PENERAPAN *COSINE SIMILARITY* PADA ALGORITMA *SHRINKING BASED SHARED NEAREST NEIGHBOR* (SSNN) PADA METODE *CLUSTERING*

SKRIPSI

Oleh :
M KHULUQ FIRMANSYAH
0710963002-96

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG
2012

UNIVERSITAS BRAWIJAYA

PENERAPAN *COSINE SIMILARITY* PADA ALGORITMA *SHRINKING BASED SHARED NEAREST NEIGHBOR* (SSNN) PADA METODE *CLUSTERING*

SKRIPSI

Sebagai salah satu syarat untuk memperoleh gelar
Sarjana Komputer dalam bidang Ilmu Komputer

Oleh :
M KHULUQ FIRMANSYAH
0710963002-96

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG
2012

LEMBAR PENGESAHAN SKRIPSI

PENERAPAN *COSINE SIMILARITY* PADA ALGORITMA *SHRINKING BASED SHARED NEAREST NEIGHBOR* (SSNN) PADA METODE *CLUSTERING*

Oleh :

M KHULUQ FIRMANSYAH
0710963002-96

Setelah dipertahankan di depan Majelis Penguji
pada tanggal 10 Juli 2012
dan dinyatakan memenuhi syarat untuk memperoleh gelar Sarjana Komputer dalam bidang Ilmu Komputer

Pembimbing I,	Pembimbing II,
Lailil Muflikhah, S.Kom, M.Sc	**Djoko Pramono, ST**
NIP 19741113 200501 2 001	**NIP 19780108 200501 1 002**

Mengetahui,
Ketua Jurusan Matematika
Fakultas MIPA Universitas Brawijaya

Dr. Abdul Rouf Alghofari, M.Sc
NIP 19670907 199203 1 001


UNIVERSITAS BRAWIJAYA

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini :

Nama	**: M. Khuluq Firmansyah**
NIM	**: 0710963002-96**
Jurusan	**: Matematika**
Program Studi	**: Ilmu Komputer**
Berjudul	**: Penerapan *Cosine Similarity* pada Algoritma *Shrinking based Shared Nearest Neighbor* (SSNN) pada Metode *Clustering***

Dengan ini menyatakan bahwa :

1. **Isi dari tugas akhir yang saya buat adalah benar-benar karya sendiri dan tidak menjiplak karya orang lain, selain nama-nama yang termaktub di isi dan tertulis di daftar pustaka dalam tugas akhir ini.**
2. **Apabila dikemudian hari ternyata tugas akhir yang saya tulis terbukti hasil jiplakan, maka saya akan bersedia menanggung segala resiko yang akan saya terima.**

Demikian pernyataan ini dibuat dengan segala kesadaran.

Malang, 10 Juli 2012
Yang menyatakan,

M. Khuluq Firmansyah
NIM 0710963002


UNIVERSITAS BRAWIJAYA

PENERAPAN *COSINE SIMILARITY* PADA ALGORITMA *SHRINKING BASED SHARED NEAREST NEIGHBOR* (SSNN) PADA METODE *CLUSTERING*

ABSTRAK

Algoritma *Shrinking based Shared Nearest Neighbor* merupakan perbaikan dari algoritma *Shared Nearest Neighbor* yang mana algoritma SSNN menggunakan konsep pergerakan data dari algoritma data *shrinking* untuk memperbesar akurasi yang diperoleh. Dalam algoritma SSNN terdapat proses perhitungan jarak antar data menggunakan *euclidean distance*. Pada skripsi ini akan dilakukan penggantian *euclidean distance* dengan *cosine similarity* sebagai rumus untuk mencari jarak antar data.

Awalnya *dataset* dinormalisasi terlebih dahulu, kemudian dihitung nilai *similarity* dengan menggunakan rumus *cosine similarity*. Kemudian nilai *similarity* tersebut dirubah ke nilai *dissimilarity*. *User* memasukan parameter k (nilai tetangga terdekat dari tiap titik yang diambil untuk proses perhitungan), CP (batas nilai minimal pemutusan bobot ketetanggaan) dan MP (batas nilai penghentian iterasi). Nilai *dissimilarity* dimasukan ke tabel jarak terdekat sebanyak k data. tabel jarak terdekat kemudian diurutkan dari nilai terkecil sampai nilai terbesar. Nilai *dissimilarity* yang lebih besar dari nilai CP akan diputuskan. Kemudian dihitung nilai prosentase perubahan bobot. Proses pemutusan dan perhitungan prosentase perubahan bobot dilakukan dalam beberapa iterasi. Tiap iterasi nilai CP akan terus diturunkan sebesar 0,01. Iterasi akan dihentikan jika nilai prosentase perubahan bobot lebih kecil dari nilai MP. Jika iterasi telah dihentikan maka dilakukan proses pembentukan *cluster*. Proses pembentukan *cluster* dibagi menjadi dua tahap. Tahap pertama memasukan titik ke dalam *cluster* awal dan tahap kedua adalah menggabungkan *cluster* yang telah terbentuk menjadi *cluster* sejumlah kelas awal *dataset*. Kemudian yang terakhir dihitung akurasi dari algoritma SSNN.

Hasil dari *clustering dataset* dengan algoritma SSNN adalah tingkat akurasi. Dari hasil uji coba untuk dua *dataset* yaitu *dataset*

iris dan *dataset wine* dengan menggunakan dua rumus jarak yaitu *euclidean distance* dan *cosine similarity* didapatkan bahwa akurasi algoritma SSNN untuk *dataset wine* dengan menggunakan *cosine similarity* adalah 0,513 sedangkan dengan menggunakan *euclidean distance* adalah 0,456. Akurasi algoritma SSNN untuk *dataset iris* dengan menggunakan *cosine similarity* adalah 0,74 sedangkan dengan menggunakan *euclidean distance* adalah 0,727. Dari nilai akurasi tersebut dapat diambil kesimpulan bahwa akurasi algoritma SSNN untuk kedua *dataset* tersebut menghasilkan akurasi yang lebih tinggi jika menggunakan *cosine similarity* dari pada menggunakan *euclidean distance* dengan selisih akurasi untuk *dataset wine* adalah 0,057 dan selisih akurasi untuk *dataset iris* adalah 0,013.

APPLYING COSINE SIMILARITY FOR SHRINKING BASED SHARED NEAREST NEIGHBOR (SSNN) IN CLUSTERING METHOD

ABSTRACT

Shrinking algorithm based Shared Nearest Neighbor algorithm is an improvement of Shared Nearest Neighbor algorithm SSNN which uses the concept of data movement algorithm to increase the accuracy of shrinking the data obtained. SSNN algorithms contained in the calculation of distances between the data using the euclidean distance. In this thesis will be the replacement of euclidean distance with cosine similarity as the formula to find the distance between the data.

First, dataset is normalized, then dissimilarity is computed using cosine similarity. Similarity value will be changed into disimilarity value. User determines three parameters, k, CP, MP. Disimilarity value is saved in closest-distance table and sorted ascending. Dissimilarity value which is bigger than CP value will be eliminated. Then a percentage of weight change will be calculated. Elimination process and a percentage of weight change calculation are done in some iteration. In each iteration, Cp value will be decreased 0,01continuously. Iteration will be terminated if percentage of weight change is smaller than MP. Then cluster is built into two steps. The first step is assigning points into first cluster and the next step is merging cluster which has built before based on the amount of class dataset. At last, the accuracy of SSNN algorithm is calculated.

This research used two datasets from UCI Learning machine repository: Iris and Wine. Cosine similarity and Euclidean distance are used and implemented as distance measurement. The experimental result shows that the accuracy of SSNN algorithm with Cosine similarity for wine dataset is 0,513, for iris dataset is 0,74 and with Euclidean distance for wine dataset is 0,456, for iris dataset is 0,727. From the result, it can be concluded that the accuracy of SSNN algorithm with Cosine similarity for both datasets is bigger

than with Euclidean distance which has 0,057 as the difference value for wine dataset and 0,013 as the difference value for iris dataset.

KATA PENGANTAR

Alhamdulillah, dengan memanjatkan puji syukur kehadirat Allah SWT yang telah memberikan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan skripsi ini. Skripsi yang berjudul "**Penerapan *Cosine Similarity* pada Algoritma *Shrinking Based Shared Nearest Neighbor* (SSNN) pada Metode *Clustering***" merupakan salah satu syarat memperoleh gelar Sarjana Komputer pada program studi Ilmu Komputer Jurusan Matematika Fakultas MIPA Universitas Brawijaya.

Tidak dapat dipungkiri bahwa tidak mungkin penulis dapat menyelesaikan skripsi ini tanpa bantuan dan dukungan dari banyak pihak. Untuk itu, dengan ketulusan dan kerendahan hati penulis menyampaikan ucapan terima kasih yang sebesar-besarnya kepada :

1. Lailil Muflikhah, S.Kom., M.Sc, selaku dosen pembimbing utama yang telah meluangkan banyak waktu untuk memberikan pengarahan dan masukan bagi penulis terkait dengan fokus skripsi ini yaitu *Data mining*.
2. Djoko Pramono, ST, selaku pembimbing kedua yang telah banyak memberikan saran serta masukan kepada penulis.
3. Dr. Abdul Rouf Alghofari, M.Sc., selaku ketua Jurusan Matematika.
4. Kedua orang tua dan adik penulis yang tidak henti-hentinya memberikan dukungan berupa materi dan moril kepada penulis.
5. Rifki F.Z., S.T., M.Kom. yang telah memberikan bimbingan kepada penulis mengenai algoritma SSNN.
6. Segenap Bapak dan Ibu dosen yang telah mendidik dan mengajarkan ilmunya kepada penulis selama menempuh pendidikan di Program Studi Ilmu Komputer Jurusan Matematika Fakultas MIPA Universitas Brawijaya.
7. Festri Purwantaris yang selalu menemani penulis dalam suka maupun duka.
8. Rekan-rekan kerja *Student Computer Service* Universitas Brawijaya yang tak henti-hentinya memberikan semangat kepada penulis untuk segera menyelesaikan skripsi ini.

9. Rekan-rekan Ilmu Komputer B ’07 yang selalu memberikan semangat tanpa henti kepada penulis untuk segera menyelesaikan skripsi ini.
10. Rekan-rekan *gamer dota mania* yang senantiasa memberi semangat dan warna baru kepada penulis.
11. Adik-adik tingkat mulai 2008 hingga 2011 yang pernah menemani penulis dalam menempuh perkuliahan.
12. Segenap staf dan karyawan di Jurusan Matematika Fakultas MIPA Universitas Brawijaya yang telah banyak membantu penulis dalam pelaksanaan penyusunan skripsi ini.
13. Dan semua pihak yang telah terlibat baik secara langsung maupun tidak langsung yang tidak dapat penulis sebutkan satu per satu terima kasih atas semua bantuan yang telah diberikan.

Penulis berharap semoga laporan ini dapat memberikan manfaat bagi semua pembaca. Penulis dapat dihubungi melalui *e-mail* di khuluqfirmansyah@gmail.com

Malang, Juli 2012

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR SOURCE CODE


UNIVERSITAS BRAWIJAYA

BAB I
PENDAHULUAN

1.1 Latar Belakang

Seiring dengan perkembangan teknologi informasi, kebutuhan akan informasi terus meningkat. Dengan banyaknya data yang tersedia maka seorang akan kesulitan untuk mencari informasi yang dibutuhkan. Metode pencarian informasi dewasa ini banyak dikembangkan untuk membantu pencarian informasi dalam sekumpulan data. Salah satu metode pencarian informasi yang sering digunakan adalah metode *Clustering.*

Clustering yaitu suatu metode membagi data menjadi kelompok-kelompok atau *cluster-cluster* berdasarkan suatu kemiripan atribut-atribut di antara data tersebut. Data yang terletak dalam satu *cluster* harus memiliki kemiripan sedangkan yang tidak berada dalam satu *cluster*, tidak memiliki kemiripan (Budiarti, 2006).

Dalam *data mining*, Algoritma *Clustering* dibagi menjadi 2 metode yaitu, *Hierarchical Clustering* (*Clustering* dengan pendekatan Hirarki) dan *Partitional Clustering* (*Clustering* dengan pendekatan Partisi). *Partitional Clustering* mengelompokkan data dengan memilah-milah data yang dianalisis ke dalam *cluster-cluster* yang ada. *Hierarchical Clustering* mengelompokkan data dengan membuat suatu hirarki berupa dendogram (*tree*) dimana data yang mirip akan ditempatkan pada hirarki yang berdekatan dan yang tidak pada hirarki yang berjauhan. Salah satu contoh algoritma *clustering* dengan pendekatan hirarki adalah algoritma *Shared Nearest Neighbor* (SNN) (Budiarti, 2006).

Menurut wijayanto (2009), Algoritma *Shared Nearest Neighbor* (SNN) adalah algoritma *clustering* yang bekerja dengan menggunakan metode *density-based*. Metode *density-based* mendefinisikan sebuah *outlier* yang merupakan sekumpulan titik data dengan kepadatan yang sangat rendah. Sebuah *outlier* didefinisikan sebagai sebuah titik data pada suatu *dataset* dimana titik tersebut sangat berbeda dibandingkan dengan titik data lainnya pada *dataset* tersebut. Menurut Zainal (2008), kekurangan utama

algoritma SNN adalah dibutuhkannya sebuah nilai ambang batas untuk menentukan penggabungan atau pemisahan *cluster*. Bahkan dapat terjadi tidak adanya nilai ambang batas yang sesuai dengan beberapa *dataset*. Untuk mengatasi kekurangan tersebut, dikembangkan sebuah algoritma SNN berbasis kepadatan (*Shrinking based Shared Nearest Neighbor*).

Algoritma Shrinking based Shared Nearest Neighbor (SSNN) menggabungkan proses SNN dengan proses pembentukan *cluster* pada DBSCAN. DBSCAN mendefinisikan *cluster* sebagai himpunan dari titik-titik kepadatan yang terkoneksi (*density-connected*). Semua objek yang tidak masuk ke dalam *cluster* manapun dianggap sebagai *noise* (Zainal , 2008).

Dalam melakukan *clustering* dengan algoritma *Shrinking based Shared Nearest Neighbor* dibutuhkan 3 parameter *input* yaitu nilai tetangga terdekat dari tiap titik yang diambil untuk proses perhitungan (k), batas nilai minimal pemutusan bobot ketetanggaan (CP) dan batas nilai penghentian iterasi (MP) (Zainal , 2008).

Pada penelitian Zainal (2008), digunakan persamaan jarak *euclidean distance* untuk menghitung nilai *dissimilarity*. Akurasi yang dihasilkan algoritma *Shrinking based Shared Nearest Neighbor* dengan menggunakan persamaan jarak *euclidean distance* untuk *dataset iris* adalah 0,727 dan *dataset wine* adalah 0,456. Menurut hasil penelitian Hamzah dkk menunjukkan bahwa hasil *clustering* dokumen terbaik adalah jika digunakan fungsi jarak *cosine distance* dengan rata-rata *f-measure* untuk seluruh koleksi adalah 0,9313 dan yang terburuk adalah jika digunakan fungsi jarak *euclidean distance* dengan rata-rata *f-measure* adalah 0,4668. Hasil penelitian Kurniawan dkk juga menunjukan bahwa *cluster K-means* dengan menggunakan *cosine similarity* memberikan hasil yang lebih baik bila dibandingkan dengan pengukuran *euclidean distance*.

Berdasarkan latar belakang tersebut maka pada skripsi ini akan dilakukan kembali proses-proses yang ada pada jurnal Zainal tersebut dan mengganti persamaan jarak *euclidean distance* dengan persamaan jarak *cosine similarity*. Oleh karena itu penulis mengambil judul "**PENERAPAN *COSINE SIMILARITY* PADA**

ALGORITMA *SHRINKING BASED SHARED NEAREST NEIGHBOR* (SSNN) PADA METODE *CLUSTERING*".

1.2 Rumusan Masalah

Rumusan masalah dalam skripsi ini adalah :

1. Bagaimana menerapkan *cosine similarity* pada algoritma *Shrinking based Shared Nearest Neighbor*.
2. Bagaimana tingkat akurasi algoritma *Shrinking based Shared Nearest Neighbor* dengan menggunakan *cosine similarity* dibandingkan dengan algoritma *Shrinking based Shared Nearest Neighbor* dengan menggunakan *euclidean distance*.

1.3 Batasan Masalah

Batasan masalah dari skripsi ini adalah :

1. Data yang digunakan dalam skripsi ini adalah *iris dataset* dan *wine dataset* yang bersumber dari *website UCI machine learning repository*.
2. Penentuan nilai ambang batas kedekatan (MP) berkisar antara 1% sampai 100%.

1.4 Tujuan

Tujuan dari skripsi ini adalah :

1. Membuat sistem untuk menerapkan *cosine similarity* pada algoritma *Shrinking based Shared Nearest Neighbor*.
2. Mengetahui tingkat akurasi *cosine similarity* pada algoritma *Shrinking based Shared Nearest Neighbor*.

1.5 Manfaat

Manfaat yang bisa didapatkan dari penelitian ini adalah sebagai referensi bahwa *cosine similarity* dapat digunakan pada algoritma *Shrinking based Shared Nearest Neighbor* untuk *clustering dataset* sebagai rumus untuk menghitung jarak antar data.

1.6 Metodologi Pemecahan Masalah

Untuk mencapai tujuan yang durumuskan sebelumnya, maka metodologi yang digunakan dalam penulisan skripsi ini adalah :

1. Studi literatur, mempelajari teori-teori yang berhubungan dengan Algoritma *Shrinking based Shared Nearest Neighbor* (SSNN).
2. Pendefinisian dan analisis masalah, mendefinisikan dan menganalisis masalah untuk mencari solusi yang tepat.
3. Perancangan dan implementasi sistem, membuat perancangan sistem dan mengimplementasikan hasil rancangan tersebut yaitu membuat sistem untuk mengimplementasikan algoritma *Shrinking based Shared Nearest Neighbor* (SSNN).
4. Uji coba dan evaluasi hasil implementasi, menguji sistem dan mengevaluasi hasil dari implementasi tersebut apakah sudah sesuai dengan tujuan yang dirumuskan sebelumnya.

1.7 Sistematika Penulisan

Laporan Tugas Akhir ini disusun dengan sistematika penulisan sebagai berikut :

BAB I PENDAHULUAN

Pada bab ini membahas mengenai latar belakang, rumusan masalah, batasan masalah, tujuan skripsi, manfaat skripsi, metodologi pemecahan masalah dan sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Pada bab ini membahas mengenai tinjauan pustaka yang digunakan dalam skripsi ini yaitu mengenai *Shrinking based Shared Nearest Neighbor* dan referensi-referensi lain yang mendukung skripsi ini baik dari buku maupun dari internet.

BAB III METODOLOGI DAN PERANCANGAN

Pada bab ini membahas mengenai metodologi dan perancangan yang digunakan dalam penyelesaian permasalahan *clustering* menggunakan algoritma *Shrinking based Shared Nearest Neighbor*.

BAB IV IMPLEMENTASI DAN PEMBAHASAN

Pada bab ini membahas mengenai implementasi metode, rancangan dan analisis sistem yang dikembangkan, yaitu apakah hasil *clustering* oleh sistem menghasilkan *cluster* yang baik.

BAB V PENUTUP

Pada bab ini berisi kesimpulan dari pembahasan dalam skripsi ini dan saran yang diharapkan bermanfaat untuk penelitian lebih lanjut.


UNIVERSITAS BRAWIJAYA

BAB II
TINJAUAN PUSTAKA

2.1 *Data mining*

Data mining didefinisikan sebagai suatu teknik yang digunakan untuk mengeksplorasi secara menyeluruh dan membawa ke permukaan relasi-relasi yang kompleks pada set data yang sangat besar. Set data yang dimaksud di sini adalah set data yang berbentuk tabulasi, seperti yang banyak diimplementasikan dalam teknologi manajemen basis data relasional. Akan tetapi, teknik-teknik *data mining* dapat juga disistemkan pada representasi data yang lain, seperti domain data spatial , berbasis text, dan multimedia (citra). *Data mining* dapat juga didefinisikan sebagai "pemodelan dan penemuan pola-pola yang tersembunyi dengan memanfaatkan data dalam volume yang besar" (Moertini, 2002).

Data mining menggunakan pendekatan *discovery-based* dimana pencocokan pola (*pattern-matching*) dan algoritma-algoritma yang lain digunakan untuk menentukan relasi-relasi kunci di dalam data yang diekplorasi. *Data mining* merupakan komponen baru pada arsitektur sistem pendukung keputusan (DSS) di perusahaan-perusahaan. Dalam *data mining* terdapat banyak teknik yang digunakan, Salah satu teknik yang populer yaitu teknik *clustering* (Moertini, 2002).

2.2 *Clustering*

Clustering adalah metode analisis data, yang sering dimasukkan sebagai salah satu metode *Data mining,* yang tujuannya adalah untuk mengelompokkan data dengan karakteristik yang sama ke suatu 'wilayah' yang sama dan data dengan karakteristik yang berbeda ke 'wilayah' yang lain (Jain dkk, 1999).

Ada beberapa pendekatan yang digunakan dalam mengembangkan metode *clustering*. Dua pendekatan utama adalah *clustering* dengan pendekatan partisi dan *clustering* dengan pendekatan hirarki. *Clustering* dengan pendekatan partisi atau sering disebut dengan *partition-based clustering* mengelompokkan data dengan memilah-milah data yang dianalisis ke dalam *cluster-cluster*

yang ada. *Clustering* dengan pendekatan hirarki atau sering disebut dengan *hierarchical clustering* mengelompokkan data dengan membuat suatu hirarki berupa dendogram dimana data yang mirip akan ditempatkan pada hirarki yang berdekatan dan yang tidak mirip ditempatkan pada hirarki yang berjauhan (Jain dkk, 1999).

2.2.1 *Clustering* Dengan Pendekatan Partisi

Clustering Dengan Pendekatan Partisi berusaha untuk mengelompokkan *Dataset* ke dalam K *cluster* berdasarkan kriteria-kriteria tertentu. *Centroid-based techniques* cocok untuk data dalam ruang matriks yang memungkinkan untuk dihitung *centroid* dari titik-titik yang diberikan. Pada *Medoid-based techniques* bekerja untuk data yang sama. Teknik ini berusaha untuk menemukan titik yang representatif untuk meminimumkan jarak titik-titik yang ada dari *medoid* terdekatnya (Karypis dkk, 1999).

2.2.2 *Clustering* dengan Pendekatan Hirarki

Clustering dengan Pendekatan Hirarki menghasilkan sejumlah *cluster* yang bersarang, dengan sebuah *cluster* di bagian atas dan sebuah *cluster* di bagian bawah. *Agglomerative hierarchical algorithm* dimulai dengan menjadikan semua titik data sebagai *cluster* yang terpisah. Masing-masing langkah akan menyatukan dua buah *cluster* yang paling banyak kemiripannya, sehingga tiap langkah akan berkurang setengahnya. Langkah ini dapat diulangi hingga jumlah *cluster* yang diinginkan didapat atau jarak antara dua *cluster* terdekat melewati jarak tertentu (Karypis dkk,1999).

2.3 Normalisasi

Perbedaan kualitas suatu hasil *clustering* bergantung pada metode yang dipakai untuk mengukur kesamaan (*similarity*) tersebut serta implementasinya. Dalam hal ini, diperlukan keseimbangan antara data yang akan di-*cluster*. Oleh karena itu, digunakan metode *Min-Max Normalization* untuk mengatasi hal tersebut. Persamaan selengkapnya dari metode ini dapat dilihat pada persamaan 2.1 (Shalabi dkk, 2006).

$$V' = \frac{V - \min_x}{\max_x - \min_x} \quad (2.1)$$

Dimana,

V' = Nilai dari data hasil *Min-Max Normalization.*
V = Nilai dari data yang akan dinormalisasi.
Min_x = Nilai miminum dari suatu *field* data yang sama.
Max_x = Nilai maksimum dari suatu *field* data yang sama.

2.4 *Similarity Measure* dan *dissimilarity measures*

Dalam melakukan *clustering* dan teknik *data mining* yang lain, *similarity measure* dan *dissimilarity measures* merupakan bagian penting yang harus diperhatikan. Menurut Hamzah dkk, Kesamaan antara data x_i dengan data y_i dapat diukur dengan fungsi *similarity* (mengukur kesamaan) atau fungsi *disimilarity* (mengukur ketidaksamaan). Beberapa fungsi *similarity* antara lain adalah *Dice, Jaccard* dan *Cosine*. Sedangkan fungsi *dissimilarity* antara lain adalah *Euclidean* dan *Manhattan* (Yampolskiy dkk, 2006).

Misalkan *dissimilarity* antara obyek i dan obyek j dinyatakan dengan d_{ij} dan *similarity* dinyatakan dengan s_{ij}. Hubungan antara *relationship dissimilarity* dengan *similarity* dinyatakan dengan

$$s_{ij} = 1 - d_{ij} \quad (2.2)$$

Dengan *similarity* terbatas pada 0 dan 1. Jika *similarity* bernilai satu (benar - benar sama), maka *dissimilarity* nol, dan jika *similarity* bernilai nol (sangat berbeda), *dissimilarity* bernilai satu (Andayani, 2007).

Berikut ini formula fungsi-fungsi tersebut (Hamzah dkk, 2008):

1. *Dice*, mengukur jarak dua buah objek dengan persamaan sebagai berikut :

$$d(x, y) = \frac{2\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i + \sum_{i=1}^{n} y_i} \quad (2.3)$$

Dimana,

$d(x, y)$ = jarak antar titik x dan titik y.

x_i = nilai titik x pada tabel ke-i

y_i = nilai titik y pada tabel ke-i

2. *Jaccard*, mengukur jarak dua buah objek dengan persamaan sebagai berikut :

$$d(x, y) = \frac{2\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i + \sum_{i=1}^{n} y_i - \sum_{i=1}^{n} x_i y_i} \quad (2.4)$$

Dimana,

$d(x, y)$ = jarak antar titik x dan titik y.

x_i = nilai titik x pada tabel ke-i

y_i = nilai titik y pada tabel ke-i

3. *Cosine*, mengukur jarak dua buah objek dengan persamaan sebagai berikut :

$$d(x, y) = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} (x_i)^2 \sum_{i=1}^{n} (y_i)^2}} \quad (2.5)$$

Dimana,

$d(x, y)$ = jarak antar titik x dan titik y.

x_i = nilai titik x pada tabel ke-i

y_i = nilai titik y pada tabel ke-i

4. *Euclidean Distance*, mengukur jarak dua buah objek dengan persamaan sebagai berikut :

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \quad (2.6)$$

Dimana,

$d(x, y)$ = jarak antar titik x dan titik y.

x_i = nilai titik x pada tabel ke-i

y_i = nilai titik y pada tabel ke-i

5. *Manhattan*, mengukur jarak dua buah objek dengan persamaan sebagai berikut :

$$d(x, y) = \sum_{i=1}^{n} | x_i - y_i | \qquad (2.7)$$

Dimana,

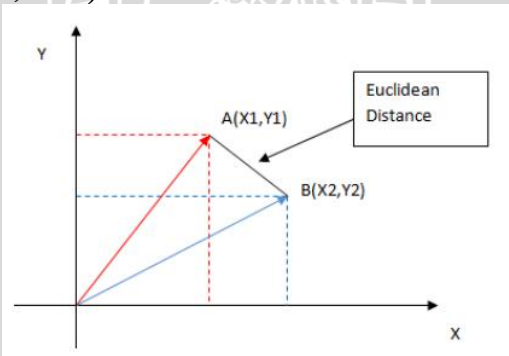
$d(x, y)$ = jarak antar titik x dan titik y.

x_i = nilai titik x pada tabel ke-i

y_i = nilai titik y pada tabel ke-i

2.5 *Euclidean Distance*

Dalam matematika, euclidean distance atau adalah jarak antara dua titik yang dapat diukur dan dihasilkan oleh formula pytagoras. Euclidean vector atau sering hanya disebut dengan vector adalah obyek geometri yang memiliki panjang (magnitude) dan arah (direction). Sedangkan ruang vektor adalah sebuah structur matematika yang dibentuk oleh sekumpulan vektor. Vektor-vektor tersebut dapat ditambahkan, dikalikan dengan bilangan real dan lain-lain (Rodiyansyah, 2010).



Gambar 2.1 Vektor *Space* 2 Dimensi

Gambar 2.1 merupakan contoh dari ruang vektor, pada ruang vektor tersebut terdapat 2 vektor yaitu vektor A dan vektor B. Untuk menghitung jarak antara vektor A dan vektor B digunakan persamaan euclidean distance. Berikut merupakan penyelesaian dalam menghitung jarak antara vektor A dan vektor B. Panjang vektor A dan B dapat didefinisikan sebagai berikut :

$$\left\|\overline{A}\right\| = \sqrt{X1^2 + Y1^2} \quad (2.8)$$

$$\left\|\overline{B}\right\| = \sqrt{X2^2 + Y2^2} \quad (2.9)$$

Dengan demikian, untuk menghitung jarak antara kedua vektor tersebut menggunakan persamaan sebagai berikut :

$$d(\overline{A},\overline{B}) = \sqrt{(X1 - X2)^2 + (Y1 - Y2)^2} \quad (2.10)$$

Sedangkan untuk n dimensi ruang vektor, jarak euclidean distance ditentukan dengan menggunakan persamaan sebagai berikut :

$$d(\overline{u},\overline{v}) = \sqrt{(u_1 - v_1)^2 + (u_2 - v_2)^2 + \ldots + (u_n - v_n)^2} \quad (2.11)$$

Nilai *euclidean distance* merupakan nilai kemiripan citra digital. Semakin dekat (mendekati nilai 0) semakin mirip citra digital tersebut (Rodiyansyah, 2010).

2.6 *Cosine Similarity*

Cosine similarity merupakan metode perhitungan jarak antara vektor d1 dan d2 yang menghasilkan sudut *cosine* x diantara kedua vektor tersebut (*Handout Lecture Standford University*.2007 : 12). Metode ini digunakan untuk menghitung nilai kedekatan vektor d dengan *centroid* atau pusat massa pada masing-masing *cluster*. Berikut rumus metode perhitungan *cosine similarity* (Gracia, E. 2006 : 1):

$$Sim\ (\mathrm{A,B}) = cosine\ \Theta = \frac{A \bullet B}{|A||B|} \quad (2.12)$$

Dengan A•B merupakan *dot product*. *Dot product* merupakan nilai yang mengekspresikan sudut antara dua vektor (*Dunlop*, Robert. 2005 : 1). *Dot product* merupakan skalar nilai hasil dari operasi dua

vektor yang memiliki jumlah komponen yang sama. Jika vektor A dan B memiliki komponen sebanyak n, maka *dot product* dapat dihitung dengan rumus sebagai berikut (*Dunlop*, Robert. 2005 : 1):

$$A \bullet B = A_1B_1 + \ldots.. + A_nB_n \qquad (2.13)$$

Dot product dapat dihitung dengan menjumlahkan *product* dari masing-masing komponen pada kedua vektor. Jika vektor A dan B merupakan vektor 3D, maka perhitungan *dot product* adalah sebagai berikut (*Dunlop*, Robert. 2005 : 1):

$$A \bullet B = A_x * B_x + A_y * B_y + A_z * B_z \qquad (2.14)$$

Sedangkan |A| merupakan panjang vektor. Panjang vektor dapat dihitung dengan rumus seperti berikut:

$$|A| = \sqrt{(x_1^2 + x_2^2 + x_3^2)} \qquad (2.15)$$

Sehingga $|A||B| = \sqrt{(x_1^2 + x_2^2 + x_3^2)} * \sqrt{(y_1^2 + y_2^2 + y_3^2)}$ (2.16)

(Widyawati, 2007)

2.7 Algoritma *Data Shrinking*

Algoritma *data Shrinking* merupakan sebuah teknik pra-proses data yang mengoptimalkan struktur data dengan menggunakan hukum grafitasi. Algoritma ini terdiri dari tiga langkah utama, yaitu penyusutan data, deteksi *cluster*, dan seleksi *cluster*. Dalam tahap penyusutan data, titik-titik data akan digerakkan searah dengan *gradien* kepadatan mensimulasikan hukum grafitasi sehingga diperoleh *cluster* yang padat dan terpisah dengan baik (Zainal , 2008).

