BAB 2
TINJAUAN PUSTAKA

2.1 *Clustering* dalam *Data mining*

Data mining merupakan kegiatan eksplorasi dan analisis data dalam jumlah besar dengan tujuan untuk menemukan pola yang bermakna. Data yang akan diproses dalam *data mining* ada kalanya belum diketahui label kelasnya. Dalam hal pengelompokan, data semacam ini dikelompokkan berdasarkan kemiripan dan ketidakmiripan antar datanya atau yang sering disebut dengan pembelajaran tidak terbimbing (*unsupervised learning*). Karena tidak ada label kelas yang digunakan dalam prosesnya, *clustering* sangat cocok digunakan untuk pengelompokan data yang label kelasnya memang sulit didapatkan pada saat pembangkitan fitur. Label kelas baru dapat diberikan setelah proses *clustering* selesai dengan cara mengamati hasil *cluster* yang didapat. Teknik *data mining clustering* juga dapat digunakan untuk mengeksplorasi dan mengkarakteristikkan set data pada pembelajaran terbimbing (*supervised learning*) (Prasetyo, 2014).

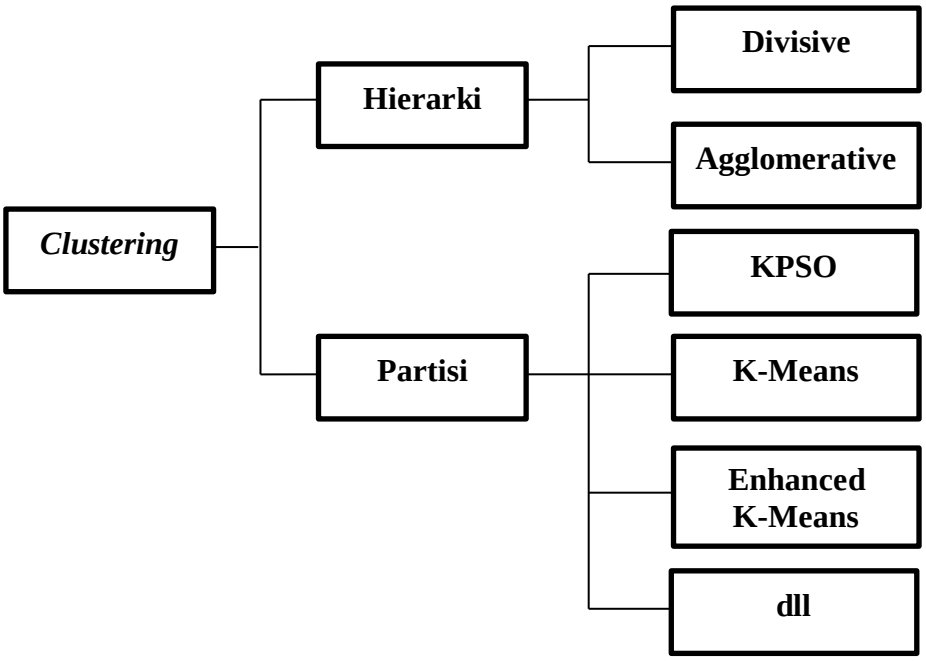
Hasil *clustering* dikatakan baik jika data dalam satu *cluster* memiliki kemiripan yang tinggi dan memiliki ketidakmiripan yang tinggi dengan data di *cluster* lain. Hal yang perlu diperhatikan dalam *clustering* adalah bahwa kemiripan (*similarity*) harus didefinisikan berdasarkan pada atribut objek. Kualitas hasil *clustering* yang terbentuk bergantung pada data dan algoritma *clustering* yang digunakan. Algoritma *clustering* yang berbeda akan cocok dengan jenis *dataset* yang berbeda pula. Oleh karena itu, diperlukan adanya kecocokan antara algoritma dan *dataset* yang akan diproses sehingga dapat dihasilkan hasil *clustering* yang baik.

