BAB II
TINJAUAN PUSTAKA

2.1 Teori Dasar

Deskripsi teori paling tidak berisi tentang penjelasan terhadap variabel-variabel yang diteliti melalui pendefinisian, dan uraian yang lengkap dan mendalam dari berbagai referensi, sehingga ruang lingkup, kedudukan dan prediksi terhadap hubungan antara variabel yang akan diteliti menjadi lebih jelas dan terarah (Sugiyono, 2014: 58).

Pada bab ini akan dijelaskan tentang beberapa teori dasar antara lain kecerdasan buatan atau *Artificial Intelligence* (*AI*) dan beberapa subdisiplin ilmunya seperti logika *fuzzy* (*fuzzy logic*), jaringan saraf tiruan (*artificial neural network*), dan sistem pakar (*expert system*); *web,* basis data, dan validitas sistem.

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 Sutojo dkk (2011: 3) pada abad ke 17 sampai abad ke 19 adalah merupakan titik awal perkembanagan kecerdasan buatan hal ini ditandai oleh penemuan-penemuan sebagai berikut:

1. Rene Descartes mengemukakan bahwa semua tidak ada yang pasti, kecuali kenyataan bahwa seseorang bisa berpikir.
2. Blaise Pascal berhasil menciptakan mesin penghitung digital mekanis pertama pada abad 1642.
3. Charles Babbage dan Ada Lovelace berhasil membuat mesin perhitungan mekanis yang dapat deprogram.
4. Bertand Russell dan Alfred Whitehead menerbitkan buku Principia Mathematica, yang merombak logika formal.
5. Walter Pitts menerbitkan Kalkulus Logis pada 1943, yang merupakan pondasi untuk jaringan saraf tiruan.

Pada tahun 1950-1970 dimana para ilmuan dan para peneliti mulai memikirkan cara agar mesin dapat melakukan pekerjaan seperti yang dikerjakan oleh manusia. Hal ini ditandai oleh beberapa penemuan-penemuan berikut:

1. Pada Februari 1951, University of Manchester telah berhasil mengembangkan komputer elektronik pertama di dunia yang diberi nama Ferranti Mark I.
2. Pada 1951 sebuah program permainan catur berhasil dibuat oleh Dietrich Prinz.
3. Alan Turing seorang matematikawan Inggris pertama kali mengusulkan adanya tes untuk melihat bisa tidaknya sebuah mesin dikatakan cerdas.
4. Jhon McCarty membuat istilah "kecerdasan buatan" pada konferensi pertama yang disediakan untuk pokok persoalan ini.
5. Eliza deprogram oleh Joseph Weinzenbaum pada tahun 1967. Program ini mampu melakukan terapi terhadap pasien dengan memberikan beberapa pertanyaan.
6. Alain Colmeraur mengembangkan bahasa komputer prolog.

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).

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.

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

Sistem pakar menggantikan seorang pakar dalam suatu situasi tertentu, maka system harus menyediakan pendukung yang diperlukan oleh pemakai yang tidak memahami masalah teknik (Hartati dan Iswanti, 2008: 4).

2. Basis pengetahuan

Basis pengetahuan merupakan kumpulan pengetahuan bidang tertentu pada tingkatan pakar dalam format tertentu. Pengetahuan ini diperoleh dari akumulasi pengetahuan pakar dan sumber-sumber pengetahuan lainnya seperti yang telah

disebutkan sebelumnya. Basis pengetahuan bersifat dinamis, bisa berkembang dari waktu ke waktu. Perkembangan ini disebabkan karena pengetahuan selalu bertambah, terupdate (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.

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)

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: 45-46).

