BAB III

METODE PENELITIAN

3.1 Sumber Data dan Variabel Penelitian

Data yang digunakan dalam penelitian ini adalah data mengenai kemiskinan yang terjadi di 221 kota/kabupaten yang tersebar di 16 Provinsi di Indonesia pada tahun 2020. Adapun rincian mengenai ke-16 provinsi tersebut dapat dilihat pada tabel 3.1. Data tersebut merupakan data sekunder yang bersumber dari Badan Pusat Statistik (BPS) setiap provinsi dan dapat diunduh melalui website resminya. Adapun parameter atau variabel yang digunakan dalam mengukur tingkat kemiskinan suatu kota/kabupaten yaitu persentase penduduk miskin (X_1), indeks kedalaman kemiskinan (X_2), dan indeks keparahan kemiskinan (X_3). Selain itu, pada penelitian ini proses pengolahan data akan dilakukan dengan bantuan aplikasi RStudio dan SPSS.

Tabel 3.1 Daftar Nama 16 Provinsi yang Digunakan dalam Penelitian

No	Nama Provinsi	Jumlah	
		Kota	Kabupaten
1	Jawa Barat	9	18
2	Jawa Tengah	6	29
3	DI Yogyakarta	1	4
4	Sumatera Barat	7	12
5	Sumatera Selatan	4	13
6	Kepulauan Riau	2	5
7	Kepulauan Bangka Belitung	1	6
8	Bali	1	8
9	Nusa Tenggara Timur	1	21
10	Kalimantan Barat	2	12
11	Kalimantan Selatan	2	11

No	Nama Provinsi	Jumlah	
		Kota	Kabupaten
12	Kalimantan Timur	3	7
13	Kalimantan Utara	1	4
14	Sulawesi Utara	4	11
15	Gorontalo	1	5
16	Maluku Utara	2	8
Jumlah Kota/Kabupaten		47	174
Jumlah total		221	

3.2 Metode Analisis Data

Penelitian ini menggunakan metode analisis klaster yang bertujuan untuk mengelompokkan objek ke dalam beberapa klaster berdasarkan kesamaan karakteristik yang dimiliki oleh objek – objek tersebut. Adapun metode atau algoritma yang digunakan dalam melakukan analisis klaster pada penelitian ini yaitu *k-medoids*. *K-medoids* merupakan salah satu metode analisis klaster berbasis partisi. Hal ini karena *k-medoids* bekerja dengan cara membagi atau mempartisi kumpulan data besar ke dalam beberapa kelompok data yang lebih kecil dengan mencari *k* objek representatif (*medoid*) yang terletak di pusat klaster berdasarkan suatu ukuran tertentu. *K-medoids* memiliki kelebihan untuk mengatasi kelemahan pada algoritma *k-means* yang sensitif terhadap pencilan (Suyanto, 2019). Selain itu, kelebihan metode ini adalah klaster yang terbentuk tidak bergantung pada urutan objek yang diperiksa dan hasil klasterisasi tidak akan berubah sehubungan dengan dilakukannya transformasi ortogonal pada titik data (Ng & Han, 2002). Kelemahan dari metode ini adalah karena di awal tahap dipilih *k* objek secara acak untuk menjadi objek representatif maka pengelompokan data yang dihasilkan dapat berbeda-beda, sehingga jika inisialisasi di awal kurang baik, maka pengelompokkan yang dihasilkan menjadi kurang optimum. Selain itu khusus untuk algoritma *Partitioning Around Medoids* (PAM), algoritma ini hanya dapat bekerja dengan efektif pada data dengan jumlah kecil hingga menengah (Han, Kamber, & Pei, 2012).

