BAB II
TINJAUAN PUSTAKA

2.1 Teori Dasar

Kecerdasan buatan berasal dari bahasa Inggris "*Artificial Intelligence"* atau sering disingkat dengan AI yaitu *Intelligence* adalah kata sifat yang berarti cerdas, sedangkan *artificial* artinya buatan. Kecerdasan buatan yang dimaksud di sini merujuk pada mesin yang mampu berpikir, menimbang tindakan yang akan diambil, dan mampu mengambil keputusan seperti yang dilakukan oleh manusia.

Para ahli mendefinisikan AI secara berbeda-beda tergantung pada sudut pandang mereka masing-masing. Ada yang fokus pada logika berpikir manusia saja, tetapi ada juga yang mendefinisikan AI secara lebih luas pada tingkah laku manusia. Berikut beberapa pendapat para ahli mengenai kecerdasan buatan:

2.1.1 Kecerdasan buatan atau *Artificial Intelligence* (*AI*)

Menurut Hartati dan Iswanti (2008: 1) kecerdasan buatan adalah salah satu bidang ilmu komputer yang mendayagunakan komputer sehingga dapat berperilaku cerdas seperti manusia. Ilmu komputer tersebut mengembangkan perangkat lunak dan perangkat keras untuk menirukan tindakan manusia seperti penalaran, pembelajaran, pemecahan masalah, dan sebagainya. Cerdas berarti memiliki pengetahuan, pengalaman, dan penalaran untuk membuat keputusan dan mengambil tindakan. Untuk membuat sebuah mesin

menjadi cerdas (dapat bertindak seperti manusia) maka harus diberi bekal pengetahuan dan diberi kemampuan untuk menalar. Kecerdasan buatan memungkinkan komputer untuk berpikir atau menalar dan menirukan proses belajar manusia sehingga informasi baru dapat diserap sebagai pengetahuan, pengalaman, dan proses pembelajaran serta dapat digunakan sebagai acuan di masa-masa yang akan datang (Sutojo, Mulyanto, dan Suhartono, 2011: 3).

Menurut Jones (2008: 3) tahun 1950 merupakan saat-saat awal dari *AI* yaitu saat awal sistem komputer dibangun dan ide-ide pembangunan mesin cerdas mulai terbentuk. Pada tahun 1950, Alan Turing menyimpan pertanyaan dalam pikirannya "apakah sebuah mesin mampu untuk berpikir". Alan Turing melakukan percobaan yang cukup sederhana untuk menentukan apakah suatu mesin bisa dikatakan cerdas. Hasil percobaannya ini disebut dengan *Turing Test*. Dalam *Turing Test*, jika sebuah mesin mampu mengelabui seseorang yang menganggap mesin itu adalah manusia, maka mesin itu dianggap telah lulus dari tes kecerdasan (*intelligence test*).

Seseorang yang menjadi subjek percobaan diminta untuk menentukan terminal mana yang terkoneksi dengan komputer. Subjek boleh mengajukan pertanyaan, membuat pernyataan, menanyakan perasaan dan motivasi selama diperlukan. Jika subjek ternyata gagal menentukan terminal mana yang terkoneksi dengan komputer, maka komputer dinyatakan lulus tes dan dikatakan memiliki *consciousness* (kesadaran)(Al Fatta, 2009: 2).

Menurut Jones (2008: 5) pada tahun 1956, *Darmouth AI Conference* membawa para peneliti yang terlibat dalam penelitian *AI* seperti John McCarthy bersama peneliti-peneliti lainnya untuk sesi diskusi dan penelitian *AI* di *Dartmouth*

College. Sejak saat itu, banyak konferensi *AI* telah diselenggarakan di seluruh dunia, dan berbagai disiplin ilmu belajar di bawah nama *AI*. Pada awal 80-an penelitian tentang *AI* sukses di bidang komersial dari *software* jenis *Expert System*. Pada era 90-an dan awal abad 21, *AI* mencapai kesuksesan terbesarnya setelah diadopsi secara luas oleh industri teknologi, memberikan sumbangan besar pada logistik, data mining, diagnosis medis dan berbagai bidang lainnya (Al Fatta, 2009: 4).

Kombinasi antara *AI* dengan bidang ilmu yang lainnya melahirkan subdisiplin ilmu dalam *AI*. Beberapa diantaranya adalah logika *fuzzy* (*fuzzy logic*), jaringan syaraf tiruan (*artificial neural network*), dan sistem pakar (*expert system*) (Sutojo, dkk., 2011: 12-25).

2.1.1.1 Logika *Fuzzy* (*Fuzzy Logic*)

Logika *fuzzy* adalah metodologi sistem kontrol pemecahan masalah, yang sesuai untuk diimplementasikan pada sistem, mulai dari sistem yang sederhana, sistem kecil, *embedded system*, jaringan komputer, *multi-channel* atau *workstation* berbasis akuisisi data, dan sistem kontrol. Dalam logika *fuzzy* memungkinkan nilai keanggotaan berada di antara 0 dan 1, artinya suatu keadaan memungkinkan mempunyai dua nilai “Ya” dan “Tidak” secara bersamaan, namun besar nilainya tergantung pada bobot keanggotaan yang dimilikinya. Logika *fuzzy* dapat digunakan di berbagai bidang seperti pada sistem diagnosis penyakit (dalam bidang kedokteran); pemodelan sistem pemasaran, sistem operasi (dalam bidang

ekonomi); kendali kualitas air, prediksi adanya gempa bumi, klasifikasi dan pencocokan pola (dalam bidang teknik) (Sutojo, dkk., 2011: 211-212).

Ada beberapa keuntungan yang dapat diambil ketika menggunakan logika *fuzzy* untuk memecahkan suatu masalah, yaitu (Sutojo, dkk., 2011: 212):

1. Perancangannya tidak memerlukan persamaan matematik yang rumit
2. Mudah dimengerti
3. Memiliki toleransi terhadap data-data yang tidak tepat
4. Mampu memodelkan fungsi-fungsi nonlinear yang sangat kompleks
5. Dapat membangun dan mengaplikasikan pengalaman-pengalaman para pakar secara langsung tanpa harus melalui proses pelatihan
6. Dapat bekerja sama dengan teknik-teknik kendali secara konvensional
7. Logika *fuzzy* didasarkan pada bahasa alami

Sistem inferensi *fuzzy* adalah cara memetakan ruang *input* menuju ruang *output* menggunakan logika *fuzzy*. Empat elemen dasar sistem inferensi *fuzzy* antara lain (Sutojo, dkk., 2011: 232):

1. Basis pengetahuan *fuzzy*, yaitu kumpulan aturan (*rule*) *fuzzy* dalam bentuk pernyataan *IF…THEN*.
2. Fuzzifikasi, yaitu proses untuk mengubah *input* sistem yang mempunyai nilai tegas menjadi variabel linguistik menggunakan fungsi keanggotaan yang disimpan dalam basis pengetahuan *fuzzy*.
3. Mesin inferensi, yaitu proses untuk mengubah *input fuzzy* menjadi *output fuzzy* dengan cara mengikuti aturan-aturan yang telah ditetapkan pada basis pengetahuan *fuzzy*.

4. Defuzzifikasi, yaitu mengubah *output fuzzy* yang diperoleh dari mesin inferensi menjadi nilai tegas menggunakan fungsi keanggotaan yang sesuai dengan saat dilakukan fuzzifikasi.

Beberapa metode yang digunakan dalam sistem inferensi *fuzzy* adalah (Sutojo, dkk., 2011: 233-237):

1. Metode Tsukamoto

Dalam inferensinya, metode Tsukamoto menggunakan tahapan sebagai berikut:

a. Fuzzifikasi
b. Pembentukan basis pengetahuan *fuzzy* (*rule* dalam bentuk *IF…THEN*)
c. Mesin inferensi menggunakan fungsi implikasi *MIN* (*Minimum*)
d. Defuzzifikasi menggunakan metode Rata-rata (*Average*)

2. Metode Mamdani

Metode ini sering digunakan karena strukturnya yang sederhana. Pada metode ini, untuk mendapatkan *output* diperlukan 4 tahapan sebagai berikut:

a. Fuzzifikasi
b. Pembentukan basis pengetahuan *fuzzy* (*rule* dalam bentuk *IF…THEN*)
c. Aplikasi fungsi implikasi menggunakan fungsi *MIN* (*Minimum*) dan komposisi antar-*rule* menggunakan fungsi *MAX*(*Maximum*) dengan menghasilkan himpunan *fuzzy* baru
d. Defuzzifikasi menggunakan metode *Centroid* (Titik Tengah)

3. Metode Sugeno

Metode ini diperkenalkan oleh Takagi-Sugeno Kang pada tahun 1985. Dalam metode ini, *output* sistem berupa konstanta atau persamaan liniear. Dalam inferensinya, metode Sugeno menggunakan tahapan sebagai berikut:

a. Fuzzifikasi
b. Pembentukan basis pengetahuan *fuzzy* (*rule* dalam bentuk *IF…THEN*)
c. Mesin inferensi menggunakan fungsi implikasi *MIN* (*Minimum*)
d. Defuzzifikasi menggunakan metode Rata-rata (*Average*)

2.1.1.2 Jaringan saraf tiruan (*artificial neural network*)

Jaringan saraf tiruan adalah paradigma pengolahan informasi yang terinspirasi oleh sistem saraf secara biologis, seperti proses informasi pada otak manusia. Elemen utamanya adalah struktur dari sistem pengolahan informasi yang terdiri dari sejumlah besar elemen pemrosesan yang saling berhubungan (*neuron*), bekerja serentak untuk menyelesaikan masalah tertentu. Cara kerja jaringan saraf tiruan sama seperti cara kerja manusia, yaitu belajar melalui contoh. Beberapa contoh aplikasi jaringan saraf tiruan adalah implementasi di bidang kedokteran, yaitu pemodelan dan diagnosis sistem kardiovaskular, hidung elektronik, dan dokter instan; dan implemetasi di bidang bisnis, yaitu jaringan saraf tiruan yang diintegrasikan dengan merek dagang *The Airline Marketing Tactician* (*AMT*) menggunakan *back-propagation* untuk membantu kontrol pemasaran dari alokasi kursi penerbangan (Sutojo, dkk., 2011: 283-288).

