BAB II

TINJAUAN PUSTAKA

2.1 Seni Tradisional

Seni tradisional adalah unsur kesenian yang menjadi bagian hidup masyarakat dalam suatu kaum/puak/suku/bangsa tertentu. Tradisional adalah aksi dan tingkah laku yang keluar alamiah karena kebutuhan dari nenek moyang yang terdahulu. Tradisi adalah bagian dari tradisional namun bisa musnah karena ketidamauan masyarakat untuk mengikuti tradisi tersebut. Seni tradisional yang ada di suatu daerah berbeda dengan yang ada di daerah lain, meski pun tidak menutup kemungkinan adanya seni tradisional yang mirip antara dua daerah yang berdekatan.

Berdasarkan bentuk dan mediumnya seni dapat diklasifikasikan dalam lima kelompok : seni rupa, seni musik, seni tari, seni teater dan seni sastra.

Table 2.1 Macam-Macam Seni

No	Cabang Seni	Bentuk Media	Indera Penikmat	Matra
1.	Rupa	Benda	Penglihatan, peraba	2 atau 3 dimensi
2.	Musik	Suara, benda, manusia, gerak	Pendengaran, penglihatan	Waktu, 3 dimensi
3.	Tari	Tubuh manusia, gerak, music	Penglihatan, pedengaran	Waktu, 3 dimensi
4.	Teater/ Pertunjukan	Manusia, benda/alam, akting, adegan, suara/music	Penglihatan, pedengaran	Waktu, 3 dimensi
5.	Sastra	Tulisan	Penglihatan	2 dimensi

1. Seni Rupa

Seni rupa merupakan salah satu cabang kesenian. Seni rupa memiliki wujud pasti dan tetap yakni dengan memanfaatkan unsur rupa sebagai salah satu wujud yang diklasifikasikan ke dalam bentuk gambar, lukis, patung, grafis, kerajinan tangan, kriya, dan multimedia.

Kompetensi dasar yang harus dicapai bidang seni rupa adalah meliputi kemampuan memahami dan berkarya lukis, kemampuan memahami dan membuat patung, kemampuan memahami dan berkarya grafis ,kemampuan memahami dan membuat kerajinan tangan, serta kemampuan memahami dan berkarya atau membuat sarana multimedia. Terminologi in pada dasarnya telah ditetapkan sebagai kecakapan seseorang yang mampu menguasai bidang kerupawanan.

Seni rupa telah mengakar mulai zaman animisme dan dinamisme hingga jaman melenium. Seni Rupa menjadi salah satu bagian cabang seni yang secara performatif mempresentasikan wujud yang kasat mata. Ilusi tentang wujud dapat diserap dan dirasakan ke dalam klasifikasi bentuk seperti telah disebut pada bagian atas. Representasi bentuk seni rupa dipertimbangkan secara sinergis melalui perhelatan media yang digunakan sebagai dasar perwujudan rupa. Secara kontekstual seni rupa merupakan wujud mediasi bentuk kasat mata yang dekat ke arah perlambang gambar, lukis, patung, kerajinan tangan kriya dan multimedia. berhubungan dengan unsur cabang kesenian.



Gambar 2.1 Seni Rupa

2. Seni Musik

Unsur bunyi adalah elemen utama seni musik. Unsur lain dalam bentuk harmoni, melodi dan notasi musik merupakan wujud sarana yang diajarkan. Media seni musik adalah vokal dan instrumen. Karakter musik instrumen dapat berbentuk alat musik Barat dan alat musik Nusantara/tradisional. Jenis alat musik tradisional antara lain terdiri dari seruling, gambang kromong, gamelan, angklung, rebana, kecapi, dan kolintang serta arumba. Jenis alat musik Barat antara lain terdiri dari piano, gitar, flute, drum, musik elektronik, sintetiserr, seksopon, dan terompet.



Gambar 2.2 Seni Musik

Kompetensi dasar yang harus dicapai dalam mempelajari seni musik meliputi kemampuan memahami dan berkarya musik, pemahaman pengetahuan musik mencakup harmoni, melodi dan notasi musik serta kecerdasan musikal

yang memungkinkan seseorang dapat beradaptasi dengan perangkat musik secara cepat. Di sisi lain, kemampuan memahami dan membuat notasi, kemampuan mengaransemen, serta praktik dasar maupun mahir dalam banyak alat atau instrumen secara terampil, serta kemampuan memahami dan membuat multimedia. Seni musik yang lebih mempromosikan unsur bunyi sebagai medium dasar musik lebih memiliki proporsi pada bunyi yang teratur, bunyi yang berirama, serta paduan bunyi yang menjurus kepada eksperimental bunyi secara harafiah tanpa ritme, melodi maupun harmoni. Seni musik banyak berkembang pada komunitas masyarakat yang memiliki aliran klasik, ekspresionis, eksperimentalis, dan fluonsis dengan memetakan perkembangan musik melalui bunyi-bunyian yang tidak berirama dan bernada. Seni musik tumbuh-kembang sejak zaman Renaissance hingga abad milenium. Secara progresif aliran musik yang berkembang pada saat ini lebih ke arah musik yang memiliki tonasi, interval, dan harmoni secara varian.

Seni musik lebih transparan dalam bentuk hasil karyanya. Bunyi sebagai media ungkap menjadi salah satu alat komunikasi dalam menginternalisasikan makna bunyi ke dalam penerjemahan kuantum dari pikiran aranjer(penata musik) ke penonton. Oleh sebab itu, dibutuhkan pemaknaan artikulasi penataan musik terhadap cara penyampaian makna musik untuk dapat dimengerti oleh penonton. Dengan demikian makna penataan musik semakin mudah dipahami, dimengerti dan menjadi media komunikasi antara penata musik dengan penghayat musiknya.

3. Seni Tari

Media ungkap tari adalah gerak. Gerak tari merupakan gerak yang diperhalus dan diberi unsur estetis. Gerak dalam tari berfungsi sebagai media untuk mengkomunikasikan maksud-maksud tertentu dari koreografer. Keindahan tari terletak pada bentuk kepuasan, kebahagiaan, baik dari koreografer, peraga dan penikmat atau penonton.

Kompetensi dasar dalam mempelajari seni tari mencakup praktik dasar dan mahir dalam penguasaan gerak tari meliputi tari tradisional maupun tari garapan, kemampuan memahami arah dan tujuan koreografer dalam konsep koreografi kelompok. Kemampuan memahami an berkarya tari (koreografi) adalah keterampilan khusus berhubungan dengan kepekaan koreografi, di sisi lain diharapkan memiliki kepekaan memahami aspek-aspek tari dan aspek keindahan secara teknis. Sebagai penyesuaian abad modern, kemampuan memahami dan membuat perangkat multimedia hubungannya dengan tari adalah bentuk penyesuaian sumber daya manusia dalam adaptasinya dengan teknologi. Perwujudan ekspresi budaya melalui gerak yang dijiwai serta diikat nilai-nilai budaya menjadi patokan dasar atau standar ukur tari untuk dikaji menjadi bentuk tari-tarian daerah di Indonesia. Sebagai salah satu unsur terpenting kesenian di Indonesia dalam wujud performa gerak, dibutuhkan adanya kehidupan sosial dan spiritual masyarakat pendukungnya. Peran dan fungsi tarian yang begitu penting hingga kini pada puncak kesenian daerah menjadi simbol dan puncak tari sebagai budaya di daerah yang bersangkutan. Jenis tari yang telah menjadi puncak budaya

daerah sangat erat untuk dijadikan sebagai tarian yang diunggulkan daerah.di mana tarian tersebut berasal.



Gambar 2.3 Seni Tari

Beraneka ragam tari-tarian yang diwarisi masyarakat daerah di Indonesia baik yang sakral maupun yang sekuler, tradisional maupun nontradisional. Bentuk tarian dari zaman prasejarah hingga zaman modern, produk dari zaman tertentu membantu sejarah kehidupan tarian untuk dapat tumbuh-kembang hingga akhir zaman.Seni tari memerlukan media gerak. Gerak murni atau wantah tidak memiliki maksud-maksud tertentu. Gerak maknawi memiliki makna maksud-maksud tertentu dan apabila dibangun dengan unsur keindahan, maka gerakan tari semakin halus, estetis, dan geraknya memiliki bangunan ekspresi bentuk yang diungkapkan manusia untuk dinikmati.Seni tari banyak dipengaruhi oleh kepercayaan dinamisme dan animisme. Oleh sebab itu, sejak zaman dulu tarian sudah memiliki peran fungsi yang sentral dalam kehidupan beragama. Peran tari dalam upacara terkait dengan cara dan tujuan yang terkait dalam prosesi suatu upacara keagamanaan atau ritual. Seni tari mewariskan bentuk-bentuk tradisi maupun nontradisi. Sifat—fungsi magis-ritual yang dipengaruhi kepercayaan animisme dinamisme mampu menjadi kekuatan sentral dalam setiap upacara

keagamaan. Dalam perkembangannya, seni tari tradisional pada akhirnya mewariskan seni pertunjukan baru dan inovatif melalui dramatari prembun, hingga sendratari jenis kesenian yang lahir pada zaman modern. Pada masyarakat modern yang dinamis ini, kehadiran seni tari memerlukan hadirnya penari yang baik, guru-guru tari yang profesional, dan pemikir-pemikir yang mampu merumuskan masa depan tari secara proporsional. Oleh sebab itu, beberapa hal harus diperhatikan menyangkut penguasaan teknik tari agar dapat memenuhi syarat sebagai penari yang profesional.