Inti dari metode *k-medoids* yaitu membentuk k klaster dari n objek yang ada dengan langkah awal memilih secara acak k objek representatif dari suatu himpunan data D untuk tiap-tiap klaster. Objek yang mewakili setiap klaster disebut dengan *medoid*. Klaster dibangun dengan menghitung ketidaksamaan yang dimiliki antara *medoid* dengan objek *nonmedoid* menggunakan ukuran ketidaksamaan. Ukuran ketidaksamaan yang digunakan adalah ukuran jarak dan jenis jarak yang digunakan yaitu jarak Euclid. Formula untuk menghitung jarak Euclid dapat dilihat pada persamaan (2.8). Formula tersebut digunakan untuk menghitung *entri-entri* matriks jarak dari data multivariat yang diberikan, sehingga diperoleh matriks perbedaan (*dissimilarity matrix*) sebagai berikut (Suyanto, 2019):

$$D_{n\times n} = \begin{bmatrix} 0 & d_{12} & \cdots & d_{1j} & \cdots & d_{1n} \\ d_{21} & 0 & \cdots & d_{2j} & \cdots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ d_{i1} & d_{i2} & \cdots & 0 & \cdots & d_{in} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & \cdots & d_{nj} & \cdots & 0 \end{bmatrix} \qquad (3.1)$$

Algoritma *k-medoids* melakukan partisi dengan cara meminimumkan jumlah ketidaksamaan antara setiap objek p dan objek representatif terdekat yaitu menggunakan jumlah kesalahan mutlak atau yang sering disebut juga sebagai total biaya (*cost*). Adapun rumus untuk menghitung total biaya yaitu sebagai berikut (Han, Kamber, & Pei, 2012):

$$E = \sum_{i=1}^{k} \sum_{p\in C_i} d(p, o_i) \qquad (3.2)$$

Keterangan:

E : jumlah kesalahan mutlak / total biaya

p : objek nonrepresentatif / *nonmedoid* dari klaster C_i

o_i : objek representatif / *medoid* dari klaster C_i

Berikut ini akan dibahas dua algoritma yang menerapkan metode *k-medoids.*

3.2.1 Partitioning Around Medoids (PAM)

Algoritma pertama yang menerapkan *k-medoids* adalah *Partitioning Around Medoids* (PAM). Algoritma ini dikembangkan oleh Kaufman dan Rousseeuw pada tahun 1990. Untuk membentuk k-klaster dari n objek data, PAM bekerja dengan memilih sebanyak k objek secara acak sebagai objek representatif (*medoid*) awal. Kemudian objek lain yang tidak terpilih menjadi *medoid* atau yang disebut objek *nonmedoid* dimasukkan ke dalam klaster yang memiliki objek representatif terdekat di antara k objek representatif yang ada. Lalu kemudian akan dilakukan penggantian sebuah objek *medoid* dengan objek *nonmedoid* guna meningkatkan kualitas klasterisasi. Proses penggantian *medoid* ini melibatkan semua kemungkinan objek *nonmedoid* dan dilakukan secara berulang hingga konvergen (Suyanto, 2019).

Berikut ini akan dipaparkan secara rinci algoritma PAM (Sihombing, Rachmatin, & Dahlan, 2019):

1) Pilih secara acak sejumlah k objek dari n objek yang ada sebagai inisialisasi *medoid* awal. Misalkan setiap *medoid* ditulis sebagai O_i
2) Hitung jarak setiap objek terhadap setiap *medoid* dan tempatkan setiap objek ke klaster yang terdekat dengan *medoid*-nya.
3) Pilih secara acak sebuah objek *nonmedoid* sebagai O_{random}.
4) Hitung total biaya (*cost*) dan S dari pertukaran medoid O_i dan O_{random}.
5) Jika S < 0, maka tukar O_i dengan O_{random} untuk menjadi *medoid* baru.
6) Ulangi langkah 2-5 sampai tidak ada perubahan.