Beberapa kelebihan yang dimiliki jaringan saraf tiruan antara lain (Sutojo, dkk., 2011: 284):

1. Belajar adaptif, yaitu kemampuan untuk mempelajari bagaimana melakukan pekerjaan berdasarkan data yang diberikan untuk pelatihan atau pengalaman awal.
2. *Self-Organization*, yaitu kemampuan membuat organisasi sendiri atau representasi dari informasi yang diterimanya selama waktu belajar.
3. *Real Time Operation*, yaitu perhitungan jaringan saraf tiruan yang dapat dilakukan secara paralel sehingga perangkat keras yang dirancang dan diproduksi secara khusus dapat mengambil keuntungan dari kemampuan ini.

Selain mempunyai beberapa kelebihan, jaringan saraf tiruan juga mempunyai kelemahan-kelemahan, yaitu (Sutojo, dkk., 2011: 284-285):

1. Tidak efektif jika digunakan untuk melakukan operasi-operasi numerik dengan presisi tinggi.
2. Tidak efisien jika digunakan untuk melakukan operasi algoritma aritmatika, operasi logika, dan simbolis.
3. Membutuhkan pelatihan untuk dapat beroperasi sehingga bila jumlah datanya besar, waktu yang digunakan untuk proses pelatihan sangat lama.

Salah satu elemen yang menentukan baik tidaknya suatu mode jaringan saraf tiruan adalah hubungan antar-*neuron* atau arsitektur jaringan. *Neuron-neuron* tersebut terkumpul dalam lapisan-lapisan yang disebut *neuron layers*. Terdapat 3 bagian lapisan penyusun jaringan saraf tiruan, yaitu (Sutojo, dkk., 2011: 292):

1. Lapisan *Input* (*Input Layer*)

Unit-unit dalam lapisan ini disebut unit-unit *input* yang bertugas menerima pola *input*-an dari luar yang menggambarkan suatu permasalahan.

2. Lapisan Tersembunyi (*Hidden Layer*)

Unit-unit dalam lapisan ini disebut unit-unit tersembunyi, yang mana nilai *output*-nya tidak dapat diamati secara langsung.

3. Lapisan *Output* (*Output Layer*)

Unit-unit dalam lapisan ini disebut unit-unit *output*, yang merupakan solusi jaringan saraf tiruan terhadap suatu permasalahan.

Beberapa arsitektur jaringan yang sering digunakan dalam jaringan saraf tiruan antara lain (Sutojo, dkk., 2011: 292-295):

1. Jaringan Lapisan Tunggal

Jaringan ini terdiri dari 1 lapisan *input* dan 1 lapisan *output*, yang mana setiap unit dalam lapisan *input* selalu terhubung dengan setiap unit yang terdapat pada lapisan *output*. Jaringan ini menerima *input* kemudian mengolahnya menjadi *output* tanpa melewati lapisan tersembunyi. Contoh jaringan saraf tiruan yang menggunakan jaringan ini adalah *ADALINE, Hopfield,* dan *Perceptron*.

2. Jaringan Lapisan Banyak

Jaringan ini mempunyai 3 jenis lapisan, yaitu lapisan *input*, lapisan tersembunyi, dan lapisan *output*. Jaringan ini dapat menyelesaikan permasalahan yang lebih kompleks dibandingkan dengan jaringan lapisan tunggal. Contoh jaringan saraf tiruan yang menggunakan jaringan ini adalah *MADALINE*, *backpropagation*, dan *Neocognitron*.

3. Jaringan dengan Lapisan Kompetitif

Jaringan ini memiliki bobot yang telah ditentukan dan tidak memiliki proses pelatihan. Jaringan ini digunakan untuk mengetahui *neuron* pemenang dari sejumlah *neuron* yang ada sehingga sekumpulan *neuron* bersaing untuk mendapatkan hak menjadi aktif. Contoh jaringan saraf tiruan yang menggunakan jaringan ini adalah *Learning Vector Quantization* (*LVQ*).

Berdasarkan cara memodifikasi bobotnya, pelatihan jaringan saraf tiruan dibagi menjadi dua, yaitu (Sutojo, dkk., 2011: 301- 392):

1. Pelatihan dengan Supervisi (pembimbing)

Dalam pelatihan ini, jaringan dipandu oleh sejumlah pasangan data (masukan dan target) yang berfungsi sebagai pembimbing untuk melatih jaringan hingga diperoleh bobot yang terbaik. Algoritma yang termasuk dalam pelatihan dengan supervisi antara lain:

a. *Hebb-Rule*

Model ini diperkenalkan oleh D.O. Hebb yang menggunakan cara menghitung bobot dan bias secara iteratif dengan memanfaatkan model pembelajaran dengan supervisi sehingga bobot dan bias dapat dihitung secara otomatis tanpa harus melakukan cara coba-coba. Arsitektur jaringan ini terdiri dari beberapa unit *input* dihubungkan langsung dengan sebuah unit *output*, ditambah dengan sebuah bias.

b. *Perceptron*

Model ini ditemukan oleh Rosenblatt (1962) dan Minsky – Papert (1969). Model jaringan ini merupakan model yang terbaik pada saat itu. Algoritma

pelatihan *perceptron* digunakan baik untuk *input* biner maupun bipolar, dengan θ tertentu.

c. *Delta-Rule*

Selama pelatihan pola, *Delta-Rule* akan mengubah bobot dengan cara meminimalkan *error* antara *output* jaringan dengan target.

d. *Backpropagation*

Backpropagation adalah metode penurunan gradien untuk meminimalkan kuadrat *error* keluaran. Pelatihan jaringan ini terdiri dari 3 tahap, yaitu tahap perambatan maju (*forward propagation*), tahap perambatan balik, dan tahap perubahan bobot dan bias. Arsitektur jaringan ini terdiri dari *input layer*, *hidden layer*, dan *output layer*.

e. *Heteroassociative Memory*

Jaringan saraf *heteroassociative memory* adalah jaringan yang dapat menyimpan kumpulan pengelompokan pola dengan cara menentukan bobot-bobotnya sedemikian rupa. Algoritma pelatihan yang biasa digunakan adalah *Hebb-Rule*.

f. *Bidirectional Associative Memory* (*BAM*)

Bidirectional Associative Memory (*BAM*) adalah model jaringan saraf yang memiliki 2 lapisan, yaitu lapisan *input* dan lapisan *output* yang mempunyai hubungan timbal balik antara keduanya (bersifat *bidirectional*). Arsitektur jaringan ini terdiri dari 3 *neuron* pada lapisan *input* dan 2 *neuron* pada lapisan *output*. Model jaringan ini terbagi menjadi 2 jenis yaitu *BAM* Diskrit dan *BAM* Kontinu.

g. *Learning Vector Quantization* (*LVQ*)

Learning Vector Quantization (*LVQ*) adalah suatu model pelatihan pada lapisan kompetitif terawasi yang akan belajar secara otomatis untuk mengklasifikasikan vektor-vektor *input* ke dalam kelas-kelas tertentu.

2. Pelatihan tanpa Supervisi

Dalam pelatihan ini, tidak ada pembimbing yang digunakan untuk memandu proses pelatihan. Jaringan hanya diberi *input* tetapi tidak mendapatkan target yang diinginkan sehingga modifikasi bobot pada jaringan dilakukan menurut parameter tertentu. Model jaringan yang termasuk dalam pelatihan tanpa supervisi adalah jaringan kohonen yang diperkenalkan oleh Prof. Teuvo Kohonen pada tahun 1982.

Pada jaringan kohonen, *neuron-neuron* pada suatu lapisan data akan menyusun dirinya sendiri berdasarkan *input* nilai tertentu dalam suatu *cluster*. *Cluster* yang dipilih sebagai pemenang adalah *cluster* yang mempunyai vektor bobot paling cocok dengan pola *input*, yaitu *cluster* yang memiliki jarak yang paling dekat.

2.1.1.3 Sistem Pakar (*Expert System*)

Sistem pakar mulai dikembangkan pada pertengahan 1960, ditandai dengan lahirnya sistem pakar pertama bernama *General-purpose Problem Solver* (*GPS*) yang dikembangkan oleh Newel dan Simon. Kemudian bermunculan sistem pakar lain di berbagai bidang seperti MYCIN untuk diagnosis penyakit, DENDRAL untuk mengidentifikasi struktur molekul campuran yang tak dikenal, XCON & XSEL untuk membantu konfigurasi sistem komputer besar, SOPHIE untuk analisis

sirkuit elektronik, Prospector digunakan di bidang geologi untuk membantu mencari dan menemukan deposit, FOLIO digunakan untuk membantu memberikan keputusan bagi seorang manajer dalam masalah stok dan investasi, DELTA dipakai untuk pemeliharaan lokomotif listrik diesel, dan sebagainya (Sutojo, dkk., 2011: 159-160).

