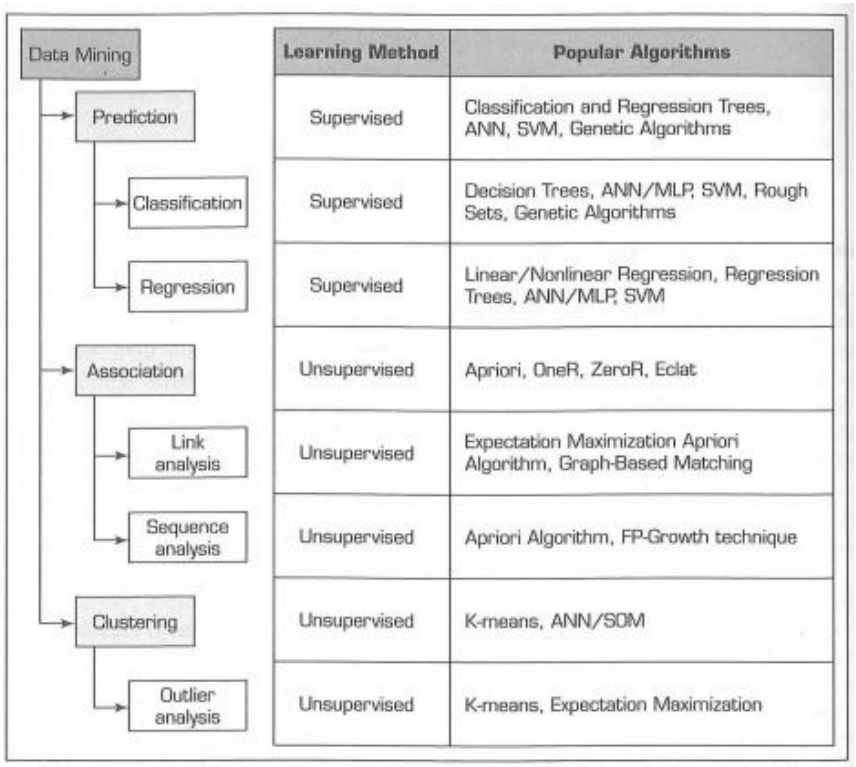
BAB 2
TINJAUAN PUSTAKA

2.1 *Clustering* Dalam *Data Mining*

Data Mining merupakan teknik untuk menemukan suatu pola, informasi, atau juga pengetahuan dari data yang tersedia dalam jumlah yang besar [**2**]. Ilmu mengenai *data mining* ini memiliki peranan penting dalam bidang tertentu. Data mining menggunakan data yang ada dan relevan, *data mining* membangun model untuk mengidentifikasi pola di antara atribut yang ada dalam dataset. Berdasarkan cara untuk mendapatkan pengenalan pola datanya, secara umun kategori pada *Data Mining* dibagi menjadi tiga bagian [**10**].



Learning Method	Popular Algorithms
Supervised	Classification and Regression Trees, ANN, SVM, Genetic Algorithms
Supervised	Decision Trees, ANN/MLP, SVM, Rough Sets, Genetic Algorithms
Supervised	Linear/Nonlinear Regression, Regression Trees, ANN/MLP, SVM
Unsupervised	Apriori, OneR, ZeroR, Eclat
Unsupervised	Expectation Maximization Apriori Algorithm, Graph-Based Matching
Unsupervised	Apriori Algorithm, FP-Growth technique
Unsupervised	K-means, ANN/SOM
Unsupervised	K-means, Expectation Maximization

Gambar 2.1 Data Mining Berdasarkan Cara Pengenalan Pola Data [11**]**

Berdasarkan Gambar 2.1 dapat dijelaskan sebagai berikut ini :