Dalam kasus di dunia nyata, *clustering* digunakan salah satunya untuk melakukan segmentasi pasar atau pelanggan yang dapat membantu dalam *Customer Relationship Management* (CRM) (Singh & Rumantir, 2015). Melalui segmentasi pelanggan, perusahaan dapat mengetahui kelompok-kelompok pelanggannya serta bagaimana karakteristik setiap kelompok tersebut, sehingga

perusahaan dapat menerapkan strategi pemasaran yang sesuai untuk setiap kelompok pelanggan. Dilihat dari hal tersebut, dapat dikatakan bahwa hasil diperlukan adanya algoritma yang tepat untuk menghasilkan hasil *clustering* yang baik.

2.2 Teknik-teknik Dalam *Clustering*

Berdasarkan strukturnya, teknik *clustering* dibagi menjadi dua yaitu hierarki dan partisi. Pemetaan teknik *clustering* tersebut dapat dilihat di Gambar 2.1.



Gambar 2.1 Pemetaan Teknik-teknik *Clustering*

Dalam *clustering* berbasis hierarki, satu data tunggal bisa dianggap sebuah *cluster*, dua atau lebih *cluster* kecil dapat bergabung menjadi sebuah *cluster* besar, begitu seterusnya hingga semua data dapat bergabung menjadi sebuah *cluster*. Salah satu algoritma hierarki adalah Hierarchical Agglomerative Clustering. Di sisi lain, *clustering* berbasis partisi membagi set data ke dalam sejumlah *cluster* yang tidak bertumpang-tindih antara satu *cluster* dengan *cluster* lain, artinya setiap data hanya menjadi anggota satu *cluster* saja (Prasetyo, 2014). Algoritma seperti K-Means, Enhanced K-Means dan Particle Swarm Optimization termasuk ke dalam kategori ini.

2.2.1 K-Means

K-Means merupakan salah satu metode pengelompokan data nonhierarki yang berusaha mempartisi data yang ada ke dalam bentuk dua atau lebih kelompok. Metode ini mempartisi data ke dalam kelompok sehingga data berkarakteristik sama dimasukkan ke dalam satu kelompok yang sama dan data yang berkarakteristik berbeda dikelompokkan ke dalam kelompok yang lain (Prasetyo, 2014). Langkah-langkah algoritma K-Means adalah sebagai berikut:

1. Tentukan jumlah *cluster*
2. Alokasikan data ke dalam *cluster* secara acak
3. Hitung pusat *cluster* (*centroid*) dari data dimasing-masing kelompok
4. Alokasikan masing-masing data ke *centroid* terdekat
5. Kembali ke langkah 3, apabila masih ada data yang berpisah kelompok, atau apabila ada perubahan nilai *centroid* di atas nilai ambang yang ditentukan, atau apabila perubahan nilai pada fungsi objektif yang digunakan masih di atas nilai ambang yang ditentukan

Kelebihan algoritma K-Means adalah dapat menghasilkan hasil *clustering* dengan akurasi terbaik baik untuk *dataset* kecil maupun *dataset* besar. Namun algoritma K-Means memiliki kekurangan yang terletak pada penentuan jumlah *cluster* dan inisialisasi *centroid* (pusat *cluster*) awal. Hal ini menyebabkan hasil *clustering*-nya berupa solusi yang sifatnya *local minimum* yang artinya algoritma K-Means stidak selalu menjamin hasil terbaik untuk akurasi hasil akhir *cluster*, bergantung pada pemilihan pusat *cluster* awal.

2.2.2 K-Means Particle Swarm Optimization

K-Means Paritcle Swarm Optimization (KPSO) merupakan perbaikan dari algoritma K-Means yang menggunakan algoritma PSO untuk menentukan pusat *cluster* awal terbaik. Algoritma PSO merupakan proses pencarian berdasarkan populasi yang dimodelkan mengikuti perilaku sosial kawanan burung (Merwe & Engelbrecht, 2003). PSO menggunakan sekumpulan partikel yang bekerjasama

untuk mengeksplorasi solusi-solusi yang memungkinkan bagi permasalahan optimasi. Langkah-langkah dalam algoritma PSO adalah sebagai berikut:

Terdapat 6 tahapan utama pada algoritma PSO (Merwe & Engelbrecht, 2003), yaitu:

1. Inisialisasi populasi partikel awal

   Langkah ini merupakan tahap awal untuk menentukan jumlah dan koordinat partikel / posisi partikel.
2. Melakukan perhitungan dan evaluasi nilai fitness dari setiap partikel

   Untuk memperoleh nilai fitness dibutuhkan nilai posisi setiap partikel tersebut untuk dievaluasi oleh fungsi fitness.
3. Menentukan *Pbest* (*personal best*)

   Pbest adalah posisi terbaik setiap partikel. Posisi ini merupakan posisi yang ditempati oleh partikel saat ini.
4. Menentukan *Gbest* (*global best*)

   Nilai *Gbest* merupakan nilai posisi partikel yang memiliki posisi paling baik dari partikel lainnya. Nilai *Gbest* ini didapat dengan membandingkan nilai fitness dari setiap partikel untuk kemudian koordinat posisi dari partikel yang memiliki fitness paling optimal tersebut akan dijadikan *Gbest*. Nilai fitness dikatakan optimal tergantung kasus yang dimiliki apakah maksimasi ataupun minimasi.
5. Menghitung nilai *velocity* (V)

   Untuk mendapatkan nilai V terbaru dapat digunakan persamaan umum kecepatan PSO untuk memperbaharui kecepatan. Rumus nilai kecepatan PSO adalah sebagai berikut:

   $V_i^{t+1} = \omega V_i^t + C_1 R_1 (P_i^t - X_i^t) + C_2 R_2 \left(P_g^t - X_i^t\right)$ ..... (2.1)

   Keterangan :

   V_i^t = Kecepatan partikel ke-i pada iterasi ke-t

   ω = Beban partikel *(Innertia weight*)

   C = Konstanta Akselarasi (*Acceralation Constans)*

   R = Bilangan acak antara 0-1

X_i^t = Posisi partikel ke I pada iterasi ke-t

P_i = Solusi terbaik Lokal (*Local Best*)

P_g = Solusi terbaik Global (*Global Best*)

6. Menghitung nilai koordinat posisi terbaru.

   Memasukan hasil perhitungan nilai V dari setiap partikel ke dalam persamaan koordinat posisi PSO. Rumus Posisi PSO adalah sebagai berikut:

   $X_i^{t+1} = X_i^t + V_i^{t+1}$ ……...……...……...……...… (2.2)

   Keterangan :

   X_i^t = Posisi partikel ke I pada iterasi ke-t

   V_i^t = Kecepatan partikel ke-i pada iterasi ke-t

Kelebihan algoritma PSO adalah tidak tergantungnya dengan inisialisasi pusat *cluster* awal sehingga hasil *clustering* lebih stabil dari algoritma K-Means. Kekurangan algoritma PSO adalah ketidakmampuan menentukan jumlah *cluster* terbaik dari kumpulan data. Kelemahan lain dari algoritma PSO adalah memiliki kinerja DBI yang semakin menurun saat jumlah data semakin banyak.

2.2.3 Hierarchical Agglomerative Clustering

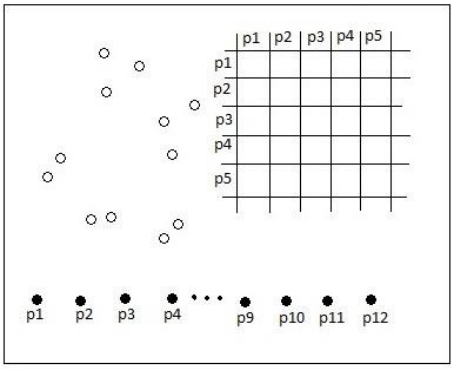
Hierarchical Agglomerative Clustering (HAC) merupakan metode *clustering* berbasis hierarki dengan pendekatan *bottom-up*, yaitu proses *clustering* dimulai dari masing-masing data sebagai satu buah *cluster*, kemudian secara rekursif mencari *cluster* terdekat sebagai pasangan untuk bergabung sebagai satu *cluster* yang lebih besar. Proses tersebut diulang terus sehingga tampak bergerak ke atas membentuk hierarki (Prasetyo, 2014). *Clustering* berbasis hierarki sering ditampilkan dalam bentuk grafis menggunakan diagram yang mirip *tree* (pohon) yang disebut dengan dendogram. Dendogram menampilkan hubungan *cluster* dengan *subcluster*-nya dalam urutan *cluster* yang digabung. Algoritma HAC adalah sebagai berikut:

Algoritma Hierarchical Agglomerative Clustering
1: Hitung matriks kedekatan berdasarkan jenis jarak yang digunakan
2: Ulangi langkah 3 sampai 4, hingga hanya satu *cluster* yang tersisa
3: Gabungkan dua *cluster* terdekat berdasarkan parameter kedekatan yang ditentukan
4: Perbarui matriks kedekatan untuk merefleksikan kedekatan di antara *cluster* baru yang tersisa

Tahapan algoritma Hierarchical Agglomerative Clustering adalah sebagai berikut (Fajar, 2013):

1. Tahap awal

Tahap awal dimulai dengan membentuk *cluster* dari setiap data dan membuat sebuah matriks kedekatan (*proximity matrix*). *Proximity matrix* yaitu matriks yang berisi jarak antar data.

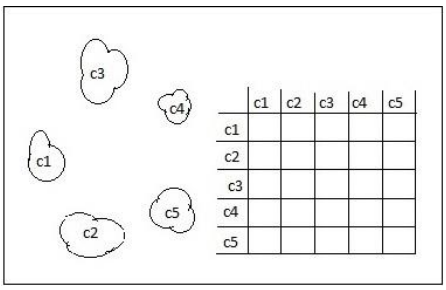


Gambar 2.2 Titik-titik Individual dan *Proximity Matrix*

Seperti yang terlihat pada Gambar 2.2, metode ini dimulai dengan meletakan setiap data sebagai sebuah *cluster* tersendiri. Misalkan terdapat 12 data mulai dari p1 hingga p12, berarti jumlah *cluster* awal yang terbentuk adalah 12 *cluster*. *Proximity matrix* diperlukan untuk menghitung jarak kedekatan antar data yaitu jarak antara p1 dengan p2, p1 dengan p3, dan seterusnya.

2. Tahap *intermediate*

Pada tahap ini dilakukan penggabungan dua *cluster* terdekat. Penggabungan ini dilakukan hingga hanya ada satu *cluster* yang tersisa.

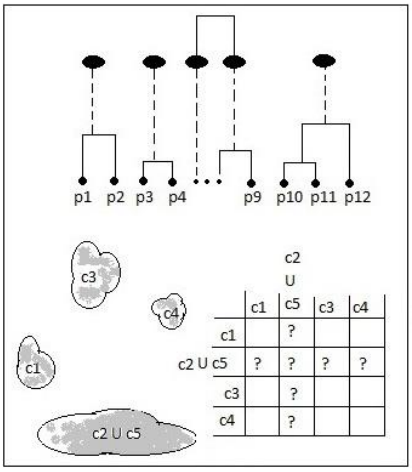


Gambar 2.3 Langkah Penggabungan *Cluster*

Gambar 2.3 memperlihatkan kondisi setelah beberapa langkah penggabungan *cluster*, yang dimisalkan diperoleh beberapa *cluster* yaitu c1, c2, c3, c4 dan c5.

3. Tahap setelah penggabungan *cluster*

Pada tahap ini dilakukan peng-*update*-an *proximity matrix*.



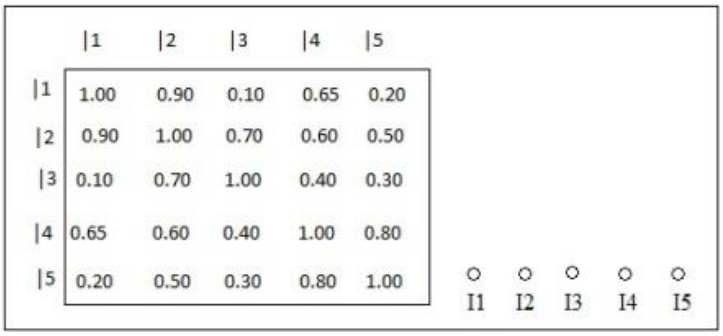
Gambar 2.4 Setelah Penggabungan

Misalkan dari lima *cluster* yang terbentuk tersebut, terjadi lagi penggabungan dua *cluster* terdekat yaitu C2 dan C5 seperti yang terlihat pada Gambar 2.4. Setiap dilakukan penggabungan dua *cluster* maka dilakukan juga peng-*update*-an *proximity matrix*, yaitu dengan menghitung kembali jarak kedekatan antara *cluster* baru dengan *cluster* yang masih tersisa.