Menurut Hartati dan Iswanti (2008: 2) sistem pakar merupakan salah satu teknik kecerdasan buatan yang menirukan proses penalaran manusia. Sistem pakar adalah suatu sistem yang dirancang untuk dapat menirukan keahlian seorang pakar dalam menjawab pertanyaan dan memecahkan suatu masalah. Dengan bantuan sistem pakar, seseorang yang bukan pakar dapat menjawab pertanyaan, menyelesaikan masalah serta mengambil keputusan yang biasanya dilakukan oleh seorang pakar (Sutojo, dkk., 2011: 13).

Pakar adalah seseorang yang memiliki pengetahuan khusus, pemahaman, pengalaman, dan metode-metode yang digunakan untuk memecahkan masalah dalam bidang tertentu. Seorang pakar memiliki kemampuan kepakaran seperti mengenali dan merumuskan suatu masalah, menyelesaikan masalah dengan cepat dan tepat, menjelaskan solusi dari suatu masalah, restrukturisasi pengetahuan, belajar dari pengalaman, memahami batas kemampuan, kemampuan untuk mengaplikasikan pengetahuannya dan memberi saran serta pemecahan masalah pada bidang tertentu (Hartati dan Iswanti, 2008: 11).

Menurut Kusrini (2006: 14-15) sistem pakar dapat digunakan oleh orang awam yang bukan pakar untuk meningkatkan kemampuan dalam memecahkan masalah. Sistem pakar juga dapat digunakan oleh pakar sebagai *assistant* yang

berpengetahuan, serta digunakan untuk memperbanyak atau menyebarkan sumber pengetahuan yang semakin langka.

Suatu sistem dikatakan sebagai sistem pakar jika memiliki ciri-ciri sebagai berikut (Sutojo, dkk., 2011: 162):

1. Terbatas pada domain keahlian tertentu
2. Dapat memberikan penalaran untuk data-data yang tidak lengkap atau tidak pasti
3. Dapat menjelaskan alasan-alasan dengan cara yang dapat dipahami
4. Bekerja berdasarkan kaidah tertentu
5. Mudah dimodifikasi
6. Basis pengetahuan dan mekanisme inferensi diletakkan terpisah
7. Keluarannya (*output*) bersifat anjuran
8. Sistem dapat mengaktifkan kaidah secara terpisah secara searah, sesuai dengan dialog dengan pengguna

Representasi pengetahuan merupakan metode yang digunakan untuk mengodekan pengetahuan dalam sebuah sistem pakar yang berbasis pengetahuan. Hal ini dimaksudkan untuk menangkap sifat-sifat penting dari suatu masalah sehingga informasi itu dapat diakses oleh prosedur pemecahan masalah. Bahasa representasi harus dirancang agar fakta-fakta dan pengetahuan lain yang terkandung di dalamnya dapat digunakan untuk penalaran (Kusrini, 2006: 24).

Menurut Hartati dan Iswanti (2008: 22) representasi pengetahuan dimaksudkan untuk mengorganisasikan pengetahuan dalam bentuk dan format tertentu agar dapat dimengerti oleh komputer. Pemilihan representasi pengetahuan yang tepat akan

menghasilkan sebuah sistem pakar yang efektif. Salah satu model representasi pengetahuan yang penting yaitu kaidah produksi (*production rule*).

Sistem pakar pada penelitian ini menggunakan model representasi pengetahuan berbasis kaidah produksi. Menurut Firebaugh (1988) dalam Hartati dan Iswanti (2008: 10) struktur sistem pakar yang berbasis kaidah produksi terdiri dari 4 komponen, yaitu:

1. Antarmuka pemakai

Menurut Kusrini (2006: 17) antarmuka merupakan penghubung antara pemakai dengan sistem pakar. Komponen ini berfungsi sebagai alat komunikasi antara sistem dan pengguna (*user*) yang penting sekali bagi pengguna. Komponen ini harus didesain sedemikian rupa sehingga efektif dan mudah digunakan terutama bagi pengguna yang tidak ahli dalam bidang yang diterapkan pada sistem pakar (Hartati dan Iswanti, 2008: 4-5).

2. Basis pengetahuan

Basis pengetahuan adalah komponen yang berisi sekumpulan kaidah yang berasal dari pengetahuan dalam domain tertentu dan secara umum disajikan dalam bentuk kaidah produksi (*IF…THEN…*). Pengetahuan pakar yang disajikan dalam format tertentu didapat dari sekumpulan pengetahuan pakar dan sumber-sumber pengetahuan lainnya seperti buku-buku, jurnal ilmiah, majalah, maupun dokumentasi tercetak lainnya. Basis pengetahuan diletakkan terpisah dari mesin inferensi agar pengembangan pengetahuan sistem pakar dapat dilakukan secara leluasa tanpa mengganggu mesin inferensi (Hartati dan Iswanti, 2008: 5).

3. Struktur kontrol (Mesin Inferensi)

Struktur kontrol merupakan *interpreter* kaidah atau mesin inferensi yang menggunakan pengetahuan-pengetahuan yang tersimpan dalam basis pengetahuan untuk memecahkan atau menyelesaikan permasalahan yang ada. Menurut Kusrini (2006: 35) inferensi merupakan proses untuk menghasilkan informasi berupa konklusi logis berdasarkan informasi yang tersedia atau fakta yang diketahui.

Dalam melakukan proses inferensi, sistem pakar memerlukan adanya proses pengujian kaidah-kaidah dalam urutan tertentu untuk mencari suatu kondisi yang sesuai dengan kondisi awal atau untuk memastikan kondisi yang sedang berjalan sudah dimasukkan ke dalam *database*. Proses pengujian itu disebut dengan perunutan atau penalaran, yaitu proses pencocokan fakta atau kondisi tertentu yang tersimpan dalam basis pengetahuan maupun pada memori kerja dengan kondisi yang dinyatakan dalam premis atau bagian kondisi pada suatu kaidah atau aturan (Hartati dan Iswanti, 2008: 45).

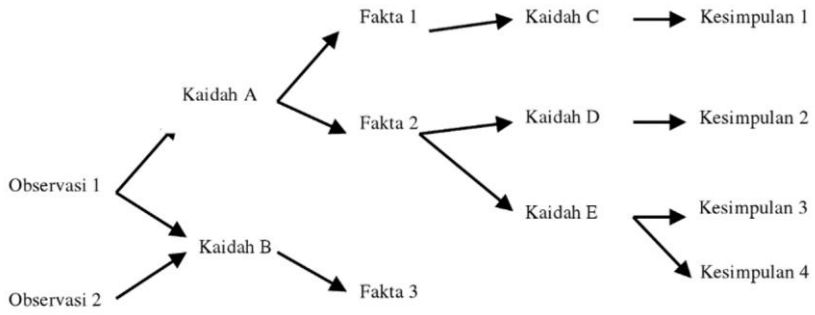
Ada beberapa konsep penalaran yang dapat digunakan oleh mesin inferensi yaitu:

a. Penalaran maju (*forward chaining*)

Konsep ini dapat juga disebut sebagai pencarian yang dimotori data (*data driven search*). Runut maju melakukan proses perunutan (penalaran) dimulai dari premis-premis atau informasi masukan (*IF*) terlebih dahulu kemudian menuju konklusi atau *derived information* (*THEN*). Konsep ini dapat dimodelkan sebagai berikut:

IF (informasi masukan)

THEN (konklusi)



Gambar 2.1. Metode *Forward Chaining*
(Sumber: Hayadi, 2016: 7)

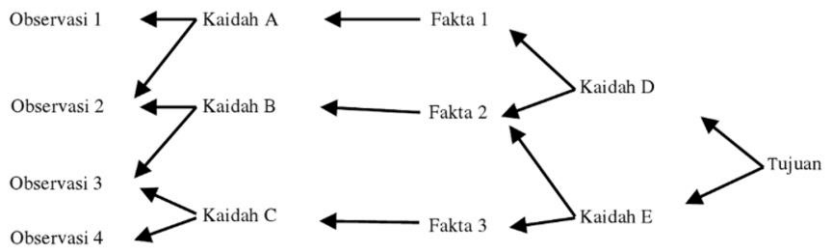
Informasi masukan dapat berupa suatu pengamatan sedangkan konklusi dapat berupa diagnosa sehingga dapat dikatakan jalannya penalaran runut maju dimulai dari pengamatan menuju diagnosa. Pada metode ini, sistem tidak melakukan praduga apapun terhadap konklusi, namun sistem akan menerima semua gejala yang diberikan pengguna lalu sistem akan memeriksa gejala-gejala tersebut dan selanjutnya mencocokkan dengan konklusi yang sesuai (Hartati dan Iswanti, 2008: 45-47).

b. Penalaran mundur (*backward chaining*)

Secara umum, konsep ini diaplikasikan ketika tujuan ditentukan sebagai kondisi atau keadaan awal. Konsep ini disebut juga *goal-driven search*. Arah penalaran atau perunutan dalam konsep ini berlawanan dengan *forward chaining*. Konsep ini dapat dimodelkan sebagai berikut:

Tujuan,

IF (kondisi)

