Implementasi Metode *Density-Based Spatial Clustering of Application with Noise* (DBSCAN) Dalam *Clustering* Titik Panas

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Reno Septa Pradana
125150207111051

PROGRAM STUDI TEKNIK INFORMATIKA
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
TAHUN
2017

PENGESAHAN

JUDUL SKRIPSI

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Reno Septa Pradana
NIM: 125150207111051

Skripsi ini telah diuji dan dinyatakan lulus pada
29 Desember 2016
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Muhammad Tanzil Furqon S.Kom, M.CompSc	Randy Cahya W, S.ST., M.Kom
NIP: 19820930 200801 1 004	NIK: 201405 880206 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 1 Desember 2016

Reno Septa Pradana

NIM: 125150207111051

KATA PENGANTAR

Puji dan syukur penulis panjatkan atas kehadirat Allah SWT yang telah melimpahkan segala rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan penulisan skripsi yang berjudul "Implementasi Metode *Density-Based Spatial Clustering of Application with Noise* (DBSCAN) Dalam *Clustering* Titik Panas", yang diajukan untuk menempuh Ujian Akhir Sarjana Program Strata Satu Jurusan Informatika.

Selesainya penulisan skripsi ini tidak terlepas dari peran serta berbagai pihak. Oleh karena itu, pada kesempatan ini penulis menyampaikan terima kasih yang sebesar-besarnya kepada:

1. Bapak Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D, selaku Dekan Fakultas Ilmu Komputer.
2. Bapak Agus Wahyu Widodo, S.T, M.Cs, selaku Ketua Program Studi Informatika / Ilmu Komputer.
3. Bapak Muhammad Tanzil Furqon S.Kom, M.CompSc dan Bapak Randy Cahya W, S.ST., M.Kom selaku dosen pembimbing penulis yang dengan sabar memberikan kritik, saran, serta arahan yang baik dalam proses pengerjaan skripsi ini.
4. Seluruh Bapak dan Ibu Dosen yang mengajar di Program Studi Informatika / Ilmu Komputer Fakultas Ilmu Komputer Universitas Brawijaya Malang.
5. Staff Akademik di Program Studi Informatika / Ilmu Komputer Fakultas Ilmu Komputer Universitas Brawijaya Malang.

Semoga Allah SWT melimpahkan rahmat dan karunia-Nya kepada semua pihak yang telah membantu penulis dalam menyelesaikan penulisan skripsi ini.

Penulis menyadari bahwa dalam penyusunan skripsi ini masih banyak terdapat kekurangan. Untuk itu penulis menyampaikan permohonan maaf sebelumnya, serta sangat diharapkan kritik dan saran yang bersifat membangun dalam penyempurnaan di masa mendatang.

Malang, 1 Desember 2016

Penulis

renn.pradana@gmail.com

ABSTRAK

Reno Septa Pradana. 2016 : Implementasi Metode *Density-Based Spatial Clustering of Application with Noise* (DBSCAN) Dalam *Clustering* Titik Panas. Skripsi Program Studi Informatika / Ilmu Komputer, Fakultas Ilmu Komputer, Universitas Brawijaya. Dosen Pembimbing : Muhammad Tanzil Furqon, S.Kom, M.CompSc dan Randy Cahya W, S.ST., M.Kom.

Kebakaran hutan yang tidak terkendali adalah salah satu bencana yang dapat mengakibatkan kerugian materi sampai kehilangan nyawa. Kebakaran hutan yang berskala besar cukup sulit untuk dipadamkan, kadang-kadang membutuhkan waktu hingga berminggu-minggu agar semua titik api bisa padam. Kebakaran seharusnya dapat dicegah dengan cara mengetahui indikasi-indikasi terjadinya kebakaran. Salah satu indikasi terjadinya kebakaran adalah titik panas. Titik panas perlu dikelompokkan berdasarkan kemiripan sehingga diperoleh suatu informasi berupa tingkat potensi terjadinya kebakaran. DBSCAN merupakan salah satu metode *clustering* yang paling populer dan dapat digunakan untuk menyelesaikan permasalahan ini. Algoritma DBSCAN dapat diimplementasikan untuk mengelompokkan titik panas ke dalam 2 tingkat potensi kebakaran. Selain itu algoritma DBSCAN menghasilkan hasil yang cukup bagus dimana hasil dari evaluasi menggunakan *Silhouette Coefficient* memberikan nilai tertinggi yaitu 0.848423572 untuk eps 0.25 dan minpts 2.

Kata Kunci : Titik panas, *data mining, clustering,* DBSCAN, *Silhouette Coefficient*.

ABSTRACT

Reno Septa Pradana. 2016 : Implementation Method of Density-Based Spatial Clustering of Application with Noise (DBSCAN) in Clustering Hotspots. Essay Studies Program Informatics / Computer Science, Faculty of Computer Science, University of Brawijaya. Supervisors : Muhammad Tanzil Furqon, S.Kom, M.CompSc dan Randy Cahya W, S.ST., M.Kom.

Uncontrolled forest fires is one disaster that could result in material loss to life loss. Large-scale forest fires is quite difficult to extinguish, sometimes take up to weeks for all fires can be extinguished. The fire could have been prevented by knowing indications of fire. One indication of the fire is hotspots. Hotspots need to be grouped by similarity to obtain an important information. DBSCAN is one of the most popular clustering methods and can be used to solve this problem. DBSCAN algorithm can be implemented to classify hot spots into a two-level fire potential. In addition DBSCAN algorithm generates pretty good results where the results of the evaluation using the Silhouette Coefficient provide the highest value 0.848423572 for eps 0.25 and minpts 2.

Keywords : Hotspot, *data mining, clustering,* DBSCAN, *Silhouette Coefficient*.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR SOURCECODE

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Kebakaran hutan yang tidak terkendali adalah salah satu bencana yang dapat mengakibatkan kerugian materi sampai kehilangan nyawa. Kebakaran hutan yang berskala besar cukup sulit untuk dipadamkan, kadang-kadang membutuhkan waktu hingga berminggu-minggu agar semua titik api bisa padam. Di Indonesia sendiri pernah terjadi kebakaran hutan dan lahan yang terjadi pada Juni hingga Oktober 2015. Kebakaran hutan dan lahan tersebut mengakibatkan kerugian finansial hingga Rp 221 triliun (Putri, 2015, para. 1). Seperti diketahui, pada periode tersebut terjadi kebakaran hutan dan lahan di Jambi, Riau, Sumatera Selatan, Kalimantan Selatan, Kalimantan Barat dan Kalimantan Timur. Penyebabnya yakni kesengajaan membakar, pembukaan lahan baru oleh sebagian masyarakat, buruknya pengelolaan ekosistem rawa gambut, musim kemarau panjang akibat *El Nino* serta lemahnya pengawasan (Putri, 2015, para. 3). Kerugian ini setara dengan 1,5 persen Produk Domestik Bruto nasional, artinya Kebakaran hutan dan lahan menghambat laju pembangunan (Putri, 2015 para. 4). Untuk menanggulanginya, BNPB mengeluarkan Rp 720 miliar untuk pemadaman kebakaran. Biaya tersebut di luar dari dana yang dikeluarkan Kementerian Lingkungan Hidup dan Kehutanan (KLHK), Kementerian Pekerjaan Umum dan Perumahan Rakyat (Kemenpupera) dan Kementerian Kesehatan. Asap yang ditimbulkan oleh kebakaran hutan berdampak langsung pada kesehatan, khususnya gangguan saluran pernapasan (Putri, 2015, para. 5). Terdata 24 orang meninggal dunia, lebih dari 600 ribu jiwa terjangkit Infeksi Saluran Pernafasan Atas (ISPA), 60 juta jiwa terpapar asap dan sebanyak 2,61 juta hektare hutan dan lahan terbakar (Putri, 2015, para. 8).

Dari keterangan diatas, apabila kebakaran sudah terjadi maka kerugian yang ditimbulkan untuk negara sangatlah besar. Seharusnya kebakaran dapat dicegah agar tidak menimbulkan kerugian. Untuk melakukan pencegahan tersebut, diperlukan informasi mengenai indikasi terjadinya kebakaran. Indikasi terjadinya kebakaran hutan dapat diketahui melalui titik panas. Titik panas memiliki kemungkinan untuk menggerombol dalam ruang secara alami mengikuti hukum Geografi 1 Tobbler yaitu semuanya terkait dengan segala sesuatu yang lain, tetapi hal-hal yang dekat lebih terkait daripada hal-hal yang jauh. *Clustering* merupakan salah satu metode *data mining* yang cocok digunakan untuk menganalisis data spasial persebaran titik panas yang berukuran besar. *Clustering* merupakan proses pengelompokkan kumpulan objek ke dalam kelas-kelas (*clusters*) sehingga objek objek dalam satu *cluster* memiliki kemiripan yang tinggi tetapi tidak mirip terhadap objek dari *cluster* lain (Usman, 2014). Salah satu metode *clustering* yang populer adalah DBSCAN.

Penelitian sebelumnya mengenai DBSCAN sudah pernah dilakukan oleh Devi, et al. dengan tujuan penentuan pelanggang potensial melalui kelas yang dihasilkan tiap *cluster*. Penelitian tersebut menghasilkan 4 *cluster* dengan *Silhouette*

Coefficient sebesar 0.900861. Nilai *Silhouette Coefficient* yang dihasilkan lebih besar dari 0 dan mendekati 1 yang mana berarti bahwa jumlah cluster yang dihasilkan pada proses *clustering* sudah optimal (Devi, et al., 2015).

Penelitian yang lain dilakukan oleh Usman, Muhammad yaitu "*Spatial Clustering* Berbasis Densitas Untuk Persebaran Titik Panas Sebagai Indikator Kebakaran Hutan Dan Lahan Gambut Di Sumatera". Penelitian tersebut mengelompokkan data titik panas di Sumatera pada tahun 2002 dan 2013 menggunakan algoritma DBSCAN. Dengan data pada tahun 2002, algoritma ini membentuk 52 cluster dengan SSE 0.26, sedangkan pada tahun 2013 algoritma ini menghasilkan 109 cluster dengan SSE 0.387 (Usman, 2014).

Penelitian ini menerapkan metode *clustering* untuk mengelompokkan data persebaran titik panas di Indonesia pada periode Juni hingga Agustus 2015. Tujuan dari penelitian ini adalah untuk membentuk *cluster* titik panas menggunakan algoritma DBSCAN berdasarkan tingkat potensi terjadinya kebakaran dan melakukan evaluasi hasil *clustering* menggunakan metode *Silhouette Coefficient*. Algoritma DBSCAN dipilih dikarenakan mampu membentuk *cluster* yang lebih fleksibel, tidak perlu menentukan jumlah *cluster*, dan juga mampu memisahkan data yang dianggap *noise* atau tidak memiliki kemiripan dengan data lain.

1.2 Rumusan Masalah

Dari latar belakang tersebut, maka dapat dirumuskan pokok permasalahan :

1. Bagaimana mengimplementasikan algoritma DBSCAN dalam *clustering* titik panas berdasarkan tingkat potensi terjadinya kebakaran?
2. Bagaimana kualitas hasil implementasi algoritma DBSCAN dalam *clustering* titik panas menggunakan metode *Silhouette Coefficient*?

1.3 Tujuan

Tujuan yang ingin dicapai adalah :

1. Menerapkan metode *Density-Based Spatial Clustering of Applications with Noise* (DBSCAN) dalam *clustering* titik panas berdasarkan tingkat terjadinya kebakaran.
2. Melakukan evaluasi terhadap hasil *cluster* menggunakan metode *Silhouette Coeficient*.

1.4 Manfaat

Manfaat yang dapat diambil dari penelitian ini adalah dapat menambah pengetahuan dan wawasan serta dapat mengaplikasikan teori yang telah diperoleh selama masa perkuliahan. Selain itu penelitian ini diharapkan mampu membantu pihak yang bersangkutan dalam memperoleh informasi tingkat potensi terjadinya kebakaran sehingga bisa diprioritaskan mana yang perlu dilakukan penanggulangan terlebih dahulu.

1.5 Batasan Masalah

Batasan Masalah yang tersedia adalah :

1. Data yang digunakan dalam penelitian ini adalah data persebaran titik panas di Indonesia yang diperoleh dari *National Aeronautics and Space Administration* (NASA) pada periode Juni hingga Agustus 2015.
2. Metode evaluasi hasil *clustering* yang digunakan adalah *Silhouette Coefficient.*

1.6 Sistematika Pembahasan

Untuk mencapai tujuan yang diharapkan, maka sistematika penulisan yang disusun adalah sebagai berikut:

1. BAB I PENDAHULUAN

   Dalam bab ini diuraikan latar belakang penelitian, tujuan penelitian, rumusan masalah, batasan masalah, manfaat penelitian dan sistematika penelitian.

2. BAB II LANDASAN KEPUSTAKAAN

   Dalam bab ini menguraikan tentang dasar teori dan referensi yang mendasari proses perancangan dan implemtasi sistem ini.

3. BAB III METODOLOGI PENELITIAN

   Bab ini menjelaskan tentang tahapan-tahapan yang dilakukan pada saat penelitian.

4. BAB IV PERANCANGAN

   Bab ini menjelaskan tentang rancangan aplikasi yang akan dibuat dalam penelitian ini.

5. BAB V IMPLEMENTASI

   Bab ini berisi tentang implementasi aplikasi dan pembahasan dari hasil perancangan dan analisi kebutuhan.

6. BAB VI PENGUJIAN

   Bab ini menjelaskan tentang hasil pengujian dan analisis terhadap aplikasi yang telah direalisasikan.

7. BAB VII PENUTUP

   Bab ini merupakan bab terakhir dari laporan penelitian yang berisi kesimpulan yang diperoleh dari pembuatan dan pengujian aplikasi serta saran-saran untuk pengembangan lebih lanjut.

BAB 2 LANDASAN KEPUSTAKAAN

Bab ini membahas tentang tinjauan pustaka yang meliputi kajian pustaka dan dasar teori yang diperlukan untuk penelitian. Kajian pustaka membahas penelitian yang telah ada dan yang diuraikan. Dasar teori membahas teori yang diperlukan untuk menyusun penelitian yang diusulkan.

2.1 Tinjauan Pustaka

Penelitian yang dibahas yaitu *"Spatial Clustering Berbasis Densitas Untuk Persebaran Titik Panas Sebagai Indikator Kebakaran Hutan Dan Lahan Gambut Di Sumatera"* Penelitian ini menerapkan algoritma DBSCAN (*Density-Based Spatial Clustering Algorithm with Noise*) yang merupakan algoritma pengelompokan berdasarkan kerapatan data untuk membentuk *cluster*. Penelitian tersebut melakukan *clustering* terhadap 91.324 *record*s data titik panas di Indonesia pada tahun 2002 dan 20.440 titik panas pada tahun 2013. Dari data tersebut diperoleh atribut lokasi lintang dan bujur serta waktu kemunculan titik panas di areal lahan gambut pulau Sumatera. Atribut lokasi (*longitude* dan *latitude*) dan tanggal munculnya titik panas merupakan atribut yang digunakan dalam penelitian tersebut. *Clustering* ini nantinya akan menghasilkan pengelompokkan terhadap daerah yang berpotensi terjadinya kebakaran. Dengan demikian, dapat dilakukan pencegahan kebakaran hutan dan lahan gambut sejak dini. Sebagian besar proses *clustering* dilakukan tanpa pengawasan (*unsupervised*), sehingga proses evaluasi *clustering* perlu dilakukan. Evaluasi *clustering* pada penelitian ini menggunakan metode *Sum Square Error (SSE)* untuk memperoleh jumlah *cluster* yang meminimalkan total *Square Error*. Analisis hasil *clustering* dilakukan berdasarkan karakteristik fisikal dari lahan gambut. Karakteristik lahan gambut mencakup tipe, kedalaman dan tutupan lahan gambut. Dari hasil *clustering* diperoleh bahwa pada tahun 2002, algoritma ini membentuk 52 cluster dengan SSE 0.26, sedangkan pada tahun 2013 algoritma ini menghasilkan 109 cluster dengan SSE 0.387 (Usman, 2014).

Penelitian lain yang dibahas, yaitu "*Implementasi Metode Clustering DBSCAN Pada Proses Pengambilan Keputusan"* Penelitian tersebut bertujuan mengimplementasikan metode *clustering* DBSCAN pada proses pengambilan keputusan untuk membantu perusahaan menentukan pelanggan potensialnya. Pada penelitian tersebut, proses *clustering* menggunakan algoritma DBSCAN ini menghasilkan 4 *cluster* dengan *silhouette coefficient* sebesar 0.900861. Nilai *silhouette coefficient* yang dihasilkan lebih besar dari 0 dan mendekati 1 yang mana berarti bahwa jumlah *cluster* yang dihasilkan pada proses *clustering* sudah optimal. Hal ini menandakan bahwa proses *clustering* menggunakan algoritma DBSCAN telah dapat dikategorikan baik (Devi, et al., 2015).

