BAB 3
ANALISIS DAN PERANCANGAN SISTEM

Bab ini menjelaskan mengenai analisa dan perancangan sistem yang dilakukan oleh tim pengembang FIKUI Mining. Kegiatan analisis meliputi analisis permasalahan, kebutuhan dan hambatan. Sedangkan kegiatan perancangan meliputi perancangan *Sequence Diagram,* dan *Class Diagram*. Analisis dan perancangan sistem ini dilakukan setelah tim pengembang selesai mempelajari kode maupun sistem yang sudah ada seperti WEKA dan juga mempelajari laporan dari tim pengembang sebelumnya.

3.1 Analisis Kebutuhan

Proses analisis kebutuhan dilakukan oleh tim pengembang berdasarkan hasil pengamatan dan percobaan tim pengembang terhadap sistem yang menjadi acuan yaitu WEKA. Selain itu, tim pengembang juga mengadakan wawancara kepada kepada pembimbing Proyek Mahasiswa dan orang-orang yang pernah mengembangkan algoritma-algoritma pada penelitian *data mining algorithms collection*, untuk mendapatkan informasi-informasi tambahan yang dapat digunakan sebagai modal dalam pengembangan sistem.

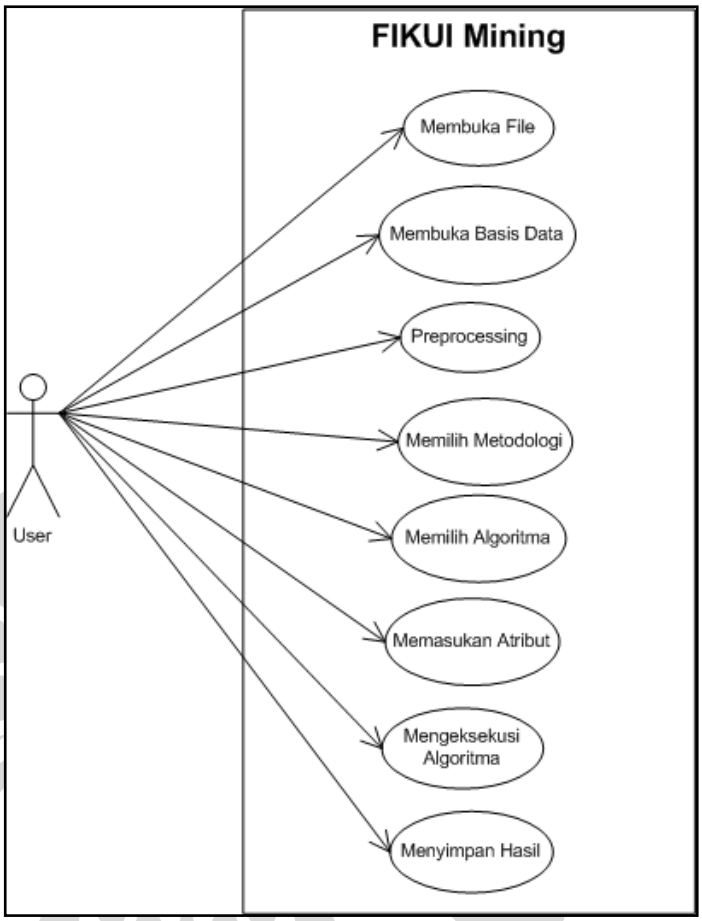
Berdasarkan hasil analisis yang dilakukan, kebutuhan sistem secara garis besar terbagi menjadi dua, yaitu kebutuhan fungsional dan kebutuhan nonfungsional.

3.1.1 Kebutuhan Fungsional

Untuk menjelaskan kebutuhan fungsional, tim pengembang menggunakan *use-case diagram*. Untuk menggambarkan *use-case diagram*, tim pengembang menggunakan aplikasi Microsoft Office Visio 2007 yang menyediakan notasi-notasi standar pembuatan *use-case*.

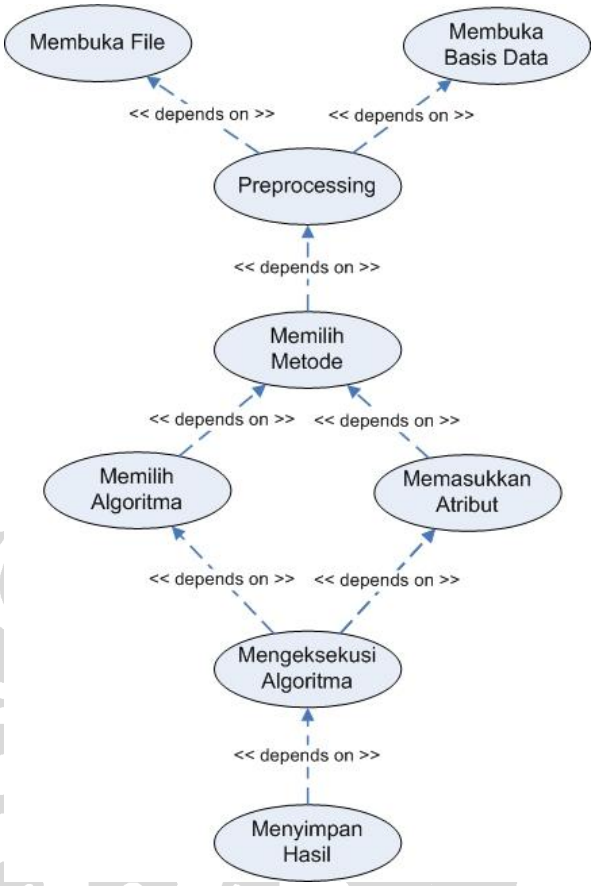
Kebutuhan fungsional dari FIKUI Mining dapat digambarkan oleh *use-case diagram* pada Gambar 3.1.

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008



Gambar 3. 1: ***Use Case Diagram***

Gambar 3.2 adalah *use-case dependency diagram* yang menunjukkan ketergantungan yang terjadi antar *use-case.*



Gambar 3.2: *Use Case Dependency Diagram*

Actor yang berperan dalam sistem ini hanya ada satu yaitu pengguna dari FIKUI Mining.

Berikut ini adalah rincian *use-case* dari FIKUI Mining:

1. **Membuka *File***

   Pada *use-case* ini *actor* dapat membuka *file* dari *local hardisk* komputer yang menjalankan FIKUI Mining. *Use-case* ini menyediakan input data untuk diproses pada FIKUI Mining. Jika komputer yang menjalankan FIKUI Mining terhubung dengan jaringan lokal atau yang sering disebut dengan LAN (*Local Area Network*), maka *actor* juga dapat membuka *file* dari komputer lain yang terhubung dengan komputer yang menjalankan FIKUI Mining.

2. **Membuka Basis Data**

   Pada *use-case* ini *actor* dapat membuka basis data untuk mengambil data-data yang ada pada basis data yang dipilih dan

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

memprosesnya pada FIKUI Mining. *Use-case* ini berfungsi untuk menyediakan input data bagi FIKUI Mining, sama seperti *use-case* membuka *file*.

3. ***Preprocessing***

   Pada *use-case* ini *actor* dapat melakukan pemrosesan awal pada input data yang dimasukan ke dalam FIKUI Mining. *Use-case* ini diperlukan untuk menyamakan format dari input data yang akan dimasukan ke dalam FIKUI Mining untuk diproses dan di-*mining* informasi yang ada di dalamnya. *Use-case* ini tidak dikerjakan pada proyek ini karena akan dikembangkan lebih lanjut untuk kebutuhan riset.

4. **Memilih Metodologi**

   Pada *use-case* ini *actor* dapat memilih metodologi untuk mengolah input data yang ada. Terdapat 3 metodologi yang berbeda yang dapat dipilih oleh *actor* untuk mengolah data yang dimasukan ke dalam FIKUI Mining.

5. **Memilih Algoritma**

   Pada *use-case* ini *actor* dapat memilih algoritma yang diinginkan untuk pengolahan input data yang dimasukan. Pemilihan algoritma ini dilakukan setelah *actor* memilih metodologi terlebih dahulu. Dari satu metodologi yang dipilih *actor*, terdapat beberapa algoritma yang dapat digunakan.

6. **Memasukkan Atribut**

   Pada *use-case* ini *actor* dapat memasukan atribut-atribut yang dapat dimiliki oleh suatu algoritma. Atribut-atribut yang dimasukkan oleh *actor* akan sangat mempengaruhi baik hasil maupun unjuk kerja dari sebuah algoritma dalam pemrosesan data yang dimasukan sebagai input. Untuk dapat memasukan atribut, seorang *actor* sebelumnya harus sudah menentukan metodologi dan juga memilih algoritma. Kemudian, barulah atribut dapat dimasukan sesuai metodologi dan algoritma yang dipilih.

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

7. Mengeksekusi Algoritma

Pada *use-case* ini *actor* dapat mengeksekusi algoritma untuk memproses input data yang sudah dimasukan ke dalam FIKUI Mining. Untuk dapat mengeksekusi sebuah algoritma, sebelumnya *actor* sudah harus membuka *file*, memilih metodologi, memilih algoritma dan memasukan atribut yang dibutuhkan untuk menjalankan proses.

8. Menyimpan Hasil

Pada *use-case* ini *actor* dapat menyimpan hasil pemrosesan yang telah dilakukan pada suatu input data tertentu yang telah dimasukan ke FIKUI Mining. Hasil penyimpanan dapat disimpan ke dalam *local harddisk* dari komputer dimana FIKUI Mining dijalankan. Jika komputer tempat menjalankan FIKUI Mining terhubung dengan komputer lainnya melalui sebuah LAN, hasil pemrosesan juga dapat disimpan ke komputer lain yang terhubung tersebut. Hasil pemrosesan akan disimpan ke dalam sebuah *file* yang nama maupun ekstensinya dapat dimasukan oleh *actor*. Untuk dapat menyimpan hasil pemrosesan, sebelumnya *actor* sudah harus mengeksekusi algoritma untuk memproses data yang dimasukan ke FIKUI Mining.

3.2.2 Kebutuhan Fungsional

Selain kebutuhan fungsional, FIKUI Mining juga mempunyai kebutuhan nonfungsional, yang merupakan kebutuhan tambahan di luar fungsi sistem. Kebutuhan nonfungsional tersebut adalah sebagai berikut.

1. Antarmuka sistem yang bersifat *user friendly*, yaitu nyaman dan tidak membuat *user* menjadi bingung. Oleh karena itu, dalam perancangan antarmuka perlu memperhatikan prinsip konsistensi dari segi penggunaan jenis dan ukuran huruf, serta posisi tombol yang tidak berubah-ubah.
2. Sistem diharapkan terintegrasi dengan situs Fasilkom UI sehingga memungkinkan semua orang untuk mencoba FIKUI Mining dari tempat yang berbeda.

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

3. FIKUI Mining diharapkan dapat menangani data dengan ukuran besar dengan cepat dan mengeluarkan hasil yang akurat. FIKUI Mining diharapkan juga dapat menggunakan *resource* dengan efektif dan efisien sehingga FIKUI Mining dapat dioperasikan pada komputer dengan spesifikasi tidak terlalu tinggi.

3.3 Analisis Algoritma

Tim pengembang sebelumnya; yaitu Anthony, Fahrian, dan Rani; sudah mengimplementasikan beberapa algoritma dari 3 metode yang berbeda. Pada subbab ini akan dianalisis secara singkat algoritma-algoritma yang ada tersebut. Agar lebih mudah untuk dianalisis, pembahasan akan dilakukan per metode yang ada yaitu *Association*, *Clustering* dan *Classification*. Untuk perbaikan maupun peningkatan algoritma akan dibahas lebih jauh pada bab 4 yaitu Implementasi Sistem.

3.3.1 *Association*

Cara kerja dari metode ini adalah mencari pola-pola yang sering muncul dalam data. Pengetahuan yang didapat adalah *rule* yang menunjukkan hubungan suatu kelompok *item* terhadap kelompok *item* yang lain. Algoritma yang dikembangkan dalam metode ini adalah *Apriori*, *FP-Growth*, dan *CT-Pro*.

a. *Apriori*

Algoritma ini dikembangkan oleh Christian Borgelt. Ide dasar dari algoritma ini adalah memprediksi pola kemunculan *item* yang muncul dalam data. Algoritma ini mendapatkan sejumlah *frequent itemset* dari data transaksi dengan beberapa iterasi. Dalam suatu iterasi, dicari suatu kelompok *frequent itemset* tertentu. Iterasi ke-*i* berarti mendapatkan semua *frequent i-itemset* (suatu *itemset* yang jumlah *item* anggotanya sejumlah *i*). Langkah umum tiap iterasi adalah menghasilkan *candidate-candidate* yang ada yaitu mencari *itemset-itemset* dengan jumlah anggota tertentu yang mungkin dari seluruh

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

keseluruhan data, kemudian melakukan perhitungan nilai *support* tiap *candidate* dan memilih *candidate* mana saja yang dianggap sebagai *frequent itemset*.

Berikut ini adalah tahapan dalam algoritma *Apriori* [HAM06]:

Keterangan:

C_i adalah kumpulan *candidate* untuk iterasi ke-*i*.

L_i adalah kumpulan *frequent itemset* untuk iterasi ke-*i*.

Untuk Iterasi pertama:

1. Hasilkan semua *candidate* untuk *1-itemset* di C_1.
2. Simpan semua *frequent itemset* di L_1.

Untuk Iterasi ke k:

1. Hasilkan *candidate* untuk dimasukkan ke C_k dari *frequent itemsets* dalam L_{k-1}
   a. Mengambil 2 anggota L_{k-1} yaitu *p* dan *q*.
   b. Jika $p.item_1 = q.item_1, \ldots p.item_{k-2} = q.item_{k-2}, p.item_{k-1} < q.item_{k-1}$, maka masukkan *candidate* yang baru adalah $p.item_1, p.item_2, \ldots p.item_{k-2}, p.item_{k-1}, q.item_{k-1}$.
   c. Hilangkan *candidate-candidate* dalam C_k dimana (*k-1*)-*subset*-nya ada yang tidak *frequent*.
2. Baca transaksi dalam data untuk menghitung nilai *support* masing-masing *candidate*.
3. Simpan *frequent itemset* dalam L_k.

Berikut ini adalah *pseudocode* dari *Apriori*.

```
Ck   : Candidate itemset of size k
Lk   : Frequent itemset of size k
L1   = {frequent items};
for (k = 1; Lk !=∅; k++) do begin
    Ck+1 = candidates generated from Lk;
    for each transaction t in data do
        increment the count of all candidates in Ck+1 that are contained in t
        Lk+1  = candidates in Ck+1 with minimum_support
    end
```

Penjelasan singkat *pseudocode*:

Nilai *support* harus ditentukan terlebih dahulu, kemudian suatu *itemset* menjadi *frequent itemset* jika nilai kemunculannya melebihi

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

nilai *support*. Nilai *support* yang digunakan merupakan hasil jumlah minimal kemunculan dibagi dengan jumlah transaksi. Dengan demikian input untuk nilai *support* sudah berupa pecahan. Untuk menghitung apakah suatu *itemset* adalah *frequent itemset* atau tidak, hasil jumlah kemunculan *itemset* dibagi dengan jumlah transaksi dibandingkan dengan nilai *support* tersebut.

Untuk iterasi pertama, setiap jenis *item* yang ditemukan dalam data dijadikan *candidate* untuk *frequent 1-itemsets*. Adapun untuk menghasilkan *candidate* atau *candidate generation* iterasi berikutnya didapatkan dari *frequent itemset-frequent itemset* yang ditemukan di iterasi sebelumnya. Jika tidak ada lagi kombinasi *candidate* yang dapat dibuat dari *frequent itemset*, maka proses iterasi ini berhenti.

b. *FP-Growth*

Algoritma *Frequent Pattern Growth* atau *FP-Growth* dikembangkan oleh Jiawei Han dan rekan-rekannya. Ide dasar dari algoritma ini adalah bagaimana melakukan pencarian *frequent itemset* tanpa menghasilkan *candidates.* Pada algoritma *Apriori*, terdapat proses yang kompleks dengan melakukan pembacaan data berulang-ulang terutama pada bagian *candidate generation* yang bertujuan untuk mendapatkan calon *frequent itemset*. Algoritma ini menggunakan struktur data FP-*tree* sebagai representasi data. Keuntungan dari algoritma ini adalah kita tidak perlu melakukan *candidate generation* dan pembacaan data tidak perlu dilakukan berulang-ulang [HAN00].

Berikut adalah *pseudocode FP-Growth* untuk mendapatkan *Frequent Itemset* [ELM00]:

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

```
Procedure FP-Growth (tree, alpha):
Begin
If tree contains a single path P then
        For each combination, beta, of the nodes in the path
                generate pattern (beta U alpha)
                with support = minimum support of nodes in beta
Else
        For each item, i, in the header of the tree do
                begin
                    generate pattern beta (i U alpha) with support = i.support
                    construct beta's conditional pattern base;
                    construct beta's conditional FP-Tree, beta_tree;
                    If beta_tree is not empty then
                            FP-Growth (beta_tree, beta);
                end;
End;
```

c. *CT-Pro*

Algoritma ini berakar dari *FP-Growth* dimana modifikasi yang dilakukan adalah pada *tree* yang digunakan [SUC05]. Algoritma ini menggunakan struktur *Compressed FP-Tree* (*CFP-Tree*) dimana informasi dari sebuah *FP-Tree* diringkas dengan struktur yang lebih kecil atau ringan, sehingga baik pembentukan *tree* maupun *frequent itemset mining* yang dilakukan menjadi lebih cepat.

Langkah yang dilakukan oleh *CT-Pro* adalah sebagai berikut.

1. Menemukan *item-item* yang *frequent.*
2. Membuat *CFP-Tree*.
3. Melakukan *mining frequent patterns*.

Berikut ini adalah *pseudocode* dari *CT-Pro*.

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

```
Pseudocode CT-Pro :
Langkah 1 dan 2 :
Input   Dataset D, Support Threshold σ
Output CFP-Tree
begin
    // Step 1: Identify frequent items
    for each transaction t ∈ D
        for each item i ∈ t
            if i ∈ ItemTable
                Increment count of i
            else
                Insert i into GlobalItemTable with count = 1
            end if
        end for
    end for
    Sort GlobalItemTable in the order of descending frequency
    Assign an index for each frequent item in the GlobalItemTable
    // Step 2: Construct CT-Tree
    Construct the left most branch of the tree
    for each transaction t ∈ D
        Initialize mappedTrans
        for each frequent item i ∈ t
            mappedTrans = mappedTrans ∪ GetIndex(i)
        end for
        Sort mappedTrans in ascending order of item ids
        InsertToCFPTree(mappedTrans)
    end for
end
Procedure InsertToCFPTree(mappedTrans)
    firstItem := mappedTrans[1]
    currNode := root of subtree pointed by ItemTable[firstItem]
    for each subsequent item i ∈ mappedTrans
        if currNode has child representing i
            Increment count[firstItem-1] of the child node
        else
            Create child node and set its count[firstItem-1] to 1
            Link the node to its respective node-link
        end if
    end for
end
```


Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

```
Langkah 3 :
Input    CFP-Tree
Output Frequent Patterns FP
Procedure Mining
    for each frequent item i ∈ GlobalItemTable from the least to the
        most frequent
        Initialize LocalFrequentPatternTree with i as the root
        ConstructLocalItemTable(x)
        for each frequent item j ∈ LocalItemTable
            Attach i as a child of x
        end for
        ConstructLocalCFP-Tree(x)
        RecMine(x)
        Traverse the LocalFrequentPatternTree to print the frequent
        patterns
    end for
end
Procedure ConstructLocalItemTable(i)
    for each occurrence of node i in the CFP-Tree
        for each item j in the path to the root
            if j ∈ LocalItemTable
                Increment count of j
            else
                Insert j into LocalItemTable with count = 1
            end if
        end for
    end for
end
Procedure ConstructLocalCFPTree(i)
    for each occurrence of node i in the CFP-Tree
        Initialize mappedTrans
        for each frequent item j ∈ LocalItemTable in the path to the
        root
            mappedTrans = mappedTrans ∪ GetIndex(j)
        end for
        Sort mappedTrans in ascending order of item ids
        InsertToCFPTree(mappedTrans)
    end for
end
Procedure RecMine(x)
    for each child i of x
        Set all counts in LocalItemTable to 0
```


Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

```
        for each occurrence of node i in the LocalCFPTree
            for each item j in the path to the root
                Increment count of j in LocalCFPTree
            end for
        end for
        for each frequent item k ∈ LocalItemTable
            Attach k as a child of i
        end for
        RecMine(i)
    end for
  end
```

3.3.2 *Classification*

Cara kerja dari metode ini adalah mencari sebuah model yang mampu melakukan prediksi pada suatu data baru yang belum pernah ada kemudian mengelompokkannya ke dalam kelas-kelas tertentu. Algoritma yang dikembangkan dalam metode ini adalah *CMAR* dan *CSFP*.

a. *CMAR*

CMAR merupakan salah satu algoritma *classification* yang menggunakan dasar *association rules*. Teori *association rules* yang dipakai adalah *FP-Growth* sehingga algoritma *CMAR* memiliki langkah awal yang sama seperti *FP-Growth*, yaitu membuat *FP-Tree*. Perbedaan antara *FP-Growth* di *Association rules* dan *CMAR* adalah *FP-Growth* di dalam *CMAR* memperhatikan perbedaan kelas saat melakukan pencarian frequent item set.