Cluster akan dideteksi dengan menemukan komponen *cell* padat yang saling berhubungan. Kedua tahapan tersebut dilakukan dalam *clustering* berbasis *grid*. Kemudian dalam langkah seleksi *cluster*, setiap *cluster* yang telah dideteksi akan dievaluasi dalam skala yang berbeda menggunakan pengukuran evaluasi *cluster* dan *cluster* terbaik akan dipilih sebagai hasil akhir (Zainal , 2008).

Dataset dalam proses algoritma *data Shrinking* akan dibagi menjadi ruang-ruang berdasarkan ukuran skala yang telah ditentukan sebelumnya. Pergerakan titik dalam masing-masing ruang dalam *dataset* akan diperlakukan sebagai satu badan yang bergerak sebagai satu kesatuan ke arah pusat data dari ruang tetangganya. Dengan

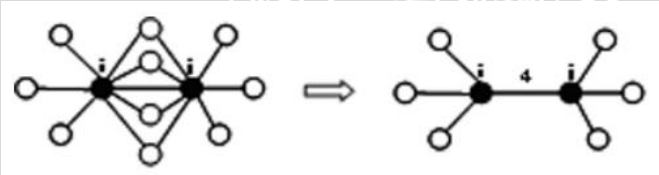
demikian, semua titik dalam berpartisipasi dalam gerakan yang sama (Zainal , 2008).

Langkah kedua dalam algoritma *data Shrinking* adalah deteksi *cluster*. Karena proses penyusutan data menghasilkan *cluster* individual yang padat dan terpisah dengan baik, maka dapat digunakan berbagai algoritma deteksi *cluster*. Pada penelitian awal *data Shrinking*, digunakan metode deteksi *cluster* berbasis skala. Untuk setiap ruang padat, ruang tetangganya dihubungkan dan dibentuk sebuah *graph* ketetanggaan. Kemudian dihitung nilai kerapatan sebelum dilakukan penyusutan data dari *cluster* yang dideteksi pada semua skala. *Cluster* yang memiliki kerapatan lebih dari nilai ambang batas tertentu akan dijadikan sebagai *cluster* hasil (Zainal , 2008).

2.8 *Shared Nearest Neighbor* (SNN)

Dalam beberapa kasus, teknik *clustering* yang bergantung pada pendekatan standar ke arah kesamaan dan kepadatan tidak menghasilkan hasil *cluster* yang diinginkan. Pendekatan SNN yang dikembangkan oleh Jarvis dan Patrick merupakan pendekatan tidak langsung terdapat kesamaan berdasarkan prinsip berikut:

Jika dua titik memiliki kesamaan terhadap titik yang sama banyak, kedua titik tersebut memiliki kesamaan satu sama lain, bahkan jika pengukuran kesamaan tidak menunjukkan kesamaan tersebut.



Gambar 2.2 Komputasi kesamaan dalam SNN

Ide kunci algoritma ini adalah mengambil jumlah dari titik-titik data untuk menentukan pengukuran kesamaan. Kesamaan dalam algoritma SNN didasarkan jumlah tetangga yang dimiliki secara bersama-sama selama kedua objek terdapat dalam daftar tetangga

terdekat masing-masing seperti diperlihatkan pada Gambar 2.1 (Zainal , 2008).

Algoritma ini bekerja baik untuk data berdimensi besar dan secara khusus bekerja baik dalam menemukan *cluster* padat. Akan tetapi, *clustering* SNN mendefinisikan *cluster* sebagai komponen-komponen *graph* ketetanggaan yang saling berhubungan sehingga pembagian *cluster* bergantung pada sebuah hubungan antara objek (Zainal , 2008).

Untuk mengatasi permasalahan dari algoritma SNN dasar tersebut, maka diciptakan algoritma SNN berbasis kepadatan. Algoritma SNN ini mengaplikasikan titik representatif DBSCAN dalam proses untuk memperoleh *cluster*. Dengan demikian, efek desau dapat dikurangi dengan menggunakan titik reperesentatif. Algoritma SNN ini terdiri dari langkah:

1. Hitung nilai kesamaan dari data set.
2. Bentuk daftar k tetangga terdekat masing-masing titik data untuk k data.
3. Bentuk *graph* ketetanggaan dari hasil daftar k tetangga terdekat.
4. Temukan kepadatan untuk setiap data.
5. Temukan titik-titik representatif.
6. Bentuk *cluster* dari titik-titik representatif tersebut.

2.9 *Shrinking based Shared Nearest Neighbor* (SSNN)

2.9.1 Pengertian SSNN

Algoritma SSNN sebenarnya merupakan perbaikan dari algoritma SNN. Akurasi *cluster* yang diperoleh algoritma SNN dapat diperbaiki dengan memperbesar kepadatan *graph* ketetanggaan yang dibentuk dalam algoritma SNN. Salah satu langkah untuk melakukan penguatan kepadatan tersebut adalah dengan menggunakan konsep pergerakan data ke arah pusat *cluster* dari algoritma *data Shrinking*. Karena, algoritma perbaikan ini akan membuat titik-titik data seakan-akan menyusut ke arah pusat *cluster*, maka algoritma ini diberi nama algoritma SNN berbasis data *Shrinking* atau *Shrinking based Shared Nearest Neighbor* (SSNN) (Zainal , 2008).

Apabila titik-titik data dalam *graph* ketetanggaan digerakkan ke arah pusat *cluster*, bobot hubungan ketetanggaan untuk titik-titik

data dalam *cluster* yang sama akan menjadi semakin besar dan bobot hubungan ketetanggaan untuk titik-titik data dalam *cluster* yang berbeda akan menjadi semakin kecil (Zainal , 2008).

2.9.2 Algoritma SSNN

Menurut Zainal, langkah-langkah algoritma SSNN, yaitu :

1. Hitung nilai *dissimilarity* dari *dataset*.
2. Bentuk daftar k tetangga terdekat masing-masing titik data.
3. Masukkan nilai ambang batas kedekatan (MP).
4. Masukkan nilai ambang batas ketetanggaan (CP) dan putuskan bobot hubungan ketetanggaan yang lebih besar dari nilai ambang batas ketetanggaan (CP).
5. Ulangi langkah 4 hingga nilai prosentase perubahan bobot lebih kecil dari nilai ambang batas kedekatan (MP). Berikut merupakan persamaan untuk menghitung prosentase perubahan bobot :

$$p_b = \frac{a-b}{b} \quad (2.17)$$

Dimana,

P_b = prosentase perubahan bobot.

a = besar bobot ketetanggaan yang dihilangkan pada iterasi yang sedang dijalankan.

b = besar bobot ketetanggaan yang dihilangkan pada iterasi yang sebelumnya.

6. Bentuk *cluster* dari komponen *graph* ketetanggaan yang masih memiliki bobot hubungan ketetanggaan
7. Hitung *weight* untuk semua data dengan rumus

$$w = \frac{\sum_{i=1}^{n} x_i}{n} \quad (2.18)$$

Dimana,

w = *weight*

x = nilai atribut ke i

n = total atribut

8. Hitung *Weight* tiap *cluster* dan cari jarak antar *cluster*. Kemudian gabungkan *cluster* yang memiliki jarak terkecil menjadi 1 *cluster*.
9. Hitung akurasi.

2.11 Akurasi

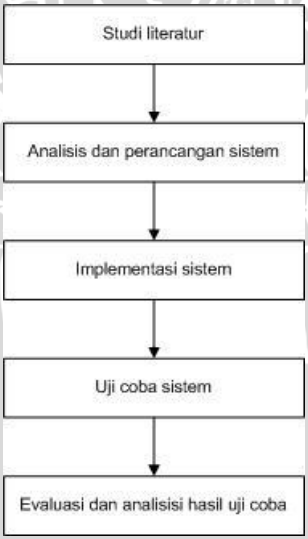
Akurasi adalah derajat kedekatan pengukuranterhadap nilai sebenarnya. Akurasi mencakup tidak hanya kesalahan acak, tetapi juga bias yangdisebabkan oleh kesalahansistematik yang tidak terkoreksi. Jika tidak ada bias kesalahan sistematik maka standar deviasi dapat dipakai untuk menyatakan akurasi (Karlita, 2011). Berikut merupakan rumus untuk menghitung nilai akurasi dari clustering dataset :

$$r = \frac{\sum_{i=1}^{k} a_i}{n} \quad (2.19)$$

BAB III
METODOLOGI DAN PERANCANGAN

Pada bab ini akan dibahas metode, rancangan yang digunakan dan langkah-langkah yang dilakukan dalam penelitian tentang penerapan *cosine similarity* pada algoritma *Shrinking based Shared Nearest Neighbor* (SSNN) pada metode *clustering.* Langkah-langkah yang dilakukan dalam penelitian ini meliputi:

1. Melakukan studi literatur tentang *Shrinking based Shared Nearest Neighbor* dan *cosine similarity.*
2. Menganalisis dan merancang sistem yang akan digunakan untuk *clustering*.
3. Membuat sistem berdasarkan analisis dan perancangan yang telah dilakukan.
4. Memasukkan data *training* ke dalam sistem yang telah dibuat kemudian memasukkan parameter yang dibutuhkan untuk melakukan uji coba pada sistem yang telah dibuat.
5. Melakukan evaluasi hasil uji coba terhadap sistem yang telah dibuat.



Gambar 3.1 Diagram Sistem

3.1 Perancangan Sistem

Sistem yang dibuat dalam penelitian ini merupakan sistem yang digunakan untuk *clustering dataset. Dataset* yang dipakai adalah *iris dataset* dan *wine dataset* yang diambil dari *website UCI machine learning repository*.

Langkah pertama algoritma *Shrinking based Shared Nearest Neighbor* adalah menghitung nilai *dissimilarity.* Sebelum menghitung nilai *dissimilarity, dataset wine* dan *dataset iris* dinormalisasi terlebih dahulu dengan algoritma *min-max normalization.*

Setelah data dinormalisasi, kemudian dihitung nilai *similarity iris dataset* dan *wine dataset* dengan menggunakan persamaan *cosine similarity* kemudian nilai *similarity* tersebut dirubah menjadi *dissimilarity.* Dari nilai *dissimilarity* tersebut didapatkan tabel jarak terdekat. Tabel jarak terdekat merupakan tabel yang berisi jarak antar data untuk semua data. Kemudian tabel jarak terdekat diurutkan secara *ascending* berdasarkan nilai *dissimilarity*. Untuk parameternya, *user* memasukan nilai k tetangga terdekat, nilai ambang batas kedekatan (MP) dan nilai ambang batas ketetanggaan (CP).

Dari tabel jarak terdekat yang telah diurutkan secara *ascending*, diambil k baris tetangga terdekat untuk tiap data. Kemudian dari tabel jarak terdekat yang nilai *dissimilarity* nya lebih besar dari nilai CP akan diputuskan hubungan ketetanggaannya. Dari pemutusan tersebut didapatkan tabel jarak terdekat yang memiliki nilai *dissimilarity* yang lebih kecil dari nilai CP. Pemutusan hubungan dilakukan dalam beberapa iterasi.

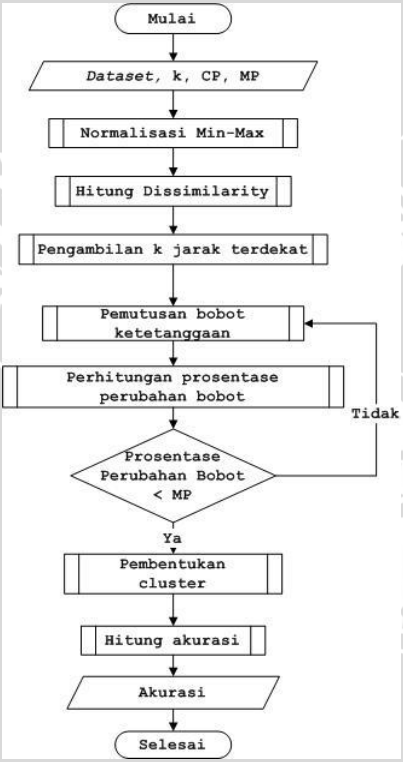
Untuk tiap iterasinya Nilai CP akan terus diturunkan sebesar 0,01. Dari tiap iterasi dihitung prosentase perubahan bobotnya. Iterasi akan berhenti pada saat nilai prosentase perubahan bobot lebih kecil dari nilai MP. Setelah iterasi berhenti maka akan dilakukan proses pembentukan *cluster* untuk tahap pertama. Kemudian dilakukan perhitungan *weight* untuk semua data. Dari *weigth* tersebut, dihitung rata-rata *weight* untuk setiap *cluster*. Kemudian dicari jarak antar *cluster*, *cluster* yang memiliki jarak terkecil akan digabung menjadi 1 *cluster*. Proses penggabungan *cluster* akan terus

dilakukan sampai jumlah *cluster* akhir yang terbentuk sama dengan jumlah *cluster dataset* awal.

Setelah proses pembentukan *cluster* selesai, maka langkah terakhir adalah perhitungan akurasi. Sebelum dilakukan perhitungan akurasi, dilakukan terlebih dahulu proses *labeling* sehingga dapat diketahui anggota *dataset* masuk ke *cluster* mana saja. Setelah proses *labeling* selesai dilakukan maka akurasi dihitung untuk setiap kemungkinan *labeling* yang terbentuk. Hasil akhir akurasi adalah nilai akurasi tertinggi dari proses pemberian *label* yang terbentuk.

.

3.2 Perancangan Proses

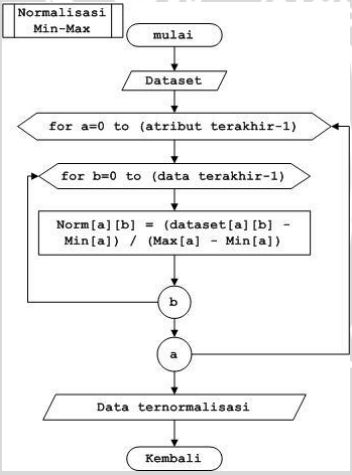


Gambar 3.2 *Flowchart clustering* dengan algoritma *Shrinking based Shared Nearest Neighbor*

Dari gambar 3.2, penjabaran dari langkah – langkah proses *clustering* dengan menggunakan algoritma *Shrinking based Shared Nearest Neighbor* adalah sebagai berikut :

1. Memasukan *dataset* ke dalam sistem dan parameternya yaitu k, CP dan MP.
2. Menghitung nilai normalisasi *min-max.*
3. Menghitung nilai *dissimilarity.*
4. Memasukan k baris jarak terdekat ke tabel jarak terdekat.
5. Memutuskan bobot ketetanggaan yang nilainya kurang dari CP.
6. Menghitung prosentase perubahan bobot untuk tiap iterasi.
7. Jika prosentase perubahan bobot lebih besar dari nilai MP maka kembali ke proses pemutusan bobot ketetanggaan.
8. Jika prosentase perubahan bobot lebih kecil dari nilai MP maka masuk ke proses selanjutnya yaitu proses pembentukan *cluster.*
9. Membentuk *cluster* dari tabel jarak terdekat.
10. Menghitung akurasi dari *cluster* yang telah dibentuk.
11. Hasil *akurasi* dari *clustering* algoritma *Shrinking based Shared Nearest Neighbor.*

3.2.1 Normalisasi *Min-Max*



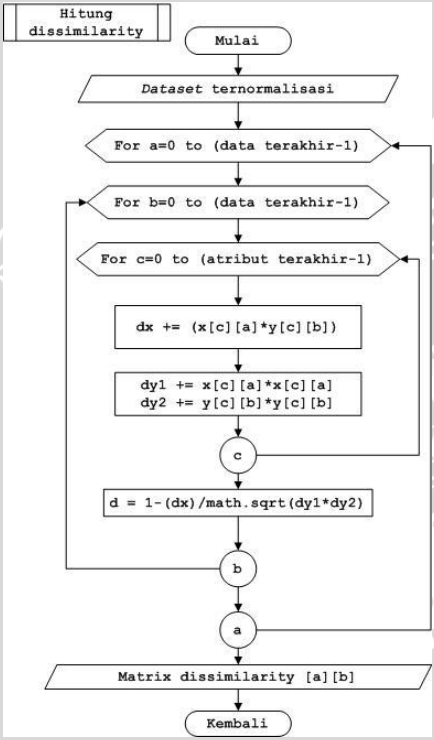
Gambar 3.3 *Flowchart Min-Max Normalization*

Dari gambar 3.3, penjabaran dari langkah – langkah proses *min-max normalization* adalah sebagai berikut :

1. Memasukan *dataset* ke dalam sistem.
2. *Looping* atribut.
3. *Looping* anggota *dataset.*
4. Menghitung nilai normalisasi untuk setiap anggota *dataset.*
5. *Dataset* telah ternormalisasi.

3.2.2 Hitung nilai *dissimilarity*

3.2.2.1 Hitung nilai *dissimilarity* dengan menggunakan *cosine similarity*

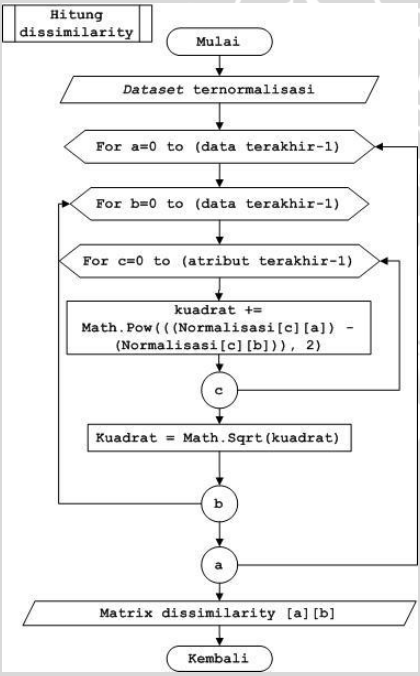


Gambar 3.4 *Flowchart* menghitung nilai *dissimilarity* dengan menggunakan *cosine similarity*

Dari gambar 3.4, penjabaran dari langkah – langkah proses menghitung nilai *dissimilarity* dengan menggunakan *cosine similarity* adalah sebagai berikut :

1. Memasukan *dataset* yang telah dinormalisasi ke dalam sistem.
2. *Looping* anggota *dataset* sebagai x.
3. *Looping* anggota *dataset* sebagai y.
4. Menghitung nilai *dissimilarity* menggunakan rumus *cosine* untuk semua anggota *dataset*.
5. Nilai *dissimilarity* untuk semua anggota *dataset* telah didapatkan.

3.2.2.2 Hitung nilai *dissimilarity* dengan menggunakan *euclidean distance*

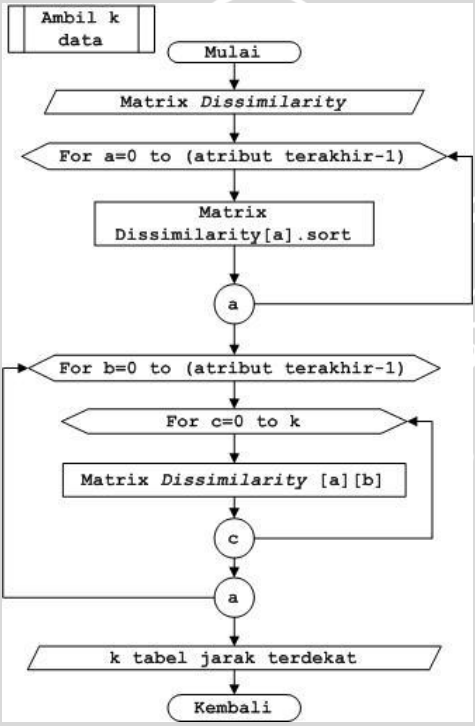


Gambar 3.5 *Flowchart* menghitung nilai *dissimilarity* dengan menggunakan *euclidean distance*

Dari gambar 3.5, penjabaran dari langkah – langkah proses menghitung nilai *dissimilarity* dengan menggunakan *euclidean distance* adalah sebagai berikut :

1. Memasukan *dataset* yang telah dinormalisasi ke dalam sistem.
2. *Looping* anggota *dataset* sebagai x.
3. *Looping* anggota *dataset* sebagai y.
4. Menghitung nilai *dissimilarity* menggunakan rumus *euclidean* untuk semua anggota *dataset*.
5. Nilai *dissimilarity* untuk semua anggota *dataset* telah didapatkan.

3.2.3 Pengambilan nilai k

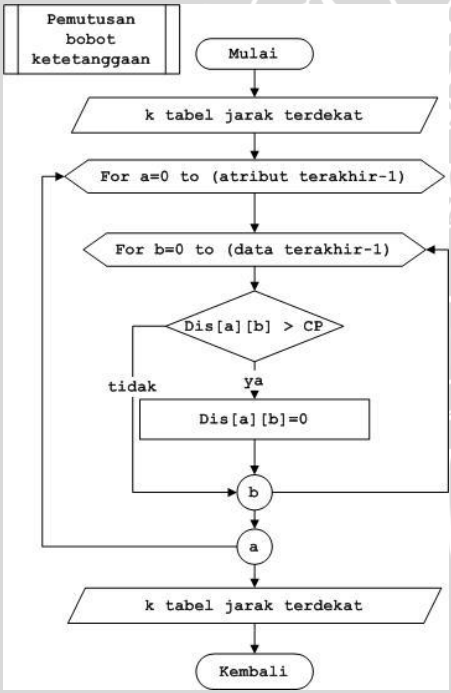


Gambar 3.6 *Flowchart* pengambilan nilai k

Dari gambar 3.6, penjabaran dari langkah – langkah proses pengambilan nilai k adalah sebagai berikut :

1. *Matrix dissimilarity* yang telah terbentuk dimasukan ke dalam sistem.
2. *Looping* atribut *matrix dissimilarity.*
3. Urutkan anggota *matrix dissimilarity* untuk setiap atribut.
4. *Looping* atribut *matrix dissimilarity.*
5. *Looping* anggota *matrix dissimilarity* sebanyak k baris.
6. Memasukan nilai *dissimilarity* sebanyak k baris ke dalam tabel jarak terdekat.
7. Tabel jarak terdekat telah terbentuk.

3.2.4 Pemutusan bobot ketetanggaan

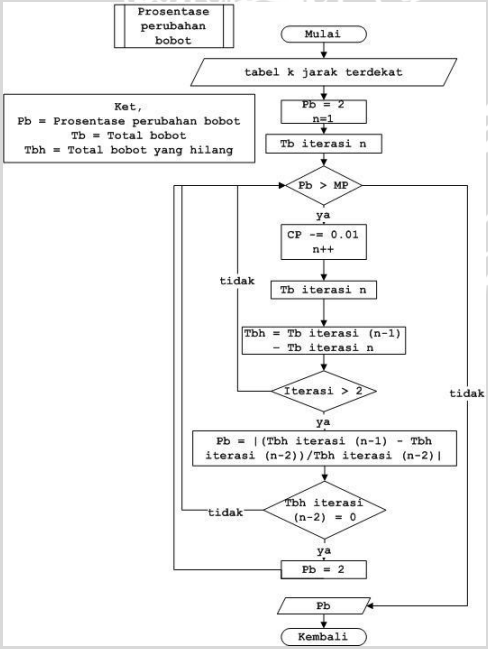


Gambar 3.7 *Flowchart* pemutusan bobot ketetanggaan

Dari gambar 3.7, penjabaran dari langkah – langkah proses pemutusan bobot ketetanggaan adalah sebagai berikut :

1. Memasukan k tabel jarak terdekat yang telah terbentuk ke dalam sistem.
2. *Looping* atribut k tabel jarak terdekat.
3. *Looping* anggota k tabel jarak terdekat.
4. Jika nilai *dissimilarity* yang dibaca lebih besar dari CP maka nilai *dissimilarity* tersebut di-set dengan nilai 0.
5. Jika nilai *dissimilarity* yang dibaca lebih kecil sama dengan CP maka nilai *dissimilarity* tersebut dimasukan ke dalam k tabel jarak terdekat.
6. k tabel jarak terdekat yang baru telah terbentuk.

3.2.5 Perhitungan nilai prosentase perubahan bobot

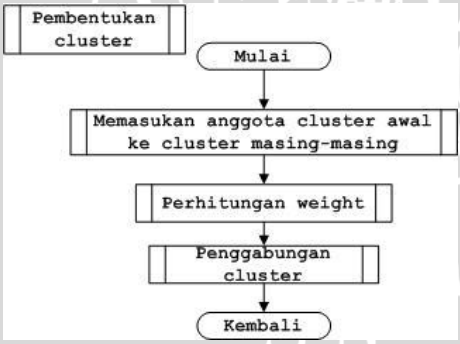


Gambar 3.8 *Flowchart* perhitungan nilai prosentase perubahan bobot

Dari gambar 3.8, penjabaran dari langkah – langkah proses perhitungan nilai prosentase perubahan bobot adalah sebagai berikut :

1. Memasukan k tabel jarak terdekat ke dalam sistem.
2. Menentukan nilai awal untuk prosentase perubahan bobot = 2 dan n=1.
3. Menghitung total bobot pada iterasi pertama.
4. Jika nilai prosentase perubahan bobot lebih besar dari nilai MP maka dilakukan *looping*.
5. Nilai CP terus diturunkan untuk setiap iterasinya sebesar 0,01 dan nilai n dinaikan sebesar 1.
6. Hitung total bobot pada iterasi ke-n.
7. Menghitug total bobot ketetanggaan yang hilang untuk setiap iterasi.
8. Jika iterasi lebih dari 2 maka menuju ke proses perhitungan prosentase perubahan bobot.
9. Menghitung prosentase perubahan bobot untuk setiap iterasi.
10. Mendapatkan prosentase perubahan bobot untuk semua iterasi.

3.2.6 Proses pembentukan *cluster*

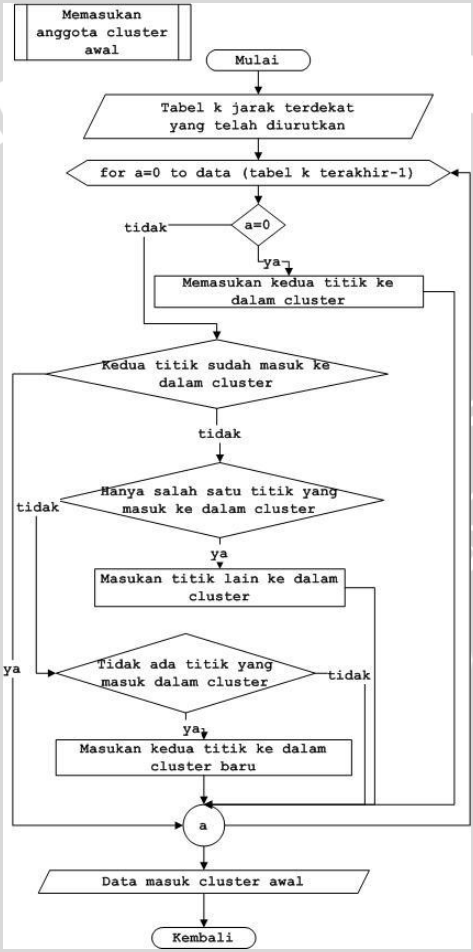


Gambar 3.9 *Flowchart* pembentukan *cluster*

Dari gambar 3.9, penjabaran dari langkah – langkah proses pembentukan *cluster* adalah sebagai berikut :

1. Proses memasukan anggota *cluster* awal ke ke dalam *cluster* masing-masing.
2. Perhitungan *weight.*
3. Penggabungan *cluster*.

3.2.7 Proses memasukan anggota *cluster* awal.

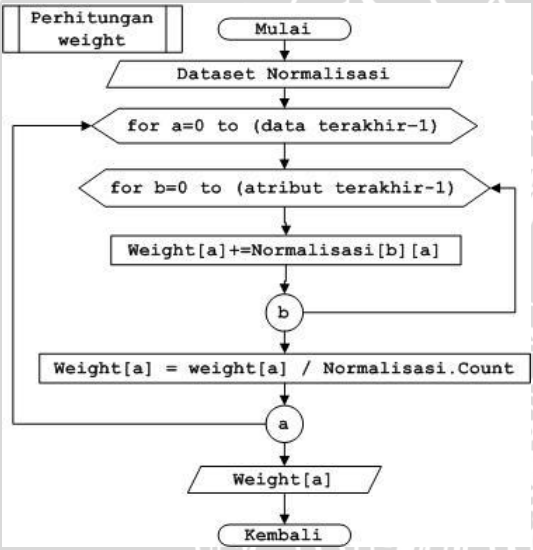


Gambar 3.10 *Flowchart* memasukan anggota *cluster* awal

Dari gambar 3.10, penjabaran dari langkah – langkah proses memasukan anggota *cluster* awal adalah sebagai berikut :

1. Memasukan tabel k jarak terdekat yang telah diurutkan.
2. *Looping* semua anggota k tabel jarak terdekat.
3. Jika yang sedang dibaca adalah data pertama maka kedua data tersebut langsung dimasukan ke dalam 1 *cluster*.
4. Jika kedua salah satu titik berada dalam suatu *cluster* maka titik pasangannya dimasukan ke dalam *cluster* yang sama.
5. Jika kedua titik belum masuk ke dalam *cluster* manapun maka masukan kedua titik kedalam *cluster* baru.
6. Semua titik telah masuk ke dalam *cluster* masing-masing.

3.2.8 Proses perhitungan *weight.*



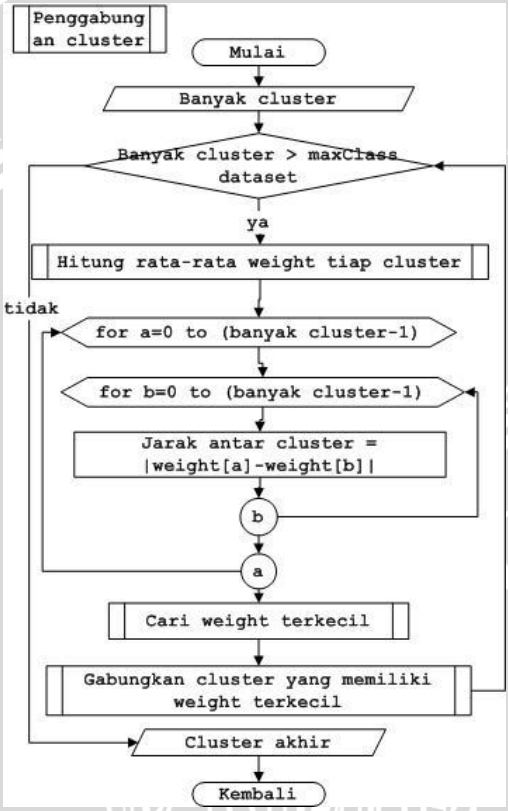
Gambar 3.11 *Flowchart* perhitungan *weight*

Dari gambar 3.11, penjabaran dari langkah – langkah proses memasukan anggota *cluster* awal adalah sebagai berikut :

1. Memasukan *dataset* yang telah dinormalisasi.
2. *Looping* anggota *dataset*.

3. *Looping* atribut *dataset*.
4. Menghitung nilai *weight* untuk setiap data.
5. Nilai *weight* untuk setiap data telah didapatkan.

3.2.9 Proses penggabungan cluster.



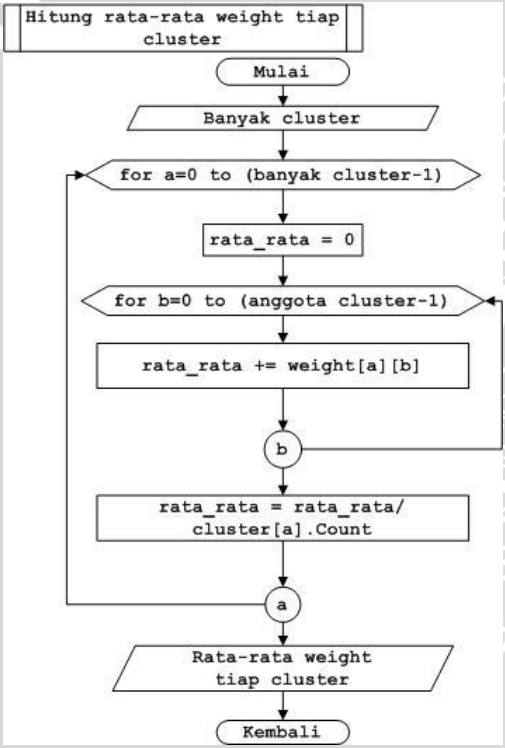
Gambar 3.12 *Flowchart* penggabungan *cluster*

Dari gambar 3.12, penjabaran dari langkah – langkah proses memasukan anggota *cluster* awal adalah sebagai berikut :

1. Memasukan banyaknya *cluster* yang telah terbentuk.
2. Jika banyaknya *cluster* lebih dari *maxclass dataset* maka akan dilakukan penggabungan *cluster*.