Penelitian lain yang dibahas, yaitu "*Implementasi Algoritma Density-Based Spatial Clustering of Application with Noise (DBSCAN) pada Mesin Pencari Dokumen Hasil Penelitian*" Untuk meningkatkan kualitas pencarian dokumen, dilakukan *Post-Retrieval Clustering* dengan menggunakan algoritma DBSCAN yang

mampu melakukan pengelompokkan berdasarkan tingkat kepadatan di dalam ruang data. Selain itu, algoritma DBSCAN juga dapat memisahkan dokumen-dokumen yang dianggap *noise* / dokumen yang tidak memiliki hubungan dengan dokumen lainnya agar tidak merusak kualitas dari *cluster – cluster* yang telah dibentuk dengan algoritma ini (Liman, et al., 2014).

Penelitian lain yang dibahas, yaitu "*Pembuatan Aplikasi Pendeteksi Anomali Pada Pola Konsumsi Listrik Pelanggan Kota Surabaya Menggunakan Algoritma Klasterisasi Berbasis Densitas*" Tujuan dari penelitian tersebut adalah untuk mengetahui adanya indikasi pencurian listrik dengan cara melakukan analisa terhadap *load profile* yang merupakan representasi dari pola konsumsi listrik seorang pelanggan di suatu area tertentu. Terdapat lima buah atribut yang digunakan dalam pendeteksian dan identifikasi anomali. Tiga atribut yang pertama adalah rata-rata konsumsi listrik pelanggan, konsumsi listrik maksimum pelanggan, dan standar deviasi dari konsumsi listrik pelanggan dalam rentan waktu tertentu. Algoritma DBSCAN digunakan untuk mendeteksi anomali dari kumpulan objek (Zakariya, et al., 2012).

2.2 Dasar Teori

2.2.1 Titik Panas (Hotspot)

Menurut Anderson, *et,al.* (1999), pada awalnya *hotspot* diidentikkan dengan titik api, namun dalam kenyataannya tidak semua *hotspot* mengindikasikan adanya titik api. Istilah *hotspot* lebih tepat bila bersinonimkan dengan titik panas.

Sebuah titik panas merupakan satu *pixel* pada potret satelit adalah suatu areal 1.1 km^2, dimana tinggi temperatur permukaannya mengindikasikan adanya kebakaran. Panas permukaan tersebut diukur oleh satelit NOAA yang dilengkapi oleh sensor-sensor radiometer mutakhir beresolusi sangat tinggi (Heryalianto, 2006).

Hotspot adalah titik panas yang diindikasikan sebagai lokasi kebakaran hutan dan lahan. Parameter ini sudah digunakan secara meluas di berbagai negara untuk memantau kebakaran hutan dan lahan dari satelit. Cara deteksi terjadinya kebakaran hutan dan lahan adalah dengan pengamatan titik panas (*hotspot*). Titik panas (*hotspot*) dapat dideteksi dengan satelit NOAA (National Oceanic and Atmospheric Administration) yang dilengkapi sensor AVHRR (Advenced Very Hight Resulation Radiometer). Dalam mendeteksi kebakaran hutan, satelit NOAA tidak mendeteksi kebakaran (suhu) secara langsung namun yang dideteksi adalah *hotspot* (Heryalianto, 2006).

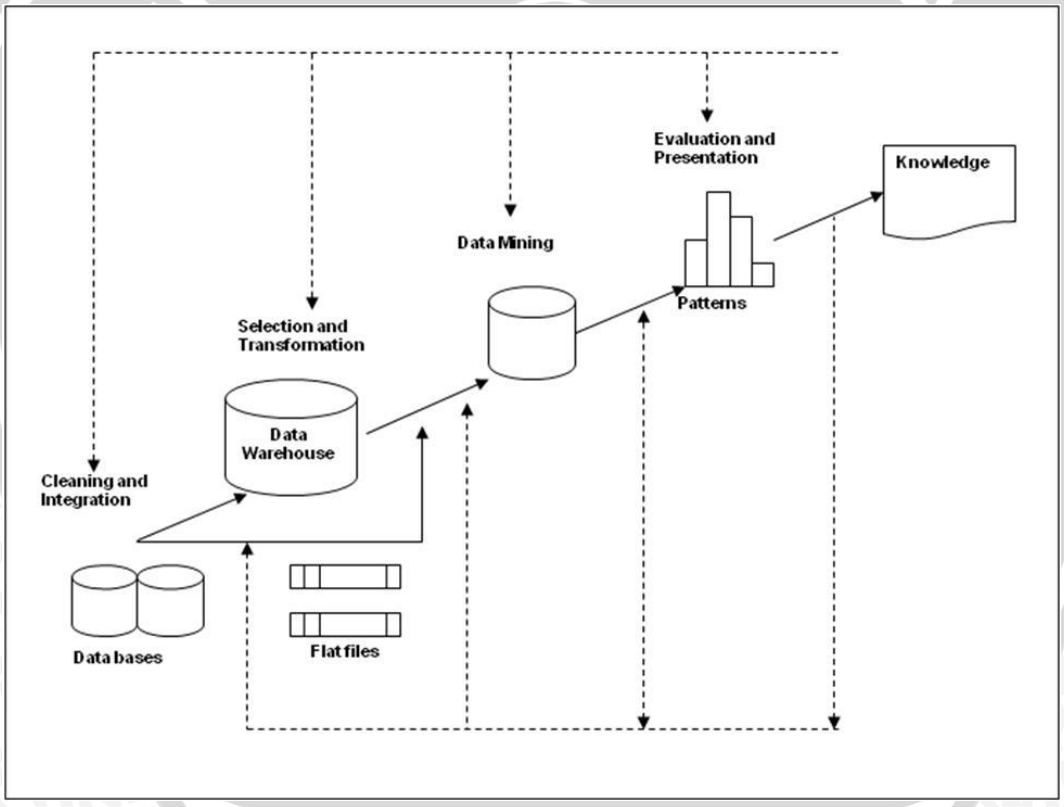
Sebuah titik panas (*hotspot*) dapat mencerminkan sebuah areal yang mungkin terbakar sebagian atau seluruhnya karena itu tidak menunjukkan secara pasti seberapa besar areal yang terbakar. Jumlah titik panas (*hotspot*) dapat sangat bervariasi dari suatu pengukuran selanjutnya tergantung dari waktu pengukuran pada hari itu (aktivitas api berkurang pada malam hari dan paling tinggi pada sore hari), cuaca (sensor yang digunakan tidak dapat menembus awan dan asap) dan organisasi apa yang memberikan data tersebut (tidak terdapat standar ambang

batas temperatur atau suhu untuk mengidentifikasikan titik panas) (Heryalianto, 2006).

Titik panas (*hotspot*) hanya memberikan sedikit informasi apabila tidak didukung oleh analisa dan interpretasi lanjutan. Kelompok titik panas (*hotspot*) dan atau titik panas (*hotspot*) yang berjumlah besar dan berlangsung secara terus menerus adalah indikator yang baik untuk kebakaran (titik api). Data titik panas (*hotspot*) bermanfaat apabila dikombinasikan dengan informasi-informasi seperti mengenai penggunaan lahan, penutupan tanaman, habitat binatang atau peta-peta lainnya. Kesalahan bias atau geografi dari sebuah titik panas (*hotspot*) dapat sampai sejauh 3 km (Heryalianto, 2006).

2.2.2 Konsep Data Mining

Data mining adalah kegiatan mengekstraksi atau menambang pengetahuan dari data yang berukuran/berjumlah besar, informasi inilah yang nantinya sangat berguna untuk pengembangan. Dimana langkah-langkah untuk melakukan data mining adalah sebagai berikut (Fadli, 2011):



Gambar 2.1 Langkah – Langkah Melakukan Data Mining

Kegunaan *data mining* adalah untuk menspesifikasikan pola yang harus ditemukan dalam tugas *data mining*. Secara umum tugas *data mining* dapat diklasifikasikan ke dalam dua kategori: deskriptif dan prediktif. Tugas menambang secara deskriptif adalah untuk mengklasifikasikan sifat umum suatu data di dalam *database*. Tugas *data mining* secara prediktif adalah untuk

mengambil kesimpulan terhadap data terakhir untuk membuat prediksi (Fadli, 2011).

Untuk melakukan *data mining* yang baik ada beberapa persoalan utama yaitu menyangkut metodologi *mining* dan interaksi *user*, *performance* dan perbedaan tipe *database*. Hal inilah yang sering kali dihadapi disaat kita ingin melakukan *data mining* (Fadli, 2011).

2.2.3 Normalisasi Min-Max

Normalisasi dilakukan untuk mentranformasi data dalam bentuk normalisasi ke dalam batas nilai 0 dan 1. Proses transformasi ini sering disebut dengan pemetaan yang bertujuan agar konvergensi lebih cepat tercapai, jika nilai rata-rata dari input data training mendekati nol. Rumus *min-max normalization* seperti yang diperkenalkan oleh Berry dan Linnof (2000), dengan Persamaan 2.10 (Fatkhiyah, 2012) :

$$N' = \frac{N - Min}{Max - Min}(New_{Max} - New_{Min}) + New_{Min} \quad (2.10)$$

Keterangan :

N' : Nilai Pemetaan

N : Nilai Original

Min : Nilai batas terendah

Max : Nilai batas tertinggi

New_{Max} : Nilai batas terbesar dalam pemetaan

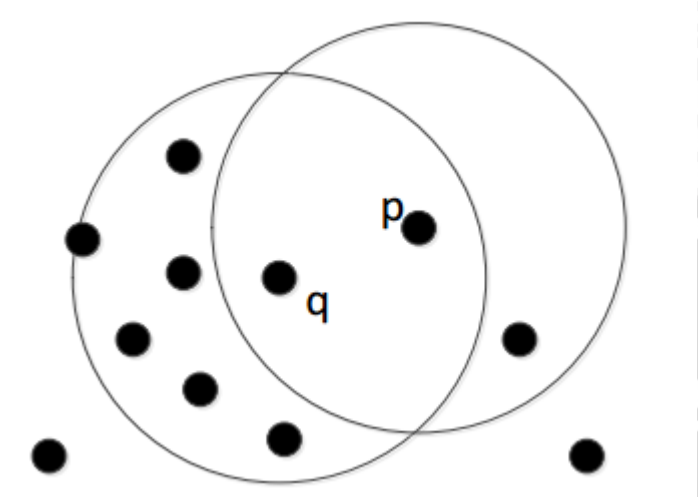
New_{Min} : Nilai batas terkecil dalam pemetaan

2.2.4 Algoritma DBSCAN

DBSCAN adalah algoritma *clustering* yang melihat bahwa sebuah *cluster* merupakan daerah – daerah yang padat objek dan terpisah dari daerah yang memiliki tingkat kepadatan yang rendah (*noise*). DBSCAN dapat membentuk daerah – daerah dengan bentuk yang tidak beraturan di dalam ruang data. Bagian terpenting dari algoritma ini adalah kepadatan objek dan hubungan antar ketetanggaan objek yang dibentuk dari objek – objek yang berdekatan (Liman, et al., 2014).

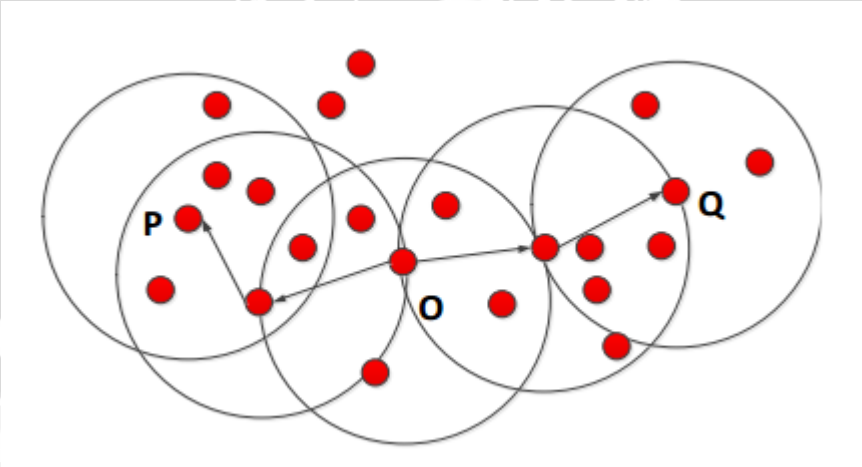
Konsep dasar dari algoritma DBSCAN adalah *density reachability* dan *density connected* serta bergantung dengan parameter radius maksimum dari sebuah ketetanggaan / *cluster* (*eps*) dan jumlah minimal objek di dalam sebuah ketetangaaan / *cluster* (*minPts*). DBSCAN menyatakan bahwa sebuah *cluster* dapat dibentuk jika untuk setiap titik data, pada dalam radius tertentu *eps* dari titik data tersebut terdapat minimal *minPts* titik objek (Liman, et al., 2014).

Density reachability menjelaskan bahwa dua titik objek yang berdekatan berada pada suatu *cluster* jika jarak antar titik objek < *eps* dan di dalam ketetanggaan dari salah satu titik objek tersebut terdapat minimal titik objek (*minPts*). Contoh penerapan dari *density reachability* dapat dilihat dari Gambar 2.2.



Gambar 2.2 Ilustrasi Density Reachability

Density Connectivity adalah tahapan lanjutan dari DBSCAN setelah *density reachibility* berhasil didefinisikan. titik objek P dan Q disebut *density connected* jika ada deretan (*sequence*) titik objek yang *density-reachable* (P,O, Q,) di antara P dan Q dimana O *density-reachable* dari Q sebagaimana ditunjukkan oleh Gambar 2.3.



Gambar 2.3 Ilustrasi Density Connectivity

Konsep dari algoritma DBSCAN adalah sebagai berikut (Zakariya, et al., 2012) :

1. *Epsilon (eps). Epsilon* adalah jarak yang menentukan kedekatan antara suatu objek dengan objek lain disekitarnya. Objek – objek yang memenuhi nilai *epsilon* dikatakan memiliki kedekatan dengan suatu objek yang diamati.

2. Jumlah objek minimum (*minPts*). *MinPts* menentukan pembentukan *cluster*. Suatu *cluster* akan dibentuk jika jumlah objek yang berdekatan telah melebihi dari nilai minimum (*minPts*) yang diberikan.
3. Selanjutnya algoritma ini menamai objek - objek yang terletak dekat dengan nilai *epsilon* yang diberikan disebut objek *core* sedangkan objek - objek yang memiliki kedekatan dengan objek *core* disebut dengan objek *border*. Di luar itu, objek - objek yang bukan merupakan objek *core* atau *border* merupakan suatu *outlier* atau anomali.

Komputasi dari algoritma *Density-Based Spatial Clustering of Application with Noise* dapat dilakukan dengan langkah-langkah sebagai berikut (Devi, et al., 2015) :

1. Inisialisasi parameter *minPts, eps*
2. Tentukan titik awal atau *p* secara acak
3. Ulangi langkah 3 sampai 5 hingga semua titik diproses
4. Hitung *eps* atau semua jarak titik yang *density reachable* terhadap *p* menggunakan persamaan 2.20

$$E(x,y) = \sqrt{\sum_{i=0}^{n} (X_i - Y_i)^2} \quad (2.20)$$

5. Jika titik yang memenuhi *eps* lebih dari *minPts* maka titik *p* adalah *core point* dan *cluster* terbentuk
6. Jika *p* adalah *border point* dan tidak ada titik yang *density reachable* terhadap *p*, maka proses dilanjutkan ke titik yang lain

2.2.5 Algoritma Sillhouette Coeficient

Silhouette Coefficient digunakan untuk melihat kualitas dan kekuatan *cluster*, seberapa baik suatu objek ditempatkan dalam suatu *cluster*. Metode ini merupakan gabungan dari metode *Cohesion* dan *Separation*. Tahapan perhitungan *Silhouette Coefficient* adalah sebagai berikut (Handoyo, et al., 2014; Kaufman & Rousseeuw, 1990) :

1. Hitung rata-rata jarak dari suatu dokumen misalkan i dengan semua dokumen lain yang berada dalam satu *cluster*

$$a(i) = \frac{1}{|A| - 1} \sum j \in A, j \neq i \; d(i,j) \quad (2.30)$$

dengan j adalah dokumen lain dalam satu *cluster* A dan d(i,j) adalah jarak antara dokumen i dengan j.