Formula untuk menghitung total biaya (*cost*) dapat dilihat pada persamaan (3.2). Adapun S dinyatakan sebagai berikut:

$$S = Total\ cost\ baru - Total\ cost\ lama \qquad (3.3)$$

dengan:

Total cost baru = jumlah biaya (*cost*) *nonmedoid*

Total cost lama = jumlah biaya (*cost*) *medoid*

Karena PAM memperhitungkan semua objek *nonmedoid* dalam proses penggantian objek *medoid* maka iterasinya memerlukan waktu yang lama. Oleh karena itu, algoritma ini hanya dapat bekerja dengan efektif pada data dengan jumlah kecil hingga menengah yaitu dibawah 100 (Kaufman & Rousseeuw, 1990).

3.2.2 Clustering Large Application based on RANdomized (CLARANS)

Algoritma CLARANS dikembangkan oleh Raymond T. Ng dan Jiawei Han pada tahun 2002. CLARANS bekerja dengan mengkombinasikan teknik *sampling* dan algoritma PAM. CLARANS menggunakan abstraksi *graph* dalam menemukan *k-medoids*. Misal terdapat n objek dalam himpunan data D, maka untuk menemukan *k-medoids* dapat dilihat melalui sebuah *graph* tertentu yang dinotasikan dengan $G_{n,k}$. Sebuah *node* dalam *graph* menggambarkan sebuah himpunan yang berisi *k-medoids* terpilih yaitu $\{O_{m1}, O_{m2}, \dots, O_{mk}\}$ dan himpunan *node* dalam sebuah *graph* dinotasikan dengan $\{\{O_{m1}, O_{m2}, \dots, O_{mk}\} | O_{m1}, O_{m2}, \dots, O_{mk} \in D\}$.

Dua buah *node* dikatakan bertetangga jika kedua *node* tersebut hanya memiliki satu objek atau satu *medoid* yang berbeda. Hal ini berarti banyaknya objek atau *medoid* yang saling beririsan antara dua buah *node* yang saling bertetangga yaitu $k-1$ dan setiap *node* memiliki $k(n-k)$ tetangga. Kemudian *cost* dari setiap *node* yaitu total ketidaksamaan antara setiap objek dengan *medoid* klasternya dan perbedaan *cost* antara 2 *node* yang bertetangga dapat dihitung menggunakan rumus yang ada pada persamaan (3.2).

Menurut Raymond T. Ng dan Jiawei Han (2002), algoritma CLARANS lebih efektif dan efisien dibandingkan algoritma PAM dan CLARA dalam melakukan analisis klaster, baik pada data berukuran kecil maupun besar. Hal ini karena CLARANS hanya melakukan pemeriksaan

terhadap sampel tetangga saja dari suatu *node* sehingga proses lebih cepat dibanding PAM. Selain itu, pada CLARANS pencarian tidak terbatas pada suatu *subgraph* tertentu atau pada area lokal saja, sehingga kualitas klasterisasi yang dihasilkan lebih baik dibanding CLARA. Oleh karena itu, pada penelitian ini algoritma yang digunakan yaitu algoritma CLARANS.