4. Seni Teater/ Pertunjukan

Kompetensi dasar bidang seni teater mencakup kemampuan memahami dan berkarya teater, kemampuan memahami dan membuat naskah, kemampuan memahami berperan di bidang casting kemampuan memahami dan membuat setting atau tata teknik pentas panggung dan penciptaan suasananya sebagai perangkat tambahan dalam membidangi seni teater.

Di sisi lain, kemampuan memahami untuk berperan di luar dirinya adalah penguasaan khusus yang harus dikuasai secara teknis dalam berkarya teater. Kemampuan memahami dan membuat sarana dan prasarana perlengkapan berbasis multimedia adalah pendekatan aktual yang harus dikuasai seorang dramawan dalam kaitannya dengan penyajian teater berbasis teknologi. Seni teater juga sebagai bagian integral kesenian memiliki media ungkap suara dalam wujud pemeranan. Cara atau teknik ini lebih mengutamakan terciptanya casting, pembawaan, diksi, intonasi, pengaturan laring dan faring secara konsisten adalah bagian penting dari penjelmaan profesi yang harus dimiliki.

5. Seni Sastra

Sastra merupakan kata serapan dari bahasa Sanskerta *śāstra*, yang berarti "teks yang mengandung instruksi" atau "pedoman", dari kata dasar *śās*- yang berarti "instruksi" atau "ajaran". Dalam bahasa Indonesia kata ini biasa digunakan untuk merujuk kepada "kesusastraan" atau sebuah jenis tulisan yang memiliki arti atau keindahan tertentu.

Yang agak bias adalah pemakaian istilah sastra dan sastrawi. Segmentasi sastra lebih mengacu sesuai defenisinya sebagai sekedar teks. Sedang sastrawi lebih mengarah pada sastra yang kental nuansa puitis atau abstraknya. Istilah sastrawan adalah salah satu contohnya, diartikan sebagai orang yang menggeluti sastrawi, bukan sastra.

Selain itu dalam arti kesusastraan, sastra bisa dibagi menjadi sastra tertulis atau sastra lisan (sastra oral). Di sini sastra tidak banyak berhubungan dengan tulisan, tetapi dengan bahasa yang dijadikan wahana untuk mengekspresikan pengalaman atau pemikiran tertentu.

Biasanya kesusastraan dibagi menurut daerah geografis atau bahasa. Yang termasuk dalam kategori Sastra adalah : novel, cerita/cerpen (tertulis/lisan), syair, pantun, lukisan/kaligrafi.



Gambar 2.4 Seni Sastra

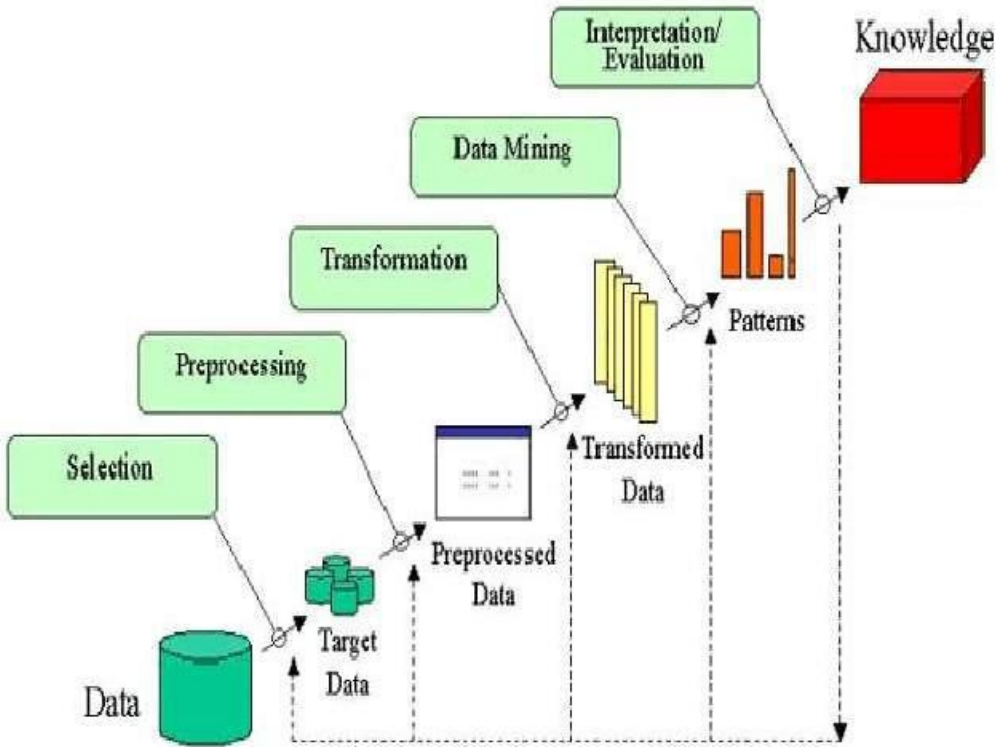
2.2 Knowledge Discovery in Databases (KDD)

Knowledge Discovery in Databases secara sederhana dapat dikatakan sebagai proses meng-*extract* atau menggali *(mining)* pengetahuan/informasi yang berharga *(Interesting knowledge)* dari sejumlah besar data baik yang disimpan di dalam *database, datawarehouse* maupun media penyimpanan informasi lainnya (Han, Kamber, 2001). Dari definisi tersebut didapatkan berbagai istilah yang beredar dan memliki arti hampir sama dengan *data mining* yaitu *knowledge mining from database, knowledge extraction, data/pattern analysis, data archeology, data dredging, information harvesting* dan *business intelligence.*

2.2.1 *Data Mining*

Istilah *data mining* dan *knowledge discovery in databases* (KDD) seringkali digunakan secara bergantian unuk menjelaskan proses penggalian informasi tersembunyi dalam suatu basis data yang besar. Sebenarnya kedua istilah tersebut memiliki konsep yang berbeda, tetapi berkaitan satu sama lain. Dan salah satu tahapan dalam keseluruhan proses KDD adalah *data mining.*

Proses KDD secara garis besar dapat terlihat pada gambar 2.5.dijelaskan sebagai berikut (Fayyad, 1996).



Gambar 2.5 Tahap-tahap dalam proses *knowladge discovery*

1. *Selection*, tahap ini bertujuan untuk mendapatkan data teks yang memiliki relavansi dengan tugas analisa pada tahap selanjutnya. Pada tahap *selection* ini akan dilakukan pemilihan dan *pemfilteran* data mentah (*raw data*) menjadi target data. Beberapa cara seleksi, antra lain:
   a. *Sampling,* adalah seleksi subset representatif dari populasi data yang besar.
   b. *Denoising,* adalah proses menghilangkan noise dari data yang akan ditransformasikan.

c. *Feature extraction*, adalah proses membuka spesifikasi data yang signifikan dalam konteks tertentu.

2. *Preprocessing*, proses ini mencakup antara lain membuang duplikasi data, memeriksa data yang inkonsisten, dan memperbaiki kesalahan pada data seperti kesalahan cetak (*tifografi*). Juga dilakukan proses *enrichment*, yaitu proses "memperkaya" data yang sudah ada denan data atau informasi lain yang relevan dan diperlukan untuk KDD, seperti data atau informasi eksternal.
3. *Transformation*, *coding* adalah proses transformasi pada data yang telah dipilih, sehingga data tersebut sesuai untuk proses *data mining*. Proses *coding* dalam KDD merupakan proses kreatif dan sangat bergantung jenis atau pola informasi yang akan dicari dalam basis data.
4. *Data Mining*, proses mencari pola atau informasi menarik dalam data terpilih dengan menggunakan teknik atau metode tertentu. Teknik, metode, atau algoritma dalam data mining sangat bervariasi. Pemilihan metode atau algoritma yang tepat sangat bergantung pada tujuan dan proses KDD secara keseluruhan.
5. *Interpretation*/ *Evaluation*, dari hasil proses *data mining* di atas akan dilakukan proses *evaluation* untuk mendapatkan *interesting knowledge* yang lebih dapat dimengerti dan berguna bagi pihak yang berkepentingan. Tahap ini mencakup pemeriksaan apakah pola atau informasi yang ditemukan bertentangan dengan fakta atau hipotesis yang ada sebelumnya.

6. *Knowledge presentation***,** penyajian pengetahuan yang digali kepada pengguna dengan menggunakan visualisasi dan teknik representasi pengetahuan.

2.2.2 *Text Mining*

Text mining merupakan salah satu aplikasi dari *data mining*. *Text mining* juga sering disebut sebagai *Text Data Mining* (TDM) dan *knowladge Discovery in Textual Databases* (KDT). *Text mining* merupakan proses mengesktrak *petterns* dan *knowladge* yang bersifat menarik dan nontrivial (penting) dari dokumen-dokumen teks. Pada intinya proses kerja *text mining* sama dengan proses kerja *data mining* pada umumnya hanya saja data yang di *mining* merupakan *text databases*.

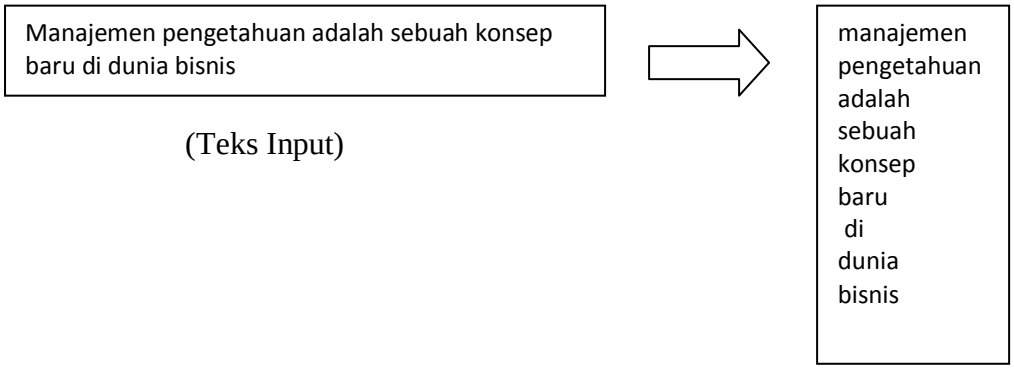
Data teks akan diproses menjadi data numerik agar dapat dilakukan proses lebih lanjut. Sehingga dalam *text mining* ada istilah *preprocessing data*, yaitu proses pendahulu yang diterapkan terhadap data teks yang bertujuan untuk menghasilkan data numerik.