Langkah-langkah dari *CMAR* adalah sebagai berikut.

1. Membaca *file*
2. Pencatatan frekuensi tiap item
3. Pembuatan *FP-Tree*
4. Pencatatan *Pattern*
5. *Prunning*
6. Perhitungan akurasi

Berikut adalah *pseudocode* dari *CMAR* yang diimplementasikan pada FIKUI Mining.

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

```
//s = support, conf = confidence, cov = coverage, D = Data Set
For each item ∈ D do
        if support item ≥ s
        Then insert item into F – list
Building FP-Tree
For each item ∈ F – list do
        Mining using FP-Growth using F-list as header array
        Traversed node having value item and look upward
Initializing candidate rules, building CR-tree
Define an empty rulesCollection
//Prunning Process
for each rules ∈ CR – tree
  if conf(r1) > conf (r2)
        then rulesCollection = rulesCollection U r1
  else if (conf(r1)==conf(r2)) ^ (supp (r1)> supp(r2))
        then rulesCollection = rulesCollection U r1
  else if (conf (r1)== conf (r2)) ^ (supp(r1)= supp(r2)) ^ (len(r1)>len(r2))
        then rulesCollection = rulesCollection U r1
for each attribute ∈ rulesCollection do
  calculate x^2 test
  if x^2 > significance level threshold
        then attribute is qualified
rulesCollection = rules with qualified attribute
define total FI = 0 , approved = 0
for each unmarked_rules ∈ rulesCollection do
  for each transaction ∈ D
        if rules C transaction
        then if class_rules ==class_transaction
                then
                totalFI ++
                coverage_level = totalFI/total_transaction
                if coverage_level >= coverage then mark_rules
for each marked_rules do
  for each transaction ∈ D do
  if rules C transaction
      then if class(rules)== class(transaction)
        then approved ++
accuracy = approved / total_transaction
```


Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

b. *CSFP*

Algoritma ini diusulkan dalam [SUC05] sebagai metode *classification* yang lebih efisien. Metode *classification* yang digunakan pada algoritma ini sama dengan *CMAR*, yaitu *associative classification*. Telah disebutkan sebelumnya bahwa *associative classification* memiliki tiga tahapan dasar. Akan tetapi, di dalam algoritma ini terdapat satu tahapan *associative classification* yang tidak dilakukan, yaitu membuat semua kemungkinan CAR. Dengan kata lain, algoritma *CSFP* menemukan *classifier* langsung tanpa class *association rules*.

Berikut adalah langkah-langkah algoritma *CSFP*.

1. Partisi menurut label kelas
2. Pembuatan *CFP-Tree*
3. *Mining frequent pattern* dengan *CT-Pro*
4. Pencarian C*andidate Class Rules* dari *CP-Tree*
5. *Prunning*

Berikut adalah *pseudocode* dari *CSFP* yang diimplementasikan pada FIKUI Mining [SUC05].

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

```
//s = support, conf = confidence, D = Data Set
If continuous attribute ∈ D
    Then discretize
For each class ∈ D do
    Partition <- item ∈ class
For each partition do
    For each frequent pattern f ∈ partition do
        CT-PRO mining
        If frequentPattern ∉ CP-Tree
            Then insert f into CP-Tree
        Else currTotal = (absSuppOfTheNode/confOfTheNode)
        100 + supp_abs(f)
        If supp_abs (f) > absSupportOfTheNode
            Then
                absSuppOfTheNode = supp_abs (f)
                classOfTheNode = class (f)
            confOfTheNode=(absSupportOfTheNode currTotal)*100
initiazile CandidateClassRules
for each frequent pattern f ∈      CP –Tree do
    traversed by depth-first search
    if conf(f)>y
        then pf = parent of f
            if(class(f)==class(pf)and  (errorRate(f)<errorRate(pf)))
                then candidateClassRules = candidatClassRules U f
Sort candidateClassRules in the rank descending order
Initialize classifierRules
While cases ∈ Dare not empty
    For each candidate rule r ∈ candidateClassRules
        For each cased ∈ D
            If r covers d
                Remove d from D
                If r is unmarked
                    Then do
                        classifierRules= classifierRules U r
                        mark r
set the majority class in remaining cases in D as the default class and remove all the
remaining cases
```

3.3.3 *Clustering*

Cara kerja dari metode ini adalah mengelompokkan data dalam sebuah *cluster* berdasarkan kemiripannya. Prinsipnya adalah memaksimalkan kemiripan dalam sebuah *cluster*, dan meminimalisasikan kemiripan antar *cluster*. Jadi data-data yang berada pada sebuah *cluster* akan memiliki kemiripan yang tinggi, dan sebaliknya data-data pada *cluster* yang berbeda

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

akan memiliki nilai kemiripan yang rendah. Algoritma yang dikembangkan dalam metode ini adalah *K-Means*, *Fuzzy C-Means* dan *Nearest Neighbour*.

a. *K-Means*

K-Means adalah salah satu algoritma yang berguna untuk memecahkan masalah pengelompokan data. *K-Means* bertujuan untuk mempartisi objek-objek atau data-data yang menjadi beberapa kluster sejumlah K. Kluster-kluster tersebut mempunyai nilai tengah yang disebut dengan *centroid*. Dengan menggunakan *K-Means*, jarak elemen-elemen antarkluster akan dibuat seminimal mungkin.

Berikut adalah cara kerja dari *K-Means*.

1. Melakukan inisialisasi nilai-nilai *centroid* secara acak.
2. Menaruh setiap data kepada kluster yang mempunyai nilai *centroid* yang paling dekat selisihnya.
3. Meng-*update* nilai setiap *centroid* dengan merata-ratakan data dalam setiap kluster.
4. Memeriksa apakah nilai *centroid* sudah sama atau belum.
5. Melakukan kembali langkah 2 jika selama nilai *centroid* masih belum sama.

Berikut adalah *pseudocode* dari *K-Means* yang diimplementasikan pada FIKUI Mining.

```
If data wants to be sorted
    Sort data
    Initialize Centroid value based in the sorted data
Else
    Initialize Centroid value randomly
End If
While maximum loop not reached AND tolerance value value still above threshold
    Assign data into cluster
    Assign new centroid
    Compare new centroid to centroid (acquire tolerance value)
    If tolerance value still above threshold
        Update centroid value with new centroid value
    End If
End While
```


Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

b. *Fuzzy C-Means*

Fuzzy C-Means adalah algoritma *clustering* yang memungkinkan suatu data tergolongkan ke dalam lebih dari satu kluster. *Fuzzy* memakai konsep *soft computing* dimana sesuatu dapat tergolongkan ke dalam suatu kelompok secara parsial atau tidak menyeluruh. Konvergensi kluster adalah suatu yang menjadi keunggulan algoritma ini dibandingkan dengan algoritma *K-Means*.

Berikut adalah cara kerja dari *Fuzzy C-Means*.

1. Inisiasi fungsi keanggotaan.
2. *Update* nilai *centroid* berdasarkan fungsi keanggotaan dari data-data yang ada.
3. *Update* fungsi keanggotaan berdasarkan nilai *centroid* dari data-data yang ada.
4. Pembandingan matriks fungsi keanggotaan yang baru dengan yang lama. Jika selisih terbesar dari elemen kedua matrks tersebut lebih kecil dari batas toleransi tertentu maka algoritma selesai. Jika tidak, kembali lakukan langkah 2.

Berikut adalah *pseudocode* dari *Fuzzy C-Means:*

```
If data wants to be sorted
    Sort data
End If
Initialize membership function (random)
    While maximum loop not reached AND tolerance value still above threshold
        Update centroid value
        Update new membership function
        Compare new membership function to membership function (acquire
        tolerance value)
        Update membership function with new membership function value
End While
```

c. *Nearest Neighbour*

Nearest Neighbour adalah algoritma *clustering* yang membagi kumpulan data menjadi beberapa kluster tanpa terlebih dahulu diketahui jumlah kluster yang akan dibuat. Konsep terpenting pada algoritma ini adalah konsep "tetangga tedekat", dimana suatu data akan dikelompokan

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

kedalam suatu kelompok tertentu berdasarkan kedekatan dengan data lain yang dekat dengan data tersebut.

Berikut adalah cara kerja dari algoritma *Nearest Neighbour*:

1. Memasukan data pertama kedalam kluster pertama.
2. Membandingkan tiap data (mulai dari data ke 2) dengan "tetangga terdekatnya".
3. Membandingkan selisihnya, jika selisihnya lebih kecil atau sama dengan suatu ambang batas tertentu, maka data tersebut masuk ke kluster dari "tetangga terdekatnya". Jika selisihnya lebih besar, data tersebut dimasukan kedalam sebuah kluster baru.

Berikut adalah pseudocode dari *Nearest Neighbour*.

```
If data wants to be sorted
    Sort data
End If
Initialize first data to be put into cluster-0
For Each data starting with the second data
    Compare data with nearest neighbour (by iterating the data that has included in a
    cluster)
    If the distance between the data is <= threshold
        Assign data to the cluster of previous data
    Else
        Assign data to a new cluster
    End If
End For
```

3.4 Perancangan *Sequence Diagram*

Sequence diagram adalah bentuk model yang paling umum digunakan dalam menggambarkan *interaction diagram*. *Interaction diagram* menggambarkan bagaimana sekelompok objek saling berkolaborasi dengan aturan tertentu yang berlaku. *Sequence diagram* menggambarkan setiap objek tersebut dan interaksi pesan dan data yang terjadi di antara objek tersebut secara sekuensial terhadap waktu. *Sequence diagram* ini dibentuk dari *use-case specifications* yang telah disusun pada subbab sebelumnya.

Bentuk *sequence diagram* akan menyerupai *timeline* proses antarobjek yang ada disertai transfer data antarobjek tersebut. Semua kemungkinan alur kejadian sistem ini digambarkan secara berurutan dari atas ke bawah secara sekuensial. Alur kemungkinan kejadian utama pada sebuah sub *use-case* digambarkan di

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

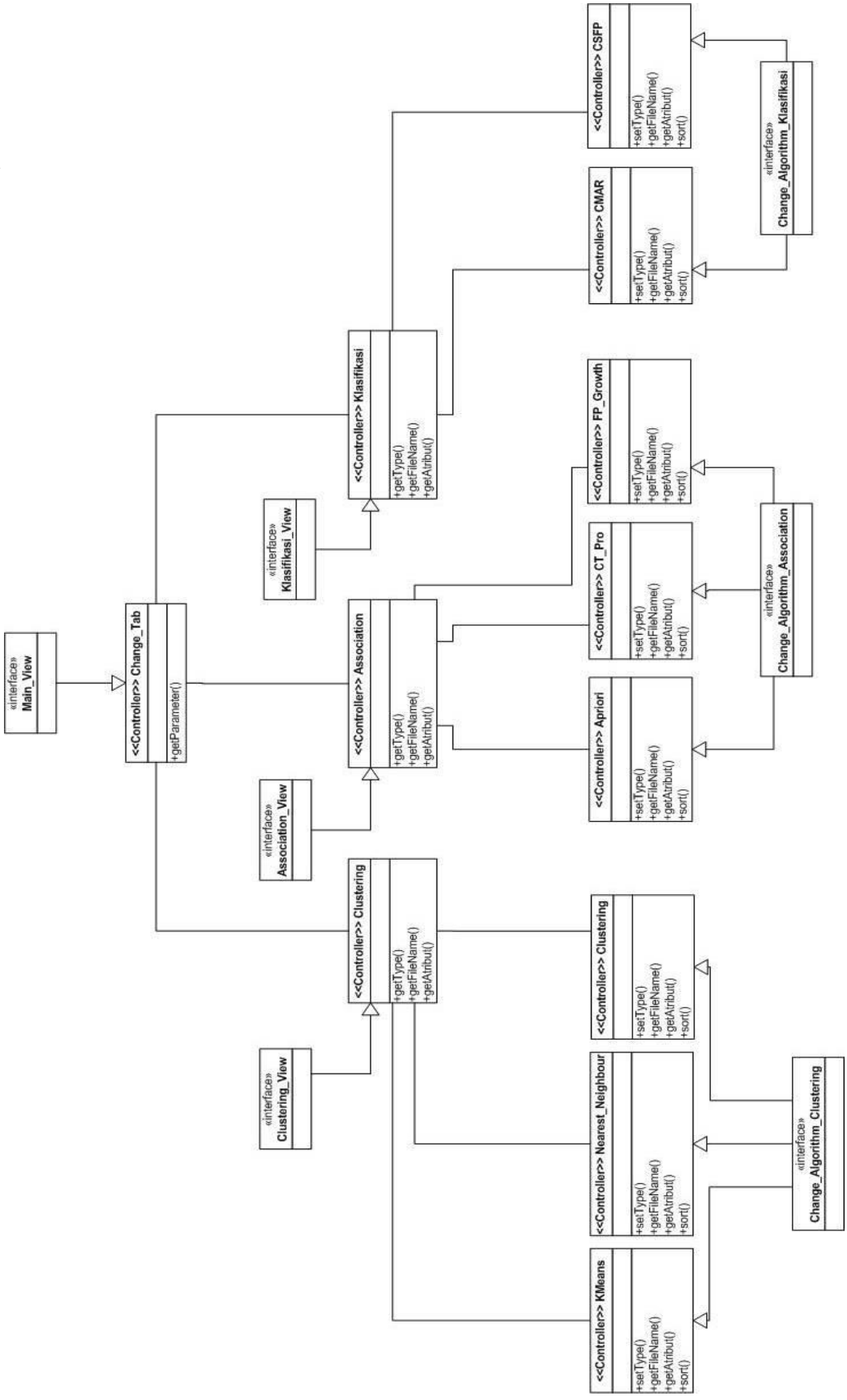
bagian paling atas dari diagram tersebut. Sedangkan alur-alur kemungkinan lainnya yang dapat terjadi pada sub *use-case* tersebut digambarkan di bawahnya. *Sequence diagram* dari FIKUI Mining dan penjelasan singkat mengenainya dapat dilihat di Lampiran A.

3.5 Perancangan *Class Diagram*

Berdasarkan *sequence diagram* yang telah dibuat, tim pengembang mengetahui objek-objek yang dibutuhkan beserta kapabilitas dari setiap objek. Kemudian objek-objek tersebut direpresentasikan ke dalam *class diagram*. *Class diagram* adalah suatu pemodelan yang menggambarkan kelas-kelas yang terdapat dalam suatu sistem, beserta hubungan antarkelas dan atribut yang dimiliki oleh masing-masing kelas. Diagram ini juga dapat diinterpretasikan sebagai model yang menggambarkan rancangan detil sistem berorientasikan objek.

Gambar 3.3 menunjukkan *class diagram* untuk menggambarkan objek-objek apa saja yang dibutuhkan dalam mengkonstruksi sebuah sistem.

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008



Gambar 3.3: ***Class Diagram***

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

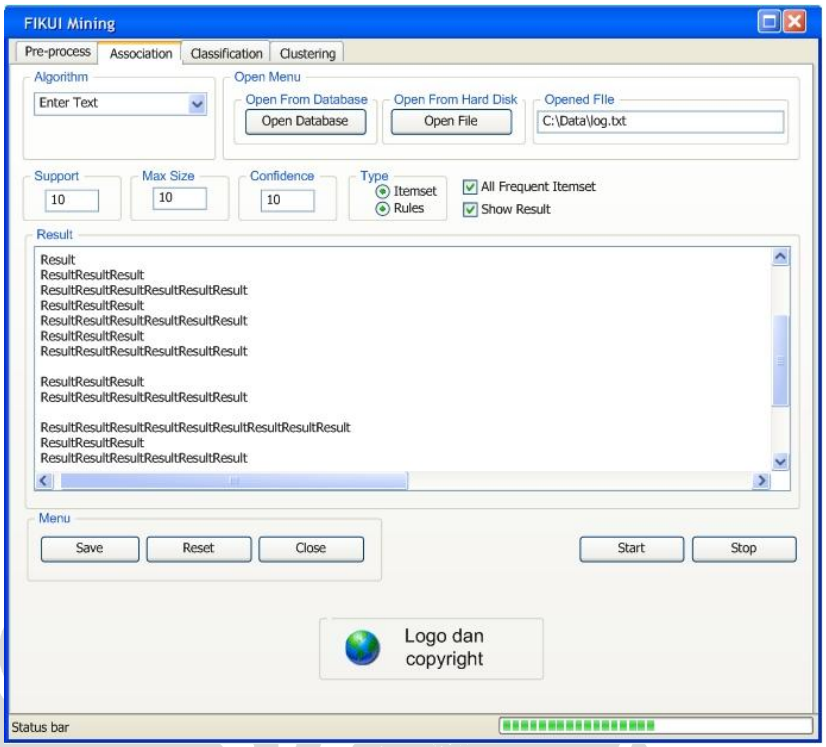
Setelah semua *class* terdefinisi melalui *class diagram* yang telah dibuat, proses implementasi dapat dimulai dengan memanfaatkan kerangka yang telah terbentuk dari *class diagram* dan *sequence diagram*.

3.6 Perancangan *User Interface*

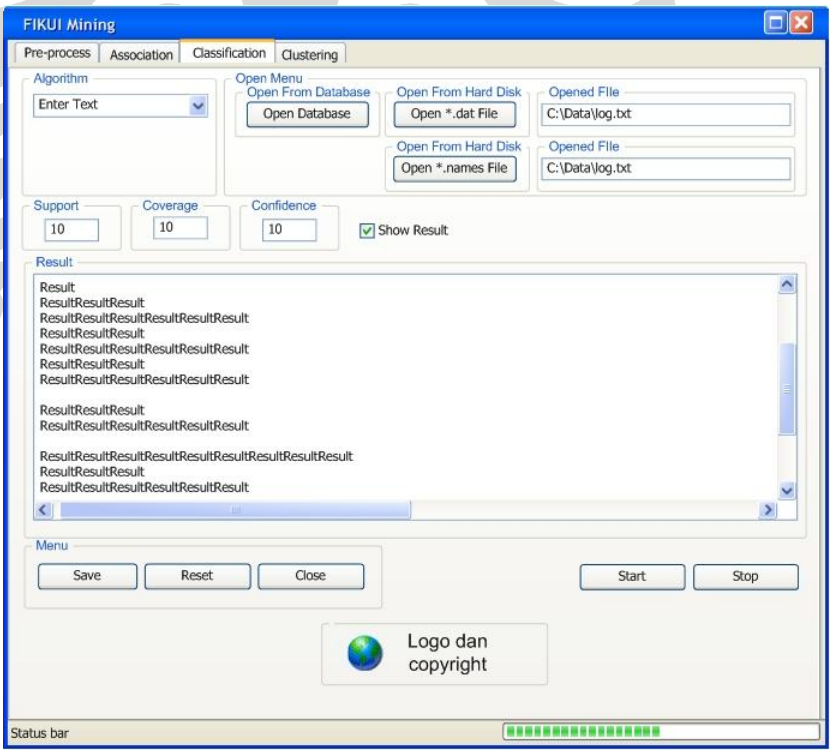
Perancangan *user-interface* sistem dilakukan dalam beberapa proses. Proses pertama untuk perancangan *user-interface* adalah dengan membaca *Use-Case Specification.* Dari *Use-Case Specification* dapat dibuat perancangan *user-interface* yang direpresentasikan melalui gambar. Gambar dibuat dengan perangkat lunak pengolah gambar pada umumnya. Gambar yang sudah dibuat akan dijadikan dasar implementasi *user-interface* sistem.

Perancangan *user-interface* memberikan sebuah gambaran dasar bagi implementasi sistem. Alur halaman sistem dapat diimplementasikan berdasarkan perancangan *user-interface* yang masih berupa gambar. Hal ini disebabkan karena gambar perancangan *user-interface* merepresentasikan alur halaman sistem yang nantinya akan diimplementasikan. Dengan cara ini proses implementasi akan sangat terbantu khususnya dalam masalah alur halaman sistem. Perancangan *user-interface* dilakukan untuk semua *use-cases* yang ada. Gambar 3.4, 3.5, dan 3.6 adalah rancangan *User Interface* untuk setiap teknik *data mining* yang diimplementasikan pada FIKUI Mining.

