BAB II

LANDASAN TEORI

2.1 Sistem Pakar

2.1.1 Definisi Sistem Pakar

Sistem pakar (**Arhami: 2005**) adalah salah satu cabang dari *artificial intelegence* yang membuat penggunaan secara luas *knowledge* yang khusus untuk penyelesaian masalah tingkat manusia oleh seorang pakar dan dirancang untuk dapat menirukan keahlian seorang pakar dalam menjawab pertanyaan dan menyelesaikan permasalahan di semua bidang. Seorang pakar adalah seorang yang mempunyai keahlian dalam bidang tertentu yaitu pakar yang mempunyai knowledge atau kemampuan khusus yang orang lain tidak mengetahui atau mampu dalam bidang yang dimilikinya. Sifat utama sistem pakar adalah ketergantungan sistem ini pada pengetahuan manusia dalam suatu bidang dalam menyusun strategi pemecahan persoalan yang dihadapi oleh sistem. Teknologi sistem pakar ini meliputi bahasa sistem pakar, program dan perangkat keras yang dirancang untuk membantu pengembangan dan pembuatan sistem pakar.

2.1.2 Ciri-Ciri Sistem Pakar

Adapun terdapat ciri-ciri sistem pakar adalah sebagai berikut :

1. Terbatas pada bidang spesifik.

2. Dapat memberikan penalaran untuk data-data yang tidak lengkap atau tidak pasti.
3. Dapat mengemukakan rangkaian alasan yang diberikannya dengan cara yang dapat dipahami.
4. Berdasarkan pada rule atay kaidah tertentu.
5. Dirancang untuk dikembangkan secara bertahap.
6. Outputnya bersifat nasihat atau anjuran.
7. Output tergantung dialog dengan user.
8. Knowledge base dan interface engine terpisah

2.1.3 Struktur Sistem Pakar

Sistem pakar terdiri dari 2 bagian pokok, yaitu: lingkungan pengembangan (*development environment*) dan lingkungan konsultasi (*consultation environment*). Lingkungan pengembangan digunakan sebagai pembangun sistem pakar baik dari segi pembangunan komponen maupun basis pengetahuan. Lingkungan konsultasi digunakan oleh seseorang yang bukan ahli untuk berkonsultasi (Kusumadewi, 2003).

Gambar 2. 1 Struktur Sistem Pakar

Komponen-komponen yang ada pada sistem pakar adalah Gambar2.1:

1. Fasilitas akuisisi pengetahuan. Melibatkan suatu interaksi antara pengembang sistem dan seseorang atau beberapa orang pakar dalam suatu bidang tertentu. Pengembang system menyerap prosedur-prosedur, strategi-strategi, dan pengalaman untuk menyelesaikan suatu masalah tertentu dari pakar tersebut dan membangunnya menjadi suatu program system pakar. Sumber pakar dapat nerupa buku dan atau ahli dalam bidang tertentu (Sampurno, 2000,h. C-13-C-14).
2. Basis pengetahuan dan basis aturan. Berisi pengetahuan-pengetahuan keahlian sebagai dasar pengambilan keputusan. Basis aturan mengandung fakta-fakta mengenai masalah-masalah yang akan dicari solusinya. Fakta-fakta yang diketahui disimpan sebagai awal. Fakta-fakta yang diperoleh dari proses inferensi ditambahkan pada database. Fakta-fakta ini berhubungan dengan semua yang diketahui selama proses inferensi..
3. Mesin inferensi (*inference engine*). Menurut Sampurno (2000, h. C-13), mesin inferensi adalah suatu perangkat lunak yang mengimplementasikan suatu operasi pelacakan dengan menggunakan basis pengetahuan dan basis data untuk mencapai solusi. Mesin inferensi menguji kaidah-kaidah dengan pola urutan tertentu untuk mencocokkan kondisi

sekarang dan kondisi awal yang diberikan oleh basis data. Jika kaidah-kaidah tersebut cocok dengan kondisi tersebut dapat diberikan pada basis data dan dapat digunakan untuk mencari fakta-fakta baru. Pada mesin inferensi deibedakan atas strategi kontrol dan strategi pelacakan. Strategi kontrol dibagi menjadi dua yaitu:

a. Pelacakan pertama melebar (*breadth first search*).