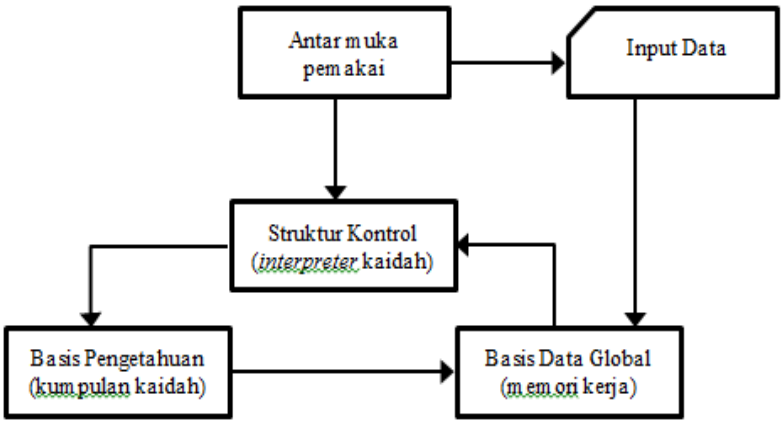


Gambar 2.2 Metode *Backward Chaining*
(Sumber: Hayadi, 2016: 8)

Proses penalaran pada *backward chaining* dimulai dari tujuan kemudian merunut balik ke jalur yang mengarah ke tujuan tersebut, untuk membuktikan bahwa bagian kondisi pada kaidah atau aturan benar-benar terpenuhi. Proses *internal* selalu memeriksa konklusi (tujuan) terlebih dahulu sebagai praduga awal, kemudian memeriksa dan memastikan gejala-gejala (kondisi) telah terpenuhi dan selanjutnya mengeluarkan konklusi sebagai *output*. Jika sistem menemukan ada bagian kondisi yang tidak terpenuhi maka sistem akan memeriksa konklusi (tujuan) pada aturan atau kaidah berikutnya (Hartati dan Iswanti, 46-47).

4. *Working memory* (memori kerja) atau basis data global

Berfungsi untuk mencatat status masalah yang terjadi dan *history* solusi. Memori kerja merupakan bagian yang berisi fakta-fakta masalah yang ditemukan dalam suatu sesi saat proses konsultasi terjadi (Kusrini, 2006: 19).



Gambar 2.3 Struktur Sistem Pakar Kaidah Produksi
(Sumber: Firebaugh, 1988 *dalam* Hartati dan Iswanti, 2008: 10)

Kusrini (2008: 33) menjelaskan bahwa kaidah menyediakan cara formal yang dituliskan dalam bentuk jika-maka (*IF-THEN*) untuk merepresentasikan rekomendasi, arahan, atau strategi. Kaidah *IF-THEN* menghubungkan antesenden (*antecedent*) dengan konsekuensi yang diakibatkannya. Berikut ini adalah contoh struktur kaidah *IF-THEN* yang menghubungkan obyek (Adedeji, 1992 *dalam* Hartati dan Iswanti, 2008: 25):

1. *IF* premis *THEN* konklusi
2. *IF* masukan *THEN* keluaran
3. *IF* kondisi *THEN* tindakan
4. *IF* antesenden *THEN* konsekuen
5. *IF* data *THEN* hasil
6. *IF* tindakan *THEN* tujuan
7. *IF* aksi *THEN* reaksi
8. *IF* gejala *THEN* diagnosa

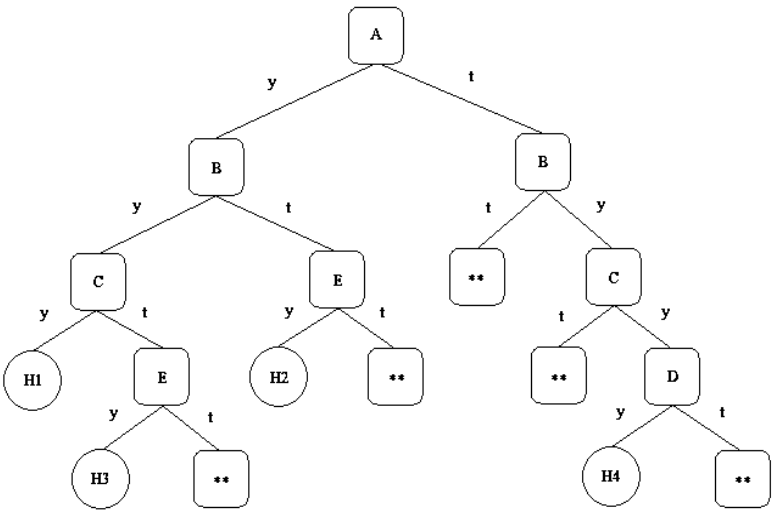
Premis mengacu pada fakta yang harus benar sebelum konklusi tertentu dapat diperoleh. Masukan mengacu pada data yang harus tersedia sebelum keluaran dapat diperoleh. Kondisi mengacu pada keadaan yang harus berlaku sebelum tindakan dapat diambil. Antesenden mengacu situasi yang terjadi sebelum konsekuensi dapat diamati. Data mengacu pada informasi yang harus tersedia sehingg sebuah hasil dapat diperoleh. Tindakan mengacu pada kegiatan yang harus dilakukan sebelum hasil dapat diharapkan. Aksi mengacu pada kegiatan yang menyebabkan munculnya efek dari tindakan tersebut. Gejala mengacu pada keadaan yang menyebabkan adanya kerusakan atau keadaan tertentu yang mendorong adanya pemeriksaan (diagnosa) (Hartati dan Iswanti, 2008: 25-26).

Sebelum sampai pada bentuk kaidah produksi, pengetahuan yang berhasil didapatkan dari domain tertentu disajikan dalam bentuk tabel keputusan kemudian dibuat pohon keputusannya. Berikut ini adalah contoh penyajian dalam bentuk tabel keputusan dan pohon keputusan (Hartati dan Iswanti, 2008: 26-39).

Tabel 2.1 Tabel Keputusan

Hipotesa / ***Evidence***	**Hipotesa 1**	**Hipotesa 2**	**Hipotesa 3**	**Hipotesa 4**
Evidence A	ya	ya	ya	Tidak
Evidence B	ya	tidak	ya	Ya
Evidence C	ya	tidak	tidak	Ya
Evidence D	tidak	tidak	tidak	Ya
Evidence E	tidak	Ya	ya	Tidak

Sumber: Hartati dan Iswanti (2008: 32)



Keterangan:

A = *evidence* A, H1 = hipotesa 1, y = ya

B = *evidence* B, H2 = hipotesa 2, t = tidak

C = *evidence* C, H3 = hipotesa 3, ** = tidak menghasilkan hipotesa tertentu

D = *evidence* D, H4 = hipotesa 4

Gambar 2.4 Pohon Keputusan
(Sumber: Hartati dan Iswanti, 2008: 33)

Dari gambar 2.4 dapat diketahui bahwa hipotesa H1 terpenuhi jika memenuhi *evidence* A, B, dan C. Hipotesa H2 terpenuhi jika memiliki *evidence* A dan *evidence* E. Hipotesa H3 akan terpenuhi jika memiliki *evidence* A, B, dan E. Hipotesa H4 akan dihasilkan jika memenuhi *evidence* B, C, dan D. Notasi "y" mengandung arti memenuhi *node* (*evidence*) di atasnya, notasi "t" artinya tidak memenuhi.

Dalam sesi konsultasi pada sistem pakar, *node-node* yang mewakili *evidence* biasanya akan menjadi pertanyaan yang diajukan oleh sistem. Dengan melihat pohon keputusan pada gambar 2.4 permasalahan dapat saja terjadi pada awal sesi konsultasi yaitu pada saat sistem pakar menanyakan "apakah memiliki *evidence* A?". Permasalahannya adalah apapun jawaban pengguna baik "ya" atau "tidak"

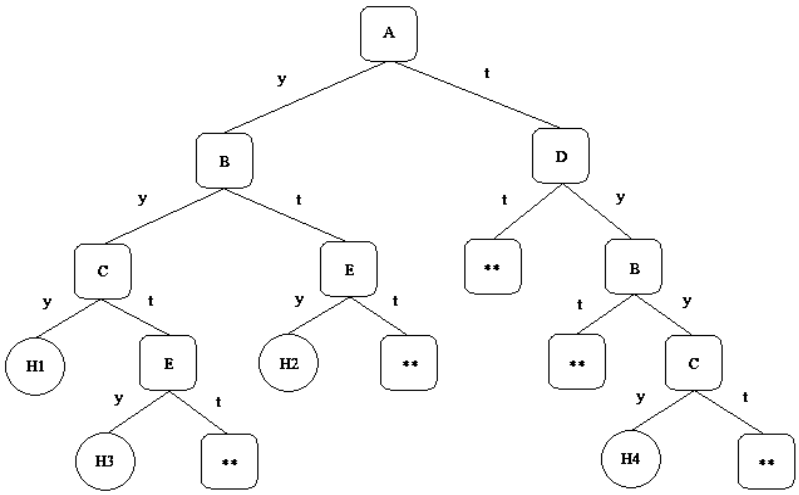
maka sistem akan menanyakan *evidence* B. Ini berarti jawaban pengguna tidak akan mempengaruhi sistem. Salah satu cara untuk mengatasi hal ini adalah dengan mengubah urutan pada tabel keputusan seperti terlihat pada tabel 2.2.