Pada proses *preprosesing* merupakan tahap dimana deskripsi di tangani untuk dapat siap diproses memasuki tahap *text mining*. Tahap-tahap tersebut adalah:

1. *Parsing/ Tokenizing*
2. *Stopwords Removal/ Filtering*
3. *Stemming*
4. *Tagging*
5. *Anayizing*

2.2.2.1 *Parsing/ Tokenizing*

Parsing adalah sebuah proses yang dilakukan seseorang untuk menjadikan sebuah kalimat menjadi lebih bermakna atau berada dengan cara memecah kalimat tersebut menjadi kata-kata atau *frase-frase ("Parsing")*. *Parsing* di dalam pembuatan aplikasi *text mining* ini merupakan proses penguraian deskripsi yang semula berupa kalimat-kalimat berisi kata-kata dan tanda pemisah antara kata seperti titik(.), koma(,), spasi dan tanda pemisah lain menjadi kata-kata saja baik itu berupa kata-kata penting maupun kata-kata tak penting. Secara sederhana proses *parsing* ini terlihat sebagai proses pengambilan kata jika ketemu tanda spasi namun pada kenyataannya tidak sesederhana itu. Contoh tahap ini adalah sebagai berikut :



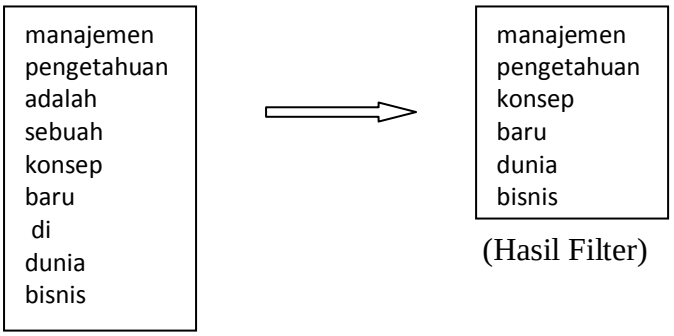
Gambar 2.6 Tahap *parsing*

2.2.2.2 *Stopwords Removal/ Filtering*

Kebanyakan bahasa resmi di berbagai negara memiliki kata fungsi dan kata sambung seperti artikel dan preposisi yang hampir selalu muncul pada dokumen-dokumen teks. Biasanya kata-kata ini memiliki arti yang lebih di dalam

memenuhi kebutuhan seorang *searcher* di dalam mencari informasi. Kata-kata tersebut (misalnya *a, an, the on* pada bahasa inggris) disebut sebagai *stopwords*. Di dalam bahasa Indonesia *stopwords* dapat disebut sebagai kata tidak penting misalnya "di", "oleh", "pada", "sebuah", "karena". Sebelum proses *stopwords removal* dilakukan, terlebih dulu dibuat daftar stopwords (stoplist). Preposisi, kata hubung dan partikel biasanya merupakan kandidat *stoplist*.

Stopwords removal merupakan proses penghilangan kata tidak penting pada deskripsi melalui pengecekan kata-kata hasil *parsing* deskripsi apakah termasuk di dalam daftar kata tidak penting *(stoplist)* atau tidak. Jika termasuk di dalam *stoplist* maka kata-kata tersebut akan di-*remove* dari deskripsi sehingga kata-kata yang tersisa di dalam deskripsi di anggap sebagai kata-kata penting atau *keywords*.



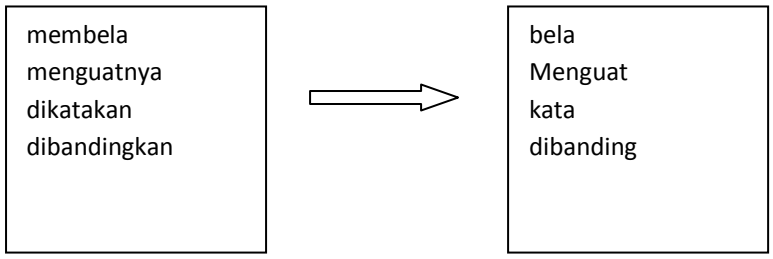
Gambar 2.7 Tahap *Stopwrds removal/ filtering*

2.2.2.3 *Stemming*

Stemming adalah proses pemetaan dan penguraian berbagai bentuk *(variants)* dari suatu kata menjadi bentuk kata dasarnya *(stem)*. Proses ini juga disebut sebagai *conflation*. Proses *stemming* secara luas sudah digunakan di dalam

Information retrieval (pencarian informasi) untuk meningkatkan kualitas informasi yang didapatkan. Kualitas informasi yang dimaksud misalnya untuk mendapatkan hubungan antara variant kata yang satu dengan yang lainnya. Sebagai contoh kata "diculik", "menculik" (melakukan tindakan menculik) dan "penculik" (orang yang menculik) yang semula mengandung arti yang berbeda dapat di-*stem* menjadi sebuah kata "culik" yang memiliki arti yang sama sehingga kata-kata diatas saling berhubungan.

Selain itu *stemming* juga dapat digunakan untuk mengurangi ukuran dari suatu ukuran *index file*. Misalnya dalam suatu deskripsi terdapat *variant* kata "memberikan", "diberikan", "memberi" dan "diberi" hanya memiliki akar kata (*stem*) yaitu "beri". Ukuran file daftar *index* yang semula berjumlah lima *record* akan di-*reduce* sehingga menjadi satu *record* saja.

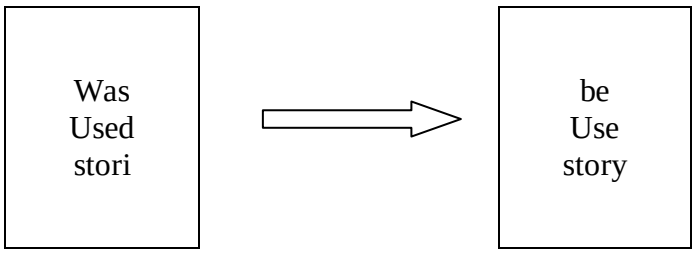


(Hasil Filtering) (Hasil Stemming)

Gambar 2.8 Tahap *Stemming*

2.2.2.4 *Tagging*

Tahap *tagging* adalah tahap mencari bentuk awal/ root dari tiap kata lampau atau kata hasil *stemming* yang bukan Bahasa Indonesia. Berikut contoh *tagging* dalam Bahasa Inggris.



(Hasil *Stemming*) (Hasil *Tagging*)

Gambar 2.9 Tahap *Tagging*

2.2.2.5 *Analyzing*

Tahap *analyzing* merupakan tahap penentuan seberapa jauh keterhubungan antara kata-kata dengan dokumen yang ada.

2.2.3 Metode *Mining*

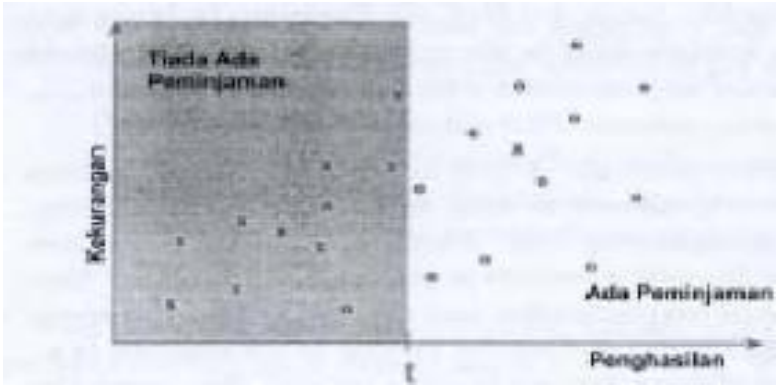
Banyak teknik dan metode yang ada untuk melakukan berbagai jenis tugas *mining*. Metode ini dikelompokkan dalam 3 paradigma utama *mining*: *Predictive Modeling*, *Discovery*, dan *Deviation Detection*.

1. *Predictive Modeling*, *a*plikasi *Predictive Modeling* menghasilkan klasifikasi atau prediksi. Tujuan dari *predictive modeling* adalah menemukan pola yang melibatkan variabel untuk memprediksi dan mengklasifikasi perilaku masa depan dari sebuah entitas. Ada dua tipe masalah yang diselesaikan oleh *predictive modeling*: klasifikasi dan regresi.

2. *Discovery*, *a*plikasi *discovery* adalah pendekatan eksploratoris untuk analisis data. Aplikasi *discovery* menggunakan teknik yang menganalisis data set yang besar untuk menemukan *association rules*(atau pola), atau menemukan kluster dari sampel yang dapat dikelompokan. Hasil dari metode *discovery* umumnya dimaksudkan untuk pengguna. Walau begitu, hasilnya juga dapat diaplikasikan ke metode *mining* yang lain.
3. *Deviation Detection,* melakukan deteksi anomali secara otomatis. Tujuannya untuk mengidentifikasi kebiasaan suatu entitas dan menetapkan sejumlah *norm* melalui *pattern discovery*. Sampel yang berdeviasi dari *norm* lalu diidentifikasi sebagai tidak biasa. teknik *Deviation Detection* melalui visualisasi melalui *parallel coordinates*, *scatterplots*, dan *surface plots*.