   Pelacakan pertama melebar merupakan strategi kontrol yang pelacakannya dilakukan selapis demi selapis, sehingga semua simpul pada tingkat yang sama akan dievaluasi terlebih dahulu sebelum pelacakan dilakukan terhadap tingkat yang lebih rendah.

b. Pelacakan pertama mendalam (*depth first search*).

   Pada pelacakan pertama mendalam, pelacakan dimulai dari simpul sampai pada tingkat yang paling rendah dan baru dilanjutkan pada simpul lain

4. Fasilitas penjelasan. Digunakan untuk melacak respon dan memberikan penjelasan tentang kelakuan sistem pakar secara interaktif melalui pertanyaan:

   a. Mengapa suatu pertanyaan ditanyakan oleh sistem pakar?

   b. Bagaimana konklusi dicapai?

   c. Mengapa ada alternatif yang dibatalkan?

d. Rencana apa yang digunakan untuk mendapatkan solusi?

5. Antarmuka. Digunakan untuk media komunikasi antara user dan program. Melalui antarmuka ini, pengguna memasukkan data awal, melakukan konsultasi, dan mendapatkan solusi permasalahan dari system pakar (Sampurno, 2000, h. C-13).
6. Fasilitas belajar mandiri. Sistem ini digunakan untuk mengevaluasi kinerja sistem pakar itu sendiri untuk melihat apakah pengetahuan-pengetahuan yang ada masih cocok untuk digunakan dimasa mendatang.

2.1.4 Keuntungan Sistem Pakar

Secara garis besar, banyak manfaat yang dapat diambil dengan adanya sistem pakar, antara lain (Kusumadewi, 2003) :

1. Memungkinkan orang awam bisa mengerjakan pekerjaan para ahli.
2. Bisa melakukan proses secara berulang secara otomatis.
3. Menyimpan pengetahuan dan keahlian para pakar.
4. Meningkatkan output dan produktivitas.
5. Menghemat waktu dalam pengambilan keputusan.
6. Mampu mengambil dan melestarikan keahlian para pakar (terutama yang termasuk keahlian langka).
7. Memiliki kemampuan untuk mengakses pengetahuan.
8. Meningkatkan kapabilitas sistem komputer.

2.1.5 Kelemahan Sistem Pakar

Disamping memiliki beberapa keuntungan, sistem pakar juga memiliki beberapa kelemahan, antara lain (Kusumadewi, 2003):

1. Biaya yang diperlukan untuk membuat dan pemeliharaannya sangat mahal.
2. Sulit dikembangkan. Hal ini tentu saja erat kaitannya dengan ketersediaan pakar dibidangnya.
3. Sistem pakar tidak 100% bernilai benar.

2.2 *Naïve Bayes*

2.2.1 Definisi *Naïve Bayes*

Merupakan pengklasifikasian dengan metode probabilitas dan statistik yang dikemukakan oleh ilmuwan Inggris Thomas Bayes, yaitu memprediksi peluang di masa depan berdasarkan pengalaman di masa sebelumnya sehingga dikenal sebagai *Teorema Bayes*. *Naive bayes* untuk setiap kelas keputusan, menghitung probabilitas dengan syarat bahwa kelas keputusan adalah benar, mengingat vektor informasi obyek. Algoritma ini mengasumsikan bahwa atribut obyek adalah independen. Probabilitas yang terlibat dalam memproduksi perkiraan akhir dihitung sebagai jumlah frekuensi dari "master" tabel keputusan. (**Olson dan Delen, 2008:102**).