Tabel 2.2 Alternatif Tabel Keputusan

Hipotesa / ***Evidence***	**Hipotesa 1**	**Hipotesa 2**	**Hipotesa 3**	**Hipotesa 4**
Evidence A	Ya	ya	ya	Tidak
Evidence D	Tidak	tidak	tidak	Ya
Evidence B	Ya	tidak	ya	ya
Evidence C	Ya	tidak	tidak	Ya
Evidence E	Tidak	ya	ya	Tidak

Sumber: Hartati dan Iswanti (2008: 34)

Berdasarkan tabel 2.2 dapat dihasilkan pohon keputusan sebagai berikut:



Keterangan:

A = *evidence* A, H1 = hipotesa 1, y = ya

B = *evidence* B, H2 = hipotesa 2, t = tidak

C = *evidence* C, H3 = hipotesa 3, ** = tidak menghasilkan hipotesa tertentu

D = *evidence* D, H4 = hipotesa 4

Gambar 2.5 Alternatif Pohon Keputusan
(Sumber: Hartati dan Iswanti, 2008: 35)

Dilihat dari gambar 2.5, masing-masing *node* yang mewakili *evidence* tertentu untuk kondisi "y" dan "t" sudah tidak mengarah pada *evidence* yang sama. Hal ini berarti jawaban pengguna yang berbeda akan mengarah pada pertanyaan yang berbeda pula.

Kaidah yang dapat dihasilkan berdasarkan pohon keputusan pada gambar 2.5 adalah sebagai berikut:

1. Kaidah 1: *IF* A *AND* B *AND* C *THEN* H1
2. Kaidah 2: *IF* A *AND* B *AND* E *THEN* H3
3. Kaidah 3: *IF* A *AND* E *THEN* H2
4. Kaidah 4: *IF* D *AND* B *AND* C *THEN* H4

Model representasi pengetahuan kaidah produksi banyak digunakan pada aplikasi sistem pakar karena model representasi ini mudah dipahami dan bersifat deklaratif sesuai dengan jalan pikiran manusia dalam menyelesaikan suatu masalah, dan mudah diinterpretasikan.

Menurut Kusrini (2006: 14) terdapat beberapa alasan mengapa sistem pakar dikembangkan untuk menggantikan seorang pakar:

1. Dapat menyediakan kepakaran setiap waktu dan di berbagai lokasi.
2. Secara otomatis mengerjakan tugas-tugas rutin yang membutuhkan seorang pakar.
3. Seorang pakar akan pensiun atau pergi.
4. Menghadirkan/menggunakan jasa seorang pakar memerlukan biaya yang mahal.

5. Kepakaran dibutuhkan juga pada lingkungan yang tidak bersahabat (*hostile environment*).

Adapun kelebihan yang dimiliki sistem pakar antara lain (Kusrini, 2006: 15):

1. Membuat seorang yang awam dapat bekerja seperti layaknya seorang pakar.
2. Dapat bekerja dengan informasi yang tidak lengkap atau tidak pasti.
3. Meningkatkan *output* dan produktifitas, bekerja lebih cepat dari manusia sehingga mengurangi jumlah pekerja yang dibutuhkan dan akan mereduksi biaya.
4. Meningkatkan kualitas.
5. Sistem pakar menyediakan nasihat yang konsisten dan dapat mengurangi tingkat kesalahan.
6. Membuat peralatan yang kompleks lebih mudah dioperasikan karena sistem pakar dapat melatih pekerja yang tidak berpengalaman.
7. Handal (*realibility*).
8. Sistem pakar tidak dapat lelah atau bosan serta konsisten dalam memberikan jawaban dan selalu memberikan perhatian penuh.
9. Memiliki kemampuan untuk memecahkan masalah yang kompleks.
10. Memungkinkan pemindahan pengetahuan ke lokasi yang jauh serta memperluas jangkauan seorang pakar, dapat diperoleh dan dipakai dimana saja. Sistem pakar merupakan arsip yang terpercaya dari sebuah keahlian sehingga *user* seolah-olah berkonsultasi langsung dengan sang pakar meskipun sang pakar sudah pensiun.

Selain memiliki beberapa kelebihan yang dapat dimanfaatkan, sistem pakar juga memiliki beberapa kekurangan, yaitu (Sutojo, dkk., 2011: 161):

1. Biaya yang sangat mahal untuk membuat dan memeliharanya.
2. Sulit dikembangkan karena keterbatasan keahlian dan ketersediaan pakar.
3. Sistem pakar tidak 100% bernilai benar.

2.1.2 *Android*

Menurut *Nasruddin Safaat*, Android merupakan suatu sistem operasi dihandphone yang bersifat terbuka dan berbasis pada sistem operasi Linux. Android bersifat multiplatform yaitu bias digunakan oleh setiap orang yang ingin menggunakannya. Android menyediakan suatu platform terbuka untuk para pengembang dalam menciptakan suatu aplikasi sendiri untuk bermacam peranti bergerak. Android memiliki berberapa kelebihan diantaranya:

1. Multitasking

   Yaitu android bisa membuka beberapa aplikasi sekaligus tanpa harus menutup salah satunya.
2. Kemudahan dalam Notifikasi

   Adanya pemberitahuan seperti yang ada di SMS, Email, Facebook dan lain-lain, sehingga tidak akan melewatkan pemberitahuan tersebut
3. Mudah terhadap macam-macam Aplikasi Android lewat Google Android App Market

   Bisa didownload gratis dengan beraneka aplikasies yang siap di download di ponsel anda.

4. Bisa menginstal ROM yang dimodifikasi
5. Widget

   Bisa mengakses berbagai setting atau pengaturan dengan cepat dan mudah.

2.1.3 *Database* (basis data)

Menurut A.S. dan Shalahuddin (2013: 43-44) sistem *database* adalah sistem terkomputerisasi yang tujuan utamanya adalan memelihara data atau informasi yang sudah diolah dan membuat informasi tersedia saat dibutuhkan. *Database* adalah media untuk menyimpan data agar dapat diakses dengan mudah dan cepat. Kebutuhan basis data meliputi memasukkan, menyimpan, dan mengambil data serta membuat laporan berdasarkan data yang telah disimpan. Salah satu bentuk basis data yang dibutuhkan dalam sebuah sistem yaitu *Database Management System* (*DBMS*). *DBMS* adalah suatu sistem aplikasi yang digunakan untuk menyimpan, mengelola, dan menampilkan data. Syarat minimal dari *DBMS* antara lain (A.S. dan Shalahuddin, 2013: 44-45):

1. Menyediakan fasilitas untuk mengelola akses data
2. Mampu menangani integritas data
3. Mampu menangani akses data yang dilakukan secara bersamaan
4. Mampu menangani *backup* data

Ada beberapa *DBMS* yang paling banyak digunakan saat ini antara lain:

1. *DBMS* versi komersial, yaitu *Oracle, Microsoft SQL Server, IBM DB2*, dan *Microsoft Access*
2. *DBMS* versi *open source*, yaitu *MySQL, PostgreSQL, Firebird*, dan *SQLite*

Dalam alur hidup basis data (*Database Life Cycle*), terdapat tahapan yang dinamakan *physical database design*. Biasanya pada tahap ini dibuat rancangan fisik *database* yaitu *Physical Data Model* (*PDM*). *PDM* adalah model yang menggunakan sejumlah tabel untuk mengambarkan data serta hubungan antar data. Setiap tabel mempunyai sejumlah kolom yang mempunyai nama unik beserta tipe data yang digunakan. *PDM* merupakan konsep yang digunakan untuk menerangkan secara detail bagaimana data disimpan dalam *database*. *PDM* sudah dalam bentuk fisik perancangan *database* yang siap diimplementasikan ke dalam *DBMS* sehingga nama tabel pada *PDM* merupakan nama asli tabel yang akan diimplementasikan ke dalam *DBMS* (A.S. dan Shalahuddin, 2013: 63).

2.1.4 Validasi Sistem

Validasi mengacu pada sekumpulan aktifitas yang berbeda yang menjamin bahwa sistem atau perangkat lunak yang dibangun telah sesuai dengan yang diharapkan. Beberapa pendekatan dalam melakukan pengujian untuk validasi sistem antara lain (A.S. dan Shalahuddin, 2013: 275-276):

1. *Black-Box Testing* (pengujian kotak hitam)

Pendekatan ini dilakukan dengan menguji sistem atau perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. Tujuannya untuk mengetahui apakah fungsi-fungsi, masukan, dan keluaran dari sistem atau perangkat lunak sesuai dengan spesifikasi yang dibutuhkan.

Pengujian dilakukan dengan membuat kasus uji yang bersifat mencoba semua fungsi dengan menggunakan sistem atau perangkat lunak apakah sesuai dengan

spesifikasi yang dibutuhkan. Kasus uji yang dibuat untuk melakukan *black-box testing* harus dibuat dengan kasus benar dan kasus sala

2. White-Box Testing (pengujian kotak putih)

Pendekatan ini dilakukan dengan menguji sistem atau perangkat lunak dari segi desain dan kode program apakah mampu menghasilkan fungsi-fungsi, masukan, dan keluaran yang sesuai dengan spesifikasi yang dibutuhkan. *White-box testing* dilakukan dengan memeriksa logika dari kode program. Pembuatan kasus uji dapat mengikuti standar pengujian dari standar pemrograman yang ada.

2.2 Variabel Penelitian

Kulit adalah bagian paling luar dari jaringan tubuh kita. Kulit membungkus otot tubuh kita. Pada saat kulit terkelupas, rasa perih menyengat. Hal itu menunjukkan bahwa betapa kulit sangat memiliki fungsi yang sangat penting. Selain membungkus otot tubuh, juga memberikan perlindungan bagi jaringan-jaringan tubuh lain yang ada dibawahnya. Kulit meliputi seluruh jaringan kulit secara umum, termasuk kulit wajah.

2.2.1 Penyakit Kulit Eksim

Maharani (2015:55) penyakit kulit eksim termasuk jenis penyakit kulit akibat gangguan inflamasi kebanyakan kondisinya jangka panjang, menyebabkan kemerahan, pembengkakan, lesi dan plak pada kulit, kulit meradang, melepuh dan berisi cairan. Biasanya menimbulkan rasa gatal.