4. *Complete link* : $\delta(C_x, C_y) = \max\{\delta(x, y) | x \epsilon C_x, y \epsilon C_y\}$

Complete link disebut juga *maximum link*, yang berarti jarak antara dua buah *cluster* dihitung dari jarak terjauh antara anggota *cluster* satu dengan *cluster* yang

kedua. Contoh: diketahui *proximity matrix* dan dendogram mula-mula adalah sebagai berikut:



Gambar 2.5 *Proximity Matrix* dan Dendogram Awal

Dari *proximity matrix* tersebut diketahui bahwa dari kelima *cluster* tersebut, *cluster* I1 dan I2 paling dekat (mempunyai similaritas paling besar), sehingga *cluster* I1 dan I2 digabung menjadi *cluster* I12. Sehingga *proximity matrix* yang baru menjadi:

Tabel 2.1 *Proximity Matrix* I1 dan I2

	I1,2	**I3**	**I4**	**I5**
I1,2	1	0.1	0.6	0.2
I3	0.1	1	0.4	0.3
I4	0.6	0.4	1	0.8
I5	0.2	0.3	0.8	1

$$\delta(I12, I3) = \max(\delta(I1, I3), \delta(I2, I3)) = \max(0.1,0.7) = 0.1$$

$$\delta(I12, I4) = \max(\delta(I1, I4), \delta(I2, I4)) = \max(0.65,0.6) = 0.6$$

$$\delta(I12, I5) = \max(\delta(I1, I5), \delta(I2, I5)) = \max(0.2,0.5) = 0.2$$

Dari *proximity matrix* di atas diketahui bahwa dari keempat *cluster* tersebut, *cluster* I4 dan I5 paling dekat (mempunyai similaritas paling besar), sehingga *cluster* I4 dan I5 digabung menjadi *cluster* I4,5. Sehingga *proximity matrix* yang baru menjadi:

Tabel 2.2 *Proximity Matrix* I4 dan I5

	I1,2	**I3**	**I4,5**
I1,2	1	0.1	0.2
I3	0.1	1	0.3
I4,5	0.2	0.3	1

$$\delta(I12, I45) = \max(\delta(I1, I4), \delta(I1, I5), \delta(I2, I4), \delta(I2, I5)) = \max(0.65,0.2,0.6,0.5) = 0.2$$

$$\delta(I13, I45) = \max(\delta(I3, I4), \delta(I3, I5)) = \max(0.4,0.3) = 0.3$$

Dari *proximity matrix* di atas diketahui bahwa dari ketiga *cluster* tersebut, *cluster* I3 dan I45 paling dekat (mempunyai similaritas paling besar), sehingga *cluster* I3 dan I4,5 digabung menjadi *cluster* I345.

Kelebihan algoritma HAC adalah tidak memerlukan inisialisasi pusat *cluster* awal dan dapat digunakan untuk menentukan jumlah *cluster* terbaik dari suatu kumpulan data. Sementara kekurangannya adalah tidak baik dalam menangani data dalam jumlah banyak.

2.2.4 Enhanced K-Means

Enhanced K-Means merupakan algoritma *clustering* pengembangan dari algoritma K-Means yang diusulkan oleh Yedla, dkk. (Yedla, Pathakota, & Srinivasa, 2010). Algoritma Enhanced K-Means diusulkan untuk mengatasi kekurangan pada algoritma K-Means. Hasil akhir *clustering* pada algoritma K-Means sangat bergantung pada ketepatan pemilihan pusat *cluster* awal yang dipilih secara acak. Hal ini menyebabkan hasil *clustering*-nya berupa solusi yang sifatnya *local minimum*. Algoritma Enhanced K-Means mengkombinasikan metode sistematis untuk mencari pusat *cluster* awal dan memberi cara yang efisien untuk memasukkan data ke dalam *cluster*.