2.2.2 Prinsip Kerja *Naïve Bayes*

Menurut Han dan Kamber (**2011:351**) Proses dari *The Naïve Bayesian classifier*, atau *Simple Bayesian Classifier*, sebagai berikut:

1. *Variable* D menjadi pelatihan *set tuple* dan label yang terkait dengan kelas. Seperti biasa, setiap *tuple* diwakili oleh vektor atribut n-dimensi, X = (x_1, x_2, ..., x_n), ini menggambarkan pengukuran n dibuat pada *tuple* dari atribut n, masing-masing, A_1, A_2, ..., An.
2. Misalkan ada kelas m, C_1, C_2, ..., C_m. Diberi sebuah *tuple*, X, *classifier* akan memprediksi X yang masuk kelompok memiliki probabilitas posterior tertinggi, kondisi-disebutkan pada X. Artinya, *classifier naive bayesian* memprediksi bahwa X *tuple* milik kelas C_i jika dan hanya jika :

$$P(C_i|\boldsymbol{X}) > P(C_j|\boldsymbol{X}) \quad \text{for } 1 \leq j \leq m, j \neq i. \qquad (2.1)$$

Jadi memaksimalkan P (C_i | X). Ci kelas yang P (C_i | X) dimaksimalkan disebut hipotesis posteriori maksimal.

Dengan teorema Bayes :

$$P(C_i|\boldsymbol{X}) = \frac{P(\boldsymbol{X}|C_i)P(C_i)}{P(\boldsymbol{X})}. \qquad (2.2)$$

Keterangan :

$P(C_i|X)$ = Probabilitas hipotesis C_i jika diberikan fakta atau *record* X (*Posterior probability*)

$P(X|C_i)$ = mencari nilai parameter yang memberi kemungkinan yang paling besar (*likelihood*)

$P(C_i)$ = Prior probability dari X (*Prior probability*)

P(X) = Jumlah probability *tuple* yg muncul

3. Ketika P (X) adalah konstan untuk semua kelas, hanya

   P (X | C_i) P (C_i) butuh dimaksimalkan. Jika probabilitas kelas sebelumnya tidak diketahui, maka umumnya diasumsikan ke dalam kelas yang sama, yaitu, P (C_1) = P (C_2) = · · · = P (C_m), maka dari itu akan memaksimalkan P (X | C_i). Jika tidak, maka akan memaksimalkan P (X | C_i) P (C_i). Perhatikan bahwa probabilitas sebelum kelas dapat diperkirakan oleh P (C_i) = | $C_{i,\ D}$ | / | D |, dimana | $C_{i,\ D}$ | adalah jumlah *tuple* pelatihan kelas C_i di D.

4. Mengingat *dataset* mempunyai banyak atribut, maka akan sangat sulit dalam mengkomputasi untuk menghitung $P(X|C_i)$. Agar dapat mengurangi perhitungan dalam mengevaluasi $P(X|C_i)$, asumsi *naïve* independensi kelas bersyarat dibuat. Dianggap bahwa nilai-nilai dari atribut adalah kondisional independen satu sama lain, diberikan

kelas label dari *tuple* (yaitu bahwa tidak ada hubungan ketergantungan diantara atribut) dengan demikian :

$$P(\boldsymbol{X}|C_i) = \prod_{k=1}^{n} P(x_k|C_i) = P(x_1|C_i) \times P(x_2|C_i) \times \cdots \times P(x_n|C_i). \quad (2.3)$$

Maka dapat dengan mudah memperkirakan probabilitas P (x_1 | C_i), P (x_2 | C_i),. . . , P (x_n | C_i) dari pelatihan *tuple*. Ingat bahwa di sini x_k mengacu pada nilai atribut A_k untuk *tuple* X. Untuk setiap atribut, dilihat dari apakah atribut tersebut kategorikal atau *continuous-valued* . Misalnya, untuk menghitung P (X | C_i) mempertimbangkan hal-hal berikut:

a. Jika A_k adalah kategorikal, maka P (X_k | C_i) adalah jumlah *tuple* kelas C_i di D memiliki nilai X_k untuk atribut A_k, dibagi dengan | C_i, D |, jumlah *tuple* kelas C_i di D.
b. Jika A_k *continuous-valued* , maka perlu melakukan sedikit lebih banyak pekerjaan, tapi perhitunganya cukup sederhana. Sebuah atribut *continuous-valued* biasanya diasumsikan memiliki distribusi *Gaussian* dengan rata-rata μ dan standar deviasi σ, didefinisikan oleh

