**BAB III**

**TINJAUAN PUSTAKA**

1. ***Data Mining***

Data mining adalah suatu istilah yang digunakan untuk menguraikan penemuan pengetahuan di dalam basis data. Data Mining adalah proses yang menggunakan teknik statistik, matematika, kecerdasan buatan, *machine learning* untuk mengekstraksi dan mengidentifikasi informasi yang bermanfaat dan pengetahuan yang terkait dari berbagai basis data besar (Kursini & Emha, 2009).

Data mining adalah suatu proses ekstraksi atau penggalian data yang belum diketahui sebelumnya, namun dapat dipahami dan berguna dari database yang besar serta digunakan untuk membuat suatu keputusan bisnis yang sangat penting (Connolly & Begg, 2010).

Pada prosesnya data mining akan mengekstrak informasi yang berharga dengan cara menganalisis adanya pola-pola ataupun hubungan keterkaitan tertentu dari data-data yang berukuran besar. Data mining berkaitan dengan bidang ilmu ilmu lain, seperti *Database System, Data Warehousing, Statistic, Machine Learning, Information Retrieval*, dan Komputasi Tingkat Tinggi. Selain itu data mining didukung oleh ilmu lain seperti *Neural Network*, Pengenalan Pola, *Spatial Data Analysis*, *Image Database, Signal Processing* (Connolly & Begg, 2010).

Berikut adalah tahap-tahap dalam data mining (Huda, 2010).

Untuk penelitian ini Cuma menggunakan 3 tahapan di data mining sebagai berikut:

1. Seleksi Data (Data Selection)

Data yang ada pada database sering kali tidak semuanya dipakai, oleh karena itu hanya data yang sesuai untuk dianalisis yang akan diambil dari database.

1. Transformasi Data (Data Transformation)

Data diubah atau digabung ke dalam format yang sesuai untuk diproses dalam data mining. Beberapa metode data mining membutuhkan format data yang khusus sebelum bisa diaplikasikan. Sebagai contoh beberapa metode standar seperti analisis asosiasi dan clustering hanya bisa menerima input data kategorikal. Karenanya data berupa angka numerik yang berlanjut perlu dibagi-bagi menjadi beberapa interval. Proses ini sering disebut transformasi data.

1. Proses *Mining*

Merupakan suatu proses utama saat metode diterapkan untuk menemukan pengetahuan berharga dan tersembunyi dari data.

1. Evaluasi Pola (*Pattern Evaluation*)

Berfungsi untuk mengidentifikasi pola-pola menarik ke dalam *knowledge based* yang ditemukan. Dalam tahap ini hasil dari teknik data mining berupa pola-pola yang khas maupun model prediksi dievaluasi untuk menilai hipotesa yang ada memang tercapai. Bila ternyata hasil yang diperoleh tidak sesuai hipotesa ada beberapa alternatif yang dapat diambil seperti menjadikannya umpan balik untuk memperbaiki proses data mining, mencoba metode data mining lain yang lebih sesuai, atau menerima hasil ini sebagai suatu hasil yang di luar dugaan yang mungkin bermanfaat.

1. Presentasi Pengetahuan (*Knowledge*)

Merupakan visualisasi dan penyajian pengetahuan mengenai metode yang digunakan untuk memperoleh pengetahuan yang diperoleh pengguna. Tahap terakhir dari proses data mining adalah memformulasikan keputusan atau aksi dari hasil analisis yang didapat. Ada kalanya hal ini harus melibatkan orang-orang yang tidak memahami data mining. Oleh sebab itu presentasi hasil data mining dalam bentuk pengetahuan yang bisa dipahami semua orang adalah satu tahapan yang diperlukan dalam proses data mining. Dalam presentasi ini, visualisasi juga bisa membantu mengkomunikasikan hasil data mining.

