(1)    private void assignToCluster(KModesPoint dp) {
(2)      int currentCluster = 0;
(3)      if (this.nIterations > 1){
(4)         currentCluster = dp.getClusterNumber();
(5)         for (int i=0; i <this.k; i++){
(6)               double tempDistance = KModesPoint.distance(dp,
       this.clusters[i].getMode(),this.clusters[i].getClusterMembe
       r());
(7)               if (tempDistance < minDistance) {
(8)                     minDistance = tempDistance;
(9)                     currentCluster = i;
(10)              }
(11)        }
(12)        if (currentCluster != dp.getClusterNumber()){
```

Segmen program 4.13 Method `assignToCluster()`

```
(13)              boolean bol =
      ((Vector)(this.clusters[dp.getClusterNumber()].getClusterMembe
      r())).removeElement(dp);
(14)         dp.assignToCluster(currentCluster);
(15)         this.clusters[currentCluster].addClusterMember(dp);
(16)         this.updateModes();
(17)       }
(18)    }else{
(19)         double tempDistance = 10000;
(20)         double minDistance = 10000;
(21)         for (int i=0; i <this.k; i++){
(22)              tempDistance = KModesPoint.distance(dp,
     this.initialModes[i]);
(23)              if (tempDistance < minDistance) {
(24)                   minDistance = tempDistance;
(25)                   currentCluster = i;
(26)              }
(27)         }
(28)         dp.assignToCluster(currentCluster);
(29)         this.clusters[currentCluster].addClusterMember(dp);
(30)         this.updateModesInitial(currentCluster);
(31)       }
(32) } // end of assignToCluster
```

Segmen program 4.13 Method `assignToCluster()` (lanjutan)

```
(1)     private void updateModesInitial(int Kth) {
(2)
(3)     Vector newMode = new Vector();
(4)     Vector dim = new Vector();
(5)     Hashtable countFreq = new Hashtable();
(6)     String modus = null;
(7)     int coloumn = 1;
(8)
(9)     newMode.addElement("updateModus");
(10)    Vector tempClusterMember =
        this.clusters[Kth].getClusterMember();
(11)    do{
(12)    for (int j=0; j<tempClusterMember.size(); j++){
(13)         KModesPoint kp = (KModesPoint)
        tempClusterMember.elementAt(j);
(14)         dim = kp.getPoint();
(15)         String dimValue = (String) dim.elementAt(coloumn
(16)    if (countFreq.containsKey(dimValue)){
(17)         Integer counter = (Integer) countFreq.get(dimValue);
(18)         int primCounter = counter.intValue()+1;
```

Segmen program 4.14 Method `updateModesInitial()`

```
(1)     }else
(2)        countFreq.put(dimValue, new Integer(1));
(3)     }  // end of for j
(4)     int pembanding = 0;
(5)     for (Enumeration e =
    countFreq.keys();e.hasMoreElements();){
(6)        String stringModus = (String) e.nextElement();
(7)        Integer count = (Integer) countFreq.get(stringModus);
(8)        int kandidatModus = count.intValue();
(9)        if (kandidatModus>pembanding){
(10)          pembanding = kandidatModus;
(11)          modus = stringModus;
(12)       }
(13)    }
(14)    countFreq.clear();
(15)    newMode.addElement(modus);
(16)    if (coloumn == (dim.size()-1)){
(17)        coloumn = 1;
(18)        KModesPoint m = new KModesPoint(newMode);
(19)        m.assignToCluster(Kth);
(20)        this.initialModes[Kth] = m;
(21)        clusters[Kth].setMode(m); newMode = new Vector();
(22)        newMode.addElement("updateModus");
(23)        break;
(24)    }else{
(25)        coloumn++;
(26)    }
(27) }while(true);
(28) } // end of updateModes()
```

Segmen program 4.14 Method `updateModesInitial()` (lanjutan)

```
(1)  private boolean updateModesFinal() {
(2)     boolean reply = true;
(3)     Vector newMode = new Vector();
(4)     Vector dim = new Vector();
(5)     Hashtable countFreq = new Hashtable();
(6)     String modus = null;
(7)     int coloumn = 1;
(8)     int i=0;
(9)     newMode.addElement("updateModus");
(10)    for (i=0; i<this.k; i++) {
(11)       Vector tempClusterMember =
    this.clusters[i].getClusterMember();
```

Segmen program 4.15 Method `updateModesFinal()`

```
(12)        for (int j=0; j<tempClusterMember.size(); j++){
(13)           KModesPoint kp = (KModesPoint)
    tempClusterMember.elementAt(j);
(14)           dim = kp.getPoint();
(15)           String dimValue = (String) dim.elementAt(coloumn);
(16)           if (countFreq.containsKey(dimValue)){
(17)              Integer counter = (Integer) countFreq.get(dimValue);
(18)              int primCounter = counter.intValue()+1;
(19)              countFreq.put(dimValue, new Integer(primCounter));
(20)           }else
(21)              countFreq.put(dimValue, new Integer(1));
(22)        } // end of for j
(23)        int pembanding = 0;
(24)        for (Enumeration e =
    countFreq.keys();e.hasMoreElements();){
(25)           String stringModus = (String) e.nextElement();
(26)           Integer count = (Integer) countFreq.get(stringModus);
(27)           int kandidatModus = count.intValue();
(28)           if (kandidatModus>pembanding){
(29)              pembanding = kandidatModus;
(30)              modus = stringModus;
(31)           }
(32)     }
(33)     countFreq.clear();
(34)     newMode.addElement(modus);
(35)     if (coloumn == (dim.size()-1)){
(36)        coloumn = 1;
(37)        KModesPoint m = new KModesPoint(newMode);
(38)        m.assignToCluster(i);
(39)        clusters[i].setMode(m);
(40)        newMode = new Vector();
(41)        newMode.addElement("updateModus");
(42)     }else{
(43)        coloumn++;
(44)        i = --i;
(45)     }
(46) }  // end of i
```

Segmen program 4.15 Method `updateModesFinal()` (lanjutan)

g) Kelas Informasi Klaster

Selama proses pembentukan klaster, informasi klaster yang terbentuk harus disimpan. Informasi klaster ini berupa posisi titik pusat klaster dan titik-titik

anggota klaster. Kelas yang digunakan untuk membentuk klaster disebut dengan kelas `Cluster`. Setiap method yang pada kelas `Cluster` ditunjukkan tabel 4.13.

Tabel 4.13 Penjelasan method kelas `Cluster()`.

Method	Keterangan
`Cluster()`	Konstruktor pada kelas `Cluster`
`setMode()`	Mengeset titik pusat/mode klaster
`getMode()`	Mendapatkan informasi mode klaster
`addClustermember()`	Menambahkan anggota klaster
`getClusterMember()`	Mendapatkan informasi angggota klaster
`setClusterNumber()`	Mengeset nomor klaster
`getClusterNumber()`	Mendapatkan informasi nomor klaster
`toString()`	Menampilkan anggota klaster

h) Kelas Perhitungan Kesalahan

Jika klaster sudah terbentuk maka untuk validasi hasil diperlukan informasi keakuratan hasil klaster Kelas yang digunakan untuk membentuk klaster disebut dengan kelas `ErrorCounts`. Setiap method yang pada kelas `ErrorCounts` ditunjukkan tabel 4.14. Method untuk menghitung kesalahan klaster ditunjukkan pada method pada segmen program 4.16.

Tabel 4.14 Penjelasan method kelas `ErrorCounts()`.

Method	Keterangan
`ErrorCounts()`	Konstruktor pada kelas `ErrorCounts`
`missClassCounts()`	Menghitung kesalahan klaster

```
(1)  Hashtable missClassCounts(){
(2)     String key = "";
(3)     Hashtable miss = new Hashtable();
(4)     for(int i=0;i<this.k;i++){
(5)        for(int j=0;j<this.k;j++){
(6)           key = ""+i+""+j;
(7)           miss.put(key,new Integer(0));
(8)        }
(9)     }
(10)    for(int i=0; i<this.result.size(); i++){
(11)       KModesPoint kmp = (KModesPoint)
        (this.result.elementAt(i));
(12)       Vector v = kmp.getPoint();
(13)       Integer index = (Integer)(v.elementAt(0));
(14)       for(int j=0; j<this.k; j++){
(15)          if (index.intValue() < this.jmlPerClass[j]){
(16)             int clustNum = kmp.getClusterNumber();
(17)             String value = ""+j+""+clustNum;
(18)             Integer counter = (Integer) miss.get(value);
(19)             int primCounter = counter.intValue()+1;
(20)             miss.put(value, new Integer(primCounter));
(21)             break;
(22)          }
(23)       }
(24)    }return miss;
(25) }
```

Segmen program 4.16 Method `missClassCounts()`

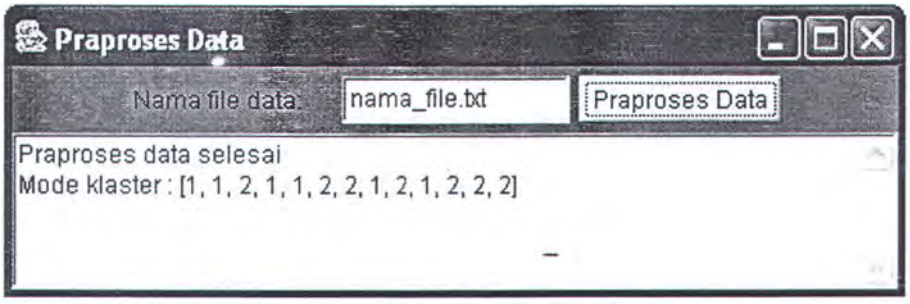
4.2.2 Implementasi Antar-muka

Sub bab berikut ini membahas tentang tahap implementasi dari desain yang diterangkan pada awal bab ini. Implementasi metode klasterisasi dilakukan dengan menggunakan bahasa pemrograman Java berbasis java.awt. Tahap implementasi ini menerangkan kelas-kelas terpenting dari perangkat lunak. Kelas yang diimplemetasikan tersebut meliputi praproses data, pembentukan data sub-sampel, proses optimasi titik pusat awal, dan proses pembentukan klaster berbasis partisi serta beberapa kelas bantuan. Antar-muka praproses data seperti

ditunjukkan pada gambar 4.23. Antar-muka pembentukan klaster seperti ditunjukkan pada gambar 4.24.

a) Implementasi antar-muka praproses data

Antar-muka praproses data diimplementasikan dengan satu label dan satu textfield yang digunakan sebagai parameter masukan nama file yang berisi data yang akan di proses. File masukan harus berekstensi .txt. Hasil implementasi antar-muka praproses data ditampilkan dalam text area seperti ditunjukkan pada gambar 4.24.



Gambar 4.24 Antar-muka praproses data

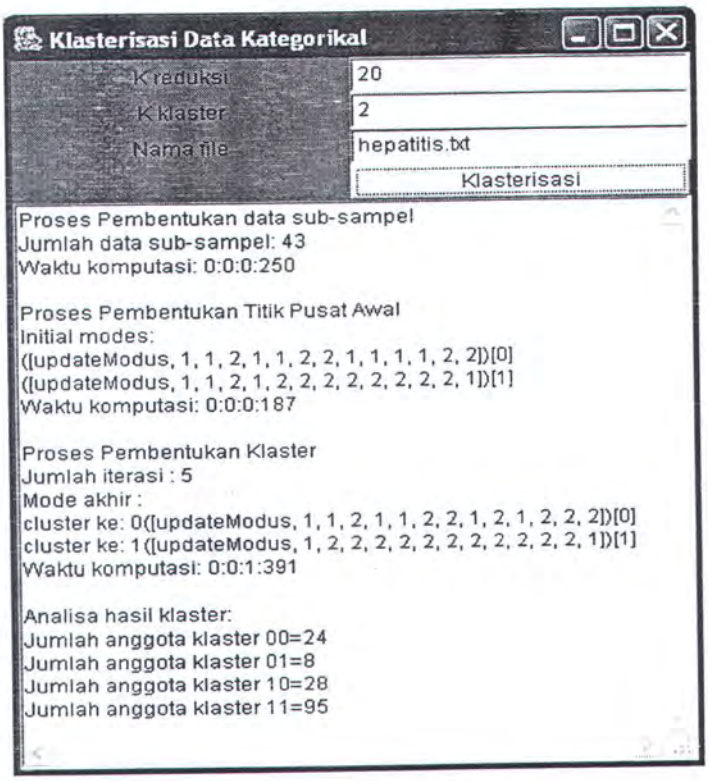
b) Implementasi antar-muka pembentukan klaster

Antar-muka pembentukan klaster membutuhkan beberapa parameter masukan dan diimplementasikan dalam bentuk komponen-komponen. Komponen yang terdapat pada dialog ini antara lain:

- 1 label dan satu textfield untuk parameter k reduksi
- 1 label dan satu textfield untuk parameter k klaster yang menunjukkan jumlah titik pusat awal yang harus dihasilkan dan jumlah klaster yang harus dibentuk

- 1 label dan satu textfield untuk data masukan berupa file yang berisi dataset
- 1 text aree untuk menampilkan informasi hasil klasterisasi

Hasil implementasi antar-muka praproses data seperti ditunjukkan pada gambar 4.25.



