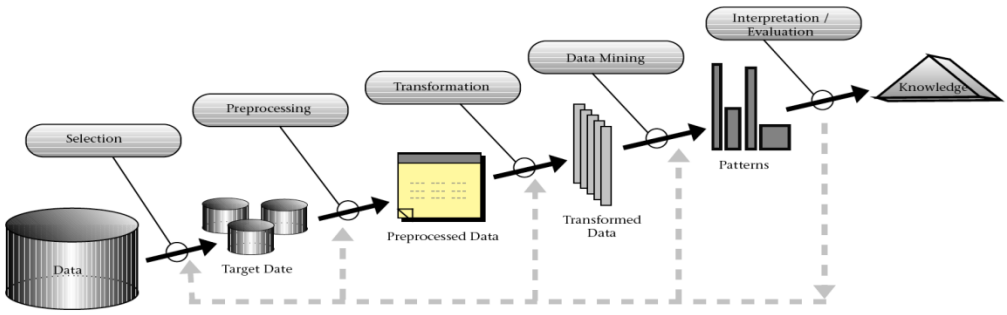
BAB 2
LANDASAN TEORI

2.1 *Data Mining*

Data mining adalah proses untuk mencari sejumlah pengetahuan, *pattern*, dan rangkaian informasi yang bermanfaat dari kumpulan data yang berukuran besar dan telah tersimpan dalam waktu lama yang dilakukan melalui sejumlah proses deskriptif, pembelajaran, dan prediksi dengan menggunakan suatu model atau algoritma (Zaki & Meira, 2014).

Data mining adalah pada dasarnya menggunakan teknik analisa statistik, pembelajaran menggunakan matematika, *artificial intelligence*, dan juga *machine learning* untuk memperoleh pengetahuan yang berguna dari suatu *database* berukuran besar (Turban & Aronson, 2005).

Banyak pihak yang menyamakan *data mining* dengan *Knowledge Discovery in Database* (KDD). Padahal sebenarnya *data mining* hanyalah salah satu tahapan di dalam KDD. KDD adalah keseluruhan proses di dalam memperoleh pengetahuan yang bermanfaat dari sekumpulan *database* berukuran besar sedangkan *data mining* adalah tahapan pada KDD yang fokus pada upaya menemukan pengetahuan tersebut dengan menggunakan algoritma (Silwattanursarn & Tuamsuk, 2012).Adapun proses dari KDD dapat dilihat pada Gambar 2.1.



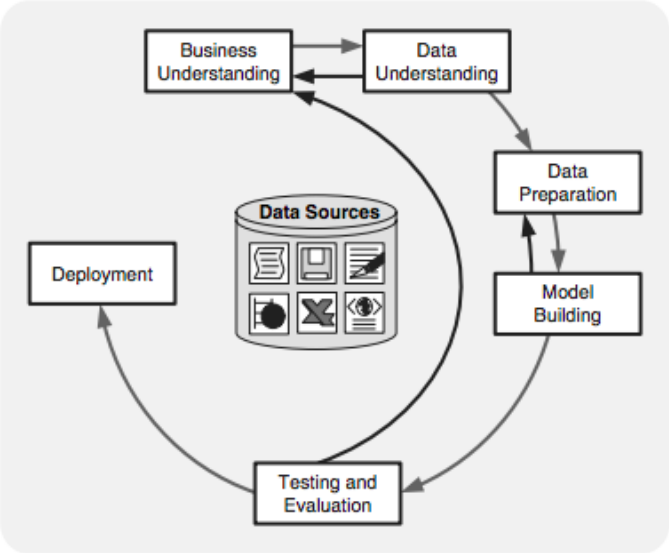
Gambar 2.1. Proses di dalam *Knowledge Discovery in Database*(Fayyad, Piatetsky-Shapiro, & Smyth, 1996)

Berdasarkan Gambar 2.1. dapat terlihat bahwa proses KDD memiliki sejumlah kegiatan berurutan, yang dapat dijabarkan sebagai berikut.

1. *Selection* :proses memilih data yang diperlukan pada *database* untuk di masukkan ke dalam proses analisis.
2. *Prepocessing* :proses penghilangan *noise* dan data yang inkonsisten, dan menggabungkan beberapa data yang berasal dari banyak sumber.
3. *Transformation* : Mentransformasi data ke dalam bentuk yang sesuai untuk proses *data mining*.
4. *Data Mining* : Memilih algoritma *data mining* yang sesuai dengan *pattern* data; Ekstraksi pola dari data.
5. *Interpretation / Evaluation* : menginterpretasi pola menjadi pengetahuan dengan menghilangkan pola yang *redundant* dan tidak relevan; Mentranslasi pola ke dalam bentuk yang dapat dimengerti oleh manusia.

2.2 Proses pada *Data Mining*

Data mining memiliki sejumlah proses seperti yang dapat dilihat pada Gambar 2.2 (Adesola & Baines, 2005).