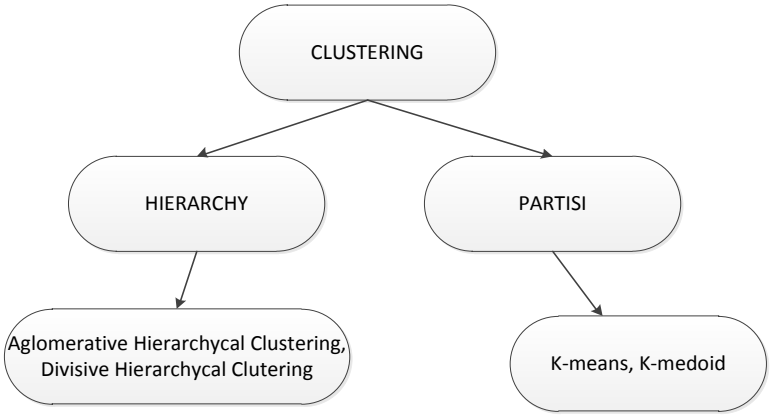
1. *Association* digunakan untuk mencari hubungan pattern(pola) pada suatu data set yang digunakan, dan berusaha menemukan aturan-aturan tertentu yang mengasosiasikan data yang satu dengan data yang lain
2. *Clustering* digunakan untuk mengelompokan data berdasarkan kemiripan antar data sehingga membentuk suatu *cluster* (kelompok), sehingga dalam satu *cluster* memiliki perbedaan dengan cl*uster* lainnya
3. *Prediction* digunakan menganalisis data historis yand disimpan dalam data set dan secara otomatis menghasilkan model yang dapat memprediksi perilaku masa depan.

Clustering dalam *data mining* merupakan salah satu dari kategori *data mining* yang bersifat tanpa arahan *(unsupervised classfication).* Pada *unsupervised classification*, pembelajaran pola untuk pembagian *class* pada data tidak diberikan, sehingga lebih banyak berfokus pada memahami pola dalam *cluster*, dan untuk menemukan persamaan dan perbedaan antar data. *Clustering* sangat cocok digunakan untuk pengelompokan data yang *class* memang sulit didapatkan pada saat pembangkitan fitur *class* data. Setelah mengamati hasil dari proses *cluster*, maka akan dibuat label *class* baru pada data berdasarkan karakteristik hasil *cluster* tersebut. Contoh penerapan *clustering* dalam dunia nyata yaitu dapat digunakan untuk melakukan segmentasi pasar [**12**]. Melalui segmentasi pelanggan, perusahaan dapat mengetahui karakteristik pelanggan, sehingga perusahaan dapat menentukan target pemasaran serta pelayanan terhadap pelanggan yang sesuai. Dalam *clustering* segmentasi pelanggan diperlukan

algoritma *clustering* yang tepat sehingga hasil dari proses *cluster* tepat untuk dijadikan bahan keputusan bisnis.

2.2 Teknik Dalam *Clustering*

Teknik dalam *clustering* terdapat dua cara yaitu dengan cara *hierarchy* dan *partition*, teknik *clustering* tersebut terdapat beberapa algoritma yang masuk didalamnya. Teknik *clustering hierarchy* terdapat algoritma Hierarchycal Aglomerative *Clustering* (HAC) dan Hierarchycal Divisive *Clustering*(HDC), sedangkan teknik *clustering* partisi terdapat algoritm K-means, K-medoids, dan variasi dari keduannya.



Gambar 2.2 Teknik Dalam *Clustering* [13**]**

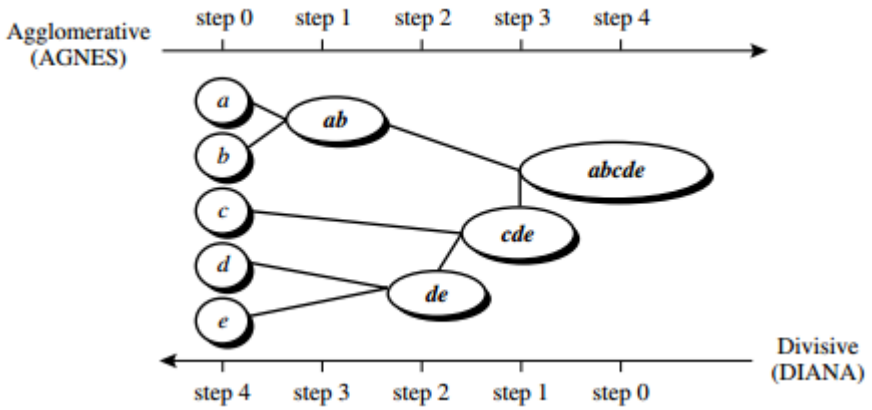
1. *Clustering* Hirarki

Dalam teknik *hierarchical clustering*, satu data 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*.