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

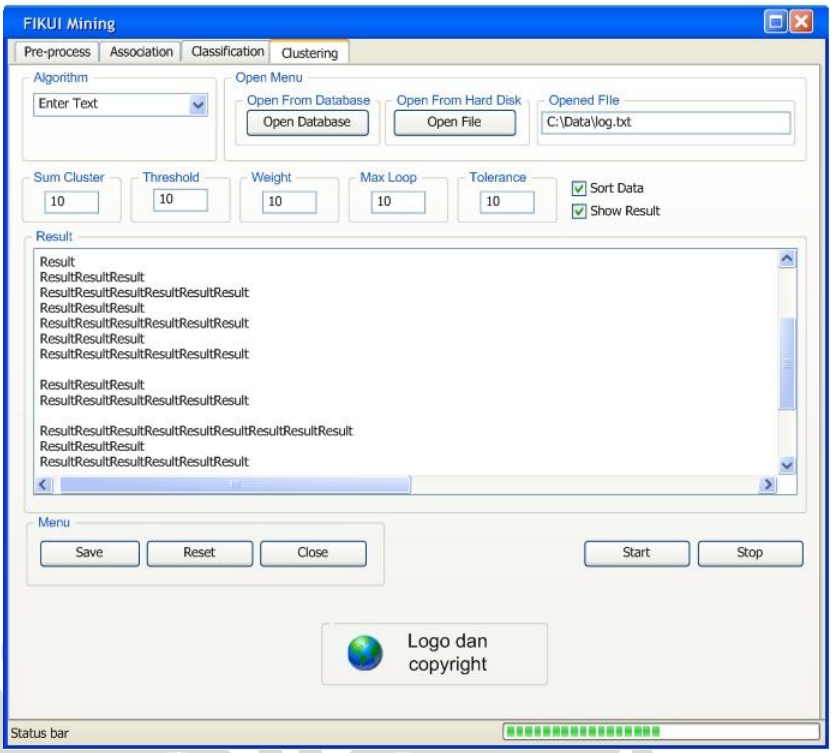


Gambar 3.4: Rancangan *User Interface* untuk Teknik *Association*



Gambar 3.5: Rancangan *User Interface* untuk Teknik *Classification*

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008



Gambar 3.6: Rancangan *User Interface* untuk Teknik *Clustering*

Keterangan:

- *Textbox support:* besaran yang menentukan frekuensi kemunculan suatu nilai di dalam data set. Satuan dalam persen (%).
- *Textbox confidence:* besaran suatu rules bila dibandingkan dengan seluruh rules identik. Satuan dalam persen (%).
- *Textbox max size:* besaran yang menentukan panjang dari *frequent itemset* yang mungkin.
- *Textbox coverage:* besaran data asli yang dapat diklasifikasikan dengan menggunakan rules.
- *Textbox sum cluster:* banyaknya jumlah kluster yg diinginkan (hanya untuk algoritma *K-Means* dan *Fuzzy C-Means*).
- *Textbox threshold:* besaran yang menentukan nilai ambang batas (hanya untuk algoritma *Nearest Neighbour*).
- *Textbox weight:* untuk menentukan besar bobot pada sebaran data (hanya untuk algoritme *Fuzzy C-Means*).
- *Textbox max loop:* untuk menentukan nilai *looping* maksimum (hanya untuk algoritma *K-Means* dan *Fuzzy C-Means*).

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

- *Textbox tolerance:* untuk menentukan nilai toleransi (hanya untuk algoritma *K-Means* dan *Fuzzy C-Means*).

Untuk melihat lebih jelas mengenai rincian *interface* sistem setiap *use-case*, akan ditampilkan *screenshot* sistem pada bab 4.

3.7 Hambatan dan Strategi untuk Menghadapinya

Selama pelaksanaan proses ini, tim mengalami berbagai hambatan dalam memenuhi tanggung jawabnya. Berikut adalah hambatan-hambatan yang dimaksud.

- Sulit untuk menemui para pengembang sebelumnya, yaitu orang-orang yang mengimplementasikan algoritma-algoritma yang terdapat di FIKUI Mining . Hal ini disebabkan oleh kesibukan para pengembang sebelumnya yang sudah bekerja secara profesional setelah mereka lulus, bahkan sudah ada yang bekerja di luar pulau jawa sehingga tidak lagi memungkinkan tim pengembang yang sekarang untuk bertemu secara tatap muka. Namun disela-sela kesibukan mereka, kami masih sempat berkomunikasi melalui *E-Mail* maupun *chating* secara *on-line.*
- Perbedaaan jadwal dan kegiatan anggota tim diluar pengembangan sistem menjadikan sulitnya komunikasi secara bersama-sama dalam memecahkan masalah yang timbul selama masa pengembangan. Untuk mengatasi masalah ini, tim pengembang membuat jadwal untuk kumpul dan mengerjakan sistem secara bersama-sama pada malam hari dengan menginap di ruang lab yang telah disediakan.
- Dokumentasi kode program dari pengembang sebelumnya yang kurang lengkap menjadikan tim pengembang yang sekarang mengalami kesulitan dalam mempelajari kode-kode program yang ada. Dengan menelusuri sedikit demi sedikit, tim pengembang berusaha untuk memahami keseluruhan kode program yang ada.

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

BAB 4
IMPLEMENTASI

Pada bagian ini, akan dijelaskan bagaimana proses implementasi dilakukan. Selain itu, akan dijelaskan pula tampilan antarmuka dari FIKUI Mining dan peningkatan unjuk kerja algoritma.

4.1 Proses Implementasi

Proses implementasi sistem FIKUI Mining, dilakukan dengan mengacu kepada desain logikal yang sudah dibuat sebelumnya. Seluruh sistem dibuat dengan menggunakan bahasa pemrograman Visual C++ .NET dan menggunakan IDE Visual Studio 2005 untuk melakukan *coding, running, debugging,* serta *testing* program. *Delivery* sistem ini berupa sebuah sistem berbasis *desktop* (*desktop-based application*).

Dalam implementasi sistem FIKUI Mining ini, kami menggunakan mesin dengan spesifikasi sebagai berikut:

- Processor: Intel Pentium 4, 2.4 GHz
- Memory: 1,5 GB DDR1
- Operating System: Windows XP Professional
- Development Tools: Microsoft Visual Studio 2005

Bahasa pemrograman Visual C++ .NET berikut IDE Visual Studio 2005 dipakai dengan beberapa pertimbangan berikut:

1. Memiliki unjuk kerja yang lebih cepat dibandingkan dengan bahasa pemrograman Java,
2. Memudahkan kami dalam membuat tampilan keseluruhan antarmuka sistem,
3. Memudahkan kami dalam melakukan *debugging* dan *testing* program.

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

Modul-modul yang diimplementasikan meliputi 3 modul utama:

1. Modul *Association*

   Modul *association* memuat algoritma-algoritma *Apriori*, *CT-Pro*, dan *FP-Growth*. Algoritma-algoritma ini digunakan untuk menemukan aturan assosiatif antara kombinasi suatu item.

2. Modul *Classification*

   Modul *classification* memuat algoritma-algoritma *CMAR* dan *CSFP*. Algoritma-algoritma ini digunakan untuk menemukan model atau fungsi yang menjelaskan suatu kelas data, dengan tujuan untuk dapat memperkirakan kelas dari suatu objek yang belum diketahui labelnya.

3. Modul *Clustering*

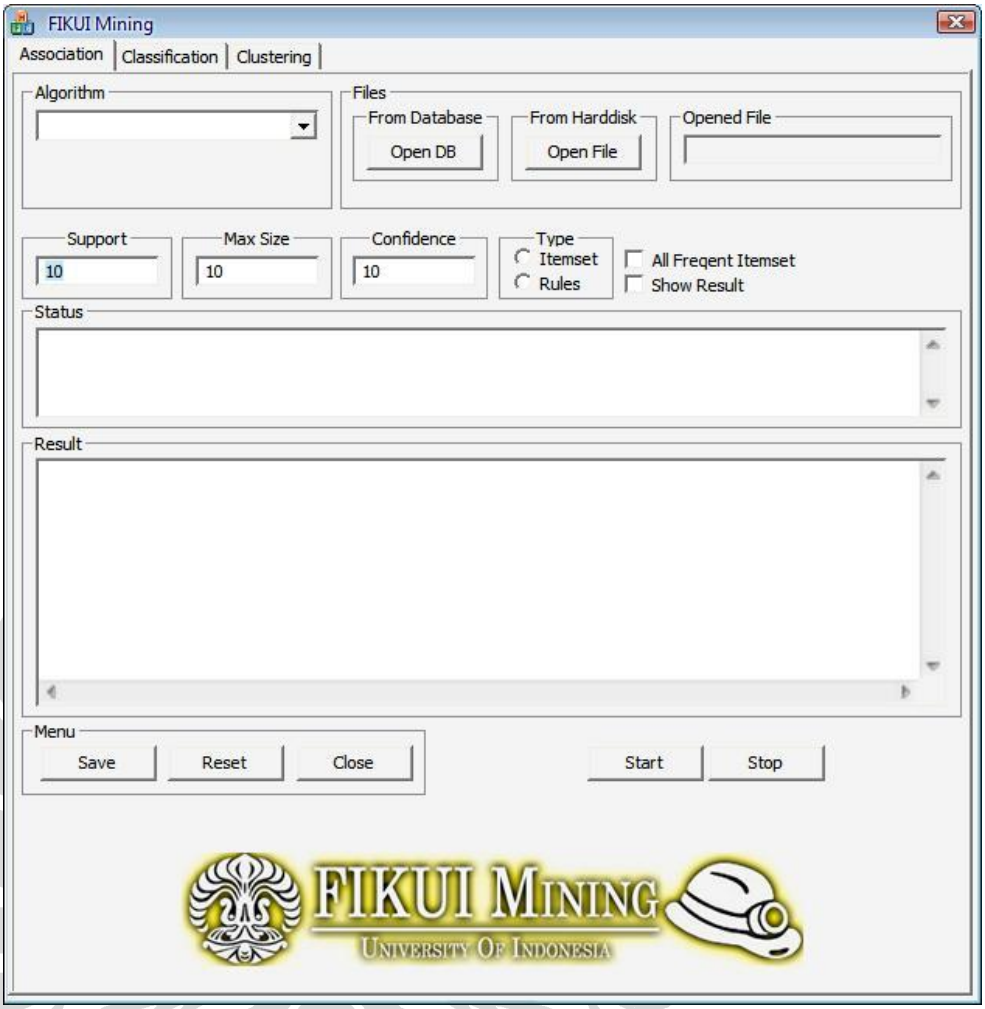
   Modul *clustering* memuat algoritma-algoritma *Fuzzy C-Means*, *K-Means*, dan *Nearest Neighbour*. Algoritma-algoritma ini juga digunakan untuk melakukan pengelompokan data seperti pada modul *classification*. Perbedaannya adalah, dalam *clustering* kita mencari hubungan pada data tanpa menggunakan referensi kelas data tertentu.

4.2 Tampilan Antarmuka

Pada bagian ini, akan dijelaskan tampilan antarmuka dari sistem FIKUI Mining. Penjelasan tampilan antarmuka ini kami bagi per modul.

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

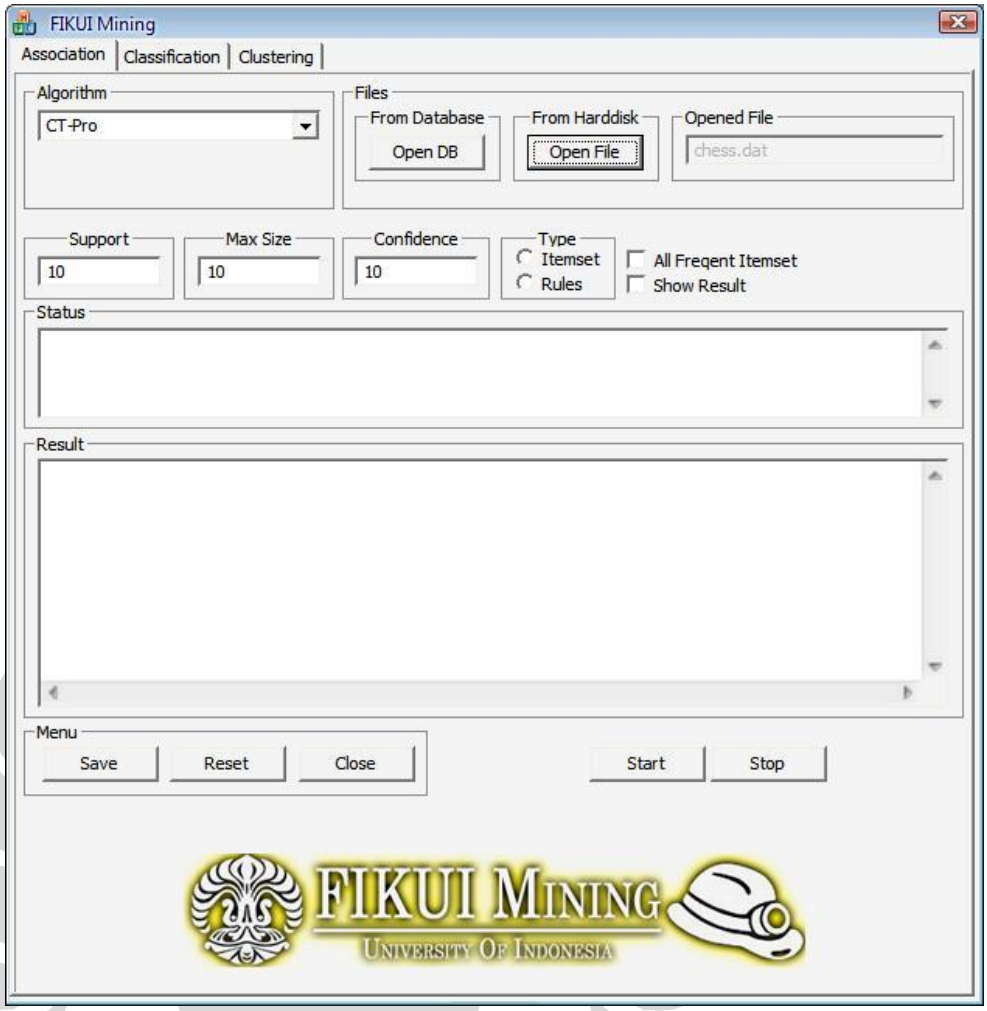
4.2.1 Modul *Association*



Gambar 4.1: Modul *Association* - Tampilan Awal

Gambar 4.1 adalah tampilan awal sistem FIKUI Mining untuk modul *association*. *User* dapat memilih algoritma-algoritma *association* dengan memilih *combo box* "Algorithm". Kemudian, *user* dapat memilih file masukan yang akan digunakan dengan menekan tombol "Open File". Apabila algoritma dan file masukan sudah terpilih, maka sistem akan menampilkan keterangan seperti gambar 4.2.

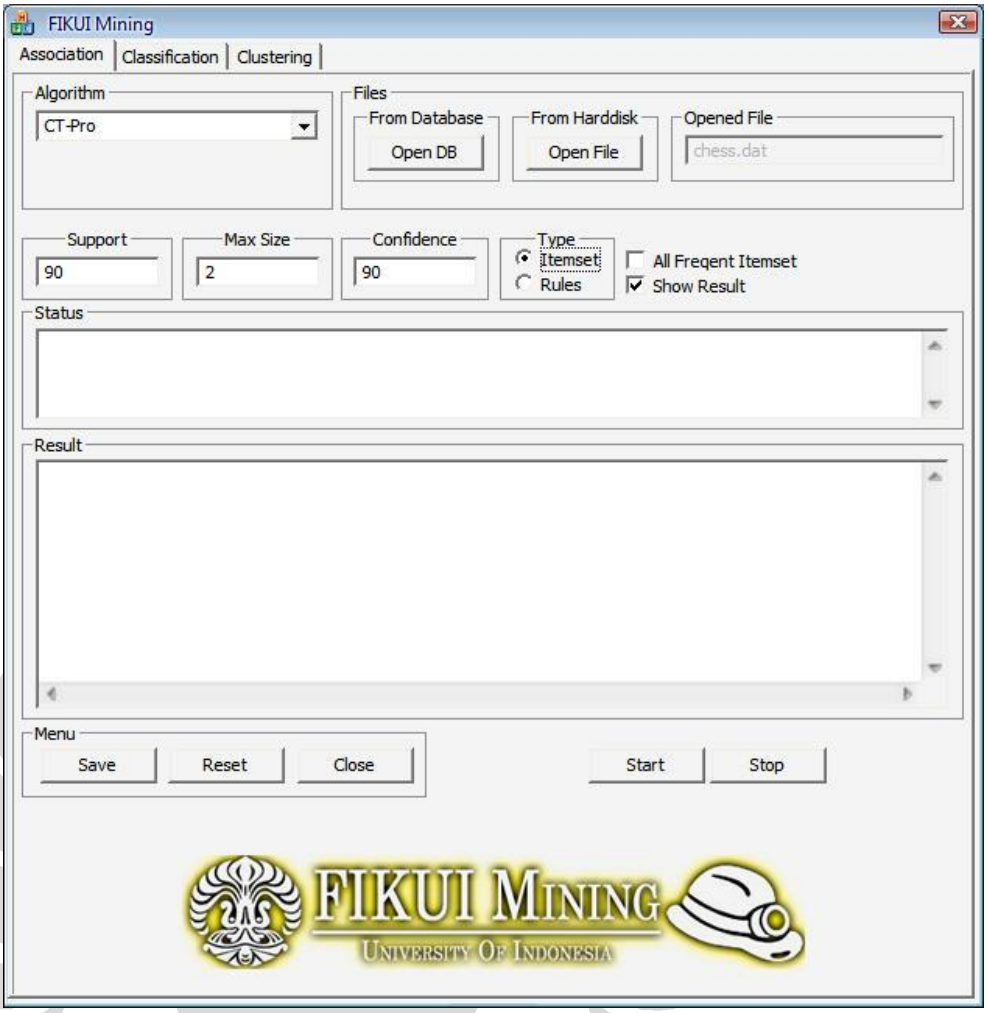
Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008



Gambar 4.2: Modul *Association* - Memilih Algoritma dan File Input

Pada gambar 4.2 di atas, dapat kita lihat *Combo box* "Algorithm" sudah terisi dengan algoritma yang dipilih oleh *user*. Begitu pula dengan *textarea* "Opened File", sudah terisi dengan file masukan yang dipilih. *User* dapat memilih nilai *support, maximum size,* dan *confidence* yang dikehendaki dengan mengisi *textarea* "Support", "Max Size", dan "Confidence". Untuk nilai *support* dan *confidence,* masukkannya berupa bilangan bulat 1-100. Sedangkan untuk nilai *maximum size,* masukkannya berupa bilangan bulat 1-5. Gambar 4.3 menunjukkan *user* sudah mengisi nilai-nilai tersebut.

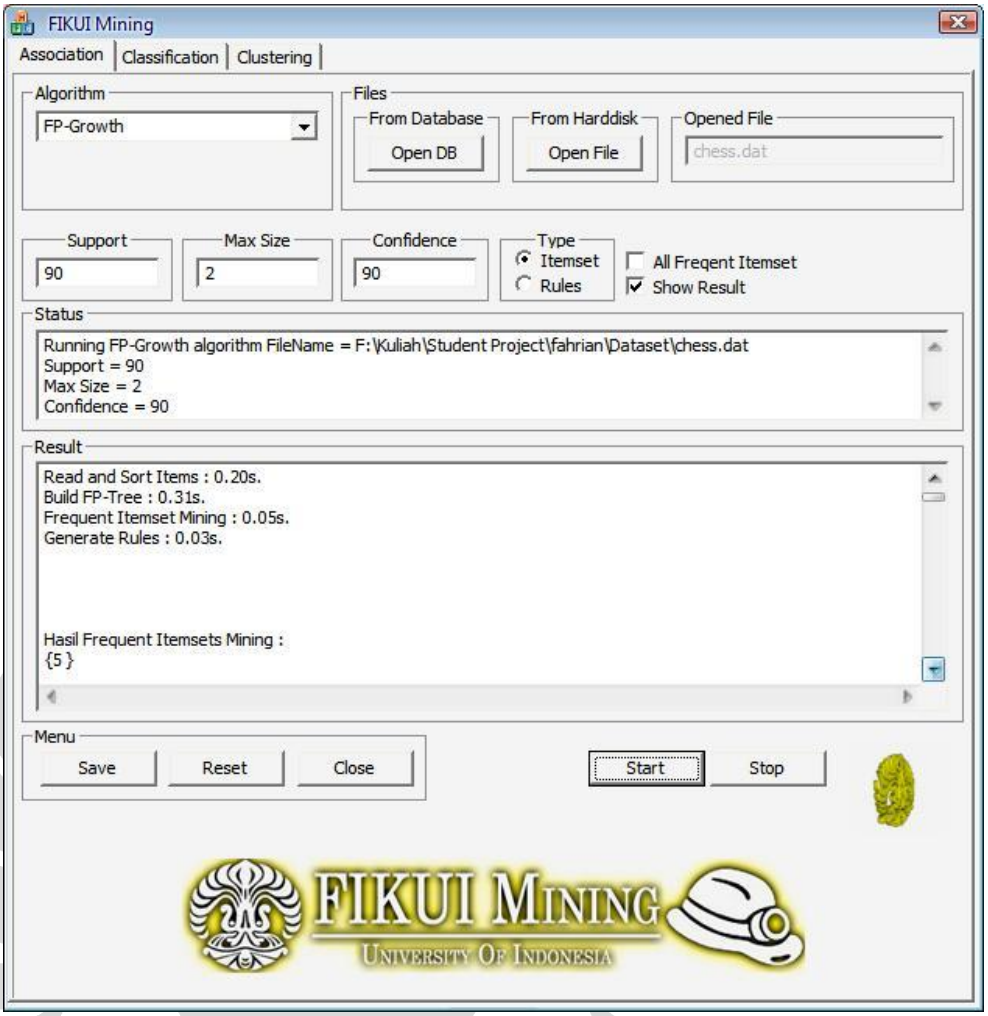
Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008



Gambar 4.3: Modul *Association* – Mengisi Nilai *Support, Max Size,* dan *Confidence*

Jika *user* telah memilih algoritma, memilih file masukan, dan mengisi nilai-nilai *support, maximum size,* dan *confidence,* maka *user* dapat menekan tombol "Start" untuk mulai menjalankan algoritma tersebut. Hasilnya akan ditampilkan seperti gambar 4.4.