2. Hitung rata-rata jarak dari dokumen i tersebut dengan semua dokumen di *cluster* lain, dan diambil nilai terkecilnya.

$$d(i,C) = \frac{1}{|A|}\sum j \in C\ d(i,j) \quad (2.31)$$

dengan d(i,C) adalah jarak rata-rata dokumen i dengan semua objek pada *cluster* lain C dimana A ≠ C.

$$b(i) = \min C \neq A\ d(i,C) \quad (2.32)$$

3. Nilai *Silhouette Coefficient* nya adalah :

$$s(i) = \frac{b(i) - a(i)}{max(a(i), b(i))} \quad (2.33)$$

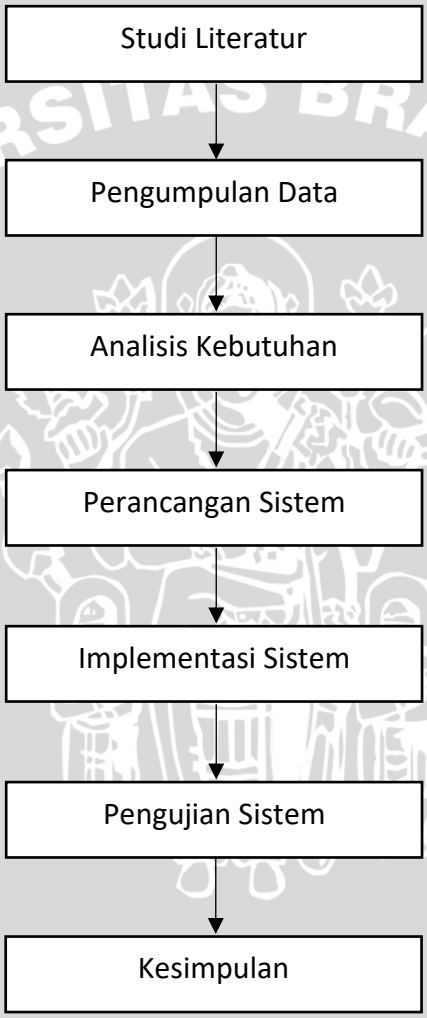
Nilai rata-rata *Silhouette* dari tiap objek dalam suatu *cluster* adalah suatu ukuran yang menunjukkan seberapa padat data dikelompokkan dalam *cluster* tersebut. Berikut ini ukuran nilai *Silhouette Coefficient* berdasarkan Kaufman dan Rousseeuw (Kaufman & Rousseeuw, 1990) :

Tabel 2.1 Ukuran Nilai *Silhouette Coefficient*

Silhouette Coefficient	Intepretasi yang diusulkan
0.71 – 1.00	*Strong structure*
0.51 – 0.70	*Medium structure*
0.26 – 0.50	*Weak structure*
≤ 0.25	*No structure*

BAB 3 METODOLOGI PENELITIAN

Bab ini menjelaskan tentang metode yang digunakan dalam implementasi metode *Density-Based Spatial Clustering of Application with Noise* (DBSCAN) dalam *clustering* titik panas. Metodologi penelitian ini dilakukan melalui beberapa tahapan, yaitu studi literatur, pengumpulan data, analisis kebutuhan, implementasi, pengujian dan pengambilan kesimpulan. Tahapan - tahapan dalam penelitian tersebut diilustrasikan dalam diagram blok metode penelitian pada Gambar 3.1.



Gambar 3.1 Diagram Blok Metodologi Penelitian

Dari Gambar 3.1. dapat diketahui alur dari diagram blok metodologi penelitian dimana tiap tahapannya mempunyai fungsi serta tugas masing – masing yang saling berkaitan dari satu tahapan ke tahapan berikutnya. Sehingga dalam melakukan penelitian untuk membangun sebuah sistem harus melewati tahapan demi tahapan tersebut. Adapun penjelasan dari tiap tahapan tersebut, yaitu :

3.1 Studi Literatur

Studi literatur pada penelitian ini bertujuan untuk mencari referensi yang relevan dengan permasalahan yang ada agar peneliti mampu meningkatkan pemahaman serta pengetahuan mengenai permasalahan yang akan diangkat. Literatur diperoleh dari buku, internet, bimbingan dengan dosen pembimbing, dan teori yang berkaitan dengan titik panas, kemudian teori tentang metode atau algoritma DBSCAN, serta sumber lain yang teorinya dapat mendukung penelitian ini. Selanjutnya setelah mendapat teori – teori yang diperlukan maka teori – teori tersebut dicantumkan dalam dokumen penelitian dengan mencantumkan sumber dari teori tersebut.

3.2 Pengumpulan Data

Data yang digunakan dalam penilitian ini adalah data persebaran titik panas di Indonesia pada periode Juni hingga Agustus 2015 yang diunduh dari halaman situs NASA. Data tersebut tersebut terdiri dari 39638 *records* dengan 13 fitur. Data yang akan digunakan dalam proses pengujian dibatasi hanya data titik panas dengan fitur *confidence* diatas 80% yang berjumlah 11171 *records*. Dari 13 fitur yang ada, hanya akan digunakan sebanyak 4 fitur. Fitur yang digunakan antara lain :

a. *Brightness,* merupakan tingkat kecerahan suhu dari satu *pixel* api pada *channel* 21/22 diukur dalam kelvin.
b. *Confidence,* merupakan tingkat keyakinan yang diperoleh pada saat proses deteksi titik api. Fitur *confidence* hanya berperan sebagai *filter*.
c. *Bright_t31,* merupakan tingkat kecerahan suhu dari satu *pixel* api pada *channel* 31 diukur dalam kelvin.
d. *Frp (fire radiative power),* menggambarkan kekuatan radiasi api dalam megawatt.

3.3 Analisis Kebutuhan

Analisa kebutuhan dilakukan dengan menentukan kebutuhan apa saja yang dibutuhkan untuk mambangun perangkat lunak. Berikut ini kebutuhan yang mendukung pembuatan sistem. Antara lain :

1. Kebutuhan *hardware*, meliputi :
   - Laptop dengan *Processor* Intel® Core™ i5-5200U CPU @2.20GHz 2.19GHz
   - *Memory* 4 GB
2. Kebutuhan *software*, meliputi :
   - Sistem operasi *Windows 10 Enterprise* 64-bit
   - *Netbeans IDE* 8.1
   - *Apache Commons CSV Library*
   - *Java Development Kit* (JDK) 8
3. Data yang dibutuhkan
   Data persebaran titik panas di Indonesia pada periode Juni hingga Agustus 2015

3.4 Perancangan Sistem

Pada tahapan perancangan sistem ini terdapat beberapa deskripsi sistem yang akan dibuat secara umum yaitu perancangan aplikasi, manualisasi, rancangan antar muka aplikasi, perancangan tabel, pengujian dan evaluasi.

3.4.1 Deskripsi Sistem

Penelitian ini dilakukan dengan metode *Density-Based Spatial Clustering of Application with Noise* (DBSCAN) dalam *clustering* titik panas. Terdapat 4 parameter dari data titik panas yaitu *brightness, confidence, bright_t31* dan *frp*. Dimana *confidence* hanya berperan sebagai filter. Output sistem ini yaitu *cluster* yang telah dikelompokkan berdasarkan kemiripan.

3.5 Implementasi Sistem

Implementasi adalah fase membangun sistem yang mengacu pada perancangan sistem dan menerapkan hal yang telah didapatkan dalam proses studi literature. Fase-fase yang ada dalam implementasi antara lain :

- Implementasi *interface,* menggunakan software NetBeans IDE 8.1.
- Implementasi *load data* berupa *file comma separated value (.csv)* menggunakan *library Apache Commons CSV.*
- Implementasi algoritma, melakukan perhitungan dengan menggunakan metode DBSCAN kedalam bahasa pemrograman *Java.*
- Implementasi ini akan menghasilkan *output* berupa jumlah *cluster* yang terbentuk, jumlah titik yang dianggap *noise* dan nilai *Silhouette Coefficient.*

3.6 Pengujian Sistem

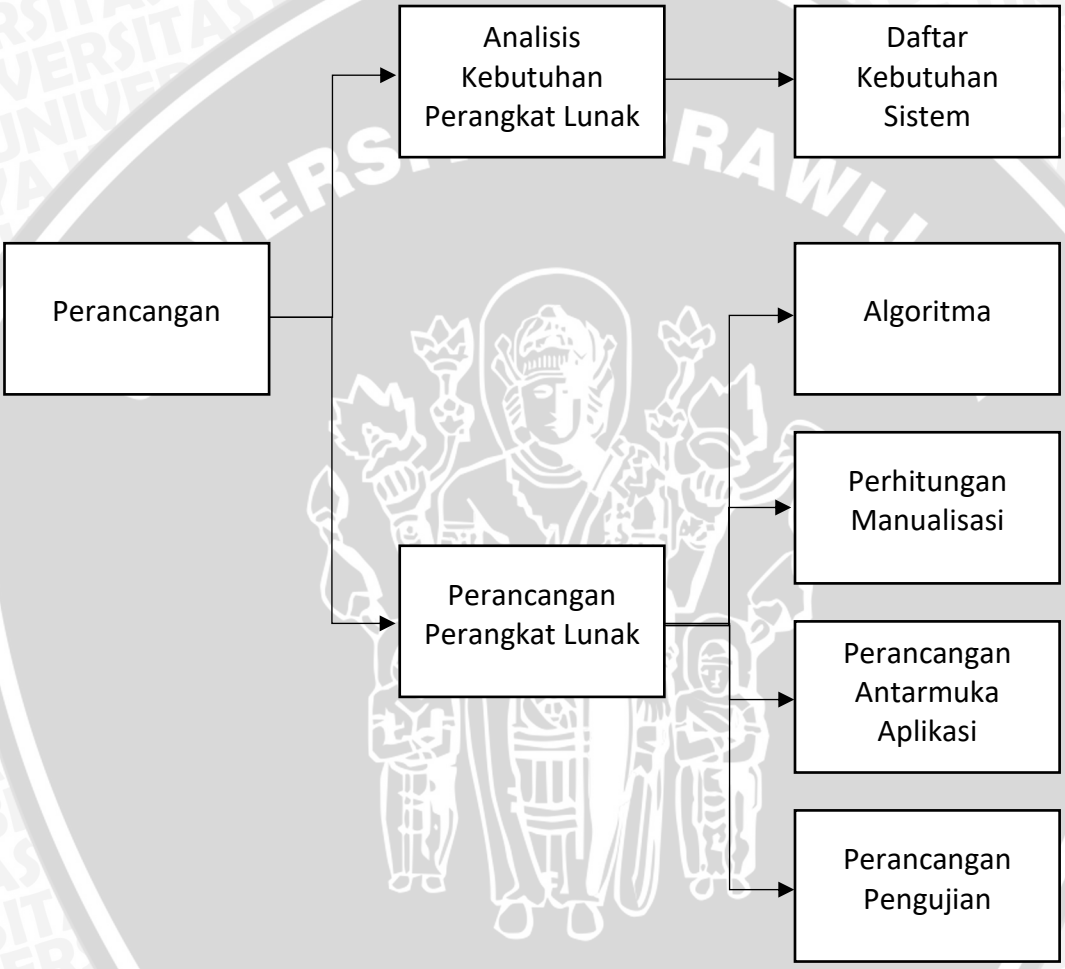
Pengujian perangkat lunak pada skripsi ini dilakukan untuk menunjukkan bahwa perangkat lunak mampu bekerja sesuai dengan spesifikasi dari kebutuhan yang telah ditentukan. Pengujian yang akan dilakukan adalah pengujian untuk mengetahui pengaruh nilai parameter *minPts* dan *eps* terhadap nilai *Silhouette Coefficient*, jumlah *cluster* yang terbentuk, dan jumlah titik yang dianggap *noise.*

3.7 Kesimpulan

Kesimpulan dilakukan setelah semua perancangan, implementasi dan pengujian yang diterapkan sudah selesai dilakukan. Kesimpulan diambil dari hasil pengujian. Tahap terakhir dari penulisan adalah saran yang dimaksudkan untuk memperbaiki kesalahan yang terjadi serta memberikan pertimbangan untuk pengembangan selanjutnya.

BAB 4 PERANCANGAN

Bab ini menjelaskan tentang analisis kebutuhan dan perancangan sistem *clustering* titik panas menggunakan metode DBSCAN. Perancangan dilakukan melalui beberapa tahapan, yaitu analisis kebutuhan perangkat lunak, perancangan perangkat lunak, perhitungan manualisasi, perancangan antarmuka aplikasi, dan perancangan pengujian. Alur perancangan sistem dapat dilihat dalam pohon perancangan pada Gambar 4.1.



Gambar 4.1 Pohon Perancangan Sistem

4.1 Analisis Kebutuhan Perangkat Lunak

Analisa sistem merupakan tahapan awal dalam perancangan sebuah sistem. Analisa kebutuhan sistem ini memiliki tujuan untuk memberikan gambaran informasi kebutuhan yang akan digunakan dalam tahapan perancangan.

4.1.1 Daftar Kebutuhan Sistem

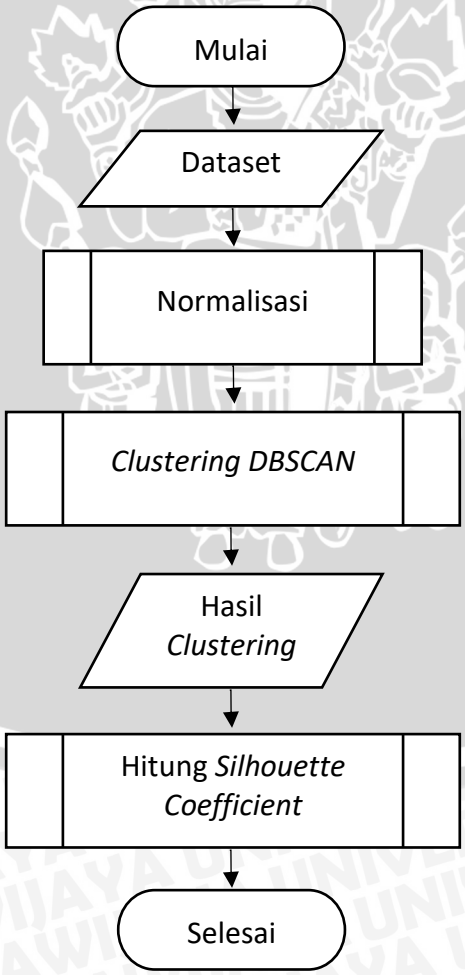
Tahapan ini bertujuan untuk menjelaskan kebutuhan sistem yang harus dipenuhi saat pengguna melakukan sebuah aksi. Daftar kebutuhan ini terdiri dari kolom yang berisi hal-hal yang harus disediakan oleh sistem. Daftar fungsionalitas sistem dapat dilihat pada Tabel 4.1.

Tabel 4.1 Daftar Kebutuhan Sistem

Kebutuhan Sistem	Aksi
Sistem menyediakan proses *load* data untuk mengambil data dari *file csv*	*Load* data
Sistem menyediakan proses untuk normalisasi data	Normalisasi
Sistem menyediakan proses untuk *clustering*	*Clustering*

4.2 Perancangan Perangkat Lunak

Tahap ini adalah tahapan dimana perancangan perangkat lunak penerapan metode DBSCAN dalam *clustering* titik panas. Cara kerja perangkat lunak ini dapat dilihat pada Gambar 4.2.