Terdapat algoritma yang termasuk pada *clustering* hirarki yaitu *Hierarchical Agglomerative clustering* (HAC) dan *Hierarchical Divisive Clustering* (HDC)

Algoritma HAC merupakan algoritma *clustering* berbasis hierarki dengan pendekatan *bottom-up*, prosesnya 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 [**14**]. Sementara untuk algoritma *Hierarchical Divisive Clustering* (HDC) merupakan proses kebalikan dari HAC.

Algoritma HAC dan HDC keduanya mengorganisasi data ke dalam struktur hirarki berbasis matrik, hasil dari *hierarcichal clustering* digambarkan dalam bentuk *binary tree* ataupun *dendogram*, *root* merupakan keseluruhan dataset dan tiap cabang merupakan data point, *clustering* akhir dapat diperoleh dari pemotongan *dendogram* pada level-level yang sesuai. Di Gambar 2.3 contoh pembentukan *cluster* menggunakan algoritma HAC dan HDC



Gambar 2.3 Agglomerative and Divisive Hierarchical *Clustering* Pada Data Objek {a, b, c, d, e} [13**].**

Berdasarkan Gambar 2.3 menjelaskan terdapat data a,b,c,d,e. Pada pendekatan *alglomerative*, masing – masing data pada awalnya menjadi *cluser* tersendiri, sehingga awalnya jumlah *cluster* = 5. Masing-masing data kemudian dihitung jaraknya terhadap data yang lain untuk dibentuk lagi menjadi *cluster* tertentu. Misalkan data a dihitung jaraknya ke data (b,c,d,e), sehingga data a dan b akan menjadi *cluster* tersendiri karena dilihat berdasarkan jarak yang paling kecil.

Dalam penggabungannya, terdapat beberapa cara untuk menghitung jarak data ke data lain yaitu dengan menggunakan *single lingkage* (mengambil jarak terkecil), *complete lingkage* (mengambil jarak terbesar), dan *average linkage* (mengambil nilai rata-rata). Data yang telah menjadi *cluster* tersendiri akan dibandingkan lagi dengan *cluster* lain, hal itu dilakukan berulang sampai menjadi satu *cluster* untuk seluruh data.

2. *Clustering* Partisi

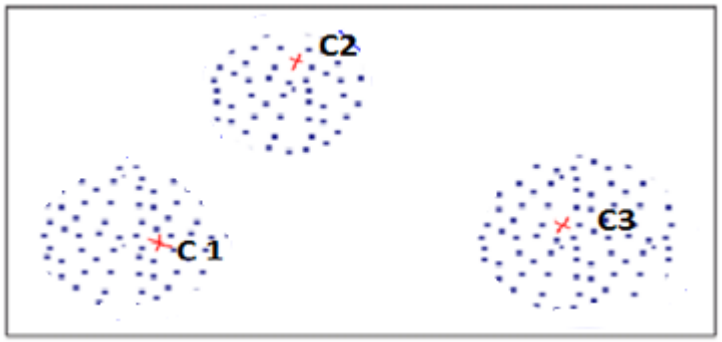
Berikut ini yang termasuk teknik *clustering* partisi salah satunya adalah algoritma K-means, dan K-medoid. Dynamic K-means, dan K-means Binary Search Centroid (KBSC) merupakan pengembangan dari algoritma K-means. Dalam *clustering* partisi ada beberapa hal yang perlu diperhatikan untuk proses *cluster* nya yaitu :

1. Penentuan Jumlah *Cluster*

Jumlah *cluster* atau disebut juga jumlah *k*, ditentukan diawal pada proses *cluster*. Penentuan jumlah *cluster* ini sangat memiliki pengaruh terhadap hasil akhir pembentukan pada proses *cluster*.

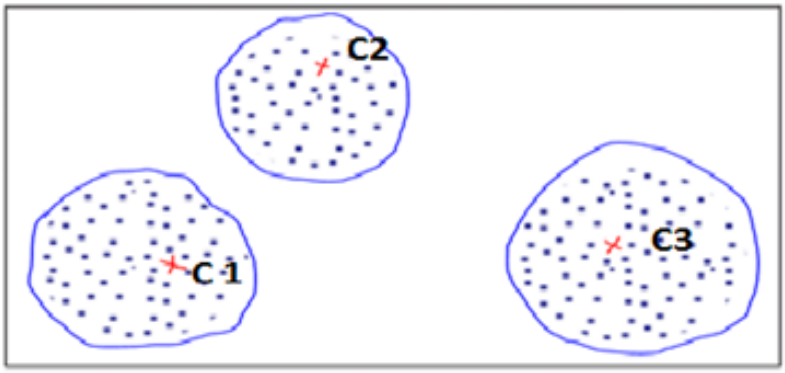
2. Titik *Centroid* Data

Merupakan titik pusat data yang mewakili dari pembentukan *cluster* yang dihasilkan. Titik *centroid* data memiliki jumlah sesuai dengan jumlah *cluster* yang ditentukan. Di Gambar 2.4 titik *centroid* dijelaskan pada tanda silang berwarna merah sebagai berikut ini :