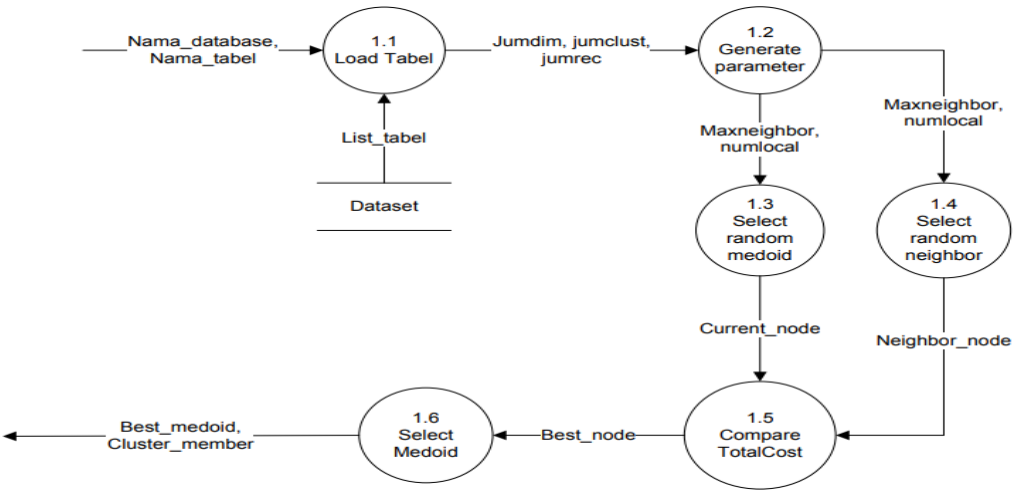
CLARANS memiliki dua parameter utama, yaitu *maxneighbor* dan *numlocal*. *Maxneighbor* adalah jumlah maksimal *node* tetangga (*neighbor*) yang diperiksa dari sebuah *node* yang terpilih, sedangkan *numlocal* adalah jumlah minimum lokal yang menentukan berapa kali *node* diperiksa atau menentukan berapa kali iterasi dilakukan. Semakin tinggi nilai *maxneighbor* maka semakin dekat CLARANS ke PAM dan semakin lama setiap pencarian minimum lokal, tetapi kualitas minimum lokal semakin baik (Ng & Han, 2002).

Berdasarkan penelitian yang dilakukan oleh Raymond T. Ng dan Jiawei Han, nilai *maxneighbor* yang digunakan yaitu $p\% \times k(n-k)$ dan nilai *numlocal* yang digunakan yaitu 2. Menurut keduanya, nilai p antara 1,25 dan 1,5 dimana k menyatakan banyaknya klaster yang terbentuk dan n menyatakan jumlah objek dalam himpunan data. Nilai – nilai tersebut dipilih guna menjaga keseimbangan antara *runtime* (waktu pengoperasian) dan kualitas klaster yang terbentuk. Perlu diingat bahwa semakin tinggi nilai *maxneighbor* dan *numlocal* maka semakin lama *runtime* yang diperlukan, tetapi semakin tinggi kualitas klaster yang dihasilkan.

Misalkan terdapat n objek dengan p variabel dalam suatu himpunan data D. Kemudian akan dibentuk k klaster ($k < n$), di mana nilai k diberikan. Jarak yang digunakan dalam menghitung ketidaksamaan antar objek yaitu jarak Euclid. Rumus untuk menghitung jarak dan *cost* dapat dilihat pada persamaan (2.8) dan (3.2). Berikut ini merupakan langkah - langkah algoritma CLARANS (Ng & Han, 2002):

1) Masukkan parameter *numlocal* dan *maxneighbor*. Inisialisasi $i\ dengan\ 1$ dan *mincost* dengan angka yang besar
2) Pilih secara acak sebuah *node* di *graph* $G_{n,k}$
3) Untuk $j = 1$
4) Pilih secara acak *node* tetangga (N) dari *node* yang terpilih (M). Kemudian hitung *cost* dari 2 *node* tersebut (*node* N dan M) berdasarkan persamaan (3.2)
5) Jika N mempunyai *cost* lebih rendah dibanding M, maka tukar *node* M dengan *node* N dan kembali ke langkah (3)
6) Sebaliknya, jika N mempunyai *cost* lebih tinggi dibanding M maka naikkan $j \rightarrow j = j + 1$. Lalu cek jika $j \leq maxneighbor$ maka kembali ke langkah (4)
7) Sebaliknya, jika $j > maxneighbor$ bandingkan *cost* antara *node* M dengan *mincost*. Jika *cost* $M < mincost$ maka *mincost* = *cost* M dan *node* terbaik = M
8) Lalu naikkan $i \rightarrow i = i + 1$. Jika $i > numlocal$ maka output = *node* terbaik dan proses berhenti. Namun jika $i < numlocal$ maka kembali ke langkah (2)

Agar lebih mudah memahami algoritma di atas, coba perhatikan ilustrasi proses algoritma CLARANS pada gambar 3.1 berikut ini!