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \quad (2.4)$$

sehingga :

$$P(x_k|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i}). \quad (2.5)$$

Setelah itu hitung μC_i dan σC_i, yang merupakan deviasi *mean* (rata-rata) dan standar masing-masing nilai atribut A_k untuk *tuple* pelatihan kelas C_i. Setelah itu gunakan kedua kuantitas dalam Persamaan, bersama-sama dengan x_k, untuk memperkirakan P (x_k | C_i).

5. Untuk memprediksi label kelas x, P(X|C_i)P(C_i) dievaluasi untuk setiap kelas C_i. *Classifier* memprediksi kelas label dari *tuple* x adalah kelas C_i, jika

$$P(\boldsymbol{X}|C_i)P(C_i) > P(\boldsymbol{X}|C_j)P(C_j) \quad \text{for } 1 \leq j \leq m, j \neq i. \quad (2.6)$$

Dengan kata lain, label kelas diprediksi adalah C_i yang mana P (X | C_i) P (C_i) adalah maksimal.

Contoh penelitian: *Wawan Singgih, P,* ***Sistem Untuk Deteksi Kerusakan Mesin Diesel Mobil Panther Dengan Metode Naïve Bayes****, STMIK Sinar Nusantara, 2014.*

2.3 PHP

Menurut Bunafit (**2004, p 139**) PHP adalah singkatan dari PHP *Hypertext Prepocessor*. PHP merupakan bahasa program yang berbentuk script yang diletakkan didalam *server* web. PHP telah diciptakan terutama

untuk kegunakkan web dan dapat menghubungkan query database serta menggunakan perintah-perintah sederhana/simple task yang dapat diluruskan dalam 3 atau 4 baris kode saja. PHP adalah bahasa pemograman yang baru dibangun sekitar tahun 1994/1995. PHP dapat menggatikan static website yang menggunakan HTML ke dinamic web pages yang berfungsi secara otomatis seperti ASP,CGI dan sebagainya.

Kelebihan menggunakan PHP adalah sebagai berikut:

a. PHP merupakan program *open source* (atau tidak memerlukan biaya lisensi).
b. Multi-Platform (dapat dijalankan pada sistem operasi yang berbeda-beda).
c. Adanya penggunakan *session* / sesi.
d. Web Server yang mendukung PHP dapat ditemukan dimana - mana dari mulai *apache, IIS, Lighttpd, hingga Xitami* dengan konfigurasi yang relatif mudah.

Dalam sisi pemahamanan, PHP adalah bahasa scripting yang paling mudah karena memiliki referensi yang banyak.

2.4 MySQL

Nugroho (**2004, p29**) mengemukakan, MySQL (*My Structure Query Language*) adalah sebuah program pembuat database yang bersifat *open source*, artinya siapa saja dapat menggunakannya secara bebas. MySQL merupakan sebuah perangkat lunak sistem manajemen basis data SQL / DBMS *(Database Management Sistem)* yang multithread, multi-user dan sekitar 6 juta instalasi diseluruh indonesia. 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.