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,

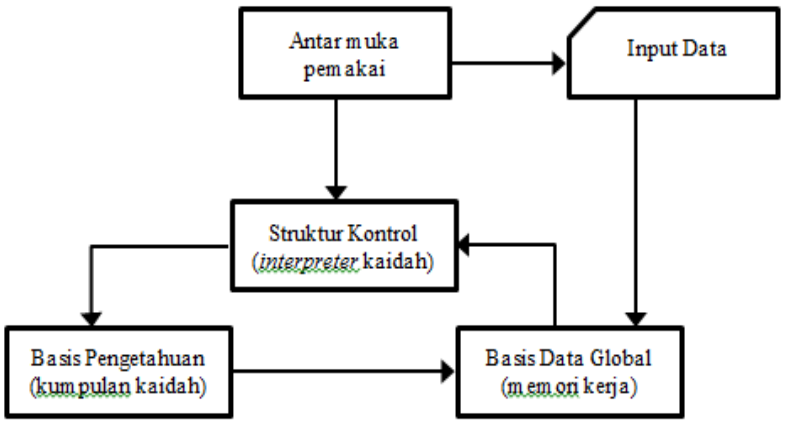
IF (kondisi)

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

Merupakan bagian dari system pakar yang menyediakan fakta-fakta yang diperoleh saat dilakukan proses konsultasi. Fakta-fakta inilah yang nantinya akan diolah oleh mesin inferensi berdasarkan pengetahuan yang disimpan dalam basis pengetahuan untuk menentukan suatu keputusan pemecahan masalah. Konklusinya bisa berupa hasil diagnosa, tindakan dan akibat (Hartati dan Iswanti, 2011: 6).



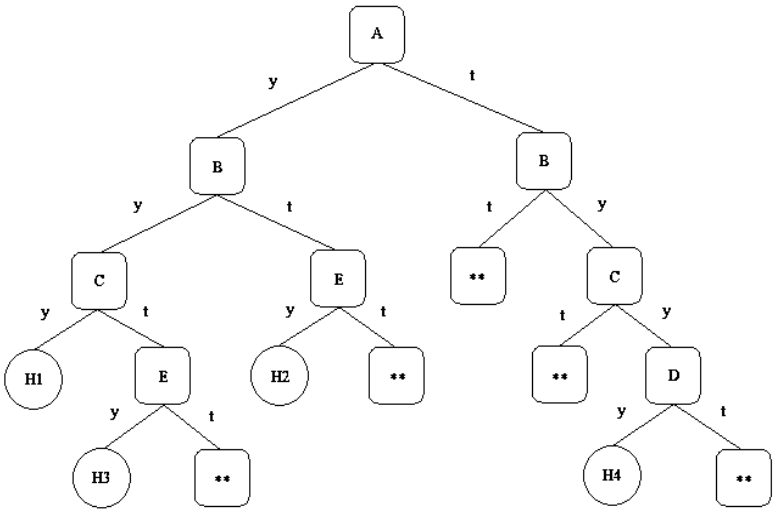
Gambar 2.1 Struktur Sistem Pakar Kaidah Produksi
(Sumber: Firebaugh, 1988 *dalam* Hartati dan Iswanti, 2008: 10)

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 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)



Gambar 2.2 Pohon Keputusan
(Sumber: Hartati dan Iswanti, 2008: 33)

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

Dari gambar 2.2 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.

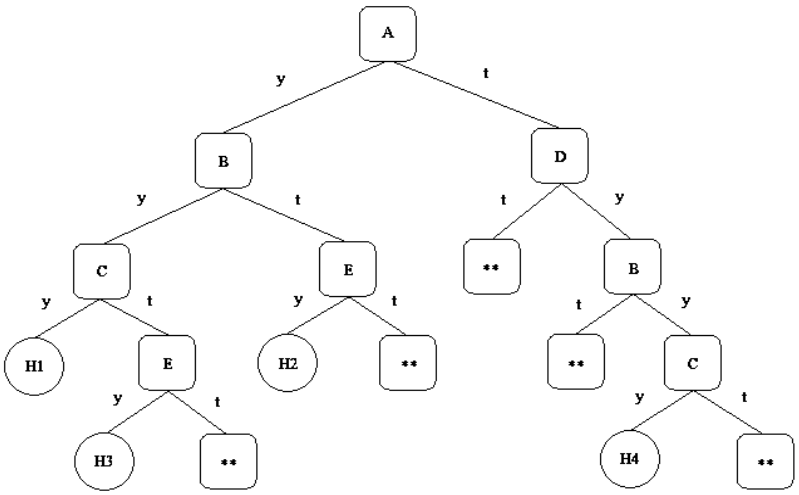
Dalam sesi konsultasi pada sistem pakar, *node-node* yang mewakili *evidence* biasanya akan menjadi pertanyaan yang diajukan oleh sistem. Dengan melihat pohon keputusan pada gambar 2.2 permasalahan dapat saja terjadi pada awal sesi konsultasi yaitu pada saat sistem pakar menanyakan "apakah memiliki *evidence* 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:



Gambar 2.3 Alternatif Pohon Keputusan
(Sumber: Hartati dan Iswanti, 2008: 35)

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

Dilihat dari gambar 2.3, 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 2.3 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.

Adapun kelebihan yang dimiliki sistem pakar antara lain (Sutojo, dkk., 2011: 161):

1. Meningkatkan produktivitas, karena system pakar dapat bekerja lebih cepat daripada manusia.
2. Membuat seorang yang awam bekerja seperti layaknya seorang pakar.
3. Meningkatkan kualitas, dengan memberi nasehat yang konsisten dan mengurangi kesalahan.
4. Mampu menangkap pengetahuan dan kepakaran seseorang.
5. Dapat beroperasi di lingkungan yang berbahaya.
6. Memudahkan akses pengetahuan seorang pakar.
7. Andal, system pakar tidak pernah menjadi bosan dan kelelahan atau sakit.
8. Meningkatkan kapasitas system komputer. Integrasi system pakar dengan system pakar komputer lain membuat system lebih efektif dan mencakup lebih banyak aplikasi.
9. Mampu bekerja dengan informasi yang tidak lengkap atau tidak pasti.
10. Bisa digunakan sebagai media pelengkap dalam pelatihan.

11. Menigkatkan kemampuan untuk menyelesaikan masalah karena system pakar mengambil sumber pengetahuan dari banyak pakar.

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 *Web*

Menurut Sidik dan Pohan (2009: 1) *WWW* (*World Wide Web*) atau yang lebih dikenal dengan *web* merupakan salah satu layanan yang didapat oleh pengguna komputer yang terhubung dengan internet. *Web* pada awalnya adalah ruang informasi dalam internet, dengan menggunakan teknologi *hypertext*. Pengguna dituntun untuk menemukan informasi dengan mengikuti *link* yang disediakan dalam dokumen *web* yang ditampilkan dalam *browser web*. Sekarang *web* menjadi standar *interface* pada layanan-layanan yang ada di internet seperti komunikasi melalui *e-mail, chatting*, transaksi bisnis, pencarian informasi, dan sebagainya.

Web memudahkan pengguna komputer untuk berinteraksi dengan pelaku internet lainnya dan menelusuri informasi di Internet. Banyak perusahaan yang mengadopsi *web* sebagai bagian dari strategi teknologi informasinya karena beberapa alasan yaitu: akses informasi yang mudah, *setup server* lebih mudah, informasi mudah didistribusikan, dan bebas *platform*, artinya informasi dapat

disajikan oleh *browser web* pada sistem operasi apapun karena adanya standar dokumen berbagai tipe data yang disajikan (Sidik dan Pohan, 2009: 2).

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):

a. *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 salah.

b. *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

Variabel penelitian merupakan segala sesuatu yang berbentuk apa saja yang dimiliki orang, obyek atau kegiatan yang mempunyai variasi tertentu yang ditetapkan oleh peneliti untuk dipelajari sehingga diperoleh informasi tentang hal tersebut, kemudian ditarik kesimpulannya (Sugiyono, 2014: 38). Obyek yang digunakan dalam penelitian ini adalah balita dan variabel penelitian yang ditetapkan yaitu kebutuhan gizi balita.

