BAB I
PENDAHULUAN

1.1 Latar Belakang

Indonesia merupakan negara hutan hujan tropis yang memiliki keanekaragaman hayati yang tinggi dan dikenal sebagai salah satu *Megabiodiversity* Country.Pulau Sumatera salah satu pulau di Indonesia yang dapat menjadi gambaran kekayaan daerah tropis di Asia.Hal tersebut bisa dilihat dari flora dan fauna yang ada di pulau ini, yaitu satu diantaranya adalah badak Sumatera (Dicerorhinus sumatrensis, Fischer 1814).

Terdapat lima jenis badak yang masih hidup di dunia, tiga jenis berada di Asia yaitu *Rhinoceros unicornis* (badak India), *Rhinoceros sundaicus* (badak Jawa), *Dicerorhinus sumatrensis* (badak Sumatera) dan dua jenis lainnya berada di Afrika yaitu *Ceratotherium simum* (badak putih Afrika) dan *Diceros bicornis* (badak hitam Afrika) Lekagul & McNelly (1977). Dari tiga jenis badak yang hidup di Asia, dua jenis diantaranya hidup di Indonesia, namun kedua jenis badak ini statusnya terancam punah (*endangered*). Badak Sumatera tersebar di Pulau Sumatera dan Kalimantan (*Borneo*).Akan tetapi daerah penyebaran badak di Kalimantan sudah sangat langka dan jarang ditemui, hanya terdapat di Sabah dengan jumlah populasi 25 individu. Demikian pula di Sumatera jumlahnya semakin menurun dengan peta sebaran yang sudah sangat terbatas pada daerah tertentu saja terutama berada di Taman Nasional Gunung Leuser (Aceh), Taman Nasional Kerinci Seblat (Jambi), Taman Nasional Bukit Barisan Selatan (Sumatera Selatan) dan Taman Nasional Way Kambas, Lampung (YMR 1998).

Dari menurunnya jumlah peta sebaran Sumatera yang sudah sangat terbatas pada daerah tertentu terlihat bahwa populasi badak Sumatera dewasa ini semakin terancam keberadaannya.Hal ini disebabkan oleh beberapa faktor, diantaranya adalah semakin maraknya perburuan liar,

rusaknya habitat alamnya yang disebabkan oleh konversi hutan yang cenderung tidak terkendali dan perkawinan antar induk-anak, saudara kandung dan sepupu, mendorong terjadinya tekanan silang dalam (inbreeding) yang menyebabkan menurunnya keragaman genetik dalam populasinya maka dapat pula berupa kesulitan untuk menghasilkan keturunan, atau keturunan yang muncul biasanya lemah atau mandul.

Dengan adanya pengelompokan badak dapat mengurangi terjadinya perkawinan silang antar kerabat (inbreeding) yang menyebabkan menurunnya keragaman genetik. Pengelompokan badak Sumatera merupakan langkah penting yang harus dilakukan.Apabila usaha ini berhasil maka sangat mendukung dalam menekan laju kepunahan dan dapat melestarikan keberadaan badak sumatera tersebut.

Pengelompokan menggunakan DNA untuk mengetahui hubungan tingkat kekerabatan antara organisme satu dengan yang lainnya. Metode yang seringkali digunakan untuk mengelompokan DNA badak adalah metode *clustering.*Algoritma *clustering* sangat efektif dalam mengatasi data yang kompleks sehingga dapat dilakukan pengelompokan sesuai dengan yang dibutuhkan.

Pada penelitian sebelumnya yang dilakukan oleh Pranoto Budi Sasongko pada tahun 2010 yang berjudul "*Algoritma Partitioning clustering untuk protein clustering"*. Walaupun dalam bidang yang lain *partitioning techniques* banyak digunakan, namun dalam *protein clustering* hanya sedikit sekali aplikasi yang ada yang berdasarkan pada *partitioning techniques*. Metode baru yang dipaparkan dalam makalah ini, yaitu Algoritma *Pro-Kmeans* yang memanfaatkan *Smith-Waterman local-alignment algorithm*. Metode*Pro-Kmeans* untuk menentukan kesamaan dari rantai protein(Sasongko P.B, 2010). Pada penelitian yang dilakukan Millatul Ulya pada tahun 2010 yang berjudul " *Klastering varietas Padi Modifikasi Metode K-Means Berbasis Ordered Weighted Averaging (OWA)",* metode *K-Means* hasil protein memiliki tingkat akurasi 89,33%.Berdasarkan nilai siluet dan SSE, maka jumlahmetode k-*means* dan*hierarchical clustering* dalam klastering varietas padi karena nilai

SSEnya palingkecil (Ulya M, 2010).Berdasarkan referensi jurnal yang ada metode *Pro-Kmeans* dan K-Means memiliki tingkat akurasi yang berbeda.Pada metode *Pro-Kmeans* memilikii hasil akurasi tingkat kesamaan yang jauh lebih baik daripada K-Means.

Berdasarkan latar belakang yang telah dipaparkanmaka judul dari penelitian ini adalah**Implementasi Algoritma *Pro-Kmeans*dalam Pengelompokkan Badak Sumatera** diharapkan dapat dijadikan data base dalam usaha manajemen populasi dan konservasinya.

1.2 Rumusan Masalah

Permasalahan yang dibahas dalam penelitian ini adalah:

1. Bagaimana merancang suatu aplikasi yang dapat digunakan untuk mengelompokan data badak Sumatera menggunakan metode *Pro-Kmeans*?.
2. Berapa besar tingkat akurasi dari metode *Pro-Kmeans*?

1.3 Tujuan dan Manfaat

1.3.1 Tujuan

Tujuan dari penelitian ini adalah :

1. Menerapkan Algoritma *Pro-Kmeans* pada proses pengelompokkan data badak Sumatera.
2 Menghitung akurasi dari proses pengelompokkan data badak Sumatera.

1.3.2 Manfaat

Diharapkan dapat memberikan informasi tentang keragaman genetik badak Sumatera sebagai penanda dalam usaha pengelolaan program-program penangkaran atau usaha konservasi seperti untuk perkawinan dalam mencegah terjadinya "*in breeding*". Disamping itu juga diharapkan dapat dimanfaatkan untuk mempelajari keragaman genetik dan biologi populasi badak Sumatera serta informasi proses penurunan keragaman genetik, dan memberikan kontribusi terhadap upaya penyelamatan badak Sumatera agar terhindar dari kepunahan.

1.4 BatasanMasalah

1. Perancangan aplikasi pengelompokan data badak Sumatera menggunakan metode *ProK-Means.*
2. Data yang digunakan dalam skripsi ini bersumber dari data genbank*(www.ncbi.nlm.nih.gov/genbank)* danjurnal Analisis DNA Mitokondria Badak Sumatera dalam Konservasi Genetika (Handayani, 2008).
3. Data badak Sumatera meliputi badak Torgamba, Bina, Rosa, Andalas.
4. Data Wilayah Sumatera meliputi Taman Nasional Gunung Leuser (Aceh), Taman Nasional Kerinci Seblat (Jambi), Taman Nasional Bukit Barisan Selatan (Sumatera Selatan) dan Taman Nasional Way Kambas, Lampung.

1.5 MetodologiPengerjaan Skripsi

1.5.1 StudiLiteratur

Dalam perancangan sistem ini terlebih dahulu mempelajari studi literature yang di lakukan dengan cara mengumpulkan data dan mempelajari segala macam informasi yang berhubungan dengan sistem informasi, metode *clustering* dan tentang badak Sumatera.

1.5.2 IdentifikasiMetodeAnalisa

Dengan melihat permasalahan yang ada dan disesuaikan dengan teori – teori yang ada, maka dicari metode yang paling tepat digunakan dalam penyelesaian permasalahan ini. Adapun metode yang dipilih untuk digunakan dalam penelitian ini adalah dengan metode *ProK-Means.*

1.5.3 Pengumpulan Data

Pengumpulan data dilakukan dengan mengambil data badak Sumatera dari data genbank*(www.ncbi.nlm.nih.gov/genbank)*dan jurnal Analisis DNA Mitokondria Badak Sumatera dalam Konservasi Genetika (Handayani, 2008).

.

1.5.4 Analisa Dan Perancangan Sistem

1.5.4.1 Analisa Sistem

Pada tahap ini akan di lakukan analisis terhadap sistem yang akan di implementasikan, di antaranya:

1. Mencatat kebutuhan sistem dan pemodelan sistem yang akan di buat.
2. Mendesain dan membuat data menggunakan Microsoft Office Excel 2010.
3. Aplikasi berbasis desktop
4. Aplikasi dibangun menggunakan bahasa C# dengan IDE Visual Studio 2010.

1.5.4.2 Perancangan Sistem

Pada tahapan ini di lakukan perancangan sistem yang akan di implementasikan, di antaranya membuat alur dari kerja sistem berdasarkan analisa yang telah di lakukan.

1.5.5 Implementasi Sistem

Membangun user interface dan pengkodingan program berbasis desktop menggunakan Visual Studio (C#) yang sesuai dengan rancangan sistem yang telah di buat.

1.5.6 Pengujian

Menguji sistem yaitu dengan membandingkan hasil pengelompokan sistem dengan hasil yang diperoleh dengan perhitungan manual.

1.5.7 Penyusunan Laporan Skripsi

Merupakan Langkah terakhir penyusunan dan penulisan laporan setelah proses implementasi dan analisis dilakukan. Dengan adanya penyusunan laporan diharapkan dokumentasi sistem tersusun dengan rapi.

1.6 Sistematika Penulisan

Adapun sistematika penulisan tugas akhir ini adalah sebagai berikut:

BAB I : PENDAHULUAN

Bab ini terdiri dari : latar belakang, perumusan masalah, batasan masalah, tujuan penelitian, metode penelitian dan sistematika penulisan.

BAB II : TEORI PENUNJANG

Bab ini menjelaskan mengenai hal-hal yang melandasi atau yang akan digunakan sebagai pedoman untuk menyelesaikan permasalahan yang terdapat dalam penelitian ini.

BAB III : PERANCANGAN SISTEM

Bab ini menjelaskan pemodelan dan rancangan yang akan digunakan dalam melakukan penelitian.

BAB IV : IMPLEMENTASI SISTEM DAN ANALISIS

Bab ini menjelaskan tentang implementasi program yang telah dikerjakan.Meliputi penjelasan tentang arsitektur sistem, desain sistem, desain interface, dan menguraikan tentang hasil uji coba serta analisa terhadap sistem.

BAB V : PENUTUP

Bab yang berisi kesimpulan dan saran yang dihasilkan dari sistem yang telah diujicobakan.

BAB II
TINJAUAN PUSTAKA

2.1 Klasifikasi

Gambar 2.1 *Dicerorhinus sumatrensis*

Termasuk dalam kelas *Mamalia*, *Ordo Mesaxonia, Sub Ordo Perissodactyla, Famili Rhinocerotidae, Genus Dicerorhinus, Spesies Dicerorhinus sumatrensis Fischer*, 1814.

Badak sumatera merupakan anggota famili Rhinocerotidae.Famili ini terdiri dari empat genus yaitu Rhinoceros (bercula satu), Dicerorhinus (bercula dua), Diceros (bercula dua) dan Ceratotherium (bercula dua).Rhinoceros terdiri dari dua species sedangkan Dicerorhinus, Diceros dan Ceratotherium terdiri dari satu spesies (Xu & Arnason 1996).Berdasarkan ukurannya maka kelima spesies tersebut termasuk dalam mamalia darat yang berukuran besar.Species dari genus Rhinoceros terdapat hanya di Asia yaitu badak jawa (*Rhinoceros sundicus*) di Ujung Kulon Jawa Barat, sedangkan Badak India (*Rhinoceros unicornis*) terdapat di India terutama di daerah Assam.*Genus Diceros* dan *Ceratotherium* terdapat di Afrika (*Diceros bicornis*) dan (*Ceratotherium simum*).Sedangkan genus Dicerorhinus satu spesies *(Dicerorhinus sumatrensis*) berada di Asia. Badak bercula dua di Asia memiliki tiga subspesies dengan sebaran geografis yang berbeda, yaitu Dicerorhinus *sumatrensis* di Sumatera-Indonesia, Semenanjung Malaysia, dan Thailand; Dicerorhinus *sumatrensis harrisoni* ditemukan di Kalimantan, sedangkan *Dicerorhinus sumatrensis lasiotis* ditemukan di Myanmar (Foose & van Strein 1997)

Ketiga spesies badak Asia dahulu tersebar luas di selatan dan tenggara Asia.Jumlah mereka cukup melimpah pada pertengahan abad ke-19.Spesies-spesies badak Asia kini menjadi mamalia paling langka dan kehidupannya

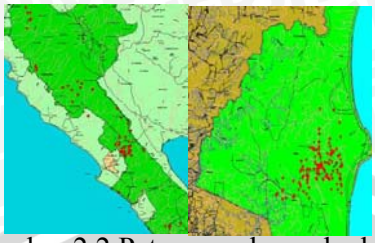
terancam di dunia.Usaha konservasi terhadap badak India cukup berhasil, sedangkan badak Jawa dan Sumatera kini berada dalam ancaman kepunahan (Foose & van Strein 1997; WWF 2002).

Badak Sumatera umumnya ditemukan di daerah berbukit-bukit yang dekat dengan air.Spesies tersebut menempati hutan hujan tropis dan hutan lumut pegunungan, tetapi juga menyukai daerah pinggiran hutan dan hutan sekunder (van Strein 1985).Badak Sumatera dapat hidup pada kisaran rentang habitat yang luas, mulai dari rawa-rawa dataran rendah hingga hutan pegunungan.Saat ini, badak sumatera di temukan di dataran tinggi karena hutan dataran rendah sudah berkurang.Dahulu, spesies ini menyebar luas mulai dari daerah dataran rendah hingga dataran tinggi, dan bahkan dapat berenang di laut untuk mencapai pesisir pulau (van Strein 1985).

2.2 Penyebaran

Populasi badak Sumatera terpecah dalam kelompok populasi yang berbeda dengan sebaran terkonsentrasi pada daerah-daerah tertentu saja. Populasi yang ada di Sumatera terdapat di daerah Taman Nasional (TN) Gunung Leuser, TN Kerinci Seblat (TNKS), Aceh Utara (Gunung Abong-abong dan Lesten Lukup), TN Bukit Barisan Selatan (TNBBS) dan TN Way Kambas (TNWK). Sebaran populasi di Kalimantan pernah dilaporkan terdapat di TN Kayan Mentarang dan Cagar Alam Ulu Sembakung, tetapi jumlahnya sampai saat ini belum di ketahui dengan pasti (Dirjen PHPA Dephut RI & YMR 1994). Adapun peta sebaran Badak sumatera di TNWK dan TNBBS dapat dilihat pada gambar 3 berikut ini.

Keadaan Populasi pada tahun 2004 diketahui populasi badak sumatera terdapat di TN Way kambas Lampung hanya berjumlah 15-25 ekor, sedangkan TNBB Selatan 60-80 ekor, dan TNKS 2-3 ekor, dan hasil survey terbaru pada tahun 2005 populasi badak Sumatera berdasarkan data jejak (footprint) yang dilaporkan hanya tinggal 20 - 27 ekor (Isnan et al. 2005)



Gambar 2.2 Peta penyebaran badak Sumatera
di TNWK dan TNBBS.
YABI (Yayasan Badak Indonesia 2005)

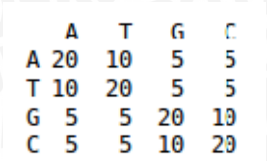
2.3 DNA

DNA (*Deoxyribonucleic acid*) merupakan bahan genetic yang memberi informasi genetik dari sel ke sel dan satugenerasi ke generasi berikutnya.Sebuah pita molekul DNAterdiri dari tiga persenyawaan kimia, yaitu asam pospat, guladeoksibosa, dan basa nitrogen. Tulang punggung asam pospatdeoksiribosaselalu sama untuk berbagai segmen dari molekulDNA (Suryo, 2001).