Gambar 2.2. Proses pada *Data Mining*

Pada Gambar 2.2. dapat dilihat bahwa *data mining* memiliki sejumlah proses sebagai berikut.

1. *Business Understanding*

Pemahaman bisnis meliputi penentuan tujuan bisnis, penilaian situasi saat ini, menetapkan tujuan penambangan data, dan mengembangkan rencana proyek.

2. *Data Understanding*

Langkah ini bisa meliputi pengumpulan data awal, deskripsi data, eksplorasi data, dan verifikasi kualitas data. Eksplorasi data seperti melihat statistik ringkasan (yang mencakup tampilan visual dari variabel kategori) dapat terjadi pada akhir fase ini. Model seperti analisis cluster juga dapat diterapkan selama fase ini, dengan maksud untuk mengidentifikasi pola pada data.

3. *Data Preparation*

Setelah sumber data tersedia untuk diidentifikasi. Sumber data ini perlu dipilih, dibersihkan, dibangun ke dalam bentuk yang diinginkan, dan diformat. Pembersihan data dan transformasi data dalam pemodelan data perlu dilakukan

pada tahap ini. Eksplorasi data pada kedalaman yang lebih besar dapat diterapkan selama fase ini, dan model tambahan digunakan untuk memberikan kesempatan kembali untuk melihat pola berdasarkan pemahaman bisnis.

4. *Modelling*

Sejumlah *tools* seperti visualisasi dan analisis *cluster* dapat digunakan pada tahapan ini. Sejumlah *tools* ini dapat digunakan sejumlah aturan dan hubungan antar aturan (*association rule*) sehingga dapat diperoleh pemahaman yang lebih mendalam terhadap data yang ada. Hal ini memudahkan di dalam memilih model yang sesuai. Untuk mendukung pemodelan ini makan diperlukan pemilihan data apa yang akan diperlakukan sebagai *data training* dan data apa yang akan diperlakukan sebagai *data testing*.

5. *Evaluation*

Hasil model harus dievaluasi dalam konteks tujuan bisnis yang ditetapkan pada tahap pertama (business understanding). Mendapatkan pemahaman bisnis adalah prosedur berulang dalam data mining, di mana hasil berbagai alat bantu visualisasi, statistik, dan kecerdasan buatan menunjukkan kepada pengguna hubungan baru yang memberikan pemahaman operasi organisasi yang lebih dalam.

6. *Deployment*

Data mining dapat digunakan untuk memverifikasi hipotesis yang digunakan sebelumnya, atau untuk penemuan pengetahuan yang berupa pengetahuan yang tidak diketahui sebelumnya.

2.3 Teknik pada *Data Mining*

Menurut (Khanbabaei, Sobhani, Alborzi, & Radfar, 2018) Terdapat sejumlah teknik pada *data mining* sebagai berikut.

1. *Classfication* (Klasifikasi): Fungsi pembelajaran yang melakukan pemetaan*item* data ke dalam satu dari beberapa*class* yang telah ditetapkan(*predefined class).*
2. *Regression*: Jenis pembelajaran yang menghubungkan suatu *instance* data menjadi nilai asli dari variabel yang diprediksi.
3. *Clustering*: Proses mengelompokkan sejumlah *instance* data ke dalam *cluster* yang sudah ditentukan.
4. *Summarization*: Fungsi pembelajaran untuk mengemukakan penjelasan yang utuh dari suatu *subset* data.
5. *Dependency Modelling*: sering juga disebut sebagai *Association Rule Learning* atau pembelajaran aturan asosiasi yang menghasilkan model yang menyatakan tingkat signifikansi ketergantungan antar variabel.
6. *Change and Deviation Detection*: sering juga disebut sebagai *anomaly detection,* yang menemukan perubahan yang signifikan dari suatu data.

2.4 *Clustering*

Clustering adalah proses mengelompokkan sekumpulan *instance* data ke dalam kelompok atau *cluster* yang lebih kecil berdasarkan kedekatan ciriyang dimiliki (Serapião, Corrêa, Gonçalves, & Carvalho, 2016).

Terdapat berbagai algoritma *clustering* yang dapat digunakan, tetapi secara umum dapat dikelompokkan menjadi beberapa kategori sebagai berikut (Rokach & Maimon, 2005).

