BAB II

TINJAUAN PUSTAKA

II.1. Pengertian Data Mining

Data mining merupakan serangkaian proses untuk menggali nilai tambah dari suatu kumpulan data berupa pengetahuan yang selama ini tidak diketahui secara manual dari suatu kumpulan data. Defenisi lain data mining adalah sebagai proses untuk mendapatkan informasi yang berguna dari gudang basis data yang besar. Data mining juga diartikan sebagai pengekstrakan informasi baru yang diambil dari bongkahan data besar yang membantu dalam pengambilan keputusan. Istilah data mining kadang disebut juga knowledge discovery. Istilah data mining dan Knowledge Discovery in Database (KDD) sering kali digunakan secara bergantian untuk menjelaskan proses penggalian informasi tersembunyi dalam suatu basis data yang besar. Sebenarnya kedua istilah tersebut memiliki konsep yang berbeda, tetapi berkaitan satu sama lain. KDD adalah kegiatan yang meliputi pengumpulan, pemakaian data, historis untuk menemukan keteraturan, pola atau hubungan dalam set data yang berukuran besar (Buaton ; 2014 : 90-91).

Data mining berguna untuk membuat keputusan yang kritis, terutama dalam strategi ,juga dapat digunakan untuk pengambilan keputusan di masa depan berdasarkan informasi yang diperoleh dari data masa lalu. Tergantung pada aplikasinya, data bisa berupa data mahasiswa, data pasien, data nasabah atau penjualan. Banyak kasus dalam kehidupan sehari-hari yang tanpa disadari bisa diselesaikan dengan data mining, diantaranya adalah :

1. Memprediksi harga saham dalam beberapa bulan ke depan berdasarkan performansi perusahaan dan data-data ekonomi.
2. Memprediksi berapa jumlah mahasiswa baru di perguruan tinggi berdasarkan data pendaftar pada tahun-tahun sebelumnya
3. Memprediksi nilai indeks prestasi mahasiswa berdasarkan nilai IP setiap semester sebelumnya.
4. Produk apa yang akan dibeli pelanggan secara bersamaan jika membeli produk di swalayan (Buaton ; 2014 : 100)

II.2. Algoritma C4.5

Algoritma C4.5 merupakan algoritma yang digunakan untuk membentuk pohon keputusan. Pohon keputusan merupakan metode klasifikasi dan prediksi yang sangat kuat dan terkenal. Metode pohon keputusan mengubah fakta yang sangat besar menjadi pohon keputusan yang merepresentasikan aturan. Aturan dapat dengan mudah dipahami dengan bahasa alami. Dan mereka juga dapat diekspresikan dalam bentuk bahasa basis data seperti *Structured Query Language* untuk mencari *record* pada kategori tertentu (Kusrini & Lutfi, 2009: 13).

Secara umum Algoritma C4.5 untuk membangun pohon keputusan adalah sebagai berikut :

a. Pilih atribut sebagai akar
b. Buat cabang untuk masing-masing nilai.
c. Bagi kasus dalam cabang.

d. Ulangi proses untuk masing-masing cabang sampai semua kasus pada cabang memiliki kelas yang sama.

Untuk memilih atribut sebagai akar, didasarkan pada nilai gain tertinggi dari atribut-atribut yang ada. Untuk menghitung gain, cari terlebih dahulu nilai *entropy*. Untuk menghitung nilai *entropy* digunakan rumus seperti yang tertera berikut:

$$Entropy(S) = \sum_{j=1}^{n} - Pj \log 2 \; Pj \quad \text{.....(1)}$$

Keterangan:

S = Himpunan kasus (*Entropy)*

n = Bnyaknya partisi S

Pj = Probabilitas yang di dapat dari kelas dibagi total kasus

Kemudian hitung nilai *gain* menggunakan rumus berikut :

$$Gain\,(S,A) = Entropy\,(S) - \sum_{i=1}^{n} \frac{|Si|}{|S|} * Entropy\,(Si) \quad \text{.....(2)}$$

Keterangan:

S : Himpunan kasus

A : Atribut

n : Jumlah partisi atribut A

|Si| : Jumlah kasus pada partisi ke i

|S| : Jumlah kasus dalam S

Secara umum langkah algoritma C4.5 untuk membangun pohon keputusan adalah sebagai berikut :

a. Hitung Gain Ratio, Split Info dan entropy dari masing-masing atribut data training yang ada.
b. Buat simpul akar dari pemilihan atribut yang memiliki Gain Ratio terbesar.
c. Hitung Gain Ratio, Split Info dan entropy dari masing-masing atribut dengan menghilangkan atribut yang telah dipilih sebelumnya.
d. Buat simpul internal dari pemilihan atribut yang memiliki Gain Ratio terbesar.
e. Cek apakah semua atribut sudah dibentuk pada pohon. Jika belum, maka ulangi proses d dan e, jika sudah maka lanjut pada proses berikutnya.
f. Lakukan pemangkasan pohon untuk menghilangkan cabang-cabang yang tidak perlu. (Raharja, 2012 : 2).

II.2.1. Studi Kasus Algoritma C4.5

Untuk memudahkan penjelasan mengenai algorita C4.5, berikut ini disertakan contoh kasus yang dituangkan dalam tabel II.1.

Tabel II.1. Kasus Keputusan Algoritma

No	Outlook	Temperature	Humidity	Windy	Play
1	Sunny	Hot	High	FALSE	No
2	Sunny	Hot	High	TRUE	No
3	Cloudy	Hot	High	FALSE	Yes
4	Rainy	Mild	High	FALSE	Yes
5	Rainy	Cool	Normal	FALSE	Yes
6	Rainy	Cool	Normal	TRUE	Yes
7	Cloudy	Cool	Normal	TRUE	Yes
8	Sunny	Mild	High	FALSE	No
9	Sunny	Cool	Normal	FALSE	Yes
10	Rainy	Mild	Normal	FALSE	Yes
11	Sunny	Mild	Normal	TRUE	Yes

12	Cloudy	Mild	High	TRUE	Yes
13	Cloudy	Hot	Normal	FALSE	Yes
14	Rainy	Mild	High	TRUE	No

(Sumber : Kusrini & Lutfi, 2009: 15)

Dalam kasus yang pada tabel II.1 akan dibuat pohon keputusan untuk menentukan *main* tenis atau tidak dengan melihat keadaan cuaca, temperature, kelembapan, dan keadaan angin.

Secara umum algoritma C4.5 untuk membangun pohon keputusan adalah sebagai berikut.

a. Pilih atribut sebagai akar
b. Buat cabang untuk tiap-tiap nilai
c. Bagai kasus dalam cabang
d. Ulangi proses untuk setiap cabang sampai semua kasus pada cabang memiliki kelas yang sama

Untuk memilih atribut sebagai akar, didasarkan pada nilai *gain* tertinggi dari atribut-atribut yang ada. Untuk menghitung *gain* digunakan rumus seperti tertera dalam persamaan 1 berikut.

$$Gain\ (S, A) = Entropy\ (S) - \sum_{i=1}^{n} \frac{|Si|}{|S|} * Entropy\ (Si) \quad \text{.....(3)}$$

Keterangan :

S : himpunan kasus

A : atribut

n : jumlah partisi atribut A

|Si| : jumlah kasus pada partisi ke-i

|S| : jumlah kasus dalam S

Sementara itu, penghitungan nilai entropi dapat dilihat pada persamaan berikut.

$$Entropy(S) = \sum_{j=1}^{n} - Pj \log 2 \ Pj \text{.....(4)}$$

Keterangan :

S : himpunan kasus

A : fitur

n : jumlah partisi S

$|p_i|$: proporsi dari Si terhadap S

Berikut ini adalah penjelasan lebih terperinci mengenai tiap-tiap langkah dalam pembentukan pohon keputusan dengan mengunakan algoritma C4.5 untuk menyelesaikan permasalah pada tabel II.1.

a. Mengitung jumlah kasus, jumlah kasus untuk keputusan ***Yes,*** jumlah kasus untuk keputusan ***No,*** dan *Entropy* dari semua kasus dan kasus yang dibagi berdasarkan atribut ***OUTLOOK***, ***TEMPERATURE, HUMIDITY, dan WINDY.*** Setelah itu, lakukan penghitungan *gain* untuk setiap atribut. Hasil perhitungan ditunjukkan.