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

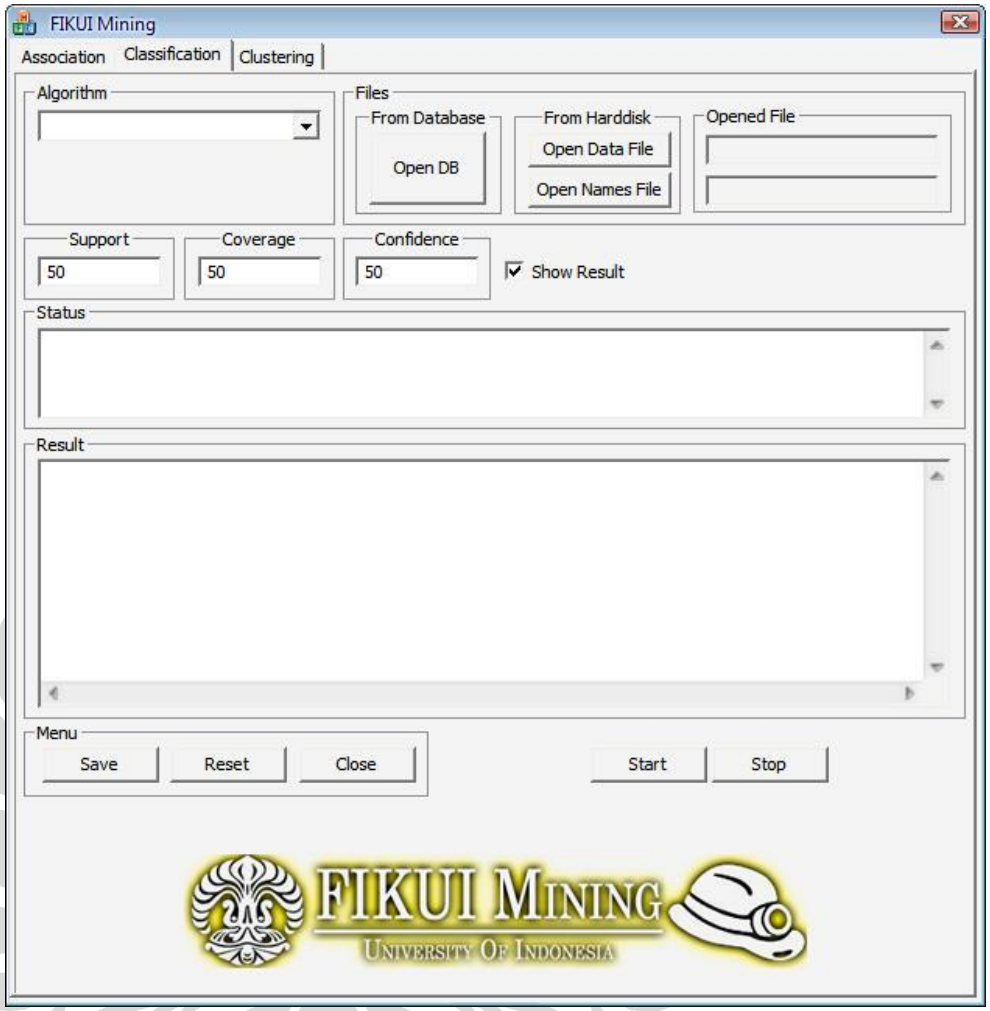


Gambar 4.4: Modul *Association* – Hasil Eksekusi Algoritma

User dapat menyimpan hasil yang ditampilkan pada sistem, dengan cara menekan tombol "Save". Hasil tersebut akan disimpan ke dalam sebuah file berekstensi *txt*. Jika *user* ingin membersihkan tampilan sistem dari hasil eksekusi algoritma sebelumnya, maka *user* dapat menekan tombol "Reset". Hal ini menyebabkan tampilan sistem kembali ke tampilan awal seperti pada gambar 4.1.

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

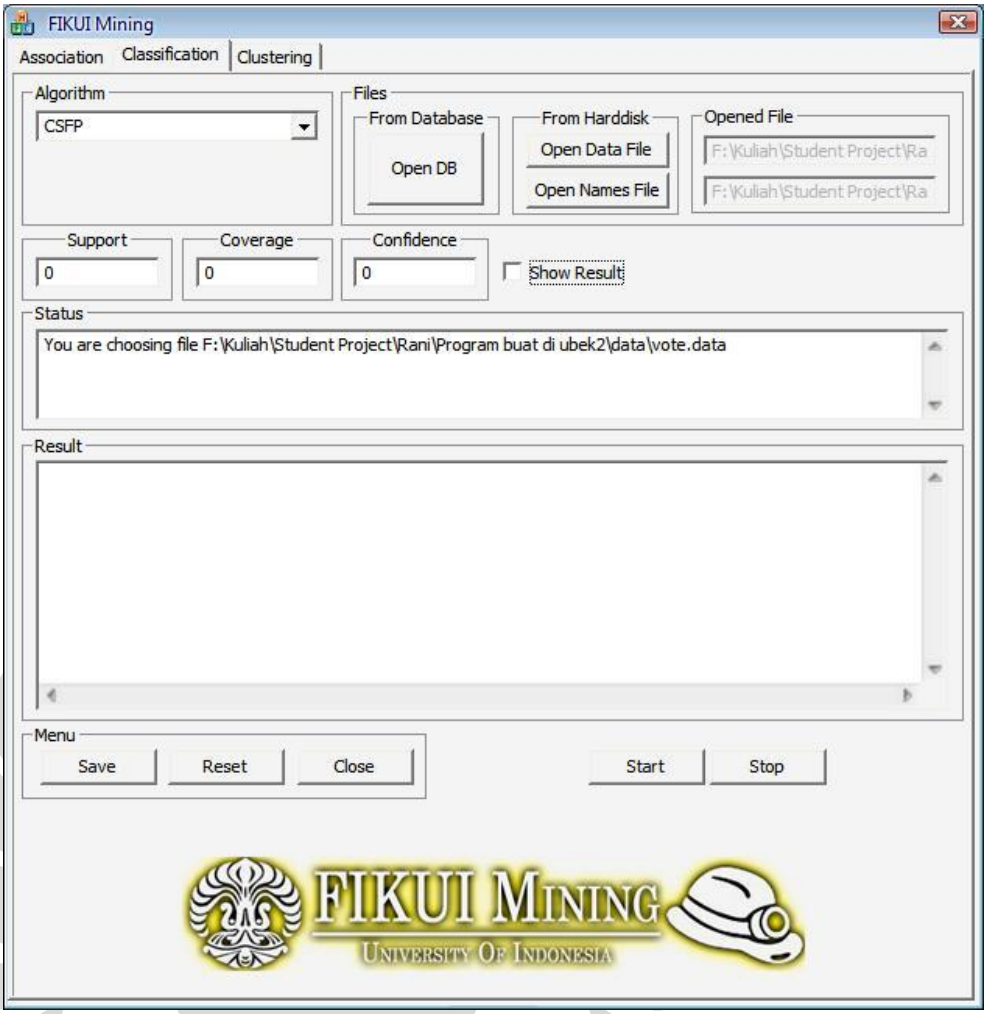
4.2.2 Modul *Classification*



Gambar 4.5: Modul *Classification* - Tampilan Awal

Gambar 4.5 adalah tampilan awal sistem FIKUI Mining untuk modul *classification*. *User* dapat memilih algoritma-algoritma *classification* dengan memilih *combo box* "Algorithm". Berbeda dengan modul *association,* pada modul ini *user* harus memilih 2 file sebagai file masukan. *User* dapat menekan tombol "Open Data File" dan "Open Names file" untuk memilih kedua file masukan. Apabila algoritma dan file masukan sudah terpilih, maka sistem akan menampilkan keterangan seperti gambar 4.6:

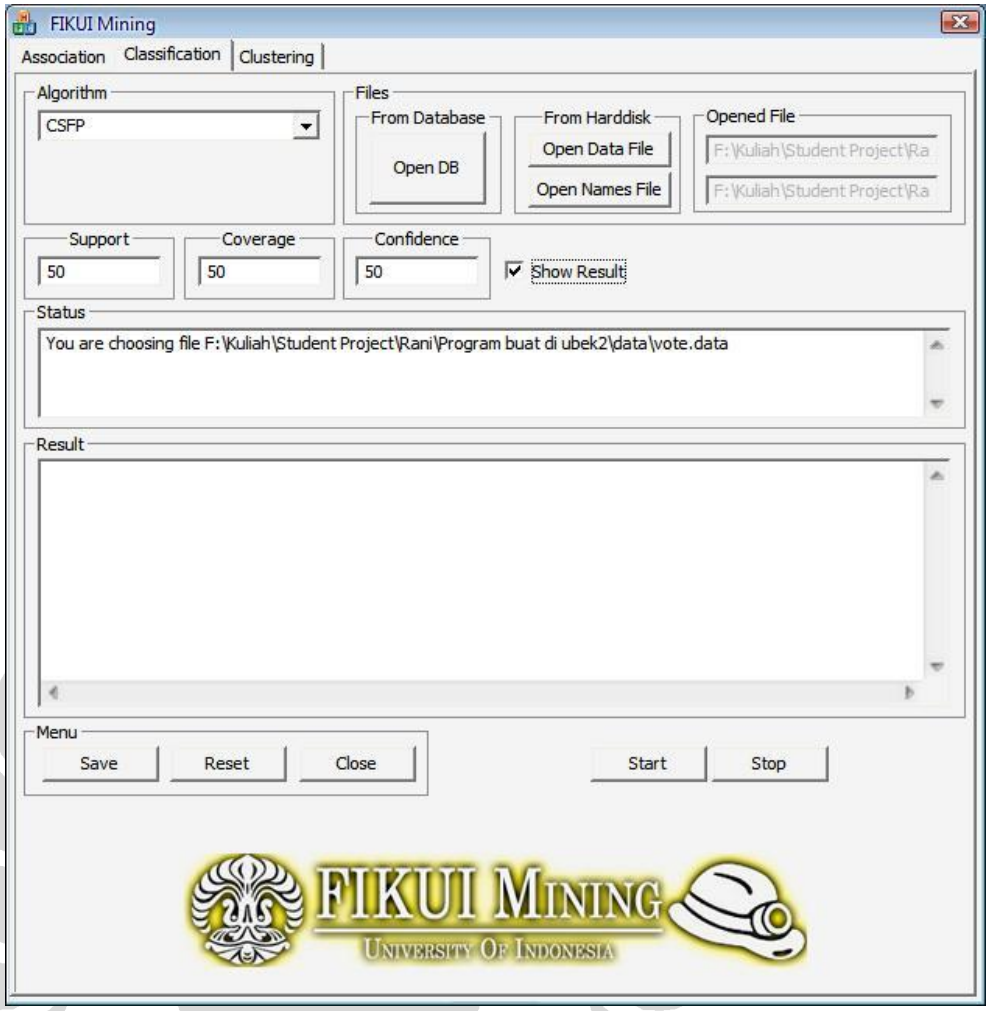
Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008



Gambar 4.6: Modul *Classification* - Memilih Algoritma dan File Input

Pada gambar 4.6 di atas, dapat kita lihat *Combo box* "Algorithm" sudah terisi dengan algoritma yang dipilih oleh *user*. Begitu pula dengan kedua *textarea* "Opened File", sudah terisi dengan file masukan yang dipilih. *User* dapat memilih nilai *support, coverage,* dan *confidence* yang dikehendaki dengan mengisi *textarea* "Support", "*Coverage*", dan "Confidence". Untuk nilai *support, coverage* dan *confidence,* masukkannya berupa bilangan bulat 1-100. Gambar 4.7 menunjukkan *user* sudah mengisi nilai-nilai tersebut.

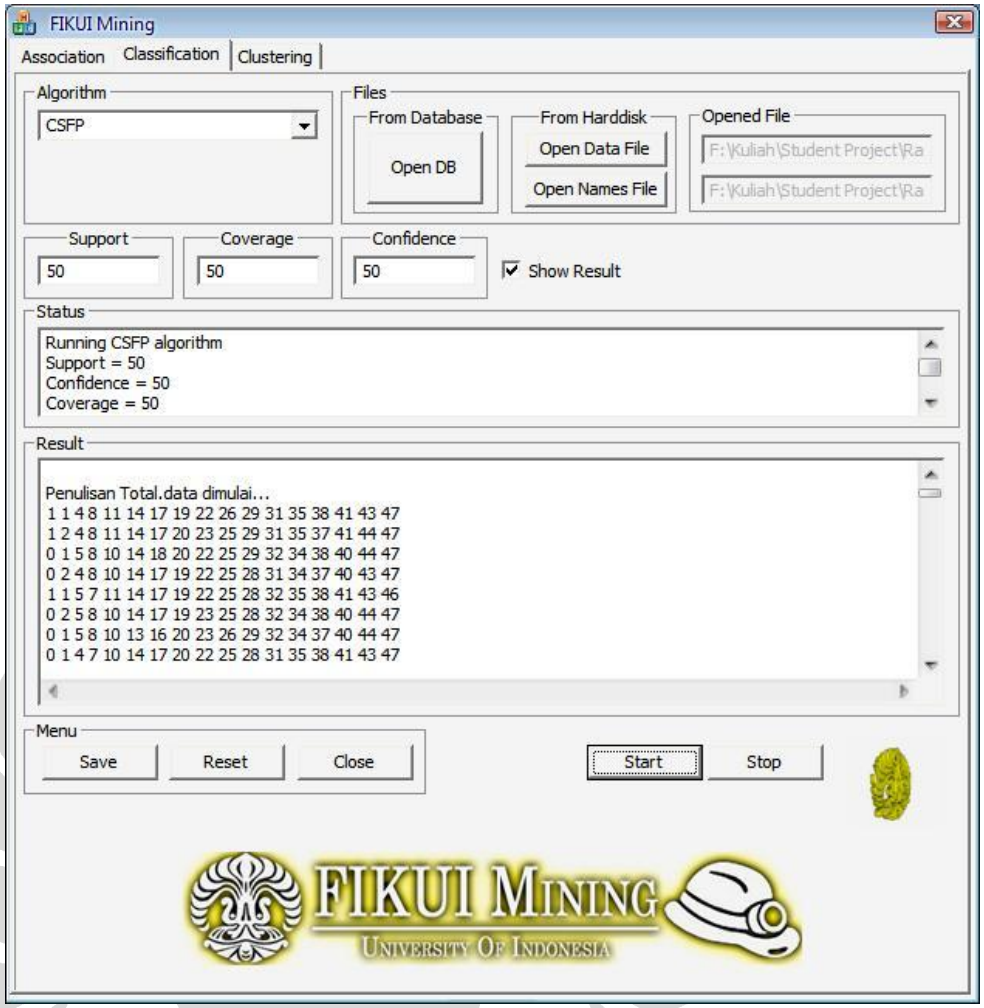
Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008



Gambar 4.7: Modul *Classification* – Mengisi Nilai *Support, Max Size, dan Confidence*

Jika *user* telah memilih algoritma, memilih file masukan, dan mengisi nilai-nilai *support, coverage,* dan *confidence,* maka *user* dapat menekan tombol "Start" untuk mulai menjalankan algoritma tersebut. Hasilnya akan ditampilkan seperti gambar 4.8.

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

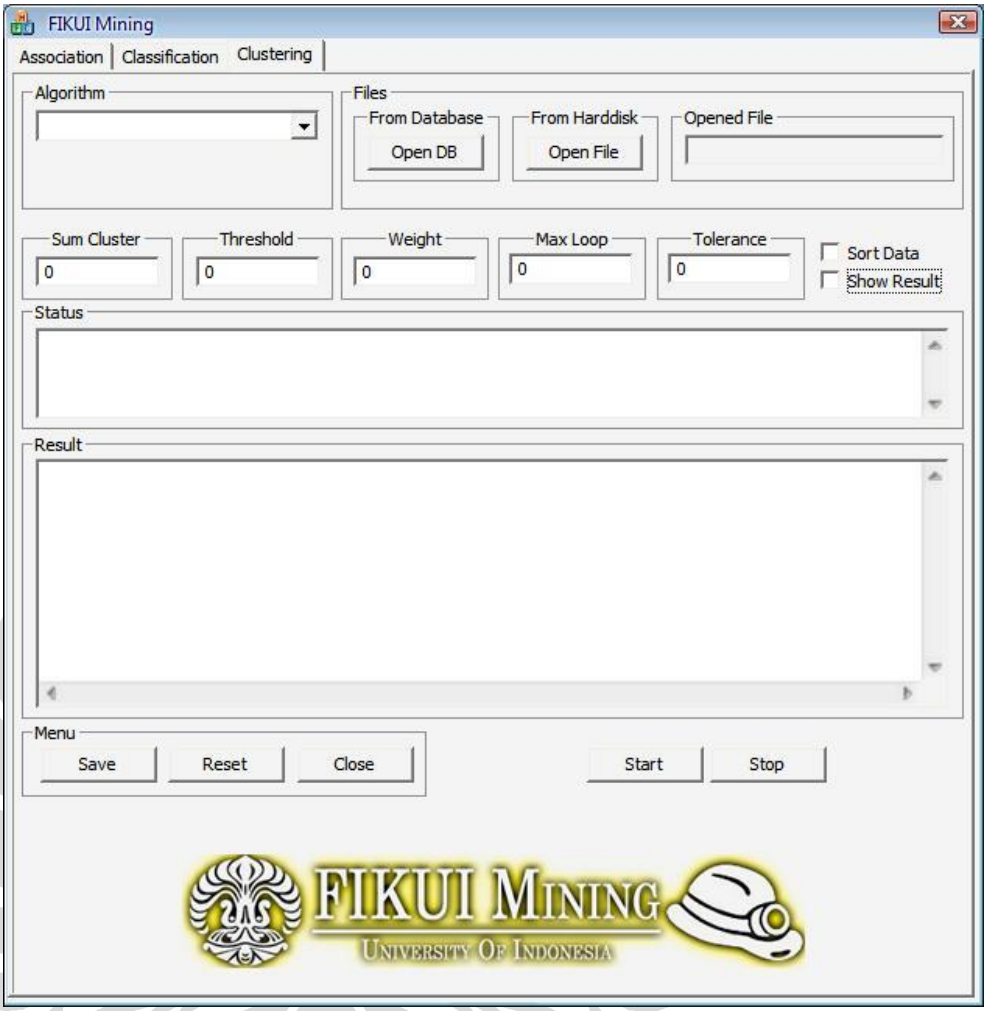


Gambar 4.8: Modul *Classification* – Hasil Eksekusi Algoritma

User dapat menyimpan hasil yang ditampilkan pada sistem, dengan cara menekan tombol "Save". Hasil tersebut akan disimpan ke dalam sebuah file berekstensi *txt*. Jika *user* ingin membersihkan tampilan sistem dari hasil eksekusi algoritma sebelumnya, maka *user* dapat menekan tombol "Reset". Hal ini menyebabkan tampilan sistem kembali ke tampilan awal seperti pada gambar 4.5.