2.2.3.1 Klasifikasi (*classification*)

Klasifikasi adalah proses pencarian sekumpulan model atau fungsi yang menggambarkan dan membedakan kelas data dengan tujuan agar model tersebut dapat digunakan untuk memprediksi kelas dari suatu obyek yang belum diketahui kelasnya.(Muhammad Rachli, 2007). Gambar 2.10 menunjukkan pembagian sederhana pada data peminjaman menjadi dua ruang kelas (punya dan tidak punya peminjaman). Pada gambar tersebut, *x* merepresentasikan peminjaman yang bermasalah, dan *o* peminjaman yang pengembaliannya lancar.



Gambar 2.10 Batas klasifikasi linier sederhana pada himpunan data peminjaman.

Klasifikasi memiliki dua proses yaitu membangun model klasifikasi dari sekumpulan kelas data yang sudah didefinisikan sebelumnya (*training data set*) dan menggunakan model tersebut untuk klasifikasi tes data serta mengukur akurasi dari model. Klasifikasi dapat dimanfaatkan dalam berbagai aplikasi seperti diagnosa medis, *selective marketing*, pengajuan kredit perbankan, dan *email*. Model klasifikasi dapat disajikan dalam berbagai macam model klasifikasi seperti *decision trees, bayesian classification, k-nearest-neighbourhood classifier, neural network* dan lain-lain.

A. Decision tree

Decision tree adalah cara merepresentasikan kumpulan aturan yang mengacu ke suatu nilai atau kelas. Misalnya mengklasifikasikan suatu proposal pinjaman uang memiliki resiko baik atau buruk. Gambar 2.11 memperlihatkan *decision tree* sederhana: *decision node, branches and leaves*.



Gambar 2.11 *Classification tree* sederhana

Komponen pertama adalah simpul *top decision*, atau simpul *root*, yang menentukan test yang akan dijalankan. Simpul *root* dalam contoh ini adalah "*income* > $40.000". Hasil dari tes ini menyebabkan *tree* terpecah menjadi dua cabang, dengan tiap cabang meepresentasikan satu dari jawaban yang mungkin. Dalam kasus ini, jawabannya adalah "ya" dan "tidak", sehingga mendapatkan dua cabang.

Bergantung pada algoritma yang digunakan. Tiap simpul bisa memiliki dua atau lebih cabang. Misalnya, CART akan menggenerate hanya dua cabang pada tiap simpul. *Tree* seperti ini disebut *binary tree*. Ketika lebih dari dua cabang diperbolehkan maka disebut sebagai *multiway tree*.

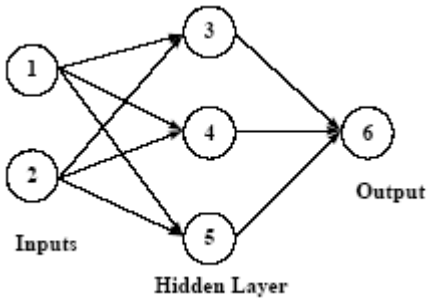
Tiap cabang akan memiliki simpul *node* yang lain atau dasar *tree*, yang disebut *leaf*. Dengan mengikuti *decision tree*, bisa memberikan nilai pada suatu kasus dengan memutuskan cabang mana yang akan diambil, dimulai dari simpul *root* dan bergerak ke bawah sampai *leaf*. Dengan menggunakan metode ini, seorang petugas yang bertanggung jawab untuk memutuskan pemberian pinjaman terhadap nasabah bisa menentukan apakah seorang nasabah memiliki resiko kredit yang baik atau buruk.

Model *decision tree* umum digunakan dalam data mining untuk menelaah data dan menginduksi *tree* dan aturan yang akan digunakan untuk membuat prediksi. Sejumlah algoritma yang berbeda bisa digunakan untuk membangun tree di antara nya adalah CHAID (Chi squared Automatic Interactin Detection), CART (Classification and Regression Trees), Quest dan C5.0.

Decision tree berkembang melalui pemecahan *iteratif* dari data ke dalam grup-grup diskrit, yang tujuannya adalah untuk memaksimalkan "jarak" antara grup pada tiap pemecahan. Contoh yang digunakan sampai saat ini sangatlah sederhana. *Tree* ini sangat mudah untuk dimengerti dan diinterpretasikan. Akan tetapi, *tree* bisa menjadi sangat kompleks. Bisa dibayangkan kompleksitas suatu *tree* yang diturunkan dari *database* dengan ratusan atribut dan variabel respon dengan lusinan kelas *input*. *Tree* sperti ini akan sangat sulit untuk dimengerti, meskipuntiap path dari *tree* biasanya dapat dimengerti. Dalam hal ini *decision tree* bisa menjelaskan prediksinya, yang merupakan keuntungan penting.

B. Neural Network

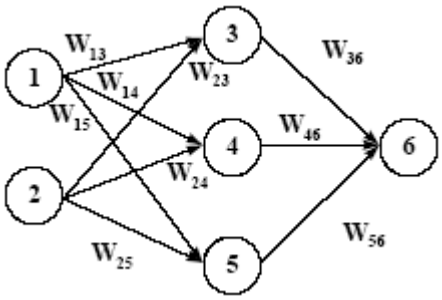
Neural Network biasa digunakan dalam masalah klasifikasi (di mana *output*nya adalah variabel kategoris) atau regresi (*output*nya kontinyu). *Neural network* dimulai dengan *layer input*, dimana tiap simpul berkorespondensi dengan variabel prediktor. Simpul-simpul *input* ini terhubung ke beberapa simpul dalam hidden *layer*. Tiap simpul *input* terhubung dengan tiap simpul dalam hidden *layer*. Simpul dalam hidden *layer* bisa jadi terhubung ke simpul lain dalam hidden *layer*, atau ke *output layer*. *Output layer* terdiri dari satu atau beberapa variabel respon.



Gambar 2.12 *Neural network* dengan satu *hidden layer*

Setelah *layer input*, tiap simpul mengambil satu himpunan *input*, mengalikan *input-input* tersebut dengan bobot W_{xy} (misalnya, bobot dari simpul 1 ke 3 adalah W_{13} – lihat gambar), menambahkan kedua bobot, menerapkan fungsi (biasa dipanggil fungsi *aktivasi* atau *squashing*), dan melewatkan *output*nya ke simpul dalam *layer* berikutnya. Misalnya, nilai yang dilewatkan dari node 4 ke node 6 adalah:

Activation function applied to ([W_{14} * value of node 1] + [W_{24} * value of node 2])



Gambar 2.13 Wxy adalah bobot dari simpul x ke simpul y

Tiap simpul bisa dilihat sebagai variabel prediktor (dalam hal ini simpul 1 dan 2) atau sebagai kombinasi dari variabel prediktor (simpul 3 sampai 6). Simpul 6 adalah kombiasi non linear dari nilai simpul 1 dan 2, karena fungsi aktivasi terhadap nilai penjumlahan di simpul-simpul tersembunyi.

C. Bayes Classification

a. Penggunaan Algoritma Bayes untuk Melakukan Klasifikasi

Sebelum mendeskripsikan bagaimana teorema *bayes* digunakan untuk klasifikasi, disusun masalah klasifikasi dari sudut pandang statistik. Jika *X* melambangkan set atribut data dan *Y* melambangkan kelas variable. Jika variable kelas memiliki hubungan non *deterministic* dengan atribut, maka dapat diperlakukan *X* dan *Y* sebagai variable acak dan menangkap peluang menggunakan P(*X* | *Y*). peluang bersyarat ini dikenal juga dengan *posterior* peluang untuk *Y,* dan sebaliknya prior P(*Y*). (Muhamad Rachli, 2007).

Selama masa *training* perlu mempelajari peluang *posterior* untuk seluruh kombinasi *X* dan *Y* berdasar informasi yang diperoleh dari *training* data. Dengan mengetahui peluang ini, *test record X'* dan dapat diklasifikasikan dengan menemukan *Y*' yang memaksimalkan peluang *posterior* P(*X* |*Y*).

Untuk mengestimasi peluang *posterior* secara akurat untuk setiap kombinasi label kelas dan nilai atribut adalah masa sulit karena membutuhkan *training set* sangat besar, meski untuk jumlah *moderate* atribut. Teorema *bayes* bermanfaat karena menyediakan pernyataan istilah peluang *posterior* dari *prior* P(*Y*) dari peluang bersyarat P(*X* | *Y*) dan bukti P(*X*):

$$P(Y \mid X) = \frac{P(X|Y)P(Y)}{P(X)}$$

Ketika membandingkan peluang *posterior* untuk nilai *Y* berbeda, istilah *dominator,* P(*X*), selalu tetap sehingga dapat diabaikan. Peluang *prior* P(*Y*) dapat dengan mudah diestimasi dari *training set* dengan menghitung pecahan *training record* yang dimiliki tiap kelas.

b. Penggunaan Naïve Bayes untuk Klasifikasi Teks

Salah satu metode klasifikasi yang dapat digunakan adalah metode Naive Bayes yang sering disebut sebagai *Naive Bayes Classifier* (*NBC*). NBC menggunakan teori probabilitas sebagai dasar teori. Ada dua tahap pada proses klasifikasi text. Tahap pertama adalah pelatihan terhadap himpunan artikel contoh (*training example*). Sedangkan tahap kedua adalah proses klasifikasi dokumen yang belum diketahui kategorinya.