$\left(\frac{4}{14} * log_2\left(\frac{4}{14}\right)\right) + \left(-\frac{10}{14} * log_2\left(\frac{10}{14}\right)\right)$Baris **TOTAL** kolom *Entropy* pada perhitungan dengan persamaan 2 sebagai berikut :

Entropy (Total) =

Entropy (Total) = 0.863120569

Sementara itu, nilai *gain* pada baris ***OUTLOOK*** dihitung dengan menggunakan persamaan 1 sebagai berikut.

$$-\sum_{k=0}^{n} \frac{|Outlook_i|}{|\text{Total}|} * \text{Entropy}(Outlook_i)$$

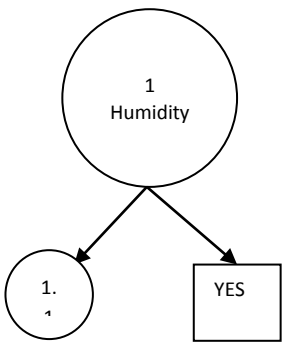
$$-\left(\left(\frac{4}{14} * 0\right) + \left(\frac{5}{14} * 0.723\right) + \left(\frac{5}{14} * 0.97\right)\right)$$*Gain(Total, Outlook*) = *Entropy (Total)*

Gain(Total, Outlook) = 0.863120569

Gain(Total, Outlook) = 0.23

Dari hasil yang diketahui bahwa atribut dengan *Gain* tertingi adalah HUMIDITY, yaitu sebesar 0.37. dengan demikian *HUMIDITY* dapat menjadi *node* akar. Ada dua nilai atribut dari *HUMIDITY*, yaitu *HIGH* dan NORMAL. Dari kedua nilai atribut tersebut, nilai atribut NORMAL sudah mengklasifikasi kasus perhitungan lebih lanjut, tetapi untuk nilai atribut *HIGH* masih perlu dilkaukan perhitungan lagi.

Dari hasil tersebut dapat digmbarkan pohon keputusan sementara tampak seperti gambar II.1



Gambar II.1. Pohon Keputusan

(Sumber : Kusrini & Lutfi, 2009: 18)

II.3. Pengertian Sistem

Secara leksikal, sistem berarti susunan yang teratur dari pandangan, teori, asas dan sebagainya. Dengan kata lain, sistem adalah suatu kesatuan usaha yang terdiri dari bagian-bagian yang berkaitan satu sama lain yang berusaha mencapai suatu tujuan dalam suatu lingkungan kompleks. Pengertian tersebut mencerminkan adanya beberapa bagian dan hubungan antara bagian, ini menunjukkan kompleksitas dari sistem yang meliputi kerja sama antara bagian yang interpenden satu sama lain. Selain itu dapat dilihat bahwa sistem berusaha mencapai tujuan. Pencapaian tujuan ini menyebabkan timbulnya dinamika, perubahan-perubahan yang terus menerus perlu dikembangkan dan dikendalikan. Definisi tersebut menunjukkan bahwa sistem sebagai gugus dan elemen-elemen yang saling berinteraksi secara teratur dalam rangka mencapai tujuan atau subtujuan (Marimin, 2008 : 1).

Sistem merupakan sekumpulan elemen-elemen yang saling terintegrasi serta melaksanakan fungsinya masing-masing untuk mencapai tujuan yang telah ditetapkan. Karakteristik sistem terdiri dari :

1. Komponen Sistem

   Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, yang artinya saling bekerja sama membentuk suatu kesatuan. Komponen-komponen

sistem atau elemen-elemen sistem dapat berupa suatu subsistem atau bagian-bagian dari sistem.