Gambar 2.4 Contoh Titik *Centroid* Pada Tiga Buah *Cluster*

Pada proses *cluster* maka setiap data akan dikelompokan berdasarkan kemiripan yang diukur menggunakan perhitungan jarak masing-masing data terhadap titik *centroid* C1,C2, dan C3, kemudian diambil jarak yang terkecil. Proses pengelompokan dilakukan secara berulang sehingga menghasilkan kelompok yang tetap. Berikut ini hasil *cluster yang* terbentuk berdasarkan kedekatan data terhadap centroid :

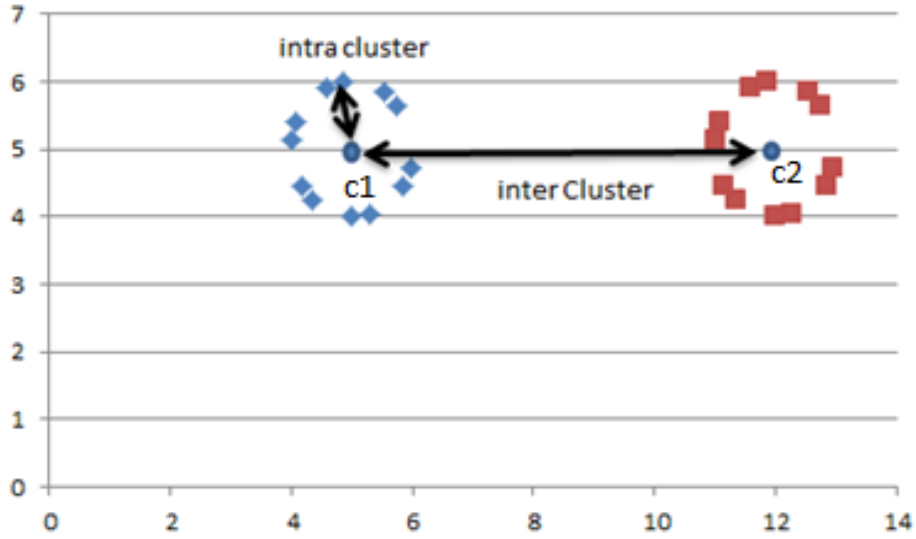


Gambar 2.5 Hasil 3 *Cluster* Berdasarkan Kemiripan Data Dengan *Centroid* C1,C2 dan C3

Berdasarkan Gambar 2.5 menjelaskan setiap data pada masing-masing *cluster* memiliki *centroid* C1,C2,dan C3 yang merupakan rata-rata dari data yang masuk pada *cluster*.

3. Intra dan Inter Cluster

Intra dan Inter cluster merupakan tingkat kemiripan antara data dalam satu *Cluster* dan juga dengan *cluster* lainnya. Berikut ini penjelasan mengenai jarak *intra* dan *inter cluster* yang menggambarkan kemiripan data terhadap titik *centroid* dalam satu *cluster*, dan perbedaan terhadap *cluster* yang lain :