Gambar 3.1 Proses Algoritma CLARANS

Proses analisis klaster dengan algoritma CLARANS pada dataset besar akan memakan waktu yang lama jika dilakukan secara manual dan lebih berisiko terjadi kesalahan dalam proses perhitungan. Oleh karena itu peneliti menggunakan bahasa pemrograman R yang dirancang guna membentuk sebuah program aplikasi pengelompokkan objek berdasarkan algoritma CLARANS. Untuk membangun program tersebut peneliti melakukan studi literatur terlebihdahulu. Beberapa sumber yang menjadi acuan penulis yaitu:

1) Skripsi karya Rizkiana Amalia dengan judul "Analisis Deteksi Outlier Menggunakan CLARANS"
2) Tulisan Tri Binty pada website medium.com yang berjudul "K-Medoids/Partitioning Around Medoids (PAM) – Non Hierarchical Clustering with R"
3) *Ebook* yang berjudul "Belajar Statistika dengan R" karya Prana Ugiana Gio dan Dasapta Erwin Irawan.

3.2.3 Simulasi Penerapan Algoritma CLARANS

Agar lebih memahami algoritma CLARANS pada pembahasan di atas, berikut akan diberikan contoh penerapan algoritma CLARANS pada dataset kecil.

Misal diberikan data nilai Matematika dan Fisika dari 5 siswa SMA X yang dipilih secara acak. Kemudian akan dibentuk 2 klaster berdasarkan nilai - nilai tersebut. Berikut ini merupakan nilai dari kelima siswa tersebut.

Tabel 3.2 Nilai Matematika dan Fisika Siswa

Siswa	Nilai Matematika	Nilai Fisika
A	3	4
B	4	7
C	6	2
D	6	4
E	7	4

Misal diketahui:

$k = 2, maxneighbor = 2,\ numlocal = 1,$ dan $mincost = \infty$

Karena $maxneighbor = 2$ dan $numlocal = 1$ maka jumlah maksimum *node* tetangga yang diperiksa dari suatu *node* terpilih yaitu 2 dan iterasi dilakukan 1 kali.

Iterasi satu

1) Dipilih secara acak sebuah *node* yaitu (A,E). Kemudian dipilih secara acak sebuah *node* tetangga dari (A,E) yaitu (A,D). Hitung jarak pusat klaster dan *cost* dari kedua *node* tersebut dengan menggunakan rumus pada persamaan (2.8) dan (3.2)
   a) Dicari semua jarak antar objek dengan menggunakan jarak Euclid sehingga diperoleh matriks jarak sebagai berikut:

Matriks Jarak

$$D_{5\times5} = \begin{bmatrix} 0 & 3{,}1623 & 3{,}6056 & 3 & 4 \\ 3{,}1623 & 0 & 5{,}3852 & 3{,}6056 & 4{,}2426 \\ 3{,}6056 & 5{,}3852 & 0 & 2 & 2{,}2361 \\ 3 & 3{,}6056 & 2 & 0 & 1 \\ 4 & 4{,}2426 & 2{,}2361 & 1 & 0 \end{bmatrix}$$

b) Kemudian akan dilakukan penempatan objek *nonmedoid* yaitu B,C,D ke dalam klaster terdekat yang medoidnya A dan E berdasarkan ukuran jarak.

Tabel 3.3 Penempatan Objek *Nonmedoid* B, C, dan D

Objek	**Jarak antara Objek *Medoid* dan *Nonmedoid***		**Penempatan**	
	A (C_1)	**E(C_2)**	**A(C_1)**	**E(C_2)**
B	3,1623	4,2426	1	0
C	3,6056	2.2361	0	1
D	3	1	0	1

Keterangan penempatan:

1 menyatakan objek tersebut berada dalam grup atau klaster tersebut.