2. Batasan Sistem

   Batasan merupakan daerah yang membatasi antara suatu sistem dengan sistem yang lainnya atau dengan lingkungan luarnya. Batasan sistem ini memungkinkan suatu sistem dipandang suatu kesatuan. Batasan suatu sistem menunjukkan ruang lingkup (*scope*) dari sistem tersebut.

3. Lingkungan Luar Sistem

   Lingkungan luar dari suatu sistem adalah apapun diluar batas dari sistem yang mempengaruhi operasi sistem. Lingkung luar sistem dapat bersifat menguntungkan dan dapat juga bersifat merugikan sistem tersebut.

4. Penghubung Sistem

   Penghubung merupakan media penghubung antara satu subsistem dengan subsistem lainnya. Melalui penghubung ini memungkinkan sumber-sumber daya mengalir dari satu subsistem ke subsistem lainnya.

5. Masukan Sistem

   Masukan sistem adalah energi yang dimasukkan ke dalam sistem. Masukan dapat berupa masukan perawatan (*maintance input*) dan masukan sinyal (*signal input*).

6. Keluaran Sistem

Keluaran sistem adalah hasil energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna dan sisa pembuangan.

7. Pengolah Sistem

   Suatu sistem dapat mempunyai suatu bagian pengolah atau sistem itu sendiri sebagai pengolahnya. Pengolah akan mengubah masukan menjadi keluaran.

8. Sasaran Sistem

   Suatu sistem mempunyai tujuan (*goal*) atau sasaran (*objective*). Kalau suatu sistem tidak mempunyai sasaran, maka operasi sistem tidak ada gunanya (Sulindawati, 2010 : 375).

II.4. *JavaScript*

Javascript bukan bahasa berorientasi objek, melainkan bahasa berbasis objek. Bahasa berorientasi objek harus mendukung tiga konsep dasar, yaitu pengkapsulan (*encapsulation*), perwarisan (*inheritance*) dan polimorfisme (*polymorphism*). Javascript hanya mendukung pengkapsulan, itupun tidak 100% benar. Program *JavaScript* dituliskan pada file *HTML* (*.html* atau *.htm*) dengan menggunakan tag container <*SCRIPT*>. Dengan kata lain, anda tidak perlu menuliskan program *JavaScript* pada file terpisah (meskipun anda bisa juga melakukannya). Ingat bahwa yang dimaksud dengan tag container adalah tag yang diawali dengan <NAMA_TAG> dan diakhiri dengan </NAMA_TAG>. Beberapa contoh tag container adalah <HTML></HTML>, <HEAD></Body>, dsb (Pranata ; 2001 : 11).

II.5. Pengertian SQLite

SQLite adalah suatu *library* yang menerapkan mesin *database self-contained, serverless, zeroconfiguration,* dan *transactional*. *Self-contained* berati SQLite membutuhkan sedikit sekali dukungan dari *library eksternal* atau dari sistem operasi. *Serverless* berati SQLite dalam mengakses *database* baik itu *read* atau *write* dapat secara langsung dari *file database* tanpa melalui proses *server* dan tidak mendukung pengaksesan secara *remote* (artinya database *SQLite* bisa dikendalikan dari jarak jauh dengan adanya jaringan komputer ("*Computer Network*"), baik melalui jaringan lokal (intranet) atau internet), dimana kebanyakan mesin *SQL database* diterapkan sebagai proses server yang terpisah. *Zeroconfiguration* menunjukkan *SQLite* tidak membutuhkan instalasi sebelum penggunaannya. *Transactional SQLite* merupakan suatu *transaksional database*, dimana dalam melakukan perubahan proses *query* menerapkan *Atomic, Consistent, Isoalated, and Durable (ACID)* (Setiyadi & Harihayati, 2015:221)

II.6. Teknik Normalisasi

Normalisasi adalah teknik perancangan yang banyak digunakan sebagai pemandu dalam merancang basis data relasional. Pada dasarnya, normalisasi adalah proses dua langkah yang meletakkan data dalam bentuk tabulasi dengan menghilangkan kelompok berulang lalu menghilangkan data yang terduplikasi dari tabel rasional.