3. Menghitung rata-rata *weight* untuk setiap *cluster.*
4. Menghitung jarak antar *cluster.*
5. Mencari jarak terdekat antar *cluster*.
6. Menggabungkan *cluster* yang memiliki jarak terdekat ke dalam 1 *cluster*.
7. Proses penggabungan *cluster* dilakukan terus menerus sampai jumlah *cluster* yang dihasilkan sama dengan nilai *maxclass dataset.*
8. *Cluster* akhir telah terbentuk.

3.2.10 Proses hitung rata-rata weight tiap cluster.

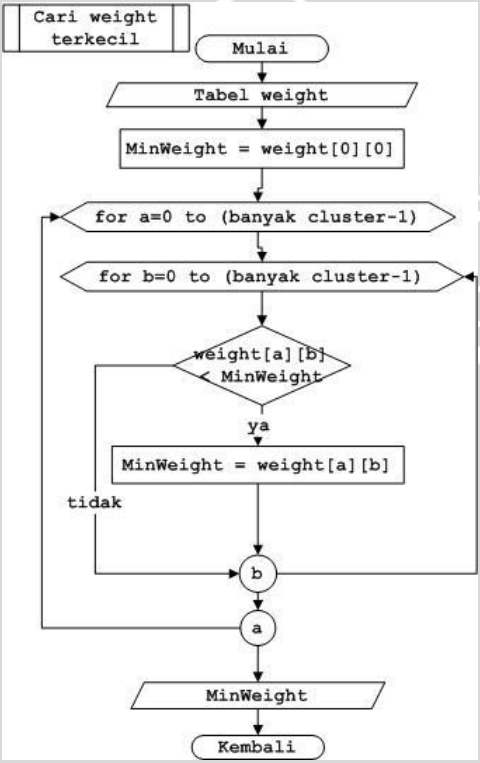


Gambar 3.13 *Flowchart* hitung rata-rata weight tiap *cluster*

Dari gambar 3.13, penjabaran dari langkah – langkah proses hitungrata-rata *weight* tiap *cluster* adalah sebagai berikut :

1. Memasukan banyaknya *cluster* yang telah terbentuk.
2. *Looping* banyaknya *cluster* yang telah terbentuk.
3. Set nilai rata-rata sama dengan 0.
4. *Looping* anggota *cluster*.
5. Tambahkan nilai rata-rata dengan nilai *weight* tiap anggota *cluster*.
6. Hitung rata-rata *weight* akhir tiap *cluster*.
7. Nilai rata-rata *weight* akhir tiap *cluster* telah didapatkan.

3.2.11 Proses cari *weight* terkecil.

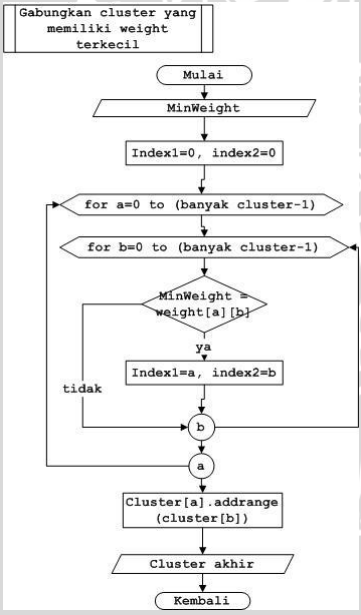


Gambar 3.14 *Flowchart* cari *weight* terkecil

Dari gambar 3.14, penjabaran dari langkah – langkah proses cari *weight* terkecil adalah sebagai berikut :

1. Masukan tabel *weight* ke dalam sistem.
2. Nilai *weight* minimal awal sama dengan nilai *weight* data pertama.
3. *Looping* banyaknya *cluster*.
4. *Looping* banyaknya *cluster*.
5. Jika nilai yang sedang dibaca kurang dari nilai *weight* sebelumnya maka nilai *weight* adalah nilai data yang sedang dibaca.
6. Nilai *weight* minimal telah didapatkan.

3.2.12 Proses penggabungan cluster yang memiliki jarak terkecil.

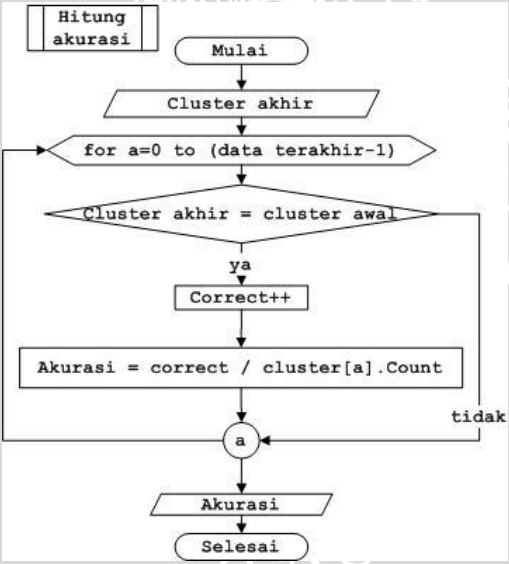


Gambar 3.15 *Flowchart* penggabungan *cluster* yang memiliki jarak terkecil

Dari gambar 3.15, penjabaran dari langkah – langkah proses penggabungan *cluster* yang memiliki jarak terkecil adalah sebagai berikut :

1. Nilai *weight* minimal dimasukan ke dalam sistem.
2. Set nilai index1 dan index2 dengan nilai 0.
3. *Looping* banyak *cluster*.
4. *Looping* banyak *cluster*
5. Jika nilai yang sedang dibaca sama dengan nilai *weigh* minimal maka index1 bernilai a dan index2 bernilai b.
6. Tambahkan *cluster* a dengan semua anggota *cluster* b.
7. *Cluster* akhir telah didapatkan.

3.2.13 Proses perhitungan akurasi



Gambar 3.16 *Flowchart* perhitungan akurasi

Dari gambar 3.16, penjabaran dari langkah – langkah proses perhitungan akurasi adalah sebagai berikut :

1. Memasukan hasil *cluster* akhir ke dalam sistem.

2. *Looping* semua data pada setiap *cluster.*
3. Jika anggota *cluster* akhir sama dengan anggota *cluster* awal maka nilai *correct* dinaikan sebesar 1.
4. Nilai *correct* untuk setiap *cluster* telah didapatkan.
5. Menghitung nilai akurasi untuk setiap *cluster.*
6. Nilai akurasi untuk setiap *cluster* telah didapatkan.

3.3 Contoh perhitungan

Misal diberikan sampel *dataset* yaitu *wine dataset.* Pada penelitian ini akan diambil 5 data untuk tiap kelas sehingga total data yang akan diambil sebagai sampel adalah 15 data. Sampel *wine dataset* akan ditunjukkan pada tabel 3.1.

Tabel 3.1 Tabel sampel *wine dataset*

Data	Att1	Att2	Att3	Att4	Att5	Att6	Att7	Att8	Att9	Att10	Att11	Att12	Att13	kelas
1	14.23	1.71	2.43	15.6	127	2.8	3.06	0.28	2.29	5.64	1.04	3.92	1065	1
2	13.2	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	0.4	1050	1
3	13.16	2.36	2.67	18.6	101	2.8	3.24	0.3	2.81	5.68	1.03	3.17	1185	1
4	14.37	1.95	2.5	16.8	113	3.85	3.49	0.24	2.18	7.8	0.86	0.45	1480	1
5	13.24	2.59	2.87	21	118	2.8	2.69	0.39	1.82	4.32	1.04	0.93	735	1
6	12.37	0.94	1.36	10.6	88	1.98	0.57	0.28	0.42	1.95	1.05	0.82	520	2
7	12.33	1.1	2.28	16	101	2.05	1.09	0.63	0.41	3.27	1.25	0.67	680	2
8	12.64	1.36	2.02	16.8	100	2.02	1.41	0.53	0.62	5.75	0.98	0.59	450	2
9	13.67	1.25	1.92	18	94	2.1	1.79	0.32	0.73	3.8	1.23	0.46	630	2
10	12.37	1.13	2.16	19	87	3.5	3.1	0.19	1.87	4.45	1.22	0.87	420	2
11	12.86	1.35	2.32	18	122	1.51	1.25	0.21	0.94	4.1	0.76	0.29	630	3
12	12.88	2.99	2.4	20	104	1.3	1.22	0.24	0.83	5.4	0.74	0.42	530	3
13	12.81	2.31	2.4	24	98	1.15	1.09	0.27	0.83	5.7	0.66	0.36	560	3
14	12.7	3.55	2.36	21.5	106	1.7	1.2	0.17	0.84	5	0.78	0.29	600	3
15	12.51	1.24	2.25	17.5	85	2	0.58	0.6	1.25	5.45	0.75	0.51	650	3

Proses awal dari algoritma *Shrinking based Shared Nearest Neighbor* adalah proses normalisasi data. Untuk menghitung nilai normalisasi digunakan persamaan 2.1. Misalkan dihitung nilai normalisasi data 1 pada atribut 1.

$$d_x = \frac{d - d_{\min}}{d_{\max} - d_{\min}}$$

$$d_x = \frac{14.23 - 12.33}{14.37 - 12.33} = 0.93137$$

Misalkan dihitung nilai normalisasi data 1 pada atribut *2*.

$$d_x = \frac{d - d_{\min}}{d_{\max} - d_{\min}}$$

$$d_x = \frac{1.71 - 0.94}{3.55 - 0.94} = 0.3$$

Dan seterusnya hingga semua data ternormalisasi.

Tabel 3.2 Tabel sampel wine *dataset* yang telah dinormalisasi

Data	Att1	Att2	Att3	Att4	Att5	Att6	Att7	Att8	Att9	Att10	Att11	Att12	Att13
1	0.93	0.30	0.71	0.37	1.00	0.61	0.85	0.24	0.78	0.63	0.64	1.00	0.61
2	0.43	0.32	0.52	0.04	0.36	0.56	0.75	0.20	0.36	0.42	0.66	0.03	0.59
3	0.41	0.54	0.87	0.60	0.38	0.61	0.91	0.28	1.00	0.64	0.63	0.79	0.72
4	1.00	0.39	0.75	0.46	0.67	1.00	1.00	0.15	0.74	1.00	0.34	0.04	1.00
5	0.45	0.63	1.00	0.78	0.79	0.61	0.73	0.48	0.59	0.41	0.64	0.18	0.30
6	0.02	0.00	0.00	0.00	0.07	0.31	0.00	0.24	0.00	0.00	0.66	0.15	0.09
7	0.00	0.06	0.61	0.40	0.38	0.33	0.18	1.00	0.00	0.23	1.00	0.10	0.25
8	0.15	0.16	0.44	0.46	0.36	0.32	0.29	0.78	0.09	0.65	0.54	0.08	0.03
9	0.66	0.12	0.37	0.55	0.21	0.35	0.42	0.33	0.13	0.32	0.97	0.05	0.20
10	0.02	0.07	0.53	0.63	0.05	0.87	0.87	0.04	0.61	0.43	0.95	0.16	0.00
11	0.26	0.16	0.64	0.55	0.88	0.13	0.23	0.09	0.22	0.37	0.17	0.00	0.20
12	0.27	0.79	0.69	0.70	0.45	0.06	0.22	0.15	0.18	0.59	0.14	0.04	0.10
13	0.24	0.52	0.69	1.00	0.31	0.00	0.18	0.22	0.18	0.64	0.00	0.02	0.13
14	0.18	1.00	0.66	0.81	0.50	0.20	0.22	0.00	0.18	0.52	0.20	0.00	0.17
15	0.09	0.11	0.59	0.51	0.00	0.31	0.00	0.93	0.35	0.60	0.15	0.06	0.22

Dari tabel sampel *wine dataset* tersebut akan dihitung nilai *similarity* untuk tiap data dengan menggunakan persamaan 2.5. Misalkan untuk menghitung nilai *similarity* data 1 dengan data 2 maka perhitungannya adalah sebagai berikut

$$d(1,2) = \frac{\sum_{i=1}^{n} x_1 y_2}{\sqrt{\sum_{i=1}^{n} (x_1)^2 \sum_{i=1}^{n} (y_2)^2}}$$

$$d(1,2) = \frac{(0.931 * 0.426) + (0.295 * 0.322) + ... \\ .+ (0.609 * 0.594)}{\sqrt{(0.931^2 + 0.295^2 + ...+ 0.609^2) * \\ (0.426^2 + 0.322^2 + ...0.594^2)}} = 0.863352$$

$$.....$$

$$d(14,15) = \frac{(0.181 * 0.088) + (1 * 0.115) + ... \\ ..+ (0.169 * 0.217)}{\sqrt{(0.181^2 + 0.1^2 + ...+ 0.169^2) \\ * (0.088^2 + 0.115^2 + ...0.217^2)}} = 0.4163$$

Dengan persamaan yang sama dihitung nilai *similarity* untuk semua data. Nilai *similarity* antar data dapat dilihat pada tabel 3.3

Tabel 3.3 Tabel *similarity*

Data	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	0,863	0,928	0,882	0,871	0,478	0,595	0,695	0,769	0,712	0,78	0,673	0,597	0,636	0,554
2	0,863	1	0,883	0,926	0,871	0,579	0,677	0,721	0,83	0,803	0,687	0,639	0,533	0,632	0,568
3	0,928	0,883	1	0,881	0,904	0,463	0,627	0,718	0,755	0,821	0,721	0,734	0,685	0,717	0,671
4	0,882	0,926	0,881	1	0,859	0,351	0,526	0,687	0,758	0,73	0,765	0,71	0,66	0,689	0,615
5	0,871	0,871	0,904	0,859	1	0,48	0,755	0,835	0,834	0,808	0,878	0,859	0,797	0,843	0,719
6	0,478	0,579	0,463	0,351	0,48	1	0,787	0,599	0,713	0,617	0,239	0,162	0,076	0,189	0,39
7	0,595	0,677	0,627	0,526	0,755	0,787	1	0,899	0,801	0,653	0,602	0,544	0,522	0,501	0,777
8	0,695	0,721	0,718	0,687	0,835	0,599	0,899	1	0,817	0,709	0,72	0,733	0,728	0,664	0,877
9	0,769	0,83	0,755	0,758	0,834	0,713	0,801	0,817	1	0,802	0,668	0,642	0,611	0,616	0,619
10	0,712	0,803	0,821	0,730	0,808	0,617	0,653	0,709	0,802	1	0,572	0,555	0,533	0,572	0,563
11	0,78	0,687	0,721	0,765	0,878	0,239	0,602	0,72	0,668	0,572	1	0,845	0,818	0,811	0,585
12	0,673	0,639	0,734	0,71	0,859	0,162	0,544	0,733	0,642	0,555	0,845	1	0,956	0,979	0,666
13	0,597	0,533	0,685	0,66	0,797	0,076	0,522	0,728	0,611	0,533	0,818	0,956	1	0,917	0,731
14	0,636	0,632	0,717	0,689	0,843	0,189	0,501	0,664	0,616	0,572	0,811	0,979	0,917	1	0,583
15	0,554	0,568	0,671	0,615	0,719	0,39	0,777	0,877	0,619	0,563	0,585	0,666	0,731	0,583	1

Kemudian nilai dari tabel *similarity* tersebut diubah dengan menggunakan persamaan 2.2 untuk mendapatkan nilai *dissimilarity*. Misal untuk menghitung nilai *dissimilarity* dari data 1 ke data 2 maka

$$d = 1 - d_i$$

$$d(1,2) = 1 - d_{1,2}$$

$$d(1,2) = 1 - 0.8633 = 0.1366$$

Dengan persamaan yang sama dihitung nilai *dissimilarity* semua data. Tabel nilai *dissimilarity* dapat dilihat pada tabel 3.4

Tabel 3.4 Tabel *dissimilarity*

Data	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	0,136	0,071	0,117	0,128	0,5217	0,404	0,304	0,23	0,287	0,219	0,326	0,402	0,363	0,445
2	0,136	0	0,116	0,073	0,128	0,420	0,322	0,278	0,169	0,196	0,312	0,36	0,466	0,367	0,431
3	0,071	0,116	0	0,118	0,095	0,536	0,372	0,281	0,244	0,178	0,278	0,265	0,314	0,282	0,328
4	0,117	0,073	0,118	0	0,14	0,648	0,473	0,312	0,241	0,269	0,234	0,289	0,339	0,31	0,384
5	0,128	0,128	0,095	0,14	0	0,519	0,244	0,164	0,165	0,191	0,121	0,14	0,202	0,156	0,28
6	0,521	0,42	0,536	0,648	0,519	0	0,212	0,40	0,286	0,382	0,76	0,837	0,923	0,81	0,609
7	0,404	0,322	0,372	0,473	0,244	0,212	0	0,10	0,198	0,346	0,397	0,455	0,477	0,498	0,222
8	0,304	0,278	0,281	0,312	0,164	0,40	0,10	0	0,182	0,29	0,279	0,266	0,271	0,335	0,122
9	0,23	0,169	0,244	0,241	0,165	0,286	0,198	0,182	0	0,197	0,331	0,357	0,388	0,383	0,38
10	0,287	0,196	0,178	0,269	0,191	0,382	0,346	0,29	0,197	0	0,427	0,444	0,466	0,427	0,436
11	0,219	0,312	0,278	0,234	0,121	0,76	0,397	0,279	0,331	0,427	0	0,154	0,181	0,188	0,414
12	0,326	0,36	0,265	0,289	0,14	0,837	0,455	0,266	0,357	0,444	0,154	0	0,043	0,02	0,333
13	0,402	0,466	0,314	0,339	0,202	0,923	0,477	0,271	0,388	0,466	0,181	0,043	0	0,082	0,268
14	0,363	0,367	0,282	0,31	0,156	0,81	0,498	0,335	0,383	0,427	0,188	0,02	0,082	0	0,416
15	0,445	0,4315	0,328	0,3845	0,28	0,6094	0,222	0,122	0,38	0,436	0,414	0,333	0,268	0,416	0

Langkah selanjutnya yaitu membuat tabel jarak terdekat dari tabel *dissimilarity* yang telah terbentuk. Tabel jarak terdekat merupakan tabel yang berisi jarak *dissimilarity* antar data untuk semua data.

Tabel 3.5 Tabel jarak terdekat

Data	1	Data	2	Data	3	Data	4
1	0	1	0,136648	1	0,071253	1	0,117908
2	0,136648	2	0	2	0,116192	2	0,073005
3	0,071253	3	0,116192	3	0	3	0,11869
4	0,117908	4	0,073005	4	0,11869	4	0
5	0,128941	5	0,128219	5	0,095331	5	0,140953
6	0,521753	6	0,420579	6	0,536755	6	0,648091
7	0,404583	7	0,322396	7	0,372761	7	0,47346
8	0,304925	8	0,278787	8	0,28106	8	0,312485
9	0,230007	9	0,169952	9	0,244408	9	0,241566
10	0,287253	10	0,196655	10	0,178958	10	0,269823
11	0,219105	11	0,312251	11	0,278836	11	0,234986
12	0,326313	12	0,360612	12	0,265366	12	0,289617
13	0,402192	13	0,466669	13	0,314601	13	0,339464
14	0,363291	14	0,367686	14	0,282877	14	0,310733
15	0,44524	15	0,431582	15	0,328558	15	0,384567
Data	5	Data	6	Data	7	Data	8
1	0,128941	1	0,521753	1	0,404583	1	0,304925
2	0,128219	2	0,420579	2	0,322396	2	0,278787
3	0,095331	3	0,536755	3	0,372761	3	0,28106
4	0,140953	4	0,648091	4	0,47346	4	0,312485
5	0	5	0,519023	5	0,244367	5	0,16434
6	0,519023	6	0	6	0,212511	6	0,400609
7	0,244367	7	0,212511	7	0	7	0,100426
8	0,16434	8	0,400609	8	0,100426	8	0
9	0,165768	9	0,286544	9	0,198275	9	0,18256
10	0,191603	10	0,382384	10	0,346902	10	0,2904
11	0,121832	11	0,760508	11	0,397524	11	0,279652
12	0,140707	12	0,837662	12	0,455722	12	0,266115
13	0,202345	13	0,923255	13	0,477554	13	0,27181
14	0,156533	14	0,810104	14	0,498572	14	0,335072
15	0,280047	15	0,609441	15	0,222929	15	0,122396
Data	9	Data	10	Data	11	Data	12
1	0,230007	1	0,287253	1	0,219105	1	0,326313

2	0,169952	2	0,196655	2	0,312251	2	0,360612
3	0,244408	3	0,178958	3	0,278836	3	0,265366
4	0,241566	4	0,269823	4	0,234986	4	0,289617
5	0,165768	5	0,191603	5	0,121832	5	0,140707
6	0,286544	6	0,382384	6	0,760508	6	0,837662
7	0,198275	7	0,346902	7	0,397524	7	0,455722
8	0,18256	8	0,2904	8	0,279652	8	0,266115
9	0	9	0,197467	9	0,331525	9	0,357024
10	0,197467	10	0	10	0,42753	10	0,44469
11	0,331525	11	0,42753	11	0	11	0,154652
12	0,357024	12	0,44469	12	0,154652	12	0
13	0,388457	13	0,466469	13	0,181056	13	0,043701
14	0,383846	14	0,427565	14	0,18831	14	0,020376
15	0,380209	15	0,436945	15	0,414004	15	0,33318

Data	13	Data	14	Data	15
1	0,402192	1	0,363291	1	0,44524
2	0,466669	2	0,367686	2	0,431582
3	0,314601	3	0,282877	3	0,328558
4	0,339464	4	0,310733	4	0,384567
5	0,202345	5	0,156533	5	0,280047
6	0,923255	6	0,810104	6	0,609441
7	0,477554	7	0,498572	7	0,222929
8	0,27181	8	0,335072	8	0,122396
9	0,388457	9	0,383846	9	0,380209
10	0,466469	10	0,427565	10	0,436945
11	0,181056	11	0,18831	11	0,414004
12	0,043701	12	0,020376	12	0,33318
13	0	13	0,082077	13	0,268109
14	0,082077	14	0	14	0,41624
15	0,268109	15	0,41624	15	0

Selanjutnya tabel jarak terdekat tersebut diurutkan dari jarak terkecil hingga jarak terbesar sesuai jarak *dissimilarity*. Nilai *dissimilarity* 0 dihilangkan karena nilai tersebut merupakan nilai *dissimilarity* dengan data itu sendiri. Data yang memiliki nilai *dissimilarity* terkecil akan berada pada tabel urutan paling atas dan data yang memiliki nilai *dissimilarity* terbesar akan berada pada tabel urutan paling bawah.

Tabel 3.6 Tabel jarak terdekat yang telah diurutkan secara *ascending*

Data	1	Data	2	Data	3	Data	4
3	0,071253	4	0,073005	1	0,071253	2	0,073005
4	0,117908	3	0,116192	5	0,095331	1	0,117908
5	0,128941	5	0,128219	2	0,116192	3	0,11869
2	0,136648	1	0,136648	4	0,11869	5	0,140953
11	0,219105	9	0,169952	10	0,178958	11	0,234986
9	0,230007	10	0,196655	9	0,244408	9	0,241566
10	0,287253	8	0,278787	12	0,265366	10	0,269823
8	0,304925	11	0,312251	11	0,278836	12	0,289617
12	0,326313	7	0,322396	8	0,28106	14	0,310733
14	0,363291	12	0,360612	14	0,282877	8	0,312485
13	0,402192	14	0,367686	13	0,314601	13	0,339464
7	0,404583	6	0,420579	15	0,328558	15	0,384567
15	0,44524	15	0,431582	7	0,372761	7	0,47346
6	0,521753	13	0,466669	6	0,536755	6	0,648091
Data	5	Data	6	Data	7	Data	8
3	0,095331	7	0,212511	8	0,100426	7	0,100426
11	0,121832	9	0,286544	9	0,198275	15	0,122396
2	0,128219	10	0,382384	6	0,212511	5	0,16434
1	0,128941	8	0,400609	15	0,222929	9	0,18256
12	0,140707	2	0,420579	5	0,244367	12	0,266115
4	0,140953	5	0,519023	2	0,322396	13	0,27181
14	0,156533	1	0,521753	10	0,346902	2	0,278787
8	0,16434	3	0,536755	3	0,372761	11	0,279652
9	0,165768	15	0,609441	11	0,397524	3	0,28106
10	0,191603	4	0,648091	1	0,404583	10	0,2904
13	0,202345	11	0,760508	12	0,455722	1	0,304925
7	0,244367	14	0,810104	4	0,47346	4	0,312485
15	0,280047	12	0,837662	13	0,477554	14	0,335072
6	0,519023	13	0,923255	14	0,498572	6	0,400609
Data	9	Data	10	Data	11	Data	12
5	0,165768	3	0,178958	5	0,121832	14	0,020376
2	0,169952	5	0,191603	12	0,154652	13	0,043701
8	0,18256	2	0,196655	13	0,181056	5	0,140707
10	0,197467	9	0,197467	14	0,18831	11	0,154652
7	0,198275	4	0,269823	1	0,219105	3	0,265366
1	0,230007	1	0,287253	4	0,234986	8	0,266115
4	0,241566	8	0,2904	3	0,278836	4	0,289617
3	0,244408	7	0,346902	8	0,279652	1	0,326313

6	0,286544	6	0,382384	2	0,312251	15	0,33318
11	0,331525	11	0,42753	9	0,331525	9	0,357024
12	0,357024	14	0,427565	7	0,397524	2	0,360612
15	0,380209	15	0,436945	15	0,414004	10	0,44469
14	0,383846	12	0,44469	10	0,42753	7	0,455722
13	0,388457	13	0,466469	6	0,760508	6	0,837662

Data	13	Data	14	Data	15
12	0,043701	12	0,020376	8	0,122396
14	0,082077	13	0,082077	7	0,222929
11	0,181056	5	0,156533	13	0,268109
5	0,202345	11	0,18831	5	0,280047
15	0,268109	3	0,282877	3	0,328558
8	0,27181	4	0,310733	12	0,33318
3	0,314601	8	0,335072	9	0,380209
4	0,339464	1	0,363291	4	0,384567
9	0,388457	2	0,367686	11	0,414004
1	0,402192	9	0,383846	14	0,41624
10	0,466469	15	0,41624	2	0,431582
2	0,466669	10	0,427565	10	0,436945
7	0,477554	7	0,498572	1	0,44524
6	0,923255	6	0,810104	6	0,609441

Langkah selanjutnya yaitu menentukan nilai k. nilai k merupakan nilai banyaknya tetangga terdekat yang akan diambil sebagai data perhitungan. Misal, jika ditentukan nilai k sebesar 10 maka akan diambil hanya 10 data jarak terdekat saja dari tabel jarak terdekat yang telah diurutkan. Sisa data yang tidak masuk dalam k tetangga terdekat akan diabaikan dalam proses perhitungan.

Tabel 3.7 Tabel jarak terdekat yang telah diurutkan secara *ascending* dengan k = 10

Data	1	Data	2	Data	3	Data	4
3	0,071253	4	0,073005	1	0,071253	2	0,073005
4	0,117908	3	0,116192	5	0,095331	1	0,117908
5	0,128941	5	0,128219	2	0,116192	3	0,11869
2	0,136648	1	0,136648	4	0,11869	5	0,140953
11	0,219105	9	0,169952	10	0,178958	11	0,234986
9	0,230007	10	0,196655	9	0,244408	9	0,241566
10	0,287253	8	0,278787	12	0,265366	10	0,269823
8	0,304925	11	0,312251	11	0,278836	12	0,289617

12	0,326313	7	0,322396	8	0,28106	14	0,310733
14	0,363291	12	0,360612	14	0,282877	8	0,312485
Data	5	Data	6	Data	7	Data	8
3	0,095331	7	0,212511	8	0,100426	7	0,100426
11	0,121832	9	0,286544	9	0,198275	15	0,122396
2	0,128219	10	0,382384	6	0,212511	5	0,16434
1	0,128941	8	0,400609	15	0,222929	9	0,18256
12	0,140707	2	0,420579	5	0,244367	12	0,266115
4	0,140953	5	0,519023	2	0,322396	13	0,27181
14	0,156533	1	0,521753	10	0,346902	2	0,278787
8	0,16434	3	0,536755	3	0,372761	11	0,279652
9	0,165768	15	0,609441	11	0,397524	3	0,28106
10	0,191603	4	0,648091	1	0,404583	10	0,2904
Data	9	Data	10	Data	11	Data	12
5	0,165768	3	0,178958	5	0,121832	14	0,020376
2	0,169952	5	0,191603	12	0,154652	13	0,043701
8	0,18256	2	0,196655	13	0,181056	5	0,140707
10	0,197467	9	0,197467	14	0,18831	11	0,154652
7	0,198275	4	0,269823	1	0,219105	3	0,265366
1	0,230007	1	0,287253	4	0,234986	8	0,266115
4	0,241566	8	0,2904	3	0,278836	4	0,289617
3	0,244408	7	0,346902	8	0,279652	1	0,326313
6	0,286544	6	0,382384	2	0,312251	15	0,33318
11	0,331525	11	0,42753	9	0,331525	9	0,357024
Data	13	Data	14	Data	15		
12	0,043701	12	0,020376	8	0,122396		
14	0,082077	13	0,082077	7	0,222929		
11	0,181056	5	0,156533	13	0,268109		
5	0,202345	11	0,18831	5	0,280047		
15	0,268109	3	0,282877	3	0,328558		
8	0,27181	4	0,310733	12	0,33318		
3	0,314601	8	0,335072	9	0,380209		
4	0,339464	1	0,363291	4	0,384567		
9	0,388457	2	0,367686	11	0,414004		
1	0,402192	9	0,383846	14	0,41624		

Selanjutnya dimasukan nilai ambang batas ketetanggaan (CP) dan nilai ambang batas kedekatan (MP). Nilai CP merupakan nilai batas minimal dari nilai *dissimilarity* yang akan diputuskan. Sedangkan nilai MP merupakan nilai batas penghentian iterasi dalam

pembentukan *cluster*. Dimisalkan nilai CP yang diinputkan adalah 0,5 maka nilai *dissimilarity* yang lebih besar dari 0,5 akan diputuskan bobot ketetanggaannya atau diabaikan dari perhitungan dan diambil nilai *dissimilarity* yang lebih kecil sama dengan 0,5 dan dimisalkan nilai MP yang dimasukan adalah 0,5 maka iterasi akan berhenti pada saat nilai prosentase perubahan bobot kurang dari 0,5 atau 50%.