Algoritma Enhanced K-Means adalah sebagai berikut:

Algoritma Enhanced K-Means
Inisiasi: $D = \{d_1, d_2, d_3, \dots, d_i, d_n\}$ // Kumpulan data sebanyak *n*. $d_i = \{x_1, x_2, x_3, \dots, x_i, x_m\}$ // Kumpulan atribut sebanyak *m*. k // Jumlah *cluster* yang akan dibentuk.
Langkah: 1: Pada dataset *D*, jika terdapat data yang atributnya bernilai positif dan negatif maka lanjutkan ke langkah 2, jika tidak maka lanjutkan ke langkah 4. 2: Cari nilai atribut minimum pada dataset D. 3: Untuk setiap data pada setiap atribut, kurangi dengan nilai minimumnya. 4: Untuk setiap data hitung jarak dengan data yang lain. 5: Urutkan data berdasarkan jaraknya. 6: Bagi data yang telah diurutkan ke dalam *k* bagian. Setiap bagian berisi data yang sama

banyak.
7: Pada setiap bagian, hitung nilai tengah sebagi titik pusat *cluster* awal
8: Hitung jarak antar setiap data d_i *(1 <= i <= n)* dengan setiap titik pusat *cluster* awal c_j *(1 <= j <= k).*
9: **Repeat**
10: Untuk setiap data d_i, cari pusat *cluster* terdekat c_j dan masukkan d_i ke dalam *cluster j*.
11: Ubah ClusterId[*i*]=*j*. // *j*:Id dari *cluster* terdekat.
12: Ubah NearestDist[*i*]= $d(d_i, c_j)$.
13: Untuk setiap *cluster j (1 <= j <= k)*, hitung ulang pusat *cluster*nya.
14: **For** setiap data d_i,
14.1 Hitung jarak dari pusat cluster baru dari cluster terdekatnya saat ini.
14.2 **If** jaraknya kurang dari atau sama dengan jarak terdekat sebelumnya, maka data tetap berada pada cluster yang sama
Else
14.2.1 Untuk setiap pusat *cluster* c_j *(1<=j<=k)* hitung daraknya dengan data $d(d_i, c_j)$.
End for;
Until kriteria bertemu ditemukan.

Clustering menggunakan Enhanced K-Means dimulai dengan pengecekan apakah *dataset* yang digunakan mengandung atribut dengan nilai negatif atau tidak. Jika *dataset* mengandung atribut yang bernilai negatif maka dilakukan pengubahan seluruh data di dalam *dataset* ke dalam bentuk positif dengan cara mengurangi setiap data dengan nilai atribut minimum dalam *dataset*. Pengubahan bentuk ini diperlukan untuk menghitung jarak dari pusat ke setiap data di dalam *dataset*. Sedangkan jika seluruh atribut di dalam *dataset* bernilai positif maka tidak perlu dilakukan pengubahan bentuk.

Selanjutnya untuk setiap data, hitung jarak antar datanya. Kemudian data diurutkan menurut jaraknya. Setelah diurutkan, bagi data ke dalam *k cluster*. Setiap *cluster* ambil nilai tengah sebagi pusat *cluster* awal. Pusat *cluster* awal ini memiliki peranan penting untuk menghasilkan hasil *clustering* yang lebih baik. Selanjutnya, hitung jarak setiap data ke semua pusat *cluster* awal. Tahap selanjutnya merupakan proses berulang dengan pendekatan heuristik untuk mengurangi kompleksitas waktu. Pada tahap selanjutnya, setiap data dimasukkan ke dalam *cluster* yang memiliki jarak terdekat dengan pusat *cluster*. *ClusterId* dari data digunakan untuk menunjukkan *cluster* mana yang memiliki data tersebut. *NearestDist* dari data digunakan untuk menunjukkan jarak terdekat saat ini dari pusat *cluster* awal terdekat.