0 menyatakan objek tersebut tidak berada dalam grup atau klaster tersebut.

Berdasarkan tabel diatas maka diperoleh informasi sebagai berikut:

Klaster 1 : Objek A dan B

Klaster 2: Objek C, D, dan E

c) Lakukan hal yang sama seperti langkah (b) untuk *node* tetangga (A,D) sehingga diperoleh hasil sebagai berikut:

Tabel 3.4 Penempatan Objek *Nonmedoid* B, C, dan E

Objek	Jarak antara Objek *Medoid* dan *Nonmedoid*		Penempatan	
	A (C_1)	**D(C_2)**	**A(C_1)**	**D(C_2)**
B	3.1623	3.6056	1	0
C	3.6056	2	0	1
E	4	1	0	1

Keterangan penempatan:

1 menyatakan objek tersebut berada dalam grup atau klaster tersebut

0 menyatakan objek tersebut tidak berada dalam grup atau klaster tersebut

Berdasarkan tabel diatas maka diperoleh informasi sebagai berikut:

Klaster 1 : Objek A dan B

Klaster 2: Objek C, D, dan E

d) Bandingkan *cost* untuk kedua *node* tersebut dengan menggunakan rumus (3.2)

- Untuk *node* (A,E)

$$E = \sum_{i=1}^{2}\sum_{p\in C_i} d(p,o_i) = \sum_{i=1}^{}\sum_{p\in C_1} d(p,o_1) + \sum_{i=2}^{}\sum_{p\in C_2} d(p,o_2)$$
$$= d(B,A) + (d(C,E) + d(D,E))$$
$$= d(B,A) + d(C,E) + d(D,E)$$
$$= 3{,}1623 + 2{,}2361 + 1$$
$$= 6{,}3984$$

- Untuk *node* (A,D)

$$E = \sum_{i=1}^{2}\sum_{p\in C_i} d(p,o_i) = \sum_{i=1}^{}\sum_{p\in C_1} d(p,o_1) + \sum_{i=2}^{}\sum_{p\in C_2} d(p,o_2)$$
$$= d(B,A) + (d(C,D) + d(E,D))$$

$$= d(B,A) + d(C,D) + d(E,D)$$
$$= 3{,}1623 + 2 + 1$$
$$= 6{,}1623$$

Tabel 3.5 *Cost Node* (A,E) dan (A,D)

Node	***Cost***
(A,E)	$d(B,A) + d(C,E) + d(D,E) = 6{,}3984$
(A,D)	$d(B,A) + d(C,D) + d(E,D) = 6{,}1623$

Karena *cost node* tetangga (A,D) lebih kecil dari *cost node* terpilih (A,E) maka (A,D) menggantikan (A,E) sebagai *node* terpilih.

2) Pilih secara acak *node* tetangga dari *node* terpilih yang baru yaitu (A,D) misal (A,C). Kemudian lakukan hal yang sama seperti langkah (1) sehingga diperoleh hasil sebagai berikut:

Tabel 3.6 *Cost Node* (A,D) dan (A,C)

Node	***Cost***
(A,D)	$d(B,A) + d(C,D) + d(E,D) = 6{,}1623$
(A,C)	$d(B,A) + d(D,C) + d(E,C) = 7{,}3984$

Karena *cost node* tetangga (A,C) lebih besar dari *cost node* terpilih (A,D) maka *node* (A,C) tidak bisa menggantikan *node* (A,D).

3) Periksa secara acak *node* tetangga lainnya dari (A,D), misal (B,D). Kemudian bandingkan *cost* dari kedua node tersebut seperti pada langkah (1). Perlu diingat bahwa pemeriksaan node tetangga dari (A,D) hanya bisa sampai (B,D) karena kita sudah menentukan bahwa $maxneighbor = 2$.

   a) Dilakukan penempatan objek *nonmedoid* yaitu B,C,D ke dalam klaster terdekat yang medoidnya A dan D berdasarkan ukuran jarak. Untuk hasil penempatan objek dapat dilihat pada tabel 7.

b) Kemudian akan dilakukan juga penempatan objek *nonmedoid* A,C,E ke dalam klaster terdekat yang medoidnya B dan D.