2.2.1 Gizi

Menurut Lumbanbatu dan Novriyeni (2014: 11) Istilah gizi berasal dari bahasa Arab *giza* yang berarti zat makanan, dalam bahasa Inggris dikenal dengan *nutrision* yang berarti bahan makanan atau zat gizi atau sering diartikan sebagai ilmu gizi.

Menurut Almatsier (2009) *dalam* Lumbanbatu dan Novriyeni (2014: 12) Zat Gizi adalah ikatan kimia yang diperlukan tubuh untuk melakukan fungsinya yaitu menghasilkan energi, membangun, memelihara jaringan serta mengatur proses-proses jaringan. Gizi merupakan bagian penting yang dibutuhkan oleh tubuh guna perkembangan dan pertumbuhan untuk memperoleh energi, agar manusia dapat melaksanakan kegiatan fisiknya sehari-hari.

Ada dua tujuan pengaturan makan untuk balita. Yang pertama adalah memberikan zat gizi yang cukup bagi kebutuhan hidup, yaitu untuk pemeliharaan atau pemulihan serta peningkatan kesehatan, pertumbuhan dan perkembangan fisik dan psikomotor, serta melakukan aktivitas fisik. Yang kedua adalah untuk mendidik kebiasaan makan yang baik.

Makanan yang baik untuk balita haruslah memenuhi syarat-syarat sebagai berikut lain:

1. Memenuhi kecukupan energy dan semua zat gizi sesuai dengan umur.
2. Susunan hidangan di sesuaikan dengan pola menu seimbang, bahan makanan yang tersedia setempat, kebiasaan makanan dan selera terhadap makanan.
3. Bentuk dan porsi makanan disesuaikan dengan daya terima, toleransi dan keadaan fisik balita.
4. Memperhatikan kebersihan perorangan dan lingkungan.

Menurut Fikawati dkk (2015: 157-160) dan diperkuat peraturan menteri kesehatan Republik Indonesia nomor 75 tahun 2013 tentang angka cakupan gizi

yang dianjurkan bagi balita bangsa Indonesia ada beberapa faktor yang harus dipenuhi diantaranya:

a. Energi

Kebutuhan energi balita mulai berkurang dibulan-bulan selanjutnya saat laju pertumbuhannya menurun.

b. Protein

Protein adalah koponen yang paling dasar pada protoplasma dalam sel, karena itu asupan protein yang cukup penting untuk pertumbuhan normal balita.

c. Lemak

Lemak merupakan sumber energi utama pada balita. Kebutuhan lemak tidak jenuh cukup tinggi terutama untuk pembentukan sel saraf.

d. Air

Jumlah air yang dibutuhkan balita lebih banyak dari jumlah yang dibutuhkan orang dewasa jika dilihat berdasarkan ukuran tubuhnya.

e. Karbohidrat

Sumber penting dari karbohidrat adalah gula dan karbahidrat kompleks. Sejak gula dapat disintesa dari asam amino dan gliserol dari lemak, tidak ada rekomendasi tidak ada rekomendasi untuk asupan karbohidrat. Namun, dianjurkan lebih dari setengah kecukupan energi pada bayi dipenuhi dari karbohidrat kompleks.

f. Serat

Serat makanan penting untuk kerja usus normal, di anjurkan sekitar 170-300 mg/kg BB. Kebutuhan ini dipenuhi dari sereal, buah-buahan, roti dan sayuran.

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: *XAMPP*, *phpMyAdmin, PHP, HTML, CSS, jQuery, MySQL, Notepade++,* dan *StarUML*.