Tabel 3.8 Tabel jarak terdekat dengan nilai *similarity* kurang dari CP

Data	1	Data	2	Data	3	Data	4
3	0,071253	4	0,073005	1	0,071253	2	0,073005
4	0,117908	3	0,116192	5	0,095331	1	0,117908
5	0,128941	5	0,128219	2	0,116192	3	0,11869
2	0,136648	1	0,136648	4	0,11869	5	0,140953
11	0,219105	9	0,169952	10	0,178958	11	0,234986
9	0,230007	10	0,196655	9	0,244408	9	0,241566
10	0,287253	8	0,278787	12	0,265366	10	0,269823
8	0,304925	11	0,312251	11	0,278836	12	0,289617
12	0,326313	7	0,322396	8	0,28106	14	0,310733
14	0,363291	12	0,360612	14	0,282877	8	0,312485
Data	5	Data	6	Data	7	Data	8
3	0,095331	7	0,212511	8	0,100426	7	0,100426
11	0,121832	9	0,286544	9	0,198275	15	0,122396
2	0,128219	10	0,382384	6	0,212511	5	0,16434
1	0,128941	8	0,400609	15	0,222929	9	0,18256
12	0,140707	2	0,420579	5	0,244367	12	0,266115
4	0,140953	5	0,519023	2	0,322396	13	0,27181
14	0,156533	1	0,521753	10	0,346902	2	0,278787
8	0,16434	3	0,536755	3	0,372761	11	0,279652
9	0,165768	15	0,609441	11	0,397524	3	0,28106
10	0,191603	4	0,648091	1	0,404583	10	0,2904
Data	9	Data	10	Data	11	Data	12
5	0,165768	3	0,178958	5	0,121832	14	0,020376
2	0,169952	5	0,191603	12	0,154652	13	0,043701
8	0,18256	2	0,196655	13	0,181056	5	0,140707
10	0,197467	9	0,197467	14	0,18831	11	0,154652
7	0,198275	4	0,269823	1	0,219105	3	0,265366
1	0,230007	1	0,287253	4	0,234986	8	0,266115
4	0,241566	8	0,2904	3	0,278836	4	0,289617
3	0,244408	7	0,346902	8	0,279652	1	0,326313
6	0,286544	6	0,382384	2	0,312251	15	0,33318

11	0,331525	11	0,42753	9	0,331525	9	0,357024
Data	13	Data	14	Data	15		
12	0,043701	12	0,020376	8	0,122396		
14	0,082077	13	0,082077	7	0,222929		
11	0,181056	5	0,156533	13	0,268109		
5	0,202345	11	0,18831	5	0,280047		
15	0,268109	3	0,282877	3	0,328558		
8	0,27181	4	0,310733	12	0,33318		
3	0,314601	8	0,335072	9	0,380209		
4	0,339464	1	0,363291	4	0,384567		
9	0,388457	2	0,367686	11	0,414004		
1	0,402192	9	0,383846	14	0,41624		

Proses selanjutnya yaitu proses pembentukan *cluster* dalam beberapa iterasi. Nilai CP akan terus diturunkan sebesar 0,1 untuk tiap iterasinya. Dimisalkan pada iterasi pertama nilai CP adalah 0,4 maka pada iterasi kedua nilai CP adalah 0,3 dan seterusnya sampai proses pembentukan *cluster* berhenti atau sampai nilai CP adalah 0.

Iterasi pertama.

Nilai CP = 0,4.

Putuskan nilai *dissimilarity* yang lebih dari 0,4 dan ambil nilai *dissimilarity* yang kurang dari sama dengan 0,4.

Tabel 3.9 Tabel jarak terdekat iterasi pertama

Data	1	Data	2	Data	3	Data	4
3	0,071253	4	0,073005	1	0,071253	2	0,073005
4	0,117908	3	0,116192	5	0,095331	1	0,117908
5	0,128941	5	0,128219	2	0,116192	3	0,11869
2	0,136648	1	0,136648	4	0,11869	5	0,140953
11	0,219105	9	0,169952	10	0,178958	11	0,234986
9	0,230007	10	0,196655	9	0,244408	9	0,241566
10	0,287253	8	0,278787	12	0,265366	10	0,269823
8	0,304925	11	0,312251	11	0,278836	12	0,289617
12	0,326313	7	0,322396	8	0,28106	14	0,310733
14	0,363291	12	0,360612	14	0,282877	8	0,312485
Data	5	Data	6	Data	7	Data	8
3	0,095331	7	0,212511	8	0,100426	7	0,100426
11	0,121832	9	0,286544	9	0,198275	15	0,122396
2	0,128219	10	0,382384	6	0,212511	5	0,16434

1	0,128941	8	0,400609	15	0,222929	9	0,18256
12	0,140707	2	0,420579	5	0,244367	12	0,266115
4	0,140953	5	0,519023	2	0,322396	13	0,27181
14	0,156533	1	0,521753	10	0,346902	2	0,278787
8	0,16434	3	0,536755	3	0,372761	11	0,279652
9	0,165768	15	0,609441	11	0,397524	3	0,28106
10	0,191603	4	0,648091	1	0,404583	10	0,2904
Data	9	Data	10	Data	11	Data	12
5	0,165768	3	0,178958	5	0,121832	14	0,020376
2	0,169952	5	0,191603	12	0,154652	13	0,043701
8	0,18256	2	0,196655	13	0,181056	5	0,140707
10	0,197467	9	0,197467	14	0,18831	11	0,154652
7	0,198275	4	0,269823	1	0,219105	3	0,265366
1	0,230007	1	0,287253	4	0,234986	8	0,266115
4	0,241566	8	0,2904	3	0,278836	4	0,289617
3	0,244408	7	0,346902	8	0,279652	1	0,326313
6	0,286544	6	0,382384	2	0,312251	15	0,33318
11	0,331525	11	0,42753	9	0,331525	9	0,357024
Data	13	Data	14	Data	15		
12	0,043701	12	0,020376	8	0,122396		
14	0,082077	13	0,082077	7	0,222929		
11	0,181056	5	0,156533	13	0,268109		
5	0,202345	11	0,18831	5	0,280047		
15	0,268109	3	0,282877	3	0,328558		
8	0,27181	4	0,310733	12	0,33318		
3	0,314601	8	0,335072	9	0,380209		
4	0,339464	1	0,363291	4	0,384567		
9	0,388457	2	0,367686	11	0,414004		
1	0,402192	9	0,383846	14	0,41624		

Dari tabel 3.9 dapat diketahui nilai *dissimilarity* mana saja yang diputuskan bobot ketetanggaannya dan hitung total bobot ketetanggaan yang diputuskan.

Tabel 3.10 Tabel total nilai *dissimilarity* yang diputuskan pada iterasi pertama

Data		Jarak
6	8	0,4006
6	2	0,4206
7	1	0,4046
10	11	0,4275
13	1	0,4022
15	11	0,414
15	14	0,4162
Total		2,8857

Tabel 3.10 menunjukkan bahwa pada iterasi pertama terdapat 7 nilai *dissimilarity* yang diputuskan karena nilainya lebih dari 0,4 dengan total nilai *dissimilarity* yang diputuskan adalah 2,8857. Data yang nilai *dissimilarity* nya diputuskan adalah jarak antara data 6 dengan data 8, data 6 dengan data 2, dan seterusnya sampai data 15 dengan data 14.

Iterasi kedua.

Nilai CP = 0,3.

Putuskan nilai *dissimilarity* yang lebih dari 0,3 dan ambil nilai *dissimilarity* yang kurang dari sama dengan 0,3.

Tabel 3.11 Tabel jarak terdekat iterasi kedua

Data	1	Data	2	Data	3	Data	4
3	0,071253	4	0,073005	1	0,071253	2	0,073005
4	0,117908	3	0,116192	5	0,095331	1	0,117908
5	0,128941	5	0,128219	2	0,116192	3	0,11869
2	0,136648	1	0,136648	4	0,11869	5	0,140953
11	0,219105	9	0,169952	10	0,178958	11	0,234986
9	0,230007	10	0,196655	9	0,244408	9	0,241566
10	0,287253	8	0,278787	12	0,265366	10	0,269823
8	0,304925	11	0,312251	11	0,278836	12	0,289617
12	0,326313	7	0,322396	8	0,28106	14	0,310733
14	0,363291	12	0,360612	14	0,282877	8	0,312485
Data	5	Data	6	Data	7	Data	8
3	0,095331	7	0,212511	8	0,100426	7	0,100426
11	0,121832	9	0,286544	9	0,198275	15	0,122396
2	0,128219	10	0,382384	6	0,212511	5	0,16434

1	0,128941	8	0,400609	15	0,222929	9	0,18256
12	0,140707	2	0,420579	5	0,244367	12	0,266115
4	0,140953	5	0,519023	2	0,322396	13	0,27181
14	0,156533	1	0,521753	10	0,346902	2	0,278787
8	0,16434	3	0,536755	3	0,372761	11	0,279652
9	0,165768	15	0,609441	11	0,397524	3	0,28106
10	0,191603	4	0,648091	1	0,404583	10	0,2904
Data	9	Data	10	Data	11	Data	12
5	0,165768	3	0,178958	5	0,121832	14	0,020376
2	0,169952	5	0,191603	12	0,154652	13	0,043701
8	0,18256	2	0,196655	13	0,181056	5	0,140707
10	0,197467	9	0,197467	14	0,18831	11	0,154652
7	0,198275	4	0,269823	1	0,219105	3	0,265366
1	0,230007	1	0,287253	4	0,234986	8	0,266115
4	0,241566	8	0,2904	3	0,278836	4	0,289617
3	0,244408	7	0,346902	8	0,279652	1	0,326313
6	0,286544	6	0,382384	2	0,312251	15	0,33318
11	0,331525	11	0,42753	9	0,331525	9	0,357024
Data	13	Data	14	Data	15		
12	0,043701	12	0,020376	8	0,122396		
14	0,082077	13	0,082077	7	0,222929		
11	0,181056	5	0,156533	13	0,268109		
5	0,202345	11	0,18831	5	0,280047		
15	0,268109	3	0,282877	3	0,328558		
8	0,27181	4	0,310733	12	0,33318		
3	0,314601	8	0,335072	9	0,380209		
4	0,339464	1	0,363291	4	0,384567		
9	0,388457	2	0,367686	11	0,414004		
1	0,402192	9	0,383846	14	0,41624		

Dari tabel 3.11 tentukan nilai *dissimilarity* mana saja yang diputuskan bobot ketetanggaannya dan hitung total bobot ketetanggaan yang diputuskan.

Tabel 3.12 Tabel total nilai *dissimilarity* yang diputuskan pada iterasi kedua

Data		Jarak
1	8	0,304924779
1	12	0,32631259
1	14	0,363291001
2	11	0,312251216
2	7	0,322395987
2	12	0,360611621
4	14	0,310733099
4	8	0,312484506
6	10	0,382383658
7	10	0,346902402
7	3	0,37276089
7	11	0,397524072
9	11	0,331524589
12	15	0,333180211
12	9	0,357024318
13	3	0,314601206
13	4	0,339464197
13	9	0,388457391
14	8	0,335072061
14	2	0,367686122
14	9	0,383845631
15	3	0,328557662
15	9	0,380208609
15	4	0,384567005
Total		8,356764825

Tabel 3.12 menunjukkan bahwa pada iterasi kedua terdapat 24 nilai *dissimilarity* yang diputuskan karena nilainya lebih dari 0,3 dengan total nilai *dissimilarity* yang diputuskan adalah 8,3567. Data yang nilai *dissimilarity* nya diputuskan adalah jarak antara data 1 dengan data 8, data 1 dengan data 12, dan seterusnya sampai dengan data 15 dengan data 4.

Pada iterasi kedua akan dihitung nilai prosentase perubahan bobot ketetanggaan dengan menggunakan persamaan 2.13.

$$p_b = \frac{a - b}{b}$$

$$p_b = \frac{8.357 - 2.885}{2.885} = 1.896$$

Dikarenakan nilai P_b lebih besar dari nilai batas ambang kedekatan (MP) maka pembentukan *cluster* dilanjutkan ke iterasi selanjutnya.

Iterasi ketiga.

Nilai CP = 0,2

Putuskan nilai *dissimilarity* yang lebih dari 0,2 dan ambil nilai *dissimilarity* yang kurang dari sama dengan 0,2.

Tabel 3.13 Tabel jarak terdekat iterasi ketiga

Data	1	Data	2	Data	3	Data	4
3	0,071253	4	0,073005	1	0,071253	2	0,073005
4	0,117908	3	0,116192	5	0,095331	1	0,117908
5	0,128941	5	0,128219	2	0,116192	3	0,11869
2	0,136648	1	0,136648	4	0,11869	5	0,140953
11	0,219105	9	0,169952	10	0,178958	11	0,234986
9	0,230007	10	0,196655	9	0,244408	9	0,241566
10	0,287253	8	0,278787	12	0,265366	10	0,269823
8	0,304925	11	0,312251	11	0,278836	12	0,289617
12	0,326313	7	0,322396	8	0,28106	14	0,310733
14	0,363291	12	0,360612	14	0,282877	8	0,312485
Data	5	Data	6	Data	7	Data	8
3	0,095331	7	0,212511	8	0,100426	7	0,100426
11	0,121832	9	0,286544	9	0,198275	15	0,122396
2	0,128219	10	0,382384	6	0,212511	5	0,16434
1	0,128941	8	0,400609	15	0,222929	9	0,18256
12	0,140707	2	0,420579	5	0,244367	12	0,266115
4	0,140953	5	0,519023	2	0,322396	13	0,27181
14	0,156533	1	0,521753	10	0,346902	2	0,278787
8	0,16434	3	0,536755	3	0,372761	11	0,279652
9	0,165768	15	0,609441	11	0,397524	3	0,28106
10	0,191603	4	0,648091	1	0,404583	10	0,2904
Data	9	Data	10	Data	11	Data	12
5	0,165768	3	0,178958	5	0,121832	14	0,020376
2	0,169952	5	0,191603	12	0,154652	13	0,043701
8	0,18256	2	0,196655	13	0,181056	5	0,140707
10	0,197467	9	0,197467	14	0,18831	11	0,154652

7	0,198275	4	0,269823	1	0,219105	3	0,265366
1	0,230007	1	0,287253	4	0,234986	8	0,266115
4	0,241566	8	0,2904	3	0,278836	4	0,289617
3	0,244408	7	0,346902	8	0,279652	1	0,326313
6	0,286544	6	0,382384	2	0,312251	15	0,33318
11	0,331525	11	0,42753	9	0,331525	9	0,357024
Data	13	Data	14	Data	15		
12	0,043701	12	0,020376	8	0,122396		
14	0,082077	13	0,082077	7	0,222929		
11	0,181056	5	0,156533	13	0,268109		
5	0,202345	11	0,18831	5	0,280047		
15	0,268109	3	0,282877	3	0,328558		
8	0,27181	4	0,310733	12	0,33318		
3	0,314601	8	0,335072	9	0,380209		
4	0,339464	1	0,363291	4	0,384567		
9	0,388457	2	0,367686	11	0,414004		
1	0,402192	9	0,383846	14	0,41624		

Dari tabel 3.13 tentukan nilai *dissimilarity* mana saja yang diputuskan bobot ketetanggaannya dan hitung total bobot ketetanggaan yang diputuskan.

Tabel 3.14 Tabel total nilai *dissimilarity* yang diputuskan pada iterasi ketiga

Data		Jarak
1	11	0,21911
1	9	0,23001
1	10	0,28725
2	8	0,27879
3	9	0,24441
3	12	0,26537
3	11	0,27884
3	8	0,28106
3	14	0,28288
4	11	0,23499
4	9	0,24157
4	10	0,26982
4	12	0,28962
6	7	0,21251
6	9	0,28654

7	15	0,22293
7	5	0,24437
8	12	0,26611
8	13	0,27181
8	11	0,27965
8	10	0,2904
13	5	0,20234
13	15	0,26811
15	5	0,28005
Total		6,22852

Tabel 3.14 menunjukkan bahwa pada iterasi ketiga terdapat 24 nilai *dissimilarity* yang diputuskan karena nilainya lebih dari 0,2 dengan total nilai *dissimilarity* yang diputuskan adalah 6,2285. Data yang nilai *dissimilarity* nya diputuskan adalah jarak antara data 1 dengan data 11, data 1 dengan data 9, dan seterusnya sampai dengan data 15 dengan data 5.

Pada iterasi ketiga akan dihitung nilai prosentase perubahan bobot ketetanggaan dengan menggunakan persamaan 2.8.

$$p_b = \frac{a-b}{b}$$

$$p_b = \frac{6.229-8.357}{8.357} = 0{,}255$$

Dikarenakan nilai P_b lebih kecil dari nilai batas ambang kedekatan (MP) maka proses pembentukan *cluster* dihentikan.

Proses selanjutnya adalah proses pembentukan *cluster* untuk tiap iterasi. Data yang memiliki jarak terdekat dengan data lainnya digabung dalam 1 *cluster*. Jika penggabungan menghasilkan *cluster* lebih dari 3 maka dihitung jarak antar *cluster* dan *cluster* yang memiliki jarak paling dekat akan digabung menjadi 1 *cluster* sampai jumlah *cluster* akhir sama dengan 3.

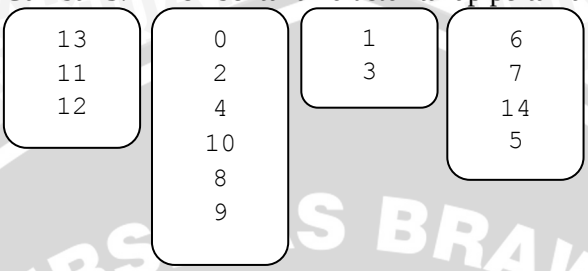
Proses selanjutnya adalah memasukan semua titik yang masuk ke dalam daftar titik terdekat ke dalam satu tabel. Kemudian tabel tersebut diurutkan berdasarkan jarak antar titik, sehingga titik yang memiliki jarak terdekat akan berada pada tabel paling atas dan titik yang memiliki jarak terjauh akan berada pada tabel paling bawah. Tabel daftar tetangga terdekat dapat dilihat pada tabel 3.15.

Tabel 3.15 Daftar tetangga terdekat

Titik pertama	Titik kedua	jarak
13	11	0,0203757096233811
11	13	0,0203757096233811
12	11	0,0437012238018986
11	12	0,0437012238018986
0	2	0,0712533817189892
2	0	0,0712533817189892
	. . . . .	
12	8	0,388457390932286
6	10	0,397524072268264
5	7	0,400609109544534
12	0	0,402191782996117
6	0	0,404583289872188
14	10	0,41400437207245
14	13	0,416240056793237
5	1	0,420578783917513
9	10	0,427530359556394
5	4	0,519022594171601

Dari tabel 3.15 dapat diketahui bahwa titik terdekat adalah titik 13 dan 11 sehingga kedua titik tersebut dimasukan ke dalam 1 *cluster*. Selanjutnya adalah titik 11 dan 13, karena kedua titik tersebut telah masuk ke dalam *cluster* yang sama maka titik tersebut akan diabaikan dari pembentukan cluster. Selanjutnya adalah titik 12 dan 11, karena titik 11 telah masuk ke *cluster* 1 maka titik 12 digabung ke dalam *cluster* 1, dan seterusnya sampai pada penggabungan titik 5 dan titik 4. Jika kedua titik belum masuk ke dalam *cluster* manapun maka kedua titik tersebut dimasukan ke dalam *cluster* baru. Hasil pembentukan *cluster* tahap pertama dapat dilihat pada gambar 3.14.

Gambar 3.14 Pembentukan *cluster* tahap pertama



Karena pembentukan *cluster* pada tahap pertama menghasilkan 4 *cluster* sedangkan hasil dari pembentukan *cluster* harus memiliki *cluster* sejumlah *cluster* awal data yaitu 3 maka akan dilakukan pembentukan *cluster* tahap kedua. Pada pembentukan *cluster* tahap kedua dicari nilai *weight* terlebih dahulu untuk semua data.

Misalnya akan dicari *weight* untuk data pertama maka

$$w_1 = \frac{0{,}93 + 0{,}3 + 0{,}71 + 0{,}37 + 1 + 0{,}61 + 0{,}85 + 0{,}24 + 0{,}78 + 0{,}63 + 0{,}64 + 1 + 0{,}61}{13} = 0{,}666923$$

Begitu juga untuk data kedua dan seterusnya, semua data dicari nilai *weight* sehingga didapatkan hasil sebagai berikut.

Tabel 3.16 Tabel *weight*

Data	Att1	Att2	Att3	Att4	Att5	Att6	Att7	Att8	Att9	Att10	Att11	Att12	Att13	weight
1	14,23	1,71	2,43	15,6	127	2,8	3,06	0,28	2,29	5,64	1,04	3,92	1065	0,667
2	13,2	1,78	2,14	11,2	100	2,65	2,76	0,26	1,28	4,38	1,05	0,4	1050	0,403
3	13,16	2,36	2,67	18,6	101	2,8	3,24	0,3	2,81	5,68	1,03	3,17	1185	0,645
4	14,37	1,95	2,5	16,8	113	3,85	3,49	0,24	2,18	7,8	0,86	0,45	1480	0,657
5	13,24	2,59	2,87	21	118	2,8	2,69	0,39	1,82	4,32	1,04	0,93	735	0,584
6	12,37	0,94	1,36	10,6	88	1,98	0,57	0,28	0,42	1,95	1,05	0,82	520	0,118
7	12,33	1,1	2,28	16	101	2,05	1,09	0,63	0,41	3,27	1,25	0,67	680	0,349
8	12,64	1,36	2,02	16,8	100	2,02	1,41	0,53	0,62	5,75	0,98	0,59	450	0,335
9	13,67	1,25	1,92	18	94	2,1	1,79	0,32	0,73	3,8	1,23	0,46	630	0,36
10	12,37	1,13	2,16	19	87	3,5	3,1	0,19	1,87	4,45	1,22	0,87	420	0,402
11	12,86	1,35	2,32	18	122	1,51	1,25	0,21	0,94	4,1	0,76	0,29	630	0,3
12	12,88	2,99	2,4	20	104	1,3	1,22	0,24	0,83	5,4	0,74	0,42	530	0,337
13	12,81	2,31	2,4	24	98	1,15	1,09	0,27	0,83	5,7	0,66	0,36	560	0,318
14	12,7	3,55	2,36	21,5	106	1,7	1,2	0,17	0,84	5	0,78	0,29	600	0,357
15	12,51	1,24	2,25	17,5	85	2	0,58	0,6	1,25	5,45	0,75	0,51	650	0,302

Setelah *weight* untuk semua data telah didapatkan maka langkah selanjutnya adalah menghitung rata-rata *weight* tiap *cluster*. Dari gambar 3.14 dapat diketahui anggota mana saja yang masuk ke dalam *cluster* 1, *cluster* 2, *cluster* 3 dan *cluster* 4. Rata-rata *cluster* 1 dapat dihitung dengan cara berikut.

$$\overline{w_{C1}} = \frac{w_{11} + w_{12} + w_{13}}{3} = \frac{0{,}336 + 0{,}317 + 0{,}17}{3} = 0{,}3372$$

Cara tersebut diterapkan untuk *cluster* 2, *cluster* 3 dan *cluster* 4 sehingga didapatkan hasil sebagai berikut.

$$\overline{w_{C2}} = 0{,}4929 \quad \overline{w_{C3}} = 0{,}53 \quad \overline{w_{C4}} = 0{,}276$$

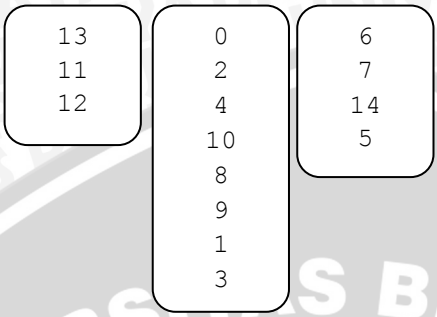
Kemudian dicari jarak antar *cluster* dengan cara sebagai berikut.

$$d_{12} = \left|\overline{w_{C1}} - \overline{w_{C2}}\right| = |0{,}3372 - 0{,}4929| = 0{,}1557$$

Rumus diatas digunakan untuk mencari jarak antar *cluster* untuk semua *cluster* sehingga didapatkan hasil sebagai berikut.

$$d_{13} = 0{,}1928 \quad d_{14} = 0{,}1928 \quad d_{23} = 0{,}0371 \quad d_{24} = 0{,}217$$
$$d_{34} = 0{,}254$$

Dari perhitungan diatas dapat diketahui bahwa jarak terdekat antar *cluster* adalah 0,0371 sehingga anggota dari *cluster* 2 dan anggota dari *cluster* 3 digabung menjadi 1 *cluster*. Hasil penggabungan *cluster* dapat dilihat pada gambar 3.15.



Gambar 3.15 Hasil penggabungan *cluster*

Dikarenakan jumlah *cluster* yang terbentuk sama dengan jumlah kelas awal yaitu 3 maka langkah selanjutnya adalah pemberian label untuk tiap *cluster* baru sehingga dapat diketahui mana yang merupakan *cluster* 1, *cluster* 2 dan *cluster* 3. Untuk melakukan proses pemberian *label* maka setiap *cluster* akhir dicocokan dengan semua *cluster* awal dan dicari data benarnya berapa. *Cluster* akhir yang memiliki nilai benar terbesar maka *cluster* akhir tersebut diasumsikan memiliki *label* yang sama dengan *cluster* awal. Berikut merupakan tabel hasil pencocokan *cluster* awal dengan *cluster* akhir.

Tabel 3.17 Tabel pencocokan anggota *cluster*

No	Cluster awal	Cluster akhir	Jumlah data benar
1	1	1	0
	2	2	2
	3	3	1
2	1	2	5
	2	3	3
	3	1	3
3	1	3	0
	2	1	0
	3	2	1

Dari tabel 3.17 dapat diketahui nilai benar terbanyak adalah pasangan *cluster* no 2 sehingga label untuk hasil *cluster* dapat dilihat pada tabel 3.18

Tabel 3.18 *Label cluster* akhir dan anggotanya

Cluster 1	Cluster 2	Cluster 3
0	6	11
2	7	12
4	14	13
10	5	
8		
9		
1		
3		

Setelah didapatkan *label* untuk *cluster* akhir maka langkah terakhir adalah menghitung nilai akurasi. Akurasi algoritma *Shrinking based Shared Nearest Neighbor* dihitung dengan menggunakan rumus 2.17. Akurasi dihitung untuk tiap *cluster*. Berikut merupakan perhitungan untuk menghitung akurasi *cluster* 1, *cluster* 2 dan *cluster* 3.

$$Akurasi_{cluster1} = \frac{Data\ Benar Cluster\ 1}{Jumlah Anggota Cluster\ 1} = \frac{5}{5} * 100\% = 100\%$$

$$Akurasi_{cluster2} = \frac{Data\ Benar Cluster\ 2}{Jumlah Anggota Cluster\ 2} = \frac{3}{5} * 100\% = 60\%$$

$$Akurasi_{cluster3} = \frac{Data\ Benar Cluster\ 3}{Jumlah Anggota Cluster\ 3} = \frac{3}{5} * 100\% = 60\%$$

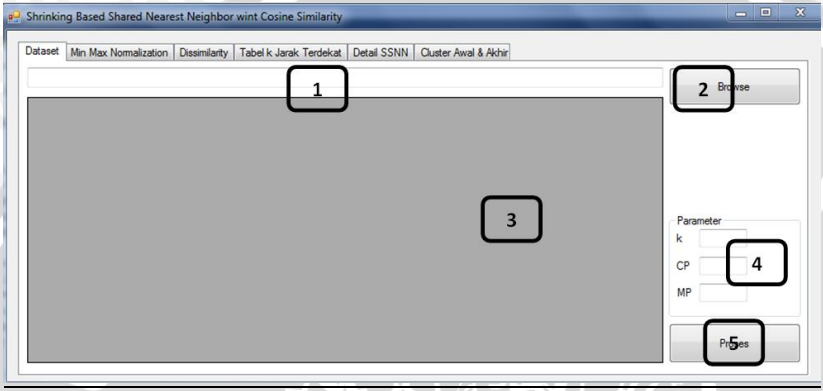
$$Rata - rata\ Akurasi = \frac{Akurasi\ Cluster1 + Akurasi\ Cluster2 + Akurasi\ Cluster3}{3}$$

$$= \frac{100\% + 60\% + 60\%}{3}$$

$$= 73{,}333\%$$

3.4 Perancangan *user interface*

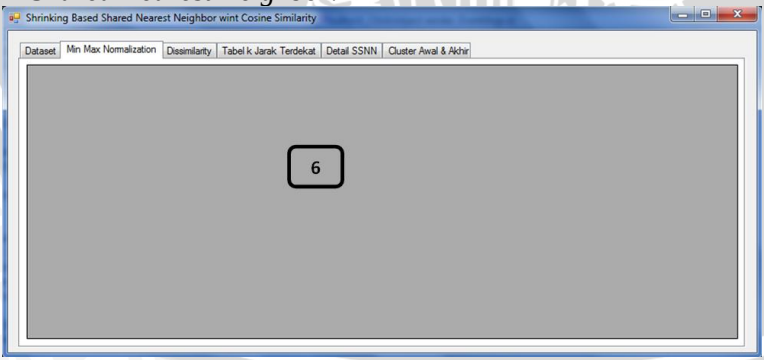
Pada subbab ini akan dijelaskan mengenai perancangan *user interface* dari sistem *clustering dataset* menggunakan algoritma *Shrinking based Shared Nearest Neighbor*.



Gambar 3.16 *User interface load dataset*

Keterangan gambar :

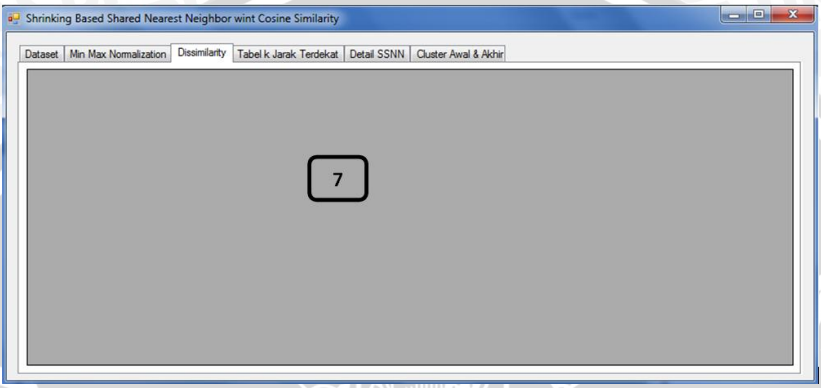
1. Menampilkan alamat *dataset* yang akan di-*load*.
2. Tombol untuk memunculkan *browse box* sehingga *user* dapat memilih *dataset* yang ingin di-*load.*
3. *Datagridview* untuk menampilkan isi *dataset.*
4. Parameter inputan untuk algoritma *Shrinking based Shared Nearest Neighbor*.
5. Tombol untuk memulai *clustering* algoritma *Shrinking based Shared Nearest Neighbor*.



Gambar 3.17 *User interface min-max normalization*

Keterangan gambar :

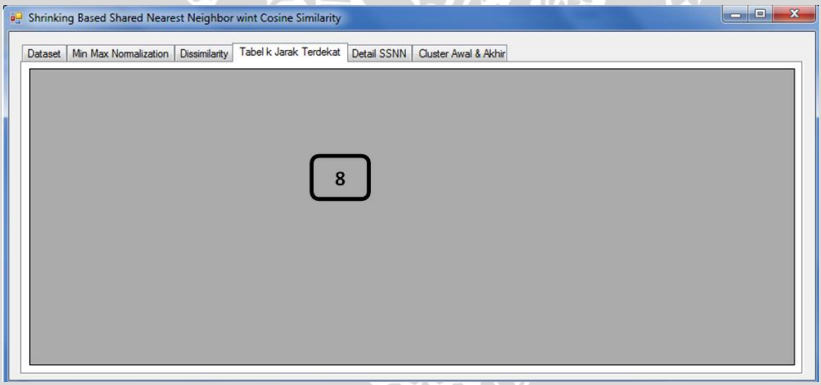
6. *Datagridview* untuk menampilkan *dataset* yang yang telah dinormalisasi.