DNA merupakan asam nukleotida yang mengandunginstruksi genetik untuk pengembangan dan fungsional padaseluruh organisme hidup dan beberapa virus.Fungsi utama darimolekul DNA adalah sebagai bagian penyimpanan informasijangka panjang.DNA juga sering diistilahkan sebagai sebuahcetak biru, resep, atau kode instruksi yang dibutuhkan untukmengkonstruk komponen sel lainnya seperti protein danmolekul RNA.Segmen DNA yang mengangkut informasigenetik ini disebut gen, tetapi beberapa DNA memiliki strukturtertentu (Triwibowo Yuwono, 2002).

2.4 Matriks Skor DNA

Matriks skor DNA lebih sederhana dibandingkan matriks skor pada asam amino. Hal ini dikarenakan oleh kombinasi hanya pada empat basa yaitu Adenin (A),Timin (T), Guanin (G), dan Sitosin (C). Matriks skor DNA ditunjukkan pada gambar 2.4 (Arthur Lesk, 2002).



	A	T	G	C
A	20	10	5	5
T	10	20	5	5
G	5	5	20	10
C	5	5	10	20

Gambar 2.3. Matriks skor DNA

2.5 Algoritma Smith-Waterman

Algoritma *Smith-Waterman* ditemukan oleh Temple Smith dan Michael Waterman pada tahun 1981. Algoritma *Smith-Waterman* memiliki proses *sequencealignment* yang mengaplikasikan secara dasar *dynamic programming.* Secara umum langkah-langkah yang digunakan adalah :

1. Memecah suatu permasalahan umum menjadi sub permasalahan yang lebih sederhana
2. Memecah semua sub permasalahan secara optimal
3. Mengkontruksi pemecahan optimal sub permasalahan sebagai pemecah optimal permasalah secara umum

Algoritma *Smith Waterman* merupakan algoritma klasik yang telah dikenal luas sebagai metode yang dapat mengidentifikasi *local similarities* yaitu proses penyusunan dua *localsequencenucleotide* atau *protein sequence* sehingga kemiripan antara dua sekuen tersebut akan terlihat. Berdasarkan fungsi proses pensejajaran sekuen tersebut, maka algoritma ini dapat dikonversikan ke dalam pemrograman komputer untuk digunakan membantu proses pendeteksian kesamaan data curah hujan antar kota antar bulan(Smith TF dan Waterman MS,1989).

Pada algoritma smith-waterman terdapat perhitungan *similarity*yang dilakukan sebagai berikut :

Similarity Score=Σ(*similarAA* + *identikAA)* / *Alignedlenght* .. (2.1)

2.6 Algoritma *Pro-Kmeans*

Pro-Kmeansalgorithm yang diajukan pada penelitian ini dimulai dengan partisi secara acak dari data set D kedalam K *clusters* dan kemudian

menggunakan Algoritma *Smith-Waterman* untuk membandingkan protein dari masing-masing *cluster* Si(i€|1..k|) dan untuk menghitung *SumScore*(Si, Oi) dari masing-masing protein j pada Si sebagai berikut(SasongkoP.B., 2010).

SumScore(Si,Oi)=∑ w€|1..m|≠j Score(Oj,Ow), ….. (2.2)

Dimana m adalah ukuran dari Subset Si, dimanayang termasukdalamobjekOj dan Score (Oj,Ow) merupakan hasil dari proses *similarity* yang akan di sumscore.

Sedangkan untuk perhitungan score

$Score(A,B)= \sum_{i,j} S(A_i,B_j) - \sum_n g(n)$, …. (2.3)

Dimana $\sum_{i,j} S(A_i,B_j)$ merupakan hasil dari proses *similarity value* yang akan dikurangi g(n) untuk mendapatkan centroid yang baru.

Untuk analisa *cluster* digunakan fungsi f(v) dimana fungsi tersebut merupakan fungsi global yang berguna untuk mengevaluasi hasil *cluster*. Fungsi f(v) ditunjukkan pada persaman

$f(V) = \sum_{j€|1..n|} Score(O_j,R_i)$, …. (2.4)

Dimana $\sum_{j€|1..n|} Score(O_j,R_i)$ merupakan hasil dari seluruh score (A,B).

Dimana persamaan g(n) adalah

$g(n) = P_o+(n-1)*P_e$ …… (2.5)

Po = Gap Opening Penalty

Pe = Gap Extend Penalty

Gap open penalty adalah skor yang diperolehdari inisialisasi awal dari gap pada sekuens. Gap extendpenalty ditambahkan pada gap penalty untuk setiap gap padaresidu. Gap extend memiliki nilai yang lebih kecil dari gapopen. Nilai gap open merupakan bilangan bulat positif dannilai gap extend merupakan bilangan desimal positif.

Protein Oj di masing-masing *cluster* yang memiliki *SumScore*(Si,Oj) maksimum dipertimbangkan sebagai centroid Ri dari *cluster*. Algoritma *Smith-Waterman* yang digunakan di sini untuk membandingkan masing-masing protein Oh dari data set D dengan *centroids* Ri dan untuk memasukkan protein yang ditinjau ke dalam *cluster* terdekatDimana Ri memiliki nilai maksimum dari kemiripan dengan object Oh. *Pro-Kmeans* melakukan prosedur q kali, untuk memaksimumkan fungsi f(V). Jumlah masukan adalah jumlah *cluster*s, K, dan

jumlah iterasi q, dan akan menghasilkan partisi terbaik dari data set D dan center atau rata-rata dari *cluster* Si.

Pseudo-code untuk *Pro-Kmeans Algorithm*:

Input: A training set D, D= {Oh}h=1..n; n adalah ukuran dari D

Initialize: f(V)max = 0; iterasi = 0;

Repeat

Begin

1. Partisi secara acak D ke dalam K *cluster* yang tidak kosong
2. For each I € [I .. K] do
   - Hitung *SumScore*(Si,Oj) dari masing-masing protein j pada Si;
   - Protein j yang memiliki nilai *SumScore*(Si,Oj) maksimum pada Si akan menjadi *Centroid* Ri pada subset Si
3. For each Oh € D do
   - Hitung *similarity score* dari Oh dengan masing-masing centroid Ri(i€[i..k]), menggunakan *Smith Waterman Algorithm*
   - Assign Oh ke dalam *cluster* dengan Ri terdekat; (Ri yang memiliki maksimum *score of similarity* dengan object Oh)
4. Hitung f(V);
5. If f(V)<f(V)max then

   Iteration = iteration +1;

   Else

   F(V)max = f(V);

   BestSet = CurrentSets; (CurrentSet adalah Subset yang ada pada partition ini)

   Kembali ke langkah ke-2

Iteration = iteration +1;

Until iteration = q;

Else

BestSets

End

Output : *BestSets*. *BestSets* adalah partisi terbaik dari D ke dalam K *cluster*; masing-masing *cluster* didefinisikan dengan adanya sebuah *centroid* Ri (SasongkoP.B., 2010).

2.7 *Error Ratio*

*Error Ratio*merupakan sebuah nilai kesalahan yang diperoleh dari hasil perhitungan klaster oleh suatu metode dimana hasil yang diharapkan dari suatu proses pengklasteran tidak sesuai dengan yang diharapkan. *Error Ratio* dipakai jika *dataset* yang digunakan adalah s*upervised*, namun bisa juga digunakan untuk mengukur tingkat presisi dari metode klasterisasi. Menurut Barakbah dan Arai (2007), nilai *error* ini diperoleh dari perhitungan matematis sesuai rumus pada persamaan 2.9:

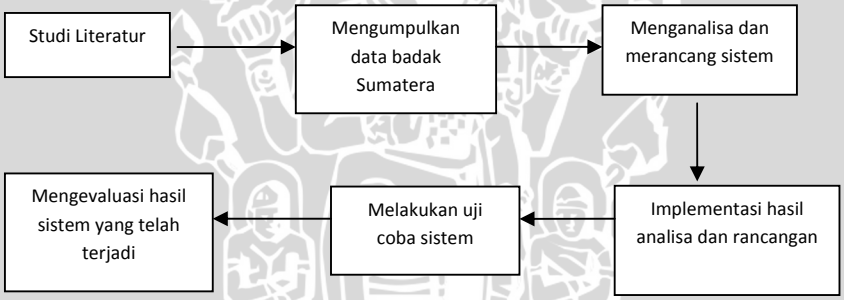
$$Error = \frac{missclassified}{jumlahdata} \times 100\%$$

Halaman ini sengaja dikosongkan

BAB III
METODOLOGI DAN PERANCANGAN SISTEM

Penelitian tentang implementasi algoritma *Pro-Kmeans* dalam menentukan pola badak Sumatera ini harus mengikuti beberapa tahapan. Tahap-tahap untuk melakukan penelitian ini adalah sebagai berikut :

1. Melakukan studi literatur dengan mengambil data DNA dari jurnal,
2. Mengumpulkan data badak Sumatera yang telah disediakan,
3. Melakukan perancangan sistem dengan menggunakan metode *Pro-Kmeans*,
4. Mengimplementasikan hasil analisa dan rancangan yang dilakukan pada tahap sebelumnya menjadi sistem pensejajaran data badak Sumatera,
5. Melakukan uji coba sistem dengan menggunakan data yang akan di uji coba,
6. Mengevaluasi hasil sistem dalam melakukan penyejajaran data badak Sumatera yang sesuai dengan aturan yang telah ditetapkan.



Gambar 3.1. Bagan alur tahapan penelitian

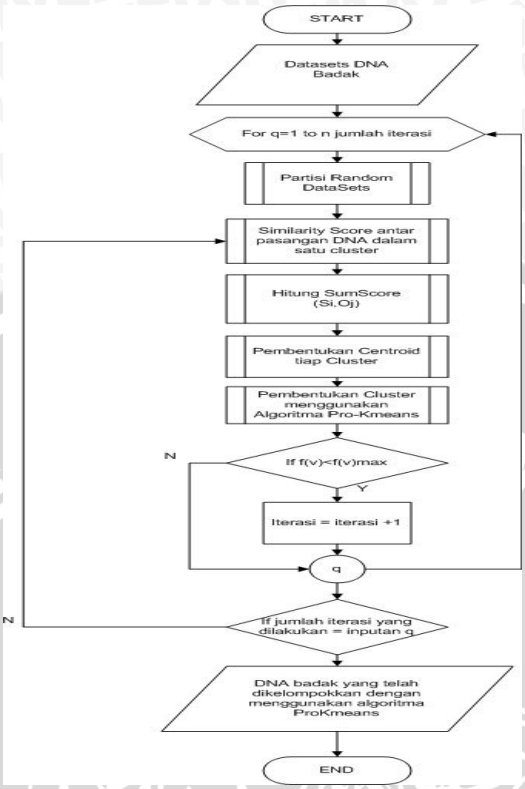
3.1 Perancangan Sistem

3.1.1 Deskripsi Sistem

Sistem yang akan dibuat merupakan sistem yang digunakan untuk pengelompokan data badak Sumatera dengan menggunakan metode *Pro-Kmeans*.

3.1.2 Batasan Sistem

Batasan sistem penelitian ini bahwa algoritma *Pro-Kmeans* menggunakan data badak Sumatera.Berikut flowchart perancangan sistem dapat dilihat pada gambar 3.2



Gambar 3.2 Flowchart Perancangan Sistem

1. Perancangan aplikasi pengelompokan data badak Sumatera menggunakan metode *ProK-Means.*
2. Proses yang dilakukan pada sistem ini adalah melakukan partisi data secara random ke dalam K *cluster*.
3. Menghitung Sum score untuk menentukan *centroid* dalam tiap *cluster*.
4. Mencari *similarity* pada masing-masing objek .
5. Menghitung f(v) untuk mendapatkan *BestSets*badak Sumatera. *BestSets* adalah partisi terbaik dari D ke dalam K *cluster.*

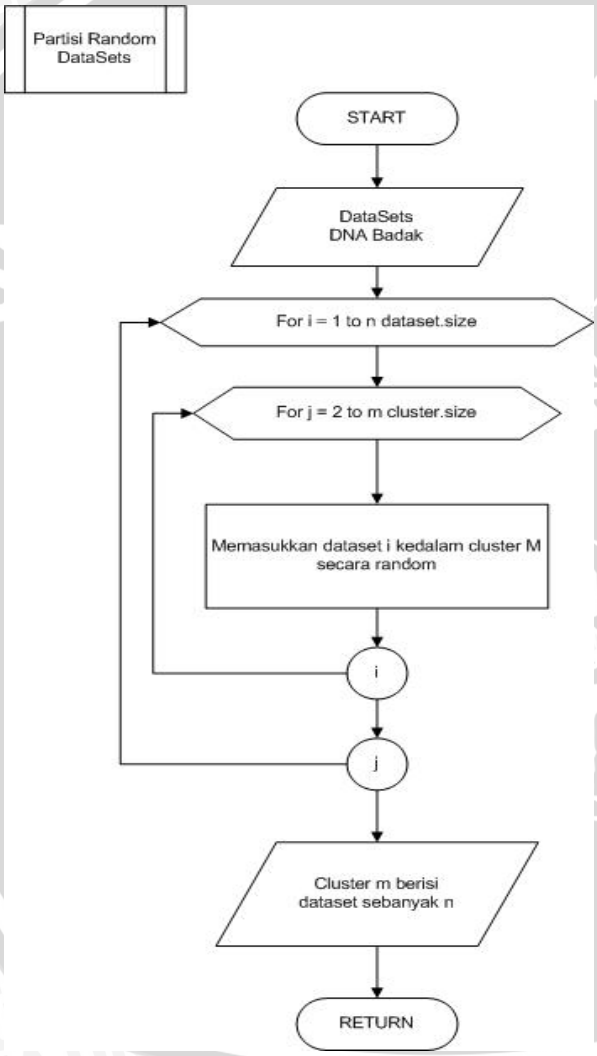
3.2 Perancangan Penelitian

Terdapat beberapa proses dalam perancangan penelitian, yaitu :

3.2.1 Proses Partisi Data

Proses Partisi data merupakan proses pengambilan data secara acak (random) ke dalam K *cluster* yang tidak kosong.