2.3.1 *XAMPP* (*X Apache MySQL PHP Perl*)

Gambar 2.4 Logo *XAMPP*
(Sumber: https://wiki.bitnami.com/@api/deki/files/527/=xampp-logo.jpg)

Menurut Sidik dan Pohan (2009: 6) *server web* adalah komputer yang digunakan untuk menyimpan dokumen-dokumen *web* yang akan melayani permintaan dokumen *web* dari kliennya. *XAMPP* adalah sebuah perangkat lunak *web server apache* yang didalamnya sudah tersedia *database server MySQL* dan

dapat mendukung pemrograman *PHP*. *XAMPP* merupakan *software* yang mudah digunakan , gratis, dan mendukung instalasi di *Linux* dan *Windows*. Keuntungan lainnya adalah cukup dengan menginstal *XAMPP* sudah tersedia *Apache Web Server*, *MySQL Database Server*, *PHP support* (*PHP* 4 dan *PHP* 5) dan beberapa modul lainnya (Februariyanti dan Zuliarso, 2012: 129).

2.3.2 *phpMyAdmin*

Gambar 2.5 Logo *phpMyAdmin*
(Sumber: https://www.phpmyadmin.net/static/images/logo-og.png)

phpMyAdmin adalah perangkat lunak gratis yang ditulis dalam bahasa pemrograman *PHP* bertujuan untuk menangani administrasi *MySQL* melalui *web*. *phpMyAdmin* mendukung berbagai operasi pada *MySQL* dan *MariaDB*. Operasi-operasi yang sering digunakan seperti mengelola *database*, tabel, kolom, relasi, indeks, *users*, *permissions*, dan lain-lain, dapat dilakukan melalui antarmuka pengguna dengan tetap dapat mengeksekusi pernyataan *SQL* secara langsung. (www.phpmyadmin.net/).

2.3.3 *PHP: Hypertext Preprocessor* (*PHP*)

Gambar 2.6 Logo *PHP*
(Sumber: https://www.php.net/download-logos.php)

Menurut Winarno dan Zaki (2014: 49) *PHP Hypertext Prepocessor* (*PHP*) adalah bahasa script yang sangat cocok untuk pengembangan web dan dapat dimasukkan kedalam teks *HTML*.

PHP awalnya dikembangkan oleh programmer bernama Ramus Lerdorf pada tahun 1995, namun semenjak itu selalu dikembangkan oleh kelompok indenpenden yang di sebut Group *PHP* dan kelompok ini juga yang mendefinisikan standart *de facto* untuk *PHP* karena tidak ada spesifikasi normal. Saat ini pengembangannya dipimpin oleh duo maut, Andi Gutmans dan Zeev Suraski.

Yang menyebabkan *PHP* dipakai banyak orang adalah karena *PHP* adalah perangkat lunak bebas (*open source*) yang dirilis di bawah lisensi *PHP*. Artinya untuk menggunakan bahasa pemograman ini gratis, bebas dan terbuka. Untuk web, *PHP* adalah bahasa scripting yang bisa dipakai untuk tujuan apapun. Diantaranya cocok untuk pengembangan aplikasi web berbasis server (*server-side*) dimana *PHP* nantinya dijalankan di server web.

Setiap kode *PHP* akan dieksekusi oleh *runtime PHP,* hasilnya adalah kode *PHP* yang dinamis tergantung kepada script *PHP* yang dituliskan. *PHP* dapat digunakan dibanyak server web, system operasi dan *platform.* Selain itu dan digunakan juga di sitem manajemen *database relasional (RDBMS).* Semuanya ini bisa diperoleh gratis dan group *PHP* menyediakan kode sumber lengkap bagi pengguna untuk membangun, menyesuaikan dan mengutak-atik sesuai fungsi yang mereka inginkan.

2.3.4 *HTML* (*Hyper Text Markup Language*)

Gambar 2.7 Logo *HTML*
(Sumber: https://www.w3.org/html/logo/downloads/HTML5_Logo_512.png)

HTML adalah bahasa pemrograman yang digunakan untuk membuat dokumen *HTML* atau dikenal sebagai *web page*. Dokumen *HTML* merupakan file teks murni yang dapat dibuat menggunakan *editor* teks apapun. Dokumen *HTML* disajikan dalam *browser web surfer* seperti *Mozzila*, *Google Chrome*, *Internet Explorer*, dan sebagainya. Dokumen ini biasanya berisi informasi atau *interface* aplikasi di dalam internet (Sidik dan Pohan, 2009: 9).