1. **Klasifikasi**

Dalam data mining terdapat beberapa teknik yang memiliki fungsi berbeda-beda yaitu prediksi, estimasi, deskripsi, klasifikasi, pengklasteran dan asosiasi. Klasifikasi merupakan metode yang digunakan untuk menemukan model atau fungsi yang digambarkan dengan perbedaan kelas data atau konsep yang berfungsi untuk memprediksi kelas dari objek yang label sudah diketahui (Annur, 2018).

Proses penemuan model (atau fungsi) yang menggambarkan dan membedakan kelas data atau konsep yang bertujuan agar bisa digunakan untuk memprediksi kelas dari objek yang label kelasnya tidak diketahui Algoritma klasifikasi yang banyak digunakan secara luas, yaitu Decision classification trees, Bayesian classifiers Naïve Bayes classifiers, Neural networks, Analisa Statistik, Algoritma Genetika, Rough sets, k-nearest neighbor, Metode Rule Based, Memory based reasoning, dan Support vector machines (Annur, 2018).

Dalam klasifikasi akan terjadi 2 proses, proses pertama adalah learning (fase training), dimana algoritma klasifikasi diciptakan untuk menganalisa data training kemudian direpresentasikan ke dalam bentuk rule klasifikasi. Proses kedua adalah klasifikasi, dimana data testing akan dipakai untuk memperkirakan akurasi dari rule klasifikasi (Gorunescu, 2010).

Terdapat komponen yang mendasari proses klasifikasi sebagai berikut (Gorunescu, 2010):

1. *Class*

Variabel dependen atau variabel yang memiliki ketergantungan (terikat) berupa data kategorikal yang merepresentasikan suatu label yang melekat pada objek.

1. *Predictor*

Variabel dependen variabel yang memiliki ketergantungan (terikat) yang direpresentasikan oleh (data) atribut tertentu.

1. *Training dataset*

Sekumpulan data yang mengandung nilai dari kedua komponen sebelumnya yang dimanfaatkan untuk pemilihan kelas yang tepat berdasarkan predictor.

1. *Testing dataset*

Sekumpulan data baru yang akan melalui proses klasifikasi menggunakan model yang telah ditentukan. Hasil dari klasifikasi tersebut yang kemudian akan dievaluasi keakuratannya.

1. **Algoritma Naïve Bayes**

Naïve Bayes Classifier dikenal sebagai pengklasifikasi bayesian sederhana dan telah menjadi model probabilistik yang penting dan telah berhasil dalam praktiknya. Walaupun memiliki asumsi independensi yang kuat, Naïve Bayes Classifier telah terbukti efektif dalam klasifikasi dalam bentuk teks, diagnosa medis dan manajemen kinerja computer (Rahman & Kurniawan, 2018).

Naive Bayes merupakan sebuah pengklasifikasian probalistik sederhana yang menghitung sekumpulan probabilitas dengan menjumlahkan frekuensi dan kombinasi nilai dari dataset yang diberikan. Algoritma menggunakan teorema bayes dan mengansumsikan semua atribut independen atau tidak saling ketergantungan yang diberikan oleh nilai pada variabel kelas. Naive Bayes juga didefinisikan sebagai pengklasifikasian dengan metode probabilitas dan statistik yang dikemukakan oleh ilmuan inggis Thomas Bayes, yaitu memprediksi peluang di masa depan berdasarkan pengalaman di masa sebelumnya (Septiari, 2016).

Pada sub bab ini akan dijelaskan mengenai teorema Bayes, Naïve Bayes untuk klasifikasi, serta karakteristik Naïve Bayes.

1. **Teorema Bayes**

Bayes merupakan teknik prediksi berbasis probabilistik sederhana yang berdasar pada penerapan teorema Bayes (atau aturan Bayes) dengan asumsi independensi (ketidaktergantungan) yang kuat (naif). Dengan kata lain, dalam Naïve Bayes, model yang digunakan adalah “model fitur independen” (Prasetyo, 2012).