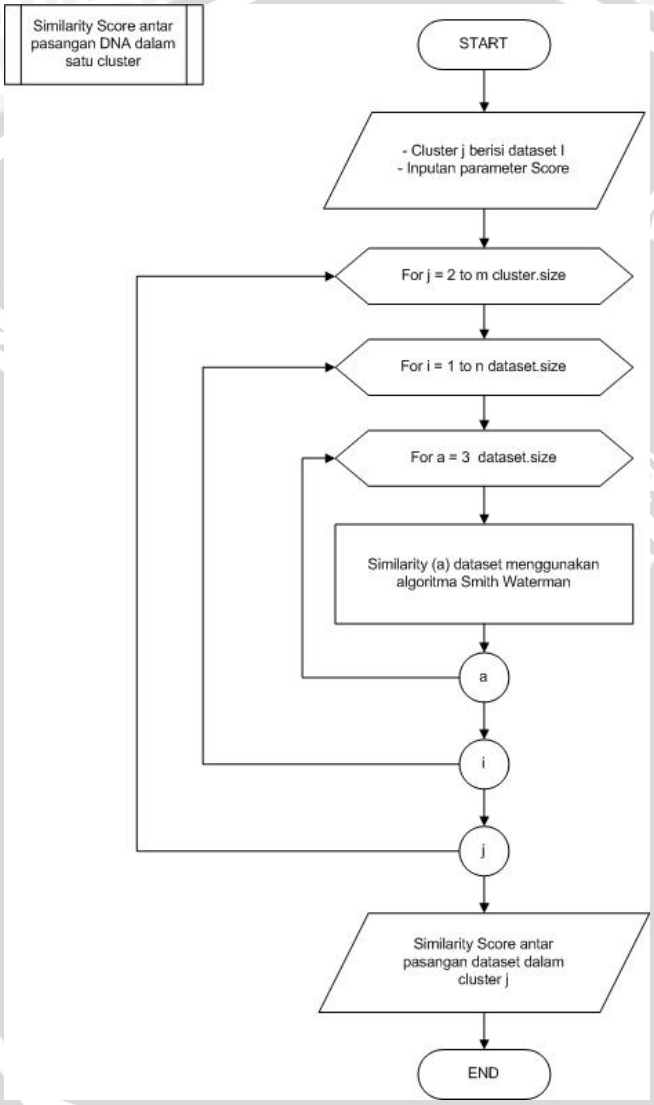
Flowchart proses partisi data dapat dilihat pada gambar 3.3



Gambar 3.3 Flowchart Proses Partisi data

3.2.2 Proses *Similarity* Score Antar Pasangan DNA dalam 1 *cluster*

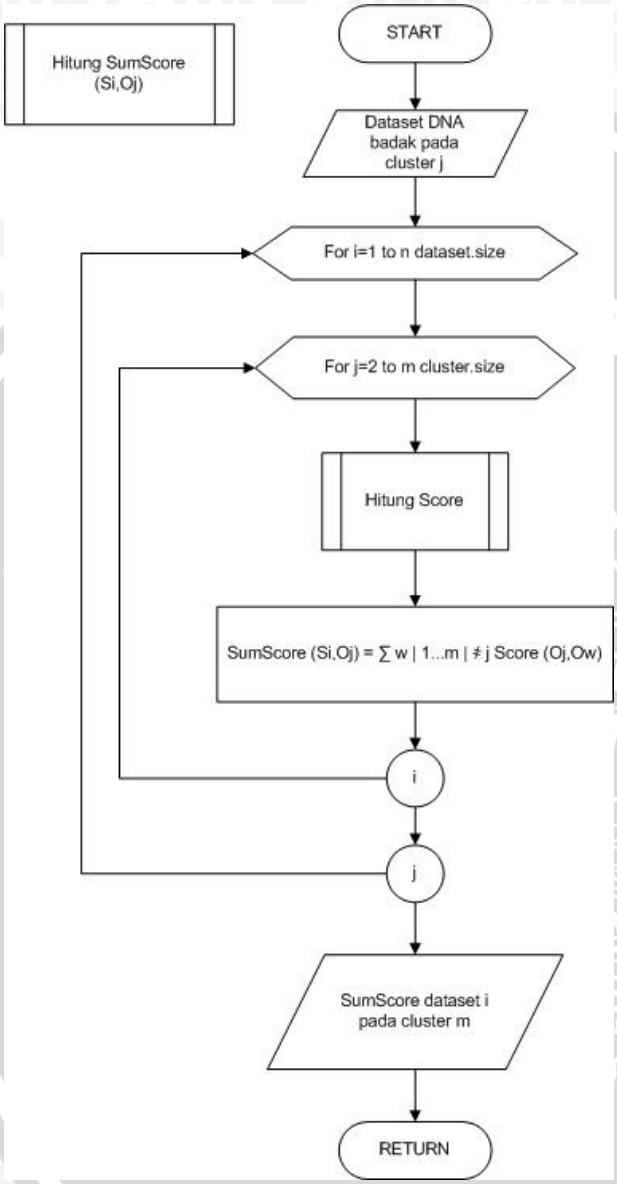
Pada proses hitung *Similarity* Score Antar Pasangan DNA dalam 1 *cluster*merupakan proses dari perhitungan untuk menentukan pasangan dalam tiap *cluster*. Flowchart dapat dilihat pada gambar 3.4



Gambar 3.4 Flowchart Proses *Similarity* Score Antar Pasangan DNA dalam 1 *cluster*

3.2.3 Proses *SumScore*

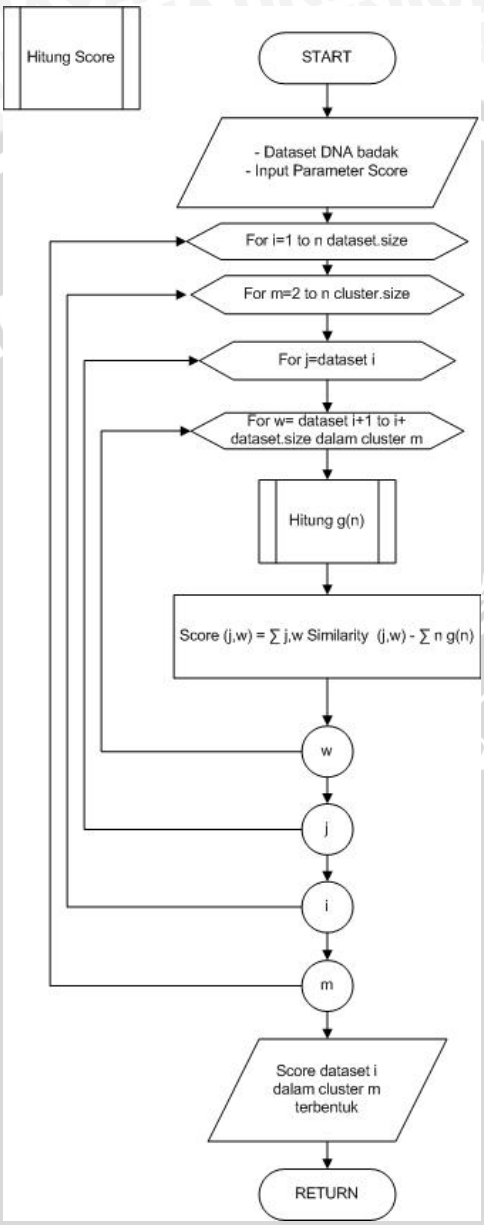
Proses *SumScore*dapat dilihat pada flowchart gambar 3.5



Gambar 3.5 Flowchart Proses *SumScore*

3.2.3.1 Proses HitungScore

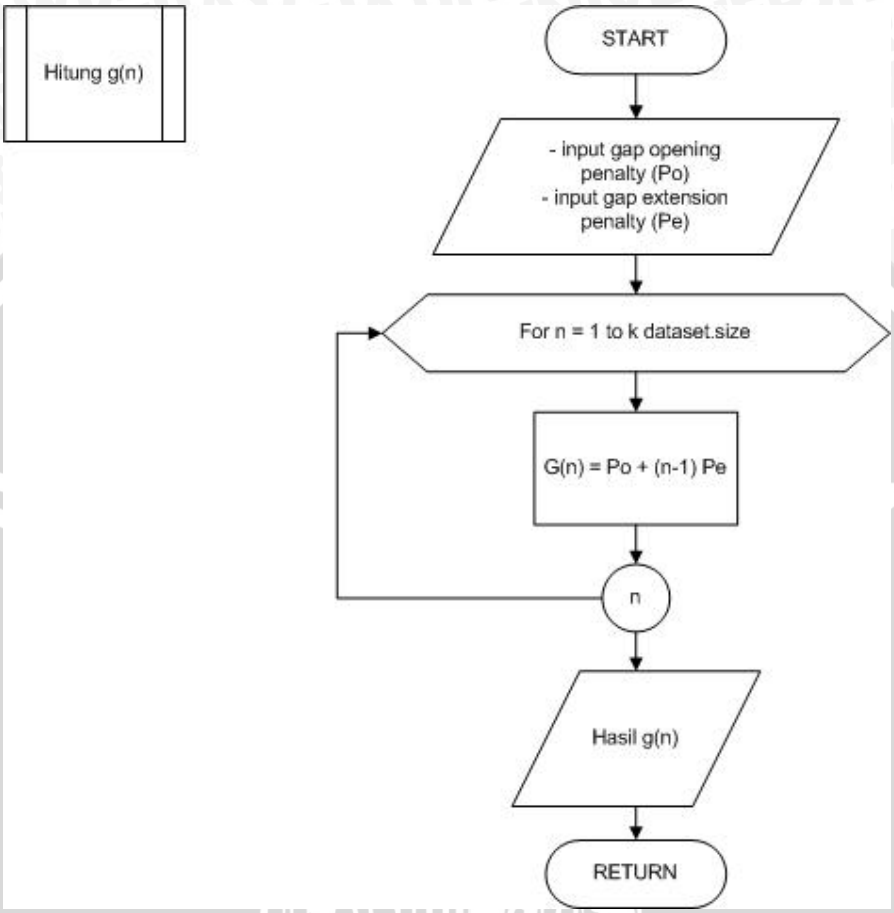
Flowchart pada proses hitung score terdapat pada gambar 3.6



Gambar 3.6 Flowchart Proses Hitung Score

3.2.3.2 Proses Hitung g(n)

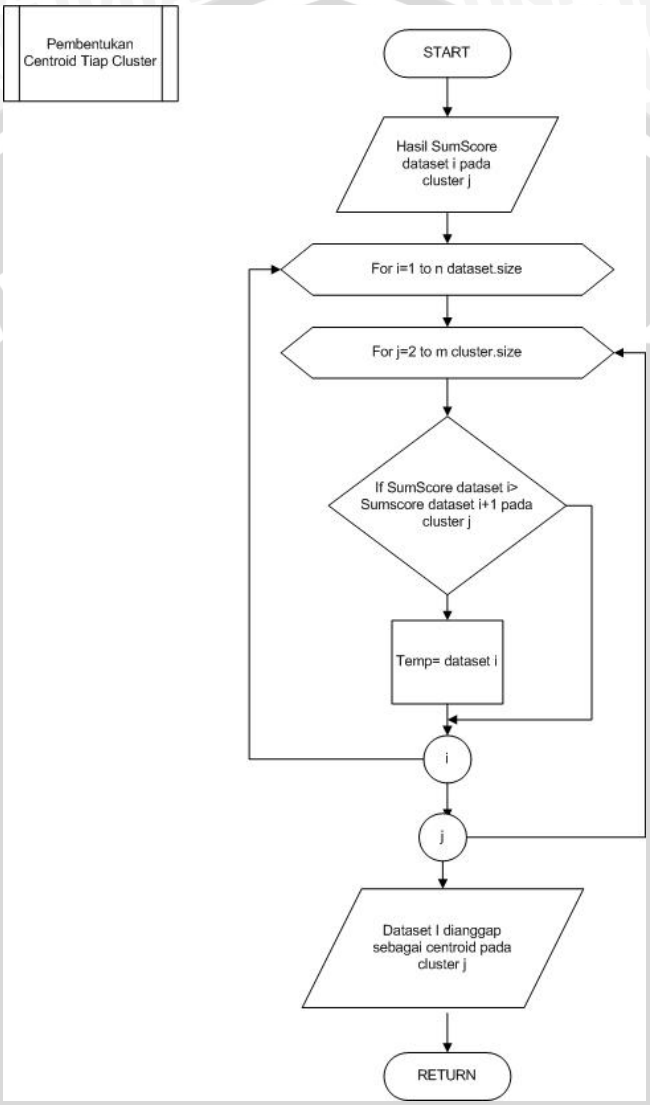
Flowchart pada proses hitung g(n) terdapat pada gambar 3.7



Gambar 3.7 Flowchart Proses Hitung g(n)

3.2.4 Proses Pembentukan Centroid Tiap *Cluster*

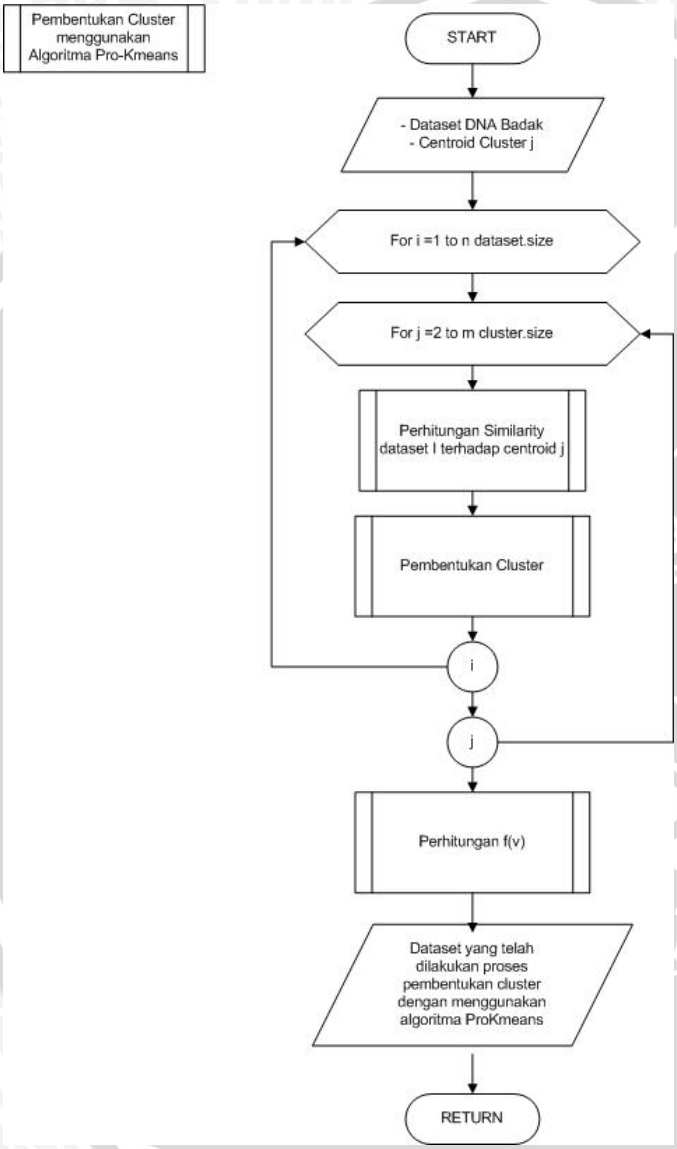
Pada proses Pembentukan Centroid Tiap *Cluster*terdapat pada gambar 3.8



Gambar 3.8Flowchart Proses Pembentukan Centroid Tiap *Cluster*.