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

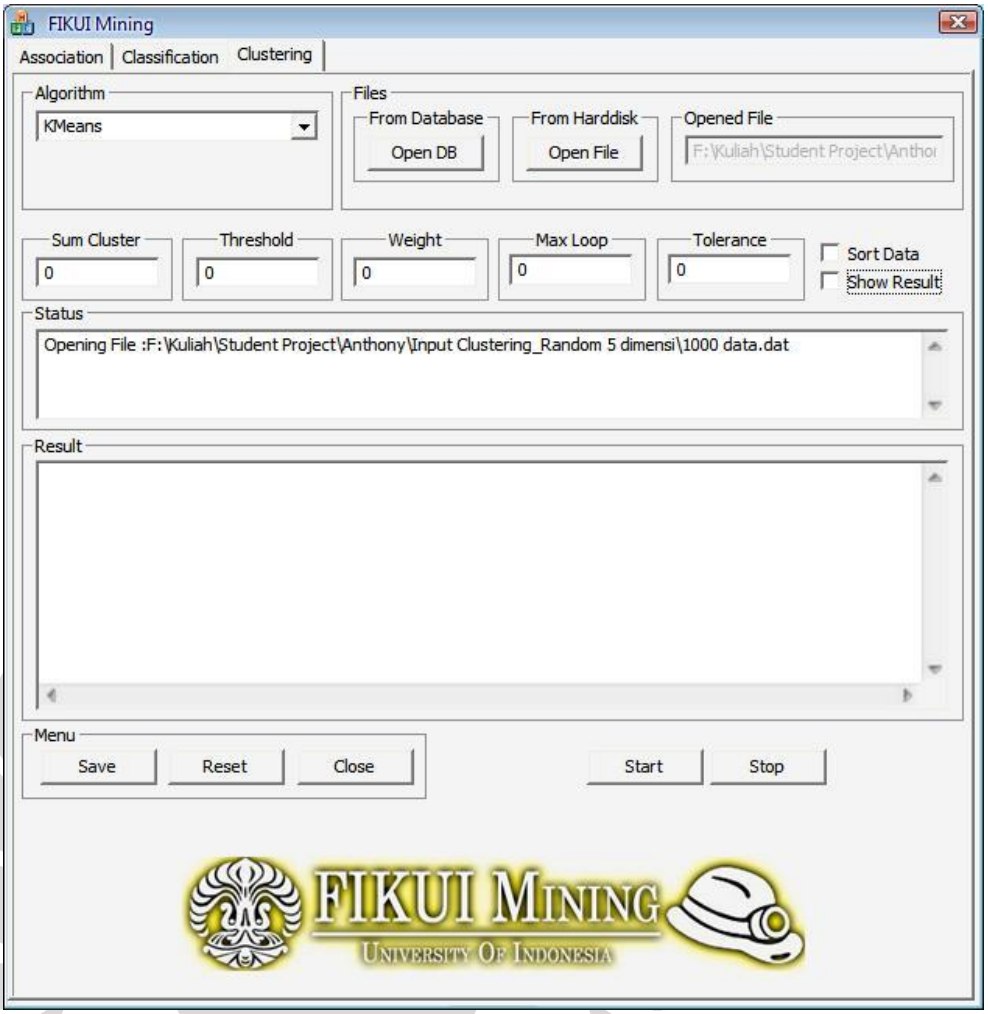
4.2.3 Modul *Clustering*



Gambar 4.9: Modul *Clustering* - Tampilan Awal

Gambar 4.9 adalah tampilan awal sistem FIKUI Mining untuk modul *clustering*. *User* dapat memilih algoritma-algoritma *clustering* dengan memilih *combo box* "Algorithm". Kemudian, *user* dapat memilih file masukan yang akan digunakan dengan menekan tombol "Open File". Apabila algoritma dan file masukan sudah terpilih, maka sistem akan menampilkan keterangan seperti gambar 4.10.

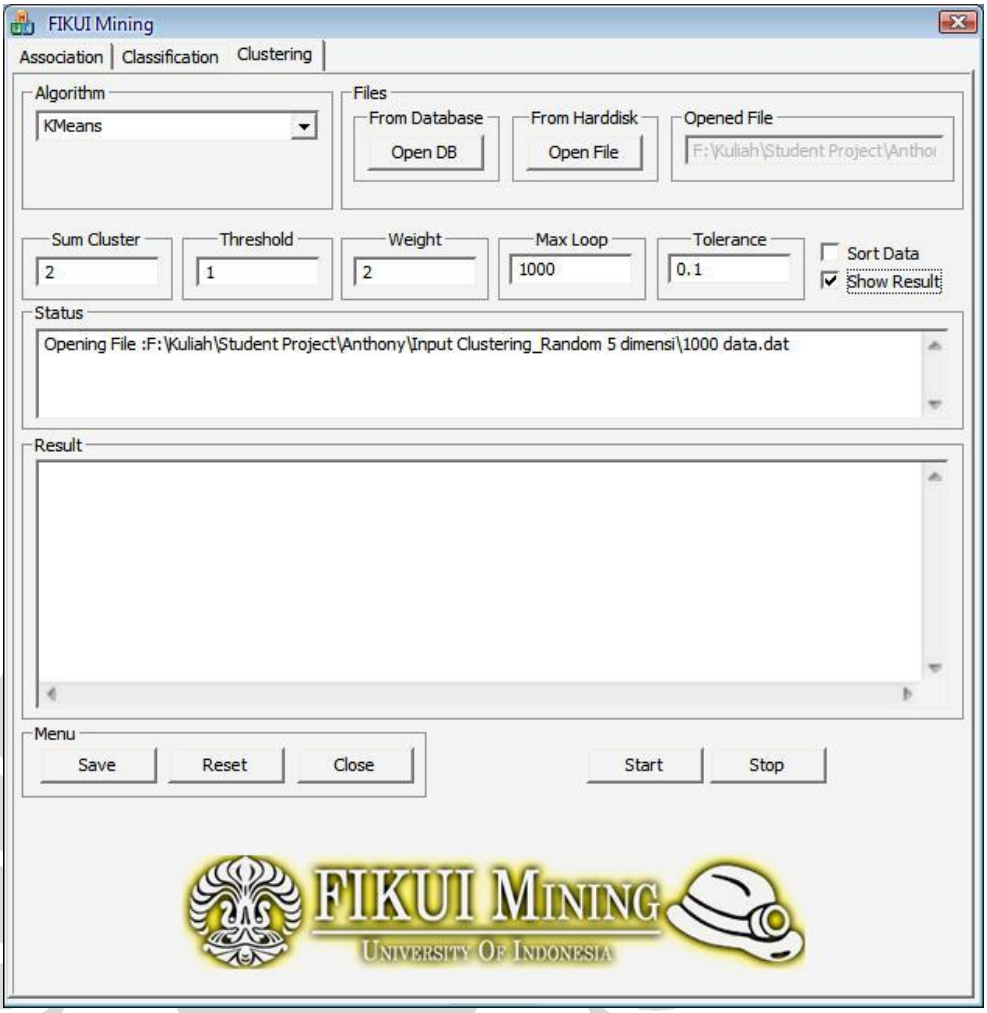
Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008



Gambar 4.10: Modul *Clustering* - Memilih Algoritma dan File Input

Pada gambar 4.10 di atas, dapat kita lihat *Combo box* "Algorithm" sudah terisi dengan algoritma yang dipilih oleh *user*. Begitu pula dengan *textarea* "Opened File", sudah terisi dengan file masukan yang dipilih. *User* dapat memilih nilai *sum cluster, threshold, weight, max loop,* dan *tolerance* yang dikehendaki dengan mengisi *textarea* "Sum Cluster", "Threshold", "Weight", "Max Loop", dan "Tolerance". Gambar 4.11 menunjukkan *user* sudah mengisi nilai-nilai tersebut.

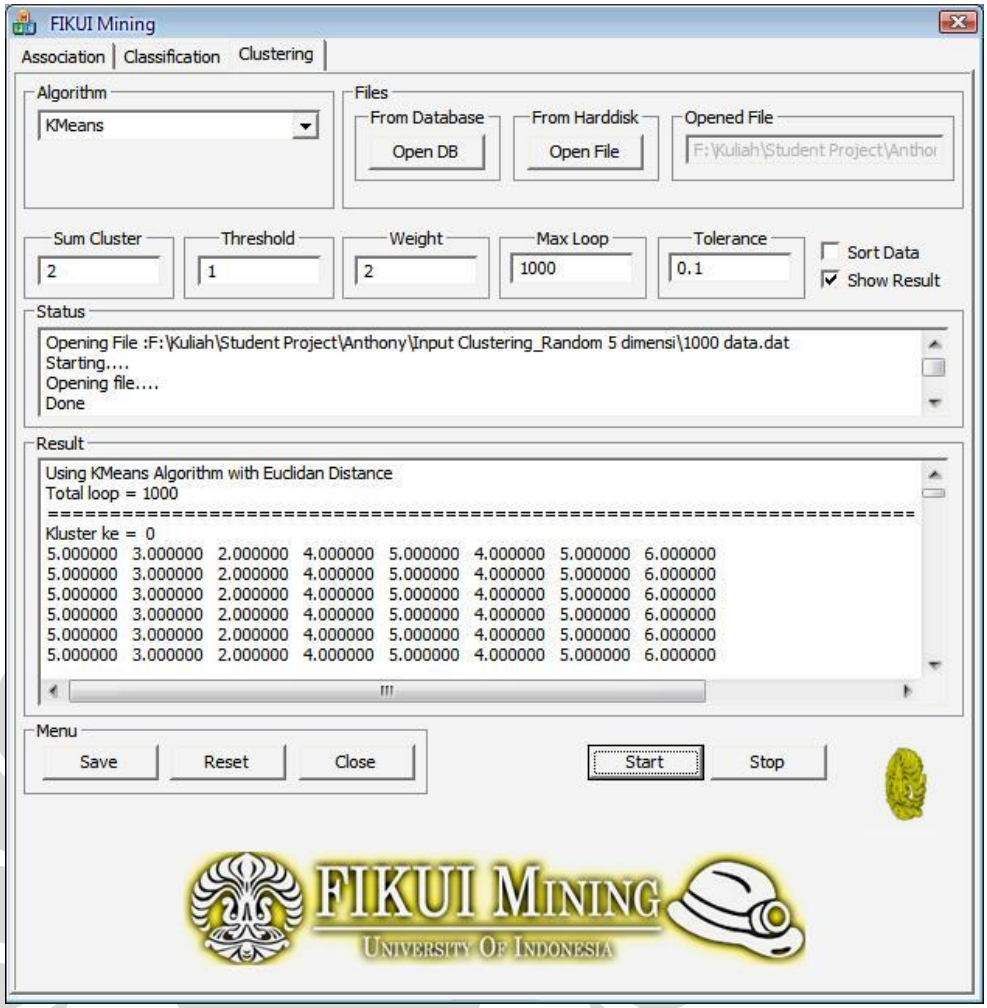
Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008



Gambar 4.11: Modul *Clustering* – Mengisi Nilai *Support, Max Size, dan Confidence*

Jika *user* telah memilih algoritma, memilih file masukan, dan mengisi nilai-nilai *sum cluster, threshold, weight, max loop,* dan *tolerance,* maka *user* dapat menekan tombol "Start" untuk mulai menjalankan algoritma tersebut. Hasilnya akan ditampilkan seperti gambar 4.12.

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008



Gambar 4.12: Modul *Custering* – Hasil Eksekusi Algoritma

User dapat menyimpan hasil yang ditampilkan pada sistem, dengan cara menekan tombol "Save". Hasil tersebut akan disimpan ke dalam sebuah file berekstensi *txt*. Jika *user* ingin membersihkan tampilan sistem dari hasil eksekusi algoritma sebelumnya, maka *user* dapat menekan tombol "Reset". Hal ini menyebabkan tampilan sistem kembali ke tampilan awal seperti pada gambar 4.9.

4.3 Peningkatan Unjuk Kerja Algoritma

Pada bagian ini, kami akan menjelaskan mengenai peningkatan unjuk kerja algoritma pada sistem FIKUI Mining yang kami kembangkan, dibandingkan dengan sistem FIKUI Mining yang sudah ada.

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

4.3.1 Modul *Association*

Pada saat konversi dari Microsoft Visual Studio 6 ke Microsoft Visual Studio 8, ada beberapa kode program yang tidak berjalan seperti yang diharapkan. Ada parameter dari sebuah *loop* yang menghasilkan sebuah *vector subcript out of range error.* Kesalahan ini muncul karena ada kode program yang mengakses sebuah elemen kosong pada sebuah vektor. Parameter *loop* ini dapat di atasi. Ketika diuji coba dengan input data *chess*, algoritmapun sudah berjalan dengan baik, meskipun secara kecepatan masih lebih cepat pada saat algoritma ini belum diintegrasikan dengan FIKUI Mining.

Perbaikan lainnya ada pada pemakaian tipe data yang memungkinkan terjadinya kehilangan presisi dari data yang diolah. Contohnya adalah pada saat mengubah data dari tipe data `double` ke dalam tipe data `float`. Pada FIKUI Mining ini, kehilangan presisi data sudah di atasi dengan cara memberikan tipe data `double` jika memang data yang diolah membutuhkan perhitungan yang menghasilkan angka yang tidak bulat, dan memberikan tipe data `int` jika data yang diolah maupun yang ingin dihasilkan merupakan sebuah bilangan bulat.

4.3.2 Modul *Classification*

Perbaikan yang telah kami lakukan pada modul *classification* adalah sebagai berikut.

a. Perbaikan atas kesalahan pada code algoritma *CMAR* dan *CSFP*.

Kami menemukan kesalahan pada *code* dari algoritma *CMAR* dan *CSFP* pada sistem sebelumnya. Kesalahan terjadi ketika algoritma tersebut dijalankan dua kali atau lebih. Hasil pengujian yang pertama dan yang kedua akan berbeda. Hal ini disebabkan karena pada saat awal dijalankannya algoritma, ada beberapa variabel yang belum di set nilainya menjadi nol. Akibatnya ketika algoritma dijalankan lebih dari satu kali, nilai di dalam variabel masih menggunakan nilai yang lama. Hal inilah yang menyebabkan hasil pengujian pertama dan kedua akan

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

berbeda. Kami telah memperbaiki *code* kedua algoritma pada sistem yang kami kembangkan sehingga kesalahan tersebut sudah teratasi.

b. Optimalisasi penggunaan memori pada algoritma *CMAR*

Pada *code* algoritma *CMAR*, kami menemukan beberapa variabel yang sudah dideklarasikan, tetapi tidak terpakai. Hal ini tentu saja mempengaruhi efisiensi penggunaan memori. Oleh karena itu, kami membuang variabel-variabel yang sudah dideklarasikan, tetapi tidak terpakai tersebut.

Pada code algoritma *CMAR*, terdapat fungsi `print_patterns()` dan `count_contigency()`. Kami menemukan penggunaan variabel yang tidak perlu pada kedua fungsi tersebut. Terdapat variabel `print` bertipe data `String`, yang digunakan untuk menampung *output* dari sistem. Namun, variabel `print` tersebut ternyata digunakan hanya sebagai variabel *temporary* saja. Nilai dari variabel `print` tersebut akan kembali dilempar ke dalam variabel `fileOutput`*`CMAR`*. Variabel `fileOutput`*`CMAR`* adalah variabel yang menampung semua hasil eksekusi algoritma. Seharusnya output dari sistem tadi tidak perlu ditampung dulu pada variabel `print`, namun dapat langsung dimasukkan ke dalam variabel `fileOutput`*`CMAR`*.

Selain itu pada beberapa bagian *code*, kami menemukan pemanggilan fungsi `String::empty()`. Kami menduga pemanggilan fungsi ini bertujuan untuk mengosongkan suatu variabel bertipe data string. Namun, setelah kami melakukan pengecekan, ternyata fungsi tersebut adalah fungsi untuk mengetahui apakah suatu string kosong atau tidak. Oleh karena itu, kami menghapus pemanggilan fungsi ini.

4.3.3 Modul *Clustering*

Pada beberapa bagian *code* algoritma di modul *clustering* ini, terdapat kesalahan dalam pendeklarasian variabel. Kesalahan ini berupa pendeklarasian variabel yang diulang-ulang. Pendeklarasian variabel seharusnya cukup sekali saja, tidak perlu dilakukan berulang kali. Hal ini tentu akan mempengaruhi efisiensi unjuk kerja algoritma. Kami telah

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

memperbaiki hal tersebut, dengan cara menghapus pendeklarasian variabel yang tidak perlu. Ada juga beberapa variabel yang dibuat atau dideklarasikan tapi tidak pernah dipergunakan. Untuk mengefisienkan memori, kami menghapus variabel-variabel yang tidak digunakan tersebut. Dalam *code* program sebelumnya juga terdapat konversi data yang memungkinkan hilangnya presisi data, sebagai contohnya adalah konversi data dari `float` ke dalam tipe data `double`. Hal ini kami atasi dengan cara mengganti tipe data yang memungkinkan hilangnya presisi data tersebut dengan cara merubah tipe data tersebut menjadi tipe data `double`. Dengan menggunakan Visual Studio 2005, kami dapat mendeteksi adanya *method* yang harus dispesifikan tipe *return*-nya. Sebagai contoh jika terdapat sebuah method yang menggunakan `.size()`, kami menambahkan dengan kode `(int)` didepan method tersebut agar data *return* menjadi valid.

Untuk uji coba algoritma pada modul *clustering* ini, kami juga sudah berhasil melakukan pengujian dengan *dataset* berukuran 500.000 baris data.

4.3.4 Keseluruhan Sistem

Hal yang cukup mempengaruhi unjuk kerja keseluruhan sistem FIKUI Mining sebelumnya adalah penggunaan fungsi `FormatMessage()`. Fungsi ini merupakan fungsi yang disediakan oleh C++ untuk mengatur *alignment* suatu teks sehingga output yang dihasilkan ke layar menjadi rapi. Kelemahan dari fungsi ini adalah penggunaan memori yang cukup banyak. Kami melakukan penghapusan penggunaan fungsi ini di ketiga modul tersebut. Sebagai solusi, kami merapikan secara manual output yang akan ditampilkan di layar menggunakan "\r\n". Hal ini akan lebih menghemat penggunaan memori komputer, sekaligus meningkatkan unjuk kerja sistem FIKUI Mining yang baru.

Selain itu, kami juga berusaha meningkatkan unjuk kerja sistem FIKUI Mining yang baru dengan cara lebih menghemat pemakaian memori komputer. Optimalisasi ini kami lakukan dengan cara mengkosongkan memori ketika penggunaannya sudah selesai. Fungsi yang menangani hal ini adalah fungsi *free()*, yang sudah disediakan oleh C++.

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

BAB 5
UJI COBA DAN ANALISIS KELUARAN

Bab ini akan membahas mengenai hasil uji coba dan analisis keluaran dari 8 algoritma yang diimplementasikan pada sistem FIKUI Mining. Pengujian dilakukan dengan menggunakan berbagai macam *input* data yang disesuaikan dengan algoritma yang di ujikan. Untuk tiap metode, dimasukan atribut-atribut yang berbeda sesuai dengan algoritma yang akan di uji coba.

5.1 Pengujian Metode *Association*

Pengujian metode *Association* menggunakan data *chess* yang mempunyai 3196 data dimana tiap datanya terdiri dari 37 item. Setiap algoritma di dalam metode *Assosciation,*yaitu *Apriori, CT-Pro* dan *FP-Growth*, ketiganya diuji coba dengan menggunakan *input* data *chess* tersebut.

5.1.1 Pengujian Algoritma *Apriori*

Berikut adalah tabel hasil pemrosesan data *chess* yang diproses dengan menggunakan algoritma *Apriori* pada FIKUI Mining. *Confidence* yang digunakan sebesar 90.

Tabel 5.1: Pengujian Algoritma *Apriori* pada FIKUI Mining

Support	99	95	90	85	80
Read Item	0.2	0.13	0.09	0.11	0.09
Frequent Itemset Mining	0.09	0.5	1.2	3.89	14.5
Generate Rules	0	0	0.09	0.88	6.31
Total Time	0.29	0.63	1.38	4.88	20.9

Berikut adalah tabel hasil pemrosesan data *chess* dengan menggunakan algoritma *Apriori* pada pengembangan yang dilakukan sebelumnya yang belum tergabung dalam FIKUI Mining [FAH06].

Tabel 5.2 : Pengujian Algoritma *Apriori* versi Fahrian.

Support	99	95	90	85	80
Read Items	0.29	0.18	0.08	0.08	0.11
Frequent Itemset Mining	0.36	0.69	1.03	3.6	10.8
Generate Rules	0	0.01	0.06	1.01	35.3
Total	0.65	0.88	1.17	4.69	46.2

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

5.1.2 Pengujian Algoritma *CT-Pro*

Berikut adalah tabel hasil pemrosesan data *chess* yang diproses dengan menggunakan algoritma *CT-Pro* pada FIKUI Mining.

Tabel 5.3: Pengujian Algoritma *CT-Pro* pada FIKUI Mining

Support	99	95	90	85	80
Read Item	0.06	0.06	0.08	0.08	0.08
Construct Tree	0.08	0.08	0.06	0.08	0.08
Frequent Itemset Mining	0	0	0.01	0.08	0.27
Generate Rules	0	0	0.13	1.53	12.1
Total Time	0.14	0.14	0.28	1.77	12.6

Berikut adalah tabel hasil pemrosesan data *chess* dengan menggunakan algoritma *CT-Pro* pada pengembangan yang dilakukan sebelumnya yang belum tergabung dalam FIKUI Mining [FAH06].

Tabel 5.4: Pengujian Algoritma *CT-Pro* versi Fahrian

Support	99	95	90	85	80
Read Item	0.06	0.06	0.05	0.05	0.06
Construct Tree	0.06	0.05	0.06	0.06	0.08
Frequent Itemset Mining	0	0	0.02	0.06	0.42
Generate Rules	0	0.02	0.06	1.25	33.7
Total Time	0.12	0.13	0.19	1.42	34.3

5.1.3 Pengujian Algoritma *FP-Growth*

Berikut adalah tabel hasil pemrosesan data *chess* yang diproses dengan menggunakan algoritma *FP-Growth* pada FIKUI Mining. *Confidence* yang digunakan sebesar 90.

Tabel 5.5: Pengujian Algoritma *FP-Growth*

Support	99	95	90	85	80
Read and Sort Items	0.09	0.09	0.11	0.09	0.09
Build FP-Tree	0.16	0.19	0.17	0.14	0.17
Frequent Itemset Mining	0	0.02	0.11	1.14	9.92
Generate Rules	0	0	0.11	0.94	6.61
Total Time	0.34	0.39	0.61	2.4	16.9

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

Berikut adalah tabel hasil pemrosesan data *chess* dengan menggunakan algoritma *FP-Growth* pada pengembangan yang dilakukan sebelumnya yang belum tergabung dalam FIKUI Mining [FAH06].