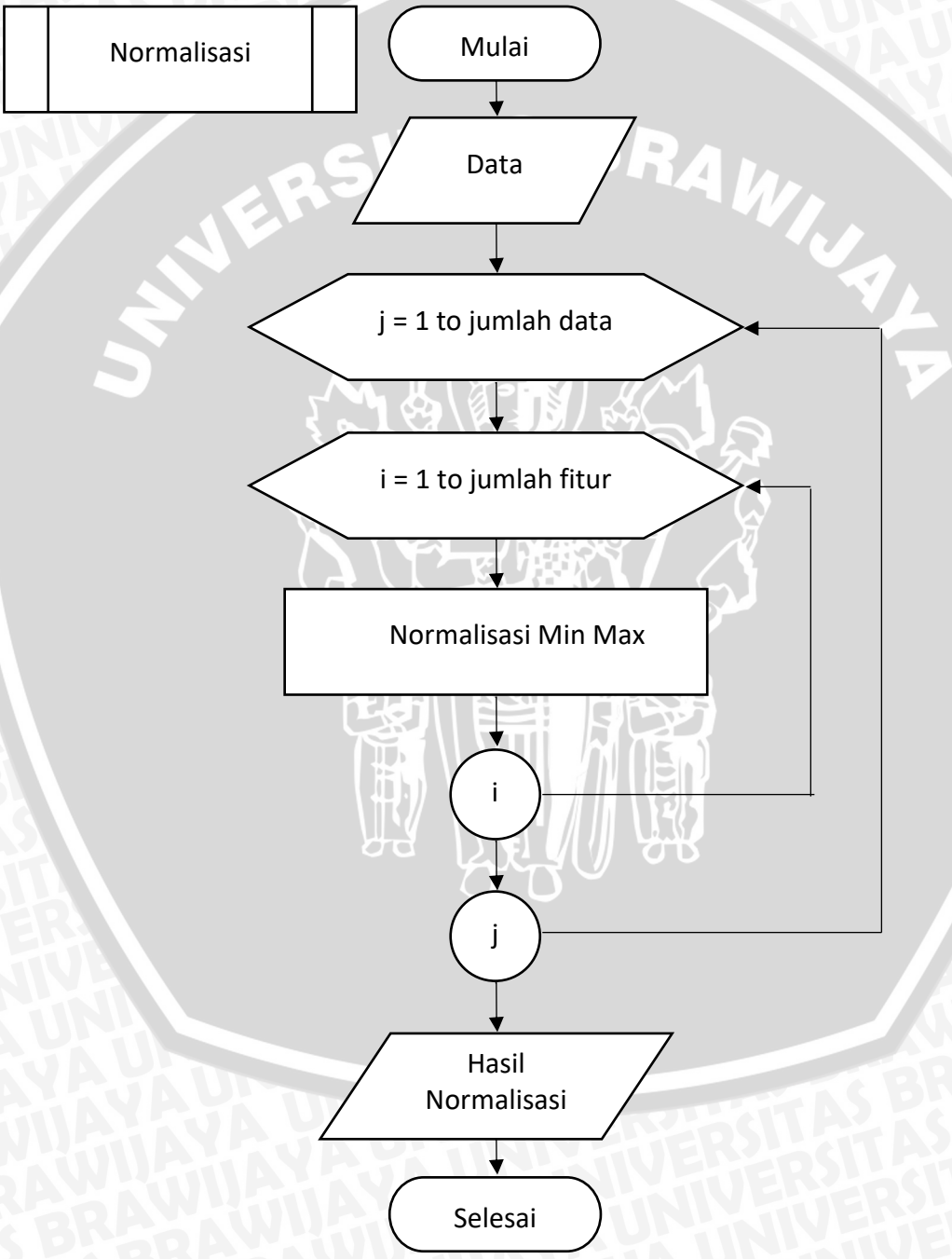
Gambar 4.2 Diagram Alir Proses Perangkat Lunak

4.2.1 Algoritma

Pada tahapan ini menjelaskan alur dari algoritma yang akan digunakan dalam proses *clustering* titik panas. Terdapat tiga algoritma utama yang digunakan yaitu proses normalisasi data, proses *clustering* DBSCAN, dan proses perhitungan *Silhouette Coefficient.*

4.2.1.1 Proses Normalisasi Data

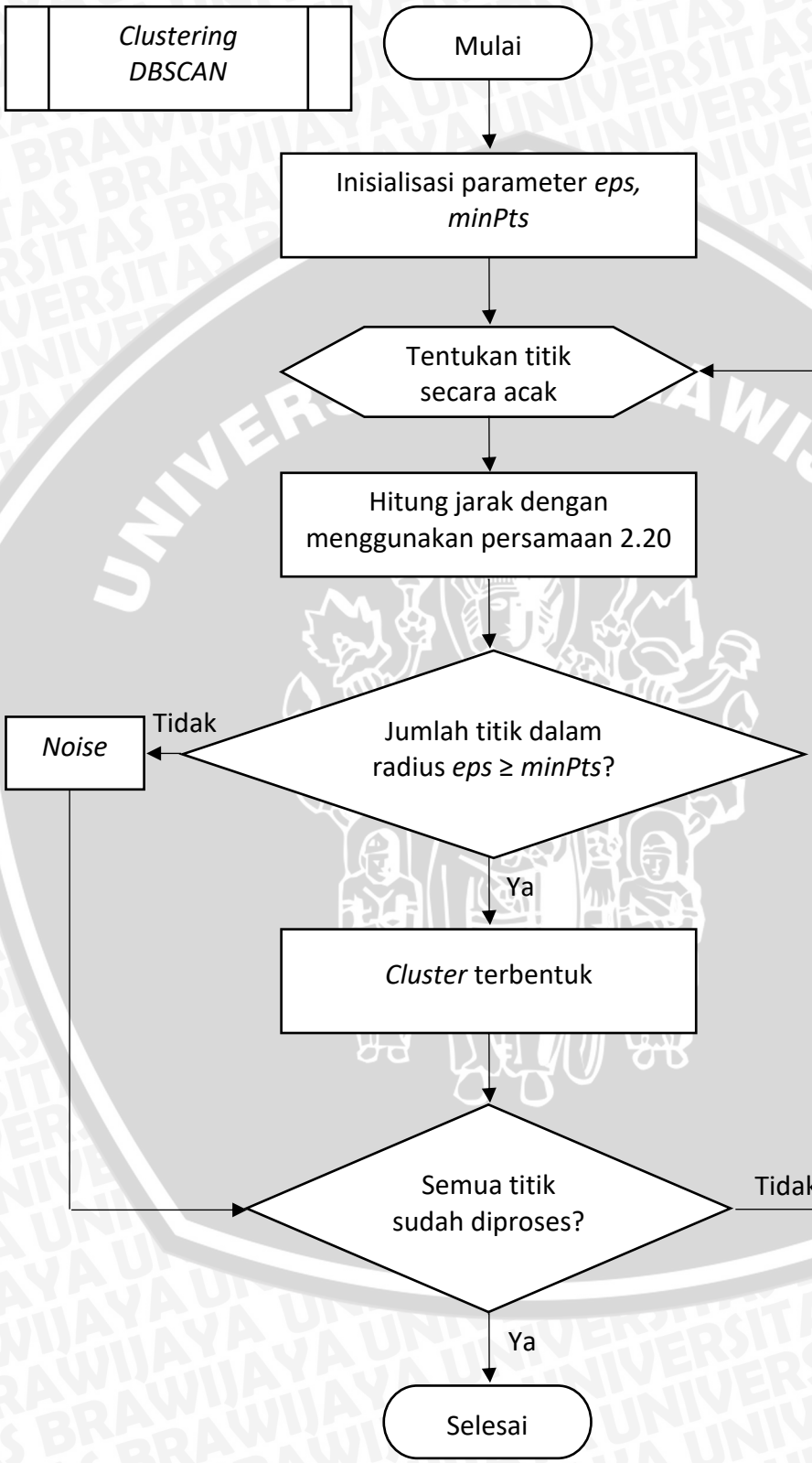
Secara garis besar proses dari normalisasi ditunjukkan pada Gambar 4.3.



Gambar 4.3 Diagram Alir Proses Normalisasi Data

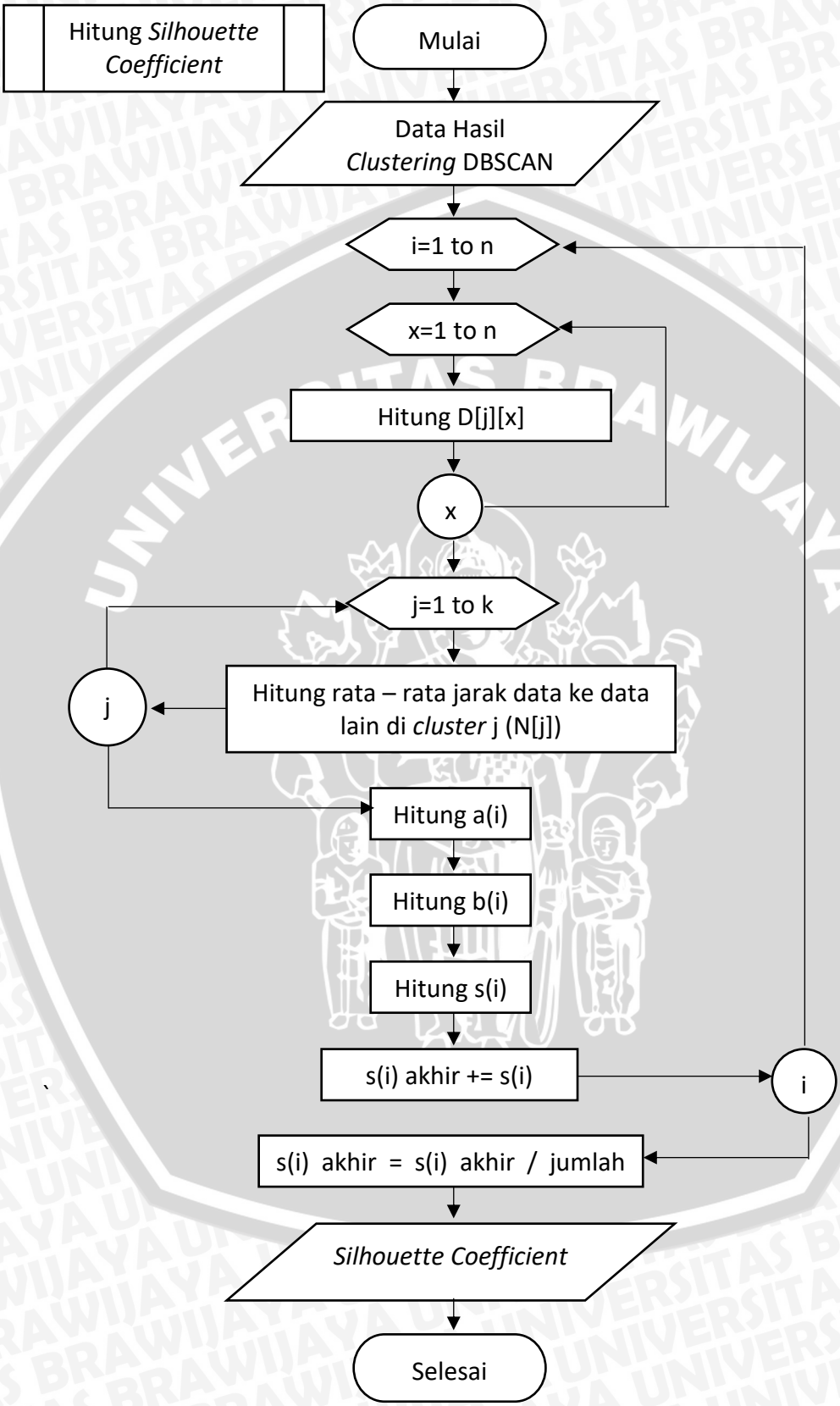
4.2.1.2 Proses Clustering DBSCAN

Secara garis besar proses dari *clustering* ditunjukkan pada Gambar 4.4.



Gambar 4.4 Proses *Clustering* DBSCAN

4.2.1.3 Proses Perhitungan Silhouette Coefficient



Gambar 4.5 Proses Perhitungan *Silhouette Coefficient*

4.2.2 Perhitungan Manualisasi

Perhitungan manualisasi akan dilakukan dengan menggunakan *dataset sample* yang berisi 10 *records* seperti yang ditunjukkan pada Tabel 4.2.

Langkah 1 : *load dataset*

Terdapat 4 fitur yang akan digunakan yaitu *brightness, confidence, bright_t31,* dan *frp (fire radiative power).* Tetapi fitur *confidence* hanya berperan sebagai *filter.*

Tabel 4.2 *Dataset Sample*

TITIK	BRIGHTNESS	BRIGHT_T31	FRP
A	327.6	297.3	35.1
B	319.3	291.6	18.7
C	323.2	293.8	24.4
D	321.5	296.4	17.7
E	325.6	299.4	24.3
F	348	298.9	73.6
G	321.8	299.3	44.4
H	324.6	303	18.7
I	333.7	297.5	74.7
J	336.1	303.4	65.4

Langkah 2 : Normalisasi data, rumus bisa dilihat di Persamaan 2.10.

Misal, untuk titik A fitur *brightness* :

$$N' = \frac{327.6 - 319.3}{348.0 - 319.3}(1 - 0) + 0 = 0.289$$

Setelah semua data di normalisasi ke dalam rentan 0 - 1, hasilnya dapat dilihat pada Tabel 4.3.

Tabel 4.3 Hasil Normalisasi Data

TITIK	BRIGHTNESS	BRIGHT_T31	FRP
A	0.289	0.483	0.305
B	0.000	0.000	0.018
C	0.136	0.186	0.118
D	0.077	0.407	0.000
E	0.220	0.661	0.116
F	1.000	0.619	0.981
G	0.087	0.653	0.468
H	0.185	0.966	0.018
I	0.502	0.500	1.000
J	0.585	1.000	0.837

Langkah 3 : Inisialisasi parameter *minPts* = 2 dan *eps* = 0.600

Langkah 4 : Titik A dipilih sebagai titik awal atau *p.*

Langkah 5 : Hitung jarak antar setiap semua pasangan objek dengan perhitungan rumus di Persamaan 2.20.

Misal, untuk jarak antara titik A dengan titik B :

$$d(A,B) = \sqrt{(0.289 - 0.000)^2 + (0.483 - 0.000)^2 + (0.305 - 0.018)^2}$$
$$= 0.632$$

Besar jarak antar titik dapat dilihat pada Tabel 4.4.

Tabel 4.4 *Euclidean Distance*

TITIK	A	B	C	D	E
A	0	0.632	0.383	0.380	0.269
B	0.632	0	0.251	0.414	0.703
C	0.383	0.251	0	0.257	0.482
D	0.380	0.414	0.257	0	0.314
E	0.269	0.703	0.482	0.314	0
F	0.990	1.520	1.296	1.364	1.166
G	0.310	0.798	0.585	0.529	0.377
H	0.494	0.984	0.788	0.570	0.322
I	0.727	1.211	1.005	1.091	0.942
J	0.798	1.419	1.175	1.145	0.877

F	G	H	I	J	TITIK
0.990	0.310	0.494	0.727	0.798	A
1.520	0.798	0.984	1.211	1.419	B
1.296	0.585	0.788	1.005	1.175	C
1.364	0.529	0.570	1.091	1.145	D
1.166	0.377	0.322	0.942	0.877	E
0	1.047	1.309	0.513	0.581	F
1.047	0	0.558	0.691	0.710	G
1.309	0.558	0	1.133	0.913	H
0.513	0.691	1.133	0	0.533	I
0.581	0.710	0.913	0.533	0	J

Langkah 6 : Jika titik yang memenuhi *eps* lebih dari *minPts* maka titik *p* adalah *core point* dan *cluster* terbentuk. Terdapat 5 titik yang jaraknya ≤ *eps* yaitu titik C, D, E, G, dan H. Karena jumlah titik bertetanggan dengan titik A adalah 6 (termasuk titik A) ≥ *minPts*, maka *cluster* terbentuk dan diberi nama dengan *cluster1*.

Tabel 4.5 Titik yang berada di dalam radius *p*

Titik		Jarak
A	C	0.383
	D	0.380
	E	0.269
	G	0.310
	H	0.494

Kemudian dilanjutkan ke titik berikutnya. Titik B dipilih secara acak. Terdapat 2 titik yang jaraknya ≤ *eps* yaitu titik C dan D. Karean titik B *density reachable* dari titik A melalui titik C maupun D, maka titik B akan bergabung kedalam *cluster1*.

Tabel 4.6 Titik yang berada di dalam radius *p*

Titik		Jarak
B	C	0.251
	D	0.414

Kemudian dilanjutkan ke titik berikutnya. Titik F dipilih secara acak. Terdapat 2 titik yang jaraknya ≤ *eps* yaitu titik I dan J. Jumlah tetangga titik F (termasuk titik F) ≥ *minPts*, sehingga titik F, I dan J membentuk *cluster* baru dan diberi nama *cluster2*.

Tabel 4.7 Titik yang berada di dalam radius *p*

Titik		Jarak
F	I	0.513
	J	0.581

Proses dilanjutkan sampai semua titik diproses sehingga diperoleh hasil sebagai berikut :

Tabel 4.8 Hasil Akhir *Clustering*

TITIK	CLUSTER	TITIK	CLUSTER
A	*cluster1*	F	*cluster2*
B	*cluster1*	G	*cluster1*
C	*cluster1*	H	*cluster1*
D	*cluster1*	I	*cluster2*
E	*cluster1*	J	*cluster2*

Setalah proses *clustering* selesai, kemudian dilanjutkan dengan proses evaluasi hasil *clustering* menggunakan metode *Silhouette Coefficient*.