Gambar 2.6 Jarak *Intra* dan *Inter Cluster*

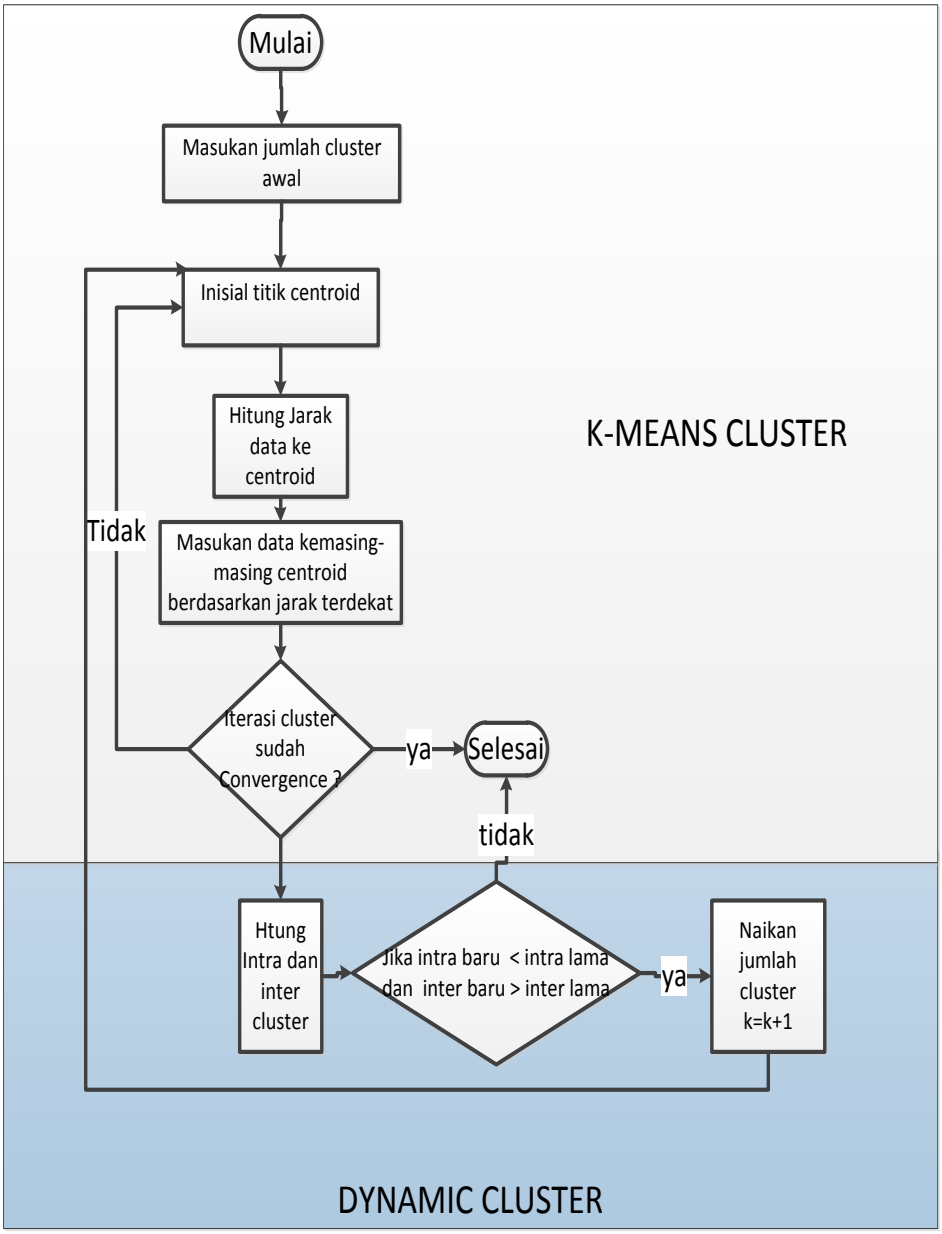
Intra cluster merupakan jarak antara data ke titik *centroid*, sedangkan *inter cluster* merupakan jarak antar titik *centroid*. Hasil *cluster* yang baik yaitu *high intra class similarity* (memiliki tingkat kemiripan yang *maximum*) pada satu *cluster* yang sama, dan karakteristik dari inter *cluster* adalah *law inter class similarity* (memiliki tingkat kemiripan yang *minimum*) pada *cluster* yang lain [**13**].

2.2.1 Algoritma Dynamic K-means (DK)

Dynamic K-means (DK) adalah algoritma yang dikembangkan dari algoritma K-means. Cara kerjanya sama dengan algoritma K-means, namun ketika anggota pada satu *cluster* sudah tidak berubah atau tidak berpindah ke *cluster* lain dilakukan perhitungan jarak *intra* dan *inter cluster*. Jika jarak *intra* semakin kecil dan jika jarak *intra* semakin besar, maka algoritma akan menghitung *cluster* baru

dengan menambahkan jumlah *cluster* dengan satu atau *k=k+1* disetiap iterasi [**15**].