Pada NBC setiap dokumen deskripsi dipresentasikan dengan pasangan atribut (a_1,a_2,a_3...a_n) dimana a_1 adalah kata pertama, a_2 dan seterusnya. (Wibisono, 2005). Sedangkan V adalah himpunan jenis dokumen. Pada saat klasifikasi, pendekatan Bayes akan menghasilkan label kategori yang paling tinggi probabiltasnya (V_{MAP}) dengan masukan atribut (a_1,a_2,a_3...a_n)

$$V_{MAP} = \underset{v_j \in V}{\arg\max}\ P(v_j \mid a_1, a_2, a_3 \ldots a_n) \qquad (2.1)$$

Teorema Bayes menyatakan:

$$P(A \mid B) = \frac{P(B|A)P(A)}{P(B)} \qquad (2.2)$$

Menggunakan Teorema Bayes ini, persamaan diatas akan dapat ditulis menjadi :

$$V_{MAP} = \underset{v_j \in V}{\arg\max}\ \frac{P\left(a_1, a_2, a_3 \ldots a_n \mid v_j\right) P\left(v_j\right)}{P\left(a_1, a_2, a_3 \ldots a_n\right)} \qquad (2.3)$$

$P(a_1,a_2,a_3...a_n)$ nilainya konstan untuk semua v_j sehingga persamaan ini dapat ditulis sebagai berikut:

$$V_{MAP} = \underset{v_j \in V}{\arg\max}\ P(v_j \mid a_1, a_2, a_3 \ldots a_n)\ P\left(v_j\right) \qquad (2.4)$$

Tingkat kesulitan menghitung $P(a_1,a_2,a_3...a_n|v_j)$ menjadi tinggi karena jumlah term $P(a_1,a_2,a_3...a_n|v_j)$ bisa jadi akan sangat besar. Ini disebabkan jumlah term tersebut sama dengan jumlah semua kombinasi posisi kata dikali dengan jumlah kategori yang ada.

Naive Bayes Classfier menyederhanakan hal ini dengan mengasumsikan bahwa di dalam setiap kategori, setiap kata independen satu sama lain. Dengan kata lain:

$$P(a_1, a_2,a_3 \ldots a_n \mid v_j) = \prod_i P(a_i \mid v_j) \qquad (2.5)$$

Dengan men-substitusikan persamaan ini dengan persamaan di atas akan menghasilkan:

$$V_{MAP} = \underset{v_j \in V}{\arg\max} P(v_j) \prod_i P(a_i \mid v_j) \qquad (2.6)$$

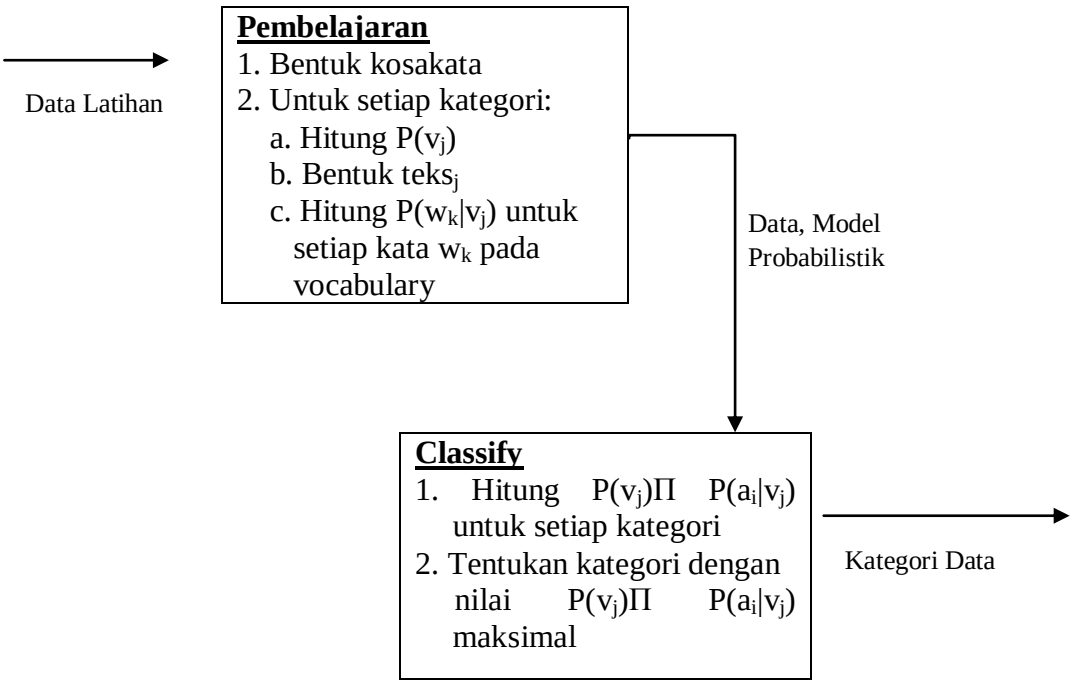
$P(v_j)$ dan probabilitas kata w_k untuk setiap kategori $P(w_k|v_j)$ dihitung pada saat pelatihan.

$$P(v_j) \leftarrow \frac{|docs\ j|}{|contoh|} \qquad (2.7)$$

$$P(w_k|v_j) = \frac{n_k + 1}{n^+ |kosakata|} \qquad (2.8)$$

Di mana |*docs j*| adalah jumlah data pada kategori j dan |*contoh*| adalah jumlah dokumen yang digunakan dalam pelatihan. Sedangkan n_k adalah jumlah kemnculan kata w_k pada kategori v_j, n adalah jumlah semua kata pada kategori v_j dan |kosakata| adalah jumlah kata yang unik (*distinc*) pada semua data latihan.

Ringkasan algoritma untuk *Naïve Bayes Clasifier* dapat dilihat pada gambar 2.14 berikut :



Gambar 2.14 Algoritma Naïve Bayes Classifier (Masayu, 2007)

Berikut adalah penjelasan gambar 2.14 Algoritma Naïve Bayes Classifier :

A. Proses pelatihan. Input adalah dokumen-dokumen contoh yang telah diketahui kategorinya:

1. Kosakata ← himpunan semua kata yang unik dari dokumen-dokumen contoh.
2. Untuk setiap kategori v_j lakukan:

   a. $Docs_j$ ← Himpunan dokumen-dokumen yang berada pada kategori v_j

   b. Hitung $P(v_j)$ dengan persamaan 2.7

c. Untuk setiap kata w_k pada kosakata lakukan:

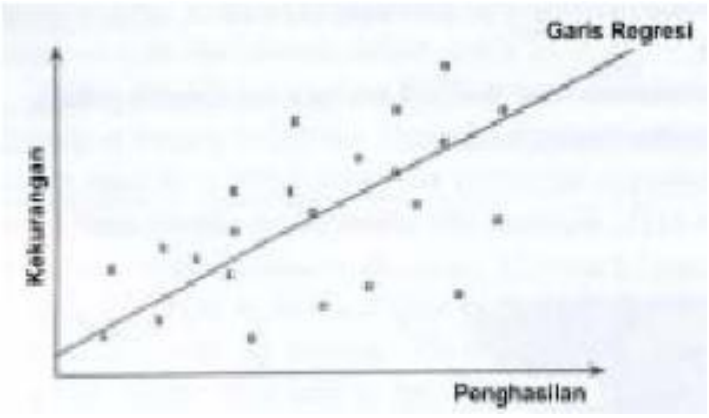
i. Hitung $P(w_k \mid v_j)$ dengan persamaan 2.8

B. Proses klasifikasi. Input adalah dokumen yang belum diketahui kategorinya:

1. Hasilkan V_{MAP} sesuai dengan persamaan 2.6 dengan menggunakan $P(v_j)$ dan $P(w_k \mid v_j)$ yang telah diperoleh dari pelatihan.

2.2.3.2 Regresi (*regression*)

Regresi adalah fungsi pembelajaran yang memetakan sebuah unsur data ke sebuah variabel prediksi bernilai nyata. Aplikasi dari regresi ini misalnya pada prediksi volume biomasa di hutan dengan didasari pada pengukuran gelombang mikro penginderaan jarak jauh (*remotely-sensed*), prediksi kebutuhan kustomer terhadap sebuah produk baru sebagai fungsi dari pembiayaan advertensi. Gambar 2.14 menunjukkan regresi linear sederhana dimana "total peminjaman" (*total debt*) diplot sebagai fungsi linier dari penghasilan (*income*): pengeplotan ini menghasilkan kesalahan besar karena hanya ada korelasi sedikit antara kedua variabel ini.

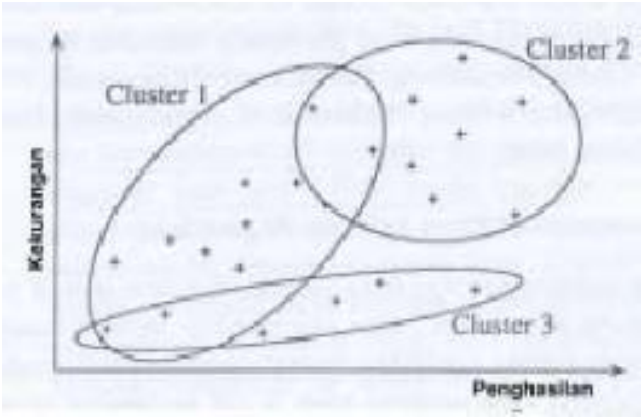


Gambar 2.15 Regresi linier sederhana untuk himpunan data peminjaman.

Regresi melibatkan model yang memetakan data contoh ke prediksi real-valued. Teknik regresi meliputi neural networks dan *decision* (atau *regression*) *trees*.