Gambar 2.6 Penyakit eksim pada kulit
(Sumber: Maharani, 2015: 55)

Eksim adalah istilah kedokteran untuk kelainan kulit yang mana kulit tampak meradang dan iritasi. Peradangan ini bisa terjadi dimana saja namun yang paling sering terkena adalah tangan dan kaki.

2.2.2 Penyakit Kulit Kurap

Maharani (2015: 104) penyakit kurap adalah salah satu penyakit kulit yang menular yang disebabkan oleh fungi. Masa infeksi kurap hingga terkena penyakit adalah beberapa hari.

Gambar 2.7 Penyakit kurap pada kulit
(Sumber: Maharani, 2015: 104)

2.2.3 Penyakit Kulit Panu

Maharani (2015: 102) panu adalah penyakit kulit yang menyerang manusia yang disebabkan oleh jamur. Penyakit panu ditandai dengan bercak yang terdapat pada kulit disertai rasa gatal pada saat berkeringat. Bercak-bercak ini bisa berwarna putih, coklat atau merah tergantung warna kulit si penderita. Penyakit ini biasanya menyerang pada semua bagian kulit. Umumnya menular, biasanya ditemukan pada kulit kepala, lipatan lengan, leher, wajah dan kaki.

Gambar 2.8 Penyakit panu pada kulit
(Sumber: Maharani, 2015: 102)

2.2.4 Penyakit Kulit Bisul

Maharani (2015: 94) bisul merupakan sekumpulan nanah (neutrofil mati) yang telah terakumulasi di rongga jaringan setelah terinfeksi sesuatu (umumnya karena bakteri dan parasit) atau barang asing (seperti luka tembakan/tikaman).

Bisul merupakan reaksi ketahanan jaringan untuk menghindari menyebarnya barang asing di dalam tubuh. Organisme atau barang asing membunuh sel disekitarnya, mengakibatkan keluarnya toksin. Toksin tersebut

menyebabkan radang, sel darah putih mengalir menuju tempat tersebut dan kemudian meningkatkan aliran darah di tempat tersebut.

Struktur terakhir bisul adalah dinding bisul yang terbentuk oleh sel sehat untuk mencegah barang asing tersebut masuk ke dalam tubuh dan mencegah terkenanya sel lain.

Gambar 2.9 Penyakit bisul pada kulit
(Sumber: Maharani, 2015: 94)

2.2.5 Penyakit Kulit Herpes

Maharani (2015: 83) penyakit herpes adalah jenis penyakit kulit yang disebabkan oleh virus. Bintik-bintik kecil yang tumbuh ini berubah menjadi gelembung-gelembung transparan berisi cairan, persis seperti pada cacar air namun hanya bergerombol di sepanjang kulit yang dilalui oleh saraf yang terkena. Bintik-bintik baru dapat terus bermunculan dan membesar sampai seminggu kemudian.

Gambar 2.10 Penyakit herpes pada kulit
(Sumber: Maharani, 2015: 83)

2.3 Software Pendukung

Software pendukung merupakan beberapa perangkat lunak yang digunakan untuk mendukung pembuatan sistem pakar dalam penelitian ini. Perangkat lunak tersebut antara lain: SDK *Android*, Bahasa Pemrograman *Java*, *Eclipse* dan *StarUML*.

2.3.1 *StarUML*

Gambar 2.11 Logo *StarUML*
(Sumber: A.S. dan Shalahuddin, 2011: 1)

StarUML merupakan salah satu *CASE* (*Computer-Aided Software Engineering*) *tools* atau perangkat pembantu berbasis komputer untuk rekayasa perangkat lunak yang mendukung alur hidup perangkat lunak (*life cycle support*). *StarUML* termasuk ke dalam kelompok *upper CASE tools* yang mendukung perencanaan strategis dan pembangunan perangkat lunak (A.S. dan Shalahuddin, 2013: 122-123).

Terdapat 13 macam diagram dalam *UML 2.3* yang dibagi menjadi 3 kategori yaitu (A.S. dan Shalahuddin, 2013: 140-141):

1. *Structure diagrams*

Kategori ini terdiri dari kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan. Diagram UML yang termasuk dalam kategori ini antara lain *class diagram, object diagram, component diagram, composite structure diagram, package diagram,* dan *deployment diagram*.

2. *Behaviour diagrams*

Kategori ini terdiri dari kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem. Diagram UML yang termasuk dalam kategori ini antara lain *use case diagram, acitivity diagram,* dan *state machine diagram*.

3. *Interaction diagrams*

Kategori ini terdiri dari kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antar subsistem pada suatu sistem. Diagram UML yang termasuk dalam kategori ini

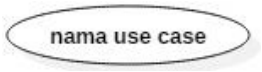
antara lain *sequence diagram, communication diagram, timing diagram,* dan *interaction overview diagram*.

Menurut A.S. dan Shalahudin (2013: 18) *use case* dan *sequence diagram* merupakan bagian dari desain sistem. Dalam penelitian ini, diagram yang akan digunakan untuk desain sistem yaitu:

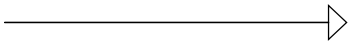
1. *Use case diagram*

Use case diagram merupakan pemodelan untuk menggambarkan kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu sistem atau lebih aktor dengan sistem informasi yang akan dibuat. *Use case diagram* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi-fungsi itu. Ada 2 hal utama yang terdapat pada *use case* yaitu aktor dan *use case*. Berikut ini adalah simbol-simbol yang digunakan dalam *use case diagram* (A.S. dan Shalahuddin, 2013: 155).

Tabel 2.3 Simbol *Use Case Diagram*

Simbol	**Deskripsi**
Use case 	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama *use case*

Tabel 2.3 Tabel lanjutan

Aktor/*actor*  nama aktor	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri. Aktor belum tentu merupakan orang, biasanya dinyatakan menggunakan kata benda di awal frase nama aktor
asosiasi/*association* 	Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor
Ekstensi/*extend* <<extend>> 	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walaupun tanpa *use case* tambahan itu. Arah panah mengarah pada *use case* yang ditambahkan.
Generalisasi/*generalization* 	Hubungan generalisasi dan spesialisasi (umum – khusus) antara 2 buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari fungsi lainnya. Arah panah mengarah pada *use case* yang menjadi generalisasinya (umum)

Sumber: A.S. dan Shalahuddin (2013: 162)

2. *Activity diagram*

sActivity diagram merupakan diagram yang menggambarkan *workflow* (aliran kerja) atau aktifitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Jadi dapat dikatakan bahwa *activity diagram* menggambarkan aktifitas sistem, bukan apa yang dilakukan oleh aktor. Simbol-simbol yang

digunakan dalam *activity diagram* ditampilkan dalam tabel berikut (A.S. dan Shalahuddin: 2013: 161-162).

Tabel 2.4 Simbol *Activity Diagram*

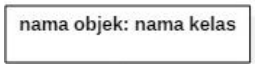
Simbol	**Deskripsi**
Status awal 	Status awal aktivitas sistem, sebuah diagram aktifitas memiliki sebuah status awal
Aktifitas 	Aktifitas yang dilakukan sistem, aktifitas biasanya diawali dengan kata kerja
Percabangan/*decision* 	Asosiasi percabangan dimana jika ada pilihan aktifitas lebih dari satu
Penggabungan/*join*	Asosiasi penggabungan dimana lebih dari satu aktifitas digabungkan menjadi satu
Status akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktifitas memiliki sebuah status akhir
Swimlane  atau 	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktifitas yang terjadi

Sumber: A.S. dan Shalahuddin (2013: 162-163)

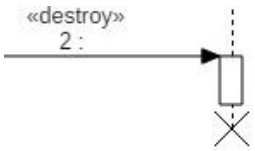
3. *Sequence diagram*

Sequence diagram menggambarkan kelakuan objek pada *use csase* dengan mendeskripsikan waktu hidup (*life cycle*) objek dan *message* (pesan) yang dikirimkan dan diterima antar objek. Jumlah *sequence diagram* yang harus digambar minimal sebanyak pendefinisian *use case* yang memiliki proses sendiri. Semakin banyak *use case* yang didefinisikan semakin banyak pula *sequence diagram* yang harus dibuat. Simbol-simbol yang digunakan pada *sequence diagram* ditampilkan dalam tabel berikut (A.S. dan Shalahuddin, 2013: 165-166).

Tabel 2.5 Simbol *Sequence Diagram*

Simbol	**Deskripsi**
Aktor/*actor*  	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri. Aktor belum tentu merupakan orang, biasanya dinyatakan menggunakan kata benda di awal frase nama aktor
Garis hidup/*lifeline* 	Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor
Objek  	Menyatakan objek yang berinteraksi pesan
Waktu aktif 	Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan di dalamnya. Aktor tidak memiliki waktu aktif

Tabel 2.5 Lanjutan

Pesan tipe *create* <<create>>	Menyatakan suatu objek membuat objek yang lain. Arah panah mengarah pada objek yang dibuat
pesan tipe *call* 1 : nama_metode()	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri. Arah panah mengarah pada objek yang memiliki operasi/metode.
Pesan tipe *send* 1 : masukan	Menyatakan bahwa suatu objek mengirimkan data/masukan/informasi ke objek lainnya. Arah panah mengarah pada objek yang dituju
pesan tipe *return* 1 : keluaran	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu. Arah panah mengarah pada objek penerima
Pesan tipe *destroy* 	Menyatakan suatu objek mengakhiri hidup objek lain. Arah panah mengarah pada objek yang diakhiri

Sumber: A.S. dan Shalahuddin (2013: 166- 167)

2.3.2 *Android*

Gambar 2.12 Logo *Android*
(Sumber: Tim EMS, 2014:2)

Android merupakan sebuah sistem operasi untuk perangkat lunak *mobile* berbasis linux yang mencakup sistem opearasi, *midleware* dan aplikasi. Android menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka. Terdapat dua jenis distributor sistem operasi Android. Pertama yang mendapat dukungan penuh dari Google atau *Google Mail Services* (GMS) dan kedua adalah yang benar-benar bebas dari distribusinya tanpa dukungan langsung dari Google atau dikenal sebagai *Open Handset Distribution* (OHD). Android dipuji sebagai "*platform mobile* pertama yang Lengkap, Terbuka, dan Bebas".