Hasil *clustering* untuk algoritma Dynamic K-means sangat dipengaruhi oleh titik *centroid* data yang dibuat pada awal proses *clustering* karena pada proses pembentukan *cluster* berdasarkan titik centroid yang dibuat, akan berpengaruh pada jarak *intra* dan *inter cluster*. Berikut ini merupakan langkah dari algoritma Dynamic K-means yang dijelaskan di Gambar 2.7.



Gambar 2.7 Flowchart Algoritma Dynamic K-means

Penjelasan flowchart di Gambar 2.7 adalah sebagai berikut :

1. Masukan jumlah *cluster* (*k*).
2. Menentukan titik *centroid* data.

Penentuan awal titik *centroid* data *cluster* ini bisa dilakukan dengan berbagai cara, namun yang paling sering dilakukan adalah dengan cara acak oleh proses algoritma.

3. Kelompokan semua data atau objek yang memiliki kemiripan dengan cara menghitung jarak ke titik *centroid* terdekat. Jarak ini diartikan sebagai kesamaan suatu objek data dengan titik *centroid* yang telah ditentukan. Kedekatan dua objek ditentukan berdasarkan perhitungan jarak objek tersebut. Untuk menghitung jarak semua data ke setiap titik *centroid cluster* dapat menggunakan teori jarak *Euclidean Distance* yang dirumuskan sebagai berikut :

$$D(i,j) = \sqrt{(x_{1i} - x_{1j})^2 + (x - x_{2j})^2 + \cdots + (x - x_{kj})^2} \ldots\ldots\ldots\ldots (2.1)$$

Keterangan :

D(i, j) : Jarak data ke i ke *centroid Cluster* j

x_{kt} : Data ke i pada variabel data ke *k*

x_{kj} : Titik *centroid* ke j pada variabel ke *k*

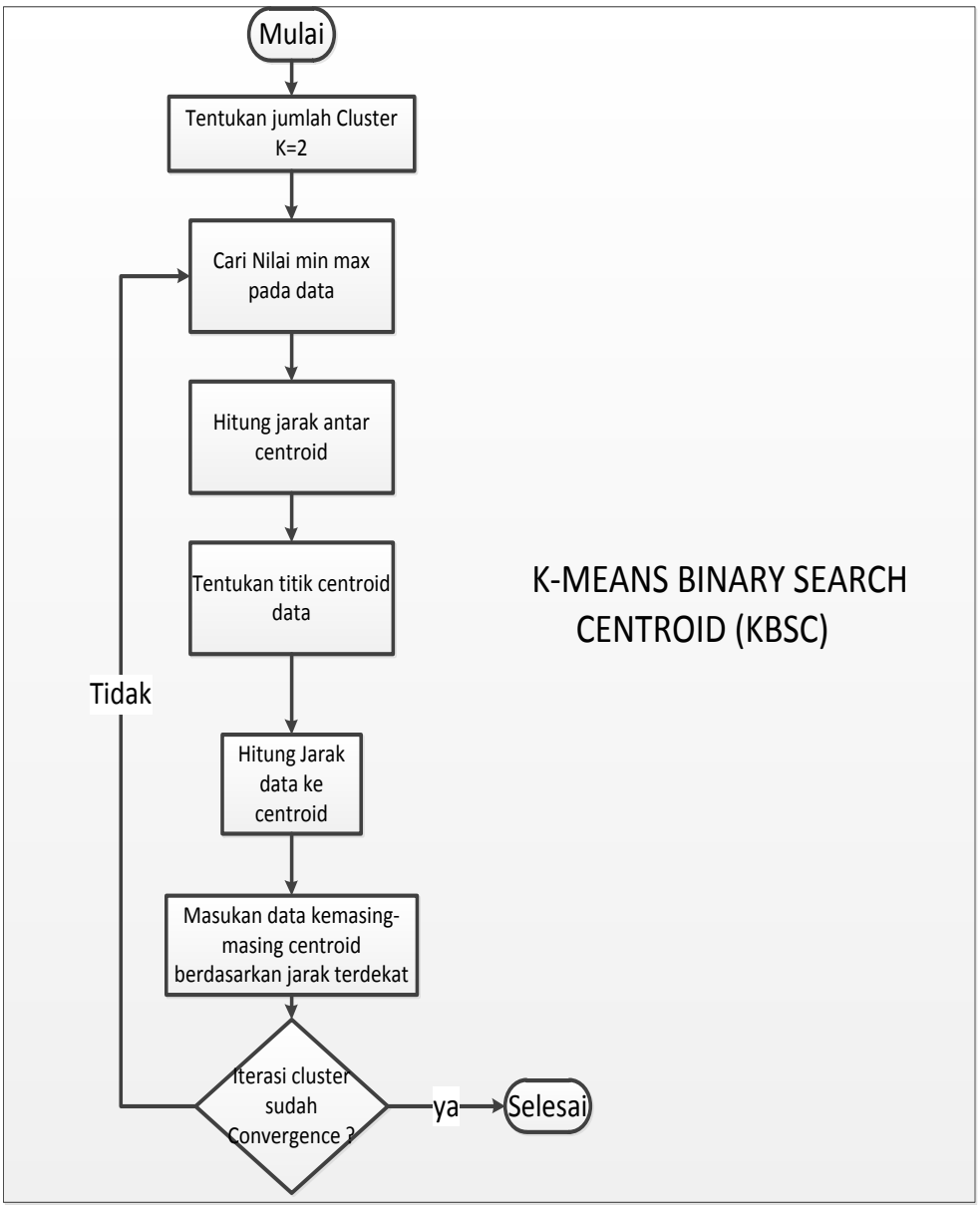
4. Hitung kembali titik *centroid* baru berdasarkan keanggotaan yang terbentuk. Titik *centroid* baru adalah rata-rata dari semua data atau objek dalam *cluster* tertentu.
5. Jika anggota *cluster covergence* tidak berubah maka lakukan pada proses hitung intra *cluster* dan juga inter *cluster*.
6. Jika *intra* baru < *intra* lama dan *inter* baru > *inter* lama, maka jumlah *cluster* akan ditambahkan dengan *k=k+1* dan kembali pada proses kedua menentukan