Teori normalisasi didasarkan pada konsep bentuk normal. Sebuah tabel relasional dikatakan berada pada bentuk normal tertentu jika tabel memenuhi himpunan batasan tertentu. Ada lima bentuk normal yang tekah ditemukan.

II.6.1.Bentuk-bentuk Normalisasi

1. **Bentuk normal tahap pertama (1" Normal Form)**

   Contoh yang kita gunakan di sini adalah sebuah perusahaan yang mendapatkan barang dari sejumlah pemasok. Masing-masing pemasok berada pada satu kota. Sebuah kota dapat mempunyai lebih dari satu pemasok dan masing-masing kota mempunyai kode status tersendiri.

2. **Bentuk normal tahap kedua (2^{nd} normal form)**

   Definisi bentuk normal kedua menyatakan bahwa tabel dengan kunci utama gabungan hanya dapat berada pada 1NF, tetapi tidak pada 2NF. Sebuah tabel relasional berada pada bentuk normal kedua jika dia berada pada bentuk normal kedua jika dia berada pada 1NF dan setiap kolom bukan kunci yang sepenuhnya tergantung pada seluruh kolom yang membentuk kunci utama.

3. **Bentuk normal tahap ketiga (3^{rd} normal form)**

   Bentuk normal ketiga mengharuskan semua kolom pada tabel relasional tergantung hanya pada kunci utama. Secara definisi, sebuah tabel berada pada bentuk normal ketiga (3NF) jika tabel sudah berada

pada 2NF dan setiap kolom yang bukan kunci tidak tergantung secara transitif pada kunci utamanya.

4. **Boyce Code Normal Form (BCNF)**

   Setelah 3NF, semua masalah normalisasi hanya melibatkan tabel yang mempunyai tiga kolom atau lebih dan semua kolom adalah kunci. Banyak praktisi berpendapat bahwa menempatkan entitas pada 3NF sudah cukup karena sangat jarang entitas yang berada pada 3NF bukan merupakan 4NF dan 5NF.

5. **Bentuk Normal Tahap Keempat dan Kelima**

   Sebuah tabel relasional berada pada bentuk normal keempat (4NF) jika dia dalam BCNF dan semua ketergantungan multivalue merupakan ketergantungan fungsional. Bentuk normal keempat (4NF) didasarkan pada konsep ketergantungan multivalue (MVD).

   Sebuah tabel berada pada bentuk normal kelima (5NF) jika ia tidak dapat mempunyai dekomposisi lossless menjadi sejumlah tabel lebih kecil. Empat bentuk normal pertama berdasarkan pada konsep ketergantungan fungsional, sedangkan bentuk normal kelima berdasarkan pada konsep ketergantungan gabungan (*join dependence*) (Supardi, 2010 : 448).

II.7. UML (*Unified Modeling Language*)

Menurut Windu Gata (2013 : 4) Hasil pemodelan pada OOAD terdokumentasikan dalam bentuk *Unified Modeling Language* (UML). UML

adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak.

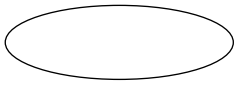
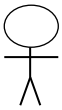
UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem. UML saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodelan umum dalam industri perangkat lunak dan pengembangan sistem.

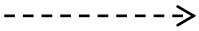
Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut :

1. *Use case* Diagram

   Use case diagram merupakan pemodelan untuk kelakukan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Dapat dikatakan *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Simbol-simbol yang digunakan dalam *use case* diagram, yaitu :

Tabel II.2. Simbol *Use Case*

Gambar	Keterangan
	*Use case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukan pesan antar unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal nama *use case*.
	Aktor adalah *abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasikan aktor, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. Orang atau sistem bisa muncul dalam beberapa peran. Perlu dicatat bahwa aktor berinteraksi dengan