Langkah 1 : mencari a(i) yaitu jarak rata-rata titik i dengan titik lain dalam satu *cluster.* Rumus dapat dilihat pada Persamaan 2.30.

Misal, nilai a(i) dari titik A adalah rata-rata jarak titik A ke B, C, D, E, G, dan H.

$$a(A) = \frac{0.632 + 0.383 + 0.380 + 0.269 + 0.310 + 0.572}{6} = \frac{2.546}{6} = 0.424$$

Hasil nilai a(i) dari semua titik dapat dilihat pada Tabel 4.9.

Tabel 4.9 Nilai a(i)

Titik	Nilai a(i)	Titik	Nilai a(i)
A	0.424	F	0.547
B	0.630	G	0.526
C	0.458	H	0.632
D	0.411	I	0.523
E	0.411	J	0.557

Langkah 2 : mencari b(i) yaitu jarak rata-rata titik i dengan titik lain di *cluster* lain dan diambil nilai terkecilnya. Rumus dapat dilihat pada Persamaan 2.31 dan 2.32.

Misal, nilai b(i) dari titik A adalah rata-rata jarak titik A ke F, I, dan J.

$$b(A) = \frac{0.990 + 0.727 + 0.798}{3} = \frac{2.515}{3} = 0.838$$

Hasil nilai b(i) dari semua titik dapat dilihat pada Tabel 4.10.

Tabel 4.10 Nilai b(i)

Titik	Nilai b(i)
A	0.838
B	1.383
C	1.159
D	1.200
E	0.995
F	1.242
G	0.816
H	1.118
I	0.971
J	1.005

Langkah 3 : mencari s(i) yaitu nilai *Silhouette Coefficient* titik i. Rumus dapat dilihat pada Persamaan 2.33.

Misal, nilai s(i) dari titik A adalah :

$$s(A) = \frac{0.838 - 0.424}{\max(0.424,\ 0.838)} = \frac{0.414}{0{,}838} = 0.494$$

Hasil nilai s(i) dari semua titik dapat dilihat pada Tabel 4.11.

Tabel 4.11 Nilai s(i)

Titik	Nilai s(i)
A	0.494
B	0.544
C	0.605
D	0.658
E	0.587
F	0.559
G	0.355
H	0.435
I	0.462
J	0.446

Kemudian untuk nilai s(i) akhir adalah rata-rata dari s(i)

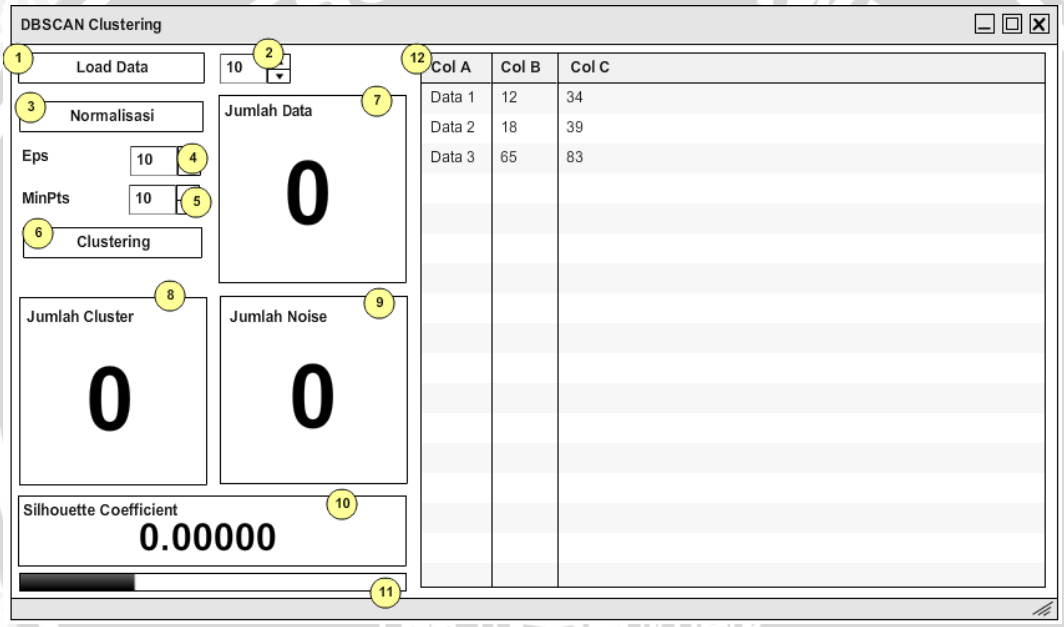
$$= \frac{0.494 + 0.544 + 0.605 + 0.658 + 0.587 + 0.559 + 0.355 + 0.435 + 0.462 + 0.446}{10}$$

$$= \frac{5.145}{10} = 0.5145$$

Dengan *minPts* = 2 dan *eps* ≤ 0.600, algoritma ini menghasilkan nilai *Silhouette Coefficient* sebesar 0.5145 yang menurut Kaufman dan Rousseeuw tergolong dalam *medium structure*.

4.2.3 Perancangan Antarmuka Aplikasi

Berikut ini adalah Gambar 4.6 yang merupakan rancangan antarmuka yang digunakan pada perangkat lunak.



Gambar 4.6 Rancangan Antarmuka Aplikasi

Terdapat sepuluh bagian yang berfungsi dalam berjalannya aplikasi, antara lain *button Load Data, spinner filter confidence, button* Normalisasi*, spinner eps, spinner minPts, button Clustering, label* Jumlah Data*, label* Jumlah *Cluster, label* Jumlah *Noise, label Silhouette Coefficient, progress bar* dan *table Cluster.* Berikut ini adalah keterangan dari Gambar 4.6 :

1. *Button Load Data* adalah tombol yang berfungsi untuk menampilkan *dataset* yang akan digunakan dalam proses perhitungan. *Dataset* akan ditampilkan pada *table cluster* pada no 12.
2. *Spinner filter confidence* adalah tempat untuk memasukkan nilai yang akan digunakan sebagai *filter*. Dimana data yang diambil adalah data dengan *confidence >= filter.*
3. *Button* Normalisasi adalah tombol untuk memulai proses normalisasi.
4. *Spinner eps* adalah tempat untuk memasukkan nilai parameter *epsilon.*

5. *Spinner minPts* adalah tempat untuk memasukkan nilai parameter *minPts.*
6. *Button Clustering* adalah tombol untuk memulai proses *clustering* sampai menghitung nilai *Silhouette Coefficient.*
7. *Label Data* adalah *label* yang digunakan untuk menampilkan berapa banyak jumlah data yang akan diproses.
8. *Label Cluster* adalah *label* yang digunakan untuk menampilkan berapa banyak jumlah *cluster* yang terbentuk.
9. *Label Noise* adalah *label* yang digunakan untuk menampilkan berapa banyak jumlah titik yang dianggap *noise*.
10. *Label Silhouette* adalah *label* yang digunakan untuk menampilkan nilai dari *Silhouette Coefficient.*
11. *Progress bar* adalah sebuah *bar* yang menunjukkan bahwa proses *clustering* sedang berlangsung
12. *Table Cluster* adalah tabel yang digunakan untuk menampilkan data titik panas.

4.2.4 Perancangan Pengujian

Pengujian yang akan dilakukan adalah pengujian untuk mengetahui pengaruh nilai parameter *minPts* dan *eps* terhadap nilai *Silhouette Coefficient*, jumlah *cluster* yang terbentuk, dan jumlah titik yang dianggap sebagai *noise.*

Perancangan tabel pengujian *minPts* dan *eps* terhadap nilai *Silhouette Coefficient*, jumlah *cluster* yang terbentuk, dan jumlah titik yang dianggap sebagai *noise* dapat dilihat pada Tabel 4.12.

Tabel 4.12 Perancangan Pengujian Parameter *MinPts* dan *Eps*

Eps	MinPts	Jumlah Cluster	Jumlah Noise	Silhouette Coefficient

BAB 5 IMPLEMENTASI

Bab ini membahas mengenai implementasi sistem yang dibuat berdasarkan analisis dan perancangan sistem.

5.1 Spesifikasi Sistem

Spesifikasi sistem yang digunakan untuk mengelompokkan titik panas dibagi menjadi spesifikasi perangkat keras dan spesifikasi perangkat lunak.

5.1.1 Spesifikasi Perangkat Keras

Spesifikasi perangkat keras yang digunakan untuk mengelompokkan titik panas dijelaskan pada Tabel 5.1.

Tabel 5.1 Spesifikasi Perangkat Keras

Nama Komponen	Spesifikasi
Prosesor	Intel® Core™ i5-5200U CPU @2.20GHz 2.19GHz
Memori	4 GB
Kartu Grafis	NVIDIA GeForce 930M
Hardisk	500GB

5.1.2 Spesifikasi Perangkat Lunak

Spesifikasi perangkat lunak yang digunakan untuk mengelompokkan titik panas dijelaskan pada Tabel 5.2.

Tabel 5.2 Spesifikasi Perangkat Lunak

Nama	Spesifikasi
Sistem Operasi	Windows 10 64bit
Bahasa Pemrograman	Java
Tools Pemrograman	NetBeans IDE 8.1
Library	Apache Commons CSV

5.2 Batasan Implementasi

Batasan implementasi yang digunakan untuk mengimplementasikan pengelompokan titik panas adalah sebagai berikut:

a. Sistem diimplementasikan dalam bentuk aplikasi *desktop* menggunakan bahasan pemrograman *Java*.

b. Metode yang digunakan dalam implementasi sistem adalah DBSCAN.

c. Input yang digunakan dalam proses perhitungan adalah data persebaran titik panas di Indonesia pada periode Juni hingga Agustus 2015 dan output yang dihasilkan sistem adalah jumlah *cluster* yang terbentuk serta nilai *Silhouette Coefficient*.

5.3 Implementasi Algoritma

Pada Implementasi metode *Density-Based Spatial Clustering of Application with Noise* (DBSCAN) dalam *clustering* titik panas ini mempunyai beberapa proses algoritma.

5.3.1 Load Data

Proses ini akan mengambil data berupa *comma separated value (.csv)* yang berisi nilai *brightness, confidence, bright_t31,* dan *frp.*

```
1   public class LoadData {
2       //membaca file (.csv)
3       public void read(String filePath, int confi) throws
    FileNotFoundException, IOException {
4           FileReader fileReader = null;
5           CSVParser csvFileParser = null;
6           CSVFormat csvFileFormat =
    CSVFormat.EXCEL.withFirstRecordAsHeader();
7           fileReader = new FileReader(filePath);
8           csvFileParser = new CSVParser(fileReader,
    csvFileFormat);
9           List csvRecords = csvFileParser.getRecords();
10          for (int i = 0; i < csvRecords.size(); i++) {
11              CSVRecord record = (CSVRecord)
    csvRecords.get(i);
12              double confidence =
    Double.parseDouble(record.get("confidence"));
13              if (confidence >= confi) {
14                  Point p = new Point();
15  p.setBrightness(Double.parseDouble(record.get("brightness")
    ));
16  p.setBright_t31(Double.parseDouble(record.get("bright_t31")
    ));
17  p.setFrp(Double.parseDouble(record.get("frp")));
18                  p.add(p);
19              }
20          }
21      }
22  }
```

Sourcecode 5.1 Implementasi *Load Data*

Penjelasan *source code* dari implementasi *Load Data* adalah sebagai berikut :

1. Pada baris 4 – 11 merupakan *code* dari *library apache commons csv* yang digunakan untuk mengambil data berupa *file csv*.
2. Pada baris 13 merupakan *code* untuk memfilter data yang memiliki tingkat *confidence* tertentu.
3. Pada baris 14 – 17 merupakan *code* untuk mengambil fitur – fitur yang akan digunakan yaitu *brightness*, *bright_t31* dan *frp*.
4. Pada baris 18 merupakan *code* untuk menyimpan data yang telah di ambil dari *file csv* kedalam *list point.*

5.3.2 Normalisasi Min-Max

Pada proses ini dilakukan normalisasi data menggunakan fungsi Normalisasi Min-Max ke dalam *range* 0 sampai 1.

```
1   public class Normalisasi {
2       //normalisasi menggunakan algoritma minmax dengan
    rentan 0-1
3       public void minMax(List<Point> Titik) {
4           double maxBrightness, maxBright_t31, maxFrp;
5           double minBrightness, minBright_t31, minFrp;
6           maxBrightness = maxBright_t31 = maxFrp =
    Double.MIN_VALUE;
7           minBrightness = minBright_t31 = minFrp =
    Double.MAX_VALUE;
8           double nBrightness, nBright_t31, nFrp;
9           double rMax = 1.0;
10          double rMin = 0.0;
11          //min max fitur brightness
12          for (int i = 0; i < Titik.size(); i++) {
13              if (Titik.get(i).getBrightness() >
    maxBrightness) {
14                  maxBrightness =
    Titik.get(i).getBrightness();
15              }
16          }
17          for (int i = 0; i < Titik.size(); i++) {
18              if (Titik.get(i).getBrightness() <
    minBrightness) {
19                  minBrightness =
    Titik.get(i).getBrightness();
20              }
21          }
22          //min max fitur bright_t31
23          for (int i = 0; i < Titik.size(); i++) {
24              if (Titik.get(i).getBright_t31() >
    maxBright_t31) {
25                  maxBright_t31 =
    Titik.get(i).getBright_t31();
26              }
27          }
28          for (int i = 0; i < Titik.size(); i++) {
29              if (Titik.get(i).getBright_t31() <
    minBright_t31) {
30                  minBright_t31 =
    Titik.get(i).getBright_t31();
31              }
32          }
33          //min max fitur frp
34          for (int i = 0; i < Titik.size(); i++) {
35              if (Titik.get(i).getFrp() > maxFrp) {
36                  maxFrp = Titik.get(i).getFrp();
37              }
38          }
39          for (int i = 0; i < Titik.size(); i++) {
40              if (Titik.get(i).getFrp() < minFrp) {
41                  minFrp = Titik.get(i).getFrp();
42              }
43          }
```

```
44          // proses normalisasi
45          for (int i = 0; i < Titik.size(); i++) {
46              Point t = new Point();
47              nBrightness = (Titik.get(i).getBrightness() -
    minBrightness) / (maxBrightness - minBrightness) * ((rMax -
    rMin) + rMin);
48              nBright_t31 = (Titik.get(i).getBright_t31() -
    minBright_t31) / (maxBright_t31 - minBright_t31) * ((rMax -
    rMin) + rMin);
49              nFrp = (Titik.get(i).getFrp() - minFrp) /
    (maxFrp - minFrp) * ((rMax - rMin) + rMin);
50              t.setBrightness(nBrightness);
51              t.setBright_t31(nBright_t31);
52              t.setFrp(nFrp);
53              new Point().update(i, t);
54          }
55      }
56  }
```

Sourcecode 5.2 Implementasi Normalisasi Min-Max

Penjelasan *source code* dari implementasi algoritma Normalisasi adalah sebagai berikut :

1. Pada baris 4 – 10 merupakan *code* inisialisasi parameter yang akan digunakan dalam normalisasi.
2. Pada baris 12 – 43 merupakan *code* untuk mencari nilai *min* dan *max* untuk setiap fitur.
3. Pada baris 44 – 53 merupakan *code* penerapan rumus normalisasi dengan algoritma *min-max*. Kemudian hasil normalisasi akan di-*update* kedalam *list point*.