titik *centroid* data dan mengalokasikan data. Jika tidak maka proses *cluster* akan berhenti pada k=k.

Kelebihan dari algoritma Dynamic K-means yaitu dalam penentuan jumlah *cluster*, jumlah *cluster* yang dihasilkan berdasarkan perbandingan nilai *intra* dan *inter cluster*. Namun kelemahan dari Dynamic K-means adalah ketika dalam proses pembentukan titik *centroid*, jika titik *centroid* dilakukan secara acak maka akan berpengaruh pada nilai *intra* dan *inter cluster* yang menyebabkan pembentukan untuk jumlah *cluster* akan berubah-ubah tidak konsisten, dan proses *cluster* memungkinkan akan terjadi kondisi yang membutuhkan iterasi yang banyak untuk membentuk *cluster convergence*.

2.2.2 Algoritma K-means Binary Search Centroid (KBSC)

KBSC merupakan pengembangan dari algoritma *K-mean*s dengan cara penggabungan antara algoritma K-means dengan algoritma tambahan pada area proses pembentukan titik *centroid* data menggunakan pendekatan *Binary Search* [**8**]. Algoritma KBSC ini dibuat oleh Hatamlao [**16**]. Berikut ini merupakan gambar dari *flowchart* mengenai lagkah-langkah pada algoritma KBSC.



Gambar 2.8 Flowchart K-means Binary Search Centroid

Penjelasan Gambar 2.8 adalah sebagai berikut ini :

1. Tentukan jumlah *cluster* (k) nilai k=1, 2, 3… m.
2. Hitung nilai maximum dan minimum pada data untuk masing-masing atribut data.

3. Hitung range antara titik *centroid*, perhitungan range antar titik *centroid* pada suatu data dengan cara berikut ini :

$$M = \frac{max(a_i) - min(a_i)}{k} \ldots\ldots\ldots\ldots (2.2)$$

Berdasarkan rumus 2.2 digunakan untuk menghitung suatu nilai pada variabel M yang merupakan jarak spesifik antar titik *centroid data* sebelum memberikan hasil titik *centroid data*. $min(a_i)$ nilai minimal dari masing-masing atribut data, $max(a_i)$ merupakan nilai maksimum dari masing-masing atribut data, *k* adalah jumlah *cluster* yang akan dibentuk.

4. Tentukan titik *centroid* data, Untuk menghasilkan titik *centroid* menggunakan rumus sebagai berikut :

$$C_k = \min(a_i) + (k-1)M \ldots\ldots\ldots\ldots (2.3)$$

Berdasarakan rumus 2.3 , C_k merupakan *centroid* untuk *cluster k*, $\min(a_i)$ nilai minimum data, dan M range antara titik *centroid* data.

