TUGAS AKHIR – KI1502

IMPLEMENTASI ALGORITMA PENENTUAN PARAMETER DENSITAS PADA METODE DBSCAN UNTUK PENGELOMPOKAN DATA

AKHMAD BAKHRUL ILMI
NRP 5111100087

Dosen Pembimbing I
Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.

Dosen Pembimbing II
Diana Purwitasari, S.Kom, M.Sc.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016

UNDERGRADUATE THESIS – KI1502

A MODIFICATION FOR DETERMINING DENSITY PARAMETER VALUES ON DBSCAN ALGORITHM

AKHMAD BAKHRUL ILMI
NRP 5111100087

Supervisor I
Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.

Supervisor II
Diana Purwitasari, S.Kom, M.Sc.

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2016

KATA PENGANTAR

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Segala puji dan syukur, kehadirat Allah Subhanahu wa ta'ala yang telah memberikan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul "IMPLEMENTASI ALGORITMA PENENTUAN PARAMETER DENSITAS PADA METODE DBSCAN UNTUK PENGELOMPOKAN DATA".

Pengerjaan tugas akhir ini adalah momen bagi penulis untuk mengeluarkan seluruh kemampuan, hasrat, dan keinginan yang terpendam di dalam hati mulai dari masuk kuliah hingga lulus sekarang ini, lebih tepatnya di kampus Teknik Informatika Institut Teknologi Sepuluh Nopember Surabaya.

Dalam pelaksanaan dan pembuatan tugas akhir ini tentunya sangat banyak bantuan-bantuan yang penulis terima dari berbagai pihak. Melalui lembar ini, penulis ingin secara khusus menyampaikan ucapan terima kasih kepada:

1. Allah SWT dan Nabi Muhammad SAW.
2. Orang tua, Ayah dan Ibu yang selalu mendukung, mendoakan dan memberikan kasih sayang kepada penulis hingga saat ini.
3. Kedua kakak, kerabat dan keluarga besar yang memberikan motivasi penulis untuk menyelesaikan Tugas Akhir ini.
4. Ibu Chastine Fatichah selaku pembimbing I yang telah membantu, membimbing, dan memberikan motivasi kepada penulis dalam menyelesaikan Tugas Akhir ini dengan sabar.
5. Ibu Diana Purwitasari selaku pembimbing II yang juga telah membantu, membimbing, dan memotivasi kepada penulis dalam menyelesaikan Tugas Akhir ini.
6. Bapak Radityo Anggoro selaku dosen koordinator Tugas Akhir yang telah membantu penulis atas segala sesuatu mengenai syarat – syarat dan terlaksananya sidang Tugas Akhir.

7. Bapak Darlis Herumurti selaku Ketua Jurusan Teknik Informatika ITS yang selama ini memberikan bantuan kepada penulis.
8. Ibu Handayani Tjandrasa selaku dosen wali yang telah membimbing penulis dalam pengambilan mata kuliah.
9. Bapak Ahmad Saikhu selaku dosen pembimbing kuliah prakter yang telah membimbing penulis dalam penulisan laporan kerja praktek.
10. Dosen-dosen Teknik Informatika yang dengan sabar mendidik dan memberikan pengalaman baru kepada penulis selama di Teknik Informatika.
11. Staf TU Teknik Informatika ITS yang senantiasa memudahkan segala urusan penulis di jurusan.
12. Teman – teman yang memberikan *support* pada saat sedang kesusahan terhadap penulis.
13. Rekan – rekan dan sahabat - sahabat angkatan 2011 yang memberikan dorongan motivasi dan bantuan kepada penulis dan membuat hati senang dengan obrolan - obrolannya.
14. Teman – teman Laboratorium Kecerdasan, Citra, dan Visi yang sudah memberikan semangat kepada penulis.
15. Teman – teman PXL yang setia memberikan motivasi untuk mengerjakan Tugas Akhir ini.
16. Pihak-pihak lain yang telah turut membantu penulis dalan menyelesaikan Tugas Akhir ini.

Penulis telah berusaha sebaik mungkin dalam menyusun tugas akhir ini, namun penulis mohon maaf apabila terdapat kekurangan, kesalahan maupun kelalaian yang telah penulis lakukan. Kritik dan saran yang membangun dapat disampaikan sebagai bahan perbaikan selanjutnya.

Surabaya, Januari 2016

LEMBAR PENGESAHAN

Implementasi Algoritma Penentuan Parameter Densitas pada Metode DBSCAN untuk Pengelompokan Data

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Cerdas Visi
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh
AKHMAD BAKHRUL ILMI
NRP : 5111 100 087

Disetujui oleh Dosen Pembimbing Tugas Akhir:

1. Dr.Eng. Chastine Fatichah, S.Kom., M.Kom. ..... (Pembimbing 1)
   NIP: 197512201995121001

2. Diana Purwitasari, S.Kom., M.Sc ..... (Pembimbing 2)
   NIP: 197804102003122001

SURABAYA
JANUARI, 2016

IMPLEMENTASI ALGORITMA PENENTUAN PARAMETER DENSITAS PADA METODE DBSCAN UNTUK PENGELOMPOKAN DATA

Nama Mahasiswa	**: AKHMAD BAKHRUL ILMI**
NRP	**: 5111100087**
Jurusan	**: Teknik Informatika FTIF-ITS**
Dosen Pembimbing 1	**: Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.**
Dosen Pembimbing 2	**: Diana Purwitasari, S.Kom , M.Sc**

Abstrak

DBSCAN adalah salah satu metode klastering dengan konsep kerapatan data. Ketika data memiliki densitas beragam maka hasil pengklasteran DBSCAN tidak maksimal. Hal ini disebabkan nilai parameter densitas bersifat global untuk seluruh data. Implementasi tugas akhir ini menyelesaikan permasalahan tersebut menggunakan modifikasi DBSCAN sehingga nilai parameter densitas akan berbeda untuk setiap klaster. Nilai parameter densitas didapatkan dari hasil k-nearest neighbor beberapa data agar data terambil bukanlah noise atau outlier.

Uji coba dilakukan dengan membandingkan hasil metode DBSCAN dengan DBSCAN yang telah dimodifikasi. Indikator keberhasilan uji coba menggunakan uji validitas klaster Indeks Dunn. Hasil uji coba validitas indeks menunjukkan bahwa DBSCAN modifikasi memiliki hasil klaster yang kurang baik dibandingkan hasil DBSCAN dengan nilai rata-rata Indeks Dunn 0.12 dan 0.146. Uji coba juga dilakukan dengan melihat label data dari kelas yang dihasilkan dengan kelas groundtruth. Pada uji coba ini DBSCAN modifikasi dapat mengidentifikasi hasil klaster yang lebih mirip dengan data aslinya dibanding dengan hasil DBSCAN tanpa modifikasi.

Kata kunci: DBSCAN, parameter densitas, indeks dunn.

A MODIFICATION FOR DETERMINING DENSITY PARAMETER VALUES ON DBSCAN ALGORITHM

Student's Name	**: AKHMAD BAKHRUL ILMI**
Student's ID	**: 5111100087**
Department	**: Teknik Informatika FTIF-ITS**
First Advisor	**: Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.**
Second Advisor	**: Diana Purwitasari, S.Kom , M.Sc**

Abstract

DBSCAN is a clustering algorithm based on density concept. DBSCAN clustering results could not be optimal if data have a variation of densities level because density parameter values applied for the entire data clusters. Our implementation resolved the problems using a modified DBSCAN so that the density parameter values will be different for each cluster. Density parameter values are obtained from the k-nearest neighbor implementation in some data to recognize data outliers.

Our experiments were comparing clustering results of DBSCAN and modified DBSCAN algorithms. We used Dunn Index as cluster validity measures. The results showed that Dunn Index values of modified DBSCAN were not better compared to the results of standard DBSCAN with Dunn Index of 0.12 and 0.146 respectively. However our experiments also compared data label of clustering results with label in ground-truth dataset. Labelling experiments showed that clustering results of modified DBSCAN algorithms had more similar label with ground-truth dataset.

Keywords: DBSCAN, density parameter, Dunn Index.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE SUMBER

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pengenalan pola adalah sebuah proses untuk mengelompokkan data–data dalam sejumlah kategori atau kelompok berdasarkan parameter yang ditentukan. Ada 2 macam pengenalan pola berdasarkan pembelajarannya, yaitu *supervised learning* dan *unsupervised learning*. *Supervised learning* adalah sebuah pembelajaran dimana pembelajarannya mengacu pada sebuah data *training* yang terdapat kategori dalam data tersebut. Sedangkan *unsupervised learning* adalah sebuah pembelajaran dimana pembelajarannya tidak menggunakan data *training,* sehingga untuk mengetahui pola-nya menggunakan prosedur–prosedur. Salah satu implementasi untuk *unsupervised learning* adalah klastering [1].

Klastering merupakan sebuah prosedur yang digunakan mengelompokkan beberapa objek yang memiliki kemiripan satu sama lain. Kelompok yang dihasilkan dalam klastering dinamakan klaster. Klastering biasanya digunakan dalam beberapa bidang contohnya pemrosesan gambar, *bioinformatics,* dan lain-lain [2].

Salah satu metode klastering adalah *density-based spatial clustering of applications with noise* (DBSCAN). Metode ini merupakan metode yang cukup bagus untuk digunakan pada klastering karena dapat digunakan untuk menemukan sebuah kelompok berdasarkan seberapa rapat data – data yang ada dan dapat mengetahui *noise* atau *outlier* [3].

DBSCAN bisa digunakan sebagai segmentasi berdasarkan region warna, misalnya pada gambar – gambar tumor. Segmentasi pada tumor ini dilakukan pada indentifikasi region warna yang homogen agar dapat diketaui struktur tumor pada suatu gambar. Segmentasi tersebut sangat penting untuk ekstraksi fitur dan proses setelahnya [4].

Penggunaan lain pada DBSCAN adalah untuk deteksi kelainan pada suatu data. Identifikasi pola kelainan pada suatu dataset sangatlah penting untuk beberapa domain seperti seperti deteksi penipuan kartu kredit, detekesi gangguan pada sistem komunikasi, dan deteksi penyakit menular pada data kesehatan. [5]

DBSCAN juga dapat digunakan untuk mengetahui komunitas pada jaringan sosial. Jaringan sosial ini direpresentasikan dengan graf. DBSCAN ini termasuk salah satu algoritma klaster yang efektif untuk mengetahui ini komunitas pada jaringan sosial. [6]

Penggunaan lain dari DBSCAN ialah pengolahan data penggunaan situs. Pengolahan tersebut mencari pola penggunaan pada sebuah situs. DBSCAN dapat mengetahui tindakan – tindakan yang biasanya dilakukan oleh pengguna pada saat menggunakan situs. [7]

Identifikasi *spam* pada email juga dapat menggunakan DBSCAN. Dalam penggunaannya DBSCAN dikombinasikan dengan algoritma nilsimsa mengidentifikasi seluruh email yang memiliki kesamaan dalam beberapa klaster. Dengan beberapa simulasi bentuk dari *spam* dapat diketahui bahwa bentuknya tidak reguler. [8]

DBSCAN memiliki beberapa kelamahan dalam penggunaannya. Salah satu kelemahannya adalah ketika sebuah data memiliki kerapatan yang berbeda untuk tiap kelompoknya. Kelemahan tersebut terjadi dikarenakan oleh parameter densitas yang ada pada DBSCAN. Parameter tersebut digunakan untuk seluruh kelompok kerapatan akan tetapi nilai parameter tersebut tidak selalu tepat untuk setiap kelompok [2].

Untuk mengatasi masalah tersebut, dibutuhkan algoritma untuk pencarian parameter densitas untuk setiap kelompoknya. Algoritma ini dilakukan dengan cara mengetahui kerapatan untuk setiap datanya menggunakan *nearest neighbor* yang menjadi acuan dalam algoritma ini [2].

1.2 Rumusan Masalah

Adapun permasalahan dan pertanyaan yang akan diselesaikan dalam pengerjaan Tugas Akhir ini adalah sebagai berikut:

1. Bagaimana menginplementasikan algoritma penentuan parameter densitas pada metode DBSCAN?
2. Bagaimana mengelompokkan data menggunakan metode DBSCAN?
3. Bagaimana mengevaluasi kinerja metode yang digunakan?

1.3 Batasan Masalah

Permasalahan yang dibahas dalam Tugas Akhir ini memiliki beberapa batasan, yaitu sebagai berikut:

1. Aplikasi ini merupakakan aplikasi desktop.
2. Dataset yang digunakan adalah beberapa dataset sintesis.

1.4 Tujuan

Tujuan dari pembuatan tugas akhir ini adalah mengimplementasikan algoritma penentuan parameter densitas pada metode DBSCAN untuk pengelompokan data.