Elemen merupakan istilah bagi komponen-komponen dasar pembentuk dokumen *HTML*. *Tag HTML* digunakan untuk menandai berbagai elemen dalam suatu dokumen *HTML*. Elemen dasar *HTML* yang dibutuhkan untuk membuat suatu dokumen *HTML* ditandai dengan *tag* <html>, <head>, dan <body> berikut *tag-tag* pasangannya yaitu </html>, </head>, dan </body>. Setiap dokumen *HTML* harus mempunyai pola dasar sebagai berikut (Sidik dan Pohan, 2009: 10-12):

<html>
<head>
Berisi informasi tentang head dokumen *HTML*
</head>
<body>
Berisi informasi yang ditampilkan dalam *browser web*
</body>
</html>

1. Setiap dokumen *HTML* harus diawali dengan menuliskan *tag* <html> dan *tag* </html> di akhir dokumen. *Tag* ini menandai elemen *html* yang berarti dokumen ini adalah dokumen *HTML*.
2. Elemen *head* ditandai dengan *tag* <head> dan *tag* </head>. Elemen ini berisi informasi tentang dokumen *HTML*. Minimal informasi yang dituliskan dalam elemen ini adalah judul dokumen yang ditandai dengan menggunakan *tag* <title> dan diakhiri dengan *tag* </title>.
3. Elemen *body* ditandai dengan *tag* <body> dan diakhiri dengan *tag* </body>. Elemen ini merupakan elemen terbesar di dalam dokumen *HTML*. Elemen ini mengandung isi dokumen yang akan ditampilkan pada *browser* yang meliputi paragraf, grafik, *link*, tabel, dan sebagainya.

2.3.5 *CSS* (*Cascading Style Sheet*)

Menurut Sidik dan Pohan (2009: 132) *CSS* merupakan fitur baru dari *HTML 4.0*. Hal ini diperlukan setelah melihat perkembangan *HTML* menjadi kurang praktis karena *web pages* terlalu banyak dibebani hal-hal yang berkaitan dengan faktor tampilan seperti *font*, dan lain-lain. Bentuk penggunaan *CSS* dapat dimodelkan sebagai berikut :

Selector {*property*: *value*}

Selector merupakan elemen yang akan didefinisikan, *property* adalah *atribute* yang akan diubah, dan *value* adalah nilai yang akan diberikan. Jika nilai yang akan diberikan berupa kata-kata, gunakan tanda petik ganda sebelum dan sesudah *value* (”*value*”).

Gambar 2.8 Logo *CSS*
(Sumber: http://w3widgets.com/responsive-slider/img/css3.png)

Terdapat tiga cara pendefinisian dalam menggunakan *CSS*, yaitu (Sidik dan Pohan, 2009: 134-137):

1. *Style sheet external*

Pada teknik ini, *style sheet* didefinisikan di luar dokumen *HTML* dan disimpan dalam *file* berekstensi *css* (*.css). Dalam pendefinisian *external* tidak perlu lagi menggunakan *tag html* diawal dan akhir dokumen.

2. *Style sheet internal*

Style sheet didefinisikan secara *internal* biasanya karena *web page* tertentu bersifat sangat unik sehingga membutuhkan definisi terpisah dibandingkan dengan *web page* lainnya.

3. *Inline style sheet*

Style sheet inline hanya bisa digunakan pada lokasi yang sangat spesifik dimana *style sheet* ditempatkan. Kekurangan dari teknik ini adalah dokumen menjadi lebih besar karena *style* didefinisikan satu per satu.