Tabel 5.6: Pengujian Algoritma *FP-Growth* versi Fahrian

Support	99	95	90	85	80
Read and Sort Items	0.05	0.06	0.06	0.06	0.06
Build FP-Tree	0.08	0.08	0.08	0.09	0.13
Frequent Itemset Mining	0	0	0.05	0.58	5.71
Generate Rules	0.02	0	0.06	1.03	32.9
Total Time	0.2	0.2	0.31	1.82	38.9

Dari tabel 5.1, 5.2, 5.3, 5.4, 5.5 dan 5.6 dapat diambil kesimpulan bahwa semakin kecil nilai *support* yang diberikan, maka jalannya algoritma akan semakin lambat. Hal ini dikarenakan semakin banyaknya hubungan antardata yang akan diproses.

5.2 Pengujian Metode *Classification*

Pengujian metode *classification* menggunakan 3 *dataset* yang berbeda. Yang dimaksud dengan *dataset* di sini adalah suatu file yang berisi data masukan dengan jumlah data tertentu. Ketiga *dataset* tersebut adalah *dataset* golf yang berisi 14 data, *dataset* vote yang berisi 300 data, dan *dataset* waveformd yang berisi 5000 data. Setiap algoritma di dalam metode *classification* diuji dengan ketiga *dataset* tersebut.

5.2.1 Pengujian Algoritma *CMAR*

Pengujian pada algoritma *CMAR* menggunakan *dataset* golf dan *dataset* vote dengan nilai *support* 50%, 30%, dan 10%. Begitu juga dengan nilai *confidence* dan nilai *coverage* yang digunakan, menggunakan nilai 50%, 30%, dan 10%. Tabel 5.7 menunjukkan hasil pengujian tersebut.

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

Tabel 5.7: Hasil Pengujian Algoritma *CMAR*

Dataset	Support (%)	Confidence (%)	*Coverage* (%)	Jumlah Rules Yang Dihasilkan	Waktu (s)
Golf	50	50	50	4	0.91
			40	4	1.09
			30	4	1.28
		30	50	4	0.86
			40	4	0.97
			30	4	1.41
		10	50	4	1.03
			40	4	1.22
			30	4	0.92
	30	50	50	28	0.72
			40	28	0.72
			30	28	0.67
		30	50	28	0.75
			40	28	0.75
			30	28	0.77
		10	50	28	0.7
			40	28	0.72
			30	28	0.77
	10	50	50	43	1.17
			40	43	1.06
			30	43	1
		30	50	43	0.92
			40	43	0.69
			30	43	0.72
		10	50	43	0.92
			40	43	0.86
			30	43	0.66
Vote	50	50	50	452	2.92
			40	452	3.03
			30	452	3.3
		30	50	452	3.19
			40	452	3.02
			30	452	2.98
		10	50	452	3.33
			40	452	3.16
			30	452	3.78

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

Pengujian algoritma *CMAR* hanya menggunakan *dataset* golf dan *dataset* vote saja. Hal ini dikarenakan algoritma *CMAR* kurang cocok jika digunakan untuk pemrosesan data berukuran besar.

5.2.2 Pengujian Algoritma *CSFP*

Pengujian algoritma *CSFP* menggunakan ketiga *dataset* yang sudah disebutkan sebelumnya. Nilai *support, confidence,* dan *coverage* yang digunakan juga sama, yaitu 50%, 30%, dan 10%. Tabel sekian menunjukkan hasil pengujian terhadap algoritma *CSFP*.

Tabel 5.8: Pengujian Algoritma *CSFP*

Dataset	Support (%)	Confidence (%)	*Coverage* (%)	Jumlah Rules Yang Dihasilkan	Waktu (s)
Golf	50	50	50	5	0.92
			40	5	0.55
			30	5	0.59
		30	50	5	0.66
			40	5	0.61
			30	5	0.81
		10	50	5	0.56
			40	5	0.55
			30	5	0.55
	30	50	50	9	0.58
			40	9	0.58
			30	9	0.63
		30	50	10	1.23
			40	10	0.64
			30	10	0.61
		10	50	10	0.55
			40	10	1.95
			30	10	0.92
	10	50	50	8	0.73
			40	8	0.59
			30	8	0.75
		30	50	10	0.78
			40	10	0.86
			30	10	0.77
		10	50	10	0.5
			40	10	0.45
			30	10	0.58

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

Vote	50	50	50	27	1.01
			40	27	0.64
Vote	30		30	27	0.98
		30	50	27	0.67
			40	27	0.81
			30	27	0.8
		10	50	27	1.03
			40	27	0.77
			30	27	0.86
	30	50	50	31	0.78
			40	31	0.97
			30	31	0.67
		30	50	31	0.77
			40	31	0.69
			30	31	0.67
		10	50	31	0.94
			40	31	0.66
			30	31	0.67
	10	50	50	34	0.7
			40	34	0.97
			30	34	0.81
		30	50	34	0.73
			40	34	0.73
			30	34	0.88
		10	50	34	0.73
			40	34	0.61
			30	34	0.64
Waveformd	50	50	50	6	2.19
			40	6	1.83
			30	6	1.95
		30	50	6	2.25
			40	6	1.91
			30	6	1.78
		10	50	6	1.72
			40	6	1.78
			30	6	1.95
	30	50	50	37	3.78
			40	37	3.69
			30	37	3.58
		30	50	41	3.8

			40	41	3.36
			30	41	3.58
Waveformd		10	50	41	3.88
			40	41	4.11
			30	41	3.8
	10	50	50	103	5.94
			40	103	5.92
			30	103	6.05
		30	50	103	7.23
			40	103	6.91
			30	103	6.94
		10	50	103	6.88
			40	103	7.24
			30	103	7.05

Algoritma *CSFP* sangat cocok untuk menjalankan data yang berukuran besar. Hal ini dapat terlihat dari pengujian di atas. *Dataset* waveformd yang berjumlah 5000 data, dapat diselesaikan dengan baik menggunakan algoritma ini.

5.3 Pengujian Metode *Clustering*

Pada metode *clustering*, pengujian menggunakan sebuah dokumen *file* yang berisi sepuluh ribu data dengan dimensi data sebanyak 4. Data yang ada dibuat secara acak dengan menggunakan program kecil dibuat menghasilkan data secara acak sebanyak parameter yang dimasukan. Data-data tersebut mempunyai kisaran besar data dari nol sampai sembilan puluh sembilan. Berikut adalah kode program dengan menggunakan bahasa pemrograman Java untuk membuat data tersebut.

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

```
import java.io.*;
import java.util.*;
public class GenerateData
{
   public static int a;
   public static void main(String [] args)
   {
      PrintWriter pw=null;

      try
      {
pw=new PrintWriter(new
FileOutputStream("10000dataC.dat"))
        for (int i=1;i<10000;i++)
        {
           for(int j=0;j<3;j++)
           {
a=(int)Math.round(Math.random()*100);
           pw.print(a+",");
           }
           pw.print(a);
           pw.println();
        }
      }
      catch(Exception e)
      {
         System.out.println("error ");
      }
      return;
   }
}
```

5.3.1 Pengujian Algoritma *K-Means*

Pada pengujian algoritma *K-Means* berikut ini, menggunakan atribut sebesar 0,1 dan maksimum *looping* sebanyak 1000. Berikut adalah tabel hasil pengujian *K-Means* yang diimplementasikan pada FIKUI Mining.

Tabel 5.9: Pengujian Algoritma *K-Means* pada FIKUI Mining

Jumlah data	Jumlah kluster	Total waktu (ms)	Cluster kosong	Total waktu (sorted) (ms)	Kluster kosong (sorted)
10000	1000	2344	48	10656	0
10000	2000	4333	199	12772	1
10000	3000	6469	451	13406	2
10000	4000	8760	778	12688	6
10000	5000	10786	1175	16671	26

Pada pengujian algoritma ini, ikut di uji juga algoritma *simple K-Means* yang di miliki oleh WEKA. Dalam pengujian tersebut, dimasukan

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

atribut seed sebesar 10.Data yang digunakan dalam pengujian juga data yang diambil secara acak, yang dihasilkan oleh program kecil yang dicantumkan pada sub bab 5.3. Berikut adalah tabel hasil pengujian *simple K-Means* yang dimiliki oleh WEKA.

Tabel 5.10: Pengujian Algoritam *Simple K-Means* pada WEKA

Jumlah data	Jumlah kluster	Total waktu (ms)	Number of iteration
10000	1000	104267.5	16
10000	2000	130710	16
10000	3000	177215.5	12
10000	4000	240640	7.5
10000	5000	238634.5	6

5.3.2 Pengujian Algoritma *Nearest Neighbour*

Pada pengujian algoritma *Nearest Neighbour* berikut, menggunakan data yang sama seperti yang digunakan pada pengujian algoritma *K-Means.* Berikut adalah tabel hasil pengujian *Nearest Neighbour* yang diimplementasikan pada FIKUI Mining.

Tabel 5.11: Pengujian Algoritma *Nearest Neighbour* pada FIKUI Mining

Jumlah data	Threshold	Total waktu (ms)	Jumlah kluster	Total waktu (sorted) (ms)	Jumlah kluster (sorted)
10000	5	8124	2725	24095	1825
10000	10	4946	397	19049	2
10000	15	4652	116	18551	0
10000	20	4300	59	17186	0
10000	25	3539	32	15425	0

5.3.3 Pengujian Algoritma *Fuzzy C-Means*

Sama halnya dengan 2 algoritma sebelumnya, algoritma inipun diuji coba dengan menggunakan data yang sama, yaitu yang dihasilkan secara acak oleh program yang dicantumkan pada sub bab 5.1. Berikut adalah tabel hasil pengujian *Fuzzy C-Means* yang diimplementasikan pada FIKUI Mining.

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

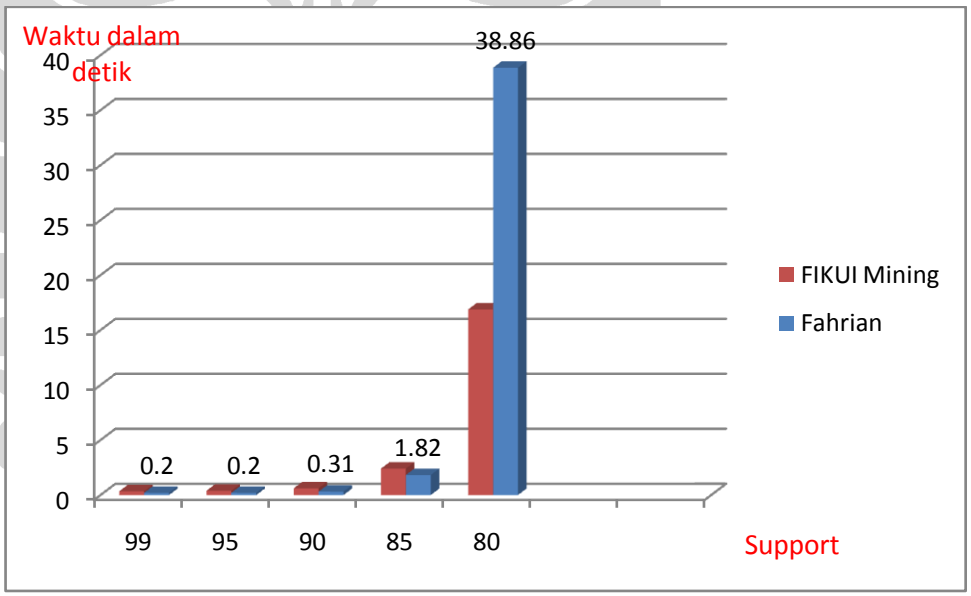
Tabel 5.12: Pengujian Algoritma *Fuzzy C-Means* pada FIKUI Mining

Jumlah data	Weight	Total waktu (ms)	Keanggotaan	Total waktu (sorted)(ms)	Keanggotaan (sorted)
10000	1.8	3824	0.462937	18989	0.462937
10000	1.9	2438	0.482088	19313	0.482088
10000	2	2318	0.5	19611	0.5
10000	2.1	3413	0.516779	20441	0.516779
10000	2.2	4501	0.532521	20641	0.532521

5.4 Analisis Metode *Association*

5.4.1 Analisis Algoritma *Apriori*

Berikut adalah gambar grafik dari lama pemrosesan data *chess* yang diolah dengan menggunakan algoritma *Apriori* yang diimplementasikan pada FIKUI Mining.

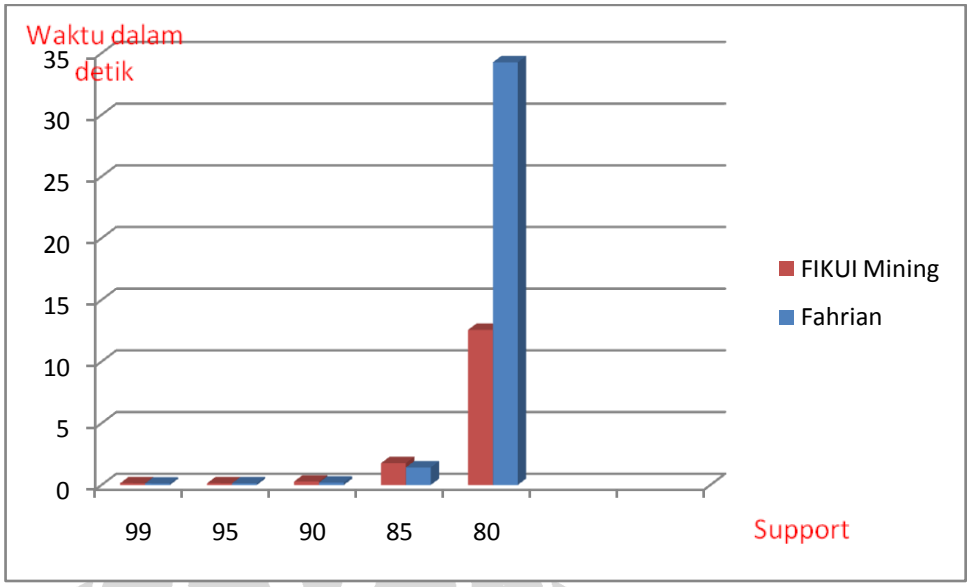


Gambar 5.1: Grafik Percobaan *Apriori*

5.4.2 Analisis Algoritma *CT-Pro*

Berikut adalah gambar grafik dari lama pemrosesan data *chess* yang diolah dengan menggunakan algoritma *CT-Pro* yang diimplementasikan pada FIKUI Mining.

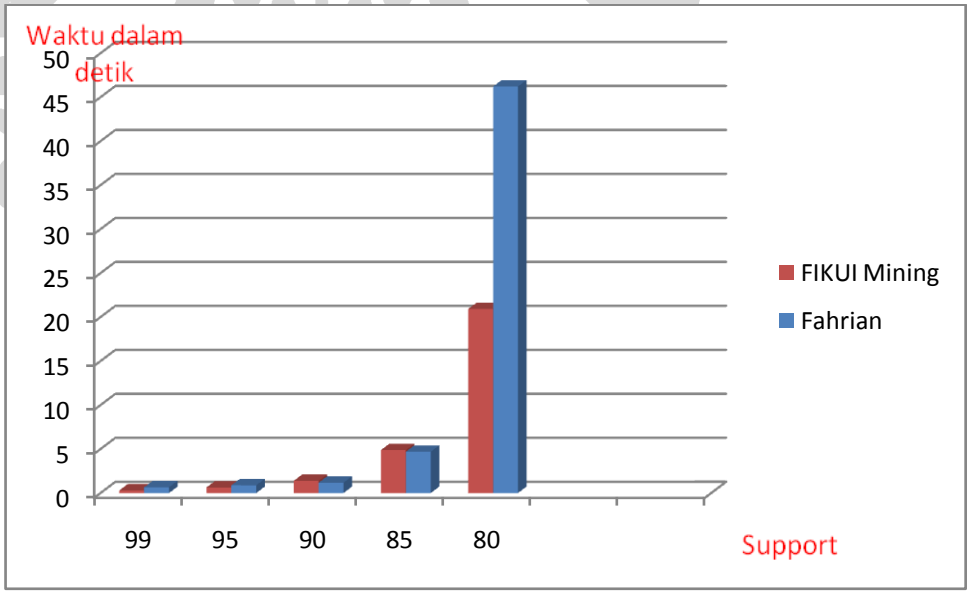
Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008



Gambar 5.2 : Grafik Percobaan *CT-Pro*

5.4.3 Analisis Algoritma *FP-Growth*

Berikut adalah gambar grafik dari lama pemrosesan data *chess* yang diolah dengan menggunakan algoritma *FP-Growth* yang diimplementasikan pada FIKUI Mining.



Gambar 5.3: Grafik Percobaan *FP-Growth*

Pada grafik 5.1, 5.2 dan 5.3 bisa dilihat bahwa nilai *support* sangatlah mempengaruhi lama pemrosesan dari data *chess*. Pada grafik bisa dilihat

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

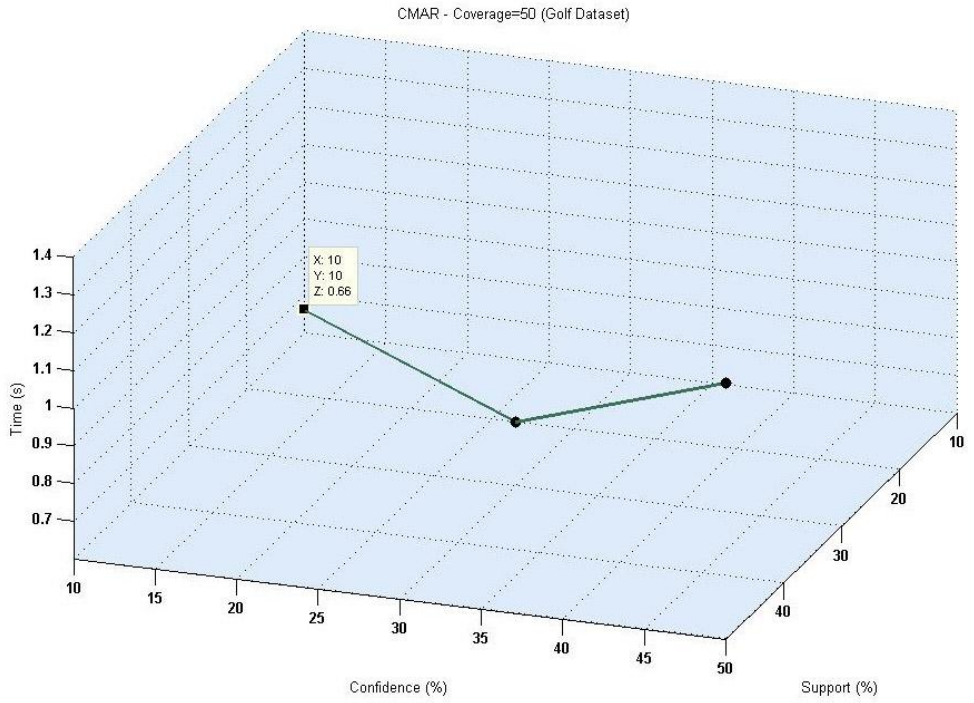
terjadi loncatan waktu pengerjaan yang cukup besar ketika *support* diturunkan dengan skala 5 dari angka 80 hingga 70. Hal ini menyatakan besarnya data yang diolah ketika *support* diturunkan dalam skala tersebut menjadi lebih besar secara signifikan.

5.5 Analisis Metode *Classification*

Pada sub bab ini akan dibahas mengenai analisis dari hasil percobaan masing-masing algoritma pada metode *classification*, yaitu *CMAR* dan *CSFP*. Analisis akan difokuskan bagaimana nilai *support*, *confidence*, dan *coverage* yang berbeda akan menghasilkan waktu yang berbeda untuk lama waktu pemrosesan data.

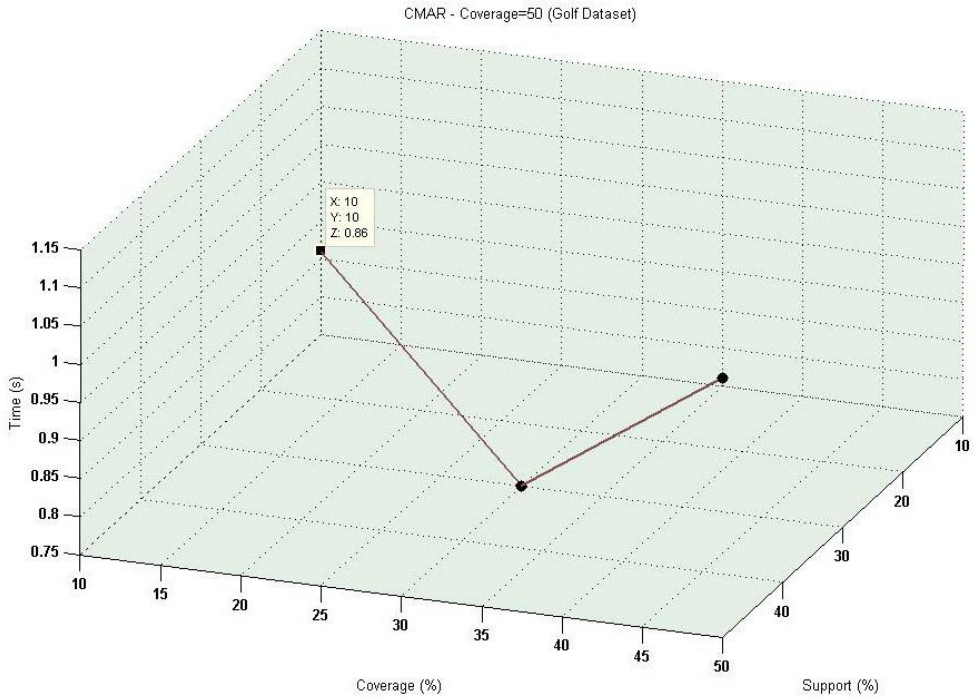
5.5.1 Analisis Algoritma *CMAR*

Berikut adalah grafik hasil pengujian algoritma *CMAR* dengan menggunakan *dataset* golf. Pada grafik 5.2, 5.3, dan 5.4, yang berubah adalah nilai *support* dan *confidence*, sedangkan nilai *coverage* tetap.