3.2.5 Proses Pembentukan *Cluster* Menggunakan Algoritma ProKmeans

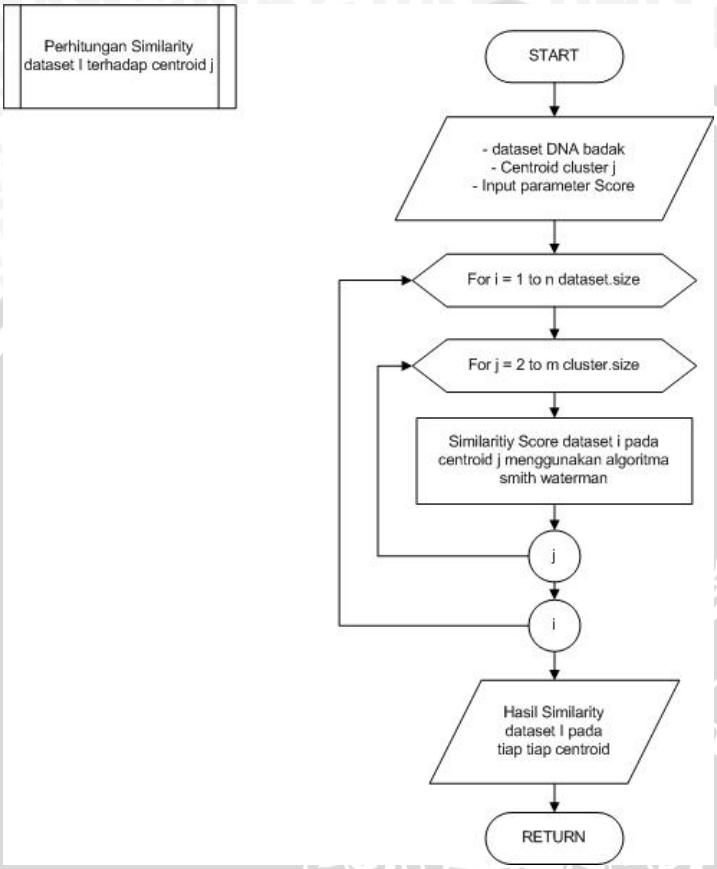
Pada proses Pembentukan *Cluster* Menggunakan Algoritma ProKmeansterdapat pada gambar 3.9



Gambar 3.9 Flowchart Proses Pembentukan *Cluster* Menggunakan Algoritma *ProKmeans.*

3.2.5.1 Proses Perhitungan *Similarity* Dataset i terhadap *Centroid* j

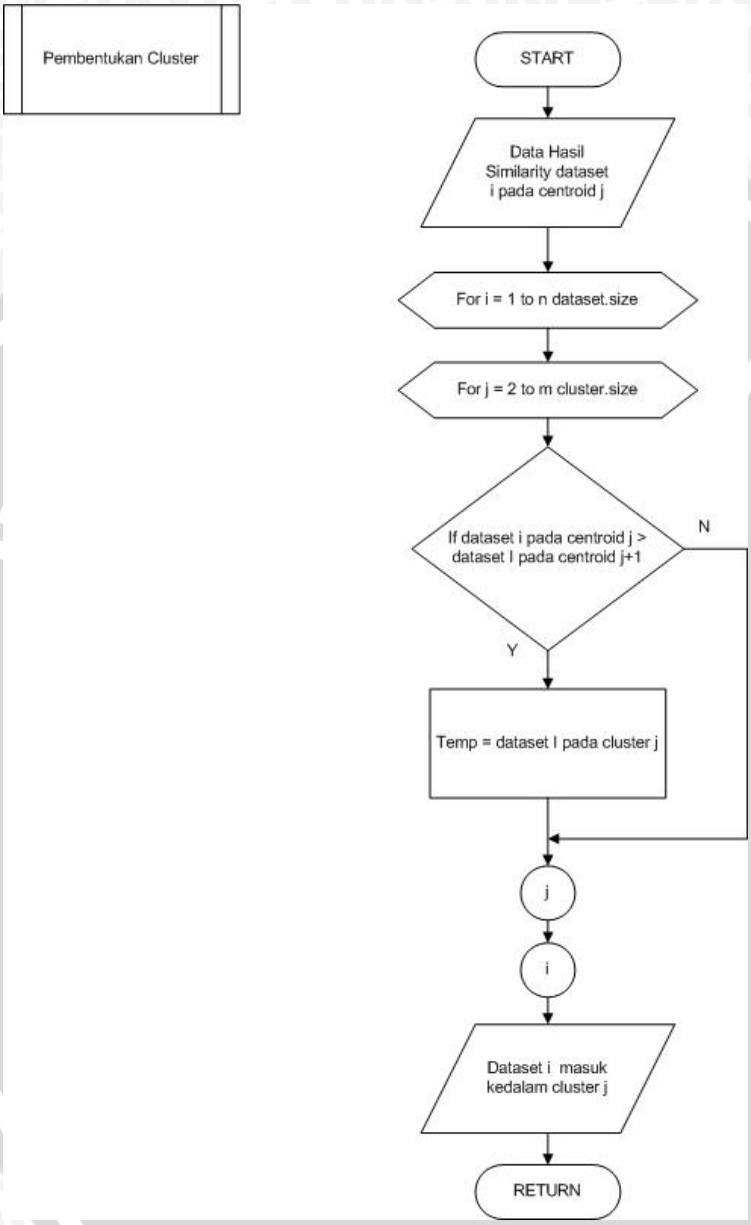
Pada prosesPerhitungan *Similarity* Dataset i terhadap Centroid j terdapat pada gambar 3.10



Gambar 3.10 Flowchart Proses Perhitungan *Similarity* dataset I terhadap *Centroid* J

3.2.5.2 Proses Pembentukan *Cluster*

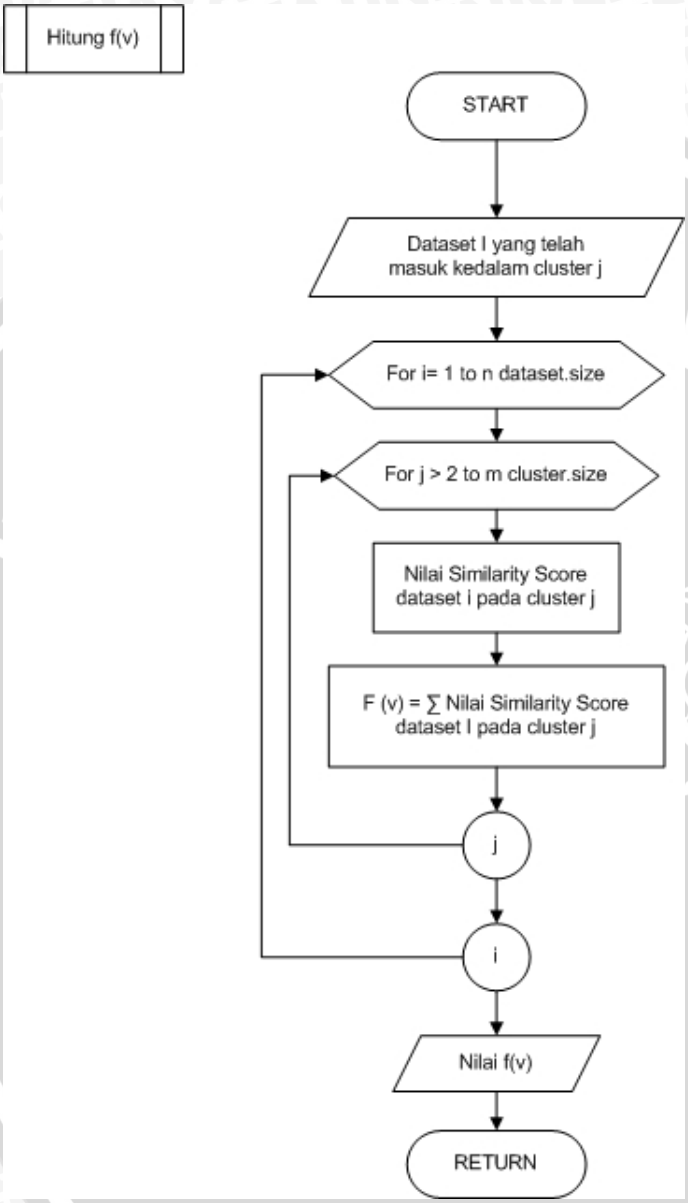
Pada proses pembentukan *cluster* terdapat pada gambar 3.11



Gambar 3.11 Flowchart Proses Pembentukan *Cluster*

3.2.5.3 Proses Perhitungan f(v)

Pada proses perhitungan f(v) terdapat pada gambar 3.12



Gambar 3.12 Flowchart Proses Perhitungan f(v)

3.3 Implementasi Pada Data

3.3.1 Data

Data yang akan dilakukan perhitungan menggunakan data badak Sumatera yang memiliki atribut DNA.

Berikut data yang telah ditentukan dapat dilihat pada tabel 3. 1.

Tabel 3.1 Data Badak Sumatera

Atribut	Struktur DNA
A1	CACAGGCAAGGGTAGGTAGGTAGCA
A2	AACAGGCAAGGGTTCGTAGCTTTCT
A3	AACAATCCTATCTACGTAGATTTCT
A4	CACAATCCTGTTGGGAGAGCGTTCT
A5	CAGTAAGCTGAACTAGGACATTCTT

Keterangan : A1-A5 merupakan jenis-jenis badak

3.3.2 Partisi Objek secara Random Kedalam *Cluster* k

Dilakukan partisi random secara acak pada kelima badak tersebut dengan penentuan *cluster* =2.Didapatkan

- *Cluster* 1 :A1, A2.
- *Cluster* 2 :A3, A4, A5.

3.3.3 Hitung *Similarity* Score tiap pasangan yang terdapat dalam K *cluster*

Melakukan perhitungan *Similarity Score*untuk A1 dan A2 dengan berpatokan kemiripan pada gambar 2.3.Berikut perhitungan *Similarity Score* dapat dilihat pada tabel 3.2

Tabel 3.2 Perhitungan *Similarity Score*

Atribut	Nilai *Similarity* Score
A1,A2	0.92
A3,A4	0.64
A3,A5	0.68
A4,A5	0.72

3.3.4 Perhitungan *SumScore*

Pada perhitungan *SumScore* dilakukan untuk mencari nilai maksimum centroid yang digunakan untuk menghitung proses *Similarity* value. Maka *SumScore* nya adalah

(Si,A1) = ∑ *Similarity* Score (A1,A2) = 0.92

(Si,A2) = ∑ *Similarity* Score (A2,A1) = 0.92

(Si,A3) = ∑ *Similarity* Score(A3,A4)(A3,A5)=0.64+0.68=1.32

(Si,A4) = ∑ *Similarity* Score(A4,A3)(A4,A5)=0,64+0.72=1.36

(Si,A5) = ∑ *Similarity* Score(A5,A3)(A5,A4)=0.68+0.72=1.4

SumScore (Si,Aj) yang terbesar akan dijadikan centroid
Jadi Centroid *Cluster* 1 = A1
Centroid *Cluster* 2 = A5

3.3.5 Perhitungan *Similarity Value* Menggunakan *Smith-Waterman*

Dari perhitungan *Sum Score* tersebut diatas didapatkan nilai *centroid* (pusat *cluster*) yang ditentukan berdasarkan nilai *max Sum Score* obyekyang terdapat pada masing-masing *cluster*.

Untuk *cluster 1 (*A1) yang menjadi *centroid* 1 dan *cluster* 2 (A3) menjadi centroid 2.Setelah ditetapkan nilai keduа*centroid* dilakukan perhitungan *similarity value* menggunakan *Smith-Waterman.*

Sebelum melakukan perhitungan *similarityvalue,* melakukan proses perhitungan$\sum g(n)$.

n=1 g(n)= 1+(1-1)0.1= 1
n=2 g(n)= 1+(2-1)0.1= 1.1
n=3 g(n)= 1+(3-1)0.1= 1.2
n=4 g(n)= 1+(4-1)0.1= 1.3
n=5 g(n)= 1+(5-1)0.1= 1.4

Sehingga didapatkan $\sum g(n)$ = 1+1.1+1.2+1.3+1.4 = 6

Pada proses perhitungan *SimilarityValue*sama dengan proses perhitungan pada sub bab 1.3.3. Berikut perhitungan *Similarity Value* dengan dikurangi $\sum g(n)$dapat dilihat pada tabel 3.3

Tabel 3.3 Proses Perhitungan *Similarity Value* dengan $\sum g(n)$

	A1	A5
A1	-5	-5.4
A2	-5.08	-5.48
A3	-5.32	-5.32
A4	-5.44	-5.28
A5	-5.4	-5

3.3.6 Perhitungan f(v)

Proses perhitungan f(v) dilakukan dengan mengambil salah satu nilai centroid terbesar pada table 3.4. Maka proses perhitungannya sebagai berikut

f(v) = -5 + -5.08 + -5.32 + -5.28 + -5 =-

25.68

Karena Hasil nya f(v) < f(v) max maka iterasi dianggap menjadi iterasi 1

3.3.7 Iterasi ke 2

Dari perhitungan diatas didapatkan *Cluster* 1 = A1, A2 dan *Cluster* 2 = A3, A4 dan A5 dan dilakukan proses perhitungan *Similarity* Score yang didapatkan pada table 3.5.

Tabel 3.4 Perhitungan *Similarity* Score

Atribut	Nilai *Similarity* Score
A1,A2	0.92
A3,A4	0.64
A3,A5	0.68
A4,A5	0.72

Kemudian Menghitung *SumScore*untuk mencari nilai maksimum centroid yang digunakan untuk menghitung proses *Similarity* value. Maka *SumScore* nya adalah

(Si,A1) = ∑ *Similarity* Score (A1,A2) = 0.92

(Si,A2) = ∑ *Similarity* Score (A2,A1) = 0.92

(Si,A3)=∑*Similarity*Score(A3,A4)(A3,A5)=0.64+0.68=1.32

(Si,A4) = ∑ *Similarity* Score(A4,A3)(A4,A5)=0,64+0.72=1.36

(Si,A5)=∑ *Similarity* Score(A5,A3)(A5,A4)=0.68+0.72=1.4

SumScore (Si,Aj) yang terbesar akan dijadikan centroid
Jadi Centroid *Cluster* 1 = A1
Centroid *Cluster* 2 = A5

Lalu Hitung nilai *Similarity* Value nya dengan centroid *cluster* 1 = A1 dan *cluster* 2 = A5. Berikut perhitungan *Similarity Value*dengan dikurangi$\sum g(n)$ dapat dilihat pada tabel 3.5

Tabel 3.5 Perhitungan *Similarity* Valuedengan $\sum g(n)$

	A1	A5
A1	-5	-5.4
A2	-5.08	-5.48
A3	-5.32	-5.32
A4	-5.44	-5.28
A5	-5.4	-5

Proses perhitungan f(v) dilakukan dengan mengambil salah satu nilai centroid terbesar pada table 3.5. Maka proses perhitungannya sebagai berikut

f(v) =-5+-5.08+-5.32 +-5.28+-5 =-
25.68

Karena Hasil nya f(v) < f(v) max maka iterasi dianggap menjadi iterasi 2

3.3.8 Iterasi ke 3

Dari perhitungan diatas didapatkan *Cluster* 1 = A1, A2, A3 dan *Cluster* 2 = A3 dan A4 dan dilakukan proses perhitungan *Similarity* Score yang didapatkan pada table 3.5.