Tabel 3.7 Penempatan Objek *Nonmedoid* A, C, dan E

Objek	Jarak antara Objek *Medoid* dan *Nonmedoid*		Penempatan	
	B (C_1)	**D(C_2)**	**B(C_1)**	**D(C_2)**
A	3.1623	3	0	1
C	5,3852	2	0	1
E	4,2426	1	0	1

Keterangan penempatan:

1 menyatakan objek tersebut berada dalam grup atau klaster tersebut.

0 menyatakan objek tersebut tidak berada dalam grup atau klaster tersebut.

Berdasarkan tabel diatas maka diperoleh informasi sebagai berikut:

Klaster 1 : Objek B

Klaster 2: Objek A, C, D, dan E

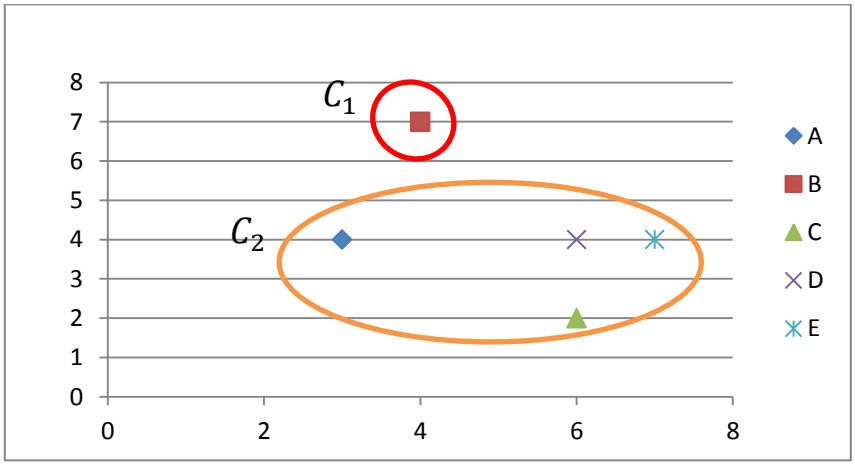
c) Bandingkan *cost* untuk kedua *node* tersebut dengan menggunakan rumus (3.2)

Tabel 3.8 *Cost Node* (A,D) dan (B,D)

Node	***Cost***
(A,D)	$d(B,A) + d(C,D) + d(E,D) = 6{,}1623$
(B,D)	$d(A,D) + d(C,D) + d(E,D) = 6$

Karena *cost node* tetangga (B,D) lebih kecil dari *cost node* terpilih (A,D) maka *node* (B,D) akan menggantikan *node* (A,D) sebagai *node* terpilih. Selain itu, iterasi berhenti sampai disini karena telah

ditentukan bahwa $numlocal = 1$, sehingga diperoleh *node* terbaik yaitu (B,D) dengan *mincost* = 6.

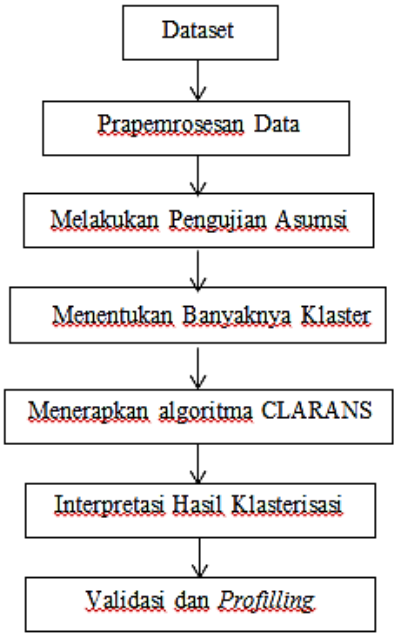


Gambar 3.2 Hasil Akhir Klasterisasi

Dari gambar tersebut maka diperoleh informasi bahwa:
Klaster 1 hanya terdiri dari objek atau siswa B saja
Klaster 2 terdiri dari objek atau siswa A, C, D dan E
dengan *medoid* setiap klasternya yaitu objek B dan D