Gambar 4.25 Antar-muka pembentukan klaster

BAB 5

UJI COBA DAN EVALUASI

BAB 5

UJI COBA DAN EVALUASI

Dalam bab ini dijelaskan mengenai uji coba yang telah dilaksanakan. Pertama dibahas tentang lingkungan pelaksanaan uji coba dari aplikasi. Kemudian dijelaskan mengenai beberapa data transaksi yang digunakan dalam uji coba, dan diakhiri dengan pelaksanaan uji coba serta evaluasi hasil uji coba.

5.1 Lingkungan Uji Coba

Untuk mengevaluasi efektivitas dan efisiensi kinerja dari aplikasi, maka dilakukan pengujian dengan menggunakan beberapa skenario yang berbeda. Pengujian ini meliputi pengujian data kecil dan pengujian data besar. Semua eksperimen dilakukan pada sebuah Personal Computer dengan prosesor Intel ® Pentium ® 4 CPU 1.8GHz dan memori 1 GB RAM. Sistem operasi yang dijalankan adalah Microsoft Windows 2000 Professional. Sistem aplikasi klasterisasi data kategorikal diimplementasikan dengan bahasa pemrograman Java menggunakan kompiler jdk1.4.1.

5.2 Data Uji Coba

Dalam penelitian ini digunakan beberapa dataset yang semua atributnya berdomain kategorikal. Sehingga semua nilai atribut dalam dataset ini berupa nilai kategorikal. Dataset ujicoba diambil dari website yaitu dari UCI Machine

masing dataset. Keterangan lebih lanjut mengenai domain atribut dan nilai atribut bisa dilihat pada lampiran A.

Tabel 5.1. Karakteristik Dataset Masukan

Nama dataset	Jumlah Kelas	Jumlah Atribut	Jumlah Data
Soybean Disease	4	34	47
Congressional Votes	2	16	435
Wisconsin Breast Cancer	2	9	699
Breast Cancer	2	9	286
Hepatitis Domain	2	13	155

Dibawah ini diberikan informasi lebih detail mengenai dataset uji coba :

a. Soybean Disease

Data ini adalah data set standar yang sering digunakan dalam mengevaluasi metode-metode klasterisasi secara konseptual. Data ini disusun oleh Michalski dan Stepp, pada tahun 1983. Dataset ini tidak mengandung nilai atribut yang hilang. Tabel 5.2 adalah karakteristik dataset Soybean Disease dalam bentuk tabel.

Tabel 5.2. Karakteristik Data Soybean Disease

No	Nama Penyakit	Kode	Jumlah Data
1	Diaporte Stem Canker	D1	10
2	Charcoal Rot	D2	10
3	Rhizoctonia Root Rot	D3	10
4	Phytophthora Rot	D4	17
Jumlah data			47
Jumlah atribut			35

Mode konseptual pada dataset Soybean Disease ditunjukkan pada tabel 5.3. Masing-masing kelas direpresentasikan oleh satu buah mode yang memiliki 35 atribut. Tanda koma (,) pada atribut mode, misalnya 3,6 pada atribut pertama

atribut. Tanda koma (,) pada atribut mode, misalnya 3,6 pada atribut pertama kelas D1, menunjukkan bahwa pada atribut pertama mode kelas D1 bisa diwakili oleh nilai 3 atau 6 karena jumlah kemunculan nilai 3 dan 6 adalah sama. Nilai atribut dikodekan dengan angka bertipe integer.

Tabel 5.3. Mode konseptual dataset Soybean Disease

Kelas	Mode Konseptual
D1	3,6 0 2 1 0 1 0 1 0,1 2 1 1 0 2 2 0 0 0 1 0 3 1 1 1 0 0 0 0 4 0 0 0 0 0 0
D2	5,6 0 0 2 1 1,3 2,3 1 0,1 0 1 1 0 2 2 0 0 0 1 0 0 3 0 0 0 2 1 0 4 0 0 0 0 0 0
D3	0 1 2 0 0 0,3 1 1,2 0 1,2 1 0 0 2 2 0 0 0 1 0 1 1 0 1 0,1 0 0 3 4 0 0 0 0 0 0
D4	0,1 1 2 1 0 3 1 2 1 0 1 1 0 2 2 0 0 0 1 0 2 2 0 0 0 0 0 3 4 0 0 0 0 0 1

b. Congressional Votes

Berisi tentang hasil pelaksanaan United States Congressional Voting Records pada tahun 1984. Semua atribut bertipe boolean yang bernilai Y dan N. Dataset ini tidak mengandung nilai yang hilang. Karakteristik dataset Congressional Votes bisa dilihat pada tabel 5.4. Mode konseptual pada dataset Congressional Votes ditunjukkan pada tabel 5.5. Masing-masing kelas direpresentasikan oleh satu buah mode yang memiliki 16 atribut.

Tabel 5.4. Karakteristik dataset Congressional Votes

No	Nama Penyakit	Jumlah Data
1	Democrats	267
2	Republicans	168
Jumlah data		435
Jumlah atribut		16

Tabel 5.5. Mode konseptual dataset Soybean Disease

Kelas	Mode Konseptual
Democrats	y, y, y, n, n, n, y, y, y, n, y, n, n, n, y, y
Republicans	n, y, n, y, y, y, n, n, n, y, n, y, y, y, n, y

c. Wisconsin Breast Cancer

Dataset ini berasal dari University of Wisconsin Hospitals, Madison dan megandung nilai atribut yang hilang. Karakteristik dataset Breast Cancer bisa dilihat pada tabel 5.6. Mode konseptual pada dataset Wisconsin Breast Cancer ditunjukkan pada tabel 5.7. Masing-masing kelas direpresentasikan oleh satu buah mode yang memiliki 9 atribut.

Tabel 5.6. Karakteristik dataset Wisconsin Breast Cancer

No	Nama Penyakit	Jumlah Data
1	Benign	458
2	Malignant	241
Jumlah data		699
Jumlah atribut		9

Tabel 5.7. Mode konseptual dataset Wisconsin Breast Cancer

Kelas	Mode Konseptual
Benign	1, 1, 1, 1, 2, 1, 2, 1, 1
Malignant	10, 10, 10, 10, 3, 10, 7, 10, 1

d. Hepatitis Domain

Dataset ini mengandung nilai atribut yang hilang. Karakteristik dataset Hepatitis Domain bisa dilihat pada tabel 5.8. Mode konseptual pada dataset Hepatitis Domain ditunjukkan pada tabel 5.9. Masing-masing kelas direpresentasikan oleh satu buah mode yang memiliki 13 atribut.

Tabel 5.8. Karakteristik dataset Hepatitis Domain

No	Nama Penyakit	Jumlah Data
1	Die	32
2	Live	123
Jumlah data		155
Jumlah atribut		13

Tabel 5.9. Mode konseptual dataset Hepatitis Domain

Kelas	Mode Konseptual
Die	1, 1, 2, 1, 1, 2, 2, 2, 2, 1, 2, 2, 2
Live	1, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 1

e. Breast Cancer

Dataset Breast Cancer diambil terdiri dari 286 data dengan 201 data termasuk kategori no-recurrence-events dan 85 data termasuk kategori recurrence-events. Tiap data dipresentasikan oleh 9 atribut. Dataset ini mengandung nilai atribut yang hilang Karakteristik dataset Breast Cancer bisa dilihat pada tabel 5.10. Mode konseptual dataset Breast Cancer ditunjukkan pada tabel 5.11. Masing-masing kelas direpresentasikan oleh satu buah mode yang memiliki 9 atribut.

Tabel 5.10. Karakteristik dataset Breast Cancer

Nama Penyakit	Jumlah Data
no-recurrence-events	201
recurrence-events	85
Jumlah data	286
Jumlah atribut	9

Tabel 5.11. Mode konseptual dataset Breast Cancer

Kelas	Mode Konseptual
no-recurrence-events	50-59, premeno, 25-29, 0-2, no, 2, left, left_low, no
recurrence-events	40-49, premeno, 30-34, 0-2, no, 3, left, left_low, no

5.3 Skenario Uji Coba

Untuk mengevalusi algoritma klasterisasi yang telah didesain dan diimplementasikan maka dilakukan uji coba dan evaluasi. Uji coba dan evaluasi dibagi menjadi tiga topik utama, yaitu yang bertujuan untuk menguji pengaruh parameter, mengukur akurasi dan yang bertujuan untuk mengukur kinerja.

5.3.1 Uji Pengaruh Parameter

Uji pengaruh parameter dilakukan untuk mengevaluasi pengaruh parameter yang diperlukan oleh algoritma reduksi data dalam menghasilkan klaster. Dalam hal ini adalah tingkat akurasi hasil klaster.

5.3.2 Uji Akurasi

Uji keakuratan hasil klaster dibagi menjadi dua yaitu menjalankan algoritma tanpa reduksi data (HPTR) dan dengan reduksi data (HPR). Hal ini untuk mengevaluasi apakah hasil klaster dengan reduksi dan tanpa reduksi menghasilkan klaster yang sama (stabil). Keakuratan hasil klaster dihitung untuk mengukur tingkat akurasi klaster. Kesalahan klaster terjadi apabila lagoritma menempatkan titik pada klaster yang berbeda dengan klaster yang ada pada dataset. Untuk mengevaluasi tingkat akurasi hasil klaster digunakan persamaan (5.1) dimana r adalah tingkat akurasi hasil klaster, n adalah jumlah data dalam data set, dan ai adalah jumlah data pada klaster yang benar

$$r = \frac{\sum_{i=1}^{k} a_i}{n} \tag{5.1}$$

Sedang tingkat kesalahan hasil klaster (e) dihitung dengan persamaan:

$$e = 1 - r \tag{5.2}$$

5.3.3 Uji Kinerja

Uji kinerja dilakukan dengan cara menghitung waktu komputasi algoritma yang dijalankan.

5.3.4 Uji Perbandingan Algoritma

Dilakukan uji coba dataset masukan dengan menggunakan algoritma K-Modes (KM). Hasil klaster akan dibandingkan dengan algoritma perbaikan penentuan titik pusat awal yang telah didesain dan diimplementasikan.

Evaluasi keakuratan hasil klaster digunakan sebagai evaluasi pengukuran tingkat akurasi algoritma, sedang kecepatan waktu komputasi digunakan sebagai pengukuran kinerja algoritma. Hasil evaluasi terhadap algoritma perbaikan penentuan titik pusat awal berbasis hirarki yang selanjutnya disebut dengan algoritma Hirarki Partisi Reduksi (HPR) akan dibandingkan dengan algoritma Hirarki Partisi Tanpa Reduksi (HPTR), K-Modes (KM), dan Iterative K-Modes (IKM). Hasil HPR dibedakan menjadi dua yaitu HPR yang dijalankan dengan nilai k reduksi bernilai bervariasi (HPR-V) dan HPR yang dijalankan dengan nilai k reduksi bernilai tetap, yaitu ketika proses reduksi data dengan nilai k reduksi tertentu sudah mendapatkan hasil reduksi data yang mempunyai mode tetap (HPR-S). Alasan algoritma HPR dibagi dua ini adalah untuk menguji pengaruh

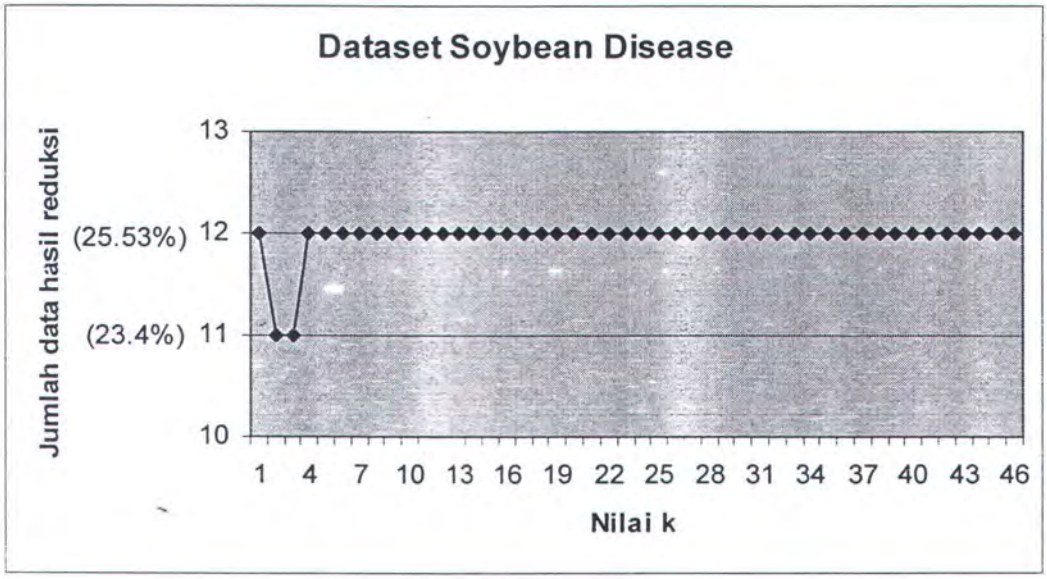
pemilihan nilai parameter k-reduksi pada algoritma HPR dan untuk mengukur waktu komputasi, dimana dalam mengukur komputasi harus digunakan k reduksi yang tetap.