1. *Partitioning Methods*. Diberikan himpunan dari *n* objek. Metode partisi akan mengelompokkan *k* partisi dari data. Dimana setiap partisi merepresentasikan sebuah *cluster* dan $k \leq n$. Setiap objek yang ada merupakan bagian dari sebuah *clusterk*. Beberapa algoritma yang sering dipakai, yang termasuk dalam kategori *partitioning methods* adalah algoritma *K-Means* dan *K-Medoids*.
2. *Hierarchical Methods*. Pada metode berbasis hirarki ini akan dibangkitkan *hierarchical decomposition* (dekomposisi berurutan) dari himpunan data objek. Berbeda dengan metode *partitioning* yang mengelompokkan data ke dalam kelompok-kelompok. Metode *hierarchical* mengelompokkan data ke dalam hirarki atau *tree* dari *cluster*. Representasi data dalam bentuk hirarki adalah diperlukan untuk keperluan penyajiandan visualisasi data. Strategi pengembangan dari metode ini dapat dibagi menjadi 2 jenis yaitu *Agglomerative* (*Bottom-Up*) dan *Devisive* (*Top-Down*). Metode *Agglomerative*merupakan metode yang sering digunakan dan terdiri atas metode: *Single Linkage*, *Complete Linkage*, dan *Average Linkage*.
3. *Density-Based Methods*. Metode *Density-Based* merupakan metode yang dikembangkan berdasarkan *density* (kepadatan) tertentu. Metode ini menganggap *cluster* sebagai suatu area yang berisi objek-objek yang padat/sesak, yang dipisahkan oleh area yang memiliki kepadatan rendah (merepresentasikan *noise*). Beberapa algoritma yang termasuk di dalam

Density-Based adalah DBSCAN (*Density Based Spatial Clustering of Application with Noise*) dan OPTICS (*Ordering Points to Identify the Clustering Structure*).

4. *Grid-Based Methods*. Pendekatan *Grid-Based Methods* menempatkan ruang objek ke dalam jumlah berhingga sel yang membentuk struktur *grid*, sehingga dikatakan juga bahwa metode ini menggunakan *multiresolution* pada struktur data *grid* (jaringan). Salah satu algoritma yang mendasarkan pada metode ini adalah STING (*Statistical Information Grid*).

2.5 *K-Means*

Algoritma *K-Means* merupakan salah satu algoritma pembelajaran yang bersifat *unsupervised* dan digunakan untuk proses *clustering* melalui proses perulangan yang sederhana untuk mendapatkan solusi yang bersifat *local minimal* (Serapião, Corrêa, Gonçalves, & Carvalho, 2016). Hal yang penting di dalam algoritma *K-Means* adalah penentuan *centroid*. Penentuan *centroid* dapat menentukan besarnya kesalahan penempatan data di dalam suatu *class* dan juga keseimbangan jumlah data di dalam suatu *class* yang jika tidak ditangani dengan baik dapat mengarah pada terjadinya permasalahan *class imbalance* (Hartono, Sitompul, Tulus, & Nababan, 2018).

Algoritma *K-Means* pertama sekali dikemukakan oleh MacQueen pada tahun 1967 yang merupakan suatu algoritma yang sederhana, dan merupakan algoritma yang bersifat pembelajaran tidak terawasi (Juanying & Shuai, 2010).

Algoritma *K-Means* digunakan sebagai alternatif metode *clustering*untuk data yang berukuran besar karena memiliki kecepatan yang lebih tinggi

dibandingkan dengan metode Hirarki. Algoritma ini terdiri-dari 2 (dua) fase yaitu sebagai berikut (Loohach & Garg, 2012).

1. Fase pertama, *user* menentukan nilai *centroidk* dengan membangkitkan suatu bilangan *random*, di mana jumlah *cluster K* sudah ditentukan terlebih dahulu. Untuk menempatkan tiap *instance* ke *centroid* terdekat. Beberapa fungsi perhitungan jarak dapat digunakan untuk menghitung jarak dari tiap *instance* terhadap *centroid*. Ketika setiap *intance* sudah ditempatkan ke dalam beberapa *cluster*, maka tahap pertama sudah dilengkapi dan pengelompokan sudah selesai.
2. Fase kedua, mengkalkulasi kembali rata-rata dari *cluster* yang telah diformulasikan sebelumnya. Proses iterasi (perulangan) ini akan dilanjutkan sampai fungsi kriteria menjadi minimum.

2.6 Perhitungan Jarak Antar *Instance* ke *Centroid*

Terdapat beberapa metode perhitungan jarak yang dapat digunakan di dalam mengukur kedekatan suatu *instance* terhadap *centroid*. Beberapa metode perhitungan tersebud adalah: *Euclidean Distance*, *Manhattan Distance*, dan *Chebichev Distance*. *Eucliden Distance* merupakan perhitungan jarak yang umum digunakan karena metode ini dapat memperoleh jarak terdekat antara dua titik yang diperhitungkan. Adapun *Manhattan Distance* sering digunakan untuk mendeteksi keberadaan suatu *outlier*, sedangkan *Chebichev Distance* digunakan ketika kita ingin memperoleh jarak *magnitude* absolut dari tiap koordinat *centroid* terhadap *cluster*(Grabusts, 2011).