Gambar 3.18 *User interface* nilai *dissimilarity*

Keterangan gambar :

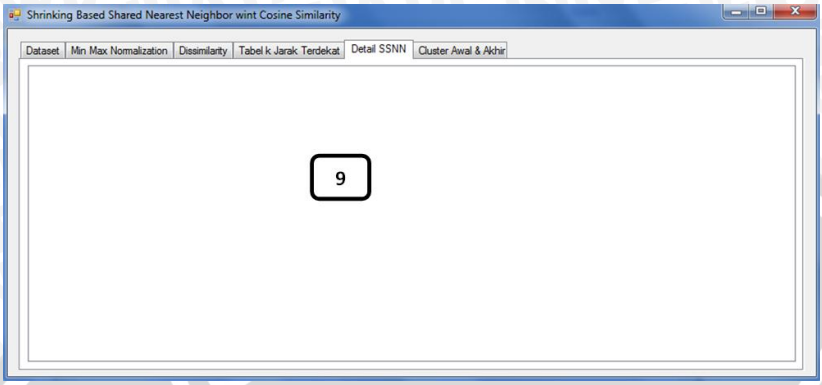
7. *Datagridview* untuk menampilkan nilai *dissimilarity.*



Gambar 3.19 *User interface* k tabel jarak terdekat

Keterangan gambar :

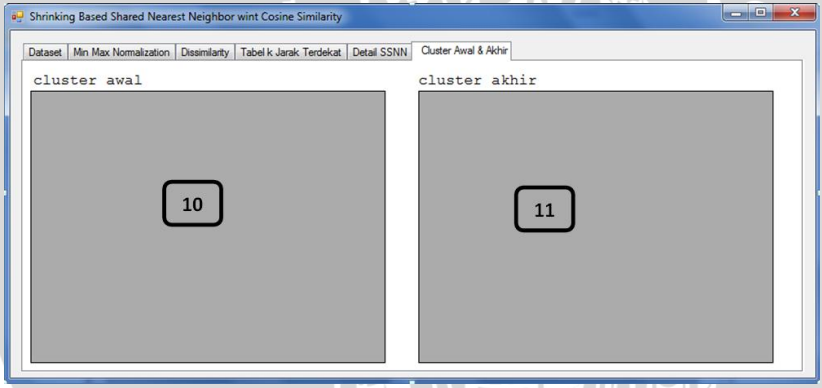
8. *Datagridview* untuk menampilkan k tabel jarak terdekat beserta titiknya.



Gambar 3.20 *User interface* detail SSNN

Keterangan gambar :

9. *Listbox* untuk menampilkan detail perhitungan algoritma *Shrinking based Shared Nearest Neighbor.*



Gambar 3.21 *User interface cluster* awal dan *cluster* akhir

Keterangan gambar :

10. *Datagridview* untuk menampilkan anggota *cluster* awal.
11. *Datagridview* untuk menampilkan anggota *cluster* akhir hasil perhitungan algoritma *Shrinking based Shared Nearest Neighbor.*

3.5 Perancangan uji coba

Pada penelitian ini akan dilakukan uji coba *clustering dataset wine* dan *dataset iris* dengan menggunakan *cosine similarity* pada algoritma *Shrinking based Shared Nearest Neighbor.*

Pengujian untuk *dataset iris* dan *dataset wine* akan dilakukan beberapa kali dengan mengubah parameter-parameternya. nilai k tetangga terdekat adalah mulai dari 1 sampai nilai k maksimal yaitu 149 untuk *dataset iris* dan 177 untuk *dataset wine*, nilai ambang batas ketetanggaan (CP) yang digunakan adalah mulai dari 0,01 sampai dengan 2 dan nilai ambang batas kedekatan (MP) yang digunakan adalah mulai dari 1% sampai dengan 100%. Setelah dilakukan pengujian dengan variasi nilai k, CP dan MP maka akan dipilih akurasi terbesar sebagai hasil dari *clustering* algoritma *Shrinking based Shared Nearest Neighbor*. Pada setiap pengujian, nilai akurasi disimpan dalam tabel pengujian yang dapat dilihat pada tabel 3.20

Tabel 3.20 Tabel evaluasi *Shrinking based Shared Nearest Neighbor*

CP	k	MP	Akurasi

BAB IV
IMPLEMENTASI DAN PEMBAHASAN

4.1 Lingkungan Implementasi

Lingkungan implementasi yang akan dijelaskan dalam subbab ini adalah lingkungan implementasi perangkat keras dan perangkat lunak.

4.1.1 Lingkungan implementasi perangkat keras

Perangkat keras yang digunakan dalam penelitian ini adalah :

1. *Processor Intel Core* i3-2310M
2. *Memory* 4 GB
3. *VGA* 2 GB
4. *Harddisk* dengan kapasitas 500 GB
5. *Monitor* 14”
6. *Keyboard*
7. *Mouse*

4.1.2 Lingkungan implementasi perangkat lunak

Perangkat lunak yang digunakan dalam penelitian ini adalah :

1. Sistem Operasi *Microsoft Windows 7Ultimate 64-bit*
2. *Microsoft Visual Studio Express C# 2010*
3. *Microsoft Office Excel* 2007

4.2 Implementasi Program

Pada subbab implementasi program ini akan dijelaskan mengenai implementasi dari rancangan sistem yang dijelaskan sebelumnya pada bab III.

4.2.1 Memasukan nilai *dataset* ke *List*

Proses pertama adalah memasukan nilai dari *dataset* yang telah ditampilkan ke dalam *list*. Tujuan dari proses ini adalah supaya nilai dari *dataset* dapat diolah untuk proses-proses selanjutnya. Proses memasukan nilai *dataset* ke *list* ditunjukan pada *Source code* 4.1

```
List<List<double>> dataset = new List<List<double>>();
for (int a = 0; a < DGDataset.ColumnCount; a++)
{
    List<double> anggota = new List<double>();
    dataset.Add(anggota);
```

```
}

// memasukan datagridview ke dalam List Dataset
for (int a = 0; a < DGDataset.ColumnCount; a++)
{
    for (int b = 0; b < DGDataset.RowCount; b++)
    {
        double n = Convert.ToDouble(DGDataset[a, b].Value);
        dataset[a].Add(n);
    }
}
```

Source code 4.1 Fungsi untuk memasukan *dataset* ke *list*

Mula-mula *list* 2 dimensi didefinisikan terlebih dahulu dengan nama *dataset*. Kemudian didefinisikan *list* 1 dimensi dengan nama anggota sejumlah banyaknya atribut *dataset*. *List dataset* ditambahkan dengan *list* anggota sejumlah atribut *dataset* sehingga terbentuklah *list dataset* dengan kolom sejumlah kolom *dataset* awal. Dilakukan perulangan sebanyak kolom *dataset* awal dan baris *dataset* awal. Nilai n didefinisikan dengan tipe *double* dimana n adalah nilai dari tiap anggota *dataset*. Kemudian nilai n ditambahkan ke *list dataset*.

4.2.2 Menentukan kelas maksimal *dataset*

Pada kolom terakhir *list dataset* merupakan kolom kelas. Kolom kelas menunjukan bahwa data tersebut masuk ke dalam kelas berapa. Kelas didefinisikan dengan bilangan mulai 0. Program akan mencari berapa nilai maksimal kelas yang ada dalam *list dataset*. Tujuan dari pencarian nilai kelas maksimal adalah untuk penghentian iterasi pada proses selanjutnya. Proses menentukan kelas maksimal *dataset* ditunjukan pada *Source code* 4.2

```
int kolomClass = dataset.Count - 1, MaxClass = 0;
int LebarDataset = dataset[0].Count;
int PanjangDataset = dataset.Count - 1;
for (int a = 0; a < LebarDataset; a++)
{
    int n1 = Convert.ToInt32(dataset[kolomClass][a]);
    if (n1 > MaxClass)
        MaxClass = n1;
}
```

Source code 4.2 Fungsi untuk mencari kelas maksimal

Pertama kolom kelas didefinisikan nilainya. Kolom kelas merupakan kolom terakhir jadi nilai dari kolom kelas adalah banyaknya kolom dikurangi dengan 1. Nilai awal *maxclass* didefinisikan dengan 0. Lebar *dataset* didefinisikan dengan panjang data pada kolom pertama *dataset*. Panjang *dataset* didefinisikan dari kolom pertama sampai kolom terakhir – 1 sehingga kolom kelas tidak termasuk didalamnya. Dilakukan perulangan dari data pertama sampai data terakhir hanya untuk kolom kelas. Semua nilai dari kolom kelas dibandingkan dan yang terbesar akan diambil sebagai nilai *maxclass*.

4.2.3 Memasukan anggota *cluster* awal ke dalam *List*

Anggota dari *cluster* awal dimasukan ke dalam *list*. Proses ini digunakan sebagai tolak ukur pencocokan akurasi pada proses selanjutnya. Proses Memasukan anggota *cluster* awal ke dalam *list* ditunjukan pada *Source code* 4.3

```
List<List<int>> ClusterAwal = new List<List<int>>();
for (int a = 0; a < MaxClass; a++)
{
    List<int> anggota = new List<int>();
    ClusterAwal.Add(anggota);
}
int m1;
for (int a = 0; a < DGDataset.RowCount; a++)
{
    m1 = Convert.ToInt32(DGDataset[kolomClass, a].Value);
    ClusterAwal[m1].Add(a);
}
```

Source code 4.3 Fungsi untuk memasukan anggota *cluster* ke *list*

List 2 dimensi didefinisikan terlebih dahulu dengan nama ClusterAwal. *List* 1 dimensi didefinisikan dengan nama anggota sebanyak jumlah kelas maksimal. Tiap *list* anggota ditambahkan ke dalam *list* ClusterAwal sehingga terbentuk *list* 2 dimensi ClusterAwal dengan banyaknya kolom sejumlah kelas maksimal. Inisialisasi nilai m1. Perulangan untuk semua data dari *dataset* awal. Nilai m1 merupakan nilai *index* data yang masuk ke dalam *cluster* masing-masing. Nilai m1 ditambahkan ke *list* 2 dimensi ClusterAwal sehingga tiap data *dataset* awal telah masuk ke dalam *cluster* masing-masing.

4.2.4 Proses min-max normalization

Data dari *dataset* awal harus dinormalisasi terlebih dahulu sebelum data digunakan untuk proses-proses selanjutnya. Proses *min-max normalization* ditunjukan pada *Source code* 4.4

```
double norm, max, min;
List<double> MinMax = new List<double>();
List<double> Min = new List<double>();
List<double> Max = new List<double>();
List<List<double>> Normalisasi = new List<List<double>>();
for (int a = 0; a < PanjangDataset; a++)
{
    List<double> Anggota = new List<double>();
    Normalisasi.Add(Anggota);
}
for (int a = 0; a < PanjangDataset; a++)
{
    max = dataset[a].Max();
    min = dataset[a].Min();
    Max.Add(max);
    Min.Add(min);
}
for (int a = 0; a < dataset.Count - 1; a++)
{
    for (int b = 0; b < dataset[a].Count; b++)
    {
        norm = (dataset[a][b] - Min[a]) / (Max[a] - Min[a]);
        Normalisasi[a].Add(norm);
    }
}
```

Source code 4.4 Normalisasi *Min Max*

Norm, *max*, dan *min* diinisialisasi dengan tipe *double*. *List* 1 dimensi diinisialisasi dengan nama *MinMax*, *Min* dan *Max* dengan masing-masing *list* bertipe *double*. *List* 2 dimensi diinisialisasi dengan nama Normalisasi. Dilakukan perulangan sebanyak panjang *dataset* dimana panjang *dataset* telah diinisialisasikan nilainya pada proses sebelumnya. Tiap perulangan diinisialisasikan *list* 1 dimensi dengan nama *list* Anggota dan tiap *list* Anggota ditambahkan ke dalam *list* Normalisasi sehingga terbentuklah *list* 2 dimensi dengan panjang kolom sebanyak panjang *dataset*.

Dilakukan *looping* untuk tiap atribut *dataset*. Nilai *max* dan *min* dicari untuk setiap atribut dan kemudian nilai *max* ditambahkan ke dalam *list max* dan nilai *min* ditambahkan ke dalam *list min*. dilakukan *looping* untuk semua data untuk mencari nilai normalisasinya. Setelah nilai normalisasi didapatkan maka nilai

normalisasi ditambahkan ke dalam *list* normalisasi sehingga terbentuklah *list* 2 dimensi untuk semua nilai normalisasi.

4.2.5 Proses perhitungan nilai *dissimilarity*

Langkah selanjutnya adalah menghitung nilai dissimilarity untuk tiap data dengan semua data. Data pertama akan dihitung nilai dissimilarity-nya dengan data pertama sampai data terakhir. Proses tersebut terus dilakukan sampai dengan perhitungan nilai dissimilarity mencapai data terakhir.

4.2.5.1 Perhitungan nilai *dissimilarity* dengan menggunakan *cosine similarity*

Proses perhitungan nilai *dissimilarity* dengan menggunakan *cosine similarity* ditunjukan pada *Source code* 4.5

```
double up, downX, downY, resultDown, result;
List<double> ListUp = new List<double>();
List<double> ListDownX = new List<double>();
List<double> ListDownY = new List<double>();
List<List<double>> ListDis = new List<List<double>>();
for (int a = 0; a < LebarDataset; a++)
{
    List<double> Anggota = new List<double>();
    ListDis.Add(Anggota);
}
double ResultUp, ResultDownX, ResultDownY;
for (int a = 0; a < LebarDataset; a++)
{
    for (int b = 0; b < LebarDataset; b++)
    {
        ListUp.Clear();
        ListDownX.Clear();
        ListDownY.Clear();
        for (int c = 0; c < PanjangDataset; c++)
        {
up = Normalisasi[c][a] * Normalisasi[c][b];
ListUp.Add(up);
downX = Math.Pow(Normalisasi[c][a], 2);
ListDownX.Add(downX);
downY = Math.Pow(Normalisasi[c][b], 2);
ListDownY.Add(downY);
        }
        ResultUp = ListUp.Sum();
        ResultDownX = ListDownX.Sum();
        ResultDownY = ListDownY.Sum();
        resultDown = Math.Sqrt(ResultDownX * ResultDownY);
        result = 1 - (ResultUp / resultDown);
        if (a == b)
result = 0;
        ListDis[a].Add(result);
    }
}
```

Source code 4.5 Fungsi untuk perhitungan nilai *Dissimilarity* dengan menggunakan *cosine similarity*

Pertama-tama variabel *up*, *downX*, *downY*, *resultDown* dan *result* diinisialisasikan dengan tipe *double*. Kemudian dibuat 3 *list* 1 dimensi yaitu *Listup*, *ListdownX* dan *ListdownY* dan 1 *list* 2 dimensi yaitu *ListDis*. Kemudian *list* 1 dimensi dengan tipe *double* diinisialisasikan dengan nama anggota. *List* anggota dibuat sebanyak banyaknya data pada *dataset*. *List* anggota ditambahkan ke dalam *list* 2 dimensi *listdis* sehingga terbentuk *list* 2 dimensi sebagai tempat penyimpanan nilai *dissimilarity*. Variabel *ResultUp*, *ResultDownX*, *ResultDownY* didefinisikan dengan tipe *double*.

Kemudian program akan melakukan perulangan untuk setiap data dengan semua data pada *dataset* normalisasi. Setiap akan menghitung nilai *dissimilarity* dengan data yang berbeda maka *listup*, *listdownX* dan *listdownY* harus dihapus nilainya. Hal ini dimaksudkan karena setiap perhitungan dengan data lain nilai *list* adalah 0. Kemudian program akan melakukan perulangan untuk kolom atau atribut dan melakukan perhitungan nilai *dissimilarity* untuk semua data. Jika nilai *dissimilarity* telah didapatkan maka tiap nilai akan disimpan dalam *listDis*.

4.2.5.2 Perhitungan nilai *dissimilarity* dengan menggunakan *euclidean distance*

Proses perhitungan nilai *dissimilarity* dengan menggunakan *euclidean distance* ditunjukan pada *Source code* 4.6

```
List<List<double>> ListDis = new List<List<double>>();
for (int a = 0; a < LebarDataset; a++)
{
    List<double> Anggota = new List<double>();
    ListDis.Add(Anggota);
}
double kuadrat, result;
List<double> ListKuadrat = new List<double>();

for (int a = 0; a < LebarDataset; a++)
{
    for (int b = 0; b < LebarDataset; b++)
    {
        ListKuadrat.Clear();
        for (int c = 0; c < PanjangDataset; c++)
        {
kuadrat = Math.Pow(((Normalisasi[c][a]) - (Normalisasi[c][b])),
2);
ListKuadrat.Add(kuadrat);
        }
        result = ListKuadrat.Sum();
        result = Math.Sqrt(result);
```

```
        if (a == b)
result = 0;
        ListDis[a].Add(result);
    }
}
```

Source code 4.6 Fungsi untuk perhitungan nilai *Dissimilarity* dengan menggunakan *euclidean distance*

List 2 dimensi didefinisikan untuk menyimpan hasil perhitungan jarak. Variabel kuadrat dan result didedinisikan. List ListKuadrat didefinisikan untuk menyimpan hasil kuadrat. Looping semua anggota dataset dan dihitung nilai dari kuadrat hasil looping atribut. Nilai kuadrat disimpan ke dalam list ListKuadrat. Jika pembacaan atribut telah selesai maka semua nilai ListKuadrat dijumlahkan kemudian diakar. Hasil dari perhitungan jarak tersebut kemudian disimpan ke dalam list 2 dimensi sehingga terbentuk matrix jarak antar data.

4.2.6 Proses mendapatkan tabel jarak terdekat

Setelah nilai *dissimilarity* didapatkan maka proses selanjutnya adalah mengurutkan nilai tersebut berdasarkan jarak terkecil sampai jarak terbesar sehingga diketahui titik mana saja yang memiliki nilai *dissimilarity* terdekat dan terjauh. Proses mendapatkan tabel jarak terdekat ditunjukan pada *Source code* 4.7

```
List<List<double>> ListDisBaru = new List<List<double>>();
for (int a = 0; a < ListDis.Count; a++)
{
    List<double> Anggota = new List<double>();
    ListDisBaru.Add(Anggota);
}
for (int a = 0; a < ListDis.Count; a++)
{
    for (int b = 0; b < ListDis[a].Count; b++)
    {
        ListDisBaru[a].Add(ListDis[a][b]);
    }
}
for (int a = 0; a < ListDisBaru.Count; a++)
{
    ListDisBaru[a].Sort();
}
```

Source code 4.7 Fungsi untuk table jarak terdekat

Pada proses ini dibuat *list* 2 dimensi dengan nama ListDisBaru sebagai tempat penyimpanan nilai jarak terdekat. *List* tersebut memiliki panjang sama dengan kolom *dissimilarity*. Semua nilai *dissimilarity* ditambahkan ke dalam ListDisBaru sehingga semua nilai dalam ListDisBaru sama dengan *listDis*. Kemudian perkolom atau peratribut untuk nilai ListDisBaru diurutkan secara *ascending* sehingga terbentuklah *list* 2 dimensi yang sama dengan *list dissimilarity* hanya saja ListDisBaru nilainya telah diurutkan.

4.2.7 Proses mendapatkan titik terdekat

Setelah mendapatkan jarak terdekat proses selanjutnya adalah mendapatkan titik terdekat. Jadi dari tabel jarak terdekat dapat diketahui titik mana saja yang jaraknya paling dekat dan titik mana saja yang jaraknya paling jauh. Proses mendapatkan titik terdekat ditunjukan pada *Source code* 4.8

```
List<List<int>> IndexMinDistance = new List<List<int>>();
for (int a = 0; a < ListDis.Count; a++)
{
    List<int> Anggota = new List<int>();
    IndexMinDistance.Add(Anggota);
}
for (int a = 0; a < ListDisBaru.Count; a++)
{
    for (int b = 0; b < ListDisBaru[a].Count; b++)
    {
        for (int c = 0; c < ListDis[a].Count; c++)
        {
if (ListDisBaru[a][b] == ListDis[a][c])
{
    IndexMinDistance[a].Add(c);
    break;
}
        }
    }
}
```

Source code 4.8 Fungsi untuk mendapatkan titik terdekat

Mulanya *list* 2 dimensi dibuat terlebih dahulu dengan nama *IndexMinDistance* untuk menyimpan nilai dari titik atau *index* terdekat. *List* tersebut dibuat dengan panjang sebanyak kolom *dissimilarity*. Kemudian akan dilakukan perulangan untuk semua data pada *list dissimilarity*. Data tersebut akan dicocokan dengan nilai pada *list dissimilarity* yang baru. Jika sama maka *index* terdekat telah ditemukan. *Index* ditambahkan ke *list indexMinDistance* dan

proses pencarian. Proses tersebut akan terus dilakukan sampai semua data telah dicocokan.

4.2.8 Proses mendapatkan jarak dan titik terdekat berdasarkan k inputan

Setelah didapatkan daftar jarak terdekat dan daftar titik terdekat. Langkah selanjutnya adalah mendapatkan titik dan jarak terdekat sesuai inputan k. k adalah banyaknya jarak dan titik terdekat yang akan diambil dan digunakan untuk proses perhitungan. Proses mendapatkan jarak dan titik terdekat berdasarkan k inputan ditunjukan pada *Source code* 4.9

```
int k = Convert.ToInt32(textBoxk.Text) + 1;
List<List<double>> kJarak = new List<List<double>>();
List<List<int>> kTitik = new List<List<int>>();

for (int a = 0; a < ListDis.Count; a++)
{
    List<double> Anggota = new List<double>();
    kJarak.Add(Anggota);
}
for (int a = 0; a < ListDis.Count; a++)
{
    List<int> Anggota = new List<int>();
    kTitik.Add(Anggota);
}

for (int a = 0; a < ListDisBaru.Count; a++)
{
    for (int b = 1; b < k; b++)
    {
        double n = ListDisBaru[a][b];
        kJarak[a].Add(n);
    }
}
for (int a = 0; a < IndexMinDistance.Count; a++)
{
    for (int b = 1; b < k; b++)
    {
        int n = IndexMinDistance[a][b];
        kTitik[a].Add(n);
    }
}
```

Source code 4.9 Fungsi untuk mendapatkan jarak dan titik terdekat berdasarkan k inputan

Pada proses ini *user* memasukan nilai k sebagai batas nilai jarak dan titik terdekat yang akan diambil untuk proses selanjutnya. Sebelumnya dibuat *list* 2 dimensi untuk menyimpan jarak dan titik terdekat. Kedua *list* tersebut dibuat dengan panjang masing-masing sama dengan panjang kolom *list dissimilarity*. Kemudian akan

dilakukan perulangan untuk jarak dan titik terdekat untuk semua kolom dan untuk baris hanya akan diambil dari baris ke 2 sampai baris ke-(k+1). hal itu dikarenakan baris pertama jaraknya bernilai 0 dan titik terdekatnya merupakan titik itu sendiri.

4.2.9 Proses perhitungan total bobot untuk iterasi pertama

Langkah selanjutnya adalah menghitung total bobot untuk setiap iterasi. Untuk iterasi pertama total bobot dihitung berdasarkan inputan *user* yaitu Nilai CP. Proses perhitungan total bobot untuk iterasi pertama ditunjukan pada *Source code* 4.10

```
double CP = Convert.ToDouble(textBoxCP.Text);
double MP = Convert.ToDouble(textBoxMP.Text) / 100;
List<double> ListTotBobot = new List<double>();
double totalbobot = 0;
double bobothilang;
List<double> BobotHilang = new List<double>();
int iterasi = 0;
double ProsBobotHilang = 2;

for (int a = 0; a < kJarak.Count; a++)
{
    for (int b = 0; b < kJarak[a].Count; b++)
    {
        if (kJarak[a][b] > CP)
        {
kJarak[a][b] = 0;
        }
        totalbobot += kJarak[a][b];
    }
}
ListTotBobot.Add(totalbobot);
```

Source code 4.10 Fungsi untuk mendapatkan bobot iterasi pertama

Awalnya nilai CP dan MP didapatkan dari inputan *user*. Kemudian didefinisikan *list* 1 dimensi dengan nama ListTotBobot dengan tipe *double*. *List* tersebut digunakan untuk menyimpan total bobot pada iterasi pertama. Kemudian diinisialisasikan nilai awal untuk total bobot samadengan 0. Dianggap tidak ada bobot yang hilang sebelum proses dilakukan. Diinisialisasikan variabel bobothilang dengan tipe *double*. Kemudian *list* 1 dimensi dibuat dengan nama BobotHilang yang digunakan untuk menyimpan banyaknya bobot yang hilang untuk tiap iterasi. Nilai iterasi didefinisikan dengan nilai 0. Hal itu bermaksud bahwa iterasi awal sama dengan 0 bukan 1. Didefinisikan variabel prosBobotHilang

dengan nilai 1. prosBobotHilang bernilai 1 karena untuk proses selanjutnya dilakukan *looping* dengan nilai prosBobotHilang akan terus diturunkan.

Kemudian akan dilakukan perulangan untuk semua data dalam *list* k jarak terdekat. Jika data tersebut lebih dari nilai CP maka data tersebut akan di-set dengan nilai 0 dan data tersebut akan ditambahkan ke variabel totalbobot. Setelah total bobot didapatkan untuk semua data yang dilakukan perulangan maka total bobot disimpan dalam *list* total bobot.

4.2.10 Proses perhitungan total bobot untuk iterasi selanjutnya

Setelah proses iterasi untuk tahap pertama selesai maka proses selanjutnya adalah proses perhitungan total bobot untuk iterasi selanjutnya. Untuk iterasi selanjutnya akan ada syarat untuk menghentikan perulangan yaitu apabila prosentase bobot ketetanggaan yang hilang sama dengan atau kurang dari nilai MP. Proses perhitungan total bobot untuk iterasi selanjutnya ditunjukan pada *Source code* 4.11

```
While (ProsBobotHilang > MP)
{
    iterasi++;
    CP -= 0.01;
    totalbobot = 0;
    for (int a = 0; a < kJarak.Count; a++)
    {
        for (int b = 0; b < kJarak[a].Count; b++)
        {
if (kJarak[a][b] > CP)
{
    kJarak[a][b] = 0;
}
totalbobot += kJarak[a][b];
        }
    }
    ListTotBobot.Add(totalbobot);
    bobothilang = ListTotBobot[iterasi - 1] -
ListTotBobot[iterasi];
    BobotHilang.Add(bobothilang);
    listBox1.Items.Add("bobot ketetanggaan yang hilang iterasi-
" + iterasi + " = " + bobothilang);
    if (iterasi > 1)
    {
        ProsBobotHilang = Math.Abs((BobotHilang[iterasi - 1] -
BobotHilang[iterasi - 2]) / BobotHilang[iterasi - 2]);
        if (BobotHilang[iterasi - 2] == 0)
        {
ProsBobotHilang = 2;
```

```
        }
    }
    listBox1.Items.Add("prosentase perubahan bobot iterasi-" +
iterasi + " = " + ProsBobotHilang);
}
listBox1.Items.Add("total iterasi = " + iterasi.ToString());
```

Source code 4.11 Fungsi untuk menghitung total bobot iterasi kedua dan seterusnya

Selama prosentase bobot ketetanggaan yang hilang lebih dari nilai MP maka untuk setiap iterasi nilai iterasi akan ditambahkan 1, nilai CP diturunkan sebesar 0.01 dan total bobot nilainya di-set 0. Kemudian semua jarak pada *list* kjarak dijumlahkan ke dalam total bobot. Total bobot dimasukan ke dalam *list*. Kemudian bobot ketetanggan yang hilang dihitung dengan cara mengurangkan total bobot pada iterasi sebelumnya dengan total bobot pada iterasi yang sedang dijalankan. Dengan tujuan supaya iterasi terus dilakukan dan menghindari nilai *NaN* maka jika iterasi lebih dari 1 dan bobot ketetanggan yang hilang pada iterasi sebelumnya bernilai 0 maka prosentase bobot yang hilang adalah sama dengan 1. Kemudian prosentase perubahan bobot untuk tiap iterasi dicetak ke *list box* dan jika iterasi berhenti maka akan dicetak total iterasi yang telah dilakukan sehingga dapat diketahui pada iterasi keberapa iterasi tersebut dihentikan.

4.2.11 Proses pencarian titik terdekat beserta jaraknya

Setelah proses penghentian iterasi selesai maka proses selanjutnya adalah proses pencarian titik terdekat beserta jaraknya. Titik titik tersebut akan dimasukan ke *list* 2 dimensi beserta jaraknya masing-masing kemudian diurutkan berdasarkan jarak terdekat sehingga dari *list* tersebut dapat diketahui titik mana yang memiliki jarak terdekat dan tikik mana yang memiliki jarak terjauh. Proses pencarian titik terdekat beserta jaraknya ditunjukan pada *Source code* 4.12

```
List<double> JarakTerdekat = new List<double>();
for (int a = 0; a < kJarak.Count; a++)
{
    for (int b = 0; b < kJarak[a].Count; b++)
    {
        JarakTerdekat.Add(kJarak[a][b]);
    }
```

```
}
JarakTerdekat.Sort();
for (int a = 0; a < JarakTerdekat.Count; a++)
{
    if (JarakTerdekat[a] != 0)
    {
        JarakTerdekat.RemoveRange(0, a);
        break;
    }
}
for (int a = 0; a < JarakTerdekat.Count; a++)
{
    for (int b = 0; b < JarakTerdekat.Count; b++)
    {
        if (a == b)
break;
        if (JarakTerdekat[a] == JarakTerdekat[b])
JarakTerdekat[b] = 0;
    }
}
JarakTerdekat.Sort();
for (int a = 0; a < JarakTerdekat.Count; a++)
{
    if (JarakTerdekat[a] != 0)
    {
        JarakTerdekat.RemoveRange(0, a);
        break;
    }
}

List<double> titik1 = new List<double>();
List<double> titik2 = new List<double>();
for (int a = 0; a < JarakTerdekat.Count; a++)
{
    double n = JarakTerdekat[a];
    for (int b = 0; b < ListDis.Count; b++)
    {
        if (ListDis[b].Contains(n))
        {
titik1.Add(b);
for (int c = 0; c < ListDis[b].Count; c++)
{
    if (ListDis[b][c] == n)
    {
        titik2.Add(c);
        break;
    }
}
break;
        }
    }
}

List<List<double>> Final = new List<List<double>>();
```

```
Final.Add(JarakTerdekat);
Final.Add(titik1);
Final.Add(titik2);
```

Source code 4.12 Fungsi untuk mencari titik terdekat beserta jaraknya

Mulanya didefinisikan *list* 1 dimensi untuk menyimpan jarak terdekat. Kemudian jarak terdekat yang telah dimasukan *list* diurutkan secara *ascending*. Semua nilai jarak terdekat di cek apakah ada yang bernili 0. Jika ada maka nilai tersebut dihapus dari *list* karena nilai tersebut merupakan nilai titik dengan titik itu sendiri. Jarak yang sama akan dihapus dari tabel karena jarak tersebut sebenarnya sama dan akan memperlama proses perhitungan.

List 1 dimensi didefinisikan untuk menyimpan titik pertama dan titik kedua. Nilai dari *list* jarak terdekat akan dicocokan dengan nilai yang ada pada *list dissimilarity*. Jika nilai tersebut ditemukan maka untuk titik pertama adalah *index* kolom *dissimilarity* dan titik kedua adalah *index* baris *dissimilarity*. Kemudian kedua titik tersebut dimasukan ke *list* masing-masing.

List jarak terdekat, *list* titik pertama dan *list* titik kedua dimasukan ke dalam *list* Final sehingga terbentuk *list* 2 dimensi yang berisikan jarak terdekat beserta titik titik yang berhubungan.