Keistimewaan MySQL yakni :

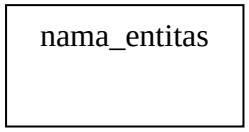
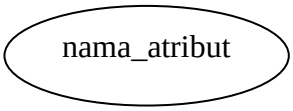
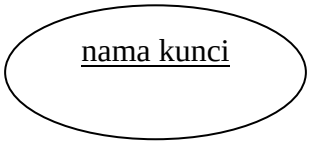
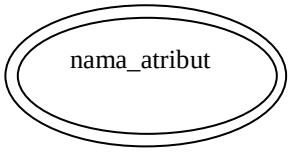
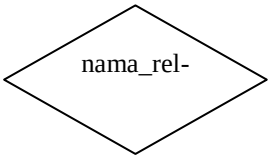
a. *Portabilitas*. MySQL dapat berjalan stabil pada berbagai sistem operasi seperti Windows, Linux, FreeBSD, Mac Os X Server, Solaris, Amiga, dan masih banyak lagi.
b. *Open Source*. MySQL didistribusikan secara *open source*, dibawah lisensi GPL sehingga dapat digunakan secara cuma-cuma.
c. *Multiuser*. MySQL dapat digunakan oleh beberapa user dalam waktu yang bersamaan tanpa mengalami masalah atau konflik.
d. *Performance tuning*. MySQL memiliki kecepatan yang menakjubkan dalam menangani *query* sederhana, dengan kata lain dapat memproses lebih banyak SQL per satuan waktu.
e. Jenis Kolom. MySQL memiliki tipe kolom yang sangat kompleks, seperti *signed / unsigned integer, float, double, char, text, date, timestamp*, dan lain-lain.
f. Perintah dan Fungsi. MySQL memiliki operator dan fungsi secara penuh yang mendukung perintah *Select* dan *Where* dalam perintah (*query*).

g. Keamanan. MySQL memiliki beberapa lapisan sekuritas seperti *level subnetmask*, nama *host*, dan izin akses *user* dengan sistem perizinan yang mendetail serta sandi terenkripsi

2.5 *Entity Relationship Diagram* (ERD)

Pemodelan awal basis data yang paling banyak digunakan adalah menggunakan *Entity Relationship Diagram* (ERD). ERD dikembangkan berdasarkan teori himpunan dalam bidang matematika. ERD digunakan untuk pemodelan basis data relasional. Sehingga jika penyimpanan basis data menggunakan OODBMS maka perancangan basis data tidak perlu menggunakan ERD (Rosa, 2011). Tabel 2.1 berikut memperlihatkan simbol-simbol yang digunakan pada ERD (Rosa, 2011).

Tabel 2.1 Simbol-simbol ERD

Simbol	**Deskripsi**
Entitas / entity 	Entitas merupakan data inti yang akan disimpan; bakal tabel pada basis data.
Atribut 	*Field* atau kolom data yang butuh disimpan dalam suatu entitas.
Atribut kunci primer 	*Field* atau kolom data yang butuh disimpan dalam suatu entitas dan digunakan sebagai kunci akses *record* yang diinginkan; biasanya berupa id.
Atribut multinilai/*multivalue* 	*Field* atau kolom data yang butuh disimpan dalam suatu entitas yang dapat memiliki nilai lebih dari satu.
Relasi 	Relasi yang menghubungkan antar entitas; biasanya diawali dengan kata kerja.

Asosiasi / *association* 1 0..* _____	Penghubung antara relasi dan entitas dimana di kedua ujungnya memiliki *multiplicity* kemungkinan jumlah pemakaian.

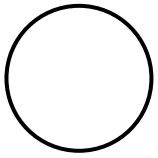
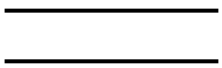
2.6 Diagram Alir Data (DAD)

Diagram Alir Data (DAD) atau *Data Flow Diagram* (DFD) adalah representasi grafik yang menggambarkan aliran informasi dan transformasi informasi yang diaplikasikan sebagai data yang mengalir dari masukan (*input*) dan keluaran (*output*) (Rosa, 2011).

DAD dapat digunakan untuk mempresentasikan sebuah sistem atau perangkat lunak pada beberapa level abstraksi. DAD dapat dibagi menjadi beberapa level yang lebih detail untuk mempresentasikan aliran informasi atau fungsi yang lebih detail. DAD menyediakan mekanisme untuk pemodelan fungsional ataupun pemodelan aliran informasi. Oleh karena itu, DAD lebih sesuai digunakan untuk memodelkan fungsi-fungsi perangkat lunak yang akan diimplementasikan menggunakan pemrograman terstruktur karena pemrograman terstruktur membagi-bagi bagiannya dengan fungsi-fungsi dan prosedur-prosedur.