Adapun fungsi objektifuntuk menghitung jarak dengan menggunakan *Euclidean Distance* dapat dilihat pada Persamaan 2.1(Serapião, Corrêa, Gonçalves, & Carvalho, 2016).

$$dij = \sqrt{\sum_{k=1}^{p} (xik - xjk)^2} \quad (2.1)$$

Dimana:

dij = Jarak kuadrat *euclidean* antar objek ke-i dengan objek ke-j

p = Jumlah variabel *cluster*

Xik = Nilai atau data dari objek ke-i pada variabel ke-k

Xjk = Nilai atau data dari objek ke-j pada variabel ke-k

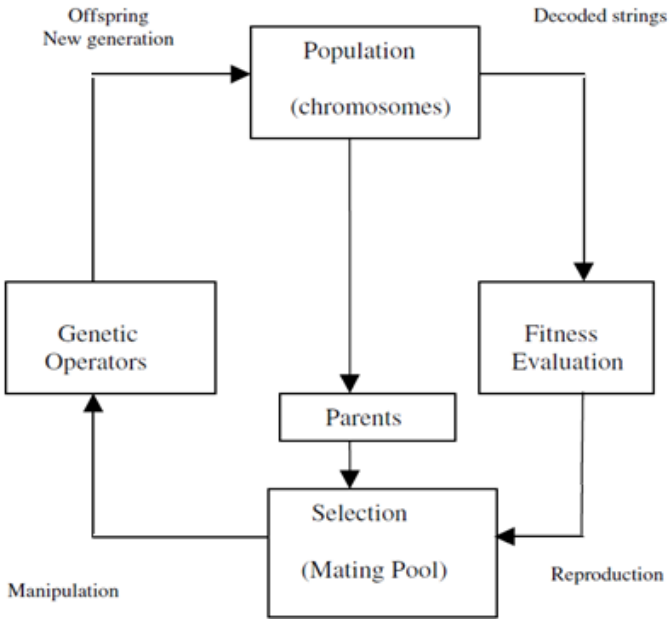
Sedangkan fungsi objektif (f) untuk menghitung jarak dengan menggunakan *Manhattan Distance* dapat dilihat pada Persamaan 2.2.

$$dij = |Xik - Xjk| \quad (2.2)$$

2.7 Algoritma Genetika

Pada tahun 1975, John Holland, di dalam bukunya yang berjudul "*Adaption in Natural and Artificial Systems*", mengemukakan komputasi berbasis evolusi. Tujuannya adalah untuk membuat komputer dapat melakukan apa yang terdapat di alam. Sebagai seorang pakar komputer, Holland memfokuskan diri pada manipulasi dari *string* dalam bentuk *binary* bit. Holland mengemukakan algoritma tersebut sebagai suatu konsep abstrak dari evolusi alam. Tahapan Algoritma Genetika yang dikemukakan oleh Holland dapat direpresentasikan sebagai suatu tahapan berurutan sebagai suatu bentuk populasi dari kromosom buatan menjadi sebuah populasi baru (Negnevitsky, 2005).

Adapun proses dari algroritma genetika secara umum dapat dilihat pada Gambar 2.3.



Gambar 2.3. Siklus Algoritma Genetika (Konar, 2005)

Fase awal dari algoritma genetika adalah inisialisasi populasi yang menyatakan alternatif solusi. Elemen dari populasi adalah dideskripsikan dalam bentuk deretan *bit string* yang berisi *bit* 0 atau 1 yang disebut sebagai kromosom. Kemudian langkah selanjutnya adalah menghitung nilai *fitness* berdasarkan gen yang ada pada kromosom dalam tiap populasi. Berdasarkan nilai *fitness* dari tiap koromosom, maka tahapan selanjutnya adalah tahapan seleksi yang berfungsi untuk memilih kromosom yang terpilih sebagai *parent* yang akan menjalani *crossover*. Proses *crossover* yang berjalan dengan beberapa variasi operator *crossover* berperan penting di dalam membentuk kromosom anak (*offspring*) yang juga berperan penting untuk menambah keanekaragaman *string* di dalam suatu populasi. Kromosom selanjutya akan masuk ke dalam tahap mutasi yang berfungsi untuk memastikan bahwa keanekaragaman (*diversity*) dari kromosom

dalam suatu populasi tetap terjaga, untuk menghindari terjadinya konvergensi prematur yang berujung pada terjadinya solusi yang *local optima*.

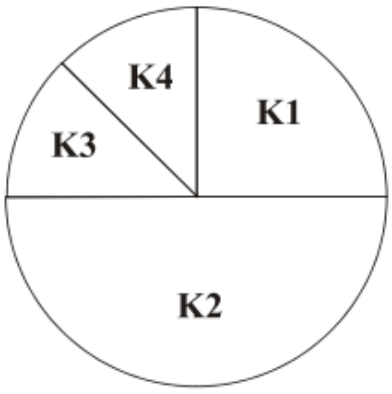
2.8 Proses Seleksi

Nilai *fitness* yang diperoleh oleh setiap individu salah satunya ditentukan oleh proses seleksi yang ada (Reeves & Rowe, 2003).Ada dua metode yang umum digunakan di dalam proses seleksi, yaitu sebagai berikut (Chipperfield, Fleming, Pohlheim, & Fonseca, 2005):

1. Seleksi roda *roulette* (*Roulette wheel selection*)

   Pada metode ini tiap individu akan ditempatkan di dalam suatu segmen yang besarnya sama dengan nilai *fitness* dari tiap individu. Kemudian *roulette wheel* akan berputar untuk memilih individu yang akan terseleksi. Semakin baik nilai *fitness* berarti semakin pula segmen yang ditempati oleh suatu individu sehingga semakin besar peluang individu tersebut untuk terpilih(Kumar & Jyotishree, 2012). Metode *roulette wheel selection* dapat dilihat pada Gambar 2.4.