1.5 Manfaat

Manfaat dari hasil pembuatan tugas akhir ini adalah dapat digunakan untuk pengelompokan data yang digunakan pada data medical, astronomy, ekonomi dan lainnya yang memiliki kerapatan data yang besar karena metode clustering ini dengan parameter yang didapatkan untuk seluruh densitas, dapat mengatasi masalah–masalah tersebut.

1.6 Metodologi

Tahapan-tahapan yang dilakukan dalam pengerjaan Tugas Akhir ini adalah sebagai berikut:

1. Penyusunan proposal Tugas Akhir.

   Tahap awal untuk memulai pengerjaan Tugas Akhir adalah penyusunan proposal Tugas Akhir. Proposal Tugas Akhir yang diajukan memiliki gagasan yang sama dengan Tugas Akhir ini.

2. Studi literatur

   Pada tahapan ini dilakukan proses pencarian, pengumpulan, pembelajaran dan pemahaman informasi dan literature yang diperlukan untuk pembuatan implementasi algoritma penentuan parameter densitas pada DBSCAN. Yang dipelajari untuk algoritma tersebut adalah K-Nearest Neighbor yang digunakan untuk mengetahui parameter densitas. Kemudian algoritma DBSCAN yang digunakan untuk melakukan klastering. Dan Euclidean distance sebagai penentuan jarak antar tiap data. Referensi utama pada saat pembelajaran tentang algoritma penentuan parameter densitas pada DBSCAN adalah pada "*A Clustering Algorithm for Discovering Varied Density Clusters*" oleh Ahmed M. Fahim.

3. Analisis dan desain perangkat lunak

   Tahap ini meliputi perancangan sistem berdasarkan studi literatur dan pembelajaran konsep teknologi dari perangkat lunak yang ada. Tahap ini mendefinisikan alur dari implementasi. Langkah-langkah yang dikerjakan juga didefinisikan pada tahap ini. Pada tahapan ini dibuat prototype sistem, yang merupakan rancangan dasar dari sistem yang akan dibuat. Serta dilakukan desain suatu sistem dan desain proses-proses yang ada. Modul–modul yang digunakan pada tahapan ini antara lain adalah modul untuk mencari nilai *k-nearest neighbor* yang nantinya digunakan sebagai parameter densitas pada DBSCAN, modul untuk mencari kerapatan pada setiap data yang nantinya digunakan sebagai acuan dalam pengambilan nilai parameter yang berdasarkan nilai *k-nearest neighbor* sebelumnya, kemudian modul untuk

mengelompokkan data berdasarkan nilai parameter densitas yang sudah ditentukan dengan cara mengambil data yang paling rapat kemudian memakai jarak *k-nearest neighbor* dari data yang paling rapat.

4. Implementasi perangkat lunak

   Implementasi merupakan tahap membangun rancangan program yang telah dibuat. Pada tahapan ini direalisasikan apa yang terdapat tahapan sebelumnya, sehingga menjadi sebuah program yang sesuai dengan apa yang telah direncanakan. Implementasi perangkat lunak ini akan menggunakan bahasa pemrograman C#. Bahasa pemrograman ini dikembangkan oleh *Microsoft*. IDE yang digunakan pada implementasi ini menggunakan *Microsoft Visual Studio* 2013

5. Pengujian dan evaluasi

   Pada tahapan ini dilakukan uji coba terhadap purwarupa sistem informasi yang sudah dibuat pada tahap sebelumnya. Pengujian dan evaluasi dilakukan dengan menggunakan beberapa dataset. Skenario ujicoba yang digunakan pada tahanpan pengujian dan evaluasi ini adalah pengujian uji hasil klastering berdasarkan nilai Indeks Dunn.Tahapan ini dimaksudkan untuk mengevaluasi kesesuaian algoritma dan program serta mencari masalah yang mungkin timbul dan mengadakan perbaikan jika terdapat kesalahan.

6. Penyusunan buku Tugas Akhir.

   Pada tahapan ini disusun buku yang memuat dokumentasi mengenai pembuatan serta hasil dari implementasi perangkat lunak yang telah dibuat.

1.7 Sistematika Penulisan Laporan Tugas Akhir

Secara garis besar, buku Tugas Akhir terdiri atas beberapa bagian yang disusun secara sistematis seperti berikut ini:

Bab I Pendahuluan

Bab yang berisi mengenai latar belakang, tujuan, dan manfaat dari pembuatan Tugas Akhir. Selain itu permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penulisan juga merupakan bagian dari bab ini.

Pada bab ini dijelaskan latar belakang mulai dari pengenalan pola, klastering dan DBSCAN. DBSCAN merupakan salah satu metode klastering yang dapat mengetahui adanya *noise / outlier*. Metode ini menggunakan kerapatan suatu data dalam mendapatkan klasternya. DBSCAN dapat dimanfaatkan untuk segmentasi region warna pada suatu gambar, deteksi kelainan atau pengenalan pola kelainan pada suatu data, mengetahui komunitas pada jaringan sosial, pengolahan data pada sebuah situs, indentifikasi spam pada email dan lain-lain.

Bab II Tinjauan Pustaka

Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan Tugas Akhir ini diantaranya penjelasan tentang *clustering,* algoritma DBSCAN, algoritma penentuan parameter densitas dan juga pengujian dengan Indeks Dunn

Bab III Perancangan Perangkat Lunak

Bab ini berisi tentang desain sistem yang disajikan dalam bentuk pseudocode.

Bab IV Implementasi

Bab ini membahas implementasi dari desain yang telah dibuat pada bab sebelumnya. Penjelasan berupa code yang digunakan untuk proses implementasi.

Bab V Uji Coba Dan Evaluasi

Bab ini menjelaskan kemampuan perangkat lunak dengan melakukan pengujian kebenaran dan pengujian kinerja dari sistem yang telah dibuat.

Bab VI Kesimpulan Dan Saran

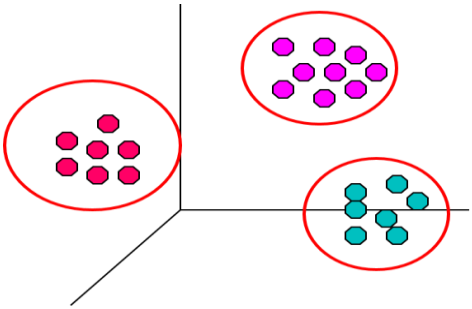
Bab ini merupakan bab terakhir yang menyampaikan kesimpulan dari analisa hasil uji coba yang dilakukan dan saran untuk pengembangan perangkat lunak ke depannya.

BAB II
TINJAUAN PUSTAKA

Bab ini berisi pembahasan dari istilah-istilah khusus dan dasar-dasar teori yang digunakan dalam Tugas Akhir. Istilah dan dasar teori yang dibahas juga mencakup teknologi-teknologi yang digunakan pada Tugas Akhir.

2.1 Klastering

Klastering adalah suatu proses pengelompokan objek ke dalam kelas dengan cara membagi dataset yang ada menjadi beberapa subset yang berbeda, yang tiap objek dalam satu klasternya mirip satu sama lain, sedangkan objek yang berada dalam klaster yang berbeda atau tidak mirip. Klastering juga disebut sebagai segmentasi data pada beberapa aplikasinya karena klastering memartisi dataset yang besar menjadi sebuah grup menurut kemiripan tiap datanya. Contoh hasil klastering terdapat pada Gambar 2.1.



Gambar 2.1 Hasil Klastering

Klastering bermanfaat untuk menganalisis data – data ekspresi gen. Misalkan pada data ekspresi gen kanker, klastering digunakan untuk mengenali ekspresi gen kanker dan

membedakannya dari gen – gen yang sehat. Hasil klastering tersebut digunakan untuk mendiagnosis kanker dan perawatan pada penyakit kanker tersebut. [9]

Penggunaan lain dari klastering adalah pada mengidentifikasi sebuah pola pada suatu data spasial. Hal ini dilakukan dengan memahami struktur spasial pada data tersebut dengan menggunakan suatu metode klastering. [10]

Klastering juga dimanfaatkan sebagai pengelompokan hasil pencarian dokumen pada sebuah pencarian situs. Pengelompokan ini digunakan untuk menaikkan cakupan dari dokumen yang ditampilkan kepada pengguna untuk dilakukan pengulasan tanpa mengurangi waktu untuk mengulas dokumen tersebut. [11]

Klastering juga dapat digunakan sebagai pengelompokan data berupa gambar. Dengan melakukan klastering dapat memberikan penganalisaan yang mudah dan validasi yang baik. Pengelompokan pada data gambar dengan cara menentukan kemirikan pada suatu gambar. Misalkan foto objek yang sama akan tetapi dari sudut yang berbeda. [12]

Klastering dibagi menjadi 2 tipe yaitu *partitional clustering*, dan *hierarchical clustering*. *Partitional Clustering* adalah Klastering yang membagi data menjadi beberapa subset atau klaster. *Hierarchical Clustering* adalah Klastering yang berasal dari 2 subset digabung menjadi 1 klaster baru yang terorganisir sebagai pohon hirarki [13].

2.2 *Density-Based Spatial Clustering of Applications with Noise* (DBSCAN)

Density-Based Spatial Clustering of Applications with Noise atau yang biasa disebut dengan DBSCAN adalah sebuah algoritma pengelompokan data berbasis densitas yang didesain untuk menemukan noise pada sebuah data spasial. Pada algoritma ini dibutuhkan 2 parameter untuk melakukannya yaitu *Epsilon* dan *MinPts*. *Epsilon* adalah jarak yang ditentukan dari suatu data untuk menghitung banyaknya neighbor pada data tersebut. *MinPts* adalah

banyaknya minimal neighbor yang dibutuhkan untuk membuat sebuah region yang rapat pada suatu data. [3]

Pertama, perlu diketahui bahwa pada setiap data memiliki kedekatan pada data yang lain. Untuk setiap data akan memiliki suatu klaster ketika data tersebut memiliki sedikitnya *MinPts* data pada radius *Epsilon*. Akan tetapi syarat tersebut tidaklah cukup dikarenakan ada 2 macam data yang berada pada klaster yaitu *core point* dan *border point.* [3]

$$N_{Eps}(p) = \{q \in DB | dist(p,q) < Epsilon\} \quad \text{.....(2.1)}$$

p = sebuah titik / data ke-1

q = sebuah titik / data ke-2

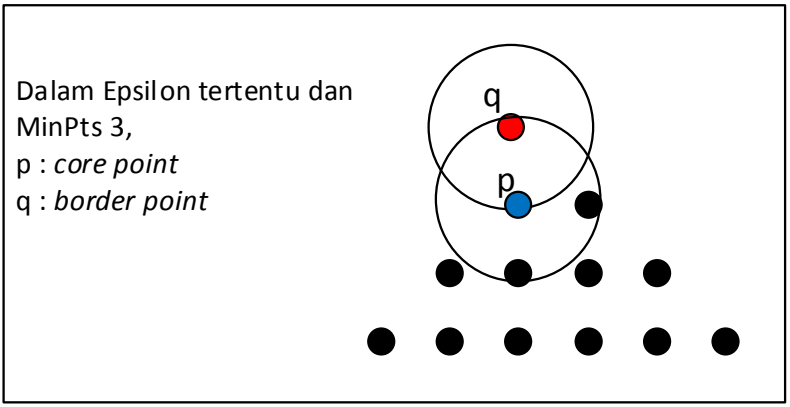
$dist\ (p,q)$ = Jarak dari titik p ke titik q dengan menggunakan *Euclidean Distance*

$N_{Eps}(p)$ = Jumlah Tetangga dari titik p

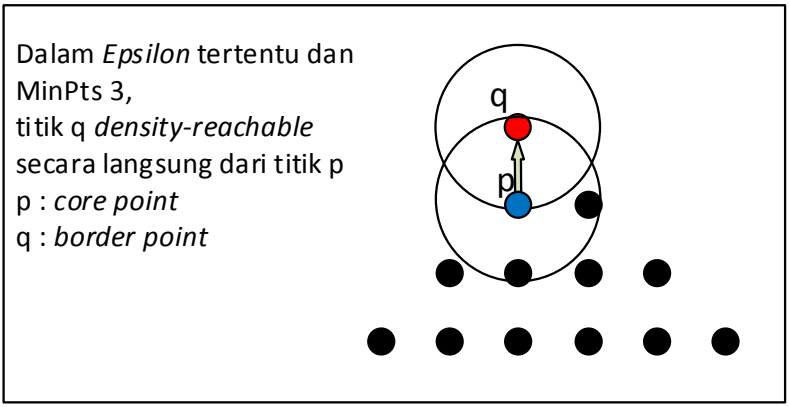
Sebuah data dikatakan sebagai *core point* ketika jumlah tetangga pada radius *epsilon* sama atau lebih dari *MinPts*. Sebuah *core point* akan menjadi sebuah klaster. Berbeda dengan *core point, border point* berada pada ujung klaster dan tidak memilki jumlah tetangga yang sama atau lebih dari *MinPts* akan tetapi bisa masuk ke dalam klaster dikarenakan bertetangga dengan sebuah *core point* [3]. Contoh *core point* dan *border point* ada pada Gambar 2.2.