5.4 Pelaksanaan Uji Coba

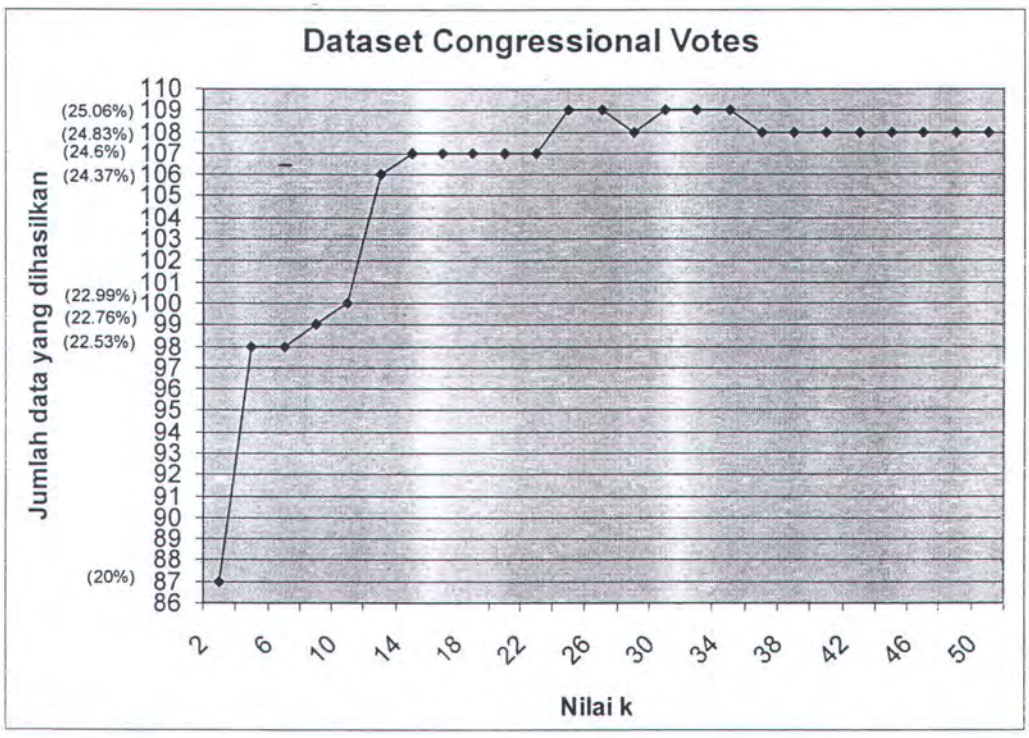
Uji coba dilakukan pada dataset masukan yang telah diuraikan pada sub bab sebelumnya. Pada masing-masing dataset dilakukan empat kali menjalankan algoritma yaitu algoritma HPR-V, HPR-S, HPTR dan K-Modes.

5.4.1 Uji Pengaruh Parameter

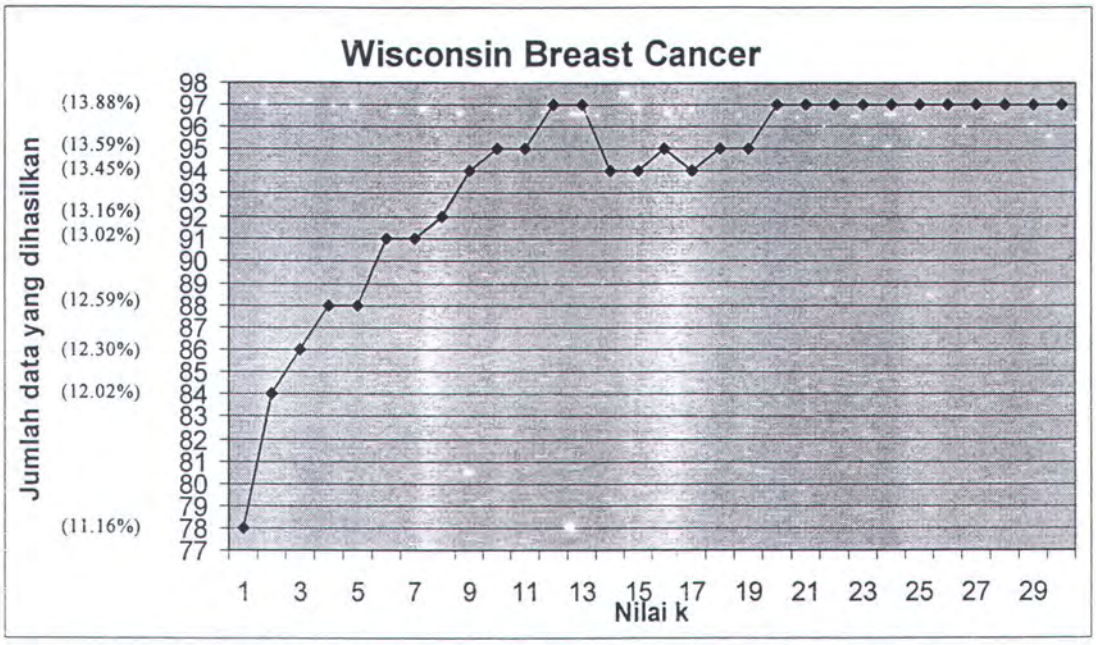
Untuk melihat pengaruh nilai k reduksi maka pada tiap-tiap dataset dilakukan uji coba dengan memberikan nilai parameter k reduksi yang bervariasi sehingga akan diketahui nilai k reduksi mana yang bisa menghasilkan data tereduksi yang mempunyai jumlah data tereduksi stabil. Dari beberapa kali menjalankan algoritma reduksi data maka hasil 20 kali menjalankan dengan nilai k reduksi bervariasi akan ditampilkan pada gambar. Gambar 5.1 sampai Gambar 5.5 berturut-turut menunjukkan jumlah data yang dihasilkan untuk 20 kali menjalankan reduksi data dataset Soybean Disease, Congressional Votes, Wisconsin Breast Cancer, Hepatitis Domain, dan Breast Cancer.



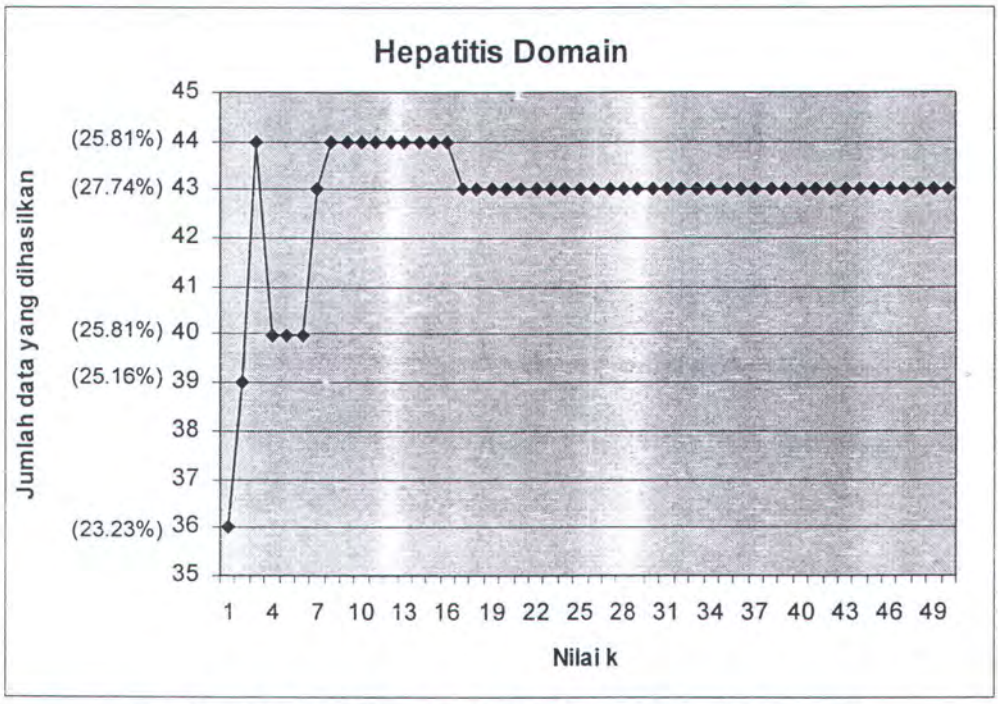
Gambar 5.1 Grafik hasil reduksi dataset Soybean Disease



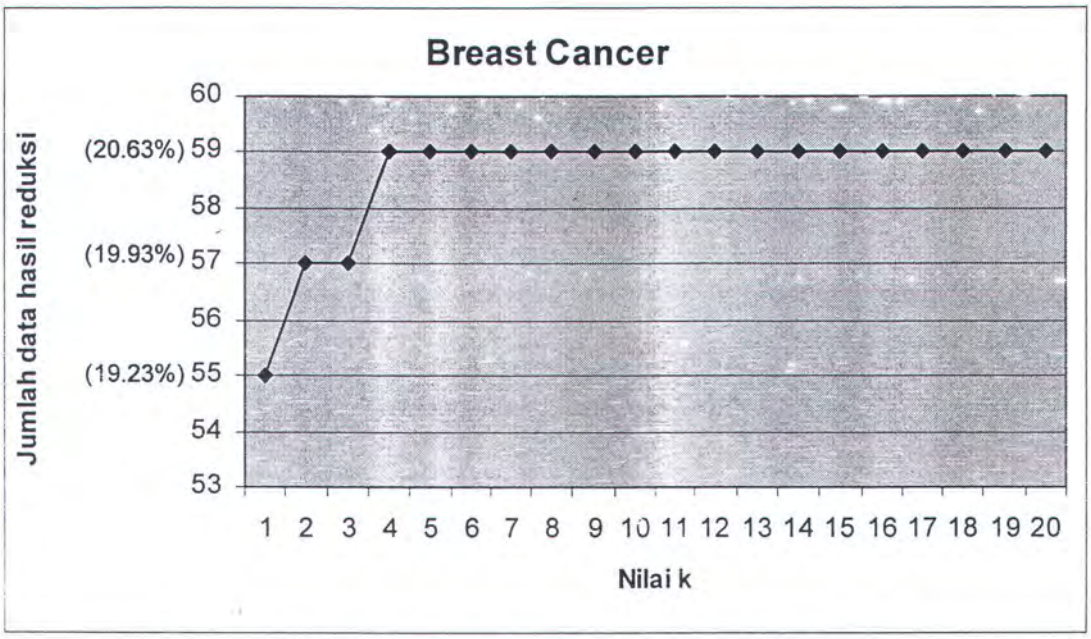
Gambar 5.2 Grafik hasil reduksi dataset Congressional Votes



Gambar 5.3 Grafik hasil reduksi dataset Wisconsin Breast Cancer



Gambar 5.4 Grafik hasil reduksi dataset Hepatitis Domain



Gambar 5.5 Grafik hasil reduksi dataset Breast Cancer

Selanjutnya pada mulai nilai k reduksi stabil tersebut akan ditentukan lagi k reduksi yang menghasilkan nilai mode yang stabil. Pada k reduksi yang menghasilkan nilai mode stabil tersebutlah yang akan digunakan untuk menjalankan algoritma HPR-S. Untuk tiap data set akan ditampilkan nilai mode dari 20 kali menjalankan reduksi data. Tabel 5.12 sampai Tabel 5.16 berturut-turut menunjukkan 10 mode data yang dihasilkan dengan nilai k reduksi yang berbeda mulai dari k reduksi yang memberikan jumlah data sub-sampel stabil untuk dataset Soybean Disease, Congressional Votes, Wisconsin Breast Cancer, Hepatitis Domain, dan Breast Cancer. Baris yang dicetak tebal pada tabel-tabel tersebut menunjukkan bahwa pada nilai k reduksi baris yang dicetak tebal, mode yang dihasilkan mulai stabil.