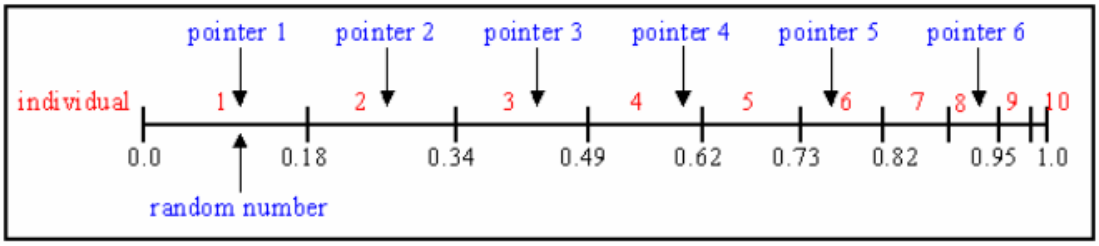
Kromosom	Nilai fitness
K1	1
K2	2
K3	0.5
K4	0.5
Jumlah	4



Gambar 2.4 Metode *Roulette Wheel Selection*(Suyanto, 2005)

2. *Stochastic Universal Sampling*

Metode ini akan menempatkan individu ke dalam segmen-segmen garis berurutan dimana tiap segmen berukuran sama dengan ukuran nilai *fitness* dari tiap individu. Kemudian akan terdapat *pointer* dengan jumlah sebanyak *n* individu. Kemudian jarak antara *pointer* adalah sebanyak 1/n dan posisi *pointer* pertama akan diberikan secara acak dalam range [1, 1/N](Pencheva, Atanassov, & Shannon, 2009).Metode *stochastic universal sampling* dapat dilihat pada Gambar 2.5.



Gambar 2.5 Metode *Stochastic Universal Sampling* (Pencheva, Atanassov, & Shannon, 2009)

2.9 Pindah Silang (*Crossover*)

Proses pindah silang (*crossover*) pada algoritma genetika adalah proses yang mengkombinasikan dua kromosom *parents* sehingga dihasilkan kromosom *offspring*. Pada proses kawin silang ini diharapkan akan dihasilkan kromosom baru yang lebih baik daripada kromosom *parent*. Proses *crossover* akan ditentukan dengan suatu variabel *crossover probability* yang telah ditentukan sebelumnya oleh pengguna (Abuiziah & Shakarneh, 2013).

Kromosom yang terpilih untuk mengalami *crossover* ditentukan melalui nilai *probability crossover* (Pc). Suatu kromosom terpilih untuk mengalami *crossover* jika nilai *random* kromosom (Rc) < Pc. Besarnya nilai Pc adalah

diantara 0.4 sampai dengan 0.9 (Coley, 1999). Beberapa jenis *crossover* sebagai berikut:

1) *Crossover* Pengkodean Biner

Ada beberapa metode *crossover* dengan pengkodean biner, yaitu sebagai berikut:

a. *One Point Crossover*

Pada*one point crossover*, sebuah bilangan acak mendefinisikan segmen yang membagi kromosom ke dalam dua bagian. Kromosom *offspring* dihasilkan melalui kombinasi kromosom yang dihasilkan pada *segmen point* berdasarkan bilangan acak tersebut. Bagian pertama adalah sebelum *segmen point* mengambil kromosom *parent* yang pertama dan bagian kedua setelah bilangan *random* adalah mengambil kromosom *parent* yang kedua(Andrade, Errico, Aquino, Pereira, & Barbosa, 2008). Illustasi dari proses *one point crossover* dapat dilihat pada Tabel 2.1.

Tabel 2.1 *One Point Crossover*

Kromosom *Parent 1*	**11001**011
Kromosom *Parent 2*	11011**111**
Offspring	**11001111**

b. *Two Point Crossover*

Two point crossover hampir sama dengan *one point crossover*. Perbedaannya adalah bahwa pada *two point crossover*, cut point yang digunakan adalah sebanyak 2, dan dibangkitkan secara acak (Mendes, 2013). Illustrasi dari proes *two point crossover* dapat dilihat pada Tabel 2.2.