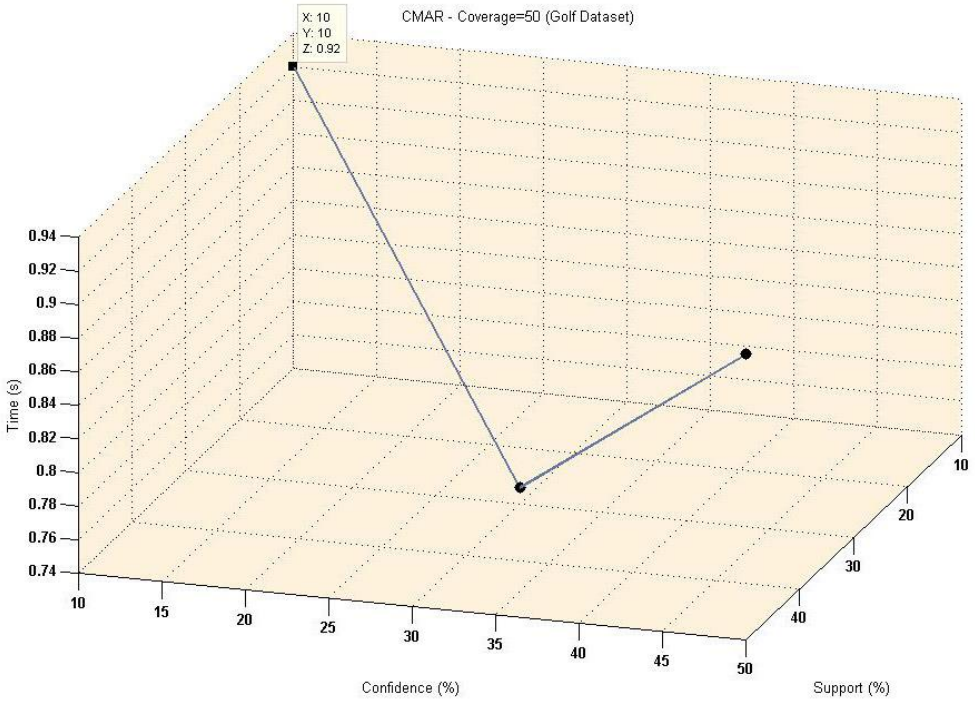


Gambar 5.4: Grafik Hasil Percobaan Algoritma *CMAR- Coverage* 30

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008



Gambar 5.5: Grafik Hasil Percobaan Algoritma *CMAR* - *Coverage* 40



Gambar 5.6: Grafik Hasil Percobaan Algoritma *CMAR* - *Coverage* 50

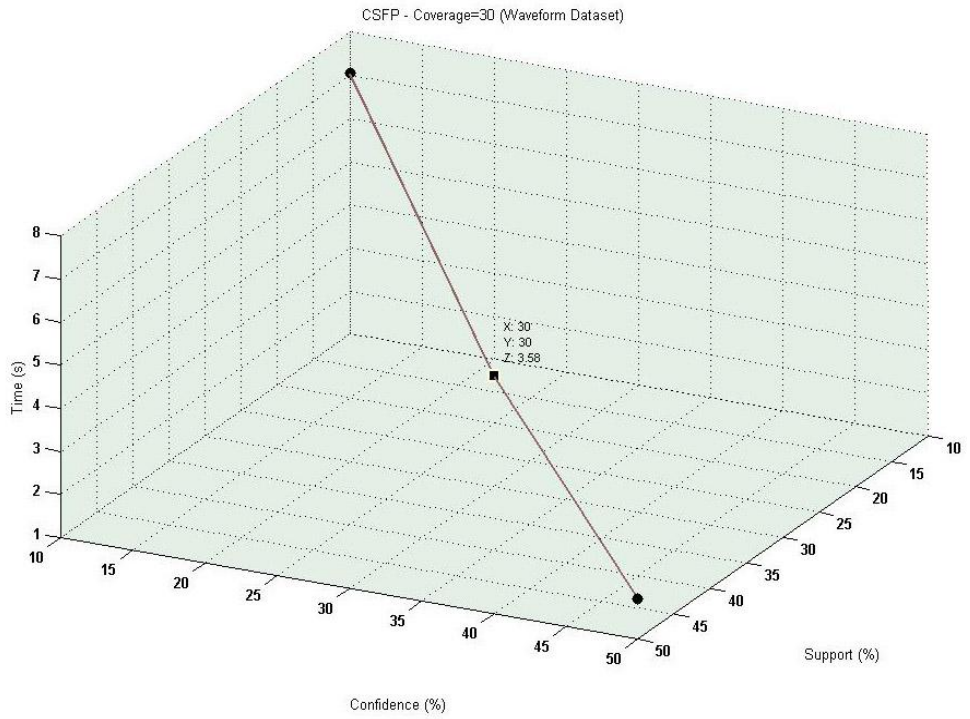
Dari grafik hasil percobaan di atas, secara umum dapat disimpulkan bahwa semakin kecil nilai *support*, *confidence* dan *coverage* yang

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

digunakan, maka akan semakin lama waktu yang dibutuhkan untuk pemrosesan data. Nilai *support* akan mempengaruhi persentase kombinasi item di dalam data. Semakin kecil nilai *support*, maka akan semakin banyak data yang terambil. Begitu juga nilai *confidence* yang akan mempengaruhi kuat atau tidaknya hubungan antar item dalam aturan assosiatif. Semakin banyak data yang terambil, maka akan mempengaruhi lama waktu yang dibutuhkan untuk pemrosesan data.

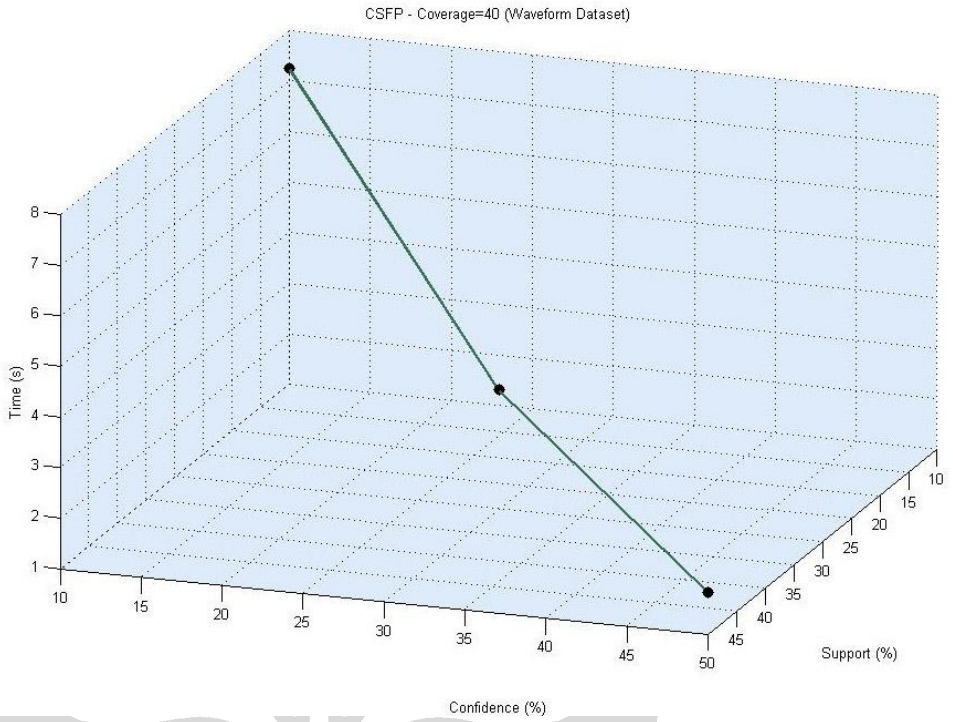
5.5.2 Analisis Algoritma *CSFP*

Berikut adalah grafik hasil pengujian algoritma *CSFP* dengan menggunakan *dataset* waveformd. Pada grafik 5.5, 5.6, dan 5.7, yang berubah adalah nilai *support* dan *confidence*, sedangkan nilai *coverage* tetap.

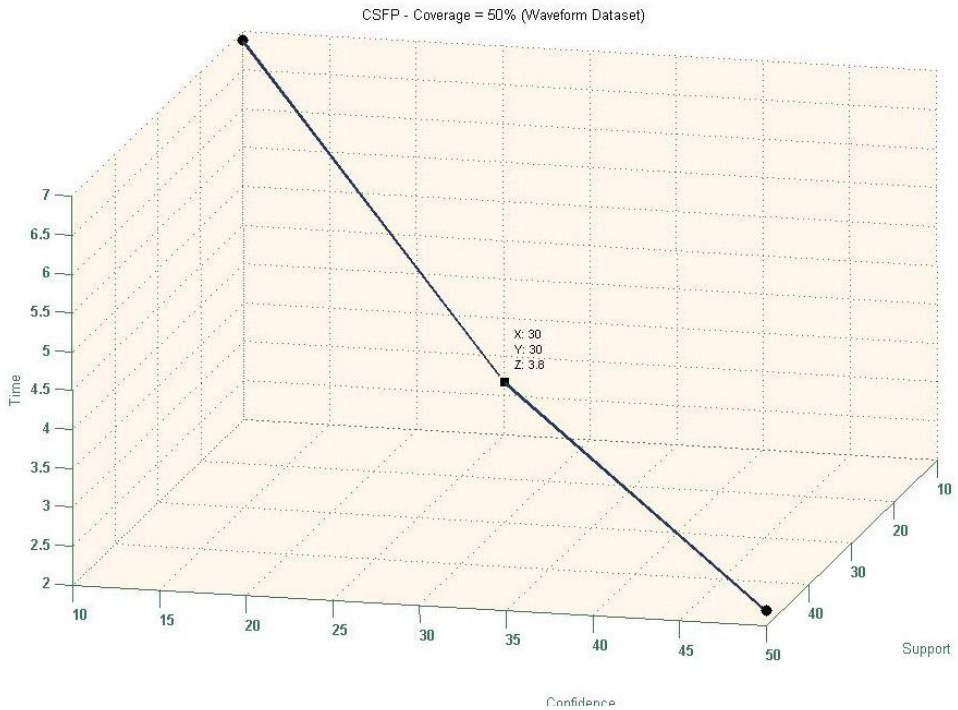


Gambar 5.7: Grafik Hasil Percobaan Algoritma *CSFP* - *Coverage* 30

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008



Gambar 5.8: Grafik Hasil Percobaan Algoritma *CSFP - Coverage* 40



Gambar 5.9: Grafik Hasil Percobaan Algoritma *CSFP - Coverage* 50

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

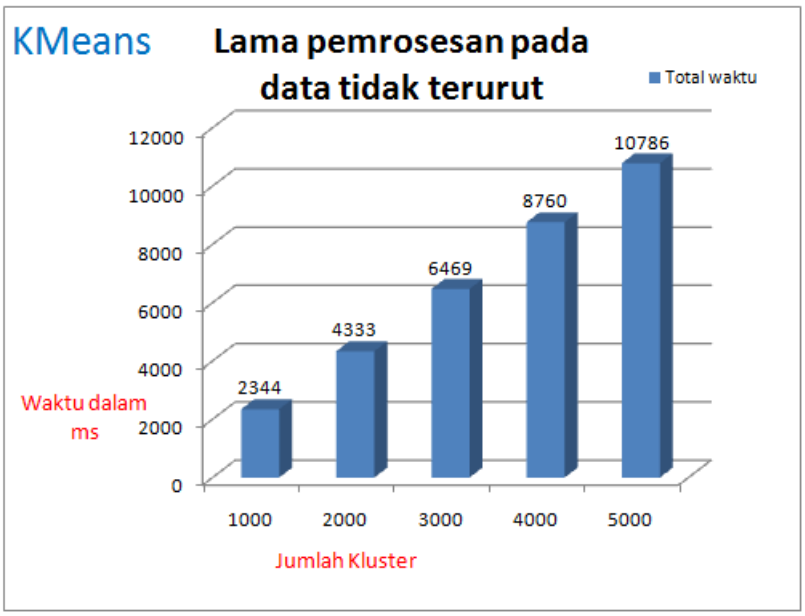
Dari ketiga grafik di atas, dapat kita simpulkan bahwa, semakin kecil nilai *support, confidence,* dan *coverage* yang digunakan, maka waktu pemrosesan data akan semakin bertambah. Hal ini disebabkan, semakin kecil *support, confidence,* dan *coverage* yang digunakan, maka kategori klasifikasi yang terbentuk akan semakin banyak, dan membutuhkan waktu pemrosesan yang lebih lama.

5.6 Analisis Metode *Clustering*

Pada sub bab ini akan dibahas mengenai hasil analisis dari masing-masing algoritma yang ada pada metode *clustering* yaitu *K-Means*, *Nearest Neighbour*, dan *Fuzzy C-Means*. Untuk tiap algoritma akan dianalisis beberapa aspek yang unik satu dengan yang lainnya. Kesamaan aspek yang bisa dibahas adalah aspek waktu yang dibutuhkan untuk mengolah data yang dimasukan ke dalam FIKUI Mining. Penyajian hasil analisis akan disajikan dalam bentuk grafik agar lebih mudah untuk dibaca.

5.6.1 Analisis Algoritma *K-Means*

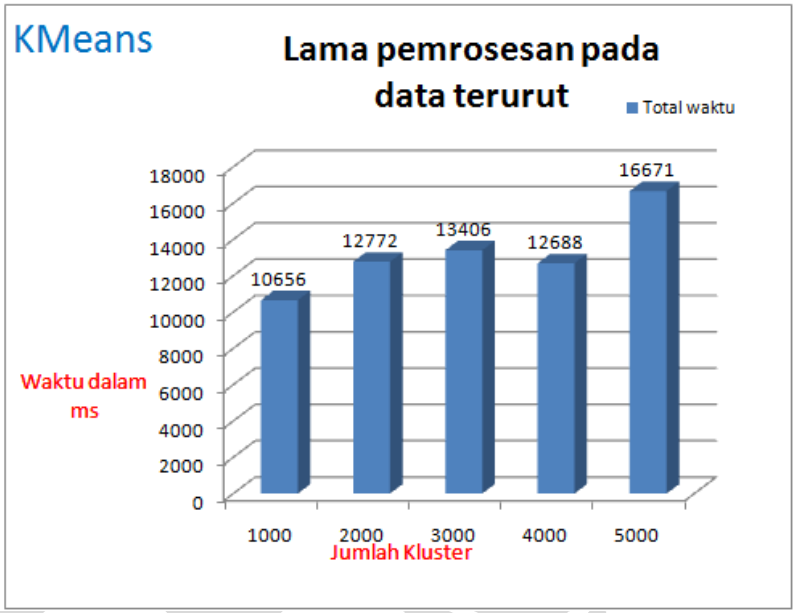
Berikut adalah grafik untuk hasil pengujian algoritma *K-Means* dengan data tidak terurut. Fokus pengamatan pada lama pemrosesan data.



Gambar 5.10: Grafik Pemrosessan Data Tidak Terurut Pada *K-Means*

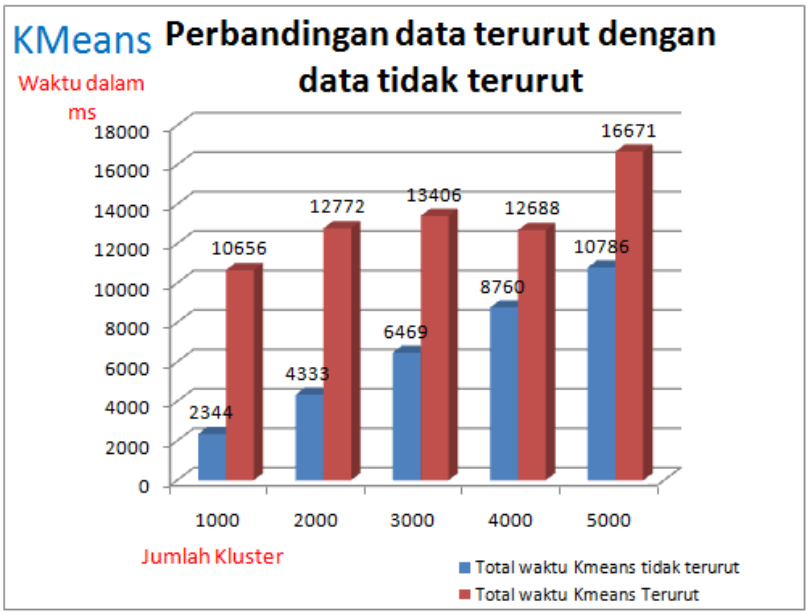
Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

Berikut adalah grafik untuk hasil pengujian algoritma *K-Means* dengan data terurut. Fokus pengamatan pada lama pemrosesan data.



Gambar 5.11: Grafik Pemrosesan Data Terurut Pada *K-Means*

Berikut adalah grafik perbandingan lama pemrosesan data terurut dan tidak terurut pada *K-Means.*

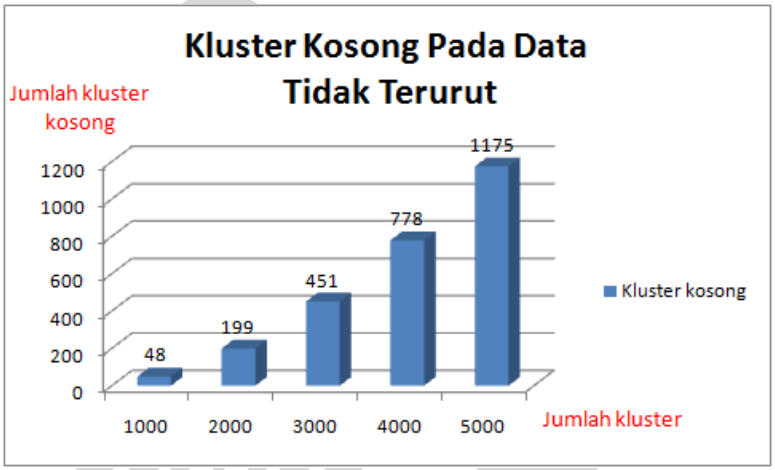


Gambar 5.12: Grafik Perbandingan Lama Pemrosesan Data Pada *K-Means*

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

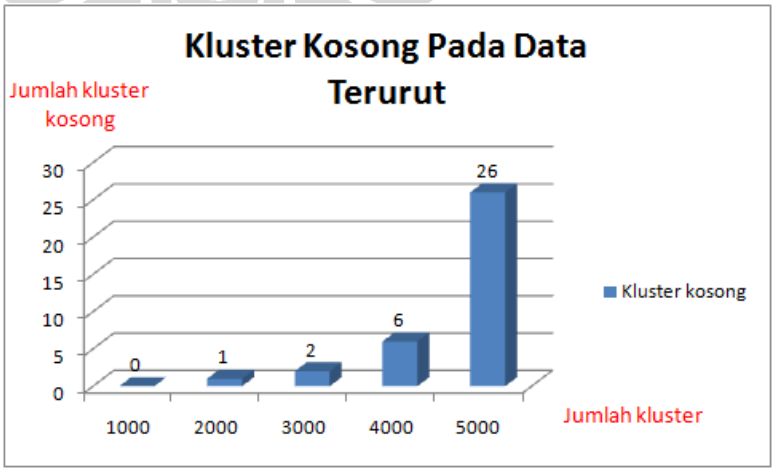
Banyaknya data dan kluster, mempengaruhi lamanya pemrosesan data pada FIKUI Mining. Semakin banyak kluster, maka akan semakin banyak waktu yang dibutuhkan untuk dapat memproses data. Pada data terurut, pemrosesan memakan waktu lebih lama dibandingkan dengan data yang tidak terurut. Hal ini disebabkan karena pada data terurut konvergensi kluster lebih lambat untuk didapatkan.

Berikut adalah grafik yang menggambarkan jumlah kluster kosong pada data tidak terurut.