3.3 Tahapan Penelitian

Langkah – langkah yang akan dilakukan dalam penelitian ini dapat dilihat pada tabel berikut:



Gambar 3.3 Tahapan Penelitian

3.3.1 Prapemrosesan Data

Tahap awal yang perlu dilakukan dalam penelitian ini adalah prapemrosesan data Tahap ini perlu dilakukan guna meningkatkan kualitas data sehingga hasil klasterisasi menjadi lebih baik. Prapemrosesan data dapat dilakukan dengan pembersihan atau integrasi atau reduksi atau penambahan atau transformasi data. Hal ini disesuaikan dengan data yang dimiliki.

3.3.2 Melakukan Pengujian Asumsi

Asumsi yang perlu diuji yaitu uji keberadaan pencilan (*outliers*) dan multikolinearitas. Deteksi *outliers* digunakan untuk mencari data yang berbeda dengan mayoritas data yang lain. Walaupun memiliki perilaku yang berbeda dengan mayoritas data yang lain dan sering dianggap *noise*, tetapi *outliers* sering kali mengandung informasi yang sangat berguna. Selain itu, setiap variabel dalam analisis klaster tidak boleh saling berkorelasi atau terjadi multikolinearitas, sebab korelasi tersebut akan menyebabkan pembobotan yang tidak berimbang dan berpengaruh

terhadap hasil klasterisasi. Rumus untuk menguji kedua asumsi tersebut dapat dilihat pada persamaan (2.9) dan (2.10). Jika terjadi multikolinearitas pada data maka perlu dilakukan transformasi variabel menggunakan Analisis Komponen Utama (AKU).

3.3.3 Menentukan Banyaknya Klaster

Pada algoritma berbasis partisi seperti CLARANS, banyaknya klaster yang akan dibentuk ditentukan terlebih dahulu dengan metode Elbow. Rumus untuk menentukan banyaknya klaster dapat dilihat pada persamaan (2.17).

3.3.4 Menerapkan Algoritma CLARANS

Di tahap ini, nilai parameter *maxneighbor* dan *numlocal* perlu ditentukan terlebih dahulu. Nilai *maxneighbor* yang digunakan yaitu $p\% \times k(n-k)$ dan nilai *numlocal* yang digunakan yaitu 2, dengan nilai p antara $1{,}25$ dan 1,5. Adapun k menyatakan banyaknya klaster yang terbentuk dan n menyatakan jumlah objek dalam himpunan data.

3.3.5 Interpretasi Hasil Klasterisasi

Pada tahap ini akan dijabarkan ukuran dari setiap klaster, nilai medoidnya, jarak objek dalam satu klaster, jarak antar klaster, dan anggota dari setiap klaster.

3.3.6 Validasi dan *Profilling*

Tahap akhir yang perlu dilakukan yaitu validasi dan *profilling*. Validasi merupakan proses mengukur kualitas hasil klasterisasi dengan menggunakan metode *silhoutte coefficient* yang dijelaskan secara rinci pada subbab 2.3.5.2. Nilai *silhoutte coefficient* berada antara 0 sampai 1. Semakin tinggi nilai *silhoutte coefficient* maka semakin baik kualitas hasil pengelompokkan data yang telah dibuat. Sedangkan *profiling* merupakan proses menggambarkan karakteristik tiap klaster untuk menjelaskan bahwa klaster-klaster tersebut berbeda pada dimensi yang relevan.