Metode Naive Bayes Classifier menggunakan konsep probabilitas yang bertujuan untuk melakukan klasifikasi data pada class tertentu, metode Naive Bayes Classifier merupakan penyederhanaan dari teorema bayes. Prediksi bayes didasarkan pada teorema bayes dengan formula umum sebagai berikut: (Uriawan & Yusfira, 2018)

Penjelasan dari formula 1 adalah sebagai berikut (Uriawan & Yusfira, 2018):

P(H|E) : Probabilitas akhir bersyarat (conditional probability) suatu hipotesis H terjadi jika diberikan bukti (evidence) E terjadi.

P(E|H) : Probabilitas sebuah bukti E terjadi akan pengaruh hipotesis H

P(H) : Probabilitas awal (Priori) hipotesis H terjadi tanpa memandang bukti apapun.

P(E) : Probabilitas awal (Priori) bukti E terjadi tanpa memandang hipotesis/bukti yang lain.

Ide dasar dari aturan Bayes adalah bahwa hasil dari hipotesis atau peristiwa (H) dapat diperkirakan berdasarkan pada beberapa bukti (E) yang diamati. Ada beberapa hal penting dari aturan Bayes tersebut, yaitu:

1. Sebuah probabilitas awal/priori H atau P(H) adalah probabilitas dari suatu hipotesis sebelum bukti diamati.
2. Sebuah probabilitas akhir H atau P(H|E) adalah probabilitas dari suatu hipotesis setelah bukti diamati.

Teorema Bayes juga bisa menangani beberapa bukti, misalnya ada E1, E2, dan E3, sehingga probabilitas akhir dapat dihitung dengan cara berikut (Prasetyo, 2012).

Karena asumsi yang digunakan untuk bukti adalah independen, bentuk formula 2 dapat diubah menjadi sebagai berikut (Prasetyo, 2012).

1. **Naïve Bayes untuk Klasifikasi**

NBC adalah metode klasifikasi yang berdasarkan probabilitas dan Teorema Bayesian dengan asumsi bahwa setiap variable X bersifat bebas (independence). Dengan kata lain, NBC mengansumsikan bahwa keberadaan sebuah atribut (variable) tidak ada kaitannya dengan keberadaan atribut (variable) yang lain. Metode NBC menempuh dua tahap dalam proses klasifikasi teks,yaitu tahap pelatihan dan tahap klasifikasi (Indriani, 2014).

Kaitan antara Naïve Bayes dengan klasifikasi, korelasi hipotesis, dan bukti dengan klasifikasi adalah bahwa hipotesis dalam teorema Bayes merupakan label kelas yang menjadi target pemetaan dalam klasifikasi, sedangkan bukti merupakan fitur-fitur yang menjadi masukan dalam model klasifikasi. Jika X adalah vektor masukan yang berisi fitur dan Y adalah label kelas, Naïve Bayes dituliskan dengan P(Y|X). Notasi tersebut berarti probabilitas label kelas Y didapatkan setelah fitur-fitur X diamati. Notasi ini disebut juga probabilitas akhir (posterior probability) untuk Y, sedangkan P(Y) disebut probabilitas awal (prior probability) Y.

Selama proses pelatihan harus dilakukan pembelajaran probabilitas akhir (P(Y|X) pada model untuk setiap kombinasi X dan Y berdasarkan informasi yang didapat dari data latih. Dengan membangun model tersebut, suatu data uji X’ dapat diklasifikasikan dengan mencari nilai Y’ dengan memaksimalkan nilai P(Y’|X’) yang didapat.

Formulasi Naïve Bayes untuk klasifikasi adalah sebagi berikut (Prasetyo, 2012).

P(Y|X) adalah probabilitas data dengan vector X pada kelas Y. P(Y) adalah probabilitas awal kelas Y. adalah probabilitas independen kelas Y dari semua fitur dalam vector X. Nilai P(X) selalu tetap sehingga dalam perhitungan prediksi nantinya hanya menghitung bagian P(Y) dengan memilih yang terbesar sebagai kelas yang dipilih sebagai hasil prediksi. Sementara probabilitas independen tersebut merupakan pengaruh semua fitur dari data terhadap setiap kelas Y, yang dinotasikan dengan persamaan berikut (Prasetyo, 2012).