2.2.3.3 Pengelompokan (*clustering*)

Pengelompokan (*clustering*) merupakan tugas deskripsi yang banyak digunakan dalam mengidentifikasi sebuah himpunan terbatas pada kategori atau *cluster* untuk mendeskripsikan data yang ditelaah. Kategori-kategori ini dapat bersifat eksklusif dan ekshaustif mutual, atau mengandung representasi yang lebih kaya seperti kategori yang hirarkis atau saling menumpu (*overlapping*). Gambar 2.14 menunjukkan pembagian himpunan data peminjaman menjadi 3 *cluster*. Di sini, *cluster* - *cluster* dapat saling menumpu, sehingga titik-titik data dapat menjadi anggota lebih dari satu *cluster*. (Label *x* dan *o* pada gambar sebelumnya diubah menjadi + untuk mengindikasikan bahwa keanggotaan kelas diasumsikan belum diketahui.).



Gambar 2.16 Pengelompokan himpunan data peminjaman menjadi 3 *cluster*

2.2.3.4 *Association Rule*

Association rule mining adalah teknik *mining* untuk menemukan aturan assosiatif antara suatu kombinasi item. Contoh dari aturan assosiatif dari analisa pembelian di suatu pasar swalayan adalah bisa diketahui berapa besar kemungkinan seorang pelanggan membeli roti bersamaan dengan susu. Dengan pengetahuan tersebut. pemilik pasar swalayan dapat mengatur penempatan barangnya atau merancang kampanye pemasaran dengan memakai kupon diskon untuk kombinasi barang tertentu. Penting tidaknya suatu aturan assosiatif dapat diketahui dengan dua parameter, *support* yaitu persentase kombinasi item tersebut. dalam *database* dan *confidence* yaitu kuatnya hubungan antar item dalam aturan assosiatif.

Algoritma yang paling populer dikenal sebagai Apriori dengan paradigma *generate and test*, yaitu pembuatan kandidat kombinasi item yang mungkin berdasar aturan tertentu lalu diuji apakah kombinasi item tersebut memenuhi syarat *support* minimum. Kombinasi item yang memenuhi syarat tersebut. disebut *frequent itemset*, yang nantinya dipakai untuk membuat aturanaturan yang memenuhi syarat *confidence* minimum.

2.2.3.5 *Visualization*

Visualisasi adalah penggunaan grafik komputer untuk menciptakan gambaran visuil yang membantu pemahaman yang ruwet, penyajian presentasi data yang kokoh. Visualisasi Data mining adalah proses menemukan kandungan yang tersembunyi dapat menjadi pengetahuan yang bermanfaat khususnya dari data yang besar dengan menggunakan teknik visualisasi.

2.3 *Object Oriented Programing(OOP)*

Object-Oriented Programming (OOP) adalah sebuah pendekatan untuk pengembangan / *development* suatu *software* dimana dalam struktur *software* tersebut didasarkan kepada interaksi *object* dalam penyelesaian suatu proses/tugas. Interaksi tersebut mengambil form dari pesan-pesan dan mengirimkannya kembali antar *object* tersebut. *Object* akan merespon pesan tersebut menjadi sebuah tindakan / *action* atau metode. Jika mencoba melihat bagaimana tugas disekitar diselesaikan, maka akan diketahui bahwa manusia berinteraksi dalam sebuah *object-oriented world*. Jika akan bepergian manusia pasti berinteraksi dengan *object* mobil. Sebagai sebuah *object*, mobil berisi *object-object* lain yang berinteraksi untuk melakukan tugasnya membawa manusia.

Object-oriented programs terdiri dari *objects* yang berinteraksi satu sama lainnya untuk menyelesaikan sebuah tugas. Seperti dunia nyata, *users* dari *software* programs dilibatkan dari logika proses untuk menyelesaikan tugas. Contoh, ketika mencetak sebuah halaman di *word processor*, berarti melakukan inisialisasi tindakan dengan mengklik tombol printer. Kemudian hanya menunggu respon apakah *job* tersebut sukses atau gagal, sedangkan proses terjadi internal tanpa diketahui. Tentunya setelah menekan tombol printer, maka secara simultan *object* tombol tersebut berinteraksi dengan *object* printer untuk menyelesaikan *job* tersebut. Adapun karakteristik yang dimiliki OOP yaitu :

a. *Objects*

   Object adalah sebuah *structure* yang menggabungkan data dan prosedur untuk bekerja bersama-sama.

b. Abstraction

Ketika manusia berinteraksi dengan *object-object* di dunia ini, manusia sering hanya konsentrasi dengan sebuah bagian dari propertiesnya. Tanpa kemampuan untuk mensarikan/*abstract* atau menyaring untuk dibuang properties *object* yang asing / tidak ada hubungannya, maka akan menemukan kesulitan untuk memproses informasi yang kebanyakan membombarding.

Ketika membangun *objects* dalam aplikasi OOP, adalah penting untuk menggabungkan konsep *abstraction* ini. Jika membangun aplikasi *shipping*, maka harus membangun *object* produk dengan atribut seperti ukuran dan berat. Warna adalah contoh informasi yang tidak ada hubungannya dan harus dibuang. Tetapi ketika akan membangun *order-entry application*, warna menjadi penting dan harus termasuk atribut *object* produk.

c. Encapsulation

Ciri penting lainnya dari OOP adalah *encapsulation*. *Encapsulation* adalah sebuah proses dimana tidak ada akses langsung ke data yang diberikan, bahkan *hidden*. Jika ingin mendapat data, harus berinteraksi dengan *object* yang bertanggung jawab atas data tersebut.

d. Polymorphism

Polymorphisms adalah kemampuan 2 buah *object* yang berbeda untuk merespon pesan permintaan yang sama dalam suatu cara yang unik. Contohnya, dapat mengirim pesan *print* ke *object* printer yang akan

mencetak pada printer, dan juga dapat mengirim pesan yang sama ke *object screen* yang akan menuliskan pada *screen* monitor. Dalam OOP, diterapkan tipe *polymorphism* melalui proses yang disebut *overloading*. Dapat dilakukan dalam implementasi metode yang berbeda pada sebuah *object* yang mempunyai nama yang sama.

e. Inheritance

Penggunaan *inheritance* dalam OOP untuk mengklasifikasikan *objects* dalam program sesuai karakteristik umum dan fungsinya. Hal ini akan membuat pekerjaan bersama *object* lebih mudah dan lebih intuitif. Hal ini juga membuat programming lebih mudah karena memungkinkan untuk mengkombinasikan karakteristik umum kedalam *object parent* dan mewariskan karakteristik ini ke *child object*.

2.4 *Unified Modeling Language* (UML)

UML *(Unified Modeling Language)* adalah sebuah bahasa untuk menetukan, visualisasi, kontruksi, dan mendokumentasikan *artifact* (bagian dari informasi yang digunakan atau dihasilkan dalam suatu proses pembuatan perangkat lunak. Artifact dapat berupa model, deskripsi atau perangkat lunak) dari sistem perangkat lunak, seperti pada pemodelan bisnis dan sistem non perangkat lunak lainnya.

UML merupakan suatu kumpulan teknik terbaik yang telah terbukti sukses dalam memodelkan sistem yang besar dan kompleks. UML tidak hanya

digunakan dalam proses pemodelan perangkat lunak, namun hampir dalam semua bidang yang membutuhkan pemodelan.

Bagian-bagian utama dari UML adalah view, diagram, model element, dan *general mechanism*.

1. *View*

View digunakan untuk melihat sistem yang dimodelkan dari beberapa aspek yang berbeda. *View* bukan melihat grafik, tapi merupakan suatu abstraksi yang berisi sejumlah diagram. Beberapa jenis *view* dalam UML antara lain: *use case view, logical view, component view, concurrency view, dan deployment view.*

a) Use case view

Mendeskripsikan fungsionalitas sistem yang seharusnya dilakukan sesuai yang diinginkan *external actors*. *Actor* yang berinteraksi dengan sistem dapat berupa user atau sistem lainnya. *View* ini digambarkan dalam *use case diagrams* dan kadang-kadang dengan *activity diagrams*. *View* ini digunakan terutama untuk pelanggan, perancang (*designer*), pengembang (*developer*), dan penguji sistem (*tester*).

b) Logical view

Mendeskripsikan bagaimana fungsionalitas dari sistem, struktur statis (*class, object,dan relationship*) dan kolaborasi dinamis yang terjadi ketika *object* mengirim pesan ke *object* lain dalam suatu fungsi tertentu. *View* ini digambarkan dalam class diagrams untuk struktur statis dan dalam *state, sequence, collaboration, dan activity diagram* untuk model

dinamisnya. View ini digunakan untuk perancang (*designer*) dan pengembang (*developer*).

c) *Component view*

Mendeskripsikan implementasi dan ketergantungan modul. Komponen yang merupakan tipe lainnya dari code module diperlihatkan dengan struktur dan ketergantungannya juga alokasi sumber daya komponen dan informasi administrative lainnya. View ini digambarkan dalam *component view* dan digunakan untuk pengembang (*developer*).

d) *Concurrency view*

Membagi sistem ke dalam proses dan prosesor. *View* ini digambarkan dalam diagram dinamis *(state, sequence, collaboration, dan activity diagrams)* dan diagram implementasi *(component dan deployment diagrams)* serta digunakan untuk pengembang (*developer*), pengintegrasi (*integrator*), dan penguji (*tester*).

e) *Deployment view*

Mendeskripsikan fisik dari sistem seperti komputer dan perangkat (*nodes*) dan bagaimana hubungannya dengan lainnya. *View* ini digambarkan dalam *deployment diagrams* dan digunakan untuk pengembang (*developer*), pengintegrasi (*integrator*), dan penguji (*tester*).