Tabel 3.6 Perhitungan *Similarity* Score

Atribut	Nilai *Similarity* Score
A1,A2	0.92
A1,A3	0.68
A2,A3	0.76
A4,A5	0.72

Kemudian Menghitung *SumScore*untuk mencari nilai maksimum centroid yang digunakan untuk menghitung proses *Similarity* value. Maka *SumScore* nya adalah

(Si,A1) = ∑ *Similarity* Score (A1,A2)(A1,A3)= 1.6

(Si,A2) = ∑ *Similarity* Score (A2,A1)(A2,A3)= 1.68

(Si,A3)=∑*Similarity*Score(A3,A1)(A3,A2)=1.44

(Si,A4) = ∑ *Similarity* Score(A4,A5)= 0.72

(Si,A5) = ∑ *Similarity* Score(A5,A4)=0.72

SumScore (Si,Aj) yang terbesar akan dijadikan centroid
Jadi Centroid *Cluster* 1 = A2
Centroid *Cluster* 2
=A4

Lalu Hitung nilai *Similarity* Value nya dengan centroid *cluster* 1 = A2 dan *cluster* 2 = A4. Berikut perhitungan *Similarity Value* dengan dikurangi$\sum g(n)$dapat dilihat pada tabel 3.5

Tabel 3.7 Perhitungan *Similarity* Value dengan $\sum g(n)$

	A2	A4
A1	-5.08	-5.44
A2	-5	-5.44
A3	-5.24	-5.36
A4	-5.44	-5
A5	-5.48	-5.28

Proses perhitungan f(v) dilakukan dengan mengambil salah satu nilai centroid terbesar pada table 3.5. Maka proses perhitungannya sebagai berikut

f(v) =-5.08+-5+-5.24 +-5+-5.28 =-25.6

Karena Hasil nya f(v) < f(v) max maka iterasi dianggap menjadi iterasi 3

3.3.9 Kesimpulan Perhitungan

Didapatkan *cluster-cluster* berisi anggota sebagai berikut:

- *Cluster* 1 = A1, A2, A3
- *Cluster* 2 = A4, A5

Berdasarkan karakteristik yang dimiliki tiap-tiap anggota dalam sub *cluster* maka disimpulkan :

- *Cluster* 1 = Badak Torgamba
- *Cluster* 2 = Badak Bina

3.3.10 Evaluasi

Perhitungan akurasi yang digunakan adalah *error ratio* yaitu dengan mencari jumlah data yang terklasifikasi tidak tepat dibandingkan dengan total data (dalam %) yang sesuai dengan persamaan 2.9 yaitu:

$$Error = \frac{missclassified}{jumlahdata} \times 100\%$$

Diketahui pada pelabelan data didapatkan pada tabel 3.8

Tabel 3.8 Pelabelan Data

Badak	Klasifikasi Badak
A1	Torgamba
A2	Torgamba
A3	Bina
A4	Bina
A5	Bina

Cluster 1 = badak Torgamba (A1), badak Torgamba (A2)

Cluster 2 = badak Bina (A3), badak Bina (A4), badak Bina (A5)

Sedangkan pada hasil *cluster*ing didapatkan pada tabel 3.9

Tabel 3.9 Hasil *Cluster*ing

Badak	Klasifikasi Badak
A1	Torgamba
A2	Torgamba
A3	Torgamba
A4	Bina
A5	Bina

Cluster 1 = badak Torgamba (A1), badak Torgamba (A2), badak Torgamba (A3)

Cluster 2 = badak Bina (A4), badak Bina (A5)

Missclasified terdapat pada Badak Bina (A3) sehingga didapatkan :

$$Error = \frac{\left(\frac{1}{2} + \frac{0}{3}\right)}{2} = 0.25$$

Dari hasil tersebut diambil nilai *error* yang paling kecil sehingga nilai *error ratio* untuk contoh perhitungan *Pro-Kmeans* adalah 0.25.

3.4 Bahan Pengujian

Bahan yang digunakan adalah data badak Sumatera.

3.4.1 Tujuan Pengujian

Beberapa hal yang menjadi tujuan dari pelaksanaan pengujian terhadap *cluster*ing menggunakan metode *Pro-Kmeans* ini adalah :

a) Memeriksa kesesuaian hasil implementasi data yang telah dibuat sebelumnya dengan hasil perancangan sistem.
b) Menerapkan program yang telah dibuat dengan menggunakan data yang telah disiapkan / sudah ada sebelumnya.
c) Mengevaluasi pengaruh jumlah data terhadap waktu perhitungan proses *clustering*.
d) Mengetahui tingkat wilayah curah berdasarkan kriteria yang sering muncul pada masing-masing *cluster* tersebut.

3.4.2 Kriteria Pengujian.

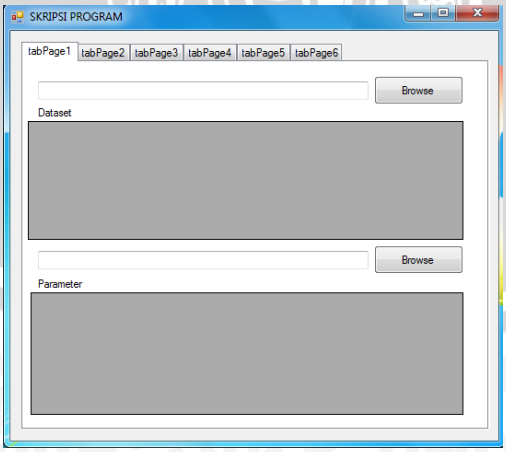
Pengujian yang dilaksanakan meliputi pengujian hasil sistem menggunakan *error ratio*dapat dilihat pada tabel 3.10

Tabel 3.10Kriteria Pengujian

Po	Pe	Wilayah	Error Ratio
1			
.			
.			
.			
K			

3.5 Perancangan Antarmuka

Rancangan antarmuka untuk sistem pada penelitian ini dapat dijelaskan pada gambar 3.13



Gambar 3.13 Rancangan Aplikasi

BAB IV
IMPLEMENTASI DAN PEMBAHASAN

4.1 Lingkungan Implementasi

Lingkungan implementasi yang akan dijelaskan dalam sub-bab ini meliputi lingkungan implementasi perangkat keras dan perangkat lunak.

4.1.1 Lingkungan Implementasi Perangkat Keras

Perangkat keras yang digunakan dalam pembuatan sistem temu kembali informasi berupa teks pada lirik lagu ini meliputi :

1. Processor Intel® Core™ i3-2370M CPU @ 2.40GHz (4CPUs)
2. Memory 2048MB RAM
3. Harddisk 320 GB

4.1.2 Lingkungan Implementasi Perangkat Lunak

Perangkat lunak yang digunakan dalam pembuatan Implementasi Algoritma *Pro-Kmeans* Untuk Pengelompokkan Data Curah Hujan Wilayah Surabaya ini meliputi :

1. Sistem Operasi *Windows 7 Home Basic.*
2. *Microsoft Visual Studio 2010 Express.*
3. *Microsot Office Excel 2010.*

4.2 Implementasi Program

Berdasarkan analisis dan perancangan yang dijelaskan pada bab 3, maka pada sub-bab ini akan dijelaskan implementasi dari perancangan proses yang telah dijabarkan pada bab 3. Sistem ini terdiri dari beberapa proses, yaitu:

a) Proses load data berupa file berekstensi .xls
b) Proses random data ke dalam k *cluster*.
c) Proses pengelompokan DNA badak menggunakan *Pro-Kmeans* yang terdiri dari perhitungan *similarity* score antar pasangan DNA pada satu *cluster* menggunakan algoritma smith-waterman, perhitungan *SumScore* tiap- tiap DNA, Perhitungan penentuan centroid pada tiap- tiap *cluster*,

dan pembentukan *cluster*menggunakan algoritma smith waterman, dan analisa *cluster* menggunakan perhitungan f(v).

4.2.1 Implementasi Load Data

Pada implementasi load data, terdapat dua proses load yaitu load data uji dan load data parameter. Keduanya bertipe .csv.Kemudian data akan ditampilkan kedalam*DataGriedView* yang telah disediakan. Implementasi load data ditampilkan ke dalam source code 4.1

```
private void Browse_Click(object sender, EventArgs e)
        {
            String nilaiBAris;
            String[] nilaiSel;

            using (OpenFileDialog op = new OpenFileDialog())
            {
                BrowseFile(op, textBoxBrowseData);
            }

            if (System.IO.File.Exists(textBoxBrowseData.Text))
            {
                System.IO.StreamReader str = new
StreamReader(textBoxBrowseData.Text);
                //membaca header
                nilaiBAris = str.ReadLine();
                nilaiSel = nilaiBAris.Split(',');

                for (int i = 0; i < nilaiSel.Count(); i++)
                {
                    DataGridViewTextBoxColumn column = new
DataGridViewTextBoxColumn();
                    column.Name = nilaiSel[i];
                    column.HeaderText = nilaiSel[i];
                    DGDataset.Columns.Add(column);
                }
//membaca content
                while (str.Peek() != -1)
                {
                    nilaiBAris = str.ReadLine();
                    nilaiSel = nilaiBAris.Split(',');
                    DGDataset.Rows.Add(nilaiSel);

                }
                str.Close();
            }

            else
            {
```

```
                MessageBox.Show("Tidak ada file yang dipilih");

            }

            for (int y = 0; y < DGDataset.RowCount; y++)
            {
                DGDataset.Rows[y].HeaderCell.Value = (y +
1).ToString();
            }

        }
//load data parameter
        private void button2_Click(object sender, EventArgs e)
        {
            String nilaiBAris;
            String[] nilaiSel;

            using (OpenFileDialog op = new OpenFileDialog())
            {
                BrowseFile(op, textBoxBrowseParameter);
            }

            if
(System.IO.File.Exists(textBoxBrowseParameter.Text))
            {
                System.IO.StreamReader str = new
StreamReader(textBoxBrowseParameter.Text);
                //membaca header
                nilaiBAris = str.ReadLine();
                nilaiSel = nilaiBAris.Split(',');

                for (int i = 0; i < nilaiSel.Count(); i++)
                {
                    DataGridViewTextBoxColumn column = new
DataGridViewTextBoxColumn();
                    column.Name = nilaiSel[i];
                    column.HeaderText = nilaiSel[i];
                    DGParameter.Columns.Add(column);
                }
                //membaca content
                while (str.Peek() != -1)
                {
                    nilaiBAris = str.ReadLine();
                    nilaiSel = nilaiBAris.Split(',');
                    DGParameter.Rows.Add(nilaiSel);

                }
                str.Close();
            }

            else
            {
                MessageBox.Show("Tidak ada file yang dipilih");

            }

  DGParameter.Rows[0].HeaderCell.Value = "A";
```

```
DGParameter.Rows[1].HeaderCell.Value = "T";
DGParameter.Rows[2].HeaderCell.Value = "G";
DGParameter.Rows[3].HeaderCell.Value = "C";

        }
```

Seperti yang ditampilkan pada source code 4.1 ada sebuah proses private void browse_click adalah proses dimana mencari sebuah data atribut yang akan dimasukkan kedalam *grid view*. Pada proses private button2_click adalah proses dimana mencari sebuah data parameter yang akan ditampilkan kedalam *grid view*.

4.2.2 Implementasi Proses random data ke dalam k *cluster*.

Proses pengambilan dataset dilakukan secara random untuk dimasukkan kedalam k *cluster*.Implementasi proses random dataset kedalam k *cluster* ditampilkan ke dalam source code 4.2

```
private void Proses_Click_1(object sender, EventArgs e)
        {
            // inisialisasi list dataset 2 dimensi untuk
menyimpan nilai dari csv dataset
            List<List<string>> dataset = new
List<List<string>>();
            for (int a = 0; a < DGDataset.ColumnCount; a++)
            {
                List<string> anggota = new List<string>();
                dataset.Add(anggota);
            }

            // memasukan nilai datagridview ke dalam List
Dataset
            for (int a = 0; a < dataset.Count; a++)
            {
                for (int b = 0; b < DGDataset.RowCount; b++)
                {
                    string n = Convert.ToString(DGDataset[a,
b].Value);
                    dataset[a].Add(n);
                }
            }

            // menampilkan dataset ke gridview
            DGDataset.ColumnCount = dataset.Count;
            for (int a = 0; a < dataset.Count; a++)
            {
                while (DGDataset.RowCount < dataset[a].Count)
                {
                    DGDataset.Rows.Add();
```

```
                }
                for (int b = 0; b < dataset[a].Count; b++)
                {
                    DGDataset[a, b].Value = dataset[a][b];
                }
            }


            List<List<string>> anggota*Cluster* = new
List<List<string>>();
            double *cluster* =
Convert.ToInt32(dataset[dataset.Count - 1].Max());
            for (int a = 0; a <*cluster*; a++)
            {
                List<string> anggota = new List<string>();
                anggota*Cluster*.Add(anggota);
            }

            for (int a = 0; a < dataset[0].Count; a++)
            {
                int index =
Convert.ToInt32(dataset[dataset.Count - 1][a]);
                anggota*Cluster*[index - 1].Add(dataset[0][a]);
            }

            // menampilkan anggota*Cluster* ke gridview
            dataGridView4.ColumnCount = anggota*Cluster*.Count;
            for (int a = 0; a < anggota*Cluster*.Count; a++)
            {
                while (dataGridView4.RowCount <
anggota*Cluster*[a].Count)
                {
                    dataGridView4.Rows.Add();
                }
                for (int b = 0; b < anggota*Cluster*[a].Count;
b++)
                {
                    dataGridView4[a, b].Value =
anggota*Cluster*[a][b];
                }
            }
            for (int a = 0; a < dataGridView4.ColumnCount; a++)
            {
                dataGridView4.Columns[a].HeaderText = "*Cluster* -
"+(a+1);
            }

            List<List<string>> Anggota*Cluster*Awal = new
List<List<string>>();
            for (int a = 0; a < anggota*Cluster*.Count; a++)
            {
                List<string> anggota = new List<string>();
                Anggota*Cluster*Awal.Add(anggota);
            }

            for (int a = 0; a < anggota*Cluster*.Count; a++)
            {
```

```
                for (int b = 0; b < anggota[Cluster][a].Count;
b++)
                {
AnggotaClusterAwal[a].Add(anggotaCluster[a][b]);
                }
            }
```

Seperti yang ditampilkan pada source code 4.2 prosesrandom data ke dalam k *cluster* dimulai dari memasukan nilai *datagridview* ke dalam *ListDataset*, menampilkan dataset ke *gridview*, lalu menampilkan anggota*Cluster* ke *gridview*.

4.2.3 Implementasi Proses pengelompokan DNA badak menggunakan *Pro-Kmeans.*

4.2.3.1 Perhitungan *similarity* score antar pasangan DNA pada satu *Cluster* menggunakan Algoritma Smith-Waterman.

Proses perhitungan *similarity* score antar pasangan DNA pada satu *cluster* menggunakan algoritma smith waterman sesuai dengan persamaan rumus 2.1 .Implementasi Proses perhitungan *similarity* score antar pasangan DNA pada satu *cluster* menggunakan algoritma smith waterman ditampilkan ke dalam source code 4.3

```
List<List<string>>similarity = new List<List<string>>();
            for (int a = 0; a < 3; a++)
            {
                List<string> anggota = new List<string>();
similarity.Add(anggota);
            }

            for (int a = 0; a < anggotaCluster.Count; a++)
            {
                for (int b = 0; b < anggotaCluster[a].Count; b++)
                {
                    for (int c = 0; c < anggotaCluster[a].Count;
c++)
                    {
                        if (b == c)
                            continue;
                        else
                        {
similarity[0].Add(anggotaCluster[a][b]);
similarity[1].Add(anggotaCluster[a][c]);
                        }
```

```
                    }
                }
            }

            dataGridView5.ColumnCount = similarity.Count;
            for (int a = 0; a <similarity.Count; a++)
            {
                while (dataGridView5.RowCount
<similarity[a].Count)
                {
                    dataGridView5.Rows.Add();
                }
                for (int b = 0; b <similarity[a].Count; b++)
                {
                    dataGridView5[a, b].Value = similarity[a][b];
                }
            }
            for (int a = 0; a < dataGridView5.ColumnCount-1; a++)
            {
                dataGridView5.Columns[a].HeaderText = "Data - " +
(a + 1);
            }
            dataGridView5.Columns[dataGridView5.ColumnCount-
1].HeaderText = "Similarity Score";
// inisialisasi list untuk menyimpan nilai dari csv parameter
            List<List<string>> parameter = new
List<List<string>>();
            for (int a = 0; a < 3; a++)
            {
                List<string> anggota = new List<string>();
                parameter.Add(anggota);
            }

            for (int a = 0; a < DGParameter.ColumnCount; a++)
            {
                for (int b = 0; b < DGParameter.ColumnCount; b++)
                {

parameter[0].Add(DGParameter.Columns[a].HeaderText);

parameter[1].Add(Convert.ToString(DGParameter.Rows[b].HeaderCell.
Value));

parameter[2].Add(Convert.ToString(DGParameter[b, a].Value));
                }
            }


            int index1 = 0, index2 = 0;
            string att1 = "", att2 = "";
            double SimScore = 0;
            List<List<string>> atribut = new
List<List<string>>();
            for (int a = 0; a < 2; a++)
            {
                List<string> anggota = new List<string>();
                atribut.Add(anggota);
```

```
            }
            List<double> atribut_ = new List<double>();
            for (int a = 0; a <similarity[0].Count; a++)
            {
                for (int b = 0; b < dataset[0].Count; b++)
                {
                    if (similarity[0][a] == dataset[0][b])
                    {
                        index1 = b;
                        break;
                    }
                }
                for (int b = 0; b < dataset[0].Count; b++)
                {
                    if (similarity[1][a] == dataset[0][b])
                    {
                        index2 = b;
                        break;
                    }
                }
                for (int b = 1; b < dataset.Count - 1; b++)
                {
                    att1 = dataset[b][index1];
                    att2 = dataset[b][index2];
                    atribut[0].Add(att1);
                    atribut[1].Add(att2);
                }
                for (int b = 0; b < atribut[0].Count; b++)
                {
                    for (int c = 0; c < parameter[0].Count; c++)
                    {
                        if (parameter[0][c] == atribut[0][b] &&
parameter[1][c] == atribut[1][b] &&
Convert.ToInt32(parameter[2][c]) >= 10)
                            atribut_.Add(1);
                    }
                }
                SimScore = atribut_.Sum() / 5;
similarity[2].Add(Convert.ToString(SimScore));
                atribut_.Clear();
                atribut[0].Clear();
                atribut[1].Clear();
            }

            dataGridView5.ColumnCount = similarity.Count;
            for (int a = 0; a <similarity.Count; a++)
            {
                while (dataGridView5.RowCount
<similarity[a].Count)
                {
                    dataGridView5.Rows.Add();
                }
                for (int b = 0; b <similarity[a].Count; b++)
                {
                    dataGridView5[a, b].Value = similarity[a][b];
                }}
```

Seperti yang ditampilkan pada source code 4.3 perhitungan *similarity* score dimulai dari proses menginisialisasi list untuk menyimpan nilai dari csv parameter, dan melakukan proses *similarity* pada setiap atribut.