4.2.12 Proses memasukan anggota *cluster* awal

Sebelum proses perhitungan akurasi dibutuhkan *list* 2 dimensi yang berisi data yang telah terbagi ke dalam *cluster* masing-masing. Dari kolom kelas *dataset* awal dapat diketahui setiap data masuk ke dalam *cluster* mana saja. Proses memasukan anggota *cluster* awal ditunjukan pada *Source code* 4.13

```
List<List<int>> cluster = new List<List<int>>();
List<int> AngClus = new List<int>();
int panjangFinal = Final[0].Count;
for (int a = 0; a < panjangFinal; a++)
{
    int n1 = Convert.ToInt32(Final[1][a]);
    int n2 = Convert.ToInt32(Final[2][a]);

    if (a == 0)
    {
        List<int> Anggotacluster = new List<int>();
        Anggotacluster.Add(n1);
        Anggotacluster.Add(n2);
        AngClus.Add(n1);
        AngClus.Add(n2);
        cluster.Add(Anggotacluster);
```

```
    }
    else
    {
        if (AngClus.Contains(n1) && AngClus.Contains(n2))
continue;
        else if (AngClus.Contains(n1) && !AngClus.Contains(n2))
        {
for (int b = 0; b < cluster.Count; b++)
{
    var anggota = cluster[b];
    if (anggota.Contains(n1) && !anggota.Contains(n2))
    {
        anggota.Add(n2);
        break;
    }
}
AngClus.Add(n2);
        }
        else if (AngClus.Contains(n2) && !AngClus.Contains(n1))
        {
for (int b = 0; b < cluster.Count; b++)
{
    var anggota = cluster[b];
    if (anggota.Contains(n2) && !anggota.Contains(n1))
    {
        anggota.Add(n1);
        break;
    }
}
AngClus.Add(n1);
        }
        else
        {
List<int> Anggotacluster = new List<int>();
Anggotacluster.Add(n1);
Anggotacluster.Add(n2);
AngClus.Add(n1);
AngClus.Add(n2);
cluster.Add(Anggotacluster);
        }
    }
}
```

Source code 4.13 Fungsi untuk memasukan anggota *cluster* awal

Mulanya *list* 2 dimensi didefinisikan dengan nama *cluster* untuk menyimpan anggota *cluster* yang telah terbentuk. Kemudian didefinisikan *list* 1 dimensi dengan nama angclus untuk menyimpan anggota *cluster* yang telah masuk ke dalam *list cluster*. *List* angclus dibuat dengan tujuan agar tidak terjadi data yang sama masuk dalam cluster yang berbeda. Kemudian dilakukan perulangan untuk *list final* dimana *list final* merupakan *list* 2 dimensi yang berisi daftar titik dengan jarak terdekat hingga terjauh. Diinisialisasikan variabel n1 dan n2. n1 adalah tititk pertama dan n2 adalah titik kedua pada *list final*.

Jika perulangan baru pertama dilakukan maka *cluster* dianggap kosong sehingga semua anggota *cluster* langsung dimasukan ke dalam *list*. Jika perulangan masuk tahap 2 dan seterusnya maka akan ada syarat sebelum kedua titik dimasukan ke dalam *list cluster*. Aturannya adalah sebagai berikut, jika cluster n1 dan n2 telah masuk pada anggota *cluster* maka perulangan akan dilewatkan dan akan dilanjutkan pada perulangan selanjutnya, jika n1 ata n2 salah satu telah masuk ke dalam *cluster* maka akan dicek anggota *cluster* mana yang mengandung n1, jika anggota *cluster* mengandung n1 dan tidak mengandung n2 maka nilai n2 akan dimasukan ke dalam *cluster* dan juga sebaliknya aturan yang terakhir adalah dibuat 1 *list* sebagai *cluster* baru dikarenakan tidak ada satupun dari n1 dan n2 yang masuk ke dalam *cluster* selanjutnya.

4.2.13 Proses perhitungan *weight*

Proses selanjutnya adalah proses perhitungan *weight*. *Weight* tiap data dihitung untuk semua data sehingga didapatkan *weight* sejumlah banyaknya data *dataset*. Proses perhitungan *weight* ditunjukan pada *Source code* 4.14

```
List<double> weight = new List<double>();
int LebarNormalisasi = Normalisasi[0].Count;
for (int a = 0; a < LebarNormalisasi; a++)
{
    double w = 0;
    for (int b = 0; b < Normalisasi.Count; b++)
    {
        w += Normalisasi[b][a];
    }
    w = w / Normalisasi.Count;
    weight.Add(w);
}
```

Source code 4.14 perhitungan *weight*

Awalnya *list* 1 dimensi diinisialisasikan untuk menyimpan *weight* masing-masing data. Kemudian dilakukan perulangan untuk semua data tiap data dilakukan perulangan untuk tiap atributnya. Nilai w didefinisikan dengan nilai 0 untuk setiap perpindahan perulangan data. Nilai w merupakan total dari nilai atribut normalisasi untuk tiap data. Setelah nilai w didapatkan kemudian nilai w dibagi dengan banyaknya atribut untuk tiap data. Setelah nilai w akhir didapatkan maka setiap nilai w dimasukan ke dalam *list*

weight sehingga didapatkan *list weight* yang berisikan *weight* untuk semua data.

4.2.14 Proses penggabungan *cluster* akhir

Tahap selanjutnya adalah menggabungkan *cluster* yang terbentuk pada tahap pertama menjadi sebanyak *cluster* dataset awal. Jika *cluster* yang terbentuk pada pembentukan *cluster* awal kurang dari *cluster* dataset awal, maka proses ini tidak dapat dijalankan dan aplikasi akan otomatis *restart*. Proses penggabungan *cluster* akhir ditunjukan pada *Source code* 4.15

```
while (cluster.Count > MaxClass)
{
    List<double> weightClus = new List<double>();
    weightClus.Clear();
    for (int a = 0; a < cluster.Count; a++)
    {
        double wClus = 0;
        for (int b = 0; b < cluster[a].Count; b++)
        {
wClus += weight[cluster[a][b]];
        }
        wClus = wClus / cluster[a].Count;
        weightClus.Add(wClus);
    }
    List<List<double>> ListWeightClus = new
List<List<double>>();
    for (int a = 0; a < weightClus.Count; a++)
    {
        List<double> anggota = new List<double>();
        ListWeightClus.Add(anggota);
    }
    for (int a = 0; a < ListWeightClus.Count; a++)
    {
        ListWeightClus[a].Clear();
    }
    for (int a = 0; a < weightClus.Count; a++)
    {
        for (int b = 0; b < weightClus.Count; b++)
        {
double ClusDistance = Math.Abs(weightClus[a] - weightClus[b]);
ListWeightClus[a].Add(ClusDistance);
        }
    }

    double minWeClus = ListWeightClus[0][1];
    for (int a = 0; a < ListWeightClus.Count; a++)
    {
        for (int b = 0; b < ListWeightClus[a].Count; b++)
        {
if (ListWeightClus[a][b] == 0)
```

```
    continue;
if (ListWeightClus[a][b] < minWeClus)
    minWeClus = ListWeightClus[a][b];
        }
    }
    int c1, c2;
    for (int a = 0; a < ListWeightClus.Count; a++)
    {
        if (ListWeightClus[a].Contains(minWeClus))
        {
for (int b = 0; b < ListWeightClus[a].Count; b++)
{
    if (ListWeightClus[a][b] == minWeClus)
    {
        c1 = a;
        c2 = b;
        cluster[c1].AddRange(cluster[c2]);
        cluster.RemoveAt(c2);
        break;
    }
}
break;
        }
    }

    for (int a = 0; a < cluster.Count; a++)
    {
        cluster[a].Sort();
    }
        }
```

Source code 4.15 penggabungan *cluster* akhir

Pertama-tama *list* 1 dimensi didefinisikan dengan nama weightClus untuk menyimpan rata-rata nilai *weight* tiap *cluster*. *List* weightClus nilainya dihapus karena adanya perulangan sehingga nilai weightClus tidak statis. Kemudian dilakukan perulangan untuk *list cluster*. Nilai wClus diset 0 untuk tiap perulangan. Kemudian dihitung nilai wClus dengan menjumlahkan semua nilai *weight* tiap titik yang ada pada *list cluster* kemudian hasil penjumlahan dibagi dengan banyaknya anggota tiap *cluster*. Kemudian nilai *weight* tersebut dimasukan ke dalam *list* weightClus. Kemudian didefinisikan *list* 2 dimensi dengan nama ListWeightClus untuk menyimpan hasil perbandingan jarak antar *weight* tiap *cluster*. Kemudian dilakukan perulangan untuk semua *weight* tiap *cluster* dan dilakukan perhitungan selisih *weight* tiap *cluster*.

Hasil selisih tiap *cluster* kemudian disimpan ke dalam *list* ListWeightClus. Kemudian didefinisikan variabel minWeClus.

Variabel tersebut merupakan nilai minimal dari *weight* tiap *cluster*. Nilai minWeClus diset awal dengan nilai awal ListWeightClus, kemudian nilai tersebut dibandingkan dengan semua nilai yang ada pada ListWeightClus. Jika ada yang lebih kecil dari nilai minWeClus maka nilai minWeClus sama dengan nilai tersebut.

Kemudian variabel c1 dan c2 didefinisikan untuk menyimpan nilai *cluster* yang akan digabung. Dilakukan perulangan untuk ListWeightClus, jika nilai minWeClus ditemukan maka nilai c1 sama dengan *index* kolom ListWeightClus dan nilai c2 sama dengan *index* baris ListWeightClus. Terakhir anggota *cluster* pada *index* c1 ditambahkan anggotanya sebanyak titik yang masuk pada *cluster* ke c2, kemudian cluster ke c2 dihapus anggotanya sehingga penggabungan *cluster* selesai dilakukan.

Tahap terakhir adalah setiap *cluster* di urutkan anggotanya dan proses ini dilakukan selama jumlah *cluster* yang terbentuk lebih dari jumlah kelas awal *dataset*.

4.2.15 Proses permutasi *cluster*

Tahap selanjutnya adalah permutasi *cluster*. Karena *cluster* yang terbentuk belum diketahui *cluster* berapa saja maka tiap *cluster* yang terbentuk akan dicocokan dengan *cluster* awal. Jika anggota *cluster* yang terbentuk memiliki jumlah anggota benar terbanyak maka *index cluster* yang terbentuk diangggap *index cluster* awal sehingga pada proses selanjutnya yaitu proses akurasi tiap *cluster* yang terbentuk akan dicocokan dengan *cluster* awal. Proses permutasi *cluster* ditunjukan pada *Source code* 4.16

```
int clusCount = cluster.Count;
List<List<double>> Permutasi = new List<List<double>>();
List<double> x = new List<double>();
List<double> y = new List<double>();
List<double> benar = new List<double>();
List<double> acc = new List<double>();
Permutasi.Add(x);
Permutasi.Add(y);
Permutasi.Add(benar);
Permutasi.Add(acc);

int p = 1;
for (int a = clusCount; a > 0; a--)
{
    p = p * a;
}
for (int b = 0; b < p; b++)
{
    for (int a = 0; a < clusCount; a++)
```

```
    {
        Permutasi[0].Add(a);
    }
}
List<List<double>> Pasangan = new List<List<double>>();
for (int a = 0; a < p; a++)
{
    List<double> anggota = new List<double>();
    Pasangan.Add(anggota);
}
for (int a = 0; a < clusCount; a++)
{
    Pasangan[0].Add(a);
}
Random rd = new Random();
int m;
bool sama;
for (int a = 1; a < Pasangan.Count; a++)
{
    sama = true;
    while (sama)
    {
        Pasangan[a].Clear();
        for (int b = 0; b < clusCount; b++)
        {
m = rd.Next(0, clusCount);
while (Pasangan[a].Contains(m))
{
    m = rd.Next(0, clusCount);
}
Pasangan[a].Add(m);
        }

        for (int b = 0; b < a; b++)
        {
for (int d = 0; d < Pasangan[0].Count; d++)
{
    if (Pasangan[a][d] != Pasangan[b][d])
    {
        sama = false;
        break;
    }
    else
        sama = true;
}
if (sama)
    break;
        }
    }
}
for (int a = 0; a < Pasangan.Count; a++)
{
    Permutasi[1].AddRange(Pasangan[a]);
}
```

Source code 4.16 permutasi *cluster*

Awalnya *list* 2 dimensi didefinisikan untuk menyimpan hasil permutasi *cluster*. Kolom pertama dan kolom kedua digunakan untuk menyimpan titik permutasi, kolom ketiga digunakan untuk menyimpan berapa data yang benar dan kolom keempat digunakan untuk menyimpan nilai akurasi tiap permutasi. Kemudian dihitung total p dimana p merupakan total kemungkinan permutasi yang dapat dibentuk. Setelah total p didapatkan maka dilakukan proses pencarian pasangan permutasi untuk setiap *cluster*. Sesuai dengan aturan permutasi maka pasangan *cluster* yang terbentuk tidak boleh sama dengan pasangan *cluster* yang lainnya. Setelah didapatkan *cluster* pasangan maka telah didapatkan tabel permutasi *cluster* dimana semua *cluster* awal telah memiliki pasangan untuk dicocokan akurasinya dengan *cluster* akhir.

4.2.16 Proses perhitungan akurasi

Setelah proses permutasi *cluster* selesai maka proses terakhir adalah proses perhitungan akurasi. Proses permutasi *cluster* ditunjukan pada *Source code* 4.17

```
int correct;
for (int a = 0; a < Permutasi[0].Count; a++)
{
    correct = 0;
    int n1 = Convert.ToInt32(Permutasi[0][a]);
    int n2 = Convert.ToInt32(Permutasi[1][a]);
    for (int b = 0; b < cluster[n2].Count; b++)
    {
        if (ClusterAwal[n1].Contains(cluster[n2][b]))
correct++;
    }
    Permutasi[2].Add(correct);
}
for (int a = 0; a < Permutasi[0].Count; a++)
{
    double akurasi = 0;
    double n1 = Permutasi[2][a];
    double n2 = ClusterAwal[Convert.ToInt32(
Permutasi[0][a])].Count;
    akurasi = n1 / n2;
    Permutasi[3].Add(akurasi);
}
List<double> ListAkurasi = new List<double>();
double q = 0;
for (int a = 0; a < Permutasi[0].Count; a++)
{
    double n1 = Permutasi[3][a];
    double n2 = Permutasi[0][a];
    q += n1;
```

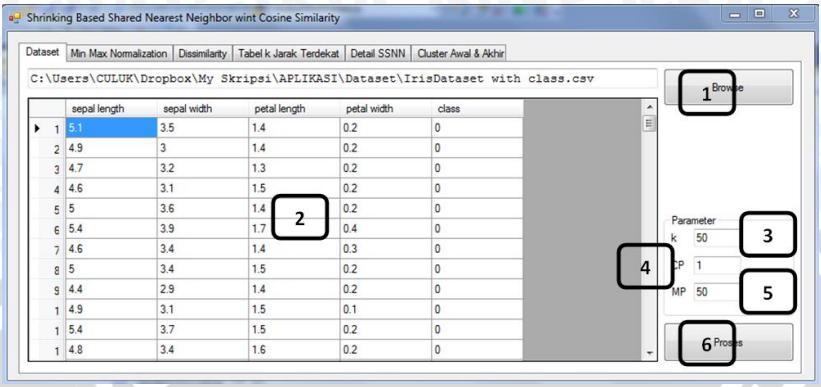
```
    if (n2 == (cluster.Count-1))
    {
        ListAkurasi.Add(q/cluster.Count);
        q = 0;
    }
}
for (int a = 0; a < ListAkurasi.Count; a++)
{
    listBox3.Items.Add(ListAkurasi[a]);
}

double maxAkurasi = ListAkurasi.Max();
listBox3.Items.Add("akurasi = "+maxAkurasi);
MessageBox.Show(maxAkurasi.ToString());
```

Source code 4.17 perhitungan akurasi

Pertama dilakukan pengecekan dari *list* permutasi untuk kolom pertama dan kolom kedua, kemudian dicocokan anggota dari *cluster* kolom kedua dengan *cluster* kolom pertama, jika anggota *cluster* pada kolom kedua sama dengan anggota *cluster* pada kolom kesatu maka nilai *correct* ditambahkan sebesar 1. Proses tersebut terus dilakukan sampai anggota kolom pertama dan kolom kedua di-*looping* semua. Setelah total data benar didapatkan maka akurasi dihitung dengan cara total data benar dibagi dengan jumlah data *cluster* awal. Kemudian dihitung akurasi rata-rata untuk setiap permutasi dan dipilih akurasi terbesar. Akurasi terbesar merupakan hasil akhir akurasi dan permutasi *cluster* dari akurasi terbesar itulah yang akan menjadi permutasi *cluster* akhir.

4.2.17 Implementasi Antar Muka

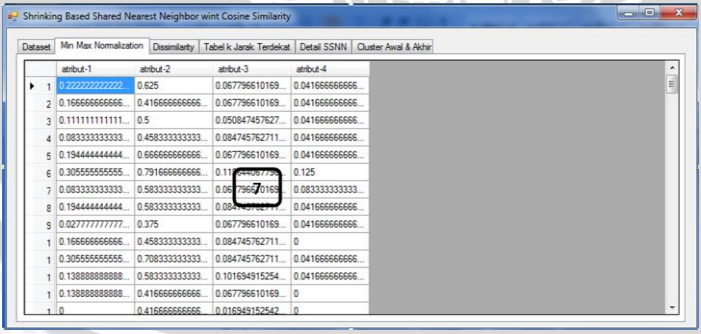
Pada aplikasi *Clustering Algoritma SSNN*, terdapat 6 tab control yaitu tab control *Dataset*, *Min-Max Normalization*, *Dissimilarity*, Tabel k Jarak Terdekat, *Detail SSNN* dan *Cluster* Awal & Akhir. Tab control tersebut akan ditunjukan pada Gambar 4.1, Gambar 4.2, Gambar 4.3, Gambar 4.4, Gambar 4.5 dan Gambar 4.6.



Gambar 4.1 *Tab control dataset*

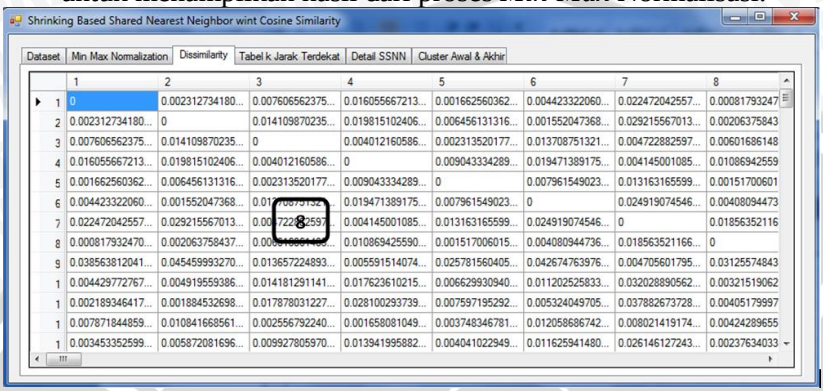
Keterangan :

1. *Browse* adalah tombol yang digunakan untuk memunculkan *browse box* dimana *user* akan memilih *dataset* yang akan digunakan.
2. *Data grid view dataset* adalah tempat untuk menampilkan *dataset* yang telah dipilih oleh *user*.
3. *Textbox* k adalah sebagai inputan nilai k pada proses Algoritma SSNN.
4. *Textbox* CP adalah sebagai inputan nilai CP pada proses algoritma SSNN.
5. *Textbox* MP adalah sebagai inputan nilai MP pada proses algoritma SSNN.
6. Proses adalah tombol untuk memulai proses perhitungan Algoritma SSNN.



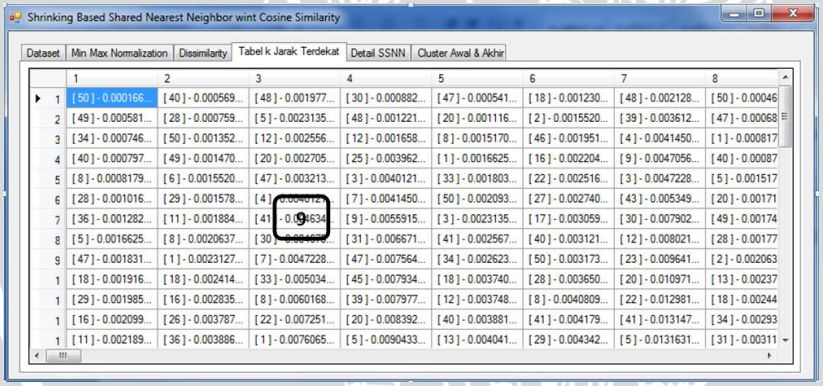
Gambar 4.2 *Tab control* Normalisasi *Min Max*

7. *Data gridview Min-Max Normalization* adalah *data gridview* untuk menampilkan hasil dari proses *Min-Max* Normalisasi.



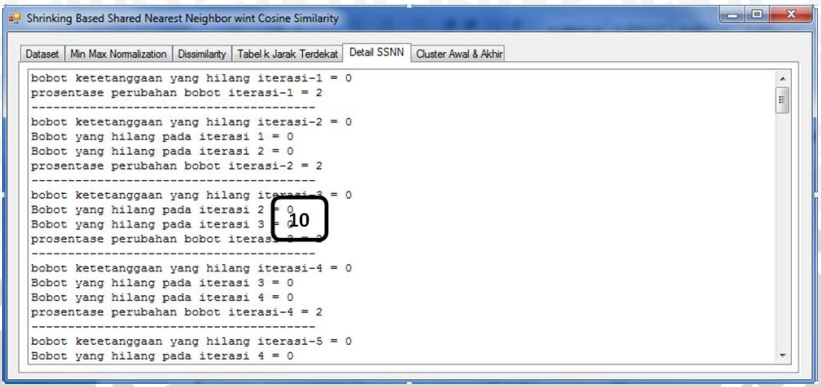
Gambar 4.3 *Tab control* Nilai *Dissimilarity*

8. *Data gridview Dissimilarity* adalah *data gridview* untuk menampilkan hasil perhitungan nilai *Dissimilarity* Algoritma SSNN.



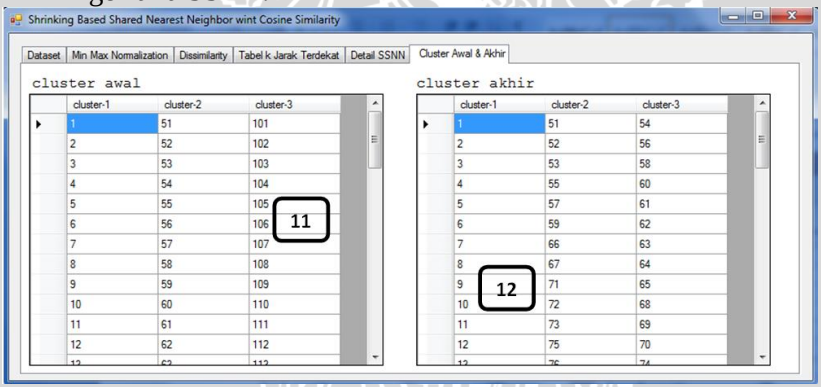
Gambar 4.4 *Tab control* Tabel k jarak terdekat

9. *Data gridview* k jarak terdekat adalah *data gridview* untuk menampilkan jarak terdekat beserta titiknya sejumlah k inputan.



Gambar 4.5 *Tab control Detail* SSNN

10. *Listbox detail* SSNN adalah *listbox* untuk menampilkan hasil dari perhitungan Algoritma SSNN diantaranya adalah total bobot ketetanggaan yang hilang tiap iterasi, prosentase perubahan bobot untuk tiap iterasi, total iterasi dan hasil akurasi Algoritma SSNN.



Gambar 4.6 *Tab control* anggota *cluster* awal dan *cluster* akhir

11. *Data gridview cluster* awal adalah *data gridview* untuk menampilkan anggota *cluster* awal.
12. *Data gridview cluster* akhir adalah *data gridview* untuk menampilkan anggota *cluster* akhir hasil *clustering* algoritma SSNN.

4.3 Implementasi pengujian

4.3.1 Hasil Uji

Hasil pengujian yang dilakukan terhadap 2 *dataset* yaitu *dataset iris* dan *dataset wine* akan dijelaskan pada subbab ini. Berdasarkan pengujian yang telah dilakukan, didapatkah hasil sebagai berikut.

4.3.1.1 Hasil uji k

Pada subbab hasil uji k akan dilakukan pengujian pengaruh nilai k terhadap nilai akurasi. Uji k dilakukan dengan cara mengubah nilai k dari 1 sampai nilai k maksimal. Untuk parameter CP dan MP di-*set* dengan nilai 2 untuk CP dan nilai 100 untuk MP. Pemilihan nilai CP sama dengan 2 adalah karena nilai *dissimilarity* untuk kedua *dataset* adalah maksimal 2 dan nilai MP sama dengan 100 karena nilai maksimal dari MP adalah 100%. Pengujian k dilakukan untuk kedua dataset yaitu *iris* dan *wine* dimana masing-masing dataset diuji dengan menggunakan rumus jarak *cosine* dan *euclidean.*

a. Uji k menggunakan *cosine*

Tabel 4.1 Akurasi uji k *dataset iris* menggunakan *cosine*

K	CP	MP	AKURASI
1	2	100	0,753
2	2	100	0,74
3	2	100	0,74
4	2	100	0,74
5	2	100	0,74
....	....	....	....
145	2	100	0,74
146	2	100	0,74
147	2	100	0,74
148	2	100	0,74
149	2	100	0,74
Rata-Rata			0,74

Tabel 4.1 merupakan hasil pengujian k dengan menggunakan *cosine* untuk *dataset iris*. Dari hasil pengujian tersebut dapat diketahui bahwa nilai CP dan MP di-*set* konstan yaitu nilai CP adalah 2 dan nilai MP adalah 100. Akurasi tertinggi dicapai pada saat

k bernilai 1 dengan nilai akurasi adalah 0,753. Sedangkan nilai akurasi terendah dicapai pada saat k bernilai 2 sampai dengan 149 dengan akurasinya adalah 0,74. Rata-rata nilai akurasi untuk uji k dengan menggunakan *cosine* untuk *dataset iris* adalah 0,74.

Tabel 4.2 Akurasi uji k *dataset wine* menggunakan *cosine*

K	CP	MP	AKURASI
1	2	100	0,516
2	2	100	0,516
3	2	100	0,516
4	2	100	0,516
5	2	100	0,516
....	....	....	....
174	2	100	0,516
175	2	100	0,516
176	2	100	0,516
177	2	100	0,516
RATA-RATA			0,516

Tabel 4.2 merupakan hasil pengujian k dengan menggunakan *cosine* untuk *dataset wine*. Dari hasil pengujian tersebut dapat diketahui bahwa nilai CP dan MP di-*set* konstan yaitu nilai CP adalah 2 dan nilai MP adalah 100. Nilai akurasi konstan mulai dari k sama dengan 1 sampai dengan k sama dengan 177 dengan nilai akurasinya adalah 0,516. Rata-rata nilai akurasi untuk uji k dengan menggunakan *cosine* untuk *dataset wine* adalah 0,516.

b. Uji k menggunakan *euclidean*

Tabel 4.3 Akurasi uj k *dataset iris* menggunakan *euclidean*

K	CP	MP	AKURASI
1	2	100	0,693
2	2	100	0,733
3	2	100	0,74
4	2	100	0,746
5	2	100	0,746
....	....	....	....
147	2	100	0,746
148	2	100	0,746
149	2	100	0,746
RATA-RATA			0,746

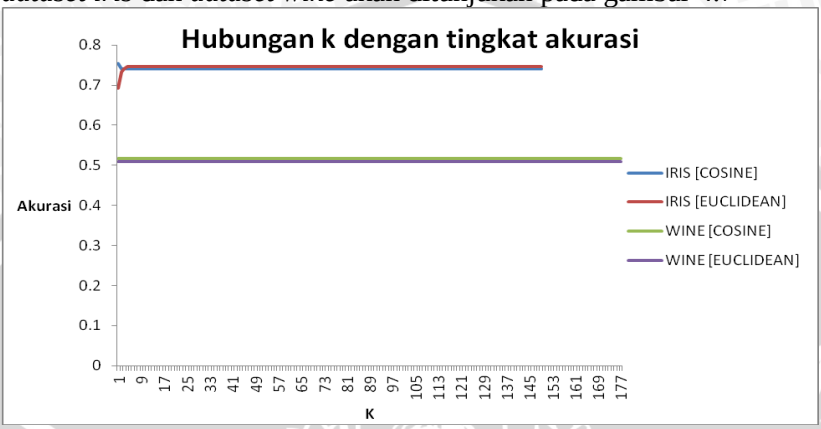
Tabel 4.3 merupakan hasil pengujian k dengan menggunakan *euclidean* untuk *dataset iris*. Dari hasil pengujian tersebut dapat diketahui bahwa nilai CP dan MP di-*set* konstan yaitu nilai CP adalah 2 dan nilai MP adalah 100. Nilai akurasi terbesar dicapai pada saat k bernilai 4 sampai dengan k bernilai 149 dengan nilai akurasinya adalah 0,746. Sedangkan nilai akurasi terkecil dicapai pada saat k bernilai 1 dengan nilai akurasi adalah 0,693. Rata-rata akurasi untuk uji k dengan menggunakan *euclidean* untuk *dataset iris* adalah 0,746.

Tabel 4.4 Akurasi uji k *dataset wine* menggunakan *euclidean*

K	CP	MP	AKURASI
1	2	100	0,509
2	2	100	0,509
3	2	100	0,509
4	2	100	0,509
5	2	100	0,509
6	2	100	0,509
7	2	100	0,509
....	....	....	....
176	2	100	0,509
177	2	100	0,509
RATA-RATA			0,509

Tabel 4.4 merupakan hasil pengujian k dengan menggunakan *euclidean* untuk *dataset wine*. Dari hasil pengujian tersebut dapat diketahui bahwa nilai CP dan MP di-*set* konstan yaitu nilai CP adalah 1 dan nilai MP adalah 100. Nilai akurasi untuk uji k dengan menggunakan *euclidean* selalu konstan mulai dari k sama dengan 1 sampai dengan k sama dengan 177 dengan nilai akurasinya adalah 0,509. Rata-rata nilai akurasi untuk uji k dengan menggunakan *euclidean* untuk *dataset wine* adalah 0,509.

Grafik hubungan nilai k dengan tingkat akurasi untuk *dataset iris* dan *dataset wine* akan ditunjukan pada gambar 4.7



Gambar 4.7 Gambar hubungan k dengan tingkat akurasi

4.3.1.2 Hasil uji CP

Pada subbab hasil uji CP akan dilakukan pengujian pengaruh CP terhadap nilai akurasi. Uji CP dilakukan dengan cara mengubah nilai CP mulai dari 0,01 sampai nilai CP maksimal yaitu 2. Untuk parameter k dan MP di-*set* dengan nilai maksimal yaitu nilai 149 untuk k *dataset iris*, nilai 177 untuk k *dataset wine* dan nilai 100 untuk MP. Pengujian CP dilakukan untuk kedua dataset yaitu *iris* dan *wine* dimana masing-masing dataset diuji dengan menggunakan rumus jarak *cosine* dan *euclidean*.

a. Uji CP menggunakan *cosine*

Tabel 4.5 Akurasi uji CP *dataset iris* menggunakan *cosine*

K	CP	MP	AKURASI
149	0,01	100	0,753
149	0,02	100	0,733
....	....	....	....
149	0,04	100	0,733
149	0,05	100	0,74
....	....	....	....
149	2	100	0,74
RATA-RATA			0,739

Tabel 4.5 merupakan hasil pengujian CP dengan menggunakan *cosine* untuk *dataset iris*. Dari hasil pengujian tersebut dapat diketahui bahwa nilai k dan MP di-*set* konstan yaitu nilai k adalah 149 dan nilai MP adalah 100. Nilai akurasi terbesar untuk uji CP *dataset iris* dengan menggunakan *cosine* dicapai pada saat nilai CP sama dengan 0,01 dengan nilai akurasinya adalah 0,753. Sedangkan nilai akurasi terkecil dicapai pada saat nilai CP adalah 0,02 sampai dengan 0,04 dengan nilai akurasinya adalah 0,733. Rata-rata akurasi untuk uji CP dengan menggunakan *cosine* untuk *dataset iris* adalah 0,739.