Tabel 2.2 *Two Point Crossover*

Kromosom *Parent* 1	**11**001**11**
Kromosom *Parent* 2	11**011**11
Offspring	**11011111**

2) *Uniform Crossover*

Pada *uniform crossover*, sebuah vektor *bit* acak yang berukuran sama dengan kromosom yang digunakan. Di dalam proses untuk menghasilkan kromosom *offspring*, akan dipilih bit-bit dalam *mask* vektor *bit* acak. Jika yang terpilih adalah bit 0, berarti kromosom *offspring* diperoleh dari *parent* 1 dan jika yang terpilih adalah bit 1 berarti kromosom *offspring* diperoleh dari *parent* 2(Andrade, Errico, Aquino, Pereira, & Barbosa, 2008). Illustrasi dari proses *uniform crossover* dapat dilihat pada Tabel 2.3.

Tabel 2.3 *Uniform Crossover*

Kromosom *Parent* 1	11001011
Kromosom *Parent* 2	11011111
Mask Bit	01010101
Offspring	11011111

3) *Arithmetic Crossover*

Kromosom *Offspring*diperoleh dengan melakukan operasi aritmatika terhadap *parent* (induk). Terdapat 3 jenis *arithmetic crossover*, yaitu sebagai berikut(Picek, Jakobovic, & Golub, 2013).

1. *Single Arithmetic Crossover*

   Pada *single arithmetic crossover*, pindah silang terjadi pada salah satu gen yang posisinya ditentukan dengan cara membangkitkan suatu bilangan acak. Pada posisi gen yang ditentukan, nilai gen akan ditentukan melalui operasi

aritmatika terhadap nilai gen dari *parent* menurut persamaan 2.2 (Eiben & Smith, 2007). Adapun operasi aritmatika pada *single arithmetic crossover* dapat dilihat pada Persamaan 2.3 dan Tabel 2.4.

$$\text{Child} = \left\langle x_1,....,x_k,\alpha.y_k + (1-\alpha).x_k,...,x_n \right\rangle \quad .....(2.4)$$

Ket:

α = Variabel pengali yang nilainya berkisar dari 0-1

Tabel 2.4 *Single Arithmetic Crossover*

Kromosom *Parent* 1	0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9
Kromosom *Parent* 2	0.3 0.2 0.3 0.2 0.3 0.2 0.3 0.2 0.3
Bilangan Acak	8
α	0.5
Kromosom *Offspring* 1	0.1 0.2 0.3 0.4 0.5 0.6 0.7 **0.5** 0.9
Kromosom *Offspring* 2	0.3 0.2 0.3 0.2 0.3 0.2 0.3 **0.5** 0.3

Pada Tabel 2.4 dapat dilihat bahwa bilangan acak yang dibangkitkan adalah 8 dan hal ini berarti bahwa gen 8 pada kromosom *parent* 1 dan gen pada 8 pada kromosom *parent* 2 akan mengalami proses *Single arithmetic crossover* dengan bilangan acak 0.5. Proses *Single Arithmetic Crossover* dilakukan dengan menggunakan Persamaan 2.4.

2. *Simple Arithmetic Crossover*

Pada *simple arithmetic* crossover, Tentukan bilangan random sebagai titik potong antara 0 sampai sepanjang kromosom pada masing-masing parent. Untuk gen pada kromosom *offspring* untuk batas sebelum titik potong disalin dari gen pada kromosom *parent*. Untuk gen setelah titik potong, gen yang ada

dibentuk dari operasi aritmatika pada gen dari kromosom *parent* dengan persamaan seperti pada persamaan 2.5(Picek, Jakobovic, & Golub, 2013). Illustrasi dari proses *simple arithmetic crossover* dapat dilihat pada Tabel 2.5.

$$\text{Child}=\left\langle x_1,...,x_k,\alpha.y_{k+1}+(1-\alpha).x_{k+1},...,\alpha.y_n+(1-\alpha).x_n\right\rangle \text{.....(2.5)}$$

Ket:

α = Variabel pengali yang nilainya berkisar dari 0-1

Tabel 2.5 *Simple Arithmetic Crossover*

Kromosom *Parent* 1	0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9
Kromosom *Parent* 2	0.3 0.2 0.3 0.2 0.3 0.2 0.3 0.2 0.3
Bilangan Acak	6
α	0.5
Kromosom *Offspring* 1	0.1 0.2 0.3 0.4 0.5 0.6 **0.5 0.5 0.6**
Kromosom *Offspring* 2	0.3 0.2 0.3 0.2 0.3 0.2 **0.5 0.5 0.6**

Pada Tabel 2.5 dapat dilihat bahwa bilangan acak yang dibangkitkan adalah 6 dan hal ini berarti bahwa gen pada kromosom *parent* 1 dan gen pada 8 pada kromosom *parent* 2 akan mengalami proses *Single arithmetic crossover* dengan bilangan acak 0.5. Proses *Single Arithmetic Crossover* dilakukan dengan menggunakan Persamaan 2.4.

3. *Whole Arithmetic Crossover*

Pada *whole arithmetic crossover*, gen pada kromosom *offspring* diperoleh dari hasil operasi aritmatika gen pada kromosom *parent*, di mana proses aritmatika yang dilakukan sesuai dengan persamaan 2.6(Eiben & Smith,

2007). Illustrasi dari proses *whole arithmetic crossover* dapat dilihat pada Tabel 2.6.