Sebuah klaster diawali dari sebuah data *core point* dimana data tersebut harus *directly density-reachable* atau *density-reachable* pada data yang lain. Yang dimaksud dengan *directly density-reachable* adalah ketika sebuah data *core point* dapat mencapai data lain yang termasuk tetangganya, dan yang dimaksud dengan *density-reachable* adalah ketika sebuah data *core point* dapat mencapai data lain atau jika suatu data dapat dicapai oleh data *core point*. Misal, jika ada deret data $P_i \ldots, P_n, P_1$=data lain

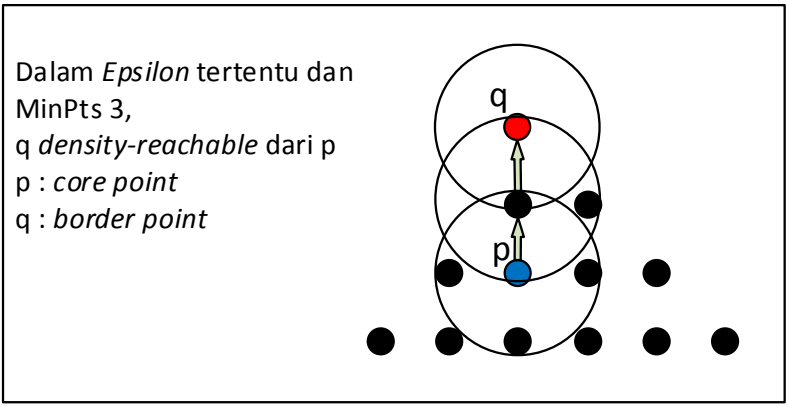
dan P_n=data *core point*, yang data P_{i+1}-nya *directly density-reachable* dengan data p_i [3]. Contoh *directly density-reachable* terdapat pada Gambar 2.3 dan Contoh untuk *density-reachable* terdapat pada Gambar 2.4.



Gambar 2.2 Contoh *Core Point* dan *Border Point*



Gambar 2.3 Contoh *Directly Density-Reachable*



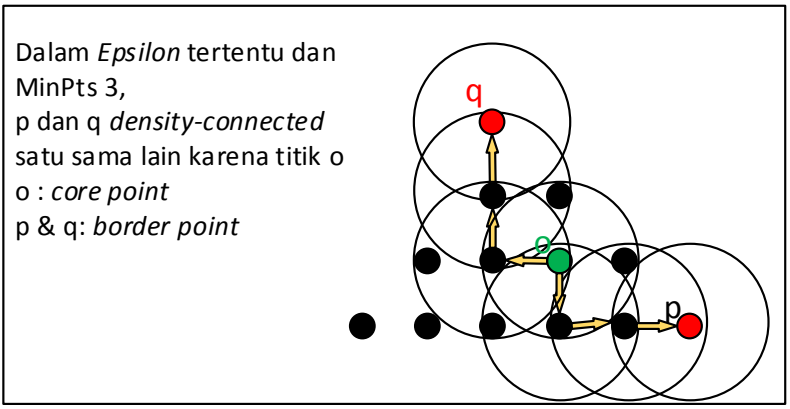
Gambar 2.4 Contoh *Density-Reachable*

Untuk 2 *border point* pada klaster yang sama, kedua data tersebut tidak *density-reachable* satu antara yang lain, melainkan *density-connected*. *Density-connected* terjadi ketika kedua *border point* density-reachable dari sebuah data yang berupa *core point*. Misalkan *border point* p *density-reachable* dengan *core point o*, begitu pula *border point q*. Maka kedua *border point p* dan *q* masing – masing *density-connected* satu sama lain oleh *core point o* [3]. *Density-connected* juga terjadi pada seluruh data pada satu klaster yang sama tidak hanya pada *border point* saja. Ilustrasi *density-connected* terdapat pada Gambar 2.5

Pada saat mulai pembentukan klaster, terdapat beberapa syarat. Syarat - syaratnya diantaranya adalah

1. Jika data *p* masuk ke dalam sebuah klaster dan data *q* *density-reachable* dari data *p* maka *q* juga masuk ke dalam klaster yang sama dengan data *p*. Hal ini bertujuan untuk memaksimalkan hasil klaster
2. Jika data *p* dan *q* terdapat pada suatu klaster yang sama maka kedua data tersebut *density-connected* satu sama lain. Hal ini bertujuan untuk mendapatkan hubungan antar suatu data.

3. *Noise / outlier* didapatkan dari seluruh data yang tidak memiliki klaster pada saat pembetukannya



Gambar 2.5 Contoh *Density-Connected*

Secara garis besar dari yang dijelaskan sebelumnya dapat disimpulkan dalam sebuah algoritma DBSCAN dengan beberapa langkah – langkah sebagai berikut [3].

1. Untuk setiap data, dicari data tetangga pada suatu data dalam radius *epsilon* dengan menggunakan pencarian jarak.
2. Jika banyaknya tetangga pada data *P* melebihi *minPts* maka klaster baru akan dibuat dengan data *P* sebagai *core point*.
3. Kemudian dilakukan pengkoleksian data yang bersifat *directly density-reachable* dari data *P* untuk bergabung menjadi klaster yang sama dengan data *P*.
4. Selanjutnya mengoleksi data yang bersifat *density-reachable* dari data *P* untuk bergabung pada klaster yang sama dengan data *P*. Hal ini dapat dilakukan dengan cara mengambil data yang bersifat *directly density-reachable* dari data tetangga pada radius *epsilon* pada data *P*.
5. Kemudian hal tersebut dilakukan hingga tidak ada data yang *density-reachable* dari p atau tidak ada data lain yang akan dimasukkan ke dalam klaster yang sama dengan data *P*.

6. Setelah tidak ada data lain yang dimasukkan pada klaster yang sama dengan data *P*, kembali melakukan langkah 2 dengan mengganti data *P* dengan data *core point* lainnya yang tidak memiliki klaster.
7. Hal ini dilakukan hingga tidak ada lagi data *core point* pada dataset.

Contoh hasil pembentukan klaster dengan DBSCAN terdapat pada Gambar 2.6.

Gambar 2.6 Contoh Hasil Klastering DBSCAN [13]

2.3 *K-Nearest Neighbor Distance*

K-Nearest Neighbor atau biasa disebut k-NN adalah sebuah metode yang umumnya dalam klasifikasi dan regresi. Metode ini termasuk *lazy learning*. Metode ini juga metode yang paling simpel diantara semua metode yang ada pada *machine learning*. Akan tetapi, yang akan dipakai disini hanya hasil jarak k-NN dan tetangga sebuah data.

K-Nearest Neighbor sebuah data mempunyai tetangga sejumlah *k*, yang dapat mencari *k* data terdekat untuk tiap data. Untuk mengetahui jarak antar tiap data, dapat menggunakan

Euclidean Distance seperti pada formula 2.2. Untuk setiap data akan dilakukan sorting berdasarkan data yang paling dekat sampai yang paling jauh. Kemudian akan diambil sebanyak *k-neighbor* yang terdekat atau mempunyai jarak terkecil dari data tersebut.

$$dist(p,q) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \quad \text{.....(2.2)}$$

p_i = Fitur ke - i pada titik p.

q_i = Fitur ke - i pada titik q.

n = Banyaknya fitur.

dist (p,q) = Jarak antara titik p dan titik q.

Untuk mengetahui jarak pada *k-nearest neighbor* suatu data, dapat dilakukan dengan mencari jarak terjauh dari pada tetangga pada suatu data. Dengan mencari jarak terjauh ini, didapatkan jarak *k-nearest neighbor* yang nantinya akan digunakan sebagai acuan untuk mencari nilai parameter densitas. Formula pencarian jarak *k-nearest neighbor* dijelaskan pada formula 2.3

$$Knn(p) = \max\{ dist(p,q) \mid q \in kNeighbor(p)\} \quad \text{..... (2.3)}$$

dist (p,q) = Jarak antara titik p dan titik q.

kNeighbor (p) = Tetangga dari titik p.

Knn (p) = Jarak tetangga terjauh dari titik p.

Hasil dari *k-nearest neighbor* tidak hanya jarak, tetapi juga tetangga dari sebuah data juga. Setiap data dicari tetangganya kemudian diurutkan berdasarkan jarak. Hal ini dilakukan supaya pada saat melakukan pelebaran klaster pada DBSCAN menjadi

semakin lebih mudah. Formula untuk pengurutan tetangga pada titik p terdapat pada formula 2.4.

$$N_k(p) = \{q \in D, d(p, q_{i-1}) \leq d(p, q_i), i = 1,...,k\} \quad \text{... (2.4)}$$

$N_k(p)$ = Tetangga dari titik p dengan jumlah tetangga sebanyak k titik.

D = Dataset yang digunakan.

k = Banyaknya tetangga pada suatu titik.

i = Nilai iterasi dari k.

$d(p,q_{i-1})$ = Jarak dari titik p ke titik q_{i-1}.

$d(p,q_{i-1})$ = Jarak dari titik p ke titik q_i.

2.4 Algoritma Pencarian Nilai Parameter Densitas dan Modifikasi DBSCAN

Algoritma ini diusulkan oleh Ahmed M. Fahim. Algoritma ini diusulkan karena DBSCAN sangat sensitif terhadap *epsilon.* Algoritma ini menggunakan jarak *k-nearest neighbor* sebagai acuan nilai *epsilon,* akan tetapi hasil *epsilon* yang didapat tidak hanya satu melainkan berdasarkan bentuk kerapatan atau klaster pada suatu data [2].

Langkah – langkah pada algoritma tersebut adalah sebagai berikut:

1. Mencari jarak *k-nearest neighbor* dan *k-neighbor* dari suatu data, menggunakan formula yang ada pada poin sebelumnya.
2. Mencari nilai kerapatan suatu data
   Untuk mendapatkan nilai kerapatan data dibutuhkan sebuah fungsi *influence* antara 2 data. Fungsi tersebut memakai *Euclidean Distance* untuk merepresentasikan pengaruh antar 2

data. Formula untuk fungsi *influence* terdapat pada formula 2.5. [2]

$$INF(p,q) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \quad \text{.....(2.5)}$$

INF (p, q) = Nilai Fungsi *Influence*

n = Banyaknya fitur

Kemudian nilai kerapatan didapatkan dengan cara menjumlahkan fungsi pengaruh yang terdapat pada *k-neighbor* pada sebuah data

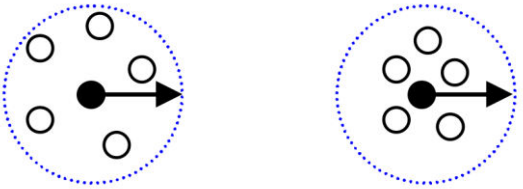
$$DEN_k(p) = \sum_{i=1}^{k} INF(p, q_i) \quad \text{.....(2.6)}$$

DEN (p) = Nilai kerapatan pada titik p.

k = Banyaknya tetangga pada titik p.

q_i = Titik tetangga ke-i dari titik p

Nilai kerapatan berdasarkan jumlah jarak pada *k-Nearest Neighbor* lebih bagus daripada menghitung data pada radius *epsilon*. Misal pada Gambar 2.7, ada 2 macam data hitam dan 5 data lainnya pada ketetanggan pada nilai *epsilon*. Pada gambar b memiliki ketetanggan yang sangat rapat dibandingkan dengan gambar a. Dari gambar tersebut dihasilkan bahwa menggunakan jumlah jarak lebih akurat dibandingkan dengan memakai ketetanggaan pada radius *epsilon*.



Gambar 2.7 Gambar Kerapatan Berdasarkan jarak k-nn

3. Mengurutkan data berdasarkan nilai kerapatan data
   Setelah didapatkan nilai kerapatan untuk keseluruhan data, dilakukan pengurutan data berdasarkan data yang memiliki kerapatan yang tinggi / atau nilai kerapatan yang kecil. Dilakukan penyusunan ini, bertujuan agar klaster yang pertama dibuat adalah klaster yang memiliki kerapatan yang besar daripada data yang tidak begitu rapat. Karena jika dimulai dari data yang kurang rapat maka ada kemungkinan data yang diambil terlebih dahulu adalah sebuah noise atau outlier.

$$SortedDen_k = \{p, q \in D, DEN_k(p) \leq DEN_k(q)\} \quad \text{.....(2.7)}$$

$DEN_k\ (p)$ = Nilai kerapatan titik p dengan banyaknya k tetangga.