2. Diagram

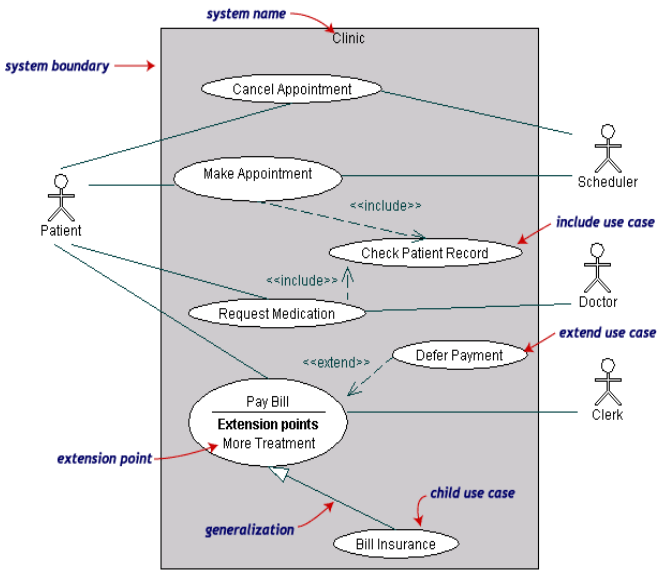
Diagram berbentuk grafik yang menunjukkan simbol elemen model yang disusun untuk mengilustrasikan bagian atau aspek tertentu dari sistem. Sebuah

diagram merupakan bagian dari suatu *view* tertentu dan ketika digambarkan biasanya dialokasikan untuk *view* tertentu. Adapun jenis diagram antara lain :

a) *Use Case Diagram*

Use case adalah abstraksi dari interaksi antara sistem dan actor. *Use case* bekerja dengan cara mendeskripsikan tipe interaksi antara *user* sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. *Use case* merupakan konstruksi untuk mendeskripsikan bagaimana sistem akan terlihat di mata *user*. Sedangkan *use case* diagram memfasilitasi komunikasi diantara analis dan pengguna serta antara analis dan *client*.

Use case diagram dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem. Contoh *use case diagram* dapat dilihat pada gambar 2.16.



Gambar 2.17 *Use Case Diagram*

b) *Class Diagram*

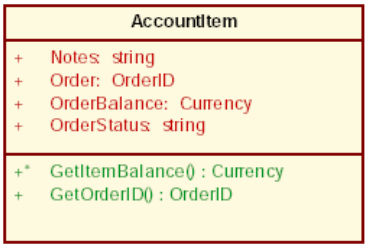
Class adalah dekripsi kelompok obyek-obyek dengan *property*, perilaku (operasi) dan relasi yang sama. Sehingga dengan adanya *class* diagram dapat memberikan pandangan global atas sebuah sistem. Hal tersebut tercermin dari class- class yang ada. Sebuah sistem biasanya mempunyai beberapa *class* diagram. *Class* diagram sangat membantu dalam visualisasi struktur kelas dari suatu sistem. *Class* memiliki tiga area pokok :

1. Nama (dan stereotype)
2. Atribut
3. Metoda

Atribut dan metoda dapat memiliki salah satu sifat berikut :

- *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan.
- *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya.
- *Public*, dapat dipanggil oleh siapa saja.

Class dapat merupakan implementasi dari sebuah *interface*, yaitu *class* abstrak yang hanya memiliki metoda. *Interface* tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class*. Contoh *class diagram* dapat dilihat pada gambar 2.17.



Gambar 2.18 *Class Diagram*

c) *Component Diagram*

Component software merupakan bagian fisik dari sebuah sistem, karena menetap di komputer tidak berada di benak para analis. Komponen merupakan implementasi *software* dari sebuah atau lebih *class*. Komponen dapat berupa *source code*, komponen biner, atau *executable component*. Sebuah komponen berisi informasi tentang *logic class* atau *class* yang diimplementasikan sehingga membuat pemetaan dari *logical view* ke *component view*. Sehingga component diagram merepresentasikan dunia nyata yaitu *component software* yang mengandung *component, interface dan relationship*.

d) *Deployment Diagram*

Menggambarkan tata letak sebuah sistem secara fisik, menampakkan bagian-bagian *software* yang berjalan pada bagian-bagian *hardware*, menunjukkan hubungan komputer dengan perangkat (*nodes*) satu sama lain dan jenis hubungannya. Di dalam *nodes*, *executeable component* dan *object* yang dialokasikan untuk memperlihatkan unit perangkat lunak yang dieksekusi oleh *node* tertentu dan ketergantungan komponen.

e) *State Diagram*

Menggambarkan semua *state* (kondisi) yang dimiliki oleh suatu *object* dari suatu *class* dan keadaan yang menyebabkan *state* berubah. Kejadian dapat berupa *object* lain yang mengirim pesan. *State class* tidak digambarkan untuk semua *class*, hanya yang mempunyai sejumlah *state* yang terdefinisi dengan baik dan kondisi *class* berubah oleh *state* yang berbeda.

f) *Sequence Diagram*

Sequence diagram digunakan untuk menggambarkan perilaku pada sebuah *scenario*. Kegunaannya untuk menunjukkan rangkaian pesan yang dikirim antara *object* juga interaksi antara *object*, sesuatu yang terjadi pada titik tertentu dalam eksekusi sistem.

f) *Collaboration Diagram*

Menggambarkan kolaborasi dinamis seperti *sequence* diagrams. Dalam menunjukkan pertukaran pesan, *collaboration* diagrams menggambarkan *object* dan hubungannya (mengacu ke konteks). Jika penekannya pada waktu atau urutan gunakan *sequence* diagrams, tapi jika penekanannya pada konteks gunakan *collaboration* diagram.

g) *Activity Diagram*

Menggambarkan rangkaian aliran dari aktivitas, digunakan untuk mendeskripsikan aktifitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktifitas lainnya seperti *use case* atau interaksi.

2.5 Pembangun Perangkat Lunak

2.5.1 Pegenalan PHP (*Personal Home Page*)

PHP (*Personal Home Page*) merupakan *script* untuk pemrograman *web server-side*, *script* yang membuat dokumen HTML secara *on the fly*, dokumen HTML yang dihasilkan dari suatu aplikasi bukan dokumen HTML yang dibuat dengan menggunakan teks *editor* atau *editor* HTML. Dengan menggunakan PHP, maintanance suatu *situs web* menjadi lebih mudah. Proses *update* data dapat dilakukan dengan menggunakan aplikasi yang dibuat dengan menggunakan *script* PHP. PHP/FI merupakan nama awal dari PHP, dibuat pertama kali oleh Rasmus Lerdoff. PHP awalnya merupakan program CGI yang dikhususkan untuk menerima *input* melalui *form* yang ditampilkan dalam *browser web*. *Software* ini disebarkan dan dilisensikan sebagai perangkat lunak *Open Source*. Adapun keunggulan yang dimiliki oleh PHP adalah:

1. *Life Cycle* yang sangat singkat, sehingga PHP selalu *up to date* mengikuti perkembangan teknologi internet.
2. *Cross Platform*, yakni PHP dapat dipakai di hampir semua *webserver* yang ada di pasaran (terutama Apache dan Microsoft IIS) dan dijalankan pada berbagai sistem operasi (Linux, Windows, FreeBSD).
3. PHP mendukung koneksi ke banyak *database* baik yang gratis maupun komersil, seperti *MySQL, mSQL, Oracle, Microsoft SQL Server*, *Interbase*, dan banyak lagi.
4. PHP bersifat *open source* dan gratis. Kemudahan dalam mendapatkan dokumentasi. di Internet, tidak akan sulit untuk mencari baik itu referensi,

kode-kode PHP yang sudah jadi dan juga mengajukan pertanyaan pada grup-grup diskusi yang di dalamnya banyak sekali para master PHP.

2.5.2 *MySQL*

MySQL merupakan sebuah *software* yang berguna sebagai suatu *database server* yang cukup terkenal. Kepopulerannya seiring dengan *user*an *script* PHP untuk *web programming*. *Database server* itu sendiri merupakan suatu *software* yang bertugas untuk melayani permintaan (*request*) *query* dari *client*.

MySQL sebagai suatu *database server* mempunyai beberapa kemampuan, salah satunya harus menyediakan suatu sistem manajemen *database* yang dapat mengatur bagaimana menyimpan, menambah, mengakses data dan transaksitransaksi *database* lainnya. *MySQL* cepat sekali berkembang, karena *MySQL* merupakan suatu *software* yang *Open Source*.

MySQL adalah *Relational Database Management System* (RDBMS) yang didistribusikan secara gratis dibawah lisensi GPL (*General Public License*). Dimana setiap orang bebas untuk menggunakan *MySQL*, namun tidak boleh dijadikan produk turunan yang bersifat komersial. *MySQL* sebenarnya merupakan turunan salah satu konsep utama dalam *database* sejak lama, yaitu SQL (*Structured Query Language*). SQL adalah sebuah konsep pengoperasian *database*, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis.

Keandalan suatu sistem *database* (DBMS) dapat diketahui dari cara kerja optimizer-nya dalam melakukan proses perintah-perintah SQL, yang dibuat oleh *user* maupun program-program aplikasinya. Sebagai *database server*, *MySQL*

dapat dikatakan lebih unggul dibandingkan *database server* lainnya dalam *query* data. Hal ini terbukti untuk *query* yang dilakukan oleh *single user*, kecepatan *query MySQL* bisa sepuluh kali lebih cepat dari PostgreSQL dan lima kali lebih cepat dibandingkan Interbase.