Child= $\left\langle \alpha . \bar{x} + (1-\alpha) . \bar{y} \right\rangle$ .....(2.6)

Ket:

α = Variabel pengali yang nilainya berkisar dari 0-1

Tabel 2.6 *Whole Arithmetic Crossover*

Kromosom *Parent* 1	0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9
Kromosom *Parent* 2	0.3 0.2 0.3 0.2 0.3 0.2 0.3 0.2 0.3
α	0.5
Kromosom *Offspring*	0.2 0.2 0.3 0.3 0.4 0.4 0.5 0.5 0.6

2.10 Algoritma GenClust++

Algoritma GenClust++ dikemukakan oleh (Islam, Estivill-Castro, Rahman, & Bossomaier, 2018) yang telah menyempurnakan metode GenClust yang dikemukakan oleh (Rahman & Islam, A Hybrid Clustering Technique Combining a Novel Genetic Algorithm with K-Means, 2014) dengan menambahkan teknik *modify K-Means (MK-Means*) dan telah menggunakan 100% kromosom yang berkualitas yang diperoleh melalui perhitungan deterministik.

Adapun *pseudocode* dari algoritma GenClust++ adalah sebagai berikut(Islam, Estivill-Castro, Rahman, & Bossomaier, 2018).

Input : A dataset R_0, and a used defined number of generations N

Output: A set of Clusters C

/* Compo. 1 : Normalize the Dataset */

$R \leftarrow Normlize\ (R_0)$;

/* Compo. 3: Initial Population with Probabilistic Selection */

P_f ←InitialPopulation (R);

CR_b ←BestChromosome (P_f, F_f = *fitness* (P_f , R)

P_s←ProbabilisticSelection (P_f, R);

For g ← 1 to |N| **do**

if g mod 10 ≠ 0**then**

/* Compo. 4: Crossover */

P_c← Crossover (P_S);

else

/* Compo. 7: Cloning with Mutation and MK-Means */

P_{cl}← ProbabilisticCloning (P_s , R);

/* Compo. 5: Elitism */

CR_b←*Elitism* (CR_b, P_{cl}, R);

P_c←MK-Means (P_{cl}, R, 15)

/* Compo. 5: Elitism */

CR_b←*Elitism* (CR_b, P_c, R);

/* Compo. 6: Mutation */

P_m←Mutation (P_c, R)

/* Compo. 5: Elitism */

CR_b←*Elitism* (CR_b, P_m, R);

if g > 10 **then**

/* Compo. 8: Chromosome Selection */

P_s←SelectChromosome (P_s, P_m, R)

Else

P_s← P_m

Component 2 /* Compo. 2: MK-Means

C_n←MK-Means (CR_b, R, 50)

C ←Denormalize (C_n, R, R_o)

return C

Tahapan *initial population* pada Algoritma GenClust++ dapat dibagi menjadi 2 (dua) *stage* sebagai berikut (Islam, Estivill-Castro, Rahman, & Bossomaier, 2018).

Input : A dataset *R*, population size *s* of genetic algorithm

Output: A set $P_f$of high quality chromosomes from which to sample the initial population

Set $P_f \leftarrow \emptyset; K = \langle 2, 3, 4 \ldots . (3\ x \frac{s}{10} + 1)\rangle;$

for *j* = 1 **to** 5 **do**

for *i* = 1 **to** |*K*| **do**

/* Produce a chromosome CR with $K_i$genes;

K_1= 2, K_2= 3 … K_9= 10

CR ←ProduceChromosome (*R*, K_i);

$P_f \leftarrow P_f \cup R$;

for *i* = 1 **to** 3 x s/10 **do**

/* Produce a random number in the range 2 to $\sqrt{|R|}$

k= ProduceRandNumber (2, $\sqrt{|R|}$);

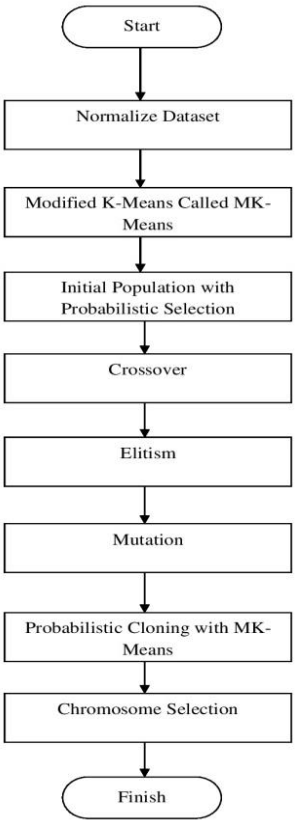
for *j* = 1 **to** 5 **do**

CR ←ProduceChromosomoe (*R*, *k*);

$P_f \leftarrow P_f \cup R$;

Return P_f

Adapun berdasarkan *pseudocode* tersebut maka dapat dilihat tahapan kerja dari algoritma *GenClust++* seperti pada Gambar 2.6.