2.3.6 *JavaScript dan jQuery*

Gambar 2.9 Logo *JavaScript*
(Sumber: http://www.w3devcampus.com/wp-content/uploads/logoAndOther/logo_JavaScript.png)

Menurut Sidik dan Pohan (2009: 267) *JavaScript* merupakan modifikasi dari bahasa *C++* dengan pola penulisan yang lebih sederhana. Beberapa hal penting dalam *JavaScript* adalah:

1. Menggunakan blok awal "{" dan blok akhir "}".
2. *Automatic conversion* dalam pengoperasian tipe data yang berbeda.
3. *Sensitive case*, yaitu membedakan antara huruf kecil dan huruf capital sehingga harus berhati-hati dalam menggunakan nama variabel , fungsi, dan lain-lain.
4. *Extension* umumnya menggunakan "*.js".
5. Setiap *statement* dapat diakhiri tanda baca *semi colon* (;) dapat juga tidak.
6. Jika tidak didukung oleh *browser* versi lama, *script* dapat disembunyikan diantara *tag* "<!--" dan "-->".
7. Jika program dalam satu baris terlalu panjang dapat disambung ke baris berikutnya menggunakan karakter *backslash* (\).

Gambar 2.10 Logo *jQuery*
(Sumber: http://precision-software.com/wp-content/uploads/2014/04/jQurery.gif)

jQuery adalah sebuah *framework* berbasiskan *JavaScript*, yang berisi kumpulan kode atau fungsi *JavaScript* yang siap digunakan sehingga mempermudah dalam membuat kode *JavaScript*. Beberapa kemampuan yang dimiliki *jQuery* antara lain (Warman dan Zahni, 2013: 33):

1. Memanipulasi elemen *HTML*.
2. Memanipulasi *CSS*.
3. Penanganan *event* pada *HTML.*
4. Efek-efek *JavaScript* dan animasi.

2.3.7 *MySQL* dan *SQL*

Menurut Winarno dan Zaki (2014: 102) *MySQL* adalah *RDBMS* (*Relational Database Management System*) sebuah software database. Database merupakan sebuah tempat untuk menyimpan data yang jenisnya beraneka ragam. *MySQL* merupakan tipe data *relasional* yang artinya *MySQL* menyimpan datanya dalam bentuk table-tabel yang saling berhubungan.

Keuntungan menyimpan data di database adalah kemudahannya dalam menyimpan dan menampilkan data karena dalam bentuk table. Untuk melakukan pengelolahan terhadap table anda dapat menggukan perintah *SQL.*

Ada beberapa keuntungan dari *MySQL* sebagai berikut:

1. Garis dan *open source.*
2. Ada versi komersilnya, digunakan jika ingin memberikan dukungan teknis.
3. Biaya yang harus dikeluarkan jauh lebih murah dibandingkan merek lainnya.
4. Tersedia dibanyak *platform.*
5. Menggunakan standar penulisan SQL ANSI.

Gambar 2.11 Logo *MySQL*
(Sumber: https://www.mysql.com/about/legal/logos.html)

Terdapat 4 teknik pengaksesan data pada *DBMS* menggunakan *SQL*, antara lain (A.S. dan Shalahuddin, 2013: 47-48):

1. Memasukkan data (*insert*)

Bentuk *query* (perintah) untuk memasukkan data dapat dimodelkan sebagai berikut:

```
INSERT INTO nama_tabel(kolom1, kolom2, ...,kolomN)
VALUES (‘isikolom1’, ‘isikolom1’, …, ‘isikolomN’);
```

2. Mengubah data (*update*)

Bentuk *query* untuk mengubah data dapat dimodelkan sebagai berikut:

```
UPDATE nama_tabel
SET
        kolom1 = ‘isikolom1’
WHERE
        kolom2 = ‘isikolom2’;
```

3. Menghapus data (*delete*)

Bentuk *query* untuk menghapus data dapat dimodelkan sebagai berikut:

```
DELETE FROM nama_tabel
WHERE
        kolom1 = ‘isikolom1’;
```