$DEN_k\ (q)$ = Nilai kerapatan titik q dengan banyaknya k tetangga.

p, q = Titik yang ada pada dataset.

$SortedDen_k$= Hasil kerapatan yang sudah terurut.

4. Pengelompokan berdasarkan DBSCAN
   Algoritma pengelompokan DBSCAN ini dimodifikasi sedimikian rupa sehingga dapat digunakan agar bisa dipakai dengan banyak nilai parameter densitas. Algoritma ini juga dapat merubah nilai parameter yang tepat untuk suatu klaster.

Algoritma ini juga dapat menemukan klaster yang memiliki perbedaan densitas atau kerapatan tiap klasternya. Dalam pengelompokannya, pengguna harus menginputkan 2 nilai parameter yaitu nilai *k* sebagai penentu nilai parameter densitas dan *minPts* sebagai jumlah minimum tetangga sebagai syarat untuk menjadi *core point* dalam radius *epsilon.*

Langkah – langkah dalam pengelompokan DBSCAN yang sudah dimodifikasi adalah sebagai berikut

- Menginisialisasi id klaster seluruh data menjadi tidak mempunyai klaster
- *clusterID* = 1 pada saat inisialisasi pertama untuk klaster 1
- Dari data yang sudah diurutkan berdasarkan nilai kerapatan, diambil sebuah data *p* yang memiliki nilai kerapatan terkecil.
- Ganti nilai *epsilon* dengan nilai jarak *k-nearest neighbor* dari data *p*.
- Ganti id klaster pada data *p* menjadi clusterID
- Seluruh tetangga pada data *p* dimasukkan ke dalam sebuah *seedlist* yang nantinya digunakan mengoleksi seluruh titik yang *directly density-reachable* dari data *p*. *Seedlist* berisi sebuah list data yang akan dilakukan perluasan klaster
- Dimulai dari *seedlist* yang paling dekat dari data *p*, dilakukan pengecekan. Misal data *q* adalah data ada pada *seedlist* dan paling dekat dari data *p*. Jika data *q* teridentifikasi sebagai *core point*, maka setiap tetangga dari data *q* yang berjarak kurang dari sama dengan nilai *epsilon* yang berupa nilai *k-nerarest neighbor* dari data p dan data tersebut tidak memiliki id klaster, akan dimasukkan ke dalam *seedlist*. Hal ini dilakukan untuk mengambil data – data yang *density-reachable* dari data *p* atau *directly density-reachable* dari data *q*. Akan tetapi jika data *q* tersebut adalah *border point* maka tidak ada data yang dimasukkan ke dalam *seedlist* karena tidak ada data yang *density-reachable* dari data *q*.
- Ganti id klaster data *q* menjadi nilai clusterID

- Perluasan klaster dilakukan hingga seluruh *seedlist* sudah dilewati seluruhnya atau hingga *seedlist* kosong.
- Mulai klaster baru dengan mengambil data lainnya yang memiliki kerapatan terkecil dan tidak memiliki klaster dan lakukan poin ke 3.
- *clusterID* = *clusterID*+1.

Dari pengelompokan tersebut didapat banyak klaster yang terbentuk akan tetapi klaster yang memiliki jumlah data per klaster sedikit akan dibuang. Untuk *threshold* penentuan jumlah data yang memenuhi klaster adalah sebuah masukan yang dimasukkan oleh pengguna.

2.5 Metode Evaluasi: Indeks Dunn

Indeks Dunn adalah indeks yang digunakan untuk menguji validitas klaster. Indeks ini adalah perbangingan antara jarak terkecil antar dua data pada klaster yang berbeda atau bisa disebut dengan jarak inter-klaster dan jarak terbesar antara dua data pada suatu klaster atau bisa disebut dengan jarak intra-klaster [14]. Hasil dari indeks dunn ini bernilai antara 0 sampai ∞. Semakin besar nilai indeks dunn semakin bagus hasil klastering. Rumus perhitungan indeks dunn ditunjukkan pada formula 2.8.

$$DI(m) = \frac{\min\limits_{1 \le i,j \le 6, i \ne j} d(C_i, C_j)}{\max\limits_{1 \le k \le m} d'(C_k)} \quad \text{..... (2.8)}$$

DI (m) = Nilai Indeks Dunn pada jumlah klaster m.

m = Banyaknya klaster yang digunakan.

$d(C_i, C_j)$ = Jarak inter-klaster antara klaster I dan klaster j.

$d'(C_k)$ = Jarak intra-klaster dari klaster k.

i,j,k = indeks dari tiap klaster.

Dari formula Indeks Dunn diatas, akan dijabar dalan sebuah contoh penghitungan indeks dunn. Tabel 2.1 berisi dataset sintesis dan id klaster tiap data.

Tabel 2.1 Contoh *Dataset* Sintesis

X	Y	Klaster
7.75	5.25	1
7.75	6.25	1
8.75	5.25	1
8.75	6.25	1
8.75	7.25	1
9.75	6.25	1
9.75	7.25	1
7.75	7.25	1
9.75	5.25	1
7.75	8.25	1
8.75	8.25	1
9.75	8.25	1
10.75	8.25	1
10.75	7.25	1
10.75	6.25	1
10.75	5.25	1
10	10.25	2
9.5	10.4	2
9.8	10.2	2
9.7	9.4	2
9.9	9.8	2
9.2	9.7	2
9.5	9.8	2

10.5	10.2	2
10.25	10	2
10.7	10.5	2

Dari data tersebut dicari jarak inter-klaster yang merupakan jarak terdekat antara 2 klaster yaitu dari klaster 1 dan klaster 2. Kemudian dicari jarak intra-klaster yang merupakan jaran terjauh dari sebuah klaster, untuk klaster 1 dan klaster 2. Pada kasus tabel 2.1 sudah didapat data - data yang memiliki jarak terjauh untuk klaster 1 dan klaster 2 dan sudah didapat data yang memiliki jarak terdekat pada 2 klaster tersebut. Untuk penghitungan jarak antara data, tetap menggunakan *euclidean distance* dikarenakan data – datanya menggunakan data yang numerik. Untuk tabel penghitungannya ada pada Tabel 2.2.

Tabel 2.2 Hasil Penghitungan Nilai Indeks Dunn

Data Terjauh pada Cluster 1	7.75	5.25	
	10.75	8.25	
	Intracluster C1		4.242641
Data Terjauh pada Cluster 2	9.2	9.7	
	10.7	10.5	
	Intracluster C2		1.7
Data terdekat antara Cluster 1 dan Cluster 2	9.75	8.25	
	9.7	9.4	
	Intercluster C1,C2		1.151086
Hasil Indeks Dunn	0.271314		

BAB III
DESAIN PERANGKAT LUNAK

Pada bab ini akan dijelaskan proses perancangan program yang dibuat. Perancangan ini dibagi menjadi tiga proses utama, yaitu:

1. Desain pencarian nilai parameter untuk tiap densitas
2. Desain klastering menggunakan modifikasi DBSCAN

Pada bab ini akan dijelaskan gambaran umum setiap program utama dalam *pseudocode*

3.1 Desain Metode Secara Umum

Pada tugas akhir ini akan dibangun suatu sistem yang dapat mengimplementasikan algoritma pencarian nilai parameter densitas dan modifikasi DBSCAN. Sistem akan menerima masukan *dataset* yang mengandung data bersifat numerik, nilai k berupa angka, dan nilai *minPts* yang berupa angka.

Dalam pengerjaan tugas akhir ini dibagi menjadi beberapa proses antara lain :

- P1 : Mencari jarak antar data
  Jarak antar data pada proses ini dilakukan dengan menggunakan *euclidean distance*. Hal ini dilakukan bersamaan pada saat pencarian jarak *k-nearest neighbor*
- P2 : Mencari jarak k-nearest neighbor untuk setiap data
  Jarak *k-nearest neighbor* dicari dengan nilai masukkan *k* sebagai penentu banyaknya tetangga untuk setiap data. Hasil dari *k-nearest neighbor* akan dipakai untuk pemakaian parameter densitas pada saat pengelompokan data.
- P3 : Mencari nilai kerapatan pada setiap data
  Nilai kerapatan didapatkan berdasarkan jumlah jarak (menggunakan *Euclidean Distance*) pada suatu data dan tetangganya. Data yang memiliki nilai kerapatan yang kecil maka data tersebut makin rapat.
- P4 : Mengurutkan data berdasarkan nilai kerapatannya

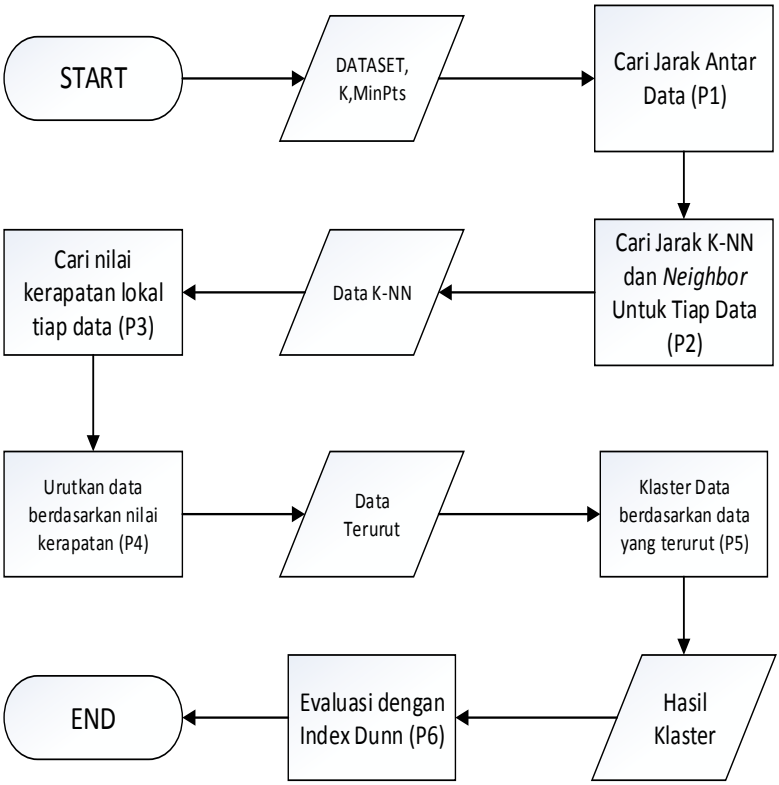
Urutan data berdasarkan nilai kerapatan ini akan digunakan untuk mengambil data pertama yang digunakan sebagai nilai parameter densitas pada pengelompokan data. Pengurutan data menggunakan *library* dari C# menggunakan *arraylist*

- P5 : Mengklaster data berdasasrkan urutan data pada P4 dengan memakai nilai parameter densitas berdasarkan urutan data tersebut.
  Pengelompokan data menggunakan konsep DBSCAN yang sudah dimodifikasi sedemikian rupa sehingga dapat digunakan untuk banyak parameter sesuai yang ditentukan untuk suatu dataset. Hasil dari klaster yang dibentuk akan dilakukan pembersihan klaster yang memiliki jumlah data kurang dari sebuah *threshold* yang merupakan masukan.
- P6 : Mengevaluasi hasil klaster pada P5 dengan memakai Indeks Dunn.
  Hasil klaster yang dihasilkan pada P5 akan dilakukan evaluasi menggunakan Indeks Dunn. Indeks Dunn dapat mengetahui validitas klaster. Semakin tinggi nilai Indeks Dunn semakin baik.

Untuk lebih jelasnya digambar untuk alur program utama pada Gambar 3.1

3.2 Desain Algoritma *K-Nearest Neighbor*

Algoritma ini digunakan untuk mencari jarak *k-nearest neighbor* dan tetangga sejumlah k untuk tiap data. Jarak tersebut akan digunakan nantinya acuan untuk pencarian nilai parameter densitas. Desain algoritma *k-nearest neighbor* dapat dilihat pada Gambar 3.2.