1. Lengkap (*Complete Platform*): Para desainer dapat melakukan pendekatan yang komprehensif ketika mereka sedang mengembangkan platform Android. Android merupakan sistem operasi yang aman dan banyak menyediakan tools dalam membangun software yang memungkinkan untuk peluang pengembangan aplikasi.
2. Terbuka (*Open Source Platform*): Platform Android disediakan melalui lisensi open source. Pengembang dapat dengan bebas untuk mengembangkan aplikasi. Android sendiri menggunakan Linux Kernel 2.6.

3. Free (*Free Platform*): Android adalah *Platform*/aplikasi yang bebas untuk *develop*. Tidak ada lisensi atau biaya royalti untuk dikembangkan pada *platform* Android. Tidak ada biaya keanggotaan yang diperlukan. Tidak ada biaya yang diperlukan untuk pengujian. Tidak ada kontrak yang diperlukan.

Aplikasi Android dapat didistribusikan dan diperdagangkan dalam bentuk apapun. Aplikasi Android ditulis dalam bahasa pemrograman java. Kode java dikompilasi bersama dengan data *file resource* yang dibutuhkan oleh aplikasi, dimana prosesnya dipackage oleh *tools* yang dinamakan "*apt tools*" ke dalam paket Android sehingga menghasilkan file dengan ekstensi apk. File apk itulah yang kita sebut aplikasi, dan nantinya dapat diinstall di perangkat mobile. Ada empat jenis komponen pada aplikasi Android yaitu:

1. *Activities*: Suatu *activity* akan menyajikan *user interface* (UI) kepada pengguna, sehingga pengguna dapat melakukan interaksi. Sebuah aplikasi Android bisa jadi hanya memiliki satu activity, tetapi umumnya aplikasi memiliki banyak activity tergantung pada tujuan aplikasi dan desain dari aplikasi tersebut.
2. *Service*: Service tidak memiliki *Graphic User Interface* (GUI), tetapi service berjalan secara backgraound, sebagai contoh dalam memainkan musik, service memainkan musik atau mengambil data dari jaringan, tetapi setiap service harus berada dalam kelas induknya.
3. *Broadcast Receiver***:** *Broadcast receiver* berfungsi menerima dan bereaksi untuk menyampaikan notifikasi. Contoh *broadcast* seperti notifikasi zona waktu berubah, baterai *low*, gambar telah selesai diambil oleh camera, atau pengubahan

referensi bahasa yang digunakan. Aplikasi juga dapat menginisiasi broadcast misalnya memberikan informasi pada aplikasi lain.

4. *Content Provider: Content Provider* membuat kumpulan aplikasi data secara spesifik sehingga bisa digunakan oleh aplikasi lain. Data disimpan dalam file sistem sistem seperti database SQLite. *Content Provider* menyediakan cara untuk mengakses data yang dibutuhkan oleh suatu *activity*, misalnya ketika kitaa menggunakan aplikasi yang membutuhkan peta (Map), atau aplikasi yang membutuhkan untuk mengakses data kontak dan navigasi, maka disinilah fungsi content navigasi.

Gambar 2.13 Tampilan *Android Developer Tools*
(Sumber Data: Nazruddin, 2015: 6)

2.3.3 Bahasa Pemrograman *Java*

Gambar 2.14 Logo Bahasa Pemrograman Java
(Sumber: Bambang Haryanto, 2014: 3)

Bahasa pemrograman Java memberi harapan menjadi perekat universal yang mengkoneksi pemakai dengan informasi dari *web server*, basis data, penyedia informasi dan sumber-sumber lain. Bahasa Java memiliki fitur keamanan *built-in*. Bahasa ava juga mempermudah pemrograman aplikasi *multithreading* (Bambang Hariyanto, 2014:1). Bahasa ava merupakan karya **Sun Microsystem Inc.** Rilis resmi level beta dilakukan pada November 1995. Pada tahun 1996, **Sun** mengeluarkan JSDK (*Java Software Development Kit*). Java telah berjalan pada segala perangkat dari laptop sampai pusat data, konsol game, sampai super komputer ilmiah.

Java telah berkembang dari semula ditujukan untuk pemprograman *applet* di *web browser* menjadi bahasa pemrograman pengembangan aneka ragam aplikasi, mulai dari yang berjalan di *handheld devices* seperti *handphone,* PDA (*Personal Digital Assistant*) sampai aplikasi tersebar skala *enterprise* di beragam komputer *server*. Java merupakan bahsa orientasi objek untuk pengembangan aplikasi mandiri, aplikasi berbasis *internet*, aplikasi untuk perangkat cerdas yang dapat

berkomunikasi lewat *internet*/jaringan komunikasi. Melalui teknologi Java, dimungkinkan perangkat *audio stereo* di rumah terhubung jaringan komputer. Java tidak lagi hanya bahasa untuk membuat **applet** yang memperindah halaman *web*, tapi Java telah menjadi bahasa untuk pengembangan aplikasi skala *enterprise* berbasis jaringan besar.

Sebagian besar bahasa pemrograman *modern* berdiri di atas pustaka-pustaka kelas yang telah ada untuk mendukung fungsionalitas. Pada bahasa Java, kelompok-kelompok kelas yang berkaitan erat dimasukkan di suatu paket, bervariasi sesuai edisi java. Masing-masing paket untuk maksud tertentu: *applet*, aplikasi standar, skala *enterprise*, dan produk konsumer. Java adalah bahasa yang dapat dijalankan di sembarang *platform*, di beragam lingkungan: *internet*, *consumer electronic products,* dan *computer applications. The java 2 Platform* tersedia dalam tiga edisi untuk keperluan berbeda berikut:

1. Java 2 *Standard Edition* (J2SE)
2. Java 2 *Enterprise Edition* (J2EE)
3. Java 2 *Micro Edition* (J2ME)

Pada pengembangan *enterprise applications,* kita menggunakan sejumlah besar paket. Pada *consumer electronic product,* hanya sejumlah kecil bagian bahasa yang digunakan. Masing-masing edisi berisi *Java 2 Software Development Kit* (SDK) untuk mengembangkan aplikasi dan *Java 2 Runtime Environment* (JRE) untuk menjalankan aplikasi. Fitur penting bahasa Java adalah bahasa ditujukan untuk membuat beragam jenis aplikasi secara seragam, yaitu:

1. Program di lingkungan *web browser*

Applet, program ini di eksekusi di *web browser* dari halaman web yang memuat *Java applet. Web Browser* kemudian menugaskan *Java Interpreter* (JRE – *Java Runtime Environtment*) untuk mengeksekusi *Java Applet* yang diterima. *Java applet* membuat langjkah besar, yang memungkinkan *web* menjadi sarana/media interaktif di mana halaman *web* menjadi dapat bereaksi terhadap masukan atau tanggapan pemakai.

2. Program di lingkungan *web server*

Java Server Pages, sebagai *web scripting* serupa dengan ASP, PHP dan sebagainya. Program ditempelkan di halaman html. Html ini tidak langsung dikirim ke *web browser* tapi diolah dulu oleh *web server* ke *web browser*. *Java Servlet*, komponen ini adalah semacam modul di *web server*. JSP akan diterjemahkan menjadi *servlet* agar mempercepat proses eksekusi.

3. Program mandiri (atau biasa disebut *stand-alone application*).

Java merupakan pilihan bagus untuk membuat *applet*. Sekaligus sebagai bahasa bermaksud umum (*general-purpose language*) untuk mengembangkan semua jenis program yang dapat dijalankam di komputer, sistem operasi apa pun asalkan terdapat *Java Interpreter* di *platform* itu. Sebuah program Java didiptakan dengan *file* teks ekstensi .java. Program ini dikompilasi menghasilkan satu *file bytecode* berekstensi .**class** atau lebih. *Bytecode* adalah serangkaian instruksi serupa kode

mesin. Perbedaannya adalah kode mesin harus dijalankan sistem komputer tertentu, sementara *bytecode* berjalan di *Java interpreter* yang tersedia di semua *platform.*

4. Program mandiri sebagai pustaka komponen

Untuk pengembangan aplikasi, java menyediakan Bean untuk mendukung RAD (rapid application development) yang berbasis visual seperti Visual Basic. Bahasa untuk pengembangan aplikasi objek-objek tersebar skala *enterprise*. Terdapat teknologi Java untuk mengembangkan komponen yaitu EJB (**Enterprises JavaBeans**) yang berjalan di *application server.* EJB mendukung *Componenent-based Software Engineering. Aplication Server* adalah *middleware* yang bertugas menjadi intermediasi beragam *server* di aplikasi tersebar skala enterprise. Dukungan Java terhadap sistem tersebar berupa spesifikasi J2EE lengkap meliputi JDBC (*Java Database Connectivity*), RMI (*Remote Method Invocation*), EJB (*Enterprise Java Beans*), JMS (*Java Messaging System*)

5. *JavaScript* bukan merupakan program Java

Namun sintaks dan semantiknya seperti bahasa Java. JavaScript tidak termasuk teknologi Java. Java adalah pemrograman orientasi objek yang berukuran kecil, sederhana, aman, diinterpretasi atau dioptimasi secara dinamis, ber-*bytecode*, netral arsitektur, mempunyai *garbage-collector, multithreading*, mempunyai mekanisme penanganan pengecualian (*exeception handling*), berbasis tipe untuk penulisan program mudah diperluas secara dinamis serta diperuntukkan untuk sistem tersebar (Bill Joy *dalam* Bambang Hariyanto, 2014: 10).