Setiap set fitur X = {, , , …, } terdiri atas q atribut (q dimensi).

1. **Karakteristik Naïve Bayes**

Klasifikasi dengan *Naïve Bayes* bekerja berdasarkan teori probabilitas yang memandang semua fitur dari data sebagai bukti dalam probabilitas. Hal ini memberikan karakteristik *Naïve Bayes* sebagai berikut (Laroussi, 2015).

1. Metode *Naïve Bayes* teguh (*robust*) terhadap data-data yang terisolasi yang biasanya merupakan data dengan karakteristik berbeda (*outlier*). *Naïve Bayes* juga bisa menangani nilai atribut yang salah dengan mengabaikan data latih selama proses pembangunan model dan prediksi.
2. Tangguh menghadapi atribut yang tidak relevan.
3. Atribut yang mempunyai korelasi bisa mendegradasi kinerja klasifikasi *Naïve Bayes* karena asumsi independensi atribut tersebut sudah tidak ada.(Laroussi, 2015).
4. ***Framework***

Web Aplikasi Framework (WAF), atau yang sering disingkat Web Framework adalah satu kumpulan kode berupa pustaka (library) dan alat (tool) yang dipadukan sedemikian rupa menjadi satu kerangka kerja (framework) guna memudahkan dan mempercepat proses pengembangan aplikasi web.

Framework secara sederhana dapat diartikan kumpulan dari fungsi – fungsi atau prosedur-prosedur dan class-class untuk tujuan tertentu yang sudah siap digunakan sehingga bias lebih mempermudah dan mempercepat pekerjaan seorang programmer, tanpa harus membuat fungsi atau class dari awal (Raharjo, 2012)..

1. ***CodeIgniter***

CodeIgniter adalah framework web untuk Bahasa pemrograman PHP, yang dibuat oleh Rick Ellis pada tahun 2006, penemu dan pendiri Ellislab ([www.ellislab.com](http://www.ellislab.com)). Ellislab adalah suatu tim kerja yang berdiri pada tahun 2002 dan bergerak di bidang pembuatan softwere dan tool untuk para pengembang web. Sejak tahun 2014 sampai sekarang, Ellipslab telah menyerahkan hak kepemilikan Codeigniter ke British Columbia Institute of Technologi (BCIT) untuk proses perkembangan lebih lanjut. Saat ini situs resmi dari codeigniter telah berubah dari [www.ellislab.com](http://www.ellislab.com) ke [www.codeigniter.com](http://www.codeigniter.com) (Raharjo, 2012).

Codeigniter merupakan aplikasi sumber terbuka yang berupa framework PHP dengan model MVC (model, View, Controler) untuk membangun website dinamis menggunakan PHP (Raharjo, 2012).

Arsitektur atau sering di sebut pola Model-View-Controler (MVC) muncul sejak era 70-an atas pemikiran Prof.Trygve Reenskaug, seorang berkebangsaan Norwegia. Dasar arsitektur ini adalah memisahkan antara logika aplikasi dengan tampilan. Menggunakan pola ini diharapkan dapat meminimalisasi penulisan perintah, sehingga resiko terjadinya bug juga minimal, serta meningkatkan efisiensi aplikasi (Raharjo, 2012).

Codeigniter pertama kali dirilis pada 28 Februari 2006. Versi stabil terakhir adalah versi 3.1.6.

Terdapat 3 jenis komponen yang membangun suatu MVC :

1. *Model*

Biasanya berhubungan langsung dengan database untuk memanipulasi data (insert, update, delete), menangani validasi dari bagian controller, namun tidak dapat berhubungan langsung dengan bagian view.

1. *View*

merupakan bagian yang menangani presentation logic. Pada suatu aplikasi web bagian ini biasanya berupa file template HTML, yang diatur oleh controller. View berfungsi untuk menerima dan mempresentasikan data kepada user. Bagian ini tidak memiliki akses langsung terhadap bagian model.