Gambar 3.1 Alur Program Utama

Masukan	*Dataset(A x B), Integer K, Objek obj*
Keluaran	K-Neighbor(A x K) *dan Jarak* K-Nearest Neighbor (A x 1)
1. Menghitung jarak menggunakan *euclidean distance untuk setiap data pada dataset* 2. *Masukkan data jarak kedalam sebuah array / list* 3. Dilakukan pengurutan jarak untuk data obj secara *ascending* 4. Dilakukan pengulangan dari 1 ≤ k untuk menyimpan tetangga data obj dari jarak yang sudah terurut	

```
5. Tetapkan nilai jarak k-nearest neighbor untuk
   data obj dengan mengambil jarak dari data obj ke
   neighbor ke-k
6. Pencarian ini dilakukan untuk semua data pada
   dataset
7. Selesai
```

Gambar 3.2 ***Pseudocode K-Nearest Neighbor***

3.3 Desain Algoritma Pencarian Nilai Parameter Densitas

Algoritma ini digunakan untuk mencari nilai parameter densitas menggunakan hasil dari *k-nearest neighbor* suatu data. Untuk data atau titik mana yang akan digunakan, didapat dari algoritma ini. Algoritma ini menentukan titik mana yang akan dijadikan acuan untuk dipakai dalam melakukan klastering

Sesuai dengan yang disebutkan pada poin sebelumnya ada beberapa langkah pada algoritma pencarian nilai parameter densitas ini, antara lain

1. Pencarian nilai kerapatan untuk setiap data
   Nilai kerapatan suatu data didapatkan dari total jarak antara suatu data dan tetangganya. Nilai kerapatan tersebut akan digunakan sebagai acuan untuk pengambilan data dan jarak *k-nearest neighbor*-nya sebagai nilai parameter densitas. Desain algoritma dijelaskan dalam bentuk *pseudocode* yang terdapat pada Gambar 3.3.

Masukan	*Dataset(A x B), Integer K, Data obj*
Keluaran	Nilai Kerapatan(A x 1)

```
1. Untuk setiap k-neighbor pada data obj dilakukan
   fungsi pengaruh yang direpresentasikan dengan
   Euclidean Distance.
2. Jumlahkan seluruh jarak dari data obj ke semua
   k-neighbor.
3. Kembalikan nilai jumlahan terebut menjadi nilai
   kerapatan data obj.
```

4. Pencarian ini dilakukan untuk seluruh data pada dataset.
5. Selesai.

Gambar 3.3 *Pseudocode* Pencarian Nilai Kerapatan

2. Pengurutan nilai kerapatan data
   Nilai kerapatan data diurutkan dari data yang memiliki data yang rapat hingga data yang renggang. Data yang rapat memiliki nilai kerapatan yang kecil sedangkan data yang renggang memiliki nilai kerapatan yang cukup besar. Desain algoritma untuk pengurutan nilai kerapatan dijelas pada *pseudocode* pada Gambar 3.4.

Masukan	*Dataset(A x B), Hasil Nilai Kerapatan (A x 1)*
Keluaran	Array Hasil Pengurutan Nilai Kerapatan (A x 1)

1. Untuk seluruh nilai kerapatan pada data dilakukan pengurutan dengan $DEN_i < DEN_{i+1}$.
2. Catat indeks data untuk setiap nilai kerapatan yang sudah terurut
3. Kembalikan hasil array pengurutan nilai kerapatan.
4. Selesai.

Gambar 3.4 *Pseudocode* Pengurutan Nilai Kerapatan

3. Pengelompokan menggunakan modifikasi DBSCAN
   Pengelompokan dilakukan mulai dari satu data p, kemudian dilakukan perluasan dengan parameter densitas sebesar jarak *k-nearesst neighbor* dari data *p*. Ketika tidak bisa dilakukan perluasan lagi, maka dilakukan kembali dengan menggunakan data yang baru berdasarkan urutan nilai kerapatan. Desain algoritma untuk pengelompokan data dengan menggunakan modifikasi DBSCAN terdapat pada Gambar 3. 5.

Masukan	*Dataset(A x B), Array hasil pengurutan nilai kerapatan(A x 1), MinPts (1)*
Keluaran	Hasil klaster (A x 1)

1. Mulai dari data yang memiliki nilai kerapatan terkecil. Misal data p
2. ClusterID <= 1
3. Tetapkan Hasil Cluster p <= ClusterID
4. Tetapkan nilai *Epsilon* menjadi jarak *k-nearest neighbor* dari data p
5. Masukkan semua tetangga pada *k-neighbor* pada *seedlist* berdasarkan jarak yang paling dekat dengan data p, yang nantinya akan diperluas.
6. Dimulai dari seedlist q dilakukan pengecekan apakah data q termasuk *core point* pada radius *epsilon*.
7. Jika data q adalah *core point*, maka dilakukan perluasan pada data q dengan memasukkan seluruh data *k-neighbor* yang jarak dari data q kurang dari sama dengan *epsilon* dan tidak memiliki klaster.
8. Jika data q adalah border point, maka tidak ada perluasan pada data q.
9. Tetapkan Hasil Cluster q <= ClusterID
10. Lakukan perluasan hingga *seedlist* menjadi kosong atau semua terlalui.
11. ClusterID <= ClusterID + 1.
12. Kembali ke poin ke 3 dengan menggunakan data yang memiliki kerapatan yang lebih dari data awal dan tidak memiliki klaster.
13. Selesai.

Gambar 3. 5 *Pseudocode* Algoritma Klastering dengan Modofikasi DBSCAN

3.4 Desain Metode Penentuan Index Dunn

Indeks Dunn merupakan suatu indeks yang dapat digunakan untuk menguji validitas klaster. Desain metode ini dapat dilihat pada. Desain metode indeks dunn dapat dilihat pada Gambar 3.6.

Masukan	*Dataset(A x B), Hasil klaster(A x 1), Input m(1) berupa banyak klaster*
Keluaran	Nilai Indeks Dunn(1)

1. Cari nilai a yang merupakan jarak inter-klaster atau jarak terpendek antar data berdasarkan euclidean distance dalam klaster yang berbeda. Pencarian dilakukan pada semua kombinasi pasangan data.
2. Cari nilai b yang merupakan jarak intra-klaster atau jarak terjauh antar data berdasarkan euclidean distance dalam klaster yang sama. Pencarian dilakukan pada semua kombinasi pasangan data pada klaster yang sama.
3. Membagi nilai a dan nilai b yang menjadi nilai uji evaluasi Indeks Dunn.
4. Mengembalikan nilai Indeks Dunn sebagai keluaran.
5. Selesai

Gambar 3.6 *Pseudocode* Algoritma Penentuan Nilai Indeks Dunn

BAB IV
IMPLEMENTASI

Pada bab ini akan dijelaskan tahap implementasi yang dilakukan berdasarkan rancangan-rancangan yang telah dibahas pada bab-bab sebelumnya. Selain itu juga akan dibahas lingkungan yang digunakan untuk melakukan implementasi sistem.

4.1 Lingkungan Implementasi

Lingkungan implementasi yang akan digunakan untuk melaukan implementasi adalah *Microsoft Visual Studio* 2013 dengan memakai bahasa pemrograman C# yang diinstall pada sistem operasi *Windows* 8.1 64-bit.

4.2 Implementasi

Pada sub-bab ini akan dijelaskan implementasi setiap subbab yang terdapat pada bab sebelumnya yaitu bab perancangan program. Pada bagian implementasi ini juga akan dijelaskan mengenai fungsi-fungsi yang digunakan pada program tugas akhir ini dan disertai dengan kode sumber masing-masing fungsi utama.

4.2.1 Implementasi Algoritma *K-Nearest Neighbor* dan *Euclidean Distance*

Algoritma ini adalah yang pertama kali dilakukan pada program utama, dimana setiap pada dataset dicari tetangga – tetangga sebanyak k data dan juga jarak *k-nearest neighbor* yang didapatkan dengan mencari jarak terjauh dari data ke tetangga – tetangganya. Untuk mendapatkan jarak tersebut dibutuhkan pencarian jarak antar data menggunakan *euclidean distance*. Implementasi *euclidean distance* dapat dilihat pada kode sumber 4.1.

```
1  double hasil = 0;
2  for (int i = 0; i < jmlFitur; i++ )
3  {
4      double data1 = a[i];
5      double data2 = b[i];
6      hasil = hasil + ((data1 - data2) * (data1
   - data2));
7  }
8  Return Math.sqrt(hasil);
```

Kode Sumber 4.1 Kode Implementasi *Euclidean Distance*

Pada baris 1 dilakukan inisialisasi nilai hasil. Kemudian pada baris 2 – 7 dilakukan perulangan berdasarkan banyaknya fitur, yang dilakukan penjumlahan antara selisih kuadrat data ke-1 dan data ke-2 pada suatu fitur. Setelah seluruh fitur sudah dilakukan penjumlahan nilai hasil dikembalikan menjadi sebuah keluaran.

Implementasi algoritma pencarian jarak *k-nearest neighbor* dan penentuan tetangga pada setiap data terdapat pada kode sumber 4.2.

```
1  Dictionary<int, double> distance = new
   Dictionary<int, double>();
2  double[] euclidTemp = new double[jumlahData];
3  for (int i = 1; i <= jumlahData-1; i++)
4  {
5      if (i != obj)
6      {
7          Double tempDist =
   Math.Round(euclidDistanceAll(dataset[i],
   dataset[obj], 2), 10);
8          euclidTemp[i] = tempDist;
9          distance.Add(i, tempDist);
10     }
11 }
12 euclideanDist[obj] = euclidTemp;
13 List<KeyValuePair<int, double>> myList =
   distance.ToList();
14 myList.Sort((x, y) =>
   x.Value.CompareTo(y.Value));
15 List<int> neighbor = new List<int>();
16 for (int i = 0; i < k; i++)
17 {
```

```
18      KeyValuePair<int,double> temp = new
        KeyValuePair<int,double>(myList[i].Key,
        myList[i].Value);
19      neighbor.Add(myList[i].Key);
20      }
21      kNeighbor[obj] = neighbor;
22      knnDistance.Add(obj, myList[k - 1].Value);
```

Kode Sumber 4.2 Kode Implementasi *K-Nearest Neighbor*

Dalam baris ke-3 hingga baris ke-11 digunakan untuk mencari jarak antara 2 data. Kemudian data jarak tersebut disimpan pada baris ke-9. Setelah itu dilakukan pengurutan berdasarkan jarak seperti pada baris ke -14. Pada baris ke-16 hingga 20 memasukkan tetangga dari suatu data. Kemudian pada baris ke-21 dilakukan penyimpanan tetangga pada data *obj* dan penyimpanan jarak *k-nearest neighbor* pada baris ke 22

4.2.2 Implementasi Algoritma Pencarian Nilai Kerapatan Data dan Pengurutan Nilai Kerapatan Data

Algoritma ini adalah yang langkah kedua setelah penentuan jarak *k-nearest neighbor*. Implementasi penentuan nilai kerapatan terdapat pada Kode Sumber 4.3.

```
1   double hasil = 0;
2   for (int i = 0; i < k; i++ )
3   {
4       hasil = hasil + influenceFunction(dataset[x],
        dataset[kNeighbor[x].ElementAt(i)], 2);
5   }
6   Return hasil;
```

Kode Sumber 4.3 Kode Implementasi Penentuan Nilai Kerapatan

Pada baris ke-1 dilakukan inisialisasi untuk nilai kerapatan. Kemudian pada baris 2 hingga 5 dilakukan penentuan nilai kerapatan berdasarkan fungsi *influence* dari sebuah data dan tetangganya. Kemudian dilakukan keluaran berupa hasil jumlahan

fungsi *influence*. Untuk Implementasi penentuan fungsi *influence*, terdapat pada Kode Sumber 4.4.

```
1  double hasil = 0;
2  for (int i = 0; i < fitur; i++ )
3  {
4      double data1 = a[i];
5      double data2 = b[i];
6      hasil = hasil + ((data1 - data2) * (data1
   - data2));
7  }
8  Return Math.sqrt(hasil);
```

Kode Sumber 4.4 Kode Implementasi Fungsi *Influence*

Pada baris 1 dilakukan inisialisasi. Kemudian pada baris 2 hingga 7 dilakukan pengulangan untuk mengetahui jarak antara 2 data. Kemudian pada baris ke-8 dilakukan keluaran berupa akar dari hasil jumlahan perbedaan tiap fitur.

Setelah didapatkan nilai kerapatan untuk setiap data, dilakukan pengurutan nilai kerapatan yang nantinya digunakan pada saat pengelompokan data. Implementasi pengurutan nilai kerapatan terdapat pada Kode Sumber 4.5.

```
1  for (int j = 1; j <= jumlahData - 1; j++)
2  {
3      LD[j] = localDensity(j, k);
4  }
5  localDense = LD.ToList();
6  localDense.Sort((x,          y)          =>
   x.Value.CompareTo(y.Value));
```

Kode Sumber 4.5 Kode Implementasi Main Penentuan Nilai Kerapatan dan Pengurutan Nilai Kerapatan

Pada baris 1 sampai 4 dilakukan perulangan untuk mendapatkan nilai kerapatan setiap data. Kemudian pada baris 5 dan 6 dilakukan pengurutan mulai dari nilai kerapatan terkecil hingga terbesar (*ascending*).