4.2.3.2 Perhitungan *SumScore* tiap- tiap DNA

Proses perhitungan *SumScore* tiap- tiap DNA sesuai dengan persamaan rumus 2.2 .Implementasi Proses *SumScore* tiap- tiap DNA ditampilkan ke dalam source code 4.4

```
List<List<string>>SumScore = new List<List<string>>();
            for (int a = 0; a < 2; a++)
            {
                List<string> anggota = new List<string>();
SumScore.Add(anggota);
            }
            for (int a = 0; a < dataset[0].Count; a++)
            {
SumScore[0].Add(dataset[0][a]);
            }
            List<double>SumScore_ = new List<double>();

            for (int a = 0; a <SumScore[0].Count; a++)
            {
                for (int b = 0; b <similarity[0].Count; b++)
                {
                    if (similarity[0][b] == SumScore[0][a])
SumScore_.Add(Convert.ToDouble(similarity[2][b]));
                }
SumScore[1].Add(Convert.ToString(SumScore_.Sum()));
SumScore_.Clear();
            }
            dataGridView7.ColumnCount = SumScore.Count;
            for (int a = 0; a <SumScore.Count; a++)
            {
                while (dataGridView7.RowCount
<SumScore[a].Count)
                {
                    dataGridView7.Rows.Add();
                }
                for (int b = 0; b <SumScore[a].Count; b++)
                {
                    dataGridView7[a, b].Value = SumScore[a][b];
                }
            }
            dataGridView7.Columns[0].HeaderText = "Data";
            dataGridView7.Columns[1].HeaderText = "SumScore";
```

Seperti yang ditampilkan pada source code 4.4 dimana proses *similarity* tiap atribut telah diketahui lalu tiap-tiap atribut dilakukan proses *SumScore*untuk mengetahui jumlah nilai masing-masing atribut.

4.2.3.3 Perhitungan penentuan centroid pada tiap- tiap *cluster*

Proses penentuan centroid pada tiap-tiap *cluster* ditampilkan ke dalam source code 4.5.

```
double max = 0;
            int IndexMax = 0;
            List<List<string>> centroid = new
List<List<string>>();
            for (int a = 0; a < 2; a++)
            {
                List<string> anggota = new List<string>();
                centroid.Add(anggota);
            }
            for (int a = 0; a < anggota[Cluster].Count; a++)
            {
                for (int b = 0; b < anggota[Cluster][a].Count;
b++)
                {
                    if (b == 0)
                    {
                        for (int c = 0; c <[SumScore][0].Count;
c++)
                        {
                            if (anggota[Cluster][a][b] ==
[SumScore][0][c])
                            {
                                max =
Convert.ToDouble([SumScore][1][c]);
                                IndexMax = c;
                                break;
                            }
                        }
                    }
                    else
                    {
                        for (int c = 0; c <[SumScore][0].Count;
c++)
                        {
                            if (anggota[Cluster][a][b] ==
[SumScore][0][c] && Convert.ToDouble([SumScore][1][c]) > max)
                            {
                                {
                                    max =
Convert.ToDouble([SumScore][1][c]);
                                    IndexMax = c;
                                    break;
                                }
                            }}
                    }}
                centroid[0].Add([SumScore][0][IndexMax]);
                centroid[1].Add(Convert.ToString(max));
            }

            dataGridView8.ColumnCount = centroid.Count;
            for (int a = 0; a < centroid.Count; a++)
            {
```

```
                while (dataGridView8.RowCount <
centroid[a].Count)
                {
                    dataGridView8.Rows.Add();
                }
                for (int b = 0; b < centroid[a].Count; b++)
                {
                    dataGridView8[a, b].Value = centroid[a][b];
                }
            }
            dataGridView8.Columns[0].HeaderText = "Centroid";
            dataGridView8.Columns[1].HeaderText = "SumScore";
```

Seperti yang ditampilkan pada source code 4.5 penentuan centroid dilakukan setelah mendapatkan hasil dari *SumScore* tersebut. Lalu dicari nilai maksimum dari *SumScore* untuk mendapatkan centroid yang baru.

4.2.3.4 Pembentukan *cluster* menggunakan algoritma smith waterman, dan analisa *cluster* menggunakan perhitungan f(v).

Proses perhitungan Pembentukan *cluster* menggunakan algoritma smith waterman, dan analisa *cluster* menggunakan perhitungan f(v) sesuai dengan persamaan rumus 2.4 ditampilkan ke dalam source code 4.6

```
double po = Convert.ToDouble(textBox1.Text);
            double pe = Convert.ToDouble(textBox3.Text);
            double n_ = dataset[0].Count;
            double gn = 0;
            for (int a = 1; a <= n_; a++)
            {
                gn += (po + (a - 1) * pe);
            }
            label7.Text = "["+gn+"]";
            List<List<string>>similarity_          =          new
List<List<string>>();
            for (int a = 0; a < 3; a++)
            {
                List<string> anggota = new List<string>();
similarity_.Add(anggota);
            }

            for (int a = 0; a < centroid[0].Count; a++)
            {
                for (int b = 0; b < dataset[0].Count; b++)
                {
similarity_[0].Add(dataset[0][b]);
similarity_[1].Add(centroid[0][a]);
                }
            }
            List<List<string>>       _atribut_       =       new
List<List<string>>();
            for (int a = 0; a < 2; a++)
```

```
            {
                List<string> anggota = new List<string>();
                _atribut_.Add(anggota);
            }
            List<double> atribut__ = new List<double>();
            for (int a = 0; a <similarity_[0].Count; a++)
            {
                for (int b = 0; b < dataset[0].Count; b++)
                {
                    if (similarity_[0][a] == dataset[0][b])
                    {
                        index1 = b;
                        break;
                    }
                }
                for (int b = 0; b < dataset[0].Count; b++)
                {
                    if (similarity_[1][a] == dataset[0][b])
                    {
                        index2 = b;
                        break;
                    }
                }
                for (int b = 1; b < dataset.Count - 1; b++)
                {
                    att1 = dataset[b][index1];
                    att2 = dataset[b][index2];
                    _atribut_[0].Add(att1);
                    _atribut_[1].Add(att2);
                }
                for (int b = 0; b < _atribut_[0].Count; b++)
                {
                    for (int c = 0; c < parameter[0].Count; c++)
                    {
                        if (parameter[0][c] == _atribut_[0][b]
&&      parameter[1][c]      ==      _atribut_[1][b]      &&
Convert.ToInt32(parameter[2][c]) >= 10)
                            atribut__.Add(1);
                    }
                }
                SimScore = atribut__.Sum() / 5;
similarity_[2].Add(Convert.ToString(SimScore));
                atribut__.Clear();
                _atribut_[0].Clear();
                _atribut_[1].Clear();
            }


            for (int a = 0; a <similarity_[0].Count; a++)
            {
similarity_[2][a]                                                =
Convert.ToString(Convert.ToDouble(similarity_[2][a]) - gn);
            }

            dataGridView11.ColumnCount = similarity_.Count;
            for (int a = 0; a <similarity_.Count; a++)
            {
```

```
                while                       (dataGridView11.RowCount
<similarity_[a].Count)
                {
                    dataGridView11.Rows.Add();
                }
                for (int b = 0; b <similarity_[a].Count; b++)
                {
                    dataGridView11[a,        b].Value        =
similarity_[a][b];
                }
            }
            dataGridView11.Columns[0].HeaderText = "Data";
            dataGridView11.Columns[1].HeaderText = "Centroid";
            dataGridView11.Columns[2].HeaderText = "Similarity";


            List<double> tempF = new List<double>();
            List<double> SumtempF = new List<double>();
            double maxtempF, fv;
            for (int a = 0; a < dataset[0].Count; a++)
            {
                for (int b = 0; b <similarity_[0].Count; b++)
                {
                    if (similarity_[0][b] == dataset[0][a])

tempF.Add(Convert.ToDouble(similarity_[2][b]));
                }
                maxtempF = tempF.Max();
                SumtempF.Add(maxtempF);
                tempF.Clear();
            }
            fv = SumtempF.Sum();
```

Seperti yang ditampilkan pada source code 4.6 proses penghitungan f(v) melalui proses penghitungan g(n) dimana melakukan perhitungan g(n) melalui proses penginputan Po dan Pe. Proses g(n) digunakan untuk mengitung nilai *similarity* pada proses tersebut. Lalu mencari nilai maksimum dari setiap atribut-atribut yang akan dijumlahkan untuk mencari nilai f(v)

4.2.3.5 Implementasi Perulangan Iterasi.

Proses perulangan iterasi ditampilkan ke dalam source code 4.7.

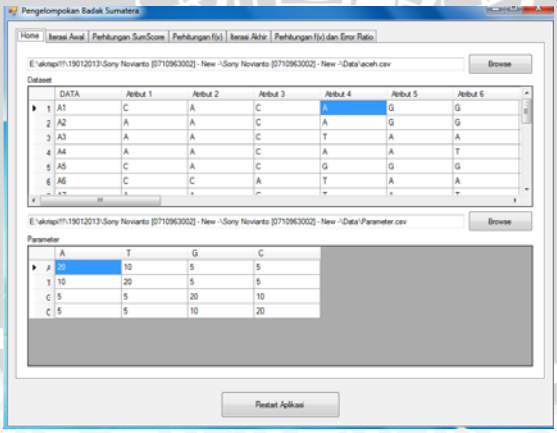
```
double fvMax = 0;
            int iterasi = 1;
            int input_iterasi = Convert.ToInt32(textBox2.Text);
            int BannyakCluster = similarity_[0].Count /
dataset[0].Count;
            List<List<string>> temp_sim = new
List<List<string>>();
            for (int a = 0; a < BannyakCluster + 1; a++)
```

```
        {
            List<string> anggota = new List<string>();
            temp_sim.Add(anggota);
        }
        for (int a = 0; a < dataset[0].Count; a++)
        {
            temp_sim[0].Add(dataset[0][a]);
        }
        int dex = 1;
        for (int a = 0; a <similarity_[0].Count; a++)
        {
            temp_sim[dex].Add(similarity_[2][a]);
            if ((a + 1) % dataset[0].Count == 0)
                dex++;
        }
```

Seperti yang ditampilkan pada source code 4.7 proses perulangan iterasi dilakukan sesuai nilai inputan iterasi yang dimasukkan.

4.2.3.6 Implementasi Antar Muka

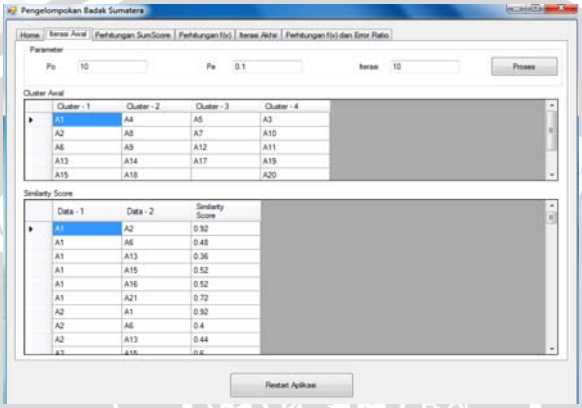
Pada aplikasi *Pro-Kmeans*, terdapat 2 tab control yaitu iterasi awal dan iterasi akhir. Di setiap tab iterasi awal dan iterasi akhir memiliki 6 proses control yaitu proses control *Dataset*, *Similarity Score*,*SumScore*, Penentuan Centroid, *Proses Similarity*dengan Centroid baru yang sudah dikurangi g(n) dan *Penentuan Centroid baru*. Proses control tersebut akan ditunjukan pada Gambar 4.8, Gambar 4.9, Gambar 4.10, Gambar 4.11, Gambar 4.12 dan Gambar 4.13.



Gambar 4.8Tampilan Proses *Dataset.*

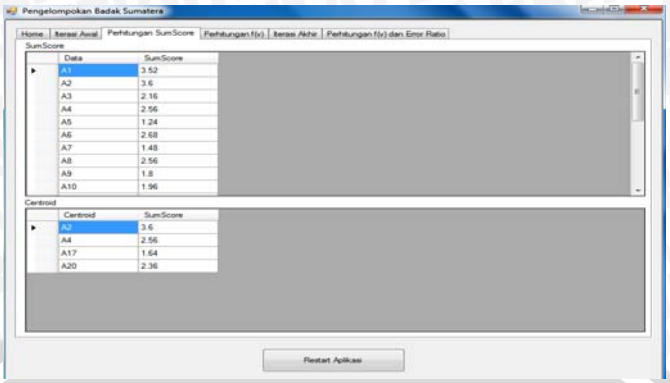
Keterangan :

1. *Browse* adalah tombol yang digunakan untuk memunculkan *browsebox* dimana *user* akan memilih *dataset* yang akan digunakan.
2. *Datagridviewdataset* adalah tempat untuk menampilkan *dataset* yang telah dipilih oleh *user*.