5. Lakukan pengelompokan data dengan cara hitung jarak data terhadap titik *centroid* yang terbentuk Ck. Perhitungan jarak menggunakan *Euclidiance distance* pada rumus :

7. Hitung kembali titik *centroid* berdasarkan *cluster* yang terbentuk, Kemudian lakukan pengelompokan ulang sampai menemukan *cluster* tidak berubah kembali *(convergence)*

Berikut ini merupakan contoh perhitungan titik *centroid* menggunakan algoritma KBSC *:*

Tabel 2.1 Contoh Data Perhitungan Pada Algoritma KBSC

Data ke	Variabel 1	Variabel 2
1	1.1	4.3
2	1.3	3.9
3	1	3.8
4	2	4.8
5	2.1	3.9
6	2.2	3.7
7	2	3.6
8	1.9	6.9
9	2.2	3.2
10	9	3.2

Alur perhitungan titik *centroid* data sebagai berikut :

1. Pada contoh data akan dibentuk jumlah *cluster* sebanyak k=3
2. Hitung nilai maximum dan minimun pada masing-masing variabel pada data.

   Berikut ini nilai maximum pada variable 1 dan 2 :

   Di variabel 1 = 9

   Di variabel 2 = 6.9

   Berikut ini nilai minimum pada variable 1 dan 2 :

   Di variabel 1 = 1.1

   Di variabel 2 = 3.2
3. Hitung Nilai dari range antar titik *centroid*

$$M = \left\{\frac{(9 - 1.1)}{3}, \frac{(6.9 - 3.2)}{3}\right\}$$

$$M = (2.63, 1.23)$$

4. Hitung titik titik *centroid* data berdasarkan jumlah *cluster* k yang dibentuk, hasil titik *centroid* data sebagai berikut :

$$C1 = \left(1.1 + \left((1-1) * 2.63\right), 3.2 + \left((1-1) * 1.23\right)\right)$$

$$= (1.1\ , 3.2)$$

$$C2 = \left(1.1 + \left((2-1) * 2.63\right), 3.2 + \left((2-1) * 1.23\right)\right)$$

$$= (3.73\ , 4.43)$$

$$C3 = \left(1.1 + \left((3-1) * 2.63\right), 3.2 + \left((3-1) * 1.23\right)\right)$$

$$= (6.36{,}5.66)$$

Kelebihan dari algoritma KBSC adalah centroid yang dihasilkan memiliki nilai dari jarak *intra cluster* yang lebih kecil dibandingkan algoritma K-means penelitian yang dilakukan oleh Yugar Kumar dan G.Sahoo [**8**]. Sementara kekurangnnya adalah tidak dapat menentukan jumlah *cluster* terbaik dari suatu kumpulan data.

2.3 Evaluasi Hasil *Clustering*

Untuk mengetahui kualitas hasil *cluster* pada data *clustering* maka perlu dievaluasi menggunakan metode evaluasi yang sudah ada. Salah satu evaluasi *clustering* adalah dengan menggunakan Devies Bouldin Index (DBI) untuk mengukur kemiripan [**14**]. DBI memiliki fokus pada kemiripan antar *cluster*, nilai kemiripan didasarkan pada nilai penyebaran dalam *cluster* dan nilai perbedaan antar *cluster*.

Nilai penyebaran dalam satu *cluster* diformulasikan oleh persamaan:

$$SSW_i = \frac{1}{m}\sum_{j=1}^{m_j} d(x_j, c_i) \quad \text{..... (2.4)}$$

Nilai perbedaan antar c*luster* diformulasikan oleh persamaan:

$$SSB_{i,j} = d(c_i, c_j) \quad \text{..... (2.5)}$$

Didefinisikan $R_{i,j}$ adalah ukuran rasio seberapa baik nilai perbandingan antara *cluster* ke-*i* dan *cluster* ke-*j*. Nilainya didapatkan dari komponen kerapatan antar data dan separasi antar data. $R_{i,j}$ diformulasikan oleh persamaan berikut:

$$R_{i,j} = \frac{SSW_i + SSW_j}{SSB_{i,j}} \quad \text{..... (2.6)}$$

$$DBI = \frac{1}{K}\sum_{i=1}^{K} \max(R_{i,j}) \quad \text{.....(2.7)}$$

Keterangan:

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

c_i : adalah *centroid cluster* ke-*i*

k : adalah jumlah *cluster*