Tabel 5.12. Mode stabil data sub-sampel dataset Soybean Disease

Kred	Mode Dataset Soybean Disease
4	4, 1, 2, 1, 0, 1, 1, 1, 0, 2, 1, 1, 0, 2, 2, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 3, 4, 0, 0, 0, 0, 0, 0
5	4, 1, 2, 1, 0, 1, 1, 1, 0, 2, 1, 1, 0, 2, 2, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 3, 4, 0, 0, 0, 0, 0, 0
6	4, 1, 2, 1, 0, 1, 1, 2, 0, 2, 1, 1, 0, 2, 2, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 3, 4, 0, 0, 0, 0, 0, 0
7	**4, 1, 2, 1, 0, 1, 1, 1, 0, 2, 1, 1, 0, 2, 2, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 3, 4, 0, 0, 0, 0, 0, 0**
8	4, 1, 2, 1, 0, 1, 1, 1, 0, 2, 1, 1, 0, 2, 2, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 3, 4, 0, 0, 0, 0, 0, 0
9	4, 1, 2, 1, 0, 1, 1, 1, 0, 2, 1, 1, 0, 2, 2, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 3, 4, 0, 0, 0, 0, 0, 0
10	4, 1, 2, 1, 0, 1, 1, 1, 0, 2, 1, 1, 0, 2, 2, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 3, 4, 0, 0, 0, 0, 0, 0
11	4, 1, 2, 1, 0, 1, 1, 1, 0, 2, 1, 1, 0, 2, 2, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 3, 4, 0, 0, 0, 0, 0, 0
12	4, 1, 2, 1, 0, 1, 1, 1, 0, 2, 1, 1, 0, 2, 2, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 3, 4, 0, 0, 0, 0, 0, 0

Tabel 5.13. Mode stabil data sub-sampel dataset Congressional Votes

Kred	Mode Dataset Congressional Votes
36	n, y, y, n, n, y, y, y, y, y, n, n, n, y, n, y
37	n, y, y, n, n, y, y, y, y, y, n, n, n, y, n, y
38	n, y, y, n, n, y, y, y, y, y, n, n, n, y, n, y
39	**n, n, y, n, n, y, y, y, y, n, n, n, n, y, n, y**
40	n, n, y, n, n, y, y, y, y, n, n, n, n, y, n, y
41	n, n, y, n, n, y, y, y, y, n, n, n, n, y, n, y
42	n, n, y, n, n, y, y, y, y, n, n, n, n, y, n, y
43	n, n, y, n, n, y, y, y, y, n, n, n, n, y, n, y
44	n, n, y, n, n, y, y, y, y, n, n, n, n, y, n, y
45	n, n, y, n, n, y, y, y, y, n, n, n, n, y, n, y

Tabel 5.14. Mode stabil data sub-sampel dataset Wisconsin Breast Cancer

Kred	Mode Dataset Wisconsin Breast Cancer
20	**1, 1, 1, 1, 2, 1, 2, 1, 1**
21	1, 1, 1, 1, 2, 1, 2, 1, 1
22	1, 1, 1, 1, 2, 1, 2, 1, 1
23	1, 1, 1, 1, 2, 1, 2, 1, 1
24	1, 1, 1, 1, 2, 1, 2, 1, 1
25	1, 1, 1, 1, 2, 1, 2, 1, 1
26	1, 1, 1, 1, 2, 1, 2, 1, 1
27	1, 1, 1, 1, 2, 1, 2, 1, 1
28	1, 1, 1, 1, 2, 1, 2, 1, 1
29	1, 1, 1, 1, 2, 1, 2, 1, 1

Tabel 5.15. Mode stabil data sub-sampel dataset Hepatitis Domain

No	Mode Dataset Hepatitis Domain
17	**1, 1, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 1**
18	1, 1, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 1
19	1, 1, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 1
20	1, 1, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 1
21	1, 1, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 1
22	1, 1, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 1
23	1, 1, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 1
24	1, 1, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 1
25	1, 1, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2
26	1, 1, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 3

Tabel 5.16. Mode stabil data sub-sampel dataset Breast Cancer

Kred	Mode Dataset Breast Cancer
4	50-59, premeno, 25-29, 0-2, no, 2, right, left_up, no
5	50-59, premeno, 20-24, 0-2, no, 2, right, left_up, no
6	50-59, premeno, 20-24, 0-2, no, 2, right, left_up, no
7	50-59, premeno, 20-24, 0-2, no, 2, left, left_up, no
8	**50-59, premeno, 20-24, 0-2, no, 2, left, left_low, no**
9	50-59, premeno, 20-24, 0-2, no, 2, left, left_low, no
10	50-59, premeno, 20-24, 0-2, no, 2, left, left_low, no
11	50-59, premeno, 20-24, 0-2, no, 2, left, left_low, no
12	50-59, premeno, 20-24, 0-2, no, 2, left, left_low, no
13	50-59, premeno, 20-24, 0-2, no, 2, left, left_low, no

Rangkuman data hasil uji coba terhadap semua dataset bisa dilihat pada tabel 5.17. Singkatan KRJS adalah nilai k reduksi yang menghasilkan jumlah data stabil, KRMS adalah nilai k reduksi yang menghasilkan mode data stabil, sedang P adalah prosentase data sub-sampel yang dihasilkan. Nilai k reduksi pada KRMS inilah yang nanti akan dipakai sebagai parameter HPR-S.

Tabel 5.17. Pengaruh parameter k reduksi

Dataset	KRJS	KRMS	P(%)
Soybean Disease	4	7	25.53
Congressional Votes	36	39	24.83
Wisconsin Breast Cancer	20	20	13.88
Hepatitis Domain	17	17	27.74
Breast Cancer	4	8	20.63

5.4.2 Uji Akurasi

Uji akurasi dilakukan pada semua dataset. Untuk mendapatkan hasil klaster yang akurat maka tiap uji coba dilakukan 20 kali baik untuk HPTR maupun untuk HPR dengan nilai k reduksi bervariasi (HPR-V). Nilai parameter k klaster adalah sejumlah kelas dataset masing-masing.

5.4.2.1 Uji Coba Dengan Mereduksi Data (HPR)

Uji coba ini dijalankan dengan menjalankan algoritma HPR-V. Hasil uji coba untuk tiap dataset adalah sebagai berikut:

a) Dataset Soybean Disease

Tabel 5.18. adalah hasil menjalankan HPRV. Tabel 5.19. menunjukkan matrik misklasifikasi antara klaster yang terbentuk dengan kelas penyakit pada dataset Soybean Disease. Sedang Tabel 5.20. menunjukkan perbandingan mode hasil algoritma HPR-V dengan mode konseptual. Waktu komputasi ditunjukkan pada tabel 5.21.

Tabel 5.18. Hasil algoritma HPR-V dataset Soybean Disease

Masukan	Parameter k reduksi data	1 s/d 20			
	Parameter k klaster	4			
Hasil		**Min**	**Mak**	**Rerata**	**SD**
	Jumlah iterasi	2	2	2	0
	Jumlah misklasifikasi	0	0	0	0
	Tingkat akurasi (%)	100	100	100	0
	Beda atribut mode	0	0	0	0

Tabel 5.19. Misklasifikasi hasil algoritma HPR-V dataset Soybean Disease

	C1	C2	C3	C4	Jumlah
D1	10				10
D2		10			10
D3			10		10
D4				17	17
Jumlah					47

Tabel 5.20. Perbandingan mode hasil algoritma HPR-V dengan mode konseptual dataset Soybean Disease

Kel		**Mode**	**Beda**
D1	Ksp	3,6 0 2 1 0 1 0 1 0,1 2 1 1 0 2 2 0 0 0 1 0 3 1 1 1 0 0 0 0 4 0 0 0 0 0 0	0
	HPR-V	6 0 2 1 0 1 0 1 1 2 1 1 0 2 2 0 0 0 1 0 3 1 1 1 0 0 0 0 4 0 0 0 0 0 0	
D2	Ksp	5,6 0 0 2 1 1,3 2,3 1 0,1 0 1 1 0 2 2 0 0 0 1 0 0 3 0 0 0 2 1 0 4 0 0 0 0 0 0	0
	HPR-V	6 0 0 2 1 3 3 1 1 0 1 1 0 2 2 0 0 0 1 0 0 3 0 0 0 2 1 0 4 0 0 0 0 0 0	
D3	Ksp	0 1 2 0 0 0,3 1 1,2 0 1,2 1 0 0 2 2 0 0 0 1 0 1 1 0 1 0,1 0 0 3 4 0 0 0 0 0 0	0
	HPR-V	0 1 2 0 0 3 1 2 0 2 1 0 0 2 2 0 0 0 1 0 1 1 0 1 1 0 0 3 4 0 0 0 0 0 0	
D4	Ksp	0,1 1 2 1 0 3 1 2 1 0 1 1 0 2 2 0 0 0 1 0 2 2 0 0 0 0 0 3 4 0 0 0 0 0 1	0
	HPR-V	1 1 2 1 0 3 1 2 1 0 1 1 0 2 2 0 0 0 1 0 2 2 0 0 0 0 0 3 4 0 0 0 0 0 1	

Tabel 5.21. Waktu komputasi algortima HPR-V dataset Soybean Disease

Fase	**Waktu Komputasi (detik)**				
	Minimum	**Maksimum**	**Mean**	**Total**	**SD**
Red	0.031	0.047	0.0368	0.1642	0.0076
Opt	0.031	0.047	0.0406		0.0077
Klast	0.078	0.094	0.0868		0.0080

b) Dataset Congressional Votes

Tabel 5.22 adalah hasil menjalankan HPRV. Tabel 5.23 menunjukkan matrik misklasifikasi antara klaster yang terbentuk dengan kelas pada dataset Congressional Votes. Sedang tabel 5.24 menunjukkan perbandingan mode hasil algoritma HPR-V dengan mode konseptual. Waktu komputasi ditunjukkan pada tabel 5.25.

Tabel 5.22. Hasil algoritma HPR-V dataset Congressional Votes

Masukan	k reduksi	1 s/d 20, 30, 40, 50, 60			
	k klaster	2			
		Min	**Mak**	**Rerata**	**SD**
	Jumlah iterasi	4	5	4.95	0.2236
	Jumlah	58	60	59.9	0.4472
	Tingkat akurasi (%)	86.21	86.67	86.23	0.0010
	Beda atribut mode	2	4	3.9	0.4472

Tabel 5.23. Misklasifikasi hasil algoritma HPR-V dengan tingkat akurasi 86.67% (k reduksi=3) dataset Congressional Votes

	C1	C2	Total
Democrats	217	50	267
Republicans	8	160	168
Jumlah data			435

Tabel 5.24. Perbandingan mode hasil algoritma HPR-V dengan tingkat akurasi 86.67% (k reduksi=3) dengan mode konseptual dataset Congressional Votes

Kelas		**Mode**	**Beda**
Democrats	Ksp	{y, y, y, n, n, n, y, y, y, n, y, n, n, n, y, y}	2
	HP	{y, **n**, y, n, n, n, y, y, y, n, **n**, n, n, n, y, y}	
Republicans	Ksp	{n, y, n, y, y, y, n, n, n, y, n, y, y, y, n, y}	0
	HP	{n, y, n, y, y, y, n, n, n, y, n, y, y, y, n, y }	

Tabel 5.25. Waktu komputasi algortima HPR-V dataset Congressional Votes

Waktu Komputasi (detik)					
Fase	**Minimum**	**Maksimum**	**Rerata**	**Total**	**Std**
Red	2.687	4.594	3.6837	9.439	0.5181
Opt	1.266	2.984	2.1071		0.4389
Klast	2.860	3.985	3.6487		0.4232

c) Dataset Wisconsin Breast Cancer

Tabel 5.26 adalah hasil menjalankan HPRV. Tabel 5.27 menunjukkan matrik misklasifikasi antara klaster yang terbentuk dengan kelas pada dataset Wisconsin Breast Cancer. Sedang tabel 5.28 menunjukkan perbandingan mode hasil algoritma HPR-V dengan mode konseptual. Waktu komputasi ditunjukkan pada tabel 5.29.

Tabel 5.26. Hasil algoritma HPR-V dataset Wisconsin Breast Cancer

	k reduksi	1 s/d 20			
	k klaster	2			
		Min	**Mak**	**Rerata**	**SD**
Masukan	Jumlah iterasi	4	4	4	0
	Jumlah misklasifikasi	95	95	95	0
	Tingkat akurasi (%)	86.41	86.41	86.41	0
	Beda atribut mode	1	1	1	0

Tabel 5.27. Misklasifikasi hasil algoritma HPR-V dataset Wisconsin Breast Cancer

	C1	C2	Total
Benign	455	3	458
Malignant	92	149	241
Jumlah data			699

Tabel 5.28. Perbandingan mode hasil algoritma HPR-V dengan mode konseptual dataset Wisconsin Breast Cancer

Kelas		Mode	Beda
Benign	Ksp	1, 1, 1, 1, 2, 1, 2, 1, 1	0
	HP	1, 1, 1, 1, 2, 1, 2, 1, 1	
Malignant	Ksp	10, 10, 10, 10, 3, 10, 7, 10, 1	1
	HP	10, 10, 10, 10, **4**, 10, 7, 10, 1	

Tabel 5.29. Waktu komputasi HPR-V dataset Wisconsin Breast Cancer

Waktu Komputasi (detik)					
Fase	**Minimum**	**Maksimum**	**Rerata**	**Total**	**SD**
Red	9.766	11.641	7.3685	11.2731	0.5853
Opt	1.515	1.844	1.1917		0.0928
Klast	3.390	4.766	2.7145		0.5351

d) Dataset Hepatitis Domain

Tabel 5.30 adalah hasil menjalankan HPRV. Tabel 5.31 menunjukkan matrik misklasifikasi antara klaster yang terbentuk dengan kelas pada dataset Hepatitis Domain. Sedang tabel 5.32 menunjukkan perbandingan mode hasil algoritma HPR-V dengan mode konseptual. Waktu komputasi ditunjukkan pada tabel 5.33.