Gambar 5.13: Grafik Jumlah Kluster Kosong Pada Data Tidak Terurut

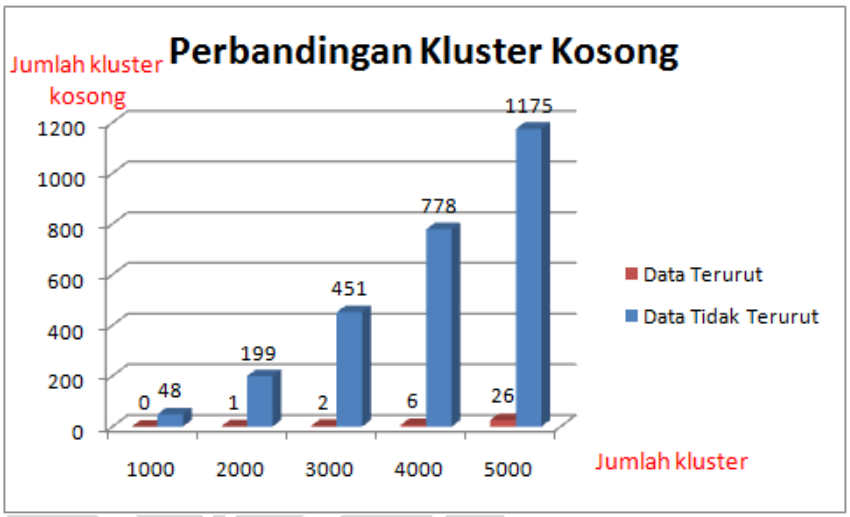
Berikut adalah grafik yang menggambarkan jumlah kluster kosong pada data terurut.



Gambar 5.14: Grafik Jumlah Kluster Kosong Pada Data Terurut

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

Berikut adalah grafik perbandingan jumlah kluster kosong pada data terurut dan tidak terurut pada *K-Means.*

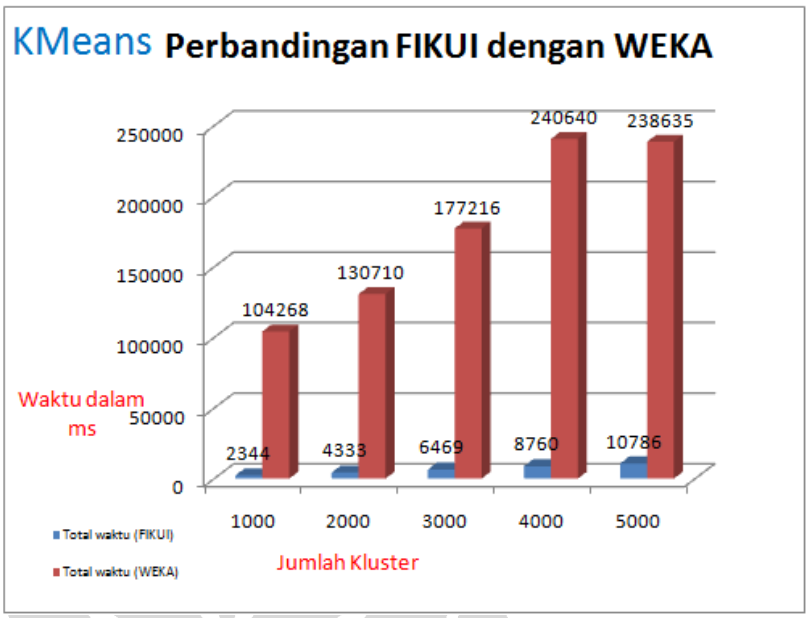


Gambar 5.15: Grafik Perbandingan Jumlah Kluster Kosong

Dari grafik di atas bisa dilihat bahwa kluster kosong yang terbentuk pada pemrosesan data yang tidak terurut lebih banyak dari pada kluster kosong pada pemrosesan data yang terurut. Hal ini dikarenakan persebaran data yang tidak tersebar dengan baik. Makin tersusun (tersebar) dengan baik suatu data, maka jumlah kluster kosong yang dihasilkanpun akan semakin sedikit jumlahnya. Bisa dilihat juga bahwa semakin banyak data, akan semakin membesar kemungkinan banyaknya kluster yang kosong.

Berikut adalah gambar grafik perbandingan antara *K-Means* yang berada pada FIKUI Mining dengan *Simple K-Means* yang terdapat pada WEKA

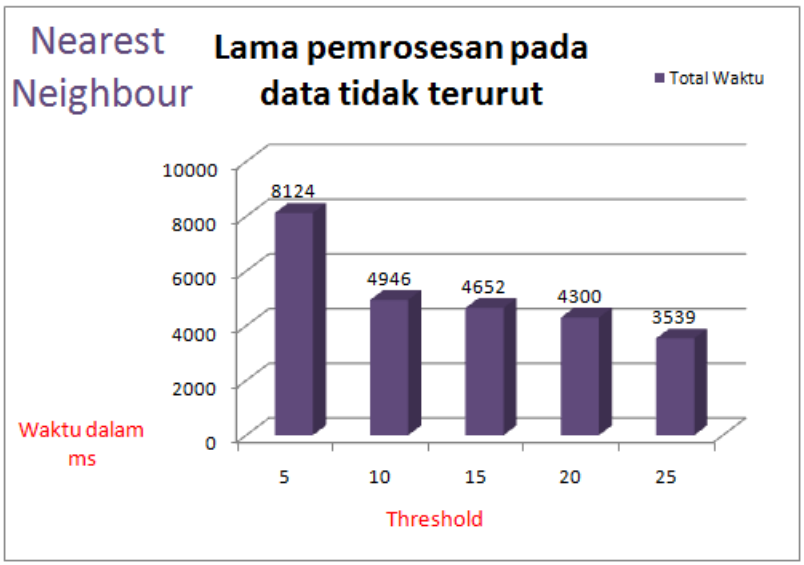
Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008



Gambar 5.16: Grafik Perbandingan *K-Means* FIKUI Mining dengan Simple *K-Means* pada WEKA

5.6.2 Analisis Algoritma *Nearest Neighbour*

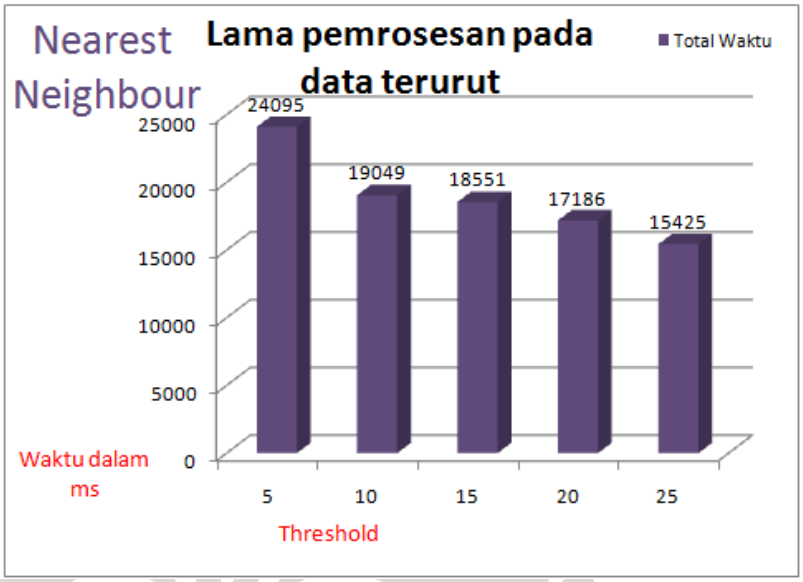
Berikut grafik tabel untuk hasil pengujian algoritma *Nearest Neighbour* dengan data tidak terurut. Fokus pengamatan pada lama pemrosesan data.



Gambar 5.17: Grafik Lama Pemrosesan Data Tidak Terurut pada *Nearest Neighbour*

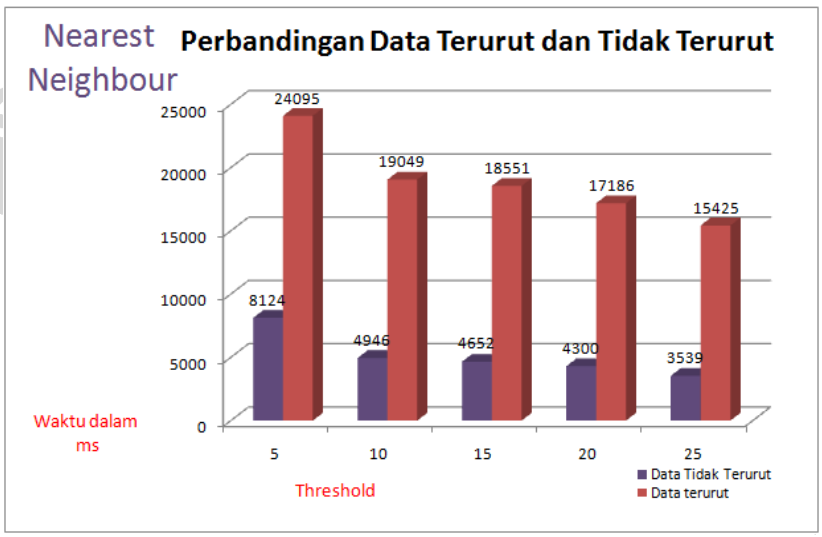
Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

Berikut adalah grafik untuk hasil pengujian algoritma *Nearest Neighbour* dengan data terurut. Fokus pengamatan pada lama pemrosesan data.



Gambar 5.18: Grafik Lama Pemrosesan Data Terurut pada *Nearest Neighbour*

Berikut adalah grafik perbandingan lama pemrosesan data terurut dan tidak terurut pada *Nearest Neighbour*.



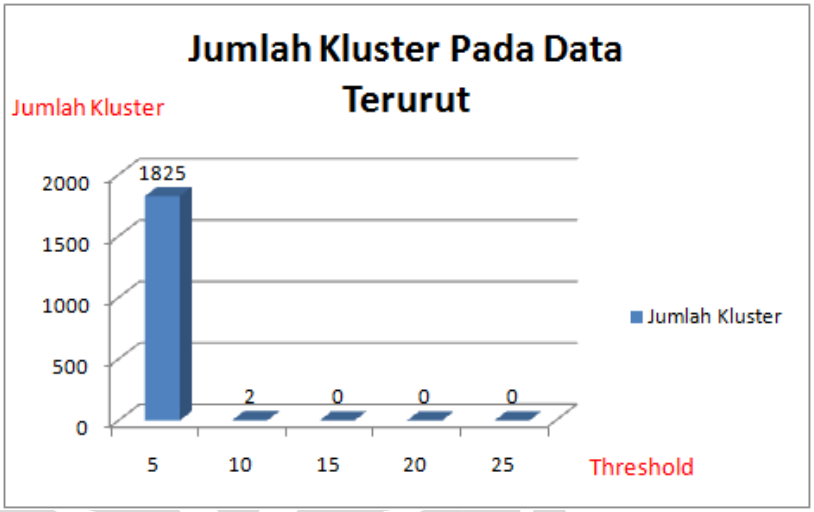
Gambar 5.19: Grafik Perbandingan Lama Pemrosesan Data pada *Nearest Neighbour*

Dari grafik tersebut dapat diketahui bahwa pemrosesan data yang tidak terurut membutuhkan waktu yang lebih sedikti dibandingkan pemrosesan pada data yang terurut. Semakin besar nilai dari *threshold* maka

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

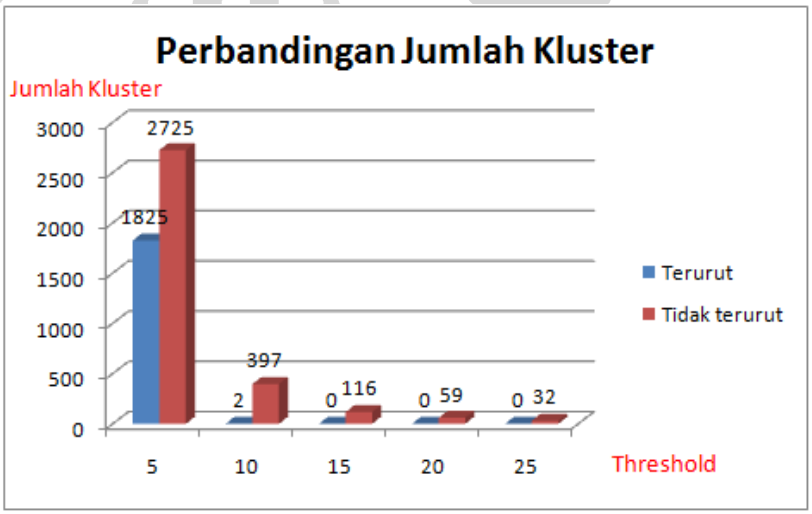
waktu pemrosesan data akan menjadi semakin singkat, dan begitupun sebaliknya.

Berikut adalah grafik jumlah kluster pada data yang tidak terurut.



Gambar 5.20: Grafik Jumlah Kluster pada Data Terurut

Berikut adalah grafik perbandingan antara jumlah kluster pada data terurut dan data tidak terurut yang dieksekusi pada FIKUI Mining.



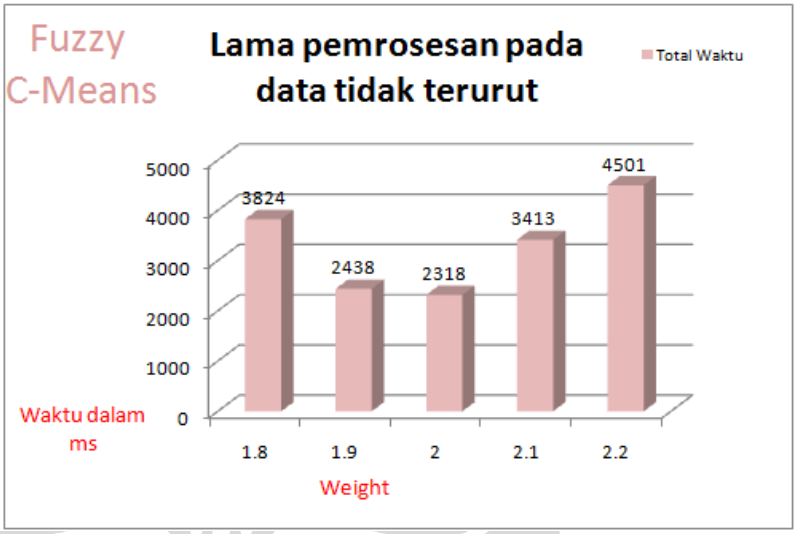
Gambar 5.21: Grafik Perbandingan Jumlah Kluster pada *Nearest Neighbour*

Jumlah kluster pada pemrosesan data tidak terurut lebih banyak daripada pemrosesan pada data terurut. Hal ini dikarenakan pada data terurut jarak data menjadi lebih rapat. Jumlah kluster yang terbentuk akan semakin sedikit apabila nilai dari *threshold* diperbesar.

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

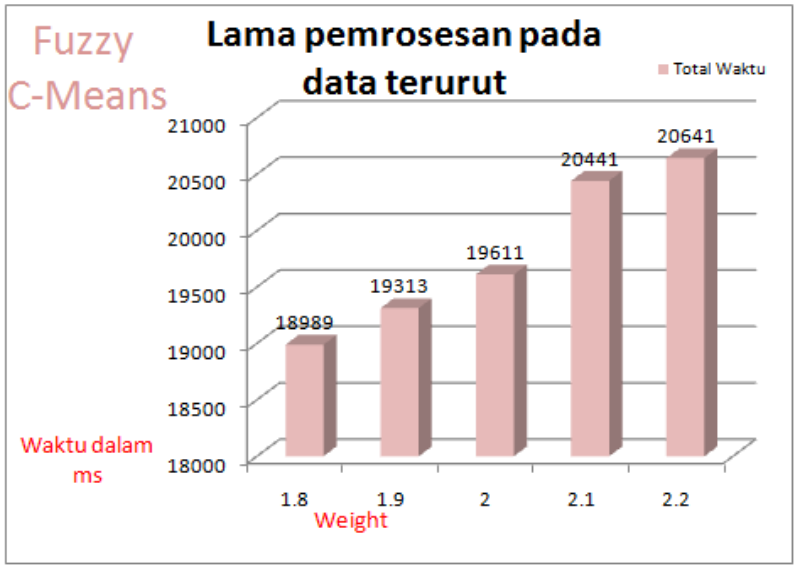
5.6.3 Analisis Algoritma *Fuzzy C-Means*

Berikut adalah tabel untuk hasil pengujian algoritma *Fuzzy C-Means* dengan data tidak terurut. Fokus pengamatan pada lama pemrosesan data.



Gambar 5.22: Grafik Lama Pemrosesan Data Tidak Terurut pada *Fuzzy C-Means*

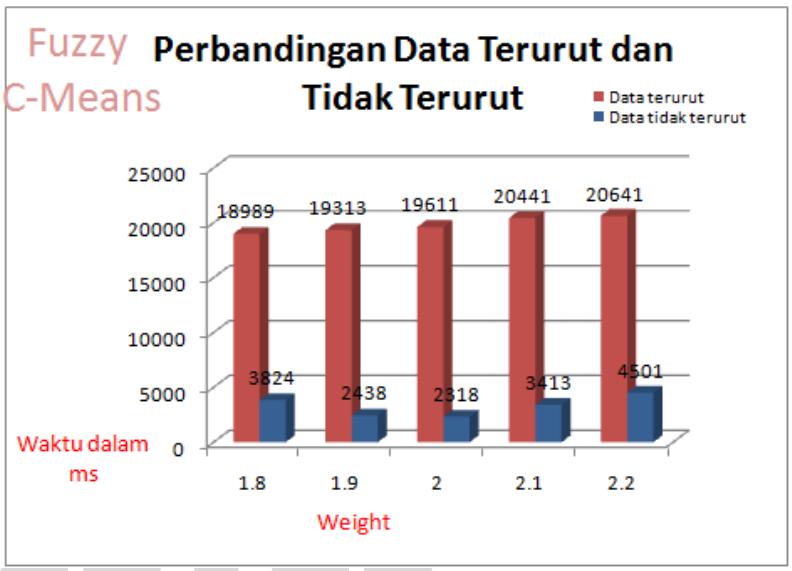
Berikut adalah grafik untuk hasil pengujian algoritma *Fuzzy C-Means* dengan data terurut. Fokus pengamatan pada lama pemrosesan data.



Gambar 5.23: Grafik Lama Pemrosesan Data Terurut pada *Fuzzy C-Means*

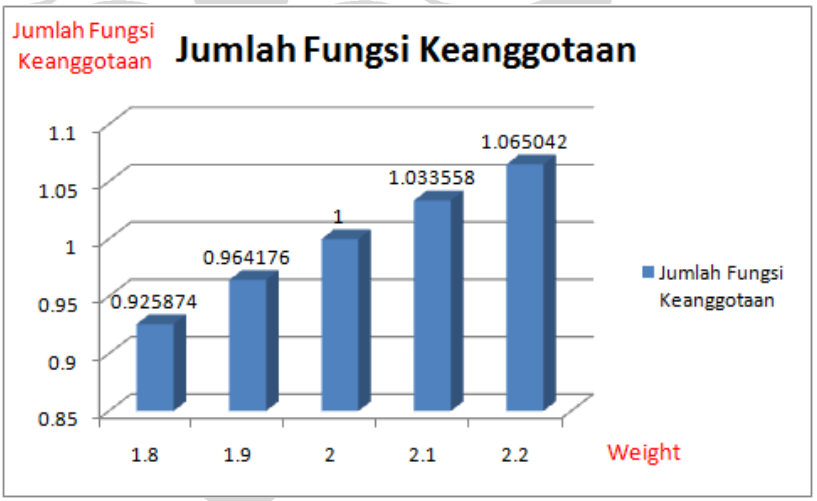
Berikut adalah grafik perbandingan lama pemrosesan data terurut dan tidak terurut pada *Fuzzy C-Means.*

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008



Gambar 5.24: Grafik Perbandingan Lama Pemrosesan Data pada *Fuzzy C-Means*

Berikut adalah grafik yang menggambarkan jumlah fungsi keanggotaan pada saat mengeksekusi *Fuzzy C-Means* pada FIKUI Mining.



Gambar 5.25: Grafik Perbandingan Jumlah Fungsi Keanggotaan pada *Fuzzy C-Means*

Dari dua grafik tersebut di atas dapat diketahui bahwa pemrosesan data yang tidak terurut membutuhkan waktu yang lebih sedikti dibandingkan pemrosesan pada data yang terurut. Ketika *weight* bernilai 2, waktu pemrosesan pada data yang tidak terurut mencapai nilai yang paling rendah yang berarti pemrosesan data terjadi dalam waktu yang paling cepat dibandingkan jika nilai dari *weight* dinaikan atau diturunkan. Jika *weight* lebih besar dari dua, fungsi keanggotaan akan berjumlah lebih besar dari

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008

satu yang artinya algoritma menjadi semakin *fuzzy* karena bisa jadi suatu data sebagian masuk satu kluster dan bagian lainnya masuk ke kluster lainnya. Jika *weight* lebih kecil dari pada dua, fungsi keanggotaan akan berjumlah kurang dari satu yang mengakibatkan algoritma semakin tidak *fuzzy*.

Integrasi dan peningkatan unjuk...,Akhda Afif Rasyidi...[et.al],FASILKOM UI, 2008