2.3.4 *Eclipse*

Gambar 2.15 Eclipse
(Sumber: Tim EMS, 2015: 37)

Untuk IDE (*Integrated Development Environment)* yang dipakai dalam pemrograman Android, baik berbasis *Java* atau pemaketan dengan HTML dengan *Cordova* dapet menggunakan *Eclipse.* Sebenarnya tools yang dapat dipakai tidak hanya *Eclipse*, tetapi ada juga tool lainnya seperti *Android Studio* (Tim EMS, 2015: 35)

Pemrograman *Android* secara garis besar terdiri atas 2 metode, di mana keduanya menggunakan *eclipse*. Yang pertama menggunakan *Java* dan yang kedua menggunakan *html* (Tim EMS, 2015: 53).

2.4 Penelitian Terdahulu

Untuk mendukung teori yang berkaitan dengan penelitian, peneliti mencantumkan beberapa penelitian terdahulu di bidang sistem pakar dalam kategori diagnosis.

Paryati (2011), Sistem Pakar Berbasis Web untuk Mendiagnosa Penyakit Kulit. Sistem pakar yang dibangun untuk mendiagnosa penyakit dan cara

penyembuhannya. Sistem ini bertujuan membantu user dapat mengetahui jenis penyakit yang di derita dan penyembuhannya melalui air belerang dan ramuan tradisional atau obat tradisional, serta informasi ramuan obat guna membantu proses penyembuhannya. Data rekomendasi yang dihasilkan dalam sistem ini dilengkapi dengan jenis pennyakit, gejala penyakit dan cara penyembuhannya sehingga user dapat mengetahui jenis penyakit yang diderita dan cara pengobatannya. Sistem ini akan menganalisis jawaban dari setiap pertanyaan yang diberikan agar dapat memperoleh jawaban berdasarkan basis pengetahuan yang terdapat dalam sistem pakar ini. Sebelum menganalisis jawaban, sistem terlebih dahulu memberikan sejumlah pertanyaan kepada *user* melalui *interface* tentang gejala penyakit yang diderita. Sistem akan menganalisis jawaban dari setiap *user* dengan melakukan proses pelacakan pada basis pengetahuan.

Joko S Dwi Raharjo, Damdam Damiyana, Supardi (2016), Sistem Pakar Diagnosa Penyakit Kulit Dengan Menggunakan Metode *Forward Chaininng* Berbasis Android. Di era digital saat ini komputer digunakan untuk mengolah pengetahuan sehingga proses pengambilan keputusan dapat lebih cepat, efisien dan akurat. Sebuah teknik untuk membuat komputer dapat mengolah pengetahuan telah dikenal sebagai teknik kecerdasan buatan (*artificial intelligence technique*). Dengan teknologi ini maka komputer dapat melakukan hal-hal yang sebelumnya hanya dapat dilakukan oleh manusia. Sistem pakar ini menggunakan *Forward Chaining. Forward Chaining* merupakan metode inferensi yang melakukan penalaran dari suatu masalah kepada solusinya. Dalam melakukan proses *Forward Chaining,* perlu suatu kumpulan aturan atau (*rules*), aturan yang ada ditelusuri satu

persatu hingga penelusuran dihentikan karena kondisi terakhir telah terpenuhi. Dalam melakukan perancangan aplikasi ini menggunakan metode *Rational Unified Model (RUP).* Metode *RUP* merupakan metode rekayasa perangkat lunak yang berfokus mengembangkan dengan model *Unified Model Language (UML)* dan menggunakan konsep Object Oriented. Hasil rancangan sistem ini dalam bentuk aplikasi *mobile android* yang dapat digunakan oleh semua orang yang ingin mengetahui tentang gejala penyakit kulit yang disebabkan oleh virus. Perancangan sistem pakar inimenggunakan bahasa pemrograman *Android Studio.*

I putu Bayu Krisnawan, I Ketut Gede Darma Putra, I Putu Agung Bayupati (2014)**,** Sistem Pakar Diagosa Penyakit Kulit dan Kelamin Dengan Metode *Certainty Factor* dan *Fuzzy Logic,* penyakit kulit dan kelamin menjadi sangat marak di kalangan masyarakat di berbagai tingkatan di negara Indonesia. Puskesmas atau rumah sakit negri jarang menyediakan fasilitas pengobatan kulit dan kelamin, mengakibatkan orang miskin harus berobat ke dokter spesialis yang memakan banyak sekali biaya. Sistem Pakar Penyakit Kulit dan Kelamin dapat memberikan hasil diagnosa penyakit kulit berupa nama penyakit yang di derita oleh pasien beserta keterangan informasi penyakit dan langkah-langkah yang harus di lakukan dalam menyembuhkan penyakit yang di derita. Sistem Pakar Penyakit Kulit dan Kelamin dapat memberikan informasi presentase kemungkinan pasien mengidap suatu penyakit dengan tingkat rata-rata akurasi diagnosa menurut pakar 73%.

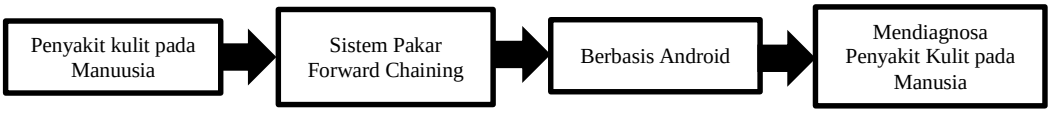
Fitri Nuraeni (2016), Aplikasi Pakar Untuk Diagnosa Penyakit Kulit Menggunakan Metode Forward Chaining di Al Arif Skin *Care* Kabupaten Ciamis.

Berbagai penyakit dan gangguan pada kulit dapat disebabkan oleh beberapa faktor seperti perubahan iklim, lingkungan dan kesehatan yang buruk, virus bakteri, daya tahan tubuh, reaksi alergi dan lain-lain. Penentuan penyakit kulit tidak boleh dilakukan secara sembarangan, kearena penyakit kulit bisa sangat berbahaya bila terjadi kesalahan dalam perawatan dan penanganannya. Oleh sebab itu, konsultasi mengenai penyakit kulit harus dilakukan dengan dokter ahli atau pakar. Permasalahan yang sering muncul adalah ketersediaan dokter ahli atau pakar yang memiliki pengetahuan di bidang tertentu cukup terbatas sementara banyak pasien yang harus segera diketahui penyakitnya dan segera ditangani. Metode penelitian yang digunakan dalam penelitian ini adalah metode *Forward Chaining,* yang dapat memberikan informasi berupa gambar dan kata-kata. Sehingga aplikasi sistem pakar ini menjadi media konsultasi bagi pasien penyakit kulit.

Linda Marlinda (2015), Sistem Pakar Diagnosa Penyakit Kulit Pada Manusia Menggunakan Apotek Hidup Menggunakan Simple Additive Weighting. Pengobatan dengan menggunakan apotek hidup sekarang ini banyak diminati oleh masyarakat, karena harganya terjangkau dan mudah didapat. Namun banyak dari masyarakat yang meracik obat-obat tradisional hanya berdasarkan perkataan orang lain atau pengalaman sendiri, sehingga menyebabkan salah penggunaan racikan dan dosis yang kurang tepat dalam pengolahannya khususnya untuk mendeteksi penyakit kulit yang sering terjadi dalam kehidupan sehari-hari. Metode yang digunakan dalam penelitian ini menggunakan metode simple additive weighting yang merupakan salah satu metode dari *attribute decision making* atau banyaknya pemilihan kriteria dari rating kinerja pada setiap alternatif.

2.5 Kerangka Pemikiran

Kerangka pemikiran memuat pemikiran terhadap alur yang dipahami sebagai acuan dalam pemecahan masalah yang diteliti secara logis dan sistematis. Kerangka berfikir yang baik akan menjelaskan secara teoritis pertautan antar variabel yang diteliti (Sugiyono, 2014: 60). Berikut ini adalah kerangka pemikiran yang mendasari penelitian ini.



Gambar 2.16 Kerangka Pemikiran
(Sumber: Data Penelitian: 2016)

Data-data yang dibutuhkan berkaitan dengan penyakit kulit yang dianalisis terlebih dahulu agar lebih sederhana dan mudah dilakukan proses pengolahan datanya. Data-data tersebut kemudian diolah menggunakan metode sistem pakar *forward chaining* untuk membuat aturan (*rule*) yang akan digunakan. Sistem pakar dengan metode *forward chaining* dibuat menggunakan bahasa pemrograman *java* yang ditampilkan dalam bentuk layout dengan format xml. Sistem pakar ini dibuat berdasarkan basis pengetahuan yang ditanamkan pada *java class,* sehingga aplikasi ini merupakan aplikasi *stand alone* yang tidak memerlukan aplikasi pihak kedua seperti SQLite untuk membangun database.