Tabel 5.30. Hasil algoritma HPR-V dataset Hepatitis Domain

Masukan	k reduksi	1 s/d 20			
	k klaster	2			
		Min	**Mak**	**Rerata**	**SD**
	Jumlah iterasi	4	6	5	0.3244
	Jumlah misklasifikasi	36	37	36.05	0.2236
	Tingkat akurasi (%)	76.13	76.77	76.74	0.1431
	Beda atribut mode	1	2	1.1	0.3078

Tabel 5.31. Perbandingan mode hasil algoritma HPR-V dengan rasio kebenaran 76.77% dengan mode konseptual dataset Hepatitis Domain

Kelas		Mode	Beda
Die	Ksp	1, 1, 2, 1, 1, 2, 2, 2, 2, 1, 2, 2, 2	1
	HPR-V	1, 1, 2, 1, 1, 2, 2, **1**, 2, 1, 2, 2, 2	
Live	Ksp	1, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 1	0
	HPR-V	1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1	

Tabel 5.32. Misklasifikasi hasil algoritma HPR-V dengan rasio kebenaran 76.77% dataset Hepatitis Domain

	C1	C2	Total
Die	8	24	32
Live	95	28	123
Jumlah data			155

Tabel 5.33 Waktu komputasi algortima HPR-V dataset Hepatitis Domain

Waktu Komputasi (detik)					
Fase	**Min**	**Mak**	**Mean**	**Total**	**SD**
Red	0.172	0.219	0.193	0.812	0.0116
Opt	0.125	0.203	0.201		0.0864
Klast	0.328	0.531	0.418		0.0344

e) Dataset Breast Cancer

Tabel 5.34 adalah hasil menjalankan HPRV. Tabel 5.35 menunjukkan matrik misklasifikasi antara klaster yang terbentuk dengan kelas pada dataset Breast Cancer. Sedang tabel 5.36 menunjukkan perbandingan mode hasil algoritma HPR-V dengan mode konseptual. Waktu komputasi ditunjukkan pada tabel 5.37.

Tabel 5.34. Hasil algoritma HPR-V dataset Breast Cancer

Masukan	k reduksi	1 s/d 20			
	k klaster	2			
		Min	**Mak**	**Rerata**	**SD**
	Jumlah iterasi	4	7	6.5	0.8885
	Jumlah misklasifikasi	119	134	122.15	6.0981
	Tingkat akurasi (%)	53.15	58.39	57.29	0.0213
	Beda atribut mode	5	6	5.7	0.4702

Tabel 5.35. Misklasifikasi hasil algoritma HPR-V dengan tingkat akurasi 58.39% dataset Breast Cancer

	C1	C2	Total
no-recurrence-events	126	75	201
recurrence-events	44	41	85
Jumlah data			286

Tabel 5.36 Perbandingan mode hasil algoritma HPR-V dengan tingkat akurasi 58.39 % dengan mode konseptual dataset Breast Cancer

Kelas		**Mode hasil Hirarki Partisi**	**Beda**
no-recurrence-events	Ksp	50-59, premeno, 25-29, 0-2, no, 2, left, left_low, no	2
	HP	**40-49**, premeno, **30-34**, 0-2, no, 2, left, left_low, no	
recurrence-events	Ksp	40-49, premeno, 30-34, 0-2, no, 3, left, left_low, no	4
	HP	**50-59, ge40**, 30-34, 0-2, no, 3, **right, left_up**, no	

Tabel 5.37. Waktu komputasi algortima HPR-V dataset Breast Cancer

Waktu Komputasi (detik)					
Fase	**Minimum**	**Maksimum**	**Mean**	**Total**	**SD**
Red	1.781	1.938	1.851	3.4921	0.0421
Opt	0.218	0.250	0.241		0.0107
Klast	1.093	2.125	1.401		0.4756

5.4.2.2 Uji Coba Tanpa Reduksi Data (HPTR)

Uji coba ini dijalankan dengan menjalankan algoritma HPTR. Hasil uji coba untuk tiap dataset adalah sebagai berikut:

a) Dataset Soybean Disease

Hasil uji coba algoritma HPTR sama dengan uji coba algoritma HPR-V sebagaimana ditunjukkan pada Tabel 5.18 sampai dengan Tabel 5.20. Waktu komputasi ditunjukkan pada Tabel 5.38.

Tabel 5.38. Waktu komputasi algortima HPTR dataset Soybean Disease

Fase	Waktu Komputasi (detik)				
	Minimum	Maksimum	Mean	Total	SD
Opt	0.468	0.485	0.4711	0.555	0.005889
Klast	0.063	0.094	0.0836		0.010425

b) Dataset Congressional Votes

Hasil klaster ditunjukkan pada Tabel 5.39. Tabel 5.40 menunjukkan misklasifikasi. Table 5.41 menunjukkan perbandingan mode hasil dengan mode konseptual dataset Congressional Votes. Hasil pengukuran waktu komputasi sebagaimana ditunjukkan pada Tabel 5.42.

Tabel 5.39. Hasil algoritma HPTR dataset Congressional Votes

Masukan	k-klaster	2
	Jumlah misklasifikasi	60
	Tingkat akurasi	86.21%
	Jumlah iterasi	5
	Beda atribut mode	4

Tabel 5.40. Misklasifikasi algoritma HPTR dataset Congressional Votes

	C1	C2	Jumlah
Democrats	218	49	267
Republicans	11	157	168
Jumlah data			435

Tabel 5.41. Perbandingan mode hasil HPTR dengan mode konseptual dataset Congressional Votes

Kelas		Mode	Beda
Democrats	Ksp	{y, y, y, n, n, n, y, y, y, n, y, n, n, n, y, y}	3
	HP	{y, **n**, y, n, n, n, y, y, y, **y**, **n**, n, n, n, y, y}	
Republicans	Ksp	{n, y, n, y, y, y, n, n, n, y, n, y, y, y, n, y}	1
	HP	{n, y, n, y, y, y, n, n, n, **n**, n, y, y, y, n, y }	

Tabel 5.42 Waktu komputasi algortima HPTR dataset Congressional Votes

Fase	Waktu Komputasi (detik)				
	Minimum	Maksimum	Mean	Total	SD
Opt	122.531	128.563	126.318	129.770	1.6364
Klast	3.000	3.985	3.452		0.4247

c) Dataset Wisconsin Breast Cancer

Klaster yang dihasilkan sama dengan uji coba menjalankan algoritma HPR-V yang ditunjukkan pada Tabel 5.26 sampai dengan 5.28. Hasil pengukuran waktu komputasi sebagaimana ditunjukkan pada tabel 5.43.

Tabel 5.43. Waktu komputasi algortima HPTR dataset Wisconsin Breast Cancer

Fase	Waktu Komputasi (detik)				
	Minimum	Maksimum	Rerata	Total	SD
Opt	246.172	256.484	249.2711	253.530	2.1228
Klast	3.406	4.844	4.2589		0.4388

d) Dataset Hepatitis Domain

Hasil klaster ditunjukkan pada Tabel 5.44. Tabel 5.45 menunjukkan misklasifikasi algoritma HPTR. Table 5.46 menunjukkan perbandingan mode hasil HPTR dengan mode konseptual dataset Hepatitis Domain. Hasil pengukuran waktu komputasi sebagaimana ditunjukkan pada Tabel 5.47.

Tabel 5.44. Hasil algoritma HPTR dataset Hepatitis Domain

Masukan	k-klaster	2
Hasil	Jumlah misklasifikasi	36
	Tingkat akurasi	76.77%
	Jumlah iterasi	5
	Beda atribut mode	2

Tabel 5.45. Misklasifikasi algoritma HPTR dataset Hepatitis Domain

	C1	C2	Total
Die	24	8	32
Live	28	95	312
Jumlah data			155

Tabel 5.46. Perbandingan mode hasil algoritma HPTR dengan mode konseptual dataset Hepatitis Domain

Kelas		**Mode**	**Beda**
Die	Ksp	1, 1, 2, 1, 1, 2, 2, 2, 2, 1, 2, 2, 2	1
	HPR-S	1, 1, 2, 1, 1, 2, 2, **1**, 2, 1, 2, 2, 2	
Live	Ksp	1, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 1	1
	HPR-S	1, 2, 2, **2**, 2, 2, 2, 2, 2, 2, 2, 2, 1	

Tabel 5.47. Waktu komputasi algortima HPTR dataset Hepatitis Domain

Fase	**Waktu Komputasi (detik)**				
	Minimum	**Maksimum**	**Mean**	**Total**	**SD**
Opt	5.312	6.579	6.179	6.6602	0.4888
Klast	0.406	0.453	0.423		0.0139

e) Dataset Breast Cancer

Hasil klaster ditunjukkan pada Tabel 5.48. Tabel 5.49 menunjukkan misklasifikasi algoritma HPTR. Table 5.50 menunjukkan perbandingan mode hasil HPTR dengan mode konseptual dataset Breast Cancer. Hasil pengukuran waktu komputasi sebagaimana ditunjukkan pada Tabel 5.51.

Tabel 5.48. Hasil algoritma HPTR dataset Breast Cancer

Masukan	Parameter k klaster	2
Hasil	Jumlah iterasi	6
	Jumlah misklasifikasi	121
	Tingkat akurasi	57.69%
	Beda atribut mode	5

Tabel 5.49. Misklasifikasi hasil klaster algoritma HPTR dataset Breast Cancer

	C1	C2	Total
no-recurrence-events	124	77	201
recurrence-events	44	41	85
Jumlah data			286

Tabel 5.50. Perbandingan mode hasil algoritma HPTR dengan mode konseptual dataset Breast Cancer

Kelas		**Mode hasil Hirarki Partisi**	**Beda**
no-recurrence-events	Ksp	50-59, premeno, 25-29, 0-2, no, 2, left, left_low, no	1
	HP	**40-49**, premeno, 25-29, 0-2, no, 2, left, left_low, no	
recurrence-events	Ksp	40-49, premeno, 30-34, 0-2, no, 3, left, left_low, no	4
	HP	**50-59**, **ge40**, 30-34, 0-2, no, 3, **right**, **left_up**, no	

Tabel 5.51. Waktu komputasi algortima HPTR dataset Breast Cancer

Fase	**Waktu Komputasi (detik)**				
	Minimum	**Maksimum**	**Mean**	**Total**	**SD**
Opt	20.562	22.610	21.504	23.128	0.5187
Klast	1.000	1.985	1.624		0.4634

5.4.3 Uji Kinerja

Untuk mengukur waktu komputasi maka digunakan parameter k reduksi yang tetap. Nilai k reduksi yang dipilih adalah nilai k reduksi yang menghasilkan mode stabil sesuai data pada Tabel 5.17. Supaya mendapatkan hasil yang lebih akurat maka uji coba dilakukan sebanyak 20 kali dengan parameter k reduksi yang tetap (HPR-S).

a) Dataset Soybean Disease

Nilai k reduksi yang dipakai adalah 7. Hasil pengukuran waktu komputasi sebagaimana ditunjukkan pada tabel 5.52. Mengenai informasi hasil klaster bisa dilihat pada Tabel 5.18 sampai dengan 5.20, karena hasil klaster HPR-S untuk dataset Soybean Disease sama dengan HPR-V dan HPTR.

Tabel 5.52. Waktu komputasi algortima HPR-S dataset Soybean Disease

Fase	Waktu Komputasi (detik)				
	Minimum	Maksimum	Mean	Total	SD
Red	0.031	0.063	0.04075	0.158	0.009335
Opt	0.031	0.062	0.03905		0.009333
Klast	0.063	0.094	0.07810		0.011031

b) Dataset Congressional Votes

Nilai k reduksi yang dipakai adalah 39. Hasil klaster algoritma HPR-S sama dengan algoritma HPTR sebagaimana ditunjukkan pada Tabel 5.39 sampai dengan Tabel 5.41. Waktu komputasi ditunjukkan pada Tabel 5.53.

Tabel 5.53. Waktu komputasi algortima HPR-S dataset Congressional Votes

Fase	Waktu Komputasi (detik)				
	Minimum	Maksimum	Rerata	Total	Std
Red	3.109	4.140	3.2343	8.9671	0.3109
Opt	2.000	3.125	2.0672		0.2494
Klast	2.781	3.907	3.6656		0.3790

c) Dataset Wisconsin Breast Cancer

Nilai k reduksi yang dipakai adalah 20. Hasil klaster algoritma HPR-S sama dengan algoritma HPR-V dan HPTR sebagaimana ditunjukkan pada Tabel 5.26 sampai dengan Tabel 5.28. Waktu komputasi ditunjukkan pada Tabel 5.54.

a) Dataset Soybean Disease

1. Algoritma K-Modes

Hasil menjalankan algoritma K-Modes pada dataset Soybean Disease ditunjukkan pada Tabel 5.60. Tabel 5.61. menunjukkan rerata tingkat akurasi algoritma K-Modes sebesar 84.68%.

Tabel 5.60. Tingkat akurasi algoritma K-Modes dataset Soybean Disease

Tingkat Akurasi	Misklasifikasi	Jumlah Dataset
100%	0	1
97.87 %	1	1
93.62 %	3	5
85.11 %	7	6
76.60 %	11	4
68.09 %	14	3
Rerata tingkat akurasi hasil klaster = **84.68%**		

Perbandingan mode hasil algoritma K-Modes dengan mode konseptual untuk dataset Soybean Disease ditunjukkan pada tabel 5.61. Mode yang ditemukan dari hasil algoritma K-Modes dengan mode konseptual berbeda 22 dari 140 atribut.