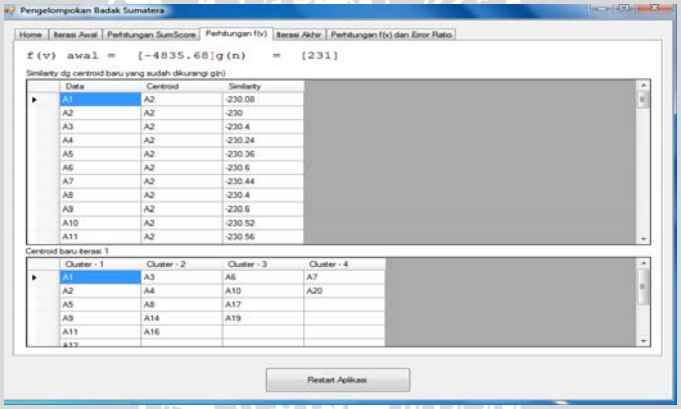
Gambar 4.9Tampilan Proses *Similarity Score*

3. *Textbox*Po adalah sebagai inputan nilai Po pada proses algoritma *Pro-Kmeans*.
4. *Textbox*Pe adalah sebagai inputan nilai Pe pada proses algoritma *Pro-Kmeans*.
5. *Textbox*iterasi adalah sebagai inputan nilai iterasi pada proses algoritma *Pro-Kmeans*.
6. Proses adalah tombol untuk memulai proses perhitungan Algoritma *Pro-Kmeans*.
7. Data *grid viewCluster* awal adalah tempat untuk menampilkan data yang termasuk *cluster* 1 dan *cluster* 2.
8. Data *grid view*adalah tempat untuk menampilkan hasil*Similarity Score.*



*Gambar 4.10 Tampilan*Proses*SumScore dan Penentuan Centroid*

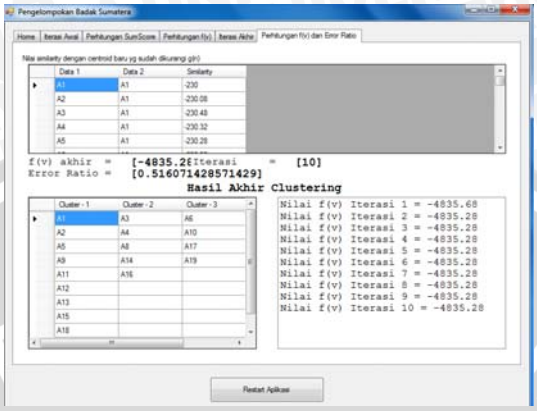
9. Data *grid viewSumScore* adalah tempat untuk menampilkan hasil *SumScore*.
10. Data *grid view* centroid adalah tempat untuk menampilkan hasil centroid yang terbentuk.



Gambar 4.11Tampilan Proses *Similarity* dengan pengurangan g(n) dan penentuan centroid baru

11. Data *grid viewSimilarity* dengan pengurangan g(n) adalah tempat untuk menampilkan hasil nilai *Similarity* dengan pengurangan g(n)
12. Data *grid view*centroid baru iterasi 1 adalah tempat untuk menampilkan hasil centroid yang terbentuk.
13. Label f(v) awal merupakan hasil dari proses perhitungan *Pro-Kmeans* pada iterasi 1.

14. Label g(n) merupakan nilai yang digunakan pada proses *Similarity* dengan pengurangan g(n).



Gambar *4.12 Tampilan*Proses Hasil akhir *Clustering*

15. *Listbox*hasil akhir *cluster*ing adalah hasil akhir dari pengelompokan *cluster* 1 dan *cluster* 2 dengan diketahui f(v) awal sampai akhir.
16. Label *Error Ratio* merupakan hasil error dari proses pengelompokan *cluster*ing tersebut.

4.3 Implementasi Pengujian

4.3.1 Hasil Uji

Hasil pengujian yang dilakukan terhadap 4 wilayah *dataset* akan dijelaskan pada subbab ini. Berdasarkan pengujian yang telah dilakukan, didapatkah hasil sebagai berikut.

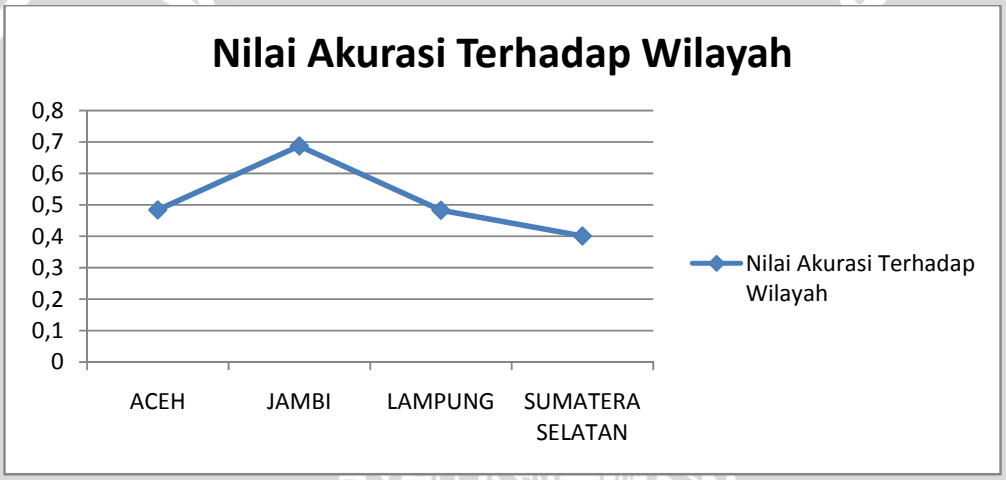
4.3.1.1 Hasil Uji Wilayah

Pada subbab hasil uji wilayah akan dilakukan pengujian menurut wilayah Sumatera yang meliputi Aceh, Jambi, Lampung, Sumatera Selatan. Uji wilayah Sumatera dilakukan dengan menggunakan nilai Po sebesar 50, nilai Pe sebesar 0.7, dan nilai iterasinya sebesar 100. Pengambilan nilai Po, Pe, dan iterasi diambil secara acak.

Tabel 4.1 Akurasi Uji Wilayah menggunakan Pro-Kmeans

Wilayah	ERROR RATIO	AKURASI
ACEH	0.516	0.484
JAMBI	0.3125	0.6875
LAMPUNG	0.517	0.483
SUMATERA SELATAN	0.599	0.401
Rata-Rata		0.513875

Grafik hubungan uji wilayah dengan tingkat akurasi akan ditunjukan pada gambar 4.13



Gambar 4.13 Grafik Akurasi Uji Wilayah

Berdasarkan hasil uji analisa pertama yang dilakukan, didapatkan tingkat kebenaran (akurasi) terbesar proses pengelompokan (*clustering*) badak Sumatera ada pada wilayah Jambi, yaitu sebesar 0.6875. Sedangkan tingkat kebenaran (akurasi) terkecil didapatkan pada wilayah Sumatera Selatan, yaitu sebesar 0.401. Rata-rata tingkat kebenaran (akurasi) yang dihasilkan oleh system pengelompokan badak Sumatera menggunakan Pro-Kmeans adalah 0.513875.

Hal-hal yang menyebabkan terjadinya perbedaan tingkat kebenaran pada tiap-tiap wilayah adalah perbedaan DNA badak dan jumlah DNA badak yang terdapat pada masing-masing wilayah, sehingga mempengaruhi tingkat *similarity* yang dihitung menggunakan algoritma*Smith-Waterman*.

Kemudian dapat disimpulkan bahwa uji analisa pertama mengalami kenaikan dan hal yang mempengaruhi perubahan nilai akurasi adalah tingkat *similarity* yang didapatkan dan perhitungan menggunakan algoritma *Smith-Waterman.*

4.3.1.2 Hasil uji Po

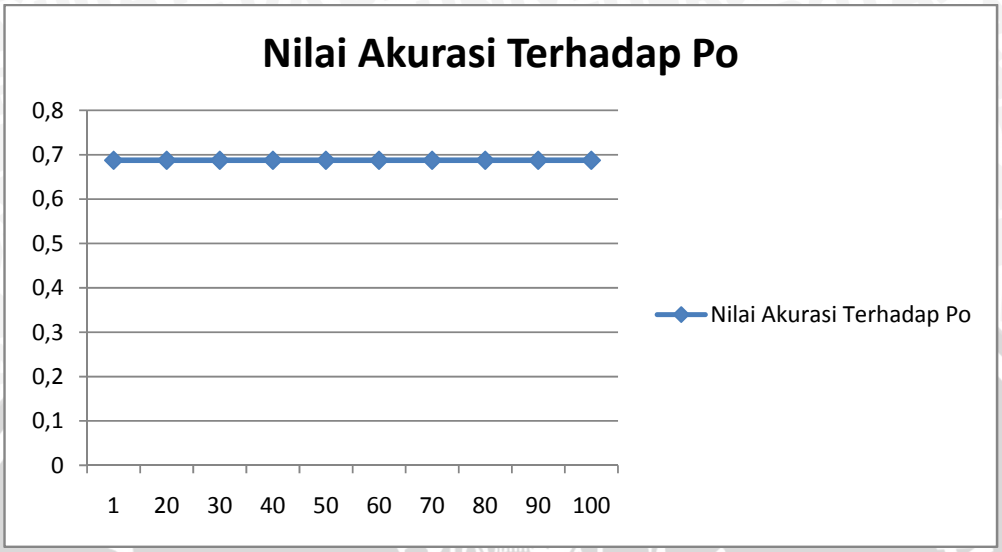
Pada subbab hasil uji Po dilakukan dengan memilih tingkat kebenaran (akurasi) wilayah yang terbesar, yaitu Jambi. Pada pengujian Po akan dilakukan pengujian pengaruh nilai Po terhadap nilai akurasi. Uji Po dilakukan dengan cara mengubah nilai Po dari 1 sampai nilai 100. Untuk parameter Pe dan dan iterasi di-*set* dengan nilai 0.7 untuk Pe dan nilai 100 untuk iterasi. Pengambilan nilai Po, Pe, dan iterasi diambil secara acak. Pengujian Po dilakukan untuk keempat dataset dimana dataset diuji dengan menggunakan rumus *Pro-Kmeans.*

Tabel 4.2 Akurasi uji Po menggunakan*Pro-Kmeans*

Po	Pe	Iterasi	AKURASI
1	0.7	100	0,6875
2	0.7	100	0,6875
4	0.7	100	0,6875
5	0.7	100	0,6875
....	....	....	....
95	0.7	100	0,6875
96	0.7	100	0,6875
99	0.7	100	0,6875
Rata-Rata			0.6875

Tabel 4.2 merupakan hasil pengujian Po dengan menggunakan algoritma *Pro-Kmeans*. Dari hasil pengujian tersebut dapat diketahui bahwa nilai akurasinya adalah 0.6875. Nilai akurasi sama dikarenakan Po mempengaruhi proses berjalannya nilai *similarity score* untuk mendapatkan nilai f(v). Rata-rata nilai akurasi untuk uji Po dengan menggunakan *Pro-Kmeans* adalah 0.6875.

Grafik hubungan nilai Po dengan tingkat akurasi akan ditunjukan pada gambar 4.14



Gambar 4.14 Grafik Akurasi Uji Po

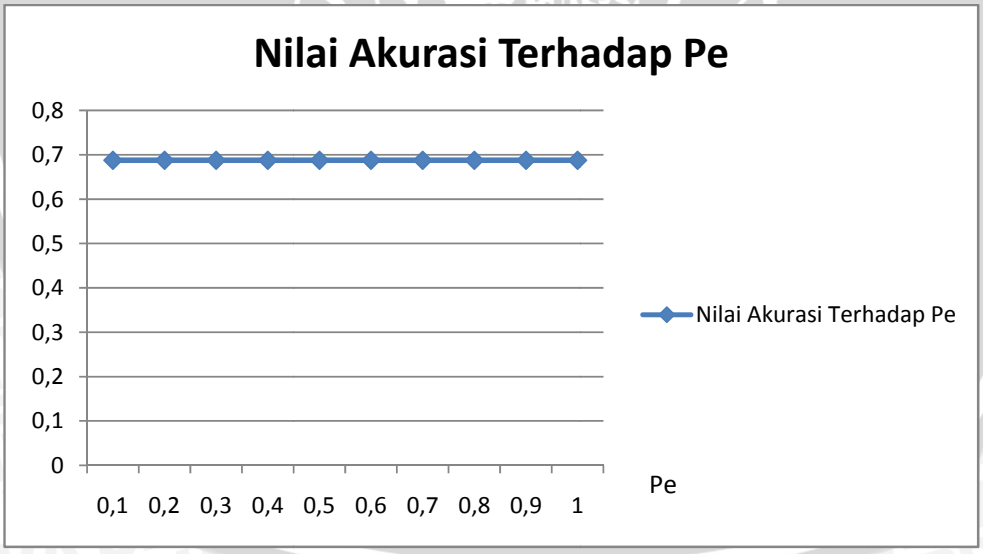
4.3.1.3 Hasil uji Pe

Pada subbab hasil uji Pe dilakukan dengan memilih tingkat kebenaran (akurasi) wilayah yang terbesar, yaitu Jambi. Pada pengujian Pe akan dilakukan pengujian pengaruh Pe terhadap nilai akurasi. Uji Pe dilakukan dengan cara mengubah nilai Pe mulai dari 0,1 sampai nilai Pe maksimal yaitu 1. Untuk parameter Po dan iterasi di-*set* dengan nilai maksimal yaitu nilai 100 untuk Po dan nilai 100 untuk iterasi. Pengambilan nilai Po, Pe, dan iterasi diambil secara acak. Pengujian Pe dilakukan dengan menggunakan rumus *Pro-Kmeans*.

Tabel 4.3 Akurasi uji Pe menggunakan *Pro-Kmeans*

Po	Pe	Iterasi	AKURASI
100	0, 1	100	0,6875
100	0, 2	100	0,6875
....	....	....	....
100	0,4	100	0,6875
100	0,5	100	0,6875
....	....	....	....
100	1	100	0,6875
RATA-RATA			0,6875

Tabel 4.3 merupakan hasil pengujian Pe dengan menggunakan algoritma *Pro-Kmeans*. Dari hasil pengujian tersebut dapat diketahui bahwa nilai akurasinya adalah 0.6875. Nilai akurasi sama dikarenakan Pe mempengaruhi proses berjalannya nilai *similarity score* untuk mendapatkan nilai f(v). Rata-rata nilai akurasi untuk uji Pe dengan menggunakan *Pro-Kmeans* adalah 0.6875.



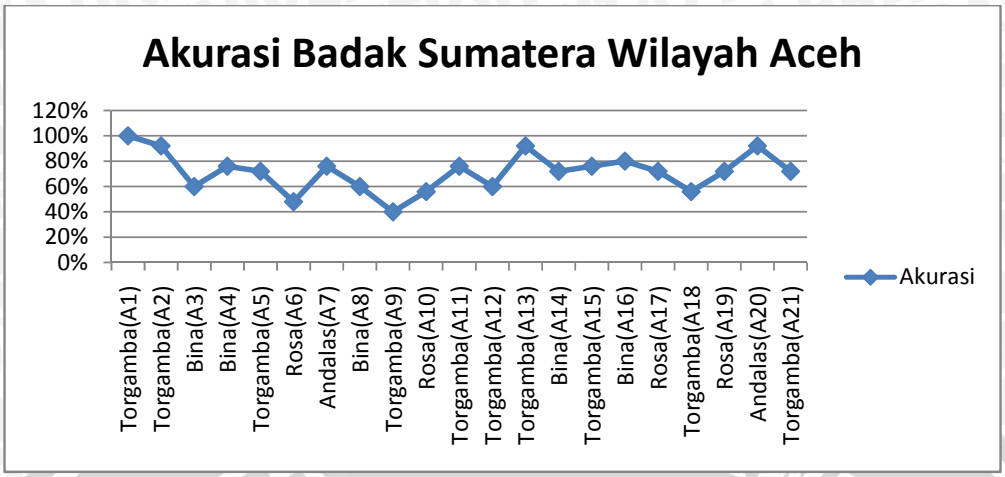
Gambar 4.15Grafik Akurasi Uji Pe

4.3.1.4 Hasil Uji Sistem

Pada subbab uji sistem meliputi 4 wilayah, yaitu Aceh, Jambi, Sumatera Selatan, dan Lampung. Dalam pengujian ini data dari hasil sistem akan dibandingkan dengan DNA asli badak Sumatera untuk menghitung nilai kebenaran (akurasi) dari sistem tersebut.Rumus perhitungan akurasi untuk analisa hasil uji sistem seperti tertera pada rumus 2.1.