5.3.3 Algoritma DBSCAN

Pada proses ini dilakukan pengelompokkan data yang sudah dinormalisasi menggunakan metode DBSCAN

```
1   public class Clustering {
2       Distance distance = new Distance();
3       Cluster clus = new Cluster();
4       private final double eps;
5       private final int minPts;
6       private enum PointStatus {
7           NOISE,
8           PART_OF_CLUSTER
9       }
10      final List<List<Point>> allData = new ArrayList<>();
11      final List<Point> noise = new ArrayList<>();
12      final JPanel panel = new JPanel();
13      public Clustering(double eps, int minPts, List<Point>
    titikPanas) {
14          allData.clear();
15          this.eps = eps;
16          this.minPts = minPts;
17          if (eps >= 0.0d && minPts >= 0) {
18              cluster(titikPanas);
19              new Silhouette(allData);
```

```
20              clus.setSize(allData.size());
21              clus.setNoise(noise.size());
22          }
23      }
24      // proses utama clustering
25      public List<List<Point>> cluster(final
    Collection<Point> points) {
26          int index = 0;
27          final Map<Point, PointStatus> visited = new
    HashMap<>();
28          for (final Point point : points) {
29              if (visited.get(point) != null) {
30                  continue;
31              }
32              final List<Point> neighbors =
    getNeighbors(point, points);
33              if (neighbors.size() < minPts) {
33                  visited.put(point, PointStatus.NOISE);
34                  noise.add(point);
35              } else {
36                  final Cluster cluster = new Cluster();
37                  expandCluster(cluster, point, neighbors,
    points, visited); //cluster meluas
38                  allData.add(index, cluster.get());
39                  index++;
40              }
41          }
42          System.out.println("Jumlah Cluster yang terbentuk -
    > " + allData.size());
43          return allData;
44      }
45      // memperluas cluster
46      private Cluster expandCluster(
                final Cluster cluster,
                final Point point,
                final List<Point> neighbors,
                final Collection<Point> points,
                final Map<Point, PointStatus> visited) {
47          cluster.add(point);
48          visited.put(point, PointStatus.PART_OF_CLUSTER);
49          List<Point> seeds = new
    ArrayList<Point>(neighbors);
50          int index = 0;
51          while (index < seeds.size()) {
52              final Point current = seeds.get(index);
53              PointStatus pStatus = visited.get(current);
54              if (pStatus == null) {
55                  final List<Point> currentNeighbors =
    getNeighbors(current, points);
56                  if (currentNeighbors.size() >= minPts) {
57                      seeds = merge(seeds, currentNeighbors);
58                  }
59              }
60              if (pStatus != PointStatus.PART_OF_CLUSTER) {
61  visited.put(current, PointStatus.PART_OF_CLUSTER);
62                  cluster.add(current);
63                  noise.remove(current);
64              }
65              index++;
```

```
66          }
67          return cluster;
68      }
69      //mencari ketetanggan
70      private List<Point> getNeighbors(final Point point,
71  final Collection<Point> points) {
72          final List<Point> neighbors = new
    ArrayList<Point>();
73          for (final Point neighbor : points) {
74              if (distance.hitung(neighbor, point) <= eps) {
75                  neighbors.add(neighbor);
76              }
77          }
78          return neighbors;
79      }
80      //penggabungan tetangga
81      private List<Point> merge(final List<Point> one, final
    List<Point> two) {
82          final Set<Point> oneSet = new HashSet<Point>(one);
83          for (Point item : two) {
84              if (!oneSet.contains(item)) {
85                  one.add(item);
86              }
87          }
88          return one;
89      }
90  }
```

Sourcecode 5.3 Implementasi Algoritma DBSCAN

Penjelasan *source code* dari implementasi algoritma *clustering* adalah sebagai berikut :

1. Pada baris 2 – 12 merupakan *code* inisialisasi parameter yang digunakan dalam proses *clustering* menggunakan algoritma DBSCAN.
2. Pada baris 13 – 23 merupakan *code construct* yang berfungsi untuk melakukan pengosongan *list allData* setiap kali perintah dijalankan, dan juga mengecek parameter *eps* dan *minpts*.
3. Pada baris 24 – 44 merupakan *code* utama dalam proses *clustering* menggunakan algoritma DBSCAN. *Code* ini akan mengambil sebuah titik p dan mencari titik-titik mana saja yang termasuk tetangga dari titik p (memanggil *method getNeighbors*), dimana jarak dari titik-titik yang menjadi tetangga harus <= *eps*. Apabila titik p memiliki jumlah tetangga < *minPts*, maka titik akan ditandai sebagai *noise* dan memasukkanya ke dalam *list noise*. Apabila jumlah tetangga titik p >= *minPts* maka *method* ini akan menjalankan *method* berikutnya yaitu *expandCluster.*
4. Pada baris 46 – 68 merupakan *code* untuk *method expandCluster. Method* ini akan mengambil titik p' yang merupakan tetangga dari titik p. Kemudian *method* ini akan mencari titik yang bertetangga dengan titik p'. Apabila jumlah tetangga titik p' >= *minPts,* maka *method* ini akan memanggil *method merge* untuk menggabungkan tetangga dari titik p' dengan tetangga dari titik p. Selain itu *method* ini akan mengecek apakah titik yang sedang di proses sudah di tandai sebagai *noise* pada *method* sebelumnya. Apabila titik tersebut sudah ditandai sebagai *noise*, maka *method* ini akan

mengubahnya menjadi *PART_OF_CLUSTER* dan menghapusnya dari *list noise*.

5. Pada baris 69 – 79 merupakan *code* untuk mencari tetangga dari titik p dengan syarat jarak antara kedua titik <= *eps*.
6. Pada baris 80 – 90 merupakan *code* untuk menggabungkan tetangga kedalam *cluster* yang sama.

5.3.4 Euclidean Distance

Proses ini mencari jarak antara titik i dengan titik lain menggunakan fungsi *Euclidian Distance*.

```
1   public class Distance {
2       //mencari jarak (euclidian)
3       public double hitung(Point neighbor, Point point) {
4           double brightness = point.getBrightness() -
    neighbor.getBrightness();
5           double bright_t31 = point.getBright_t31() -
    neighbor.getBright_t31();
6           double frp = point.getFrp() - neighbor.getFrp();
7           final double distance = Math.pow(brightness, 2) +
    Math.pow(bright_t31, 2) + Math.pow(frp, 2);
8           return Math.sqrt(distance);
9       }
10  }
```

Sourcecode 5.4 Implementasi *Euclidean Distance*

Penjelasan *source code* dari implementasi fungsi *Euclidean Distance* adalah sebagai berikut :

1. Pada baris 3 – 8 merupakan *code* untuk mencari jarak antar suatu titik menggunakan fungsi *Euclidian Distance.*

5.3.5 Silhouette Coefficient

Proses ini mencari nilai *Silhouette Coefficient* untuk mengetahui kualitas dari *cluster* yang dibentuk.

```
1   public class Silhouette {
2       Distance distance = new Distance();
3       Cluster clus = new Cluster();
4       private List<List<Point>> allData;
5       private enum PointStatus {
6           VISITED
7       }
8       private int size;
9       private double silhouette;
10      private double meanA;
11      private double meanB;
12      public Silhouette(List<List<Point>> allData) {
13          this.allData = allData;
14          jumlahData();
15          SilhouetteIndex();
16          clus.setSilhouette(silhouette);
17      }
18      private int jumlahData() {
```

```
19          for (List<Point> p : allData) {
20              size += p.size();
21          }
22          return size;
23      }
24      public double SilhouetteIndex() {
25          if (allData.size() <= 1) {
26              return silhouette = 0;
27          }
28          for (int i = 0; i < allData.size(); i++) {
29              for (int o = 0; o < allData.get(i).size(); o++)
    {
30                  double a = 0;
31                  if (allData.get(i).size() <= 1) {
32                      a = 0;
33                  } else {
34                      for (Point p : allData.get(i)) {
                            if (allData.get(i).get(o) != p) {
35                              a +=
    distance.hitung(allData.get(i).get(o), p);
36                          }
37                      }
38                      a = a / (allData.get(i).size() - 1);
39                  }
40                  double b = 0;
41                  double min = Double.MAX_VALUE;
42                  double s = 0;
43                  for (List<Point> l : allData) {
44                      if (allData.get(i) != l) {
45                          for (Point z : l) {
46                              b +=
    distance.hitung(allData.get(i).get(o), z);
47                          }
48                          b = b / l.size();
49                          if (b <= min) {
50                              min = b;
51                          }
52                      }
53                  }
54                  s = (min - a) / (Math.max(a, min));
55                  silhouette = silhouette + s;
56                  meanA = meanA + a;
57                  meanB = meanB + min;
58              }
59          }
60          silhouette = silhouette / size;
61          meanA = meanA / size;
62          meanB = meanB / size;
63          System.out.println("Jarak Intra Cluster : " +
    meanA);
64          System.out.println("Jarak Inter Cluster : " +
    meanB);
65          System.out.println("Silhouette Index : " +
    silhouette);
66          return silhouette;
67      }
```

Sourcecode 5.5 Implementasi *Silhouette Coefficient*

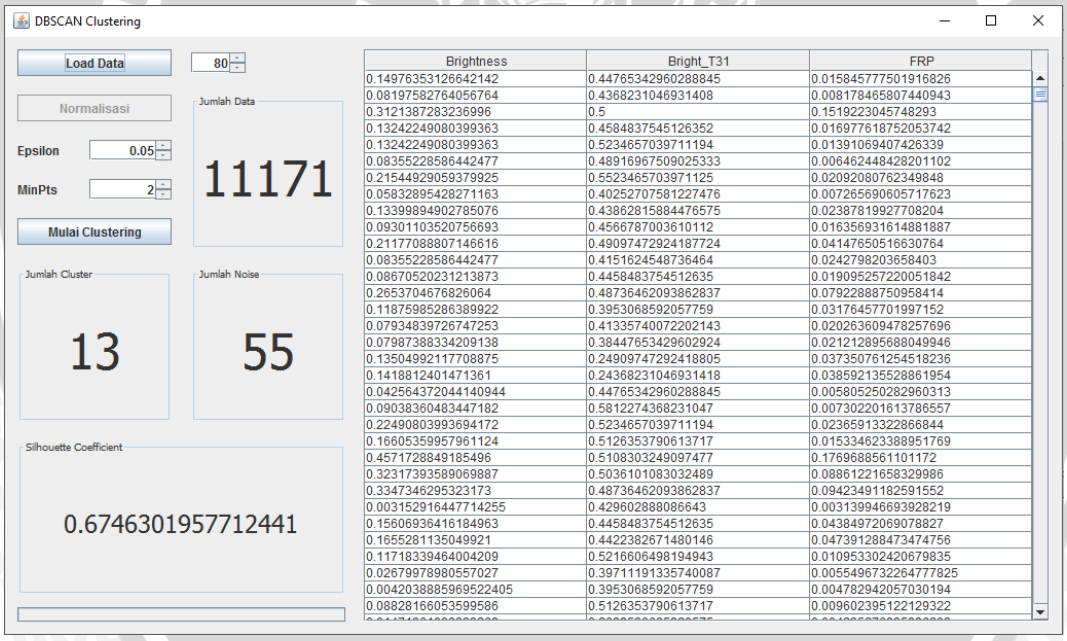
Penjelasan *source code* dari implementasi perhitungan *silhouette coefficient* adalah sebagai berikut :

1. Pada baris 2 – 11 merupakan *code* inisialisasi parameter
2. Pada baris 18 – 23 merupakan *code* untuk menghitung jumlah data yang telah dikelompokkan
3. Pada baris 31 – 39 merupakan *code* untuk mencari rata-rata jarak titik i dengan titik lain di dalam satu *cluster* atau a(i).
4. Pada baris 40 – 50 merupakan *code* untuk mencari rata-rata jarak titik i dengan titik di *cluster* lain atau b(i).
5. Pada baris 54 – 60 merupakan *code* untuk mencari nilai *Silhouette Coefficient* setiap titik i dan *Silhouette Coefficient* akhir.

5.4 Implementasi Antarmuka

Subbab ini membahas mengenai hasil implementasi antarmuka dalam bentuk aplikasi *desktop* menggunakan bahasa pemrograman *Java*.

Hasil implementasi antarmuka dapat dilihat pada Gambar 5.1.



Gambar 5.1 Implementasi Antarmuka

Gambar 5.1 merupakan tampilan program ketika dijalankan. Untuk melakukan proses *clustering,* langkah pertama yang harus dilakukan adalah men-*set* filter *confidence* (nilai *default* adalah 80). Kemudian klik tombol *Load Data,* dan pilih *file csv* yang akan digunakan. Setelah itu *set* nilai parameter *minPts* dan *eps*. Setelah parameter di-*set,* maka langkah selanjutnya adalah menekan tombol normalisasi untuk menormalkan data. Kemudian tekan tombol *clustering* untuk memulai proses *clustering*.

BAB 6 PENGUJIAN

Pada bab ini dijelaskan mengenai proses uji coba sistem *clustering* menggunakan metode DBSCAN. Pengujian yang dilakukan adalah pengujian kualitas *cluster* yang terbentuk.

6.1 Pengujian Kualitas Hasil Clustering

Pengujian kualitas hasil *clustering* dilakukan untuk mengetahui kualitas dari *cluster* yang dibentuk oleh algoritma DBSCAN. Pengujian dilakukan dengan cara mengubah nilai parameter *minPts* dan *eps.*

6.1.1 Tujuan Pengujian Kualitas Hasil Clustering

Tujuan dari melakukan pengujian kualitas *cluster* adalah untuk mengetahui bagaimana kualitas dari *cluster* yang dibentuk oleh algoritma DBSCAN. Selain itu pengujian ini dilakukan untuk mengetahui bagaimana pengaruh parameter *minPts* dan *eps* terhadap jumlah *cluster* yang dihasilkan, dan jumlah titik yang dianggap *noise*.

6.1.2 Prosedur Pengujian Kualitas Hasil Clustering

Prosedur pengujian kualitas hasil *clustering* adalah proses pengujian yang dilakukan dengan cara mencari nilai parameter *minPts* dan *eps* yang menghasilkan nilai *Silhouette Coefficient* tertinggi. Pengujian ini akan dilakukan dengan cara mengubah nilai parameter *minPts,* kemudian parameter *minPts* dengan nilai *Silhouette Coefficient* tertinggi akan digunakan pada pengujian parameter *eps*.

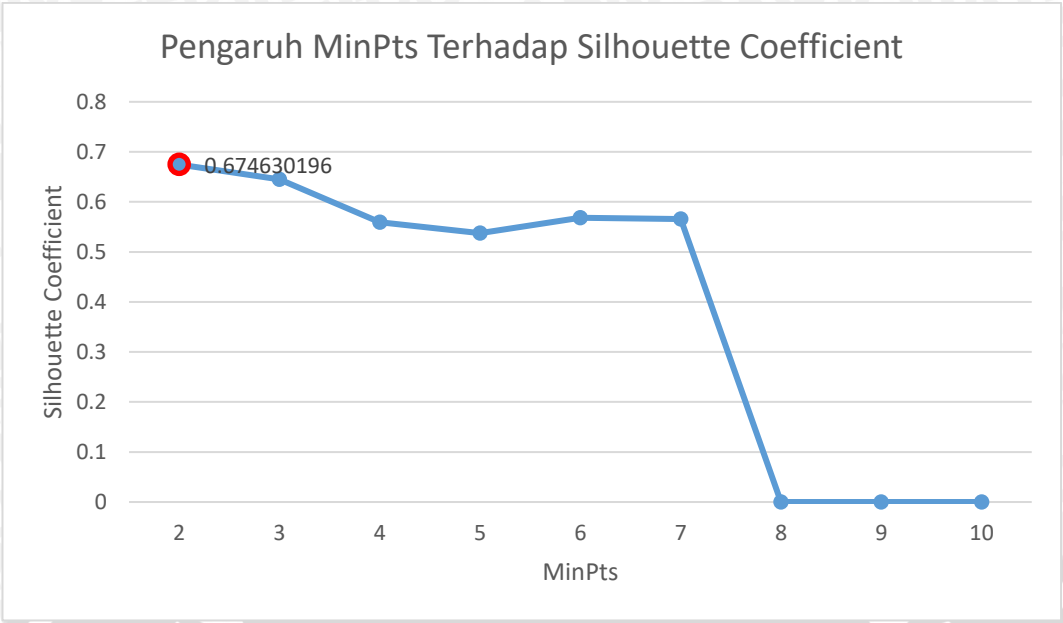
6.1.2.1 Skenario Uji Coba Parameter MinPts

Pada skenario ini dilakukan pengujian parameter *minPts* terhadap 11171 *records*. Nilai parameter *minPts* yang akan diuji adalah 2 – 10, sedangkan parameter *eps* yang akan digunakan tetap yaitu 0.05. Hasil dari pengujian dapat dilihat pada Tabel 6.1.