Tabel 5.61. Perbandingan mode algoritma K-Modes
dengan tingkat akurasi 98% dengan mode konseptual dataset Soybean Disease

Kelas		Mode	Beda
D1	Ksp	3,6 0 2 1 0 1 0 1 0,1 2 1 1 0 2 2 0 0 0 1 0 3 1 1 1 0 0 0 0 4 0 0 0 0 0 0	3
	K-Modes	**5** 0 2 1 1 1 **1** 1 1 **1** 1 0 2 2 0 0 0 1 1 3 1 1 1 0 0 0 0 4 0 0 0 0 0 0	
D2	Ksp	5,6 0 0 2 1 1,3 2,3 1 0,1 0 1 1 0 2 2 0 0 0 1 0 0 3 0 0 0 2 1 0 4 0 0 0 0 0 0	3
	K-Modes	5 0 0 **1** 1 1 2 1 1 **1** 1 1 0 2 2 0 0 0 1 **1** 0 3 0 0 0 2 1 0 4 0 0 0 0 0 0	
D3	Ksp	0 1 2 0 0 0,3 1 1,2 0 1,2 1 0 0 2 2 0 0 0 1 0 1 1 0 1 0,1 0 0 3 4 0 0 0 0 0 0	7
	K-Modes	**2** 1 2 0 **1** **1** 1 **1** 1 1 **1** 0 2 2 0 0 0 1 **1** 1 1 0 1 1 0 0 3 4 0 0 0 0 0 **1**	
D4	Ksp	0,1 1 2 1 0 3 1 2 1 0 1 1 0 2 2 0 0 0 1 0 2 2 0 0 0 0 0 3 4 0 0 0 0 0 1	9
	K-Modes	1 1 **1** 1 **1** 1 1 **1** 1 1 0 2 2 0 0 0 1 **1** 1 2 0 **1** 0 0 0 3 4 0 0 0 0 0 **0**	

Untuk mendapatkan waktu komputasi dilakukan uji coba algoritma K-Modes sebanyak 20 kali dengan lingkungan uji coba yang sama dengan lingkungan uji coba algoritma HPR-V, HPTR, dan HPR-S. Dan didapatkan rerata waktu komputasi algoritma K-Modes sebesar 0.0854 detik sebagaimana ditunjukkan tabel 5.62.

Tabel 5.62. Waktu komputasi algortima K-Modes dataset Soybean Disease

Waktu Komputasi (detik)			
Minimum	**Maksimum**	**Mean**	**SD**
0.078	0.094	0.0854	0.00804

2. Algoritma Iterative K-Modes (IKM)

Data hasil klaster algoritma Iterative K-Modes diambil berdasarkan hasil penelitian [Sun02] yang bertujuan untuk mengoptimasi titik pusat awal algoritma K-Modes. Sun menjalankan algoritma Iterative K-Modes pada dataset Soybean Disease sebanyak 20 kali dengan hasil sebagaimana ditunjukkan pada Tabel 5.63. Rerata tingkat akurasi hasil klaster algoritma K-Modes adalah 89.36%.

Tabel 5.63. Tingkat akurasi hasil klaster algoritma Iterative K-Modes dataset Soybean Disease

Tingkat Akurasi	**Jumlah Misklasifikasi**	**Jumlah Data**
97.87 %	1	14
70.21 %	14	5
65.96 %	16	1
Rerata tingkat akurasi hasil klaster = **89.36 %**		

Tabel 5.64 Menunjukkan perbandingan mode algoritma Iterative K-Modes dengan tingkat akurasi 98% yang dihasilkan oleh Sun dengan mode konseptual. Jumlah atribut mode yang dibandingkan hanya 12 buah. Ternyata mode yang dihasilkan algoritma Iterative K-Modes dan mode konseptualnya berbeda 5 dari 48 atribut.

Tabel 5.64. Perbandingan mode algoritma Iterative K-Modes dengan tingkat akurasi 98% dengan mode konseptual dataset Soybean Disease

No atribut		**2**	**3**	**4**	**7**	**21**	**22**	**23**	**24**	**26**	**27**	**28**	**35**	**Beda**
D1	Ksp	0	2	1	0	3	1	1	1	0	0	0	0	0
	HPR-S	0	2	1	0	3	1	1	1	0	0	0	0	
	Iterative	0	2	1	**0,1**	3	1	1	1	0	0	0	0	
D2	Ksp	0	0	2	2,3	0	3	0	0	2	1	0	0	1
	HPR-S	0	0	2	**3**	0	3	0	0	2	1	0	0	
	Iterative	0	0	**1**	**2**	0	3	0	0	2	1	0	0	
D3	Ksp	1	2	0	1	1	1	0	1	0	0	3	0	2
	HPR-S	1	2	0	1	1	1	0	1	0	0	3	0	
	Iterative	1	2	0	1	1	1	0	1	0	0	**2**	**1**	
D4	Ksp	1	2	1	1	2	2	0	0	0	0	3	1	2
	HPR-S	1	2	1	1	2	2	0	0	0	0	3	1	
	Iterative	1	**1**	1	1	**1**	2	0	**0,1**	0	0	3	1	

b) Dataset Congressional Votes

Hasil menjalankan algoritma K-Modes pada dataset Congressional Votes ditunjukkan pada Tabel 5.65. Tabel 5.66 menunjukkan rerata tingkat akurasi algoritma K-Modes sebesar 84.93%.

Tabel 5.65. Hasil algoritma K-Modes dataset Congressional Votes

Masukan	Parameter k-klaster			2	
Hasil	Rerata jumlah misklasifikasi			65.55	
	Rerata tingkat akurasi			84.93%	
	Jumlah iterasi	**Min**	**Mak**	**Rerata**	**SD**
		3	4	3.15	0.366
	Waktu komputasi	1.656	1.844	1.709	0.045
	Beda mode	2	4	3.75	0.639

Tabel 5.66. Tingkat akurasi hasil klaster algoritma K-Modes dataset Congressional Votes

Tingkat Akurasi	**Jumlah Misklasifikasi**	**Jumlah Dataset**
86.21 %	60	1
85.75 %	62	1
85.29 %	64	9
84.83 %	66	2
84.37 %	68	2
84.14 %	69	5
Rerata tingkat akurasi hasil klaster = **84.93%**		

c) Dataset Wisconsin Breast Cancer

Hasil menjalankan algoritma K-Modes pada dataset Wisconsin Breast Cancer ditunjukkan pada Tabel 5.67. Tabel 5.68 menunjukkan rerata tingkat akurasi algoritma K-Modes sebesar 84.93%.

Tabel 5.67 Hasil algoritma K-Modes dataset Wisconsin Breast Cancer

Masukan	k-klaster			2	
Hasil	Rerata jumlah misklasifikasi			97.48	
	Rerata tingkat akurasi			83.56%	
	Jumlah iterasi	**Min**	**Mak**	**Rerata**	**SD**
		3	4	3.1	0.301
	Waktu komputasi (detik)	1.015	2.109	1.462	0.045
	Beda atribut mode	0	10	3.65	4.793

Tabel 5.68 Tingkat akurasi hasil klaster algoritma K-Modes dataset Wisconsin Breast Cancer

Tingkat	Jumlah Misklasifikasi	Jumlah Dataset
95.28 %	33	2
94.56 %	38	2
94.42 %	39	3
92.42 %	53	2
92.28 %	54	1
91.99%	56	1
91.85 %	57	1
91.56 %	59	1
65.10 %	244	7
Rerata tingkat akurasi hasil klaster = **83.56%**		

d) Dataset Hepatitis Domain

Hasil menjalankan algoritma K-Modes pada dataset Hepatitis Domain ditunjukkan pada Tabel 5.69. Tabel 5.70 menunjukkan rerata tingkat akurasi algoritma K-Modes sebesar 70.49%.

Tabel 5.69. Hasil algoritma K-Modes dataset Hepatitis Domain

Masukan	k-klaster			2	
Hasil	Rerata jumlah misklasifikasi			45.6	
	Rerata tingkat akurasi			70.49%	
	Jumlah iterasi	**Min**	**Mak**	**Mean**	**SD**
		3	4	3.55	0.510
	Waktu komputasi (detik)	0.109	0.172	0.1414	0.019
	Beda atribut mode	1	3	1.1	0.447

Tabel 5.70. Tingkat akurasi hasil klaster algoritma K-Modes dataset Hepatitis Domain

Tingkat Akurasi	Jumlah Misklasifikasi	Jumlah Dataset
73.55%	41	1
72.26%	43	1
70.32%	46	18
Rerata tingkat akurasi hasil klaster = 70.49%		

e) Dataset Breast Cancer

Hasil menjalankan algoritma K-Modes pada dataset Breast Cancer ditunjukkan pada Tabel 5.71. Tabel 5.72 menunjukkan rerata tingkat akurasi algoritma K-Modes sebesar 52.55%.

Tabel 5.71. Hasil algoritma K-Modesdataset Breast Cancer

Masukan	Parameter k klaster			2	
Hasil	Rerata jumlah misklasifikasi			135.7	
	Rerata tingkat akurasi klaster			52.55 %	
		Min	**Mak**	**Rerata**	**SD**
	Jumlah iterasi	3	4	3.05	0.224
	Waktu komputasi (detik)	0.187	0.344	0.215	0.035
	Beda atribut mode	6		6.3	0.4702

Tabel 5.72. Tingkat akurasi hasil klaster algoritma K-Modes dataset Breast Cancer

Tingkat Akurasi	**Jumlah Misklasifikasi**	**Kesalahan atribut mode (dari 18 atribut)**	**Jumlah Dataset**
53.85	132	6	3
53.50	133	6	1
53.15	134	6	1
52.80	135	7	6
52.45	136	6	1
52.10	137	6	5
51.05	140	6	3
Rerata tingkat akurasi hasil klaster = **52.55%**			

5.5 Evaluasi

Dari uji coba berbagai macam algoritma klasterisasi terhadap dataset masukan maka bisa diringkas sebagai berikut:

a) Dataset Soybean Disease

Sebagaimana ditunjukkan pada Tabel 5.73 dan Tabel 5.74. Tingkat akurasi hasil klaster antara algoritma HPR-V, HPR-S dan algoritma HPTR adalah sama besar yaitu 100% serta mode yang dihasilkan sama dengan mode konseptual. Tingkat akurasi hasil klaster antara algoritma Iterative K-Modes adalah 89.4% dan algoritma K-Modes adalah 84.68%. Sedang mode yang dihasilkan algoritma Iterative K-Modes dan algoritma K-Modes adalah berbeda dengan mode konseptual.

Tabel 5.73. Perbandingan hasil klaster beberapa algoritma klasterisasi untuk dataset Soybean Disease

Algoritma	Tingkat Akurasi (%)	Beda Atribut Mode (dari 140 atribut)	Waktu Komputasi (detik)
HPR-V	100	0 (0 %)	0.164
HPR-S	100	0 (0 %)	0.158
HPTR	100	0 (0 %)	0.555
IKM	89.36	5 (10.42 %)	-
KM	84.68	22 (15.71%)	0.085

Tabel 5.74. Perbandingan waktu komputasi beberapa algoritma klasterisasi untuk dataset Soybean Disease

HPR-V				
Fase	**Minimum**	**Maksimum**	**Mean**	**SD**
Red	0.031	0.047	0.0368	0.0076
Opt	0.031	0.047	0.0406	0.0077
Klast	0.078	0.094	0.0868	0.0080
Total	**0.140**	**0.188**	**0.1642**	-
HPR-S				
Fase	**Minimum**	**Maksimum**	**Mean**	**SD**
Red	0.031	0.063	0.04075	0.009335
Opt	0.031	0.062	0.03905	0.009333
Klast	0.063	0.094	0.07810	0.011031
Total	**0.125**	**0.219**	**0.1579**	-

Tabel 5.74. Perbandingan waktu komputasi beberapa algoritma klasterisasi untuk dataset Soybean Disease (lanjutan)

HPTR				
Fase	**Minimum**	**Maksimum**	**Mean**	**SD**
Opt	0.468	0.485	0.4711	0.005889
Klast	0.063	0.094	0.0836	0.010425
Total	**0.531**	**0.579**	**0.5547**	-
K-Modes				
Minimum		**Maksimum**	**Mean**	**SD**
0.078		0.094	**0.08535**	0.00804

b) Dataset Congressional Votes

Ringkasan uji coba berbagai macam algoritma klasterisasi terhadap dataset Congressional Votes ditunjukkan pada Tabel 5.75 dan Tabel 5.76.

Tabel 5.75. Perbandingan hasil klaster beberapa algoritma klasterisasi untuk dataset Congressional Votes

Algoritma	**Tingkat Akurasi (%)**	**Beda Atribut Mode (dari 32 atribut)**	**Waktu Komputasi (detik)**
HPR-V	86.23	3.9 (12.19 %)	9.439
HPR-S	86.21	4 (12.50 %)	8.967
HPTR	86.21	4 (12.50 %)	129.770
K-Modes	84.93	3.75 (11.72%)	1.709

Tabel 5.76. Perbandingan waktu komputasi (detik) beberapa algoritma klasterisasi untuk dataset Congressional Votes

HPR-V				
Fase	**Minimum**	**Maksimum**	**Mean**	**SD**
Red	2.687	4.594	3.6837	0.5181
Opt	1.266	2.984	2.1071	0.4389
Klast	2.860	3.985	3.6487	0.4232
Total	**6.813**	**11.563**	**9.4395**	-

Tabel 5.76. Perbandingan waktu komputasi (detik) beberapa algoritma klasterisasi untuk dataset Congressional Votes (lanjutan)

HPR-S				
Fase	**Minimum**	**Maksimum**	**Mean**	**SD**
Red	3.109	4.140	3.2343	0.3109
Opt	2.000	3.125	2.0672	0.2494
Klast	2.781	3.907	3.6656	0.3790
Total	**7.890**	**11.172-**	**8.9671**	-
HPTR				
Fase	**Minimum**	**Maksimum**	**Mean**	**SD**
Opt	122.531	128.563	126.318	1.6364
Klast	3.000	3.985	3.452	0.4247
Total	**125.531**	**132.548**	**129.770**	-
K-Modes				
Minimum		**Maksimum**	**Mean**	**SD**
1.656		1.844	**1.709**	0.045

c) Dataset Wisconsin Breast Cancer

Ringkasan uji coba berbagai macam algoritma klasterisasi terhadap dataset Congressional Votes ditunjukkan pada Tabel 5.77 dan Tabel 5.78.