Gambar 2.6 Tahapan Kerja dari Algoritma *GenClust++*

Pada Gambar 2.6 dapat dilihat bahwa proses kerja pada algoritma GenClust++ dimulai dengan normalisasi *dataset*. Kemudian setelah itu akan dilakukan pemrosesan dengan menggunakan *Modified K-Means*. Kelebihan dari *MK-Means* ini adalah dapat melakukan pemrosesan pada *numerical attributes* maupun *categorical attributes*. Setelah itu akan dilakukan proses inisial populasi awal dengan menggunakan algoritma genetika. Adapun tahapan di dalam inisalisasi populasi awal adalah sebagai berikut.

1. Tentukan jumlah *cluster K*. Jumlah *cluster K* disesuaikan dengan *class* data yang ada pada tiap *dataset*. Misalkan jumlah *cluster K* adalah sebesar 9.

2. Tentukan ukuran populasi awal *s*. Secara default ukuran dari populasi awal s adalah sebesar *K x 2*.
3. Pembentukan separuh kromosom tahap pertama. Pembentukan separuh kromosom tahap pertama dilakukan dengan cara menghitung *density* dari tiap gen yang mewakili atribut yang ada. Ambil atribut pada *instance* dengan *density* terbesar untuk dijadikan sebagai gen pada kromosom.
4. Pembentukan separuh Kromosom tahap kedua. Pembentukan separuh kromosom tahap kedua dilakukan dengan cara membangkitkan bilangan acak dari 2 sampai dengan $\sqrt{R}$ maka kromosom yang tepat sama dengan bilangan acak yang dihasilkan, atribut yang ada akan mewakili gen pada kromosom.

Kemudian selanjutnya akan dilakukan proses *crossover*, *elitism*, mutasi yang merupakan tahapan pada algoritma genetika dengan tujuan untuk mendapatkan nilai *fitness* yang semakin baik. Kemudian setelah itu akan dilakukan proses *probabilistic cloning* dengan *MK-Means* yang secara umum menggunakan jumlah iterasi sebesar 15 sehingga akan menghasilkan kromosom yang akan terpilih untuk dijadikan sebagai *centroid*.

2.11 Penelitian-Penelitian Terkait

Penelitian mengenai upaya meningkatkan kinerja algoritma *K-Means* telah dilakukan oleh sejumlah peneliti. (Zahra, Ghazanfar, Khalid, Azam, Naeem, & Prugel-Bennett, 2015) telah mengemukakan metode penentuan *centroid* berbasis *recommender system* yang dilakukan dengan mencari nilai korelasi *pearson* dari sejumlah metode penenuan *centroid* yang telah ada dan mencari nilai

yang terdekat dengan permasalahan yang dihadapi. Kelemahan yang ada dari metode ini adalah perlu pendefinisian yang tepat dari *k* jumlah *cluster* dan dapat berakibat pada terjadinya kondisi *local optima*.(Hartono, Ongko, & Abdullah, Determining a Cluster Centroid of K-Means Clustering Using Genetic Algorithm, 2015) telah menggunakan algoritma genetika di dalam menentukan *centroid* dari suatu *cluster* pada algoritma *K-Means* dan memperoleh hasil bahwa hasil yang diperoleh lebih baik dibandingkan dengan algoritma *K-Means* klasik. Namun, penelitian yang dilakukan masih menggunakan bilangan *random* di dalam penentuan gen pada kromosom di dalam algoritma genetika.

(Rahman & Islam, A Hybrid Clustering Technique Combining a Novel Genetic Algorithm with K-Means, 2014) telah menggunakan algoritma GenClust di dalam menentukan *centroid* pada algoritma *K-Means*. Algoritma *GenClust* ini menggabungkan penggunakan bilangan *random* dan juga penentuan secara deterministik berdasarkan pada nilai *density* yang ada di dalam menentukan gen pada kromosom. Kemudian (Islam, Estivill-Castro, Rahman, & Bossomaier, 2018)memperbaiki penelitian yang dilakukan oleh (Rahman & Islam, A Hybrid Clustering Technique Combining a Novel Genetic Algorithm with K-Means, 2014) menggunakan teknik penentuan gen yang tidak menggunakan bilangan acak. Namun, penelitian ini belum memperhitungkan pengaruh *crossover* dan seleksi di dalam penentuan *centroid*. Hal ini perlu diperhitungkan di dalam penggunakan algoritma genetika, di mana penelitian yang dilakukan oleh(Muzid, 2014)menyatakan bahwa dalam algoritma genetika terdapat beberapa parameter penting yang harus didefinisikan yaitu ukuran populasi, proses seleksi, *crossover*, dan mutasi yang harus didefinisikan secara hati-hati agar tidak terjadi konvergensi

dini atau lokal optimum yaitu dimana individu-individu dalam populasi konvergen pada suatu solusi optimum lokal sehingga hasil paling optimum tidak dapat ditemukan.