Tabel 4.4Wilayah Aceh

Data Badak	Data Sistem	SpesiesDNA Badak Sumatera	Akurasi
A1	Torgamba	Torgamba	100%
A2	Torgamba	Torgamba	92%
A3	Bina	Bina	60%
A4	Bina	Bina	76%
A5	Torgamba	Torgamba	72%
A6	Rosa	Rosa	48%
A7	Andalas	Andalas	76%
A8	Bina	Bina	60%
A9	Torgamba	Torgamba	40%
A10	Rosa	Rosa	56%
A11	Torgamba	Torgamba	76%
A12	Torgamba	Torgamba	60%
A13	Torgamba	Torgamba	92%
A14	Bina	Bina	72%
A15	Torgamba	Torgamba	76%
A16	Bina	Bina	80%
A17	Rosa	Rosa	72%
A18	Torgamba	Torgamba	56%
A19	Rosa	Rosa	72%
A20	Andalas	Andalas	92%
A21	Torgamba	Torgamba	72%
Nilai rata-rata			71.42%



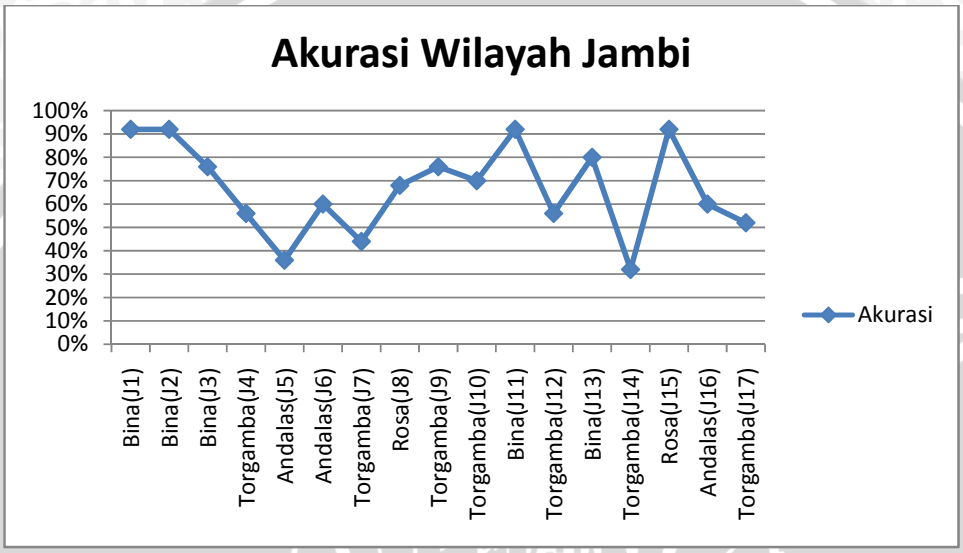
Gambar 4.16 Akurasi Badak Sumatera Wilayah Aceh

Pada tabel 4.4 dan grafik 4.16 menunjukkan bahwa badak Torgamba lebih dominan daripada badak yang lain, dapat dilihat bahwa badak Torgamba memiliki jumlah populasi sebanyak 10, badak Bina memiliki populasi sebanyak 5, badak Rosa memiliki populasi sebanyak 4, dan badak Andalas memiliki populasi sebanyak 2. Pada wilayah Aceh memiliki tingkat akurasi rata-rata 71.42%.

Tabel 4.5 Wilayah Jambi

Data Badak	Data Sistem	Spesies DNA Badak Sumatera	Akurasi
J1	Bina	Bina	92%
J2	Bina	Bina	92%
J3	Bina	Bina	76%
J4	Torgamba	Torgamba	56%
J5	Andalas	Andalas	36%
J6	Andalas	Andalas	60%
J7	Torgamba	Torgamba	44%
J8	Rosa	Rosa	68%
J9	Torgamba	Torgamba	76%
J10	Torgamba	Torgamba	70%
J11	Bina	Bina	92%
J12	Torgamba	Torgamba	56%
J13	Bina	Bina	80%

J14	Torgamba	Torgamba	32%
J15	Rosa	Rosa	92%
J16	Andalas	Andalas	60%
J17	Torgamba	Torgamba	52%
Nilai Rata-Rata			66.70 %



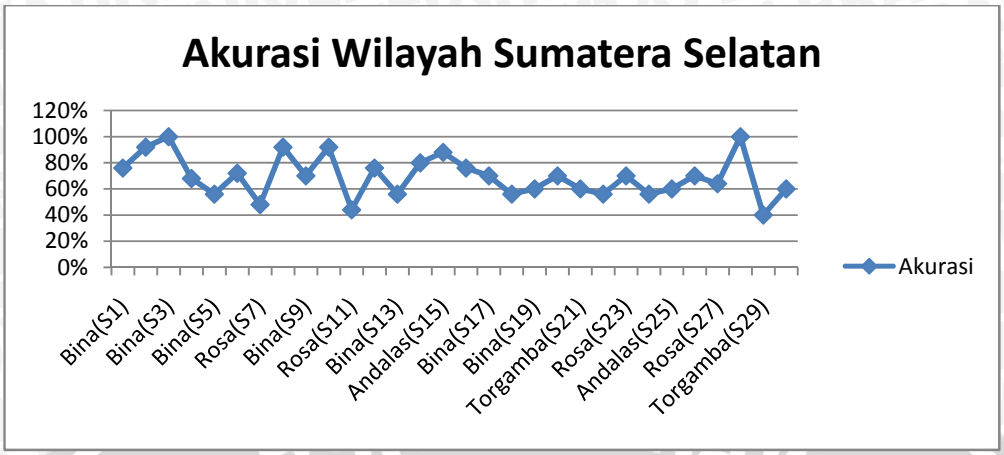
Gambar 4.17 Akurasi Wilayah Jambi

Pada tabel 4.5 dan grafik 4.17 menunjukkan bahwa badak Torgamba lebih dominan daripada badak yang lain, dapat dilihat bahwa badak Torgamba memiliki jumlah populasi sebanyak 7, badak Bina memiliki populasi sebanyak 5, badak Rosa memiliki populasi sebanyak 2, dan badak Andalas memiliki populasi sebanyak 3.Pada wilayah Jambi memiliki tingkat akurasi rata-rata 66.70 %.

Tabel 4.6 Wilayah Sumatera Selatan

Data Badak	Data Sistem	Spesies DNA Badak Sumatera	Akurasi
S1	Bina	Bina	76%
S2	Bina	Bina	92%
S3	Bina	Bina	100%
S4	Torgamba	Torgamba	68%
S5	Bina	Bina	56%
S6	Torgamba	Torgamba	72%

S7	Rosa	Rosa	48%
S8	Andalas	Andalas	92%
S9	Bina	Bina	70%
S10	Andalas	Andalas	92%
S11	Rosa	Rosa	44%
S12	Andalas	Andalas	76%
S13	Bina	Bina	56%
S14	Torgamba	Torgamba	80%
S15	Andalas	Andalas	88%
S16	Rosa	Rosa	76%
S17	Bina	Bina	70%
S18	Andalas	Andalas	56%
S19	Bina	Bina	60%
S20	Torgamba	Torgamba	70%
S21	Torgamba	Torgamba	60%
S22	Bina	Bina	56%
S23	Rosa	Rosa	70%
S24	Andalas	Andalas	56%
S25	Andalas	Andalas	60%
S26	Bina	Bina	70%
S27	Rosa	Rosa	64%
S28	Bina	Bina	100%
S29	Torgamba	Torgamba	40%
S30	Andalas	Andalas	60%
Nilai Rata-Rata			69.26%



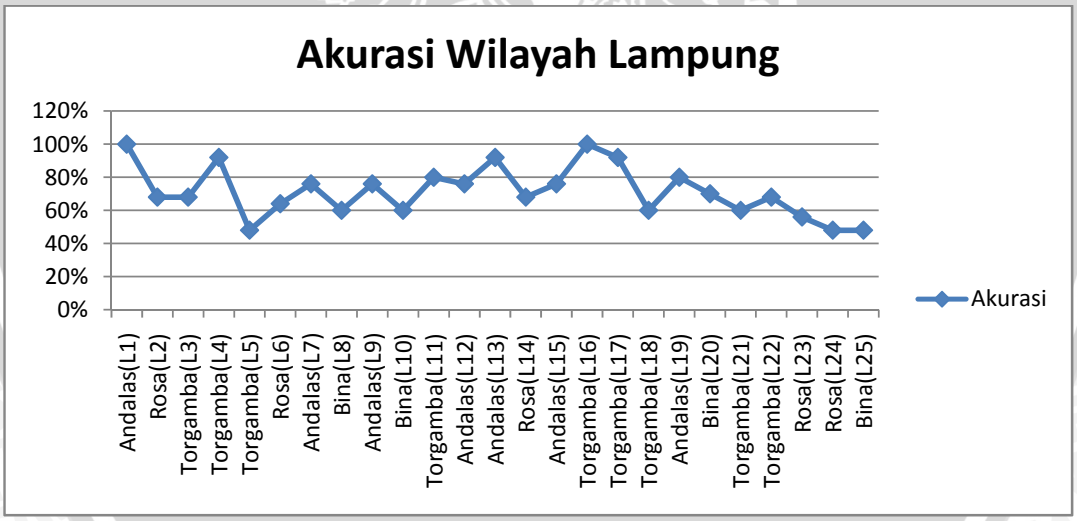
Gambar 4.18 Akurasi Wilayah Sumatera Selatan

Pada tabel 4.6 dan grafik 4.18 menunjukkan bahwa badak Bina lebih dominan daripada badak yang lain, dapat dilihat bahwa badak Torgamba memiliki jumlah populasi sebanyak 6, badak Bina memiliki populasi sebanyak 11, badak Rosa memiliki populasi sebanyak 5, dan badak Andalas memiliki populasi sebanyak 8. Pada wilayah Sumatera Selatan memiliki tingkat akurasi rata-rata 69.26%.

Tabel 4.7 Wilayah Lampung

Data Badak	Data Sistem	Spesies DNA Badak Sumatera	Akurasi
L1	Andalas	Andalas	100%
L2	Rosa	Rosa	68%
L3	Torgamba	Torgamba	68%
L4	Torgamba	Torgamba	92%
L5	Torgamba	Torgamba	48%
L6	Rosa	Rosa	64%
L7	Andalas	Andalas	76%
L8	Bina	Bina	60%
L9	Andalas	Andalas	76%
L10	Bina	Bina	60%
L11	Torgamba	Torgamba	80%
L12	Andalas	Andalas	76%
L13	Andalas	Andalas	92%

L14	Rosa	Rosa	68%
L15	Andalas	Andalas	76%
L16	Torgamba	Torgamba	100%
L17	Torgamba	Torgamba	92%
L18	Torgamba	Torgamba	60%
L19	Andalas	Andalas	80%
L20	Bina	Bina	70%
L21	Torgamba	Torgamba	60%
L22	Torgamba	Torgamba	68%
L23	Rosa	Rosa	56%
L24	Rosa	Rosa	48%
L25	Bina	Bina	48%
Nilai rata-rata			71.44%



Gambar 4.19 Akurasi Wilayah Lampung

Pada tabel 4.7 dan grafik 4.19 menunjukkan bahwa badak Torgamba lebih dominan daripada badak yang lain, dapat dilihat bahwa badak Torgamba memiliki jumlah populasi sebanyak 9, badak Bina memiliki populasi sebanyak 4, badak Rosa memiliki populasi sebanyak 5, dan badak Andalas memiliki populasi sebanyak 6. Pada wilayah Aceh memiliki tingkat akurasi rata-rata 71.44%.

4.3.2 Analisa Hasil

Berdasarkan hasil uji pada subbab 4.3.1, dapat dilakukan analisa terhadap rincian hasil yang telah didapatkan.

4.3.2.1 Analisa hasil uji Wilayah

Berdasarkan pada tabel 4.1 dan gambar 4.7 nilai akurasi pada analisa hasil uji wilayah cenderung berbeda. Nilai akurasinya adalah 0.513875.

4.3.2.2 Analisa hasil uji Po

Berdasarkan pada tabel 4.2 dan gambar 4.8 nilai akurasi pada analisa hasil uji Po cenderung tetap seiring dengan berubahnya nilai Po.Nilai akurasinya adalah 0.6875.

4.3.2.3 Analisa hasil uji Pe

Berdasarkan pada tabel 4.3 dan gambar 4.9 nilai akurasi pada analisa hasil uji Pe cenderung tetap pada nilai akurasi seiring dengan berubahnya nilai Pe. Nilai akurasinya adalah 0.6875.

4.3.2.4 Analisa hasil uji Sistem

Berdasarkan subbab 4.3.1.4 pada analisa uji sistem masing-masing wilayah memiliki tingkat akurasi yang berbeda dapat dilihat bahwa wilayah Aceh memiliki tingkat akurasi 71.42%, wilayah Jambi memiliki tingkat akurasi 66.70%, wilayah Sumatera Selatan memiliki tingkat akurasi 69.26%, wilayah Lampung memiliki tingkat akurasi 71.44%.

BAB V
PENUTUP

5.1 Kesimpulan

Dari hasil uji dan analisis yang telah dilakukan dapat diambil beberapa kesimpulan sebagai berikut:

Telah diimplementasi algoritima *Pro-Kmeans* untuk melakukan pengelompokan badak Sumatera, dalam bentuk aplikasi *desktop* berbasis C#.Hitung nilai *Similarity Score* dengan *algoritma Smith-Waterman*.Hitung *SumScore*untuk mengetahui *centroid* yang baru.Kemudian hitung nilai *Similarity* dari *centroid* yang baru dan hasil dari nilai *Similarity* dikurangi dengan ∑g(n). Lalu,hitung nilai f(v) dan hitung nilai akurasi dari *cluster* yang terbentuk.

a. Tingkat akurasi untuk uji wilayah yang dibentuk algoritma *Pro-Kmeans* dengan data uji memiliki nilai akurasi rata-rata0.513875. Hal-hal yang menyebabkan terjadinya perbedaan tingkat kebenaran pada tiap-tiap wilayah adalah perbedaan DNA badak dan jumlah DNA badak yang terdapat pada masing-masing wilayah, sehingga mempengaruhi tingkat*similarity* yang dihitung menggunakan algoritma *Smith-Waterman*.
b. Nilai akurasi untuk Po yang dibentuk algoritma *Pro-Kmeans* dalah 0.6875 dengan hasil akurasi rata-rata 0.6875.Nilai akurasi sama dikarenakan Po mempengaruhi proses berjalannya nilai *similarity score* untuk mendapatkan nilai f(v).
c. Nilai akurasi untuk Pe yang dibentuk algoritma *Pro-Kmeans* adalah 0.6875dengan hasil akurasi rata-rata 0.6875.Nilai akurasi sama dikarenakan Pe mempengaruhi proses berjalannya nilai *similarity score* untuk mendapatkan nilai f(v).
d. Nilai akurasi untuk analisa uji sistem masing-masing wilayah memiliki tingkat akurasi yang berbeda dapat dilihat bahwa wilayah Aceh memiliki tingkat akurasi 71.42%, wilayah Jambi memiliki tingkat akurasi 66.70%, wilayah Sumatera Selatan memiliki tingkat akurasi 69.26%, wilayah Lampung memiliki tingkat akurasi 71.44%.

5.2 Saran

Beberapa saran yang dapat disampaikan untuk penelitian selanjutnya yaitu untuk meningkatkan akurasi dapat ditambahkan atribut jenis-jenis badak yang lain yang dapat menjadi bahan pendukung dalam pengelompokan badak Sumatera.