Tabel 4.6 Akurasi uji CP *dataset wine* menggunakan *cosine*

K	CP	MP	AKURASI
177	0,01	100	0,087
177	0,02	100	0,327
177	0,03	100	0,5831
177	0,04	100	0,608
177	0,05	100	0,615
177	0,06	100	0,504
177	0,07	100	0,511
....	....	....	....
177	0,11	100	0,511
177	0,12	100	0,516
....	....	....	....
177	2	100	0,516
RATA-RATA			0,512

Tabel 4.6 merupakan hasil pengujian CP dengan menggunakan *cosine* untuk *dataset wine*. Dari hasil pengujian tersebut dapat diketahui bahwa nilai k dan MP di-*set* konstan yaitu nilai k adalah 177 dan nilai MP adalah 100. Nilai akurasi terbesar untuk uji CP *dataset wine* dengan menggunakan *cosine* dicapai pada saat nilai CP sama dengan 0,05 dengan nilai akurasinya adalah 0,615. Sedangkan nilai akurasi terkecil dicapai pada saat nilai CP adalah 0,01 dengan nilai akurasinya adalah 0,087. Rata-rata akurasi untuk uji CP dengan menggunakan *cosine* untuk *dataset iris* adalah 0,512.

b. Uji CP menggunakan *euclidean*

Tabel 4.7 Akurasi uji CP *dataset iris* menggunakan *euclidean*

K	CP	MP	AKURASI
149	0,01	100	0
....	....	....	....
149	0,03	100	0
149	0,04	100	0,066
149	0,05	100	0,18
149	0,06	100	0,313
149	0,07	100	0,393
149	0,08	100	0,466
149	0,09	100	0,586
149	0,1	100	0,586
149	0,11	100	0,646
149	0,12	100	0,666
149	0,13	100	0,746
149	0,14	100	0,766
149	0,15	100	0,633
149	0,16	100	0,726
....	....	....	....
149	0,19	100	0,726
149	0,2	100	0,646
....	....	....	....
149	0,25	100	0,646
149	0,26	100	0,746
....	....	....	....
149	2	100	0,746
RATA-RATA			0,688

Tabel 4.7 merupakan hasil pengujian CP dengan menggunakan *euclidean* untuk *dataset iris*. Dari hasil pengujian tersebut dapat diketahui bahwa nilai k dan MP di-*set* konstan yaitu nilai k adalah 149 dan nilai MP adalah 100. Nilai akurasi terbesar untuk uji CP *dataset iris* dengan menggunakan *euclidean* dicapai pada saat nilai CP sama dengan 0,14 dengan nilai akurasinya adalah 0,766. Sedangkan nilai akurasi terkecil dicapai pada saat nilai CP adalah 0,01 sampai dengan 0,03 dengan nilai akurasinya adalah 0.

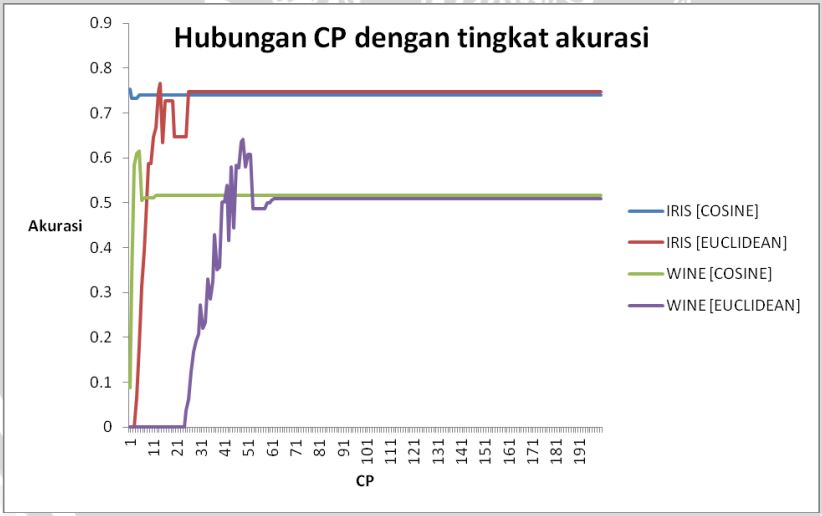
Rata-rata akurasi untuk uji CP dengan menggunakan *euclidean* untuk *dataset iris* adalah 0,688.

Tabel 4.8 Akurasi uji CP *dataset wine* menggunakan *euclidean*

K	CP	MP	AKURASI
177	0,01	100	0
....	....	....	....
177	0,24	100	0
177	0,25	100	0,036
177	0,26	100	0,061
177	0,27	100	0,123
177	0,28	100	0,167
177	0,29	100	0,191
177	0,3	100	0,206
177	0,31	100	0,272
177	0,32	100	0,219
177	0,33	100	0,232
177	0,34	100	0,330
177	0,35	100	0,285
177	0,36	100	0,324
177	0,37	100	0,429
177	0,38	100	0,349
177	0,39	100	0,356
177	0,4	100	0,5
177	0,41	100	0,5
177	0,42	100	0,539
177	0,43	100	0,416
177	0,44	100	0,579
177	0,45	100	0,442
177	0,46	100	0,582
177	0,47	100	0,577
177	0,48	100	0,636
177	0,49	100	0,640
177	0,5	100	0,579
177	0,51	100	0,608
177	0,52	100	0,608
177	0,53	100	0,487
....	....	....	....
177	0,58	100	0,487

177	0,59	100	0,5
177	0,6	100	0,5
177	0,61	100	0,504
177	0,62	100	0,509
....	....	....	....
177	2	100	0,509
RATA-RATA			0,351

Tabel 4.8 merupakan hasil pengujian CP dengan menggunakan *euclidean* untuk *dataset wine*. Dari hasil pengujian tersebut dapat diketahui bahwa nilai k dan MP di-*set* konstan yaitu nilai k adalah 177 dan nilai MP adalah 100. Nilai akurasi terbesar untuk uji CP *dataset wine* dengan menggunakan *euclidean* dicapai pada saat nilai CP sama dengan 0,49 dengan nilai akurasinya adalah 0,64. Sedangkan nilai akurasi terkecil dicapai pada saat nilai CP adalah 0,01 sampai dengan 0,24 dengan nilai akurasinya adalah 0. Rata-rata akurasi untuk uji CP dengan menggunakan *euclidean* untuk *dataset wine* adalah 0,351. Grafik hubungan nilai CP dengan tingkat akurasi untuk *dataset iris* dan *dataset wine* akan ditunjukan pada gambar 4.8



Gambar 4.8 Gambar hubungan CP dengan tingkat akurasi

4.3.1.3 Hasil uji MP

Pada subbab hasil uji MP akan dilakukan pengujian pengaruh MP terhadap nilai akurasi. Uji MP dilakukan dengan cara mengubah nilai MP dari 1 sampai nilai MP maksimal yaitu 100. Untuk parameter k dan CP di-*set* dengan nilai maksimal yaitu nilai 149 untuk k *dataset iris*, nilai 177 untuk k *dataset wine* dan nilai 1 untuk CP. Pengujian MP dilakukan untuk kedua dataset yaitu *iris* dan *wine* dimana masing-masing dataset diuji dengan menggunakan rumus jarak *cosine* dan *euclidean.*

a. Uji MP menggunakan *cosine*

Tabel 4.9 Akurasi uji MP *dataset iris* menggunakan *cosine*

K	CP	MP	AKURASI
149	2	1	0,74
149	2	2	0,74
....	....	....	....
149	2	99	0,74
149	2	100	0,74
RATA-RATA			0,74

Tabel 4.9 merupakan hasil pengujian MP dengan menggunakan *cosine* untuk *dataset iris*. Dari hasil pengujian tersebut dapat diketahui bahwa nilai k dan CP di-*set* konstan yaitu nilai k adalah 149 dan nilai CP adalah 2. Nilai akurasi hasil uji MP dengan menggunakan *cosine* untuk *dataset iris* selalu konstan milai dari MP sama dengan 1 sampai dengan MP sama dengan 100.

Tabel 4.10 Akurasi uji MP *dataset wine* menggunakan *cosine*

K	CP	MP	AKURASI
177	2	1	0
177	2	2	0,516
177	2	3	0,516
....	....	....	....
177	2	99	0,516
177	2	100	0,516
RATA-RATA			0,511

Tabel 4.10 merupakan hasil pengujian MP dengan menggunakan *cosine* untuk *dataset wine*. Dari hasil pengujian tersebut dapat diketahui bahwa nilai k dan CP di-*set* konstan yaitu nilai k adalah 177 dan nilai CP adalah 1. Nilai akurasi terbesar untuk uji MP *dataset wine* dengan menggunakan *cosine* dicapai pada saat nilai MP sama dengan 1 sampai dengan MP bernilai 100 dengan nilai akurasinya adalah 0,516. Sedangkan nilai akurasi terkecil dicapai pada saat nilai MP adalah 1 dengan nilai akurasinya adalah 0. Rata-rata akurasi untuk uji MP dengan menggunakan *cosine* untuk *dataset wine* adalah 0,511.

b. Uji MP menggunakan *euclidean*

Tabel 4.11 Akurasi uji MP *dataset iris* menggunakan *euclidean*

K	CP	MP	AKURASI
149	2	1	0,746
149	2	2	0,746
....	....	....	....
149	2	99	0,746
149	2	100	0,746
RATA-RATA			0,746

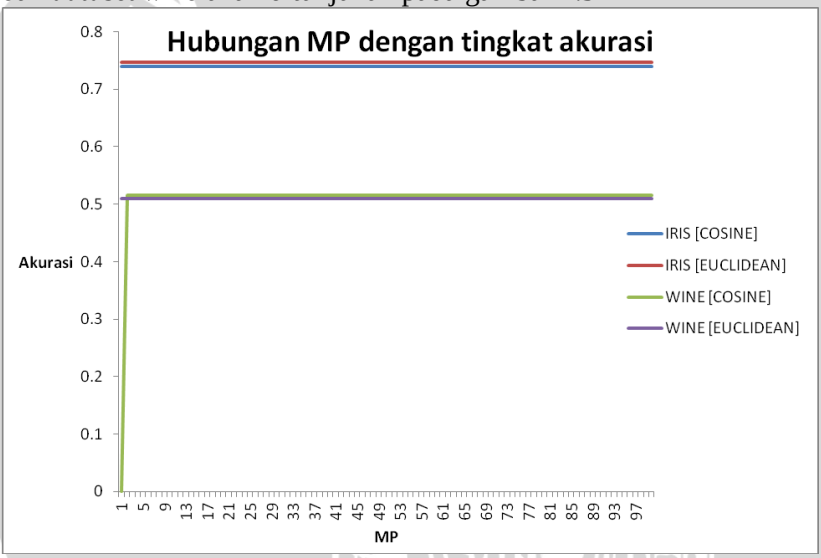
Tabel 4.11 merupakan hasil pengujian MP dengan menggunakan *euclidean* untuk *dataset iris*. Dari hasil pengujian tersebut dapat diketahui bahwa nilai k dan CP di-*set* konstan yaitu nilai k adalah 149 dan nilai CP adalah 1. Nilai akurasi untuk uji MP *dataset iris* dengan menggunakan *euclidean* selalu konstan mulai dari MP sama dengan 1 sampai dengan MP sama dengan 100 dengan nilai akurasinya adalah 0,746. Rata-rata nilai akurasi untuk uji MP dengan menggunakan *euclidean* untuk *dataset iris* adalah 0,746.

Tabel 4.12 Akurasi uji MP *dataset wine* menggunakan *euclidean*

K	CP	MP	AKURASI
177	2	1	0,509
177	2	2	0,509
....	....	....	....
177	2	99	0,509
177	2	100	0,509
RATA-RATA			0,509

Tabel 4.12 merupakan hasil pengujian MP dengan menggunakan *euclidean* untuk *dataset wine*. Dari hasil pengujian tersebut dapat diketahui bahwa nilai k dan CP di-*set* konstan yaitu nilai k adalah 177 dan nilai CP adalah 1. Nilai akurasi untuk uji MP *dataset wine* dengan menggunakan *euclidean* selalu konstan mulai dari MP sama dengan 1 sampai dengan MP sama dengan 100 dengan nilai akurasinya adalah 0,509. Rata-rata nilai akurasi untuk uji MP dengan menggunakan *euclidean* untuk *dataset wine* adalah 0,74.

Grafik hubungan nilai MP dengan tingkat akurasi untuk *dataset iris* dan *dataset wine* akan ditunjukan pada gambar 4.9



Gambar 4.9 Gambar hubungan MP dengan tingkat akurasi

4.3.2 Analisa Hasil

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

4.3.2.1 Analisa hasil uji k

Nilai akurasi untuk kedua *dataset* cenderung tetap walaupun nilai k dirubah. Untuk *dataset iris* nilai akurasinya sedikit lebih besar dari pada *dataset wine. Dataset iris* nilai akurasinya sedikit lebih baik jika menggunakan *euclidean* sebagai rumus untuk mencari jarak

dari pada menggunakan *cosine* sebagai rumus pencari jaraknya. Sedangkan untuk *dataset wine* nilai akurasi sedikit lebih baik jika menggunakan *cosine* sebagai rumus untuk mencari jarak dari pada menggunakan *euclidean*.

Berikut merupakan contoh tabel jarak terdekat dari *dataset iris* menggunakan *cosine* dengan k = 100, CP = 1 dan MP = 100% akan ditunjukan oleh tabel 4.13, k = 70, CP = 1 dan MP = 100% akan ditunjukan oleh tabel 4.14 dan k = 40, CP = 1 dan MP = 100% akan ditunjukan oleh tabel 4.15.

Tabel 4.13 Contoh tabel jarak terdekat dengan k = 100

Record	Jarak	Data Pertama	Data Kedua
1	7,39E-05	10	28
2	0,000113	143	147
3	0,000135	26	45
724	0,004013	54	107
725	0,004016	70	100
726	0,004017	66	99
1544	0,007793	46	47
1545	0,007794	128	139
1546	0,007797	64	120
1547	0,007798	21	30

Tabel 4.14 Contoh tabel jarak terdekat dengan k = 70

Record	Jarak	Data Pertama	Data Kedua
1	7,39E-05	10	28
2	0,000113	143	147
3	0,000135	26	45
724	0,004013	54	107
725	0,004016	70	100
726	0,004017	66	99
1544	0,007793	46	47
1545	0,007794	128	139
1546	0,007797	64	120
1547	0,007798	21	30

Tabel 4.15 Contoh tabel jarak terdekat dengan k = 40

Record	Jarak	Data Pertama	Data Kedua
1	7,39E-05	10	28
2	0,000113	143	147
3	0,000135	26	45
724	0,004013	54	107
725	0,004016	70	100
726	0,004017	66	99
1544	0,007793	46	47
1545	0,007794	128	139
1546	0,007797	64	120
1547	0,007798	21	30

Dari tabel tersebut dapat diketahui bahwa berapapun nilai k yang dimasukan ke dalam sistem, tabel jarak terdekat yang terbentuk adalah sama. Dikarenakan tabel jarak terdekat yang terbentuk adalah sama maka akurasi yang dihasilkan dari ketiga nilai k variasi adalah sama.

Dengan nilai k yang selalu dirubah mulai dari 1 sampai dengan jumlah k maksimal tetapi tidak berpengaruh terhadap pembentukan tabel jarak terdekat, maka dapat disimpulkan bahwa nilai k tidak mempengaruhi hasil akurasi. Hal ini disebabkan berapapun banyak nilai k yang diambil, tabel jarak terdekat yang terbentuk adalah sama. Jadi bisa dikatakan semakin banyak nilai k maka akan semakin memperbanyak proses perhitungan dan tidak mempengaruhi nilai akurasi.

4.3.2.2 Analisa hasil uji CP

Dari gambar 4.8 dapat diketahui bahwa akurasi untuk *dataset iris* menggunakan *cosine* mengalami penurunan yang tidak terlalu signifikan dan akurasi mulai konstan pada saat nilai CP adalah 0,5. Untuk akurasi *dataset iris* menggunakan *euclidean*, akurasi *dataset wine* menggunakan *cosine* dan akurasi *dataset wine* menggunakan *euclidean* mengalami kenaikan dan penurunan yang signifikan, tetapi ketiga uji coba tersebut akan mencapai akurasi konstan pada saat CP tertentu. Hal tersebut dikarenakan semakin besar nilai CP maka akan semakin banyak titik yang masuk ke dalam proses perhitungan Algoritma *Shrinking based Shared Nearest*

Neighbor. Berikut merupakan contoh daftar titik yang masuk perhitungan dengan variasi nilai CP dan dengan mengunakan sampel *dataset wine*. Untuk parameter nilai k adalah 10 dan nilai MP adalah 100%.

Tabel 4.16 Daftar titik yang masuk perhitungan dengan CP = 0,4

Data														
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0.07	0.07	0.07	0.07	0.1	0.21	0.1	0.1	0.17	0.18	0.12	0.02	0.04	0.02	0.12
0.12	0.12	0.1	0.12	0.12	0.29	0.2	0.12	0.17	0.19	0.15	0.04	0.08	0.08	0.22
0.13	0.13	0.12	0.12	0.13	0.38	0.21	0.16	0.18	0.2	0.18	0.14	0.18	0.16	0.27
0.14	0.14	0.12	0.14	0.13	0.4	0.22	0.18	0.2	0.2	0.19	0.15	0.2	0.19	0.28
0.22	0.17	0.18	0.23	0.14	0.42	0.24	0.27	0.2	0.27	0.22	0.27	0.27	0.28	0.33
0.23	0.2	0.24	0.24	0.14	0.52	0.32	0.27	0.23	0.29	0.23	0.27	0.27	0.31	0.33
0.29	0.28	0.27	0.27	0.16	0.52	0.35	0.28	0.24	0.29	0.28	0.29	0.31	0.34	0.38
0.3	0.31	0.28	0.29	0.16	0.54	0.37	0.28	0.24	0.35	0.28	0.33	0.34	0.36	0.38
0.33	0.32	0.28	0.31	0.17	0.61	0.4	0.28	0.29	0.38	0.31	0.33	0.39	0.37	0.41
0.36	0.36	0.28	0.31	0.19	0.65	0.4	0.29	0.33	0.43	0.33	0.36	0.4	0.38	0.42

Tabel 4.17 Daftar titik yang masuk perhitungan dengan CP = 0,3

Data														
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0.07	0.07	0.07	0.07	0.1	0.21	0.1	0.1	0.17	0.18	0.12	0.02	0.04	0.02	0.12
0.12	0.12	0.1	0.12	0.12	0.29	0.2	0.12	0.17	0.19	0.15	0.04	0.08	0.08	0.22
0.13	0.13	0.12	0.12	0.13	0.38	0.21	0.16	0.18	0.2	0.18	0.14	0.18	0.16	0.27
0.14	0.14	0.12	0.14	0.13	0.4	0.22	0.18	0.2	0.2	0.19	0.15	0.2	0.19	0.28
0.22	0.17	0.18	0.23	0.14	0.42	0.24	0.27	0.2	0.27	0.22	0.27	0.27	0.28	0.33
0.23	0.2	0.24	0.24	0.14	0.52	0.32	0.27	0.23	0.29	0.23	0.27	0.27	0.31	0.33
0.29	0.28	0.27	0.27	0.16	0.52	0.35	0.28	0.24	0.29	0.28	0.29	0.31	0.34	0.38
0.3	0.31	0.28	0.29	0.16	0.54	0.37	0.28	0.24	0.35	0.28	0.33	0.34	0.36	0.38
0.33	0.32	0.28	0.31	0.17	0.61	0.4	0.28	0.29	0.38	0.31	0.33	0.39	0.37	0.41
0.36	0.36	0.28	0.31	0.19	0.65	0.4	0.29	0.33	0.43	0.33	0.36	0.4	0.38	0.42

Dari tabel 4.16 dan tabel 4.17 dapat dibuktikan bahwa semakin kecil nilai CP maka banyaknya titik yang masuk ke dalam perhitungan algoritma SSNN akan semakin sedikit sehingga tabel jarak terdekat yang terbentuk akan semakin sedikit pula sehingga akan mempengaruhi hasil akurasi. Pada tabel 4.16 terdapat 13 titik jarak yang diputuskan dan tidak dimasukan ke dalam perhitungan dengan total bobot yang diputuskan adalah 6,12 dan besar akurasi adalah 0,5333 sedangkan pada tabel 4.17 terdapat 45 titik jarak yang diputuskan dan tidak dimasukan ke dalam perhitungan dengan total bobot yang diputuskan adalah 17,08 dan besar akurasi adalah 0,4. Dikarenakan banyaknya titik yang masuk ke dalam perhitungan berbeda maka tabel jarak terdekat yang terbentuk pasti berbeda dan karena tabel jarak terdekat yang terbentuk berbeda maka akurasinya juga pasti berbeda. Sehingga dapat disimpulkan nilai CP mempengaruhi tingkat akurasi. Semakin banyak CP tidak berarti semakin baik nilai akurasi dikarenakan bisa saja terjadi masuknya suatu titik ke dalam perhitungan dimana titik tersebut dapat menyebabkan titik lain masuk ke dalam *cluster* yang salah.

4.3.2.3 Analisa hasil uji MP

Untuk *dataset iris*, nilai akurasi dengan menggunakan *cosine* sedikit lebih kecil dari nilai akurasi dengan menggunakan *euclidean*. Sedangkan untuk *dataset wine* nilai akurasi dengan menggunakan *cosine* sedikit lebih besar dari pada nilai akurasi dengan menggunakan *euclidean*.

Berikut merupakan detail iterasi, bobot ketetanggaan yang hilang dan prosentase perubahan bobot pada uji MP *dataset iris* dengan nilai k adalah 149, nilai CP adalah 1 dan nilai MP adalah 90, 50 dan 20.

Tabel 4.18 Detail SSNN dengan nilai MP adalah 90

iterasi	bobot ketetanggaan yang hilang	prosentase perubahan bobot
1	0	2
2	0	2
3	194,243,096	2
4	0	2
5	0	2
6	0	2

7	0	2
8	1,841,944,102	2
9	0	2
10	0	2
11	356,440,787	2
12	0	2
13	0	2
14	173,793,301	2
15	171,824,409	0,0113289

Tabel 4.19 Detail SSNN dengan nilai MP adalah 50

iterasi	bobot ketetanggaan yang hilang	prosentase perubahan bobot
1	0	2
2	0	2
3	194,243,096	2
4	0	2
5	0	2
6	0	2
7	0	2
8	1,841,944,102	2
9	0	2
10	0	2
11	356,440,787	2
12	0	2
13	0	2
14	173,793,301	2
15	171,824,409	0,0113289

Tabel 4.20 Detail SSNN dengan nilai MP adalah 20

iterasi	bobot ketetanggaan yang hilang	prosentase perubahan bobot
1	0	2
2	0	2
3	194,243,096	2
4	0	2
5	0	2
6	0	2
7	0	2
8	1,841,944,102	2
9	0	2
10	0	2

11	356,440,787	2
12	0	2
13	0	2
14	173,793,301	2
15	171,824,409	0,0113289

Dari tabel 4.18, 4.19 dan 4.20 dapat dilihat bahwa berapapun nilai MP yang dimasukan, banyak iterasi, bobot ketetanggaan yang hilang dan prosentase perubahan bobotnya adalah sama. Dikarenakan iterasi yang dijalankan adalah sama untuk setiap uji MP maka tabel jarak terdekat yang terbentuk adalah sama. Karena tabel jarak terdekat yang terbentuk sama maka akurasi yang dihasilkan juga sama dan dapat disimpulkan bahwa nilai MP tidak mempengaruhi akurasi. Tetapi jika nilai MP terlalu kecil maka akan menyebabkan iterasi tidak berhenti. Hal itu dikarenakan nilai prosestase perubahan bobot yang dihasilkan tiap iterasi tidak ada yang lebih kecil dari nilai MP dan hal itu menyebabkan tidak didapatnya nilai akurasi.

4.3.2.4 Analisa hasil uji secara keseluruhan

Setelah ketiga uji untuk kedua *dataset* dilakukan, maka didapatkan nilai akurasi maksimal untuk kedua *dataset.* Untuk *dataset iris*, nilai akurasi maksimal adalah 0,7667 dengan nilai k = 149, CP = 0,14, MP = 100% dan menggunakan *euclidean* sebagai fungsi pencari jaraknya. Untuk *dataset wine*, nilai akurasi maksimal adalah 0,6409 dengan nilai k = 177, CP = 0,49, MP = 100% dan menggunakan *euclidean* sebagai fungsi pencari jaraknya.

Setelah didapatkan hasil uji untuk kedua *dataset*, maka dari hasil tersebut kemudian dicari rata-rata untuk semua uji dan untuk setiap *dataset*. Berikut merupakan hasil uji akhir untuk *dataset iris* dan *dataset wine.*

Tabel 4.21 Rata-rata akurasi hasil uji *dataset iris* dan *dataset wine*

	COSINE			**EUCLIDEAN**		
	UJI K	**UJI CP**	**UJI MP**	**UJI K**	**UJI CP**	**UJI MP**
DATASET IRIS	**0,74**	**0,7399**	**0,74**	**0,746**	**0,6884**	**0,7467**
DATASET WINE	**0,516**	**0,5121**	**0,5109**	**0,51**	**0,351**	**0,5095**

Tabel 4.22 Hasil akhir akurasi *dataset iris* dan *dataset wine*

	COSINE	EUCLIDEAN
DATASET WINE	**0,513**	**0,456**
DATASET IRIS	**0,74**	**0,727**

Dari tabel 4.21 dapat diketahui bahwa nilai akurasi terbesar adalah pada saat uji MP untuk *dataset iris* dengan menggunakan rumus jarak *euclidean* dengan akurasinya adalah 0,7467. Sedangkan akurasi terkecil adalah pada saat uji CP untuk *dataset wine* dengan menggunakan rumus jarak *euclidean* dengan akurasinya adalah 0,351. Pada tabel 4.22 dapat diketahui bahwa hasil akhir akurasi *dataset wine* lebih besar jika menggunakan *cosine* untuk pencarian jarak antar data dari pada menggunakan *euclidean* dengan selisih akurasinya adalah 0,057. Begitu juga dengan *dataset iris,* hasil akhir akurasi *dataset iris* lebih besar jika menggunakan *cosine* untuk pencarian jarak antar data dari pada menggunakan *euclidean* dengan selisih akurasinya adalah 0,013. Sehingga dapat disimpulkan bahwa akurasi Algoritma *Shrinking based Shared Nearest Neighbor* dengan menggunakan *cosine* sedikit lebih baik dari pada menggunakan *euclidean* sebagai rumus untuk mencari jarak antar data.

Hal itu dikarenakan rumus *cosine* tidak dipengaruhi oleh variable-variabel yang nilainya sangat besar sehingga nilai *similarity* yang dihasilkan memiliki batas antara 0 sampai dengan 1. Sedangkan untuk rumus *euclidean* nilai-nilai *similarity* yang yang dihasilkan tidak memiliki batas (mulai dari 0 sampai tak terbatas)(Wirdati, 2007).

BAB V
PENUTUP

5.1 Kesimpulan

Kesimpulan yang diperoleh dari penelitian ini adalah sebagai berikut :

1. Langkah-langkah penerapan *cosine similarity* pada algoritma *Shrinking based Shared Nearest Neighbor* yaitu normalisasi *dataset* dengan menggunakan metode *min-max normalization*, kemudian hitung nilai *dissimilarity* dengan menggunakan rumus *cosine similarity*, bentuk tabel jarak terdekat, urutkan tabel jarak terdekat dari nilai jarak terdekat sampai jarak terjauh, parameter k, CP dan MP dimasukan ke dalam sistem, ambil k tabel jarak terdekat, putuskan nilai *dissimilarity* k tabel jarak terdekat yang lebih besar dari CP, pemutusan nilai *dissimilarity* dilakukan dalam beberapa iterasi dengan nilai CP terus diturunkan sebesar 0,01, hitung prosestase perubahan bobot untuk tiap iterasi, iterasi dihentikan jika prosentase perubahan bobot kurang dari nilai MP, bentuk *cluster* dari hasil perhitungan algoritma *Shrinking based Shared nearest neighbor,* dan yang terakhir adalah hitung akurasi dari *cluster* yang telah terbentuk.
2. Akurasi yang dihasilkan dari algoritma *Shrinking based Shared nearest neighbor* dengan menggunakan *cosine* sedikit lebih baik dari pada algoritma *Shrinking based Shared nearest neighbor* dengan menggunakan *euclidean.* Hasil akurasi algoritma *Shrinking based Shared nearest neighbor* untuk *dataset wine* dengan menggunakan *cosine* adalah 0,513 sedangkan hasil akurasi dengan menggunakan *euclidean* adalah 0,456. Hasil akurasi algoritma *Shrinking based Shared nearest neighbor* untuk *dataset iris* dengan menggunakan *cosine* adalah 0,74 sedangkan hasil akurasi dengan menggunakan *euclidean* adalah 0,727.

5.2 Saran

Dalam penelitian ini terdapat beberapa hal yang masih belum dieksplorasi secara mendalam dan dapat dijadikan bahan dalam penelitian lebih lanjut, antara lain :

1. Disarankan tidak memasukan nilai k, CP dan MP terlalu kecil ke dalam sistem karena akan menyebabkan tidak didapatnya nilai akurasi dari sistem. Nilai k dan CP yang terlalu kecil dapat menyebabkan terlalu banyak bobot ketetanggaan yang hilang. Sedangkan nilai MP yang terlalu kecil dapat menyebabkan sistem *looping* terus-menerus dikarenakan nilai prosentase perubahan bobot tidak lebih kecil dari nilai MP untuk setiap iterasi.
2. Pengembangan Algoritma *Shrinking based Shared Nearest Neighbor* sehingga Algoritma *Shrinking based Shared Nearest Neighbor* dapat digunakan untuk *clustering dataset* bertipe *categorical* dan *dataset* yang memiliki *missing value*.
3. Disertakannya waktu komputasi sehingga dapat diketahui berapa waktu yang diperlukan algoritma *Shrinking based Shared Nearest Neighbor* untuk menghasilkan sebuah nilai akurasi.

DAFTAR PUSTAKA

Andayani, S., 2007, *Pembentukan Cluster Dalam Knowledge Discoveryin Database dengan Algoritma K-Means*, Jurusan Pendidikan Matematika FMIPA Universitas Negeri Yogyakarta, Yogyakarta.

Budiarti, A., 2006, *Sistem Dan Analisis Clustering Pada Data Akademik*, Fakultas Ilmu Komputer, Universitas Indonesia, Depok.

Hamzah, A., Soesianto, F., Susanto, A., & Istiyanto, J.E., 2008, *Studi Kinerja Fungsi-Fungsi Jarak Dan Similaritas Dalam Clustering Dokumen Teks Berbahasa Indonesia*, Yogyakarta.

Jain, A.K., Murty, M.N., & Flynn, P.J., 1999, *Data Clustering: A Review, ACM Computing Surveys*, Vol. 31, No. 3.

Jarvis, R.A., Patrick, E.A., 1973, *Clustering Using a Sim-ilarity Measure Based on Shared Near Neighbors*. IEEE Trans. Comput. 22(11) (1973) 1025–1034

Karlita, T., 2011, Klasterisasi Data Kategorikal dengan Menggunakan Algoritma Modes Linkage, Program Studi Teknik Informatika Politeknik Elektronika Negeri, Surabaya.

Karypis, George, Han, E.H., & Chamelon, V.K., 1999, *A Hierarchical Clustering Algorithm Using Dynamic Modeling*, IEEE.

Kurniawan, H., Aji, R.F., 2006, *Otomatisasi Pengelompokan Koleksi Perpustakaan dengan Pengukuran Cosine Similarity dan Euclidean Distance*, Yogyakarta.

Moertini, V. S., 2002, *Data mining Sebagai Solusi Bisnis*, Integral, vol. 7no. 1.

Rodiyansyah, S.F., 2010, *Ekstraksi Histogram Citra Digital Untuk Mengukur Similarity dengan Menggunakan Metode Euclidian Distance*, Magister Ilmu Komputer – Universitas Gadjah Mada, Yogyakarta.

Shalabi, L.A., Shaaban, Z., & Kasasbeh, B., 2006, *Data mining: A Preprocessing Engine*, Science University, Amman, Jordan.

UCI *Machine Learning Repository*, *http://archive.ics.uci.edu/ml/*, Diakses tanggal 23 Oktober 2011

Widyawati, N., Wibisono, Y., & Kusnendar, J., 2007, *Perbandingan Clustering Based On Frequent Word Sequence Dan K-Means Untuk Pengelompokan Dokumen Berbahasa Indonesia*, Program Ilmu Komputer Universitas Pendidikan Indonesia, Bandung.

Wijayanto & Agung, O., 2009, *Aplikasi Data Mining Menggunakan Algoritma Shared Nearest Neighbor Untuk Clustering Kondisi Perumahan Di Kota/Kabupaten Se Jawa Timur*, Institut Teknologi Sepuluh Nopember, Surabaya.