MySQL memiliki beberapa keistimewaan, antara lain :

1. *Portabilitas*. *MySQL* dapat berjalan stabil pada berbagai sistem operasi seperti *Windows, Linux, FreeBSD, Mac Os X Server, Solaris, Amiga*, dan masih banyak lagi.
2. *Open Source*. *MySQL* didistribusikan secara *open source*, dibawah lisensi GPL sehingga dapat digunakan secara cuma-cuma.
3. *Multiuse*. *MySQL* dapat digunakan oleh beberapa *user* dalam waktu yang bersamaan tanpa mengalami masalah atau konflik.
4. *Performance tuning*. *MySQL* memiliki kecepatan yang menakjubkan dalam menangani *query* sederhana, dengan kata lain dapat memproses lebih banyak SQL per satuan waktu.
5. Jenis Kolom. *MySQL* memiliki tipe kolom yang sangat kompleks, seperti *signed/unsigned integer*, *float*, *double*, *char*, *text*, *date*, *timestamp*, dan lainlain.
6. Perintah dan Fungsi. *MySQL* memiliki operator dan fungsi secara penuh yang mendukung perintah *Select* dan *Where* dalam perintah (*query*).
7. Keamanan. *MySQL* memiliki beberapa lapisan sekuritas seperti *level subnetmask*, nama *host*, dan izin akses *user* dengan sistem perizinan yang mendetail serta sandi terenkripsi.

8. Skalabilitas dan Pembatasan. *MySQL* mampu menangani basis data dalam skala besar, dengan j*UML*ah rekaman (*records)* lebih dari 50 juta dan 60 ribu tabel serta 5 milyar baris. Selain itu batas indeks yang dapat ditampung mencapai 32 indeks pada tiap tabelnya.
9. Konektivitas. *MySQL* dapat melakukan koneksi dengan klien menggunakan protokol TCP/IP, *Unix soket* (UNIX), atau *Named Pipes* (NT).
10. Lokalisasi. *MySQL* dapat mendeteksi pesan kesalahan pada *client* dengan menggunakan lebih dari dua puluh bahasa. Meskipun demikian, bahasa Indonesia belum termasuk di dalamnya.
11. Antar Muka. *MySQL* memiliki *interface* (antar muka) terhadap berbagai aplikasi dan bahasa pemrograman dengan menggunakan fungsi API (*Application Programming Interface*).
12. Klien dan Peralatan. *MySQL* dilengkapi dengan berbagai peralatan (*tool*) yang dapat digunakan untuk administrasi basisdata, dan pada setiap peralatan yang ada disertakan petunjuk *online*.
13. Struktur tabel. *MySQL* memiliki struktur tabel yang lebih fleksibel dalam menangani *ALTER TABLE*, dibandingkan basisdata lainnya semacam PostgreSQL ataupun Oracle.

2.5.3 Javascript

Javascript merupakan *cross-platform* yang dikembangkan oleh Netscape dan pertama kali digunakan dalam *browser* Netscape. Javascript dibuat agar mudah diintegrasikan kedalam program dan aplikasi lain, misalnya *browser*.

Sebagian besar *browser* saat ini sudah mendukung javascript. Oleh karena itu, *script* dari javascript biasanya dimasukkan kedalam suatu HTML dan dieksekusi di*browser* lain.

2.5.4 HTML

HTML merupakan singkatan dari *hypertext markup language*, yang merupakan program penulisan informasi pada sebuah *hompage*. Penulisan HTML dapat dilakukan menggunakan alat bantu seperti *Notepad* yang terdapat pada *windows* atau *simple text machintosh*. Selain itu juga dapat digunakan editor TML seperti *Macromedia Dreamweaver* dan *Microsoft Frontpage* yang dapat memudahkan dalam menulis HTML dan memungkinkan dokumen HTML yang dibuat dapat diakses oleh berbagai jenis *Browser*. HTML berupa kode-kode tag yang memberikan instruksi pada *Web Browser* untuk memberikan tampilan sesuai yang diinginkan.

2.5.5 Adobe Dreamweaver

Adobe Dreamweaver (dulunya Macromedia Dreamweaver) adalah pengembangan web aplikasi awalnya dibuat oleh Macromedia , dan sekarang dikembangkan oleh Adobe Sistems , yang mengakuisisi Macromedia tahun 2005. Dreamweaver tersedia untuk Mac dan Windows sistem operasi . versi terbaru telah dimasukkan dukungan untuk web teknologi seperti CSS , JavaScript , dan berbagai server-side scripting bahasa dan kerangka termasuk ASP , ColdFusion , dan PHP .

Adapun fitur-fitur yang disediakan oleh Dreamweaver yaitu :

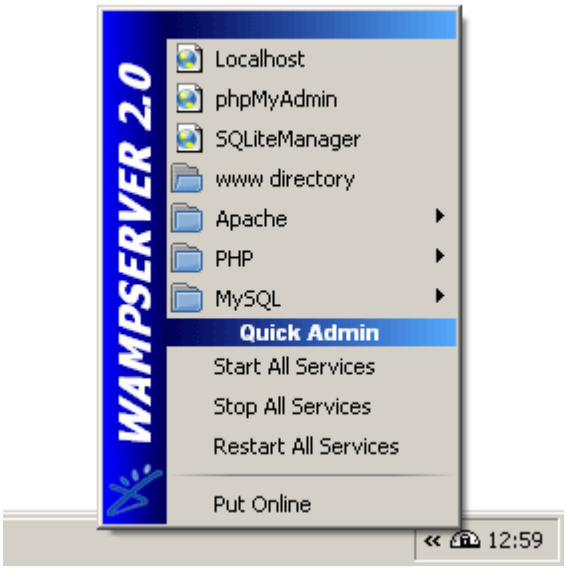
1. Dreamweaver yang WYSIWYG mode dapat menyembunyikan HTML code rincian halaman dari pengguna, sehingga memungkinkan untuk non-coders untuk membuat halaman web dan situs.
2. Dreamweaver memungkinkan pengguna untuk melihat situs di web browser secara lokal-instal. Ia juga memiliki alat manajemen situs seperti FTP / SFTP dan WebDAV mentransfer file dan fitur sinkronisasi, kemampuan untuk mencari dan mengganti baris teks atau kode oleh istilah pencarian dan ekspresi reguler di seluruh situs, dan fitur template yang memungkinkan satu sumber *update* kode bersama dan tata letak di seluruh situs tanpa server-side termasuk atau scripting. Panel perilaku juga memungkinkan penggunaan JavaScript dasar tanpa pengetahuan coding, dan integrasi dengan Adobe kerangka AJAX sigap menawarkan akses mudah ke konten dinamis yang dihasilkan dan antarmuka.
3. Dreamweaver dapat menggunakan pihak ketiga "*Extensions*" untuk memperluas fungsi inti dari aplikasi, yang setiap pengembang web bisa menulis (terutama dalam HTML dan JavaScript).

2.5.6 WAMP Server

WampServer merupakan lingkungan pengembangan web Windows. Hal ini memungkinkan untuk membuat aplikasi web dengan Apache, PHP dan *database* MySQL. Hal ini juga dilengkapi dengan PHPMyAdmin untuk dengan mudah mengelola *database* .

WampServer menginstal secara otomatis (*installer*), dan penggunaannya sangat intuitif. Anda akan dapat menyetel server tanpa menyentuh pengaturan file.

WampServer merupakan satu-satunya solusi paket yang akan memungkinkan untuk mereproduksi server produksi. WampServer juga memiliki *trayicon* untuk mengelola server dan pengaturannya.



Gambar 2.19 WAMP Server

2.5.7 *Web Server*

Web server menurut Minoli(1998,p33) berpendapat bahwa sebuah *web server* adalah seuatu program untuk menawarkan pelayanan yang bisa diperoleh seluruh jaringan. *Web server* merupakan suatu tipe *server* khusus yang dapat berkomunikasi langsung dengan *client* menggunakan HTTP, *web server*.

2.5.8 *Web Browser*

Menurut Ellsworth, *web browser* adalah *software* yang dijalankan pada komputer pemakai yang meminta informasi dari *web server* dengan menampilkannya sesuai dengan nama *file* itu sendiri.

2.5.9 *Client-Server*

Client-Server adalah arsitektur jaringan yang memisahkan *client* (biasanya aplikasi yang menggunakan GUI) dengan *server*. Masing-masing *clien*t dapat meminta data atau informasi dari *server*. Perbedaan karakteristik sistem *client* dan *server*:

1. *Service* (layanan)

   a. Hubungan antara proses yang berjalan pada mesin yang berbeda.

   b. Pemisahan fungsi berdasarkan ide layanannya.

   c. *Server* sebagai *provider* dan *client* sebagai konsumen.

2. *Sharing Resources* (sumber daya)

   Server bisa melayani beberapa *client* pada waktu yang sama dan meregulasi akses bersama untuk *share* sumber daya dalam menjamin konsistensinya.

3. *Asymmetrical Protocol* (protokol yang tidak simetris)

   Many-to-one relationship antara *client* dan *server*. *Client* selalu menginisiasikan dialog melalui layanan permintaan dan *server* menunggu secara pasif *request* dari *client*.

4. Pengaksesan yang mudah

   Proses yang dilakukan *server* boleh terletak pada mesin yang sama atau pada mesin yang berbeda melalui jaringan. Lokasi *server* harus mudah diakses dari *client*.

5. Pesan berbasiskan komunikasi

   Interaksi *server* dan *client* melalui pengiriman pesan yang menyertakan permintaan dan jawaban.

6. Pemisahan *interface* dan implementasi

   Server bisa di-*upgrade* tanpa mempengaruhi *client* selama *interface* pesan yang diterbitkan tidak berubah.