4.2.3 Implementasi Algoritma Klastering menggunakan Modifikasi DBSCAN

Algoritma ini adalah yang langkah terakhir dalam pengajuan tugas akhir ini. Algoritma ini dilakukan setelah mendapatkan urutan data yang memiliki nilai kerapatan terkecil hingga terbesar.

```
1  List<int> seedList = new List<int>();
2  double eps = 0.0;
3  int clusID = 1;
4  for(int i = 0 ; i < localDense.Count;i++)
5  {
6    int p = localDense[i].Key;
7    eps = knnDistance[p];
8    if (clusters[p] == 0)
9    {
10     clusters[p] = clusID;
11     seedList = kNeighbor[p];
12     for (int j = 0; j < seedList.Count ; j++)
13     {
14     if(checkCorePoints(seedList[j],minPts,eps)
&& clusters[seedList[j]] == 0)
15      {
16        for (int a = 0; a <
kNeighbor[seedList[j]].Count; a++)
17        {
18          if
(seedList.Contains(kNeighbor[seedList[j]][a])
== false &&
clusters[kNeighbor[seedList[j]][a]] == 0 &&
euclidDistanceAll(dataset[kNeighbor[seedList[j
]][a]], dataset[seedList[j]],2) <= eps)
19            seedList.Add
(kNeighbor[seedList[j]][a]);
20        }
21      }
22     if(clusters[seedList[j]]== 0)
23       clusters[seedList[j]] = clusID;
24     }
25     clusID = clusID+1;
```

```
26    }
27    seedList = new List<int>();
28  }
29  Return Math.sqrt(hasil);
```

Kode Sumber 4.6 Kode Implementasi DBSCAN

Pada baris ke-1 hingga ke-4 dilakukan inisialisasi untuk nilai parameter densitas, id klaster dan *seedlist*. Pada baris ke-5 dilakukan perulangan berdasarkan jumlah data dan perulangan dimulai dari data yang memiliki nilai kerapatan terkecil misal data *p*. Baris ke-6 dan 7 berisi inisialisasi nilai parameter densitas dengan jarak *k-nearest neighbor* dari data *p*. Pada baris ke-8 dilakukan pengecekan klaster pada data *p*. Jika data *p* sudah memiliki klaster, maka proses berhenti dan dilanjutkan dengan data yang memiliki nilai kerapatan terkecil selanjutnya dan jika data *p* belum memiliki klaster maka proses dilanjutkan.

Pada baris ke-10 dan 11 dilakukan penetapan klaster pada data p dan memasukkan seluruh tetangga dari data *p* ke dalam *seedlist*. Kemudian pada baris ke-12 hingga baris ke-24 dilakukan perulangan yang digunakan untuk memperlebar klaster dari data *p*. Dimulai dari data yang paling dekat dengan data *p* misal data tersebut adalah data *q*, dilakukan pengecekan pada data q. Jika data *q* adalah sebuah *core point*, maka *seedlist* ditambahkan dengan tetangga data *q* yang ada pada parameter densitas (*epsilon*). Pada baris ke-16 hingga 20 dilakukan prosedur menambahkan *seedlist* dengan tetangga data *q*. Pada baris 22 dan 23 dilakukan pengecekan untuk data *q*. Jika data *q* tidak memiliki klaster maka klaster data q sama dengan klaster data *p*.

Terakhir ketika seedlist sudah dilewati semua atau kosong, maka id klaster ditambah satu seperti pada baris ke-25 dan mulai kembali pada perulangan dengan data yang memiliki nilai kerapatan terkecil selanjutnya.

Pada saat sebuah data ingin melakukan perluasan klaster dilakukan pengecekan pada data tersebut apakah data tersebut *core point* atau *border point*. Hal ini diimplementasikan pada Kode Sumber 4.7

```
1  int counter = 0;
2  for (int i = 0; i < kNeighbor[pt].Count; i++ )
3  {
4      double dist = euclideanDist[pt][kNeighbor[pt][i]];
5      if (dist <= eps)
6          counter++;
7  }
8  if (counter >= minPts)
9      return true;
10 else
11     return false;
```

Kode Sumber 4.7 Kode Implementasi Pengecekan *Core Point*

Pada pengecekan core point suatu data ada 1 syarat dimana tetangga pada data tersebut harus melebihi atau sama dengan *minPts* pada radius tertentu. Pengecekan tersebut dilakukan mulai dari baris ke-2 hingga baris ke-11 pada implementasi Kode Sumber 4.7

4.2.4 Implementasi Algoritma Penentuan Indeks Dunn untuk Evaluasi

Algoritma ini adalah algoritma untuk evaluasi hasil klaster. Nilai dari indeks dunn ini adalah sebuah angka berkisar dari 0 hingga tak hingga. Untuk mendapatkan nilai indeks dunn diperlukan jarak inter-klaster dan jarak intra-klaster. Implementasi penentuan jarak inter-klaster terdapat pada Kode Sumber 4.8 dan penentuan jarak intra-klaster terdapat pada Kode Sumber 4.9.

Indeks dunn dihitung dengan mencari nilai minimum jarak antar klaster dan maksimum jarak intra-klaster. Implementasi penentuan indeks dunn terdapat pada Kode Sumber 4.10.

```
1  List<int> c1 = new List<int>();
2  List<int> c2 = new List<int>();
3  double interCluster = 999999;
4  for(int i = 1 ; i <= jumlahData-1 ; i++ )
5  {
6      if (clusters[i] == cluster1)
```

```
7         c1.Add(i);
8      else if (clusters[i] == cluster2)
9         c2.Add(i);
10  }
11  for(int i = 0 ; i < c1.Count ; i++)
12  {
13     for(int j = 0 ; j < c2.Count ; j++)
14     {
15        if (euclideanDist[c1.ElementAt(i)][c2.ElementAt(j)] < interCluster)
16           interCluster = euclideanDist[c1.ElementAt(i)][c2.ElementAt(j)];
17     }
18  }
19  return interCluster;
```

Kode Sumber 4.8 Kode Implementasi Penentuan Jarak *InterCluster* (C_i, C_j)

Untuk penentuan jarak inter-klaster, pada baris 1 hingga 3 dilakukan inisialisasi. Kemudian pada baris 4 hingga 10 penentuan anggota tiap klasternya. Seperti halnya *k-nearest neighbor* penentuan jarak pada Kode Sumber 4.8 juga memakai *euclidean distance*. Pada baris ke 15 dan 16 dilakukan pengecekan untuk mencari jarak minimum antar klaster.

```
1   List<int> c1 = new List<int>();
2   List<int> c2 = new List<int>();
3   double intraCluster = 0;
4   for(int i = 1 ; i <= jumlahData-1 ; i++ )
5   {
6      if (clusters[i] == cluster1)
7         c1.Add(i);
8      else if (clusters[i] == cluster2)
9         c2.Add(i);
10  }
11  for(int i = 0 ; i < c1.Count ; i++)
12  {
13     for(int j = 0 ; j < c2.Count ; j++)
```

```
14      {
15          if (euclideanDist[c1.ElementAt(i)][c2.ElementAt(j)] > intraCluster)
16              intraCluster = euclideanDist[c1.ElementAt(i)][c2.ElementAt(j)];
17      }
18  }
19  return intraCluster;
```

Kode Sumber 4.9 Kode Implementasi Penentuan Jarak *IntraCluster* (C_i)

Untuk penentuan jarak intra-klaster, pada baris 1 hingga 3 dilakukan inisialisasi. Kemudian pada baris 4 hingga 10 penentuan anggota tiap klasternya. Kemudian pada baris ke 15 dan 16 dilakukan pengecekan untuk mencari jarak maksimum data pada tiap klaster.

```
1   double indexDunn = 0.0;
2   int numClust = (input);
3   double minInter = 999999;
4   double maxIntra = 0.0;
5   for (int i = 1; i <= numClust; i++ )
6   {
7       for (int i = 1; j <= numClust; j++ )
8       {
9           if(i!=j)
10          {
11              double inter = interClusterDistance(i, j);
12              if (inter < minInter)
13                  minInter = inter;
14          }
15      }
16      double intra = intraClusterDistance(i);
17      if(intra > maxIntra)
18          maxIntra = intra;
19  }
20  indexDunn = Math.Round(minInter / maxIntra, 10);
```

```
21  return indexDunn
```

Kode Sumber 4.10 Kode Implementasi Penentuan Nilai Indeks Dunn

Pada perhitungan indeks dunn diperlukan nilai inter-klaster dan intra-klaster yang diimplementasikan pada baris 5 hingga baris 19. Setelah mendapat kedua nilai yang dibutuhkan untuk penghitungan, dilakukan penghitungan untuk mendapatkan nilai indeks dunn yang terdapat pada baris ke-20

BAB V
UJI COBA DAN EVALUASI

Pada bab ini akan dibahas mengenai tahap uji coba yang telah dilakukan terhadap sistem beserta evaluasi dari hasil uji coba tersebut. Uji coba dilakukan dengan mengundang penguji untuk melakukan pengujian berdasarkan kontrol dari penulis. Aspek yang diperhatikan dalam pengujian adalah terpenuhinya fungsionalitas yang sudah dirancang pada kasus pengujian.

5.1 Lingkungan Uji Coba

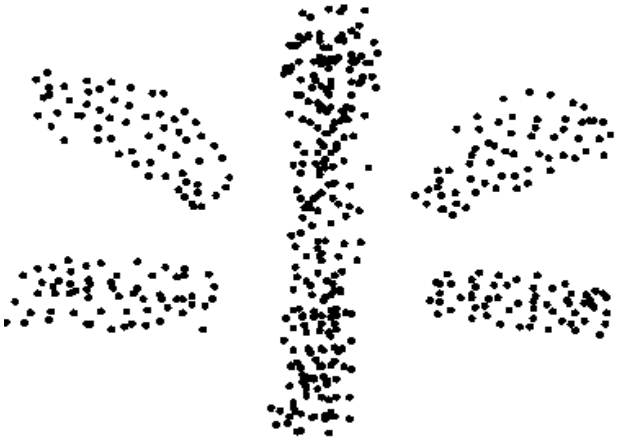
Lingkungan uji coba menjelaskan lingkungan yang digunakan untuk menguji implementasi penentuan nilai parameter densitas dan klastering berdasarkan DBSCAN pada Tugas Akhir ini. Lingkungan uji coba meliputi perangkat keras dan perangkat lunak yang dijelaskan sebagai berikut

1. Perangkat keras
   a) Processor : Intel ® Core ™ i7-4700HQ CPU @2.40GHz (8 CPUs), ~2.4GHz
   b) *Memory* RAM : 8,00 GB
   c) Tipe Sistem : 64-bit Sistem Operasi
2. Perangkat lunak
   a) Sistem Operasi : *Windows* 8.1 Pro
   b) Perangkat pengembang : *Microsoft Visual Studio* 2013

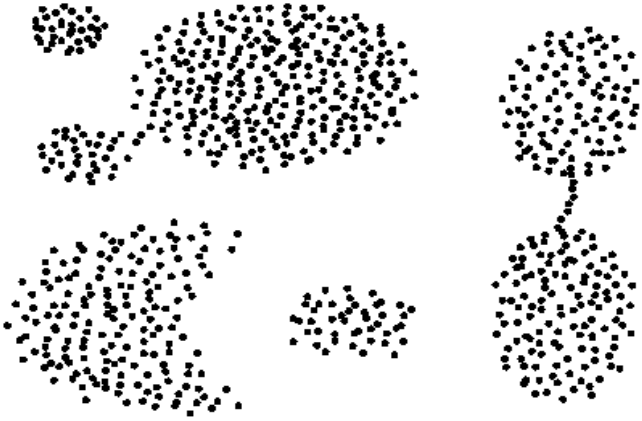
5.2 Data Uji Coba

Data yang digunakan merupakan data sintesis. Data sintesis adalah data yang didesain menyerupai situasi tertentu, yang mungkin tidak ditemui pada data *real*. Dalam Tugas akhir ini, akan dibentuk data sintesis yang menyerupai bentuk – bentuk tertentu dan memiliki densitas yang berbeda – beda.

Jumlah data yang akan dilakukan pengujian sebanyak 6 data. Berikut adalah visualisasi *dataset* yang digunakan.

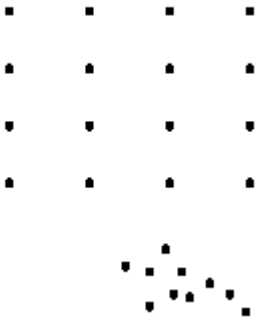


Gambar 5.1 Visualisasi *Dataset* Sintesis T1



Gambar 5.2 Visualisasi *Dataset* Sintesis T2

Gambar 5.3 Visualisasi *Dataset* Sintesis T3



Gambar 5.4 Visualisasi *Dataset* Sintesis T4

Gambar 5.5 Visualisasi *Dataset* Sintesis T5

Gambar 5.6 Visualisasi *Dataset* Sintesis T6

5.3 Skenario dan Evaluasi Pengujian

Uji coba dilakukan untuk menguji apakah fungsionalitas program telah diimplementasikan dengan benar dan berjalan sebagaimana mestinya. Uji coba akan didasarkan pada beberapa scenario untuk menguji kesesuaian dan kinerja algoritma.