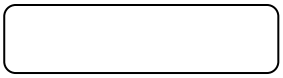
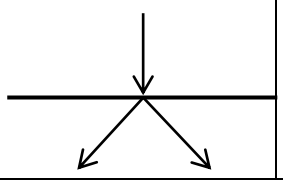
	use case, tetapi tidak memiliki control terhadap *use case*.
	Asosiasi antara aktor dan *use case*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengidikasikan aliran data.
	Asosiasi antara aktor dan *use case* yang menggunakan panah terbuka untuk mengidinkasikan bila aktor berinteraksi secara pasif dengan sistem.
	*Include*, merupakan di dalam *use case* lain (*required*) atau pemanggilan *use case* oleh *use case* lain, contohnya adalah pemanggilan sebuah fungsi program.
	Extend, merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi.

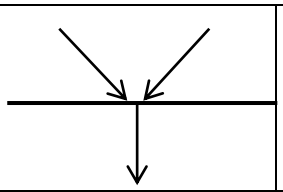
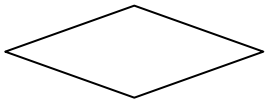
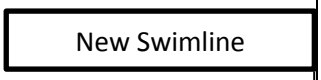
(Sumber :Gata, 2013 : 4)

2. Diagram Aktivitas (*Activity Diagram*)

Activity Diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Simbol-simbol yang digunakan dalam *activity diagram*, yaitu :

Tabel II.3. Simbol *Activity Diagram*

Gambar	Keterangan
	*Start point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas.
	*End point*, akhir aktifitas.
	*Activites*, menggambarkan suatu proses/kegiatan bisnis.
	*Fork* (Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan pararel menjadi satu.

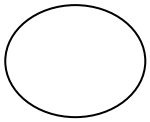
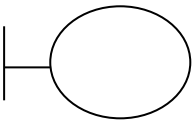
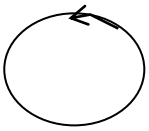
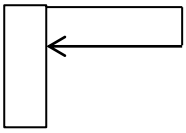
	*Join* (penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi.
	*Decision Points*, menggambarkan pilihan untuk pengambilan keputusan, *true*, *false*.
	*Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa.

(Sumber : Gata, 2013 : 6)

3. Diagram Urutan (*Sequence Diagram*)

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Simbol-simbol yang digunakan dalam *sequence diagram*, yaitu :

Tabel II.4. Simbol *Sequence Diagram*

Gambar	Keterangan
	*Entity Class,* merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data.
	*Boundary Class,* berisi kumpulan kelas yang menjadi *interface* atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan formentry dan *form* cetak.
	*Control class,* suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas, contohnya adalah kalkulasi dan aturan bisnis yang melibatkan berbagai objek.
	*Message*, simbol mengirim pesan antar *class*.
	*Recursive,* menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri.
	*Activation, activation* mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivitas sebuah operasi.
	Lifeline, garis titik-titik yang terhubung dengan objek, sepanjang *lifeline* terdapat *activation.*

(Sumber : Gata, 2013 : 7)

4. *Class Diagram* (Diagram Kelas)

   Merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggng jawab entitas yang menentukan perilaku sistem.

Class diagram juga menunjukkan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan objek yang dikoneksikan. *Class diagram* secara khas meliputi: Kelas (*Class*), Relasi, *Associations*, *Generalization* dan *Aggregation*, Atribut (*Attributes*), Operasi (*Operations/Method*), *Visibility*, tingkat akses objek eksternal kepada suatu operasi atau atribut. Hubungan antar kelas mempunyai keterangan yang disebut dengan *multiplicity* atau kardinaliti.

Tabel II.5. *Multiplicity Class Diagram*

Multiplicity	**Penjelasan**
1	Satu dan hanya satu
0..*	Boleh tidak ada atau 1 atau lebih
1..*	1 atau lebih
0..1	Boleh tidak ada, maksimal 1
n..n	Batasan antara. Contoh 2..4 mempunyai arti minimal 2 maksimum 4

(Sumber : Gata, 2013 : 9)