Tabel 5.77. Perbandingan hasil klaster beberapa algoritma klasterisasi untuk dataset Wisconsin Breast Cancer

Algoritma	Tingkat Akurasi (%)	Beda Atribut Mode (dari 18 atribut)	Waktu Komputasi (detik)
HPR-V	86.41	1 (0.06%)	11.2731
HPR-S	86.41	1 (0.06%)	16.8362
HPTR	86.41	1 (0.06%)	253.530
K-Modes	83.56	3.65 (0.20%)	1.462

Tabel 5.78. Perbandingan waktu komputasi (detik) beberapa algoritma klasterisasi untuk dataset Wisconsin Breast Cancer

HPR-V				
Fase	**Minimum**	**Maksimum**	**Mean**	**SD**
Red	9.766	11.641	7.3685	0.5853
Opt	1.515	1.844	1.1917	0.0928
Klast	3.390	4.766	2.7145	0.5351
Total	**14.671**	**18.251**	**11.2747**	-
HPR-S				
Fase	**Minimum**	**Maksimum**	**Mean**	**SD**
Red	10.125	11.594	11.1221	0.5274
Opt	1.796	1.844	1.8015	0.0115
Klast	3.297	4.469	3.9126	0.5153
Total	**15.218**	**17.907**	**16.8362**	-
HPTR				
Fase	**Minimum**	**Maksimum**	**Mean**	**SD**
Opt	246.172	256.484	249.2711	2.1228
Klast	3.406	4.844	4.2589	0.4388
Total	**249.578**	**261.328**	**253.530**	-
K-Modes				
Minimum		**Maksimum**	**Mean**	**SD**
1.015		2.109	1.462	0.045

d) Dataset Hepatitis Domain

Dari uji coba berbagai macam algoritma klasterisasi terhadap dataset Hepatitis Domain maka bisa diringkas sebagaimana ditunjukkan pada tabel 5.79 dan tabel 5.80.

Tabel 5.79. Perbandingan hasil klaster beberapa algoritma klasterisasi untuk dataset Hepatitis Domain

Algoritma	Tingkat Akurasi (%)	Beda Atribut Mode (dari 38 atribut)	Waktu Komputasi (detik)
HPR-V	76.74	1.1 (2.89%)	0.812
HPR-S	76.77	2 (5.26%)	0.805
HPTR	76.77	2 (5.26%)	6.660
KM	70.49	1.1 (2.89%)	0.141

Tabel 5.80. Perbandingan waktu komputasi (detik) beberapa algoritma klasterisasi untuk dataset Hepatitis Domain

HPR-V				
Fase	**Minimum**	**Maksimum**	**Mean**	**SD**
Red	0.172	0.219	0.193	0.0116
Opt	0.125	0.203	0.201	0.0864
Klast	0.328	0.531	0.418	0.0344
Total	**0.625**	**0.953**	**0.812**	-
HPR-S				
Fase	**Minimum**	**Maksimum**	**Mean**	**SD**
Red	0.187	0.235	0.2039	0.0096
Opt	0.172	0.203	0.1866	0.0061
Klast	0.406	0.422	0.4143	0.0079
Total	**0.765**	**0.860**	**0.8048**	-
HPTR				
Fase	**Minimum**	**Maksimum**	**Mean**	**SD**
Opt	5.312	6.579	6.179	0.4888
Klast	0.406	0.453	0.423	0.0139
Total	0.718	7.032	6.602	-
K-Modes				
Minimum		**Maksimum**	**Mean**	**SD**
0.109		0.172	0.1414	0.019

e) Dataset Breast Cancer

Dari uji coba berbagai macam algoritma klasterisasi terhadap dataset Breast Cancer maka bisa diringkas sebagaimana ditunjukkan pada tabel 5.81 dan tabel 5.82.

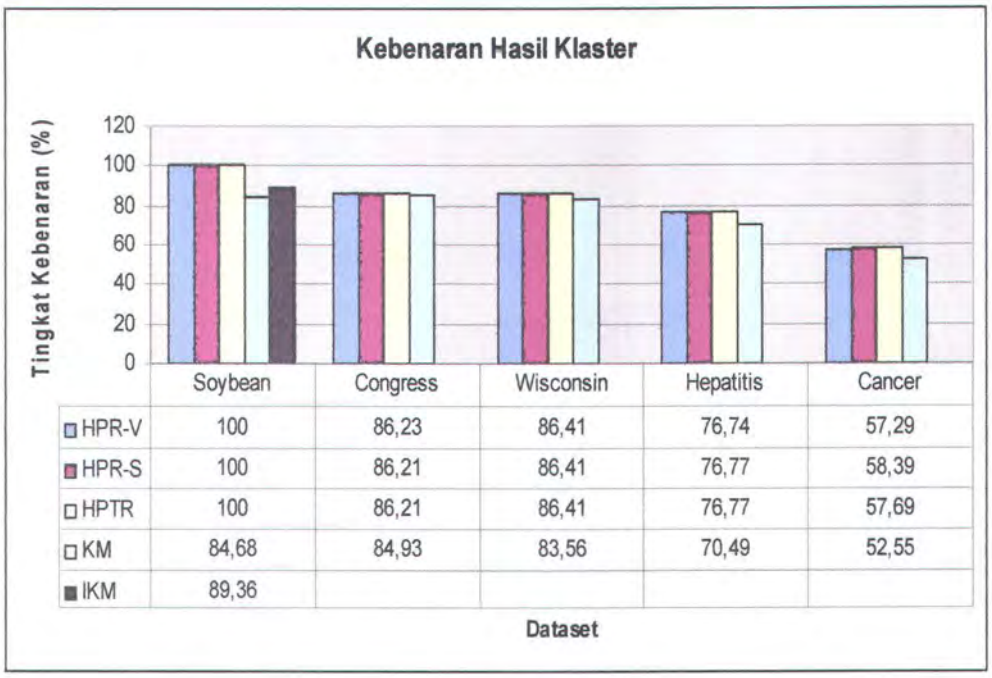
Tabel 5.81 Perbandingan hasil klaster beberapa algoritma klasterisasi untuk dataset Breast Cancer

Algoritma	Tingkat Akurasi (%)	Beda Atribut Mode (dari 18 atribut)	Waktu Komputasi (detik)
HPR-V	57.29	5.7 (0.32%)	3.4921
HPR-S	58.39	6 (0.33%)	3.679
HPTR	57.69	5 (0.28%)	23.128
K-Modes	52.55	6.3 (0.35%)	0.215

Tabel 5.82 Perbandingan waktu komputasi (detik) beberapa algoritma klasterisasi untuk dataset Breast Cancer

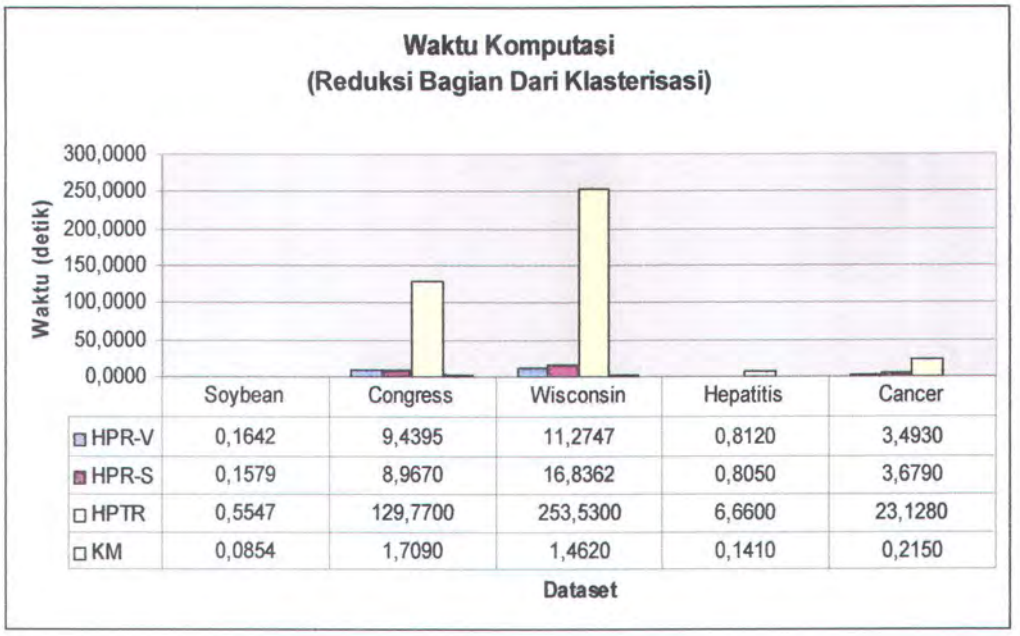
HPR-V				
Fase	**Minimum**	**Maksimum**	**Mean**	**SD**
Red	1.781	1.938	1.851	0.0421
Opt	0.218	0.250	0.241	0.0107
Klast	1.093	2.125	1.401	0.4756
Total	**3.092**	**4.313**	**3.493**	-
HPR-S				
Fase	**Minimum**	**Maksimum**	**Mean**	**SD**
Red	1.797	1.828	1.8071	0.0091
Opt	0.234	0.266	0.2447	0.0091
Klast	1.609	1.703	1.6273	0.0232
Total	**3.640**	**3.797**	**3.679**	-
HPTR				
Fase	**Minimum**	**Maksimum**	**Mean**	**SD**
Opt	20.562	22.610	21.504	0.5187
Klast	1.000	1.985	1.624	0.4634
Total	**21.562**	**24.595**	**23.128**	-
K-Modes				
Minimum		**Maksimum**	**Mean**	**SD**
0.187		0.344	**0.215**	0.035

Bila semua hasil uji coba disajikan dalam bentuk grafik, maka gambar 5.6 menunjukkan perbandingan tingkat akurasi semua dataset dengan beberapa algoritma klasterisasi. Berdasarkan gambar tersebut, algoritma HPR-V, HPR-S dan HPTR secara umum menghasilkan klaster yang lebih akurat daripada algoritma KM dan IKM.

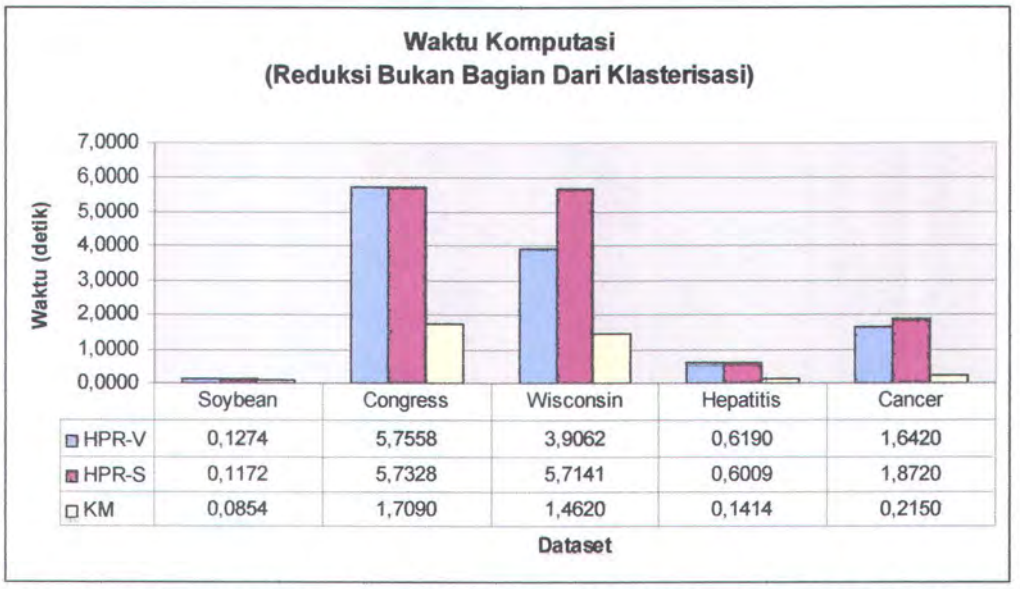


Gambar 5.6 Perbandingan tingkat akurasi

Gambar 5.7 dan gambar 5.8 menunjukkan perbandingan waktu komputasi. Pada gambar 5.7 proses reduksi dianggap termasuk sebagai bagian dari proses klasterisasi sehingga waktu yang diperlukan untuk proses reduksi data dihitung sebagai bagian dari proses klaterisasi. Tampak bahwa algoritma HPTR membutuhkan waktu paling lama karena harus mengoptimasi semua data. Sedang gambar 5.8 proses reduksi dianggap tidak termasuk sebagai bagian dari proses klasterisasi. Dari hasil ini algoritma HPR-V dan HPR-S secara umum membutuhkan waktu komputasi yang lebih lama dibandingkan dengan algoritma KM.