Pengujian dilakukan dengan uji validitas klaster dengan Indeks Dunn. Uji coba dilakukan sebanyak 6 kali yang terdiri dari 6 data sintesis yang berbeda – beda. Pengujian juga dilakukan dengan mapping hasil klaster dengan kelas yang ada pada sebuah dataset. Uji coba tersebut dilakukan untuk 2 *dataset* yang memiliki fitur kelas pada datanya. Detail data sintesis yang diberikan dapat dilihat pada Tabel 5.1.

Tabel 5.1 Detail *Dataset* Sintesis

Percobaan	Dataset yang dipakai	Ukuran Data (Jumlah X Fitur)	Jumlah Klaster
1	T1	500 x 2	5
2	T2	788 x 2	7
3	T3	150 x 5	3
4	T4	26 x 2	2
5	T5	373 x 2	2
6	T6	8000 x 2	6

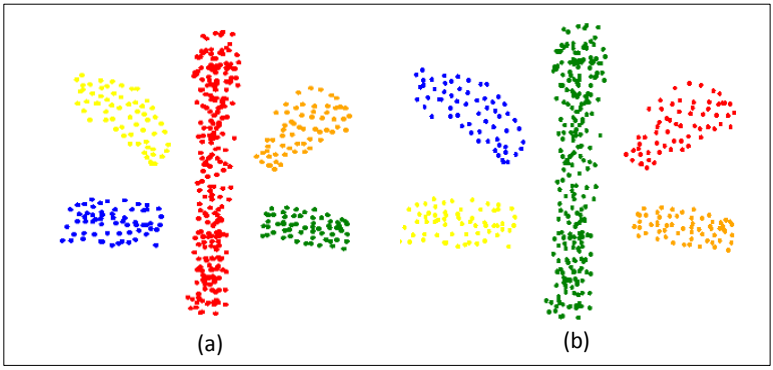
Uji coba juga dilakukan dengan menggunakan algoritma DBSCAN tanpa adanya modifikasi. Pada DBSCAN masukan berupa nilai *Epsilon* terbaik yang sudah dicoba beberapa kali untuk mendapatkan klaster pada *dataset* dengan nilai minPts yang sama dengan DBSCAN modifikasi.

Skenario uji coba untuk tiap data berbeda – beda akan tetapi memiliki nilai *minPts* yang sama untuk setiap metode. Skenario uji coba didetailkan seperti berikut ini

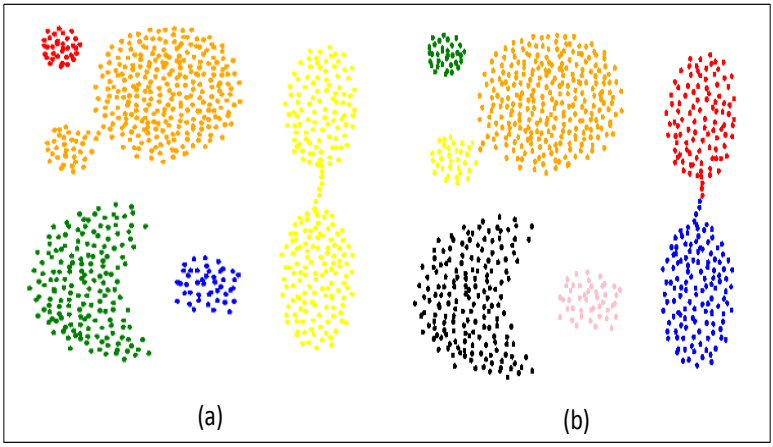
1. Untuk Data T1. Pada DBSCAN diberikan masukan nilai *Eps* = 0.06 dan *minPts* = 4 Sedangkan pada DBSCAN modifikasi diberikan nilai $k = 25$ dan *minPts* = 4.

2. Untuk Data T2. Pada DBSCAN diberikan masukan nilai *Eps* = 1.7 dan *minPts* = 7 Sedangkan pada DBSCAN modifikasi diberikan nilai *k* = 17 dan *minPts* = 7.
3. Untuk Data T3. Pada DBSCAN diberikan masukan nilai *Eps* = 0.1 dan *minPts* = 4 Sedangkan pada DBSCAN modifikasi diberikan nilai *k* = 14 dan *minPts* = 4.
4. Untuk Data T4. Pada DBSCAN diberikan masukan nilai *Eps* = 1.1 dan *minPts* = 4 Sedangkan pada DBSCAN modifikasi diberikan nilai *k* = 7 dan *minPts* = 4.
5. Untuk Data T5. Pada DBSCAN diberikan masukan nilai *Eps* = 2.5 dan *minPts* = 4 Sedangkan pada DBSCAN modifikasi diberikan nilai *k* = 22 dan *minPts* = 4.
6. Untuk Data T6. Pada DBSCAN diberikan masukan nilai *Eps* = 5.5 dan *minPts* = 4 Sedangkan pada DBSCAN modifikasi diberikan nilai *k* = 20 dan *minPts* = 4.

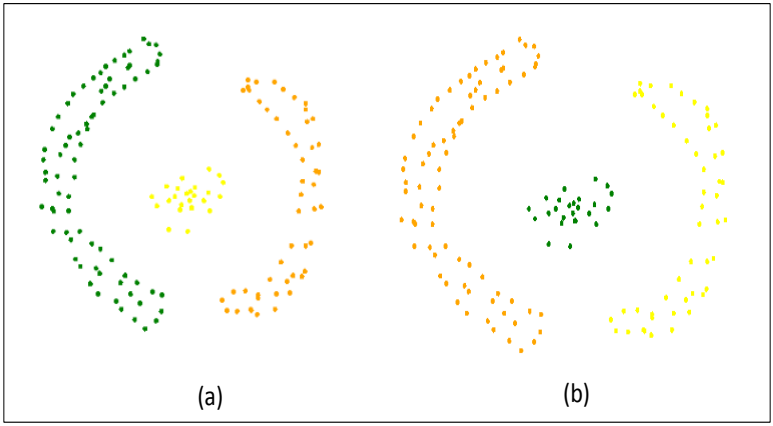
Berikut visualisasi hasil uji coba untuk setiap *dataset* dan metode yang digunakan.



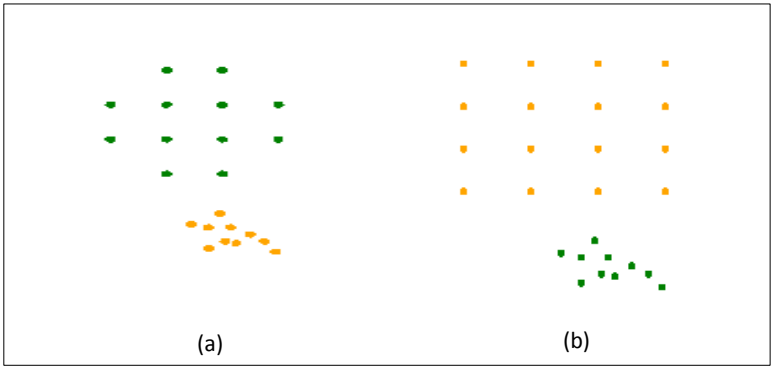
Gambar 5.7 Visualisasi Hasil Ujicoba *dataset* T1. (a) Metode DBSCAN dengan *Eps* = 0.06 dan *MinPts* = 4. (b) Metode DBSCAN modifikasi dengan *k* = 25 dan *MinPts* = 4.



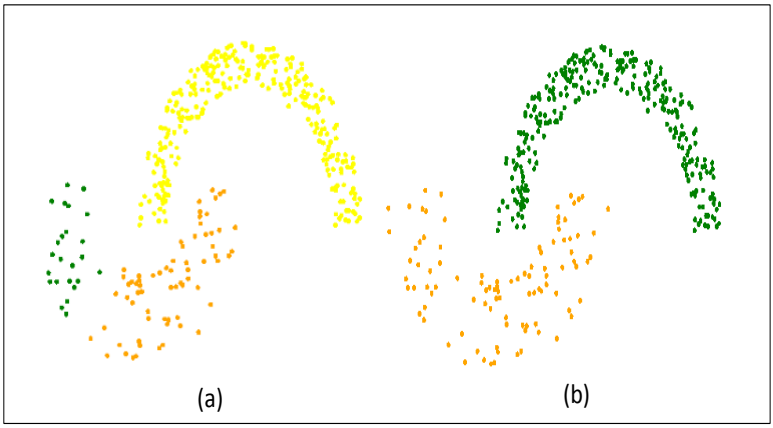
Gambar 5.8 Visualisasi Hasil Ujicoba *dataset* T2. (a) Metode DBSCAN dengan *Eps* = 1.7 dan *MinPts* = 7. (b) Metode DBSCAN modifikasi dengan *k* = 17 dan *MinPts* = 7.



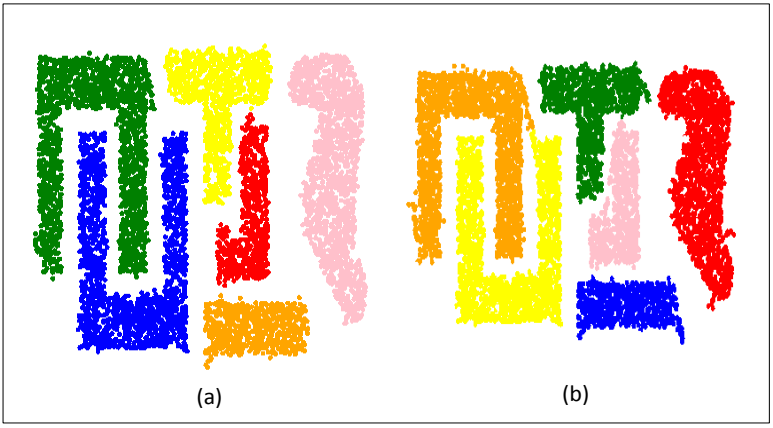
Gambar 5.9 Visualisasi Hasil Ujicoba *dataset* T3. (a) Metode DBSCAN dengan *Eps* = 0.1 dan *MinPts* = 4. (b) Metode DBSCAN modifikasi dengan *k* = 14 dan *MinPts* = 4.



Gambar 5.10 Visualisasi Hasil Ujicoba dataset T4. (a) Metode DBSCAN dengan *Eps* = 1.1 dan *MinPts* = 4. (b) Metode DBSCAN modifikasi dengan *k* = 7 dan *MinPts* = 4.



Gambar 5.11 Visualisasi Hasil Ujicoba dataset T4. (a) Metode DBSCAN dengan *Eps* = 2.5 dan *MinPts* = 4. (b) Metode DBSCAN modifikasi dengan *k* = 22 dan *MinPts* = 4.



Gambar 5.12 Visualisasi Hasil Ujicoba *dataset* T4. (a) Metode DBSCAN dengan *Eps* = 5.5 dan *MinPts* = 4. (b) Metode DBSCAN modifikasi dengan k = 20 dan *MinPts* = 4.

Hasil ujicoba berdasarkan nilai Indeks Dunn pada skenario sebelumnya terdapat pada Tabel 5.2 . Kolom yang diberi warna hijau adalah kolom yang memiliki nilai Indeks Dunn lebih baik sedangkan kolom yang diberi warna biru adalah kolom yang memiliki jumlah klaster yang sama dengan jumlah klaster yang seharusnya.

Tabel 5.2 Tabel Hasil Uji Coba

Dataset yang digunakan	Indeks Dunn		Jumlah Klaster	
	DBSCAN Modifikasi	DBSCAN	DBSCAN Modifikasi	DBSCAN
T1	0.085	0.085	5	5
T2	0.033	0.108	7	5
T3	0.219	0.219	3	3
T4	0.271	0.364	2	2
T5	0.092	0.092	2	3
T6	0.020	0.008	6	6

Diantara 6 data sintesis yang ada, terdapat 2 data yang datanya memiliki kelas karena data tersebut didapatkan dari sebuah *website* [15]. 2 Data tersebut ada terdapat pada data sintesis T2 dan T5. Dengan adanya kelas yang ada pada 2 data tersebut dapat dilakukan uji dengan menggunakan *ground truth*. Label klaster yang didapatkan dari DBSCAN dan DBSCAN Modifikasi dicocokkan dengan label klaster *ground truth*. Hasil perbandingan untuk data T2 terdapat pada Tabel 5.3 dan Tabel 5.4 dan hasil perbanding untuk data T5 terdapat pada Tabel 5.5. Sebagai contoh pada Tabel 5.5 yang memiliki jumlah klaster lebih sedikit, dengan menggunakan DBSCAN modifikasi data T5 dibagi menjadi 2 klaster antara lain klaster ke-1 memiliki 276 data dan klaster ke-2 memiliki 97 data. Terdapat klaster yang bernilai 0 atau ketika sebuah data tidak memiliki klaster.