1. *Controller*

merupakan bagian yang mengatur hubungan antara bagian model dan view, controller berfungsi untuk menerima request data dari user kemudian menentukan apa yang akan diproses oleh aplikasi.

Kelebihan Codeigniter (CI) di bandingkan dengan framework PHP lain diantaranya (Raharjo, 2012):

1. Performa sangat cepat.
2. Konfigurasi yang sangat minim (nearly zero configuration).
3. Banyak komunitas.
4. Dokumentasai yang sangat lengkap.
5. ***Basis Data MySQL***

Basis data adalah sekumpulan informasi yang diatur agar mudah dicari. Dalam arti umum basis data adalah sekumpulan data yang diproses dengan bantuan komputer yang memungkinkan data dapat diakses dengan mudah dan tepat, yang dapat digambarkan sebagai aktivitas dari satu atau lebih organisasi yang berelasi (Kustiyahningsih & Rosa, 2011).

MySQL merupakan suatu database. MySQL dapat juga dikatakan sebagai database yang sangat cocok bila dipadukan dengan PHP. Secara umum, database berfungsi sebagai tempat atau wadah untuk menyimpan, mengklasifikasikan data secara profesional. MySQL bekerja menggunakan SQL Language (Structure Query Language). Hal itu dapat diartikan bahwa MySQL merupakan standar penggunaan database di dunia untuk pengolahan data (Kustiyahningsih & Rosa, 2011).

MySQL termasuk jenis RDBMS (Relational Database Management System). Sedangkan RDBMS sendiri lebih banyak mengenal istilah seperti tabel, baris, dan kolom digunakan dalam perintah-perintah di MySQL. MySQL merupakan sebuah basis data yang mengandung satu atau sejumlah tabel. Tabel terdiri atas sejumlah baris dan setiap baris mengandung satu atau beberapa kolom. Di dalam PHP telah menyediakan fungsi untuk koneksi ke basis data dengan sejumlah fungsi untuk pengaturan baik menghubungkan maupun memutuskan koneksi dengan server database MySQL sebagai sarana untuk mengumpulkan informasi (Kadir, 2013).

Pada umumnya, perintah yang paling sering digunakan dalam mySQL adalah select (mengambil), insert (menambah), update (mengubah), dan delete (menghapus). Selain itu, SQL juga menyediakan perintah untuk membuat database, field, ataupun index guna menambah atau menghapus data (Kadir, 2013).

1. ***Unified Modelling Language* (UML)**

UML adalah sebuah “bahasa” yang telah menjadi standar dalam industri untuk *visualisasi*, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah system (Sugiarti, 2013).

Unified Modelling Language (UML) biasa digunakan untuk :

1. Menggambarkan batasan sistem dan fungsi-fungsi sistem secara umum, dibuat dengan use case dan actor
2. Menggambarkan kegiatan atau proses bisnis yang dilaksanakan secara unum, dibuat dengan interaction diagrams.
3. Menggambarkan representasi struktur statis sebuah sistem dalam bentuk class diagrams.
4. Membuat model behavior “yang menggambarkan kebiasaan atau sifat sebuah sistem” dengan state transition diagrams.
5. Menyatakan arsitektur implementasi fisik menggunakan component and development.
6. Menyampaikan atau memperluas fungsionlity dengan stereotyes.
   * 1. ***Use Case Diagram***

*Use Case* Diagram atau Diagram Use Case merupakan permodelan untuk menggambarkan kelakuan (*behavior*) sistem satu yang akan dibuat (Sugiarti, 2013).