Gambar 5.7. Waktu komputasi bila reduksi data termasuk proses klaster



Gambar 5.8. Waktu komputasi bila reduksi data tidak termasuk proses klaster

Algoritma Hirarki Partisi Reduksi membutuhkan waktu komputasi yang relatif lebih lama dibandingkan dengan algoritma K-Modes (tanpa penentuan titik

pusat awal), hal ini dikarenakan algoritma Hirarki Partisi Reduksi membutuhkan waktu untuk menentukan titik pusat awal yang terdiri dari mereduksi data dan mengoptimasi hasil reduksi data menjadi sejumlah titik pusat awal. Apabila proses reduksi dimasukkan sebagai bagian dari proses penentuan titik pusat awal maka waktu penentuan titik pusat awal sebesar 58.14% dari waktu untuk keseluruhan proses. Namun apabila proses reduksi tidak dimasukkan sebagai bagian dari proses penentuan titik pusat awal, maka waktu penentuan titik pusat awal yang diperlukan menjadi sebesar 29% dari waktu untuk keseluruhan proses.

Dari data hasil uji coba dengan menggunakan 5 dataset maka rata-rata akurasi hasil klaster dengan algoritma HPR-S adalah 81,56% sedang algoritma KM adalah 75,24%. Dengan demikian, algoritma HPR-S menghasilkan peningkatan akurasi sebesar 8,4%. Tetapi, waktu komputasi algoritma KM lebih cepat yaitu sebesar 0.723 detik sedang HPR-S (reduksi bukan bagian dari klasterisasi) sebesar 2.807 detik sehingga kenaikan waktu komputasi menjadi 288.54%. Bila waktu komputasi algoritma HPR-S (reduksi bagian dari klasterisasi) sebesar 6.089 detik dibandingkan dengan algoritma KM maka kenaikan waktu komputasi menjadi sebesar 742.79%.

BAB 6

PENUTUP

BAB 6

PENUTUP

Bab ini berisi simpulan yang dapat diambil dari hasil penelitian dan uji coba sub-bab, serta berisi saran yang bermanfaat bagi pengembangan lebih lanjut dari penelitian ini.

6.1. Simpulan

Dari hasil uji akurasi hasil klaster, uji kinerja, uji parameter dan uji perbandingan algoritma yang dilakukan pada beberapa dataset dapat diambil beberapa kesimpulan sebagai berikut :

a. Algoritma Hirarki Partisi Reduksi dapat memberikan hasil klaster yang baik dengan tingkat akurasi yang dapat mencapai 100%. Jika dibandingan dengan algoritma K-Modes [Huang97a], kenaikan tingkat akurasi algoritma Hirarki Partisi Reduksi adalah sebesar 8.4%.
b. Algoritma Hirarki Partisi Reduksi dapat memberikan hasil klaster yang lebih baik bila dibandingkan dengan algoritma Iterative K-Modes [Sun02] untuk dataset Soybean Disease. Kenaikan tingkat akurasi algoritma Hirarki Partisi Reduksi adalah sebesar 11.9%.
c. Parameter reduksi k dalam proses reduksi data dapat diset dengan nilai sembarang untuk mendapatkan tingkat akurasi yang baik. Hal ini didukung

oleh hasil uji coba yang menunjukkan bahwa hasil klaster untuk berbagai nilai reduksi k memberikan tingkat akurasi yang relatif sama.

d. Kenaikan waktu komputasi Algoritma Hirarki Partisi Reduksi adalah 288,53% (bila proses reduksi data dianggap bukan bagian dari klasterisasi) dari algoritma K-Modes. Sedang apabila proses reduksi data dianggap sebagai bagian dari klasterisasi, maka waktu komputasi meningkat menjadi 742.79% dari algoritma K-Modes.

6.2. Saran

Hasil uji coba menunjukkan bahwa waktu yang dibutuhkan untuk memperbaiki penentuan titik pusat awal masih sangat besar jika proses reduksi data dijadikan sebagai bagian dari proses klasterisasi. Oleh karena itu diperlukan penelitian lanjutan untuk mengatasi permasalahan ini, yaitu dengan mencari alternatif metode reduksi data sehingga waktu yang dibutuhkan untuk memperbaiki penentukan titik pusat awal akan menjadi lebih kecil.

DAFTAR PUSTAKA

DAFTAR PUSTAKA

[And04] P. Andritsos, "Scalable Clustering of Categorical Data And Applications", *Phd Thesis University of Toronto, Departement of Computer Science*, September, 2004.

[Barb02] D. Barbara, Y. Li, J. Couto, "COOLCAT: an entropy-based algorithm for categorical clustering," *in: Proc. of CIKM'02*, 2002. pp. 582–589.

[Brad98] P. Bradley, U. Fayyad, C. Reina, "Refining initial points for k-means clustering." *In: Proc. 15th Internat. Conf. on Machine Learning. Morgan Kaufmann,* Los Altos, CA,1998.

[Fun04] R.E. Funderlic, M.T. Chu, N. Orlowski, D. Schlor, J. Blevins, D. Canas, D, "Convergence and Other Aspects of the k-modes Algorithm for Clustering Categorical data," *http://www4.ncsu.edu/~mtchu/Research/papers/k-modes.pdf*, N.C. State University Department of Computer Science March 25, 2004. Didownload pada bulan April 2006.

[Gan99] V. Ganti, J. Gehrke, R. Ramakrishnan, "CACTUS-clustering categorical data using summaries," *in: Proc. of KDD'99*, 1999, pp.73–83.

[Gib98] D. Gibson, J. Kleiberg, P. Raghavan, "Clustering categorical data: an approach based on dynamic systems," *in: Proc. of VLDB'98,* 1998 pp. 311–323.

[Gu99] S. Guha, R. Rastogi, K. Shim, "ROCK: a robust clustering algorithm for categorical attributes," *in: Proc. of ICDE'99,* 1999, p. 512–521.

[He02] Z. He, X. Xu, S. Deng, "Squeezer: an efficient algorithm for clustering categorical data," *Journal of Computer Science and Technology 17 (5),* 2002, pp 611–624.

[He05] Z. He, S. Deng, X. Xu, "Improving k-modes algorithm considering frequencies of attribute values in mode," *International Conference on Intelligent Computing*, China, 2005.

[Huang97a] Z. Huang, "A fast clustering algorithm to cluster very large categorical data sets in data mining," *In: SIGMOD Workshop on*

Research Issues on Data Mining and Knowledge Discovery, Tucson, AZ. 1997 pp. 8.

[Huang97b] Z. Huang, "Clustering *large data sets with mixed numeric and categorical values*," *In: Proc. First Pacific Asia Knowledge Discovery and Data Mining Conf. World Scientific*, Singapore, 1997, pp. 21-34.

[Huang98] Z. Huang, "Extensions to the k-means algorithm for clustering large data sets with categorical values," *Data Mining and Knowledge Discovery*, 2, 1998, pp 283–304.

[Iman04] B.N. Iman, "Penerapan Metode Pengklasteran Hibrida Berbasis K-Mean dan Algoritma Genetika", *Thesis di Program Magister Jurusan Teknik Informatika, ITS*, Surabaya, 2004.

[Merz96] C. J. Merz, P. Merphy, "UCI Repository of Machine Learning Databases", *(http://www.ics.uci.edu/~mlearn/MLRRepository.html)* 1996.

[Mit02] P. Mitra, C. A. Murthy, S.K. Par. "Density-Based Multiscale Data Condensation," *IEEE Transactions On Pattern Analysis And Machine Intelligence, Vol. 24, No.6,* June, 2002.

[Moo01] A. Moore, "K-means and Hierarchical Clustering", *http://www-2.cs.cmu.edu/~awm/tutorials/ kmeans.html*, November, 2001. Didownload pada bulan April 2006.

[San04] O. San, V. Huynh, Y. Nakamori, "An alternative extension of the k-means algorithm for clustering categorical data," *Int. J. Appl. Math. Comput. Sci.*, Vol. 14, No. 2, 2004, pp 241-247.

[Sun02] Y. Sun, Q. Zhu, Z. Chen, "An iterative initial-points refinement algorithm for categorical data clustering," *Pattern Recognition Letters*, 23 July. 2002, 875-884.

[Sya03] I. Syarif, "Aplikasi Data Mining untuk Pendeteksian Intrusi pada Sistem Jaringan dengan Metode Klasifikasi dan Clustering", *Thesis di Program Magister Jurusan Teknik Informatika, ITS*, Surabaya, 2003.

LAMPIRAN A

INFORMASI ATRIBUT DATASET

1. Soybean Disease

No	Atribut	Nilai
1	date	april,may,june,july,august,september,october,?.
2	plant-stand	normal,lt-normal,?.
3	precip	lt-norm,norm,gt-norm,?.
4	temp	lt-norm,norm,gt-norm,?.
5	hail	yes,no,?.
6	crop-hist	diff-lst-year,same-lst-yr,same-lst-two-yrs, same-lst-sev-yrs,?.
7	area-damaged	scattered,low-areas,upper-areas,whole-field,?.
8	severity	minor,pot-severe,severe,?.
9	seed-tmt	none,fungicide,other,?.
10	germination	90-100%,80-89%,lt-80%,?.
11	plant-growth	norm,abnorm,?.
12	leaves	norm,abnorm.
13	leafspots-halo	absent,yellow-halos,no-yellow-halos,?.
14	leafspots-marg	w-s-marg,no-w-s-marg,dna,?.
15	leafspot-size	lt-1/8,gt-1/8,dna,?.
16	leaf-shread	absent,present,?.
17	leaf-malf	absent,present,?.
18	leaf-mild	absent,upper-surf,lower-surf,?.
19	stem	norm,abnorm,?.
20	lodging	yes,no,?.
21	stem-cankers	absent,below-soil,above-soil,above-sec-nde,?.
22	canker-lesion	dna,brown,dk-brown-blk,tan,?.
23	fruiting-bodies	absent,present,?.
24	external decay	absent,firm-and-dry,watery,?.
25	mycelium	absent,present,?.
26	int-discolor	none,brown,black,?.
27	sclerotia	absent,present,?.
28	fruit-pods	norm,diseased,few-present,dna,?.
29	fruit spots	absent,colored,brown-w/blk-specks,distort,dna,?.
30	seed	norm,abnorm,?.
31	mold-growth	absent,present,?.
32	seed-discolor	absent,present,?.
33	seed-size	norm,lt-norm,?.
34	shriveling	absent,present,?.
35	roots	norm,rotted,galls-cysts,?.

2. Congressional Votes

No	Atribut	Nilai
1	Class Name	2(democrat, republican)
2	handicapped-infants	2 (y,n)
3	water-project-cost-sharing	2 (y,n)
4	adoption-of-the-budget-resolution	2 (y,n)
5	physician-fee-freeze	2 (y,n)
6	el-salvador-aid	2 (y,n)
7	religious-groups-in-schools	2 (y,n)
8	anti-satellite-test-ban	2 (y,n)
9	aid-to-nicaraguan-contras	2 (y,n)
10	mx-missile	2 (y,n)
11	immigration	2 (y,n)
12	synfuels-corporation-cutback	2 (y,n)
13	education-spending	2 (y,n)
14	superfund-right-to-sue	2 (y,n)
15	crime	2 (y,n)
16	duty-free-exports	2 (y,n)
17	export-administration-act-south-africa	2 (y,n)

3. Wisconsin Breast Cancer

No	Atribut	Nilai
1	Clump Thickness	10-Jan
2	Uniformity of Cell Size	10-Jan
3	Uniformity of Cell Shape	10-Jan
4	Marginal Adhesion	10-Jan
5	Single Epithelial Cell Size	10-Jan
6	Bare Nuclei	10-Jan
7	Bland Chromatin	10-Jan
8	Normal Nucleoli	10-Jan
9	Mitoses	10-Jan
10	Class	2 for benign, 4 for malignant

4. Hepatitis Domain

No	Atribut	Nilai
1	Class	DIE, LIVE
2	SEX	male, female
3	STEROID	no, yes
4	ANTIVIRALS	no, yes
5	FATIGUE	no, yes
6	MALAISE	no, yes
7	ANOREXIA	no, yes
8	LIVER BIG	no, yes
9	LIVER FIRM	no, yes
10	SPLEEN PALPABLE	no, yes
11	SPIDERS	no, yes
12	ASCITES	no, yes
13	VARICES	no, yes
14	HISTOLOGY	no, yes

5. Breast Cancer

No	Atribut	Nilai
1	Class	no-recurrence-events, recurrence-events
2	age	10-19, 20-29, 30-39, 40-49, 50-59, 60-69, 70-79, 80-89, 90-99.
3	menopause	lt40, ge40, premeno.
4	tumor-size	0-4, 5-9, 10-14, 15-19, 20-24, 25-29, 30-34, 35-39, 40-44, 45-49, 50-54, 55-59.
5	inv-nodes	0-2, 3-5, 6-8, 9-11, 12-14, 15-17, 18-20, 21-23, 24-26, 27-29, 30-32, 33-35, 36-39.
6	node-caps	yes, no.
7	deg-malig	1, 2, 3.
8	breast	left, right.
9	breast-quad	left-up, left-low, right-up, right-low, central.
10	irradiat	yes, no.