Tabel 5.3 Tabel Hasil Ujicoba *Dataset* T2 dengan DBSCAN Modifikasi

		Klaster Prediksi DBSCAN Modifikasi							
Klaster Sebenarnya	**Ground Truth**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**0**
1	**45**	**0**	**0**	**0**	**0**	**45**	**0**	**0**	**0**
2	**170**	**0**	**0**	**0**	**0**	**0**	**169**	**0**	**1**
3	**102**	**0**	**0**	**0**	**0**	**0**	**0**	**102**	**0**
4	**273**	**0**	**273**	**0**	**0**	**0**	**0**	**0**	**0**
5	**34**	**34**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
6	**130**	**0**	**0**	**0**	**129**	**0**	**0**	**1**	**0**
7	**34**	**0**	**0**	**34**	**0**	**0**	**0**	**0**	**0**

Tabel 5.4 Tabel Hasil Ujicoba *Dataset* T2 dengan DBSCAN

		Klaster Prediksi DBSCAN					
Klaster Sebenarnya	**Ground Truth**	**1**	**2**	**3**	**4**	**5**	**0**
1	**45**	**0**	**0**	**0**	**45**	**0**	**0**
2	**170**	**169**	**0**	**0**	**0**	**0**	**1**

3	**102**	**0**	**0**	**102**	**0**	**0**	**0**
4	**273**	**0**	**273**	**0**	**0**	**0**	**0**
5	**34**	**0**	**0**	**0**	**0**	**34**	**0**
6	**130**	**0**	**0**	**130**	**0**	**0**	**0**
7	**34**	**0**	**34**	**0**	**0**	**0**	**0**

Tabel 5.5 Tabel Hasil Ujicoba *Dataset* T5

		Klaster Prediksi DBSCAN Modifikasi		**Klaster Prediksi DBSCAN**			
Klaster Sebenarnya	**Ground Truth**	**1**	**2**	**1**	**2**	**3**	**0**
1	**276**	**276**	**0**	**0**	**0**	**276**	**0**
2	**97**	**0**	**97**	**24**	**68**	**0**	**5**

5.4 Analisa Hasil Uji Coba

Analisa dilakukan berdasarkan hasil dari tiap scenario uji coba dengan evaluasi tiap pengujiannya.

Berdasarkan Gambar 5.7 Visualisasi Hasil Ujicoba *dataset* T1. (a) Metode DBSCAN dengan *Eps* = 0.06 dan *MinPts* = 4. (b) Metode DBSCAN modifikasi dengan k = 25 dan *MinPts* = 4. Pada uji coba untuk data T1 didapatkan DBSCAN dan DBSCAN yang dimodifikasi memiliki hasil yang sama. Akan tetapi pada Gambar 5.8 Visualisasi Hasil Ujicoba *dataset* T2. (a) Metode DBSCAN dengan *Eps* = 1.7 dan *MinPts* = 7. (b) Metode DBSCAN modifikasi dengan k = 17 dan *MinPts* = 7.. Untuk uji coba data T2 mengalami perbedaan jumlah klaster yang dihasilkan. Meskipun berdasarkan Tabel 5.2 Tabel Hasil Uji Coba didapatkan bahwa nilai indeks Dunn algoritma DBSCAN lebih tinggi dibandingkan algoritma DBSCAN modifikasi, jumlah klaster yang dihasilkan lebih sedikit dibandingkan jumah klaste yang seharusnya Hal tersebut terjadi dikarenakan kerapatan antara data yang besar dan kecil pada data T2 yang berdekatan.

Kemudian pada uji coba untuk data T3, didapatkan bahwa data tersebut memiliki 3 klaster. Kedua metode yang digunakan dapat mengelompokkanya dengan baik.

Pada uji coba untuk data T4 yang merupakan *dataset* terkecil yang diujikan, kedua metode yang diujikan memiliki jumlah klaster sebanyak 2. Akan tetapi hasil dari DBSCAN membuang 4 data yang dianggapnya sebagai *noise* padahal keempat data tersebut merupakan data dari sebuah klaster. Pada DBSCAN modifikasi memiliki 2 nilai epsilon yang salah satunya kecil untuk data yang rapat dan besar untuk data yang renggang. Pada Tabel 5.2 Tabel Hasil Uji Coba nilai indeks dunn pada DBSCAN memiliki nilai yang lebih besar dibandingkan DBSCAN modifikasi, akan tetapi performa untuk dataset ini DBSCAN modifikasi memiliki hasil yang lebih baik dari pada DBSCAN.

Pada uji coba untuk data T5, jumlah klaster yang dihasilkan dengan DBSCAN lebih banyak dibandingkan DBSCAN modifikasi. Hal ini dikarenakan nilai *epsilon* pada DBSCAN yang tidak bisa menangani densitas dari klaster ke-2. Meskipun nilai *epsilon* sudah diganti – ganti, nilai *epsilon* yang paling bagus terdapat pada *epsilon* 2.5. Berbeda dengan DBSCAN, pada DBSCAN modifikasi hasil yang diberikan lebih memuaskan. Yakni dengan 2 klaster yang tepat mengelompokkan antar densitas tiap klaster yang berbeda. Berdasarkan Tabel 5.2 Tabel Hasil Uji Coba, Kedua metode pada data T5 memiliki nilai indeks dunn yang sama yaitu 0.092 akan tetapi hasil antar metode tersebut tidak sama.

Pada Uji coba untuk data T6 yang merupakan *dataset* terbesar pada pengujiannya, kedua metode mendapatkan jumlah klaster yang sama yaitu 6 akan tetapi beberapa bentuk yang dihasilkan sedikit berbeda. Berdasarkan Tabel 5.2 Tabel Hasil Uji Coba, Nilai indeks dunn yang lebih baik adalah pada DBSCAN Modifikasi dengan nilai 0.02 sedangkan indeks dunn DBSCAN hanya 0.008.

Setelah analisa pengujian dengan menggunakan hasil klaster, akan dianalisa berdasarkan *ground truth* untuk data T2 dan T5.

Pada Data T2, Untuk menggunakan klaster prediksi dari DBSCAN Modifikasi diketahui terdapat hanya 2 kesalahan yang dihasilkan dari *ground truth*nya. Hal ini pun dikarenakan nilai epsilon yang berbeda – beda dan dimulai dari paling rapat. Nilai 0 dianggap sebagai *noise* atau *outlier* pada DBSCAN Modifikasi dikarenakan hasil klaster 0 memiliki data hanya 1.

Sedangkan pada DBSCAN klaster prediksinya hanya memiliki 5 klaster dan 1 data yang termasuk *noise* atau *outlier*. Penggabungan terjadi pada 2 klaster prediksi yaitu klaster 3 dan klaster 2. Hal ini disebabkan oleh nilai *epsilon* yang tidak tepat mengelompokkan tiap data.

Kemudian pada data T5, dilakukan pengujian yang sama. Pada data T5 klaster prediksi untuk DBSCAN Modifikasi memiliki 100% sama dengan hasil klaster yang sebenarnya. Akan tetapi klaster prediksi dari DBSCAN untuk klaster ke-2 terpecah menjadi 24 data pada klaster prediksi 1, 68 data pada klaster prediksi 2 dan 5 data *noise*. Hal ini terjadi dikarenakan densitas pada klaster 1 dan klaster 2 sangat amat berbeda. Untuk itu dibutuhkan nilai *epsilon* yang berbeda tiap klasternya.

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas mengenai kesimpulan yang dapat diambil dari pengerjaan Tugas Akhir ini berdasarkan hasil pengujian dan temuan temuan lainnya. Selain kesimpulan, juga terdapat saran yang penulis ajukan terhadap pengembangan Tugas Akhir ini ke depannya.

6.1 Kesimpulan

Dari hasil uji coba yang telah dilakukan terhadap pembuatan model, dapat diambil kesimpulan sebagai berikut.

1. Algoritma Penentuan Nilai Parameter Densitas pada DBSCAN dapat digunakan untuk menyelesaikan masalah dari DBSCAN yakni dapat mengelompokkan data yang memiliki densitas yang beragam.
2. Berdasarkan uji validitas klaster, didapatkan bahwa hasil klaster yang didapat dari DBSCAN modifikasi kurang baik jika dibandingkan dengan DBSCAN dengan nilai rata – rata indeks dunn 0.12 dan 0.146.
3. Berdasarkan uji coba dengan *mapping* klaster, didapatkan bahwa klaster yang dihasilkan DBSCAN modifikasi lebih baik jika dibandingkan dengan DBSCAN

6.2 Saran

Saran yang diberikan untuk pengembangan program ini adalah:

1. Untuk meningkatkan waktu pemrosesan, dapat digunakan *library* untuk penghitungan nilai *k-nearest neighbor.*
2. Perlu penelitian lebih lanjut mengenai pengambilan nilai awal yang digunakan sebagai penentu nilai parameter densitas.

DAFTAR PUSTAKA

[1] Mathworks, "http://www.mathworks.com/discovery/pattern-recognition.html," [Online].

[2] A. M. Fahim, "A Clustering Algorithm for Discovering Varied Density Clusters," *International Research Journal of Engineering and Technology (IRJET),* vol. 02, no. 08, 2015.

[3] H.-P. K. J. S. X. X. Martin Ester, "A Density-Based Algorithm for Discovering Clusters," *AAAI,* 1996.

[4] Y. A. A. R. B. M. Emre Celebi, "Mining Biomedical Images with Density-based Clustering," *Proceedings of the International Conference on Information Technology: Coding and Computing,* 2005.

[5] F. D.-Ç. A. Ş. D. Mete ÇELİK, "Anomaly Detection in Temperature Data Using," *IEEE,* 2011.

[6] R. F. M. a. N. I. G. Yomna M. ElBarawy, "Improving Social Network Community Detection," *IEEE,* 2014.

[7] D. A. D. K.Santhisree, "Web Usage Data Clustering using Dbscan algorithm and Set similarities," *International Conference on Data Storage and Data Engineering,* 2010.

[8] Y. K. Z. J. Wu Ying, "Using DBSCAN Clustering Algorithm in Spam Identifying," *2nd International Conference on Education Technology and Computer (ICETC),* 2010.

[9] H. C. J. Y. J. L. H.-S. W. G. H. a. L. L. Zhiwen Yu, "Adaptive Fuzzy Consensus Clustering Framework for Clustering Analysis of Cancer Data," *ACM TRANSACTIONS ON COMPUTATIONAL BIOLOGY AND BIOINFORMATICS,* 2015.

[10] K. K. Christina Jayakumaran, "Pattern Identification using Rough Set Clustering for Spatio-Temporal Dataset," *IEEE,* 2013.

[11] M. M. L. M. H.-V. Carlos Cobos, "Clustering of Web Search Results based on an Iterative Fuzzy C-means Algorithm and Bayesian Information Criterion," *IEEE,* 2013.

[12] P. L. P. D. S. C. Ashwini Gulhane, "A Review of Image Data Clustering Techniques," *International Journal of Soft Computing and Engineering (IJSCE),* 2012.

[13] P.-N. Tan, M. Steinbach and V. Kumar, Introduction to Data Mining, 2006.

[14] J. C. Dunn, "A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters," *Journal of Cybernetics 3,* 1973.

[15] "Clustering datasets," Speech and Image Processing Unit, School of Computing, University of Eastern Finland. [Online].

BIODATA PENULIS

Penulis dilahirkan di Sidoarjo, 13 Mei 1995, merupakan anak ketiga dari 3 bersaudara. Penulis telah menempuh pendidikan formal yaitu TK Tunas Islam (1999-2001), MI Pucang Sidoarjo (2001-2006), SMP Negeri 5 Sidoarjo (2006-2009), SMA Negeri 1 Sidoarjo (2009-2011), dan mahasiswa S1 Jurusan Teknik Informatika Institut Teknologi Sepuluh Nopember Surabaya. Selama masa kuliah, penulis aktif dalam organisasi Himpunan Mahasiswa Teknik Computer (HMTC). Diantaranya adalah menjadi staff departemen Media Informasi Himpunan Mahasiswa Teknik Computer ITS 2012-2013. Penulis juga aktif dalam kegiatan kepanitiaan Schematics. Diantaranya penulis pernah menjadi staff Hubungan Masyarakat Schematics 2012 dan staff Kesertariatan Schematics 2013. Selama kuliah di teknik informatika ITS, penulis mengambil bidang minat Komputasi Cerdas Visi (KCV). Komunikasi dengan penulis dapat melalui email: abakhrul.ilmi@gmail.com