Selanjutnya untuk setiap *cluster*, hitung pusat *cluster* baru dengan mencari rata-rata data pada *cluster* tersebut. Kemudian untuk setiap data, hitung jarak dari pusat *cluster* baru dari *cluster* terdekatnya saat ini. Jika jaraknya kurang dari atau sama dengan jarak terdekat sebelumnya, maka data tetap berada pada *cluster* yang sama, tetapi jika sebaliknya maka lakukan perhitungan jarak untuk setiap data dengan seluruh pusat *cluster*. Setelah menghitung jarak, data dimasukkan ke dalam *cluster* yang sesuai dan diberikan *ClusterId* baru dan nilai *NearestDist* diubah. Proses penempatan data ke dalam *cluster* diulang hingga kriteria bertemu ditemukan.

Algoritma Enhanced K-Means memiliki kelebihan yaitu dapat menentukan pusat *cluster* awal secara otomatis dan memiliki kinerja yang baik pada data banyak. Sementara kekurangnnya adalah tidak dapat menentukan jumlah *cluster* terbaik dari suatu kumpulan data.

2.3 Davies-Bouldin Index

Kualitas hasil *clustering* dapat diukur dengan cara melakukan evaluasi hasil *clustering.* Davies-Bouldin Index (DBI) diperkenalkan oleh David L. Davies dan Donald W. Bouldin untuk mengevaluasi hasil *clustering* (Prasetyo, 2014). DBI memiliki fokus pada similaritas antar *cluster*. Nilai similaritas ini didasarkan pada *sum of square within cluster* (nilai penyebaran dalam *cluster*) dan *sum of square between cluster* (nilai *dissimilarity* antar *cluster*).

Sum of square within cluster (SSW) diformulasikan oleh persamaan:

$$SSW_i = \frac{1}{m}\sum_{j=1}^{m_j} d(x_j, c_i) \quad \text{..... (2.3)}$$

Sum of square between cluster (SSB) diformulasikan oleh persamaan:

$$SSB_{i,j} = d(c_i, c_j) \quad \text{..... (2.4)}$$

Setelah didapatkan nilai SSW_i dan $SSB_{i,j}$ selanjutnya hitung $R_{i,j}$ yang merupakan ukuran rasio seberapa baik nilai perbandingan antara cluster ke-i dan cluster ke-j. Nilainya didapatkan dari komponen kohesi dan separasi. *Cluster* yang

baik adalah yang mempunyai kohesi yang sekecil mungkin dan separasi yang sebesar mungkin. $R_{i,j}$ diformulasikan oleh persamaan berikut:

$$R_{i,j} = \frac{SSW_i + SSW_j}{SSB_{i,j}} \quad \text{..... (2.5)}$$

Setelah didapatkan nilai $R_{i,j}$ selanjutnya hitung nilai DBI. Semakin kecil nilai DBI (non-negatif ≥ 0) menandakan semakin baik hasil *clustering* yang didapat. DBI diformulasikan oleh persamaan berikut:

$$DBI = \frac{1}{K}\sum_{i=1}^{K} \max(R_{i,j}) \quad \text{........ (2.6)}$$

Keterangan:

m_j adalah jumlah data yang berada dalam *cluster* ke-*i*

x_j adalah data ke-*j*

c_i adalah *centroid cluster* ke-*i*

k adalah jumlah *cluster*

2.4 Euclidean Distance

Euclidean distance merupakan salah satu fungsi yang digunakan dalam pengukuran similaritas antar data dalam *clustering*. Fungsi ini menyatakan jarak diantara dua data poin. *Euclidean distance* menentukan akar kuadrat perbedaan atau jarak diantara dua buah koordinat (Irani, Pise, & Phatak, 2016). *Euclidean distance* diantara dua data poin $X = (x_1, x_2, \dots, x_m)$ dan $Y = (y_1, y_2, \dots, y_m)$ diformulasikan oleh persamaan berikut:

$$d(X,Y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + (x_m - y_m)^2} \quad \text{........ (2.7)}$$

Keterangan:

X adalah data poin ke-1

Y adalah data poin ke-2

x dan y masing-masing adalah koordinat titik X dan Y