Tabel 6.1 Hasil Pengujian Parameter *MinPts*

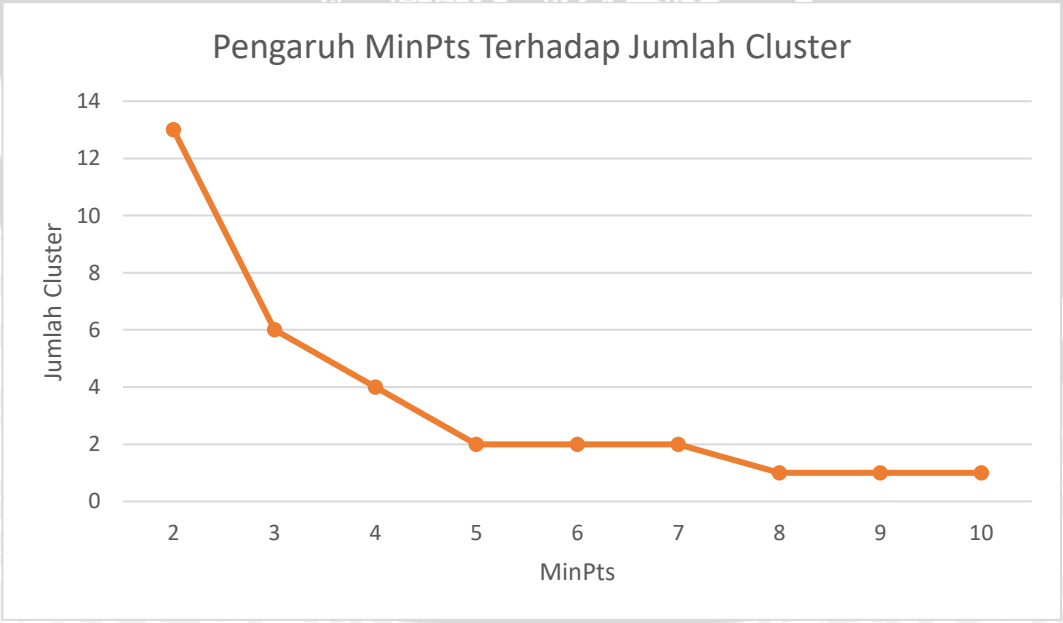
Eps	minPts	Jumlah Cluster	Jumlah Noise	Silhouette Coefficient
0.05	2	13	55	0.674630196
0.05	3	6	72	0.645304968
0.05	4	4	87	0.559303436
0.05	5	2	103	0.537567959
0.05	6	2	113	0.568013513
0.05	7	2	118	0.565573614
0.05	8	1	131	0.0
0.05	9	1	150	0.0
0.05	10	1	154	0.0

Selanjutnya Tabel 6.1 akan diubah kedalam bentuk grafik, untuk mengetahui pola pengaruh *minPts* terhadap nilai *Silhouette Coefficient,* jumlah *cluster* dan jumlah *noise*. Grafik dapat dilihat pada Gambar 6.1, Gambar 6.2 dan Gambar 6.3.



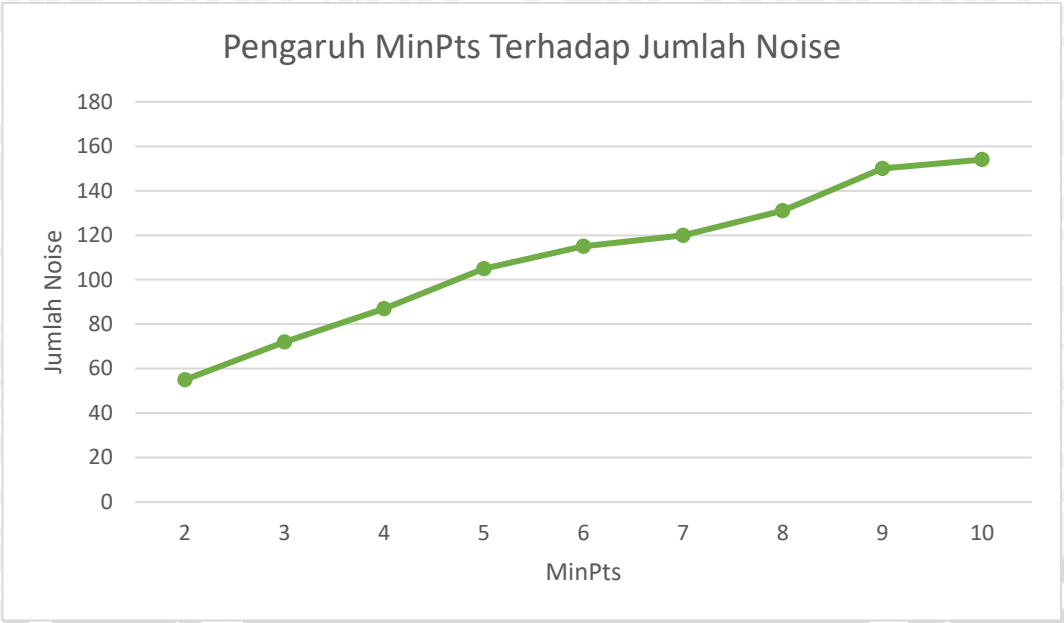
Gambar 6.1 Grafik Pengaruh *MinPts* Terhadap *Silhouette Coefficient*

Pada Gambar 6.1 merupakan grafik pengaruh *minPts* terhadap *Silhouette Coefficient* dimana nilai tertinggi yaitu 0.674630196 diperoleh pada saat nilai *minPts* = 2. Sedangkan pada saat *minPts* = 8, 9 dan 10, nilai *Silhouette Coefficientnya* = 0 dikarenakan jumlah *cluster* yang terbentuk hanya 1 sehingga tidak bisa dilakukan perhitungan nilai *Silhouette Coefficient*.



Gambar 6.2 Grafik Pengaruh *MinPts* terhadap Jumlah *Cluster*

Pada Gambar 6.2 merupakan grafik pengaruh *minPts* terhadap jumlah *cluster* yang dibentuk. Grafik membentuk sebuah pola bahwa semakin besar nilai *minPts* maka jumlah *cluster* yang terbentuk akan semakin sedikit.



Gambar 6.3 Grafik Pengaruh *MinPts* Terhadap Jumlah *Noise*

Pada Gambar 6.3 merupakan grafik pengaruh *minPts* terhadap jumlah titik yang dianggap sebagai *noise*. Grafik membentuk sebuah pola bahwa semakin besar nilai *minPts* maka jumlah *noise* akan semakin banyak.

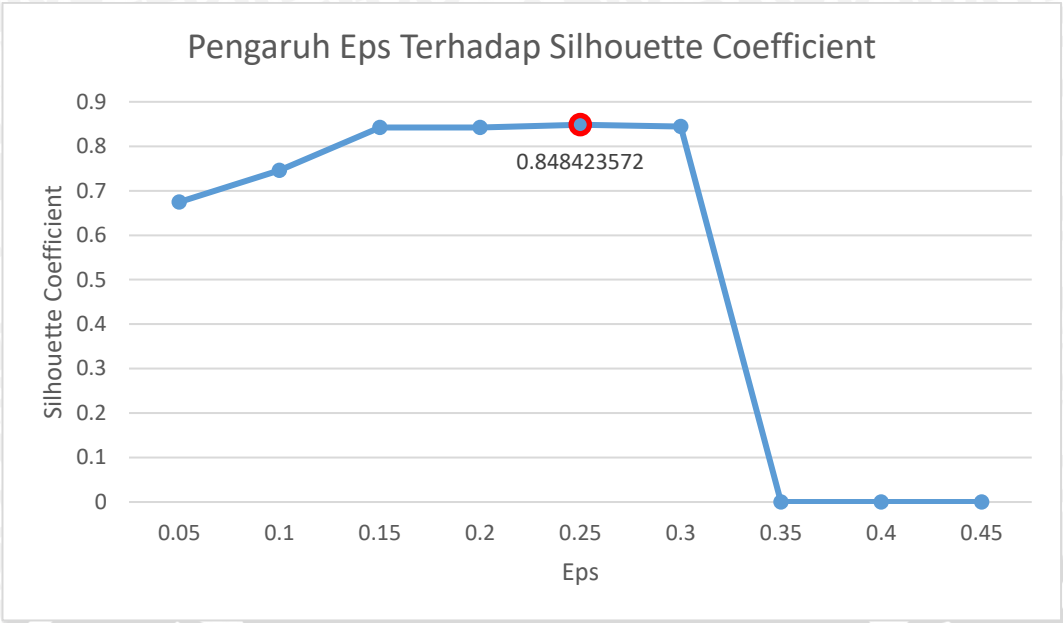
6.1.2.2 Skenario Uji Coba Parameter Eps

Pada skenario ini dilakukan pengujian parameter *eps* terhadap 11171 *records*. Nilai parameter *eps* yang akan diuji adalah 0.05 – 0.45 dengan kenaikan 0.05 setiap pengujian, sedangkan parameter *minPts* yang digunakan adalah *minPts* dengan nilai *Silhouette Coefficient* tertinggi yang dilakukan pada pengujian sebelumnya yaitu 2. Hasil pengujian parameter *eps* dapat dilihat pada Tabel 6.2.

Tabel 6.2 Hasil Pengujian Parameter *Eps*

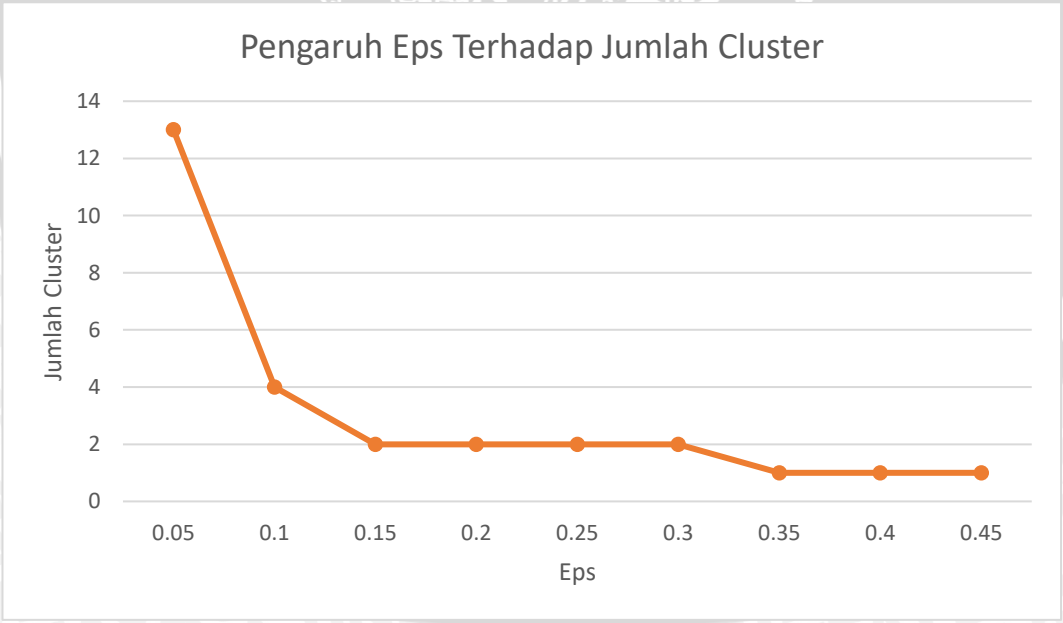
Eps	minPts	Jumlah Cluster	Jumlah Noise	Silhouette coefficient
0.05	2	13	55	0.674630196
0.10	2	4	10	0.745448544
0.15	2	2	2	0.842423282
0.20	2	2	2	0.842423282
0.25	2	2	1	0.848423572
0.30	2	2	0	0.844771013
0.35	2	1	0	0.0
0.40	2	1	0	0.0
0.45	2	1	0	0.0

Selanjutnya Tabel 6.2 akan diubah kedalam bentuk grafik, untuk mengetahui pola pengaruh *eps* terhadap nilai *Silhouette Coefficient,* jumlah *cluster* dan jumlah *noise*. Grafik dapat dilihat pada Gambar 6.4, Gambar 6.5 dan Gambar 6.6.



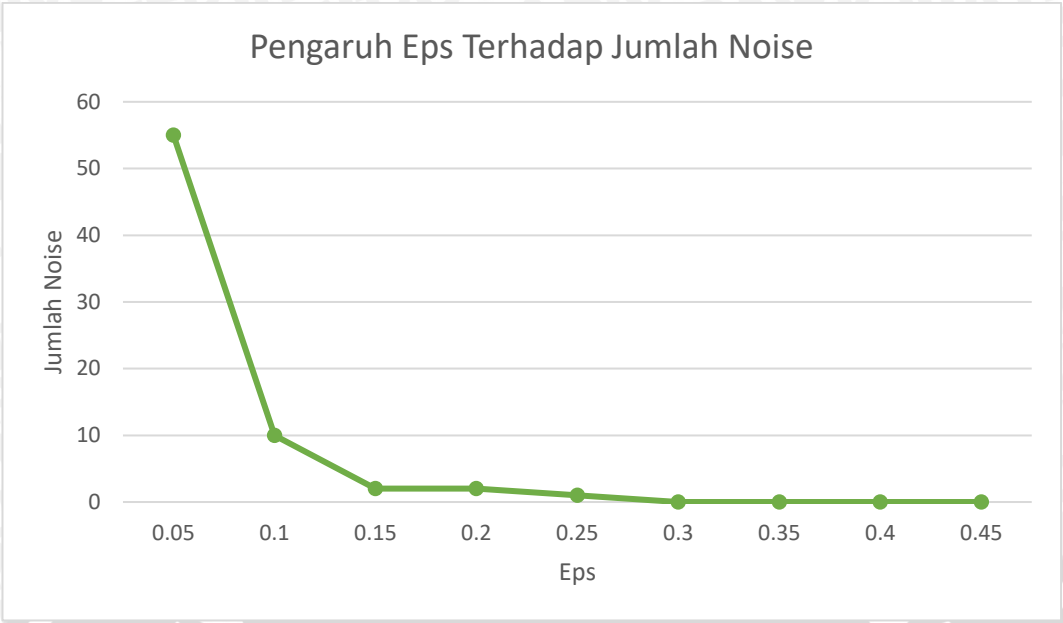
Gambar 6.4 Grafik Pengaruh *Eps* Terhadap *Silhouette Coefficient*

Pada Gambar 6.4 merupakan grafik pengaruh *eps* terhadap *Silhouette Coefficient* dimana nilai tertinggi yaitu 0.848423572 diperoleh pada saat nilai *eps* = 0.25. Sedangkan pada saat *eps* = 0.35, 0.4 dan 0.45, nilai *Silhouette Coefficientnya* = 0 dikarenakan jumlah *cluster* yang terbentuk hanya 1 sehingga tidak bisa dilakukan perhitungan nilai *Silhouette Coefficient*.



Gambar 6.5 Grafik Pengaruh *Eps* Terhadap Jumlah *Cluster*

Pada Gambar 6.5 merupakan grafik pengaruh *eps* terhadap jumlah *cluster* yang dibentuk. Grafik membentuk sebuah pola bahwa semakin besar nilai *eps* maka jumlah *cluster* yang terbentuk akan semakin sedikit.



Gambar 6.6 Grafik Pengaruh *Eps* Terhadap Jumlah *Noise*

Pada Gambar 6.6 merupakan grafik pengaruh *eps* terhadap jumlah titik yang dianggap sebagai *noise*. Grafik membentuk sebuah pola bahwa semakin besar nilai *eps* maka jumlah *noise* akan semakin sedikit.

6.1.3 Hasil Pengujian Kualitas Hasil Clustering

Dari kedua skenario uji coba yang telah dilakukan, diperoleh bahwa kombinasi nilai parameter *minPts* = 2 dan *eps* = 0.25 menghasilkan nilai *Silhouette Coefficient* tertinggi yaitu 0.848423572 dengan jumlah *cluster* yang terbentuk sebanyak 2, sedangkan jumlah *noise* sebanyak 1.

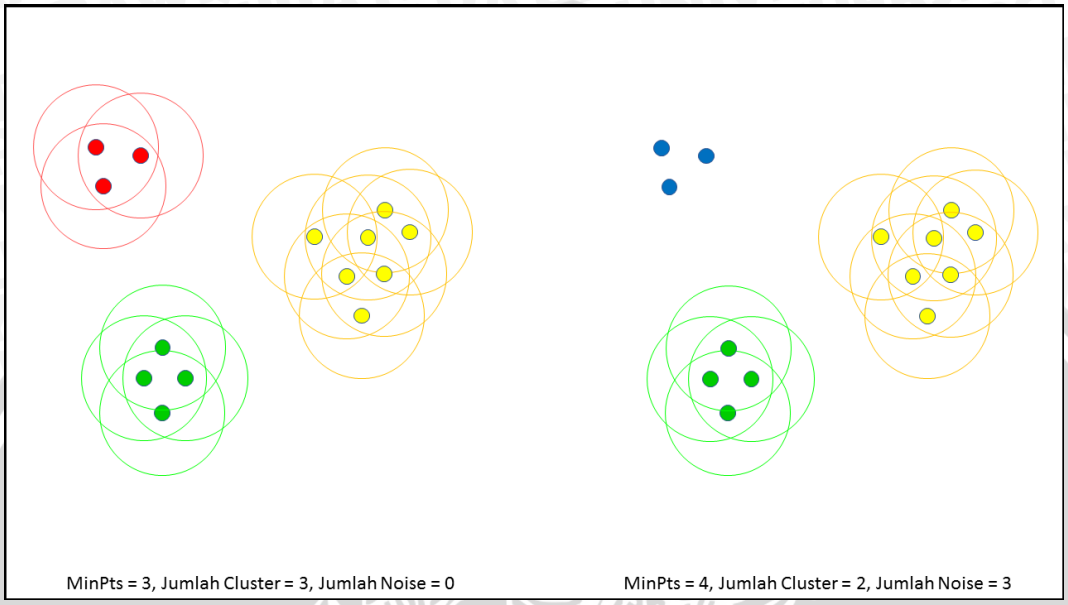
6.1.4 Analisis Pengujian Kualitas Hasil Clustering

Proses analisis dari hasil pengujian kualitas hasil clustering dilakukan dengan mengamati pola yang dihasilkan oleh metode DBSCAN.