Adapun simbol-simbol yang digunakan dalam *use case* seperti ditunjukkan pada Tabel 3.1

Tabel 3.1 Simbol-simbol dalam *Use Case Diagram* (Rosa, 2014).

| **SIMBOL** | **NAMA** | **KETERANGAN** |
| --- | --- | --- |
| Description: use_case.png | *Use Case* | Fungsionalitas yang disediakan oleh sistem sebagai unit atau *actor* yang biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama *use case*. |

Tabel Lanjutan 3.1 Simbol – simbol *Use Case* (Sugiarti, 2013).

|  |  |  |
| --- | --- | --- |
| **SIMBOL** | **NAMA** | **KETERANGAN** |
| Description: use_case - Copy.png | *Actor* | Orang atau proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan di buat di luar sistem informasi yang akan di buat itu sendiri |
| Description: use_case - Copy (2).png | *Assosiation* | Komunikasi antara *actor* dan *use case* yang berpartisipasi pada *use case* atau *use case* yang memiliki interaksi dengan aktor |
| Description: use_case - Copy (3) - Copy.png | *Extends* | Relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan dapat berdiri sendiri walapun tanpa use case tambahan itu; mirip dengan prinsip inheritance pada pemrograman berorientasi objek; biasanya use case arah panah menunjukan pada use case yang dituju. |
| Description: use_case - Copy (4).png | *Include* | Relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan memerlukan use case ini untuk menjalankan fungsinya atau sebagai syarat dijalankan use case. |

**3.7.2 *Activity Diagram***

Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, decision yang mungkin terjadi dan bagaimana mereka berakhir. Activity diagram merupakan state diagram khusus, di mana sebagian besar state adalah action dan sebagian besar transisi di trigger oleh selesainya state sebelumnya (Sugiarti, 2013).

Adapun simbol-simbol yang digunakan dalam *activity* diagram seperti ditunjukkan pada Tabel 3.2.

Tabel 3.2 simbol-simbol *Activity* Diagram (Sugiarti, 2013).

|  |  |  |
| --- | --- | --- |
| **SIMBOL** | **NAMA** | **KETERANGAN** |
|  | *Start state* | Titik awal atau permulaan |
|  | *End state* | Titik akhir atau akhir dari aktivitas |
|  | *Activity* | *Activity* atau aktivitas yang dilakukan oleh *actor* |
|  | *Decision* | Pilihan untuk mengambil keputusan |
|  | *Transition* | Sebuah kejadian yang memicu sebuah state objek dengan cara memperbaharui satu atau lebih nilai atributnya |
|  | *State* | State dari system yang mencerminkan eksekusi dari sistem |

Tabel Lanjutan 3.2 simbol-simbol Activity Diagram (Sugiarti, 2013).

|  |  |  |
| --- | --- | --- |
| **SIMBOL** | **NAMA** | **KETERANGAN** |
|  | *Join* | Untuk menggabungkan beberapa kegiatan secara pararel menjadi satu |
|  | *Fork* | Menunjukan kegiatan yang dilakukan secara pararel |

**3.7.3 *Sequence Diagram***

*Sequence Diagram* menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan *Sequence* Diagrammaka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu. Membuat *Sequence Diagram* juga dibutuhkan untuk melihat skenario yang ada pada *use case* (Sugiarti, 2013).

Adapun simbol-simbol yang digunakan dalam *Sequence Diagram* seperti ditunjukkan pada Tabel 3.3.

Tabel 3.3 Simbol-simbol dalam *Sequence Diagram* (Sugiarti, 2013).

| **SIMBOL** | **NAMA** | **KETERANGAN** |
| --- | --- | --- |
|  | *Actor* | Orang ataupun pihak yang akan mengelola system. |
|  | *Garis Hidup /*  *Lifeline* | Menggambarkan sebuah objek dalam sebuah sistem atau salah satu komponennya. |
|  | *Control* | Permodelan“perilaku mengatur”  khusus untuk satu atau beberapa use case saja. |
|  | *Entity* | Permodelan informasi yang harus disimpan oleh sistem yang memperlihatkan struktur data dari suatu sistem. |
|  | *Message* | Mengindifikasikan urutan komunikasi yang terjadi antar objek. |

**.**

**3.7.4 *Class Diagram***

*Class Diagram* atau diagram kelas menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun system (Sugiarti, 2013).

Adapun simbol-simbol yang digunakan dalam *Class Diagram* seperti ditunjukkan pada Tabel 3.4.