Tabel 2.2 berikut memperlihatkan notasi-notasi DAD (*Edward Yourdon dan Tom DeMarco*) yang digunakan (Rosa, 2011).

Tabel 2.2 Notasi-notasi DAD

Notasi	**Keterangan**
	Proses atau fungsi/prosedur; pada pemodelan perangkat lunak yang akan diimplementasikan dengan pemrograman terstruktur, maka pemodelan notasi inilah yang harusnya menjadi fungsi atau prosedur didalam kode program. Catatan: nama yang diberikan pada sebuah proses biasanya berupa kata kerja.
	File atau basis data atau penyimpanan (*storage*); pada pemodelan perangkat lunak yang akan diimplementasikan dengan pemrograman terstruktur, maka pemodelan notasi inilah yang harusnya dibuat menjadi tabel-tabel basis data yang dibutuhkan, tabel-tabel ini juga harus sesuai dengan perancangan tabel-tabel pada basis data (*Entity Relationship Diagram*(ERD), *Conceptual Data Model* (CDM), *Physical Data Model* (PDM)). Catatan: nama yang diberikan pada sebuah penyimpanan biasanya kata benda.

	Entitas luar (*external entity*) atau masukan (input) atau keluaran (output) atau orang yang memakai/berinteraksi dengan perangkat lunak yang dimodelkan atau sistem lain yang terkait dengan aliran data dari sistem yang dimodelkan. Catatan: nama yang digunakan pada masukan (input) atau keluaran (output) biasanya berupa kata benda.
	Aliran data; merupakan data yang dikirim antar proses, dari penyimpanan ke proses, atau dari proses ke masukan (input) atau keluaran (output). Catatan: nama yang digunakan pada aliran data biasanya berupa kata benda, dapat diawali dengan kata data misalnya "data siswa" atau tanpa kata data misalnya "siswa".

Berikut ini adalah tahapan-tahapan perancangan dengan menggunakan DAD (Rosa, 2011):

1. Membuat DAD level 0 atau sering disebut juga *Context Diagram*. DAD level 0 menggambarkan sistem yang akan dibuat sebagai suatu entitas tunggal yang berinteraksi dengan orang maupun sistem lain. DAD level 0 digunakan untuk menggambarkan interaksi antara sistem yang akan dikembangkan dengan entitas luar.

2. Membuat DAD Level 1

   DAD level 1 digunakan untuk menggambarkan modul-modul yang ada dalam sistem yang akan dikembangkan. DAD level 1 merupakan hasil *breakdown* DAD level 0 yang sebelumnya sudah dibuat.

3. Membuat DAD Level 2

   Modul-modul pada DAD level 1 dapat di-*breakdown* menjadi DAD level 2. Modul mana saja yang harus di-*breakdown* lebih detail tergantung pada tingkat kedetailan modul tersebut. Apabila modul tersebut sudah cukup detail dan rinci maka modul tersebut sudah tidak perlu untuk di-*breakdown* lagi. Untuk sebuah sistem, jumlah DAD level 2 sama dengan jumlah modul pada DAD level 1 yang di-*breakdown*.

4. Membuat DAD Level 3 dan seterusnya

   DAD level 3, 4, 5, dan seterusnya merupakan *breakdown* dari modul pada DAD level di-atasnya. *Breakdown* pada level 3, 4, 5, dan seterusnya aturannya sama persis dengan DAD level 1 atau level 2.

Pada satu diagram DAD sebaiknya jumlah modul tidak boleh lebih dari 20 buah. Jika lebih dari 20 buah modul, diagram akan terlihat rumit dan susah untuk dibaca sehingga menyebabkan sistem yang akan dikembangkan juga menjadi rumit.