Wirdati, D.R., 2007, *Perbandingan Rumus Cosine, Euclidean Distance, Intra Cluster Similarity, Dan Variance Pada Metode Bisecting K-Means Clusteringuntuk Pengelompokan E-Journal Berbahasa Inggris*, Universitas Brawijaya, Malang.

Yampolskiy, R.V & Govindaraju, V., 2006, *Similarity Measure Functions for Strategy-Based Biometrics* , International Journal of Biological & Medical Sciences 1:4.

Zainal, R.F., & Djunaidy, A., 2008, *Algoritma Shared Nearest Neighbor Berbasis Data Shrinking*, Vol. 7, No. 1, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember, Surabaya.

LAMPIRAN

Dataset *Iris*

No	*sepal length*	*sepal width*	*petal length*	*petal width*	kelas
1	5.1	3.5	1.4	0.2	*Setosa*
2	4.9	3	1.4	0.2	*Setosa*
3	4.7	3.2	1.3	0.2	*Setosa*
4	4.6	3.1	1.5	0.2	*Setosa*
5	5	3.6	1.4	0.2	*Setosa*
6	5.4	3.9	1.7	0.4	*Setosa*
7	4.6	3.4	1.4	0.3	*Setosa*
8	5	3.4	1.5	0.2	*Setosa*
9	4.4	2.9	1.4	0.2	*Setosa*
10	4.9	3.1	1.5	0.1	*Setosa*
11	5.4	3.7	1.5	0.2	*Setosa*
12	4.8	3.4	1.6	0.2	*Setosa*
13	4.8	3	1.4	0.1	*Setosa*
14	4.3	3	1.1	0.1	*Setosa*
15	5.8	4	1.2	0.2	*Setosa*
16	5.7	4.4	1.5	0.4	*Setosa*
17	5.4	3.9	1.3	0.4	*Setosa*
18	5.1	3.5	1.4	0.3	*Setosa*
19	5.7	3.8	1.7	0.3	*Setosa*
20	5.1	3.8	1.5	0.3	*Setosa*
21	5.4	3.4	1.7	0.2	*Setosa*
22	5.1	3.7	1.5	0.4	*Setosa*
23	4.6	3.6	1	0.2	*Setosa*
24	5.1	3.3	1.7	0.5	*Setosa*
25	4.8	3.4	1.9	0.2	*Setosa*
26	5	3	1.6	0.2	*Setosa*
27	5	3.4	1.6	0.4	*Setosa*
28	5.2	3.5	1.5	0.2	*Setosa*
29	5.2	3.4	1.4	0.2	*Setosa*
30	4.7	3.2	1.6	0.2	*Setosa*
31	4.8	3.1	1.6	0.2	*Setosa*
32	5.4	3.4	1.5	0.4	*Setosa*
33	5.2	4.1	1.5	0.1	*Setosa*
34	5.5	4.2	1.4	0.2	*Setosa*

35	4.9	3.1	1.5	0.1	*Setosa*
36	5	3.2	1.2	0.2	*Setosa*
37	5.5	3.5	1.3	0.2	*Setosa*
38	4.9	3.1	1.5	0.1	*Setosa*
39	4.4	3	1.3	0.2	*Setosa*
40	5.1	3.4	1.5	0.2	*Setosa*
41	5	3.5	1.3	0.3	*Setosa*
42	4.5	2.3	1.3	0.3	*Setosa*
43	4.4	3.2	1.3	0.2	*Setosa*
44	5	3.5	1.6	0.6	*Setosa*
45	5.1	3.8	1.9	0.4	*Setosa*
46	4.8	3	1.4	0.3	*Setosa*
47	5.1	3.8	1.6	0.2	*Setosa*
48	4.6	3.2	1.4	0.2	*Setosa*
49	5.3	3.7	1.5	0.2	*Setosa*
50	5	3.3	1.4	0.2	*Setosa*
51	7	3.2	4.7	1.4	*Versicolor*
52	6.4	3.2	4.5	1.5	*Versicolor*
53	6.9	3.1	4.9	1.5	*Versicolor*
54	5.5	2.3	4	1.3	*Versicolor*
55	6.5	2.8	4.6	1.5	*Versicolor*
56	5.7	2.8	4.5	1.3	*Versicolor*
57	6.3	3.3	4.7	1.6	*Versicolor*
58	4.9	2.4	3.3	1	*Versicolor*
59	6.6	2.9	4.6	1.3	*Versicolor*
60	5.2	2.7	3.9	1.4	*Versicolor*
61	5	2	3.5	1	*Versicolor*
62	5.9	3	4.2	1.5	*Versicolor*
63	6	2.2	4	1	*Versicolor*
64	6.1	2.9	4.7	1.4	*Versicolor*
65	5.6	2.9	3.6	1.3	*Versicolor*
66	6.7	3.1	4.4	1.4	*Versicolor*
67	5.6	3	4.5	1.5	*Versicolor*
68	5.8	2.7	4.1	1	*Versicolor*
69	6.2	2.2	4.5	1.5	*Versicolor*
70	5.6	2.5	3.9	1.1	*Versicolor*
71	5.9	3.2	4.8	1.8	*Versicolor*
72	6.1	2.8	4	1.3	*Versicolor*
73	6.3	2.5	4.9	1.5	*Versicolor*

74	6.1	2.8	4.7	1.2	*Versicolor*
75	6.4	2.9	4.3	1.3	*Versicolor*
76	6.6	3	4.4	1.4	*Versicolor*
77	6.8	2.8	4.8	1.4	*Versicolor*
78	6.7	3	5	1.7	*Versicolor*
79	6	2.9	4.5	1.5	*Versicolor*
80	5.7	2.6	3.5	1	*Versicolor*
81	5.5	2.4	3.8	1.1	*Versicolor*
82	5.5	2.4	3.7	1	*Versicolor*
83	5.8	2.7	3.9	1.2	*Versicolor*
84	6	2.7	5.1	1.6	*Versicolor*
85	5.4	3	4.5	1.5	*Versicolor*
86	6	3.4	4.5	1.6	*Versicolor*
87	6.7	3.1	4.7	1.5	*Versicolor*
88	6.3	2.3	4.4	1.3	*Versicolor*
89	5.6	3	4.1	1.3	*Versicolor*
90	5.5	2.5	4	1.3	*Versicolor*
91	5.5	2.6	4.4	1.2	*Versicolor*
92	6.1	3	4.6	1.4	*Versicolor*
93	5.8	2.6	4	1.2	*Versicolor*
94	5	2.3	3.3	1	*Versicolor*
95	5.6	2.7	4.2	1.3	*Versicolor*
96	5.7	3	4.2	1.2	*Versicolor*
97	5.7	2.9	4.2	1.3	*Versicolor*
98	6.2	2.9	4.3	1.3	*Versicolor*
99	5.1	2.5	3	1.1	*Versicolor*
100	5.7	2.8	4.1	1.3	*Versicolor*
101	6.3	3.3	6	2.5	*Virginica*
102	5.8	2.7	5.1	1.9	*Virginica*
103	7.1	3	5.9	2.1	*Virginica*
104	6.3	2.9	5.6	1.8	*Virginica*
105	6.5	3	5.8	2.2	*Virginica*
106	7.6	3	6.6	2.1	*Virginica*
107	4.9	2.5	4.5	1.7	*Virginica*
108	7.3	2.9	6.3	1.8	*Virginica*
109	6.7	2.5	5.8	1.8	*Virginica*
110	7.2	3.6	6.1	2.5	*Virginica*
111	6.5	3.2	5.1	2	*Virginica*
112	6.4	2.7	5.3	1.9	*Virginica*

113	6.8	3	5.5	2.1	*Virginica*
114	5.7	2.5	5	2	*Virginica*
115	5.8	2.8	5.1	2.4	*Virginica*
116	6.4	3.2	5.3	2.3	*Virginica*
117	6.5	3	5.5	1.8	*Virginica*
118	7.7	3.8	6.7	2.2	*Virginica*
119	7.7	2.6	6.9	2.3	*Virginica*
120	6	2.2	5	1.5	*Virginica*
121	6.9	3.2	5.7	2.3	*Virginica*
122	5.6	2.8	4.9	2	*Virginica*
123	7.7	2.8	6.7	2	*Virginica*
124	6.3	2.7	4.9	1.8	*Virginica*
125	6.7	3.3	5.7	2.1	*Virginica*
126	7.2	3.2	6	1.8	*Virginica*
127	6.2	2.8	4.8	1.8	*Virginica*
128	6.1	3	4.9	1.8	*Virginica*
129	6.4	2.8	5.6	2.1	*Virginica*
130	7.2	3	5.8	1.6	*Virginica*
131	7.4	2.8	6.1	1.9	*Virginica*
132	7.9	3.8	6.4	2	*Virginica*
133	6.4	2.8	5.6	2.2	*Virginica*
134	6.3	2.8	5.1	1.5	*Virginica*
135	6.1	2.6	5.6	1.4	*Virginica*
136	7.7	3	6.1	2.3	*Virginica*
137	6.3	3.4	5.6	2.4	*Virginica*
138	6.4	3.1	5.5	1.8	*Virginica*
139	6	3	4.8	1.8	*Virginica*
140	6.9	3.1	5.4	2.1	*Virginica*
141	6.7	3.1	5.6	2.4	*Virginica*
142	6.9	3.1	5.1	2.3	*Virginica*
143	5.8	2.7	5.1	1.9	*Virginica*
144	6.8	3.2	5.9	2.3	*Virginica*
145	6.7	3.3	5.7	2.5	*Virginica*
146	6.7	3	5.2	2.3	*Virginica*
147	6.3	2.5	5	1.9	*Virginica*
148	6.5	3	5.2	2	*Virginica*
149	6.2	3.4	5.4	2.3	*Virginica*
150	5.9	3	5.1	1.8	*Virginica*

Dataset Wine

No	Att1	Att2	Att3	Att4	Att5	Att6	Att7	Att8	Att9	Att10	Att11	Att12	Att13	kelas
1	14.23	1.71	2.43	15.6	127	2.8	3.06	0.28	2.29	5.64	1.04	3.92	1065	1
2	13.2	1.78	2.14	11.2	100	2.65	2.76	0.26	1.28	4.38	1.05	3.4	1050	1
3	13.16	2.36	2.67	18.6	101	2.8	3.24	0.3	2.81	5.68	1.03	3.17	1185	1
4	14.37	1.95	2.5	16.8	113	3.85	3.49	0.24	2.18	7.8	0.86	3.45	1480	1
5	13.24	2.59	2.87	21	118	2.8	2.69	0.39	1.82	4.32	1.04	2.93	735	1
6	14.2	1.76	2.45	15.2	112	3.27	3.39	0.34	1.97	6.75	1.05	2.85	1450	1
7	14.39	1.87	2.45	14.6	96	2.5	2.52	0.3	1.98	5.25	1.02	3.58	1290	1
8	14.06	2.15	2.61	17.6	121	2.6	2.51	0.31	1.25	5.05	1.06	3.58	1295	1
9	14.83	1.64	2.17	14	97	2.8	2.98	0.29	1.98	5.2	1.08	2.85	1045	1
10	13.86	1.35	2.27	16	98	2.98	3.15	0.22	1.85	7.22	1.01	3.55	1045	1
11	14.1	2.16	2.3	18	105	2.95	3.32	0.22	2.38	5.75	1.25	3.17	1510	1
12	14.12	1.48	2.32	16.8	95	2.2	2.43	0.26	1.57	5	1.17	2.82	1280	1
13	13.75	1.73	2.41	16	89	2.6	2.76	0.29	1.81	5.6	1.15	2.9	1320	1
14	14.75	1.73	2.39	11.4	91	3.1	3.69	0.43	2.81	5.4	1.25	2.73	1150	1
15	14.38	1.87	2.38	12	102	3.3	3.64	0.29	2.96	7.5	1.2	3	1547	1
16	13.63	1.81	2.7	17.2	112	2.85	2.91	0.3	1.46	7.3	1.28	2.88	1310	1
17	14.3	1.92	2.72	20	120	2.8	3.14	0.33	1.97	6.2	1.07	2.65	1280	1
18	13.83	1.57	2.62	20	115	2.95	3.4	0.4	1.72	6.6	1.13	2.57	1130	1
19	14.19	1.59	2.48	16.5	108	3.3	3.93	0.32	1.86	8.7	1.23	2.82	1680	1
20	13.64	3.1	2.56	15.2	116	2.7	3.03	0.17	1.66	5.1	0.96	3.36	845	1
21	14.06	1.63	2.28	16	126	3	3.17	0.24	2.1	5.65	1.09	3.71	780	1

22	12.93	3.8	2.65	18.6	102	2.41	2.41	0.25	1.98	4.5	1.03	3.52	770	1
23	13.71	1.86	2.36	16.6	101	2.61	2.88	0.27	1.69	3.8	1.11	4	1035	1
24	12.85	1.6	2.52	17.8	95	2.48	2.37	0.26	1.46	3.93	1.09	3.63	1015	1
25	13.5	1.81	2.61	20	96	2.53	2.61	0.28	1.66	3.52	1.12	3.82	845	1
26	13.05	2.05	3.22	25	124	2.63	2.68	0.47	1.92	3.58	1.13	3.2	830	1
27	13.39	1.77	2.62	16.1	93	2.85	2.94	0.34	1.45	4.8	0.92	3.22	1195	1
28	13.3	1.72	2.14	17	94	2.4	2.19	0.27	1.35	3.95	1.02	2.77	1285	1
29	13.87	1.9	2.8	19.4	107	2.95	2.97	0.37	1.76	4.5	1.25	3.4	915	1
30	14.02	1.68	2.21	16	96	2.65	2.33	0.26	1.98	4.7	1.04	3.59	1035	1
31	13.73	1.5	2.7	22.5	101	3	3.25	0.29	2.38	5.7	1.19	2.71	1285	1
32	13.58	1.66	2.36	19.1	106	2.86	3.19	0.22	1.95	6.9	1.09	2.88	1515	1
33	13.68	1.83	2.36	17.2	104	2.42	2.69	0.42	1.97	3.84	1.23	2.87	990	1
34	13.76	1.53	2.7	19.5	132	2.95	2.74	0.5	1.35	5.4	1.25	3	1235	1
35	13.51	1.8	2.65	19	110	2.35	2.53	0.29	1.54	4.2	1.1	2.87	1095	1
36	13.48	1.81	2.41	20.5	100	2.7	2.98	0.26	1.86	5.1	1.04	3.47	920	1
37	13.28	1.64	2.84	15.5	110	2.6	2.68	0.34	1.36	4.6	1.09	2.78	880	1
38	13.05	1.65	2.55	18	98	2.45	2.43	0.29	1.44	4.25	1.12	2.51	1105	1
39	13.07	1.5	2.1	15.5	98	2.4	2.64	0.28	1.37	3.7	1.18	2.69	1020	1
40	14.22	3.99	2.51	13.2	128	3	3.04	0.2	2.08	5.1	0.89	3.53	760	1
41	13.56	1.71	2.31	16.2	117	3.15	3.29	0.34	2.34	6.13	0.95	3.38	795	1
42	13.41	3.84	2.12	18.8	90	2.45	2.68	0.27	1.48	4.28	0.91	3	1035	1
43	13.88	1.89	2.59	15	101	3.25	3.56	0.17	1.7	5.43	0.88	3.56	1095	1
44	13.24	3.98	2.29	17.5	103	2.64	2.63	0.32	1.66	4.36	0.82	3	680	1
45	13.05	1.77	2.1	17	107	3	3	0.28	2.03	5.04	0.88	3.35	885	1

46	14.21	4.04	2.44	18.9	111	2.85	2.65	0.3	1.25	5.24	0.87	3.33	1080	1
47	14.38	3.59	2.28	16	102	3.25	3.17	0.27	2.19	4.9	1.04	3.44	1065	1
48	13.9	1.68	2.12	16	101	3.1	3.39	0.21	2.14	6.1	0.91	3.33	985	1
49	14.1	2.02	2.4	18.8	103	2.75	2.92	0.32	2.38	6.2	1.07	2.75	1060	1
50	13.94	1.73	2.27	17.4	108	2.88	3.54	0.32	2.08	8.9	1.12	3.1	1260	1
51	13.05	1.73	2.04	12.4	92	2.72	3.27	0.17	2.91	7.2	1.12	2.91	1150	1
52	13.83	1.65	2.6	17.2	94	2.45	2.99	0.22	2.29	5.6	1.24	3.37	1265	1
53	13.82	1.75	2.42	14	111	3.88	3.74	0.32	1.87	7.05	1.01	3.26	1190	1
54	13.77	1.9	2.68	17.1	115	3	2.79	0.39	1.68	6.3	1.13	2.93	1375	1
55	13.74	1.67	2.25	16.4	118	2.6	2.9	0.21	1.62	5.85	0.92	3.2	1060	1
56	13.56	1.73	2.46	20.5	116	2.96	2.78	0.2	2.45	6.25	0.98	3.03	1120	1
57	14.22	1.7	2.3	16.3	118	3.2	3	0.26	2.03	6.38	0.94	3.31	970	1
58	13.29	1.97	2.68	16.8	102	3	3.23	0.31	1.66	6	1.07	2.84	1270	1
59	13.72	1.43	2.5	16.7	108	3.4	3.67	0.19	2.04	6.8	0.89	2.87	1285	1
60	12.37	0.94	1.36	10.6	88	1.98	0.57	0.28	0.42	1.95	1.05	1.82	520	2
61	12.33	1.1	2.28	16	101	2.05	1.09	0.63	0.41	3.27	1.25	1.67	680	2
62	12.64	1.36	2.02	16.8	100	2.02	1.41	0.53	0.62	5.75	0.98	1.59	450	2
63	13.67	1.25	1.92	18	94	2.1	1.79	0.32	0.73	3.8	1.23	2.46	630	2
64	12.37	1.13	2.16	19	87	3.5	3.1	0.19	1.87	4.45	1.22	2.87	420	2
65	12.17	1.45	2.53	19	104	1.89	1.75	0.45	1.03	2.95	1.45	2.23	355	2
66	12.37	1.21	2.56	18.1	98	2.42	2.65	0.37	2.08	4.6	1.19	2.3	678	2
67	13.11	1.01	1.7	15	78	2.98	3.18	0.26	2.28	5.3	1.12	3.18	502	2
68	12.37	1.17	1.92	19.6	78	2.11	2	0.27	1.04	4.68	1.12	3.48	510	2
69	13.34	0.94	2.36	17	110	2.53	1.3	0.55	0.42	3.17	1.02	1.93	750	2

70	12.21	1.19	1.75	16.8	151	1.85	1.28	0.14	2.5	2.85	1.28	3.07	718	2
71	12.29	1.61	2.21	20.4	103	1.1	1.02	0.37	1.46	3.05	0.906	1.82	870	2
72	13.86	1.51	2.67	25	86	2.95	2.86	0.21	1.87	3.38	1.36	3.16	410	2
73	13.49	1.66	2.24	24	87	1.88	1.84	0.27	1.03	3.74	0.98	2.78	472	2
74	12.99	1.67	2.6	30	139	3.3	2.89	0.21	1.96	3.35	1.31	3.5	985	2
75	11.96	1.09	2.3	21	101	3.38	2.14	0.13	1.65	3.21	0.99	3.13	886	2
76	11.66	1.88	1.92	16	97	1.61	1.57	0.34	1.15	3.8	1.23	2.14	428	2
77	13.03	0.9	1.71	16	86	1.95	2.03	0.24	1.46	4.6	1.19	2.48	392	2
78	11.84	2.89	2.23	18	112	1.72	1.32	0.43	0.95	2.65	0.96	2.52	500	2
79	12.33	0.99	1.95	14.8	136	1.9	1.85	0.35	2.76	3.4	1.06	2.31	750	2
80	12.7	3.87	2.4	23	101	2.83	2.55	0.43	1.95	2.57	1.19	3.13	463	2
81	12	0.92	2	19	86	2.42	2.26	0.3	1.43	2.5	1.38	3.12	278	2
82	12.72	1.81	2.2	18.8	86	2.2	2.53	0.26	1.77	3.9	1.16	3.14	714	2
83	12.08	1.13	2.51	24	78	2	1.58	0.4	1.4	2.2	1.31	2.72	630	2
84	13.05	3.86	2.32	22.5	85	1.65	1.59	0.61	1.62	4.8	0.84	2.01	515	2
85	11.84	0.89	2.58	18	94	2.2	2.21	0.22	2.35	3.05	0.79	3.08	520	2
86	12.67	0.98	2.24	18	99	2.2	1.94	0.3	1.46	2.62	1.23	3.16	450	2
87	12.16	1.61	2.31	22.8	90	1.78	1.69	0.43	1.56	2.45	1.33	2.26	495	2
88	11.65	1.67	2.62	26	88	1.92	1.61	0.4	1.34	2.6	1.36	3.21	562	2
89	11.64	2.06	2.46	21.6	84	1.95	1.69	0.48	1.35	2.8	1	2.75	680	2
90	12.08	1.33	2.3	23.6	70	2.2	1.59	0.42	1.38	1.74	1.07	3.21	625	2
91	12.08	1.83	2.32	18.5	81	1.6	1.5	0.52	1.64	2.4	1.08	2.27	480	2
92	12	1.51	2.42	22	86	1.45	1.25	0.5	1.63	3.6	1.05	2.65	450	2
93	12.69	1.53	2.26	20.7	80	1.38	1.46	0.58	1.62	3.05	0.96	2.06	495	2

94	12.29	2.83	2.22	18	88	2.45	2.25	0.25	1.99	2.15	1.15	3.3	290	2
95	11.62	1.99	2.28	18	98	3.02	2.26	0.17	1.35	3.25	1.16	2.96	345	2
96	12.47	1.52	2.2	19	162	2.5	2.27	0.32	3.28	2.6	1.16	2.63	937	2
97	11.81	2.12	2.74	21.5	134	1.6	0.99	0.14	1.56	2.5	0.95	2.26	625	2
98	12.29	1.41	1.98	16	85	2.55	2.5	0.29	1.77	2.9	1.23	2.74	428	2
99	12.37	1.07	2.1	18.5	88	3.52	3.75	0.24	1.95	4.5	1.04	2.77	660	2
100	12.29	3.17	2.21	18	88	2.85	2.99	0.45	2.81	2.3	1.42	2.83	406	2
101	12.08	2.08	1.7	17.5	97	2.23	2.17	0.26	1.4	3.3	1.27	2.96	710	2
102	12.6	1.34	1.9	18.5	88	1.45	1.36	0.29	1.35	2.45	1.04	2.77	562	2
103	12.34	2.45	2.46	21	98	2.56	2.11	0.34	1.31	2.8	0.8	3.38	438	2
104	11.82	1.72	1.88	19.5	86	2.5	1.64	0.37	1.42	2.06	0.94	2.44	415	2
105	12.51	1.73	1.98	20.5	85	2.2	1.92	0.32	1.48	2.94	1.04	3.57	672	2
106	12.42	2.55	2.27	22	90	1.68	1.84	0.66	1.42	2.7	0.86	3.3	315	2
107	12.25	1.73	2.12	19	80	1.65	2.03	0.37	1.63	3.4	1	3.17	510	2
108	12.72	1.75	2.28	22.5	84	1.38	1.76	0.48	1.63	3.3	0.88	2.42	488	2
109	12.22	1.29	1.94	19	92	2.36	2.04	0.39	2.08	2.7	0.86	3.02	312	2
110	11.61	1.35	2.7	20	94	2.74	2.92	0.29	2.49	2.65	0.96	3.26	680	2
111	11.46	3.74	1.82	19.5	107	3.18	2.58	0.24	3.58	2.9	0.75	2.81	562	2
112	12.52	2.43	2.17	21	88	2.55	2.27	0.26	1.22	2	0.9	2.78	325	2
113	11.76	2.68	2.92	20	103	1.75	2.03	0.6	1.05	3.8	1.23	2.5	607	2
114	11.41	0.74	2.5	21	88	2.48	2.01	0.42	1.44	3.08	1.1	2.31	434	2
115	12.08	1.39	2.5	22.5	84	2.56	2.29	0.43	1.04	2.9	0.93	3.19	385	2
116	11.03	1.51	2.2	21.5	85	2.46	2.17	0.52	2.01	1.9	1.71	2.87	407	2
117	11.82	1.47	1.99	20.8	86	1.98	1.6	0.3	1.53	1.95	0.95	3.33	495	2

118	12.42	1.61	2.19	22.5	108	2	2.09	0.34	1.61	2.06	1.06	2.96	345	2
119	12.77	3.43	1.98	16	80	1.63	1.25	0.43	0.83	3.4	0.7	2.12	372	2
120	12	3.43	2	19	87	2	1.64	0.37	1.87	1.28	0.93	3.05	564	2
121	11.45	2.4	2.42	20	96	2.9	2.79	0.32	1.83	3.25	0.8	3.39	625	2
122	11.56	2.05	3.23	28.5	119	3.18	5.08	0.47	1.87	6	0.93	3.69	465	2
123	12.42	4.43	2.73	26.5	102	2.2	2.13	0.43	1.71	2.08	0.92	3.12	365	2
124	13.05	5.8	2.13	21.5	86	2.62	2.65	0.3	2.01	2.6	0.73	3.1	380	2
125	11.87	4.31	2.39	21	82	2.86	3.03	0.21	2.91	2.8	0.75	3.64	380	2
126	12.07	2.16	2.17	21	85	2.6	2.65	0.37	1.35	2.76	0.86	3.28	378	2
127	12.43	1.53	2.29	21.5	86	2.74	3.15	0.39	1.77	3.94	0.69	2.84	352	2
128	11.79	2.13	2.78	28.5	92	2.13	2.24	0.58	1.76	3	0.97	2.44	466	2
129	12.37	1.63	2.3	24.5	88	2.22	2.45	0.4	1.9	2.12	0.89	2.78	342	2
130	12.04	4.3	2.38	22	80	2.1	1.75	0.42	1.35	2.6	0.79	2.57	580	2
131	12.86	1.35	2.32	18	122	1.51	1.25	0.21	0.94	4.1	0.76	1.29	630	3
132	12.88	2.99	2.4	20	104	1.3	1.22	0.24	0.83	5.4	0.74	1.42	530	3
133	12.81	2.31	2.4	24	98	1.15	1.09	0.27	0.83	5.7	0.66	1.36	560	3
134	12.7	3.55	2.36	21.5	106	1.7	1.2	0.17	0.84	5	0.78	1.29	600	3
135	12.51	1.24	2.25	17.5	85	2	0.58	0.6	1.25	5.45	0.75	1.51	650	3
136	12.6	2.46	2.2	18.5	94	1.62	0.66	0.63	0.94	7.1	0.73	1.58	695	3
137	12.25	4.72	2.54	21	89	1.38	0.47	0.53	0.8	3.85	0.75	1.27	720	3
138	12.53	5.51	2.64	25	96	1.79	0.6	0.63	1.1	5	0.82	1.69	515	3
139	13.49	3.59	2.19	19.5	88	1.62	0.48	0.58	0.88	5.7	0.81	1.82	580	3
140	12.84	2.96	2.61	24	101	2.32	0.6	0.53	0.81	4.92	0.89	2.15	590	3
141	12.93	2.81	2.7	21	96	1.54	0.5	0.53	0.75	4.6	0.77	2.31	600	3

142	13.36	2.56	2.35	20	89	1.4	0.5	0.37	0.64	5.6	0.7	2.47	780	3
143	13.52	3.17	2.72	23.5	97	1.55	0.52	0.5	0.55	4.35	0.89	2.06	520	3
144	13.62	4.95	2.35	20	92	2	0.8	0.47	1.02	4.4	0.91	2.05	550	3
145	12.25	3.88	2.2	18.5	112	1.38	0.78	0.29	1.14	8.21	0.65	2	855	3
146	13.16	3.57	2.15	21	102	1.5	0.55	0.43	1.3	4	0.6	1.68	830	3
147	13.88	5.04	2.23	20	80	0.98	0.34	0.4	0.68	4.9	0.58	1.33	415	3
148	12.87	4.61	2.48	21.5	86	1.7	0.65	0.47	0.86	7.65	0.54	1.86	625	3
149	13.32	3.24	2.38	21.5	92	1.93	0.76	0.45	1.25	8.42	0.55	1.62	650	3
150	13.08	3.9	2.36	21.5	113	1.41	1.39	0.34	1.14	9.4	0.57	1.33	550	3
151	13.5	3.12	2.62	24	123	1.4	1.57	0.22	1.25	8.6	0.59	1.3	500	3
152	12.79	2.67	2.48	22	112	1.48	1.36	0.24	1.26	10.8	0.48	1.47	480	3
153	13.11	1.9	2.75	25.5	116	2.2	1.28	0.26	1.56	7.1	0.61	1.33	425	3
154	13.23	3.3	2.28	18.5	98	1.8	0.83	0.61	1.87	10.52	0.56	1.51	675	3
155	12.58	1.29	2.1	20	103	1.48	0.58	0.53	1.4	7.6	0.58	1.55	640	3
156	13.17	5.19	2.32	22	93	1.74	0.63	0.61	1.55	7.9	0.6	1.48	725	3
157	13.84	4.12	2.38	19.5	89	1.8	0.83	0.48	1.56	9.01	0.57	1.64	480	3
158	12.45	3.03	2.64	27	97	1.9	0.58	0.63	1.14	7.5	0.67	1.73	880	3
159	14.34	1.68	2.7	25	98	2.8	1.31	0.53	2.7	13	0.57	1.96	660	3
160	13.48	1.67	2.64	22.5	89	2.6	1.1	0.52	2.29	11.75	0.57	1.78	620	3
161	12.36	3.83	2.38	21	88	2.3	0.92	0.5	1.04	7.65	0.56	1.58	520	3
162	13.69	3.26	2.54	20	107	1.83	0.56	0.5	0.8	5.88	0.96	1.82	680	3
163	12.85	3.27	2.58	22	106	1.65	0.6	0.6	0.96	5.58	0.87	2.11	570	3
164	12.96	3.45	2.35	18.5	106	1.39	0.7	0.4	0.94	5.28	0.68	1.75	675	3
165	13.78	2.76	2.3	22	90	1.35	0.68	0.41	1.03	9.58	0.7	1.68	615	3

166	13.73	4.36	2.26	22.5	88	1.28	0.47	0.52	1.15	6.62	0.78	1.75	520	3
167	13.45	3.7	2.6	23	111	1.7	0.92	0.43	1.46	10.68	0.85	1.56	695	3
168	12.82	3.37	2.3	19.5	88	1.48	0.66	0.4	0.97	10.26	0.72	1.75	685	3
169	13.58	2.58	2.69	24.5	105	1.55	0.84	0.39	1.54	8.66	0.74	1.8	750	3
170	13.4	4.6	2.86	25	112	1.98	0.96	0.27	1.11	8.5	0.67	1.92	630	3
171	12.2	3.03	2.32	19	96	1.25	0.49	0.4	0.73	5.5	0.66	1.83	510	3
172	12.77	2.39	2.28	19.5	86	1.39	0.51	0.48	0.64	9.899999	0.57	1.63	470	3
173	14.16	2.51	2.48	20	91	1.68	0.7	0.44	1.24	9.7	0.62	1.71	660	3
174	13.71	5.65	2.45	20.5	95	1.68	0.61	0.52	1.06	7.7	0.64	1.74	740	3
175	13.4	3.91	2.48	23	102	1.8	0.75	0.43	1.41	7.3	0.7	1.56	750	3
176	13.27	4.28	2.26	20	120	1.59	0.69	0.43	1.35	10.2	0.59	1.56	835	3
177	13.17	2.59	2.37	20	120	1.65	0.68	0.53	1.46	9.3	0.6	1.62	840	3
178	14.13	4.1	2.74	24.5	96	2.05	0.76	0.56	1.35	9.2	0.61	1.6	560	3

Keterangan

Att1 : Alcohol
Att2 : Malic acid
Att3 : Ash
Att4 : Alcalinity of ash
Att5 : Magnesium
Att6 : Total phenols
Att7 : Flavanoids
Att8 : Nonflavanoid phenols
Att9 : Proanthocyanins
Att10 : Color intensity
Att11 : Hue
Att12 : OD280/OD315 of diluted wines
Att13 : Proline


UNIVERSITAS BRAWIJAYA