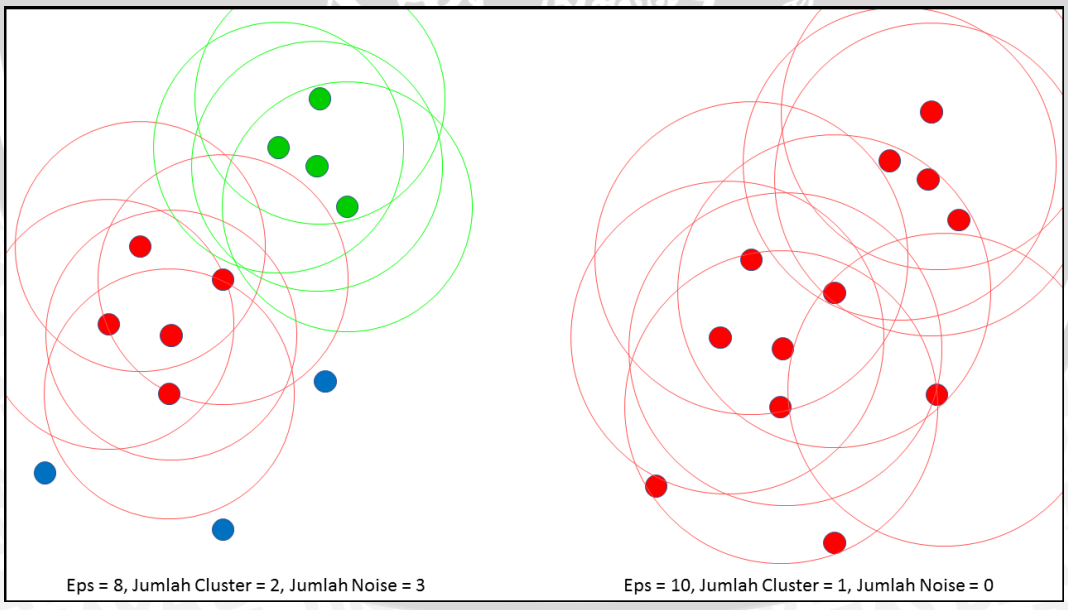
Dari hasil pengujian yang dilakukan, perubahan nilai parameter *minPts* dan *eps* sangat mempengaruhi kualitas *cluster* yang dibentuk. Dari hasil tersebut dapat diambil kesimpulan bahwa semakin besar nilai *minPts* maka nilai *Silhouette Coefficient* akan relatif turun. Hal ini kemungkinan disebabkan ketika *minPts* dinaikkan terdapat *cluster* yang tidak memenuhi syarat *minPts* tersebut. Sehingga hilangnya *cluster* tersebut berpeluang mengakibatkan turunya jarak *inter – cluster* sehingga nilai *Silhouette Coefficient* relatif akan turun. Selain itu jumlah *cluster* yang dibentuk akan semakin kecil, tetapi jumlah *noise* akan semakin banyak. Hal ini dikarenakan semakin besar nilai *minPts* maka kriteria terbentuknya sebuah *cluster* akan semakin sulit, sehingga peluang titik dianggap sebagai *noise* akan semakin besar seperti yang diilustrasikan pada Gambar 6.7.

Sedangkan semakin besar nilai *eps* maka nilai *Silhouette Coefficient* seharusnya relatif turun. Hal ini disebabkan karena semakin luas cakupan dari

suatu *cluster* akan meningkatkan rata-rata jarak antar titik dalam satu *cluster*. Selain itu, semakin besar *eps* akan menghasilkan jumlah *cluster* dan jumlah *noise* semakin sedikit. Hal ini dikarenakan semakin besar nilai *eps* maka, peluang sebuah titik masuk kedalam suatu *cluster* akan semakin besar sehingga *cluster* yang terbentuk akan semakin sedikit dan peluang sebuah titik dianggap sebagai *noise* akan semakin kecil juga seperti yang diilustrasikan pada Gambar 6.8.



Gambar 6.7 Ilustrasi Pengaruh Parameter *MinPts*



Gambar 6.8 Ilustrasi Pengaruh Parameter *Eps*

6.1.4.1 Tingkat Potensi Terjadinya Kebakaran

Tabel 6.3 Tingkat Potensi Terjadinya Kebakaran

Cluster	Rata-Rata Brightness (K)
Cluster 2	478.5
Cluster 1	336.3

Pada Tabel 6.3 merupakan hasil *clustering* menggunakan algoritma DBSCAN dengan *minPts* = 2 dan *eps* = 0.25 yang menghasilkan 2 *cluster*. Dari hasil tersebut *cluster* ke 2 memiliki rata-rata suhu permukaan (*brightness*) lebih tinggi dibandingkan dengan *cluster* ke 1, yang artinya *cluster* ke 2 memiliki tingkat potensi kebakaran lebih tinggi dibandingkan *cluster* ke 1.

BAB 7 PENUTUP

Bab ini membahas tentang kesimpulan dan saran dari penelitian yang telah dilakukan berdasarkan hasil perancangan, implementasi, dan pengujian sistem.

7.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi dan pengujian yang telah dilakukan maka dapat diambil kesimpulan sebagai berikut :

a. Algoritma DBSCAN dapat diimplementasikan untuk mengelompokkan titik panas ke dalam 2 tingkat potensi terjadinya kebakaran.

b. *Clustering* menggunakan algoritma DBSCAN menghasilkan hasil yang cukup bagus dengan cukup tingginya nilai *Silhouette Coefficient* pada proses pengujian.

7.2 Saran

Saran yang dapat diberikan untuk pengembangan sistem dalam penelitian selanjutnya adalah :

a. Dapat dilakukan penelitian lebih lanjut untuk menentukan nilai parameter *minPts* dan *eps* sehingga kualitas dari *cluster* yang dibentuk akan menjadi lebih maksimal.

DAFTAR PUSTAKA

Devi, N. M. A. S., Putra, I. K. G. D. & Sukarsa, I. M., 2015. *Implementasi Metode Clustering DBSCAN pada Proses Pengambilan Keputusan,* Bukit Jimbaran: Lontar Komputer Vol.6, No.3.

Fadli, A., 2011. *Konsep Data Mining.* [Online] Available at: http://ilmukomputer.org/2011/03/14/konsep-data-minning/ [Accessed 2 Maret 2016].

Fatkhiyah, E., 2012. *Rancangan Proses Training Untuk Mendukung Penentuan Kualitas Air Minum Kemasan,* Yogyakarta: Prosiding Seminar Nasional Aplikasi Sains & Teknologi (SNAST) Periode III.

Handoyo, R., Mangkudjaja, R. & Nasution, S. M., 2014. *Perbandingan Metode Clustering Menggunakan Metode Single Linkage dan K - Means Pada Pengelompokkan Dokumen,* Bandung: JSM STMIK Mikroskil Vol. 15, No. 2.

Heryalianto, S. C., 2006. *Studi Tentang Sebaran Titik Panas (Hotspot) Sebagai Penduga Kebakaran Hutan Dan Lahan Di Propinsi Kalimantan Barat Tahun 2003 Dan Tahun 2004.* Bogor: Institut Pertanian Bogor.

Kaufman, L. & Rousseeuw, P. J., 1990. *Finding Groups in Data : An Introduction to Cluster Analysis.* Canada: John Wiley & Sons, Inc.

Liman, N. S., I. S. & I. S. M., 2014. *Implementasi Algoritma Density-Based Spatial Clustering of Application with Noise (DBSCAN) Pada Mesin Pencari Dokumen Hasil Penelitian,* Malang: Teknik Informatika Universtias Brawijaya.

Putri, W. D., 2015. *BNPB Catat Kerugian Akibat Kebakaran Hutan 2015 Rp 221 Triliun.* [Online] Available at: http://www.republika.co.id/berita/nasional/umum/15/12/20/nzms82359-bnpb-catat-kerugian-akibat-kebakaran-hutan-2015-rp-221-triliun [Accessed 2 Maret 2016].

Usman, M., 2014. *Spatial Clustering Berbasis Densitas Untuk Persebaran Titik Panas Sebagai Indikator Kebakaran Hutan Dan Lahan Gambut Di Sumatera,* Bogor: Pascasarjana Institut Pertanian Bogor.

Zakariya, A. Z., Djunaidy, A. & Kusumawardani, R. P., 2012. *Pembuatan Aplikasi Pendeteksi Anomali Pada Pola Konsumsi Listrik Pelanggan Kota Surabaya Menggunakan Algoritma Klasterisasi Berbasis Densitas,* Surabaya: Jurnal Teknik Pomits Vol. 1, 1-5.

LAMPIRAN

A.1 Data Asli Titik Panas

latitude	longitude	brightness	scan	track	acq_date	acq_time	satellite	instrument	confidence	version	bright_t31	frp
3.483	117.039	311.3	1	1	6/1/2015	244	Terra	MODIS	31	6.1	292.4	6
2.47	117.844	312.8	1	1	6/1/2015	245	Terra	MODIS	56	6.1	295.9	6
0.811	117.535	309.2	1	1	6/1/2015	245	Terra	MODIS	30	6.1	289.4	4.8
-0.443	116.044	311.8	1	1	6/1/2015	245	Terra	MODIS	62	6.1	291.7	4.7
-1.907	115.853	312.7	1	1	6/1/2015	246	Terra	MODIS	57	6.1	294.6	5.4
-8.481	121.27	315	2.4	1.5	6/1/2015	247	Terra	MODIS	34	6.1	295.7	31
-8.114	112.929	315.4	1.1	1	6/1/2015	248	Terra	MODIS	44	6.1	295.5	12.4
4.422	95.899	319.6	1.4	1.2	6/1/2015	423	Terra	MODIS	60	6.1	299.8	14.8
-1.121	102.808	301.4	3.1	1.7	6/1/2015	1507	Terra	MODIS	33	6.1	287.3	18.6
-1.122	102.801	302.6	3.1	1.7	6/1/2015	1507	Terra	MODIS	27	6.1	287.5	19.3
-6.589	140.871	310.3	1.4	1.2	6/1/2015	401	Aqua	MODIS	0	6.1	295.8	8.5
-3.683	139.07	307	1.7	1.3	6/1/2015	402	Aqua	MODIS	52	6.1	293.2	9.5
-9.84	120.51	319.7	1	1	6/1/2015	539	Aqua	MODIS	60	6.1	301.7	6.7
-9.369	119.941	316.7	1	1	6/1/2015	539	Aqua	MODIS	55	6.1	300.3	6.3
-9.371	119.932	317.6	1	1	6/1/2015	539	Aqua	MODIS	54	6.1	300	6.4
-9.383	119.326	322.1	1	1	6/1/2015	539	Aqua	MODIS	65	6.1	300.4	12.5
-2.581	121.385	313.1	1.2	1.1	6/1/2015	541	Aqua	MODIS	23	6.1	292.3	7.6
-2.236	119.255	318.3	1	1	6/1/2015	541	Aqua	MODIS	74	6.1	298.5	8.4
-1.909	115.846	312.6	1.3	1.1	6/1/2015	541	Aqua	MODIS	36	6.1	293.3	6.6
-0.557	116.174	338.7	1.2	1.1	6/1/2015	542	Aqua	MODIS	90	6.1	293.3	43.4

-0.552	116.177	325.8	1.2	1.1	6/1/2015	542	Aqua	MODIS	81	6.1	292.7	22.4
-0.554	116.167	316.5	1.2	1.1	6/1/2015	542	Aqua	MODIS	71	6.1	292.4	10.7
0.828	117.639	313.2	1	1	6/1/2015	542	Aqua	MODIS	64	6.1	295.7	5.5
1.108	117.864	315.8	1	1	6/1/2015	542	Aqua	MODIS	56	6.1	294.9	8.4
2.24	117.928	313.2	1	1	6/1/2015	542	Aqua	MODIS	65	6.1	296	5.5
2.239	117.919	312.9	1	1	6/1/2015	542	Aqua	MODIS	64	6.1	297	5.2
3.897	117.019	315.8	1	1	6/1/2015	543	Aqua	MODIS	59	6.1	300.3	7.6
4.006	117.333	317.2	1	1	6/1/2015	543	Aqua	MODIS	70	6.1	298.6	7.1
4.085	117.114	316.4	1	1	6/1/2015	543	Aqua	MODIS	64	6.1	299.8	7.8
-2.838	102.855	312	3.2	1.7	6/1/2015	719	Aqua	MODIS	52	6.1	293.4	28.5
-2.858	101.446	369.6	2.5	1.5	6/1/2015	720	Aqua	MODIS	100	6.1	296.2	416.1
-2.847	101.46	320.6	2.5	1.5	6/1/2015	720	Aqua	MODIS	76	6.1	292.3	45.3
-2.761	101.469	314.9	2.5	1.5	6/1/2015	720	Aqua	MODIS	68	6.1	294	30.3
-2.189	100.957	310.7	2.4	1.5	6/1/2015	720	Aqua	MODIS	58	6.1	293.2	16.7
-1.127	102.809	313.2	3.4	1.7	6/1/2015	720	Aqua	MODIS	56	6.1	289.6	47.2
-1.118	102.802	316.4	3.4	1.7	6/1/2015	720	Aqua	MODIS	0	6.1	286.8	58
-1.124	102.018	311	3	1.6	6/1/2015	720	Aqua	MODIS	58	6.1	291.9	26
-1.123	102.01	314.6	3	1.6	6/1/2015	720	Aqua	MODIS	67	6.1	292.5	39.4
-0.817	101.63	319.7	2.8	1.6	6/1/2015	720	Aqua	MODIS	74	6.1	293.6	52.4
-0.247	101.881	318.3	3	1.7	6/1/2015	720	Aqua	MODIS	66	6.1	293.3	47.9
-0.242	101.901	328.9	3	1.7	6/1/2015	720	Aqua	MODIS	79	6.1	291.1	100.7
0.362	102.205	311	3.3	1.7	6/1/2015	720	Aqua	MODIS	30	6.1	290.8	33

A.2 Keterangan Setiap Fitur

Attribute	Short Description	Long Description
Latitude	Latitude	Center of 1km fire pixel but not necessarily the actual location of the fire as one or more fires can be detected within the 1km pixel.
Longitude	Longitude	Center of 1km fire pixel but not necessarily the actual location of the fire as one or more fires can be detected within the 1km pixel.
Brightness	Brightness temperature 21 (Kelvin)	Channel 21/22 brightness temperature of the fire pixel measured in Kelvin.
Scan	Along Scan pixel size	The algorithm produces 1km fire pixels but MODIS pixels get bigger toward the edge of scan. Scan and track reflect actual pixel size.
Track	Along Track pixel size	The algorithm produces 1km fire pixels but MODIS pixels get bigger toward the edge of scan. Scan and track reflect actual pixel size.
Acq_Date	Acquisition Date	Data of MODIS acquisition.
Acq_Time	Acquisition Time	Time of acquisition/overpass of the satellite (in UTC).
Satellite	Satellite	A = Aqua and T = Terra.
Confidence	Confidence (0-100%)	This value is based on a collection of intermediate algorithm quantities used in the detection process. It is intended to help users gauge the quality of individual hotspot/fire pixels. Confidence estimates range between 0 and 100% and are assigned one of the three fire classes (low-confidence fire, nominal-confidence fire, or high-confidence fire).
Version	Version (Collection and source)	Version identifies the collection (e.g. MODIS Collection 6) and source of data processing: Near Real-Time (NRT suffix added to collection) or Standard Processing (collection only). "6.0NRT" - Collection 6 NRT processing. "6.0" - Collection 6 Standard processing. Find out more on collections and on the differences between FIRMS data sourced from LANCE FIRMS and University of Maryland.
Bright_T31	Brightness temperature 31 (Kelvin)	Channel 31 brightness temperature of the fire pixel measured in Kelvin.
FRP	Fire Radiative Power (MW - megawatts)	Depicts the pixel-integrated fire radiative power in MW (megawatts).