Tabel 3.4 Simbol-simbol dalam *Class Diagram* (Sugiarti, 2013).

|  |  |  |
| --- | --- | --- |
| **SIMBOL** | **NAMA** | **KETERANGAN** |
|  | *Class* | Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama |
|  | *Generalization* | Garis yang melambangkan konsep pewarisan dari sautu kelas ke satu atau lebih sub kelas |
| **........................** | *Association* | Apa yang menghubungkan antara objek satu dengan objek lainnya |
| Description: Description: class diagram simbol | *Interface* | Sama dengan konsep interfase dalam pemprograman berorientasi objek. |
| Description: Description: class diagram simbol | *Directed Association* | Relasi antar kelas dengan makna kelas yang satu digunakan dengan kelas yang lain. Asosiasi biasanya juga disertai dengan *Multiplycity.* |
| Description: Description: class diagram simbol | *Aggregation* | Relasi antar kelas dengan makna kelas yang satu digunakan dengan kelas yang lain. Asosiasi biasanya juga disertai dengan *Multiplycity.* |

1. **Pengujian Sistem**

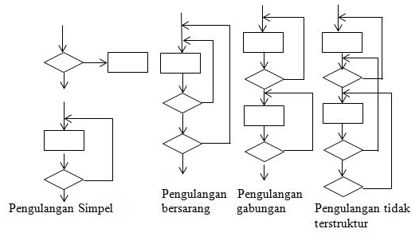
Pengujian sistem atau testing merupakan proses pengeksekusian program untuk menemukan kesalahan-kesalahan yang terdapat di dalam system, kemudian dilakukan pembenahan. Tahap ini merupakan tahap yang penting dalam pengembangan sistem, karena dalam tahap ini merupakan tahapan untuk memastikan bahwa suatu sistem terbebas dari kesalahan (Mulyanto, 2009).

1. Pengujian *Black-Box*

Pengujian kotak hitam yaitu menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. Pengujian dimaksudkan untuk mengetahui apakah fungsi-fungsi, masukan dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan. Pengujian kotak hitam dilakukan dengan membuat kasus uji yang bersifat mencoba semua fungsi dengan memakai perangkat lunak apakah sesuai dengan spesifikasi yang dibutuhkan (Rosa & Shalahuddin, 2013). Kasus uji yang dibuat untuk melakukan pengujian kotak hitam harus dibuat dengan kasus benar dan kasus salah, misalkan untuk kasus proses login maka kasus uji yang dibuat adalah:

1. Jika *user* memasukkan nama pemakai *(username)* dan kata sandi *(password)* yang benar.
2. Jika *user* memasukkan nama pemakai *(username)* dan kata sandi *(password)* yang salah, misalnya nama pemakai benar tapi kata sandi salah, atau sebaliknya, atu keduanya salah.
3. Pengujian *White-Box*

Pengujian kotak putih yaitu, menguji perangkat lunak dari segi desain dan kode program apakah mampu menghasilkan fungsi-fungsi, masukan dan keluaran yang sesuai dengan spesifikasi kebutuhan. Pengujian kotak putih dilakukan dengan memeriksa lojik dari kode program. Pembuatan kasus uji bisa mengikuti standar pengujian dari standar pemrograman yang seharusnya (Rosa & Shalahuddin, 2013). Contoh dari pengujian kotak putih misalkan menguji alur (dengan menelusuri) pengulangan *(looping)* pada logika pemrograman seperti ilustrasi berikut:



Gambar 3.1 Pengujian Kotak Putih (Rosa & Shalahuddin, 2013)

Penghitungan CC (Cyclomatic Complexity).

CC dilambangkan dengan V(G).

Rumus penghitungan CC adalah :

V(G) = E – N + 2 atau

V(G) = P + 1

Dimana V(G) = Cyclomatic Complexity

Keterangan :

E = jumlah edge pada flowgraph

N = jumlah node pada flowgraph

P = jumlah predicate node pada flowgraph