4. Menampilkan data (*select*)

Bentuk *query* untuk menampilkan data dapat dimodelkan sebagai berikut:

```
SELECT kolom1, kolom2
FROM nama_tabel
WHERE
        kolom1 = ‘isikolom1’;
```

2.3.8 *Notepad*++

Gambar 2.12 Logo *Notepad*++
(Sumber: https://www.gavick.com/blog/wp-content/uploads/2010/07/notepadd.jpg)

Menurut Gilmore (2010: 36) *Notepad*++ merupakan editor teks *open source* yang matang dan diakui sebagai pengganti *Notepad. Notepad*++ tersedia untuk platform *Windows* yang dapat digunakan untuk menulis kode dengan beberapa pilihan bahasa (pemrograman). *Notepad*++ menawarkan beragam kenyamanan fitur yang diharapkan dari setiap kemampuan *IDE* (*Integrated Development Environment*), termasuk kemampuan untuk menunjukkan baris tertentu dari suatu dokumen sebagai referensi yang mudah; sintaks, tanda kurung, *indentation highlighting*, fasilitas pencarian yang tangguh, *macro recording* untuk tugas-tugas seperti memasukkan *template* komentar, dan sebagainya. Salah satu kelebihan *Notepad*++ adalah dukungan dasar untuk *auto-completion* dari nama fungsi yang ditawarkan sehingga akan mengurangi beberapa proses pengetikan kode.

2.3.9 *StarUML*

Salah satu pemodelan yang saat ini paling banyak digunakan adalah *UML* (*Unified Modeling Language*). *UML* adalah salah satu standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement*, membuat analisis dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek (A.S. dan Shalahuddin, 2013: 133).

Menurut A.S. dan Shalahuddin (2013: 137-138) *UML* muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun dan dokumentasi dari sistem perangkat lunak. *UML* tidak terbatas pada metodologi pemrograman tertentu, meskipun pada kenyataanya *UML* paling banyak digunakan pada metodologi berorientasi objek.

Gambar 2.13 Logo *StarUML*
(Sumber: http://staruml.sourceforge.net/image/staruml-logo.jpg)

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*.

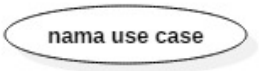
Menurut A.S. dan Shalahudin (2013: 18) *use case* dan *sequence diagram* merupakan bagian dari desain sistem. Dalam penelitian ini, diagram yang akan digunakan untuk desain sistem yaitu:

1. *Use case diagram*

Use case diagram merupakan pemodelan untuk menggambarkan kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah

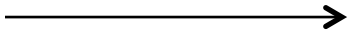
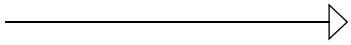
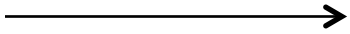
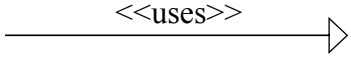
interaksi antara satu sistem atau lebih aktor dengan sistem informasi yang akan dibuat. *Use case diagram* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi-fungsi itu. Ada 2 hal utama yang terdapat pada *use case* yaitu aktor dan *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*
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
asosiasi/*association* ————	Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor

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

Lanjutan Tabel 2.3

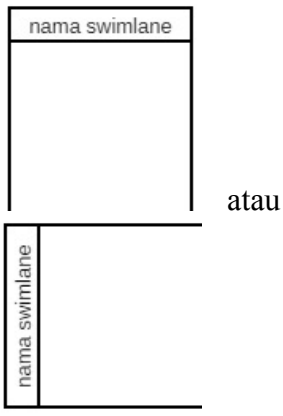
Simbol	Deskripsi
Ekstensi/*extend* <<extend>> 	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walaupun tanpa *use case* tambahan itu. Arah panah mengarah pada *use case* yang ditambahkan.
generalisasi/*generalization* 	Hubungan generalisasi dan 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)
Menggunakan/*include*/*uses* <<include>>  <<uses>> 	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankannya *use case* ini. Arah panah mengarah pada *use case* yang ditambahkan

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

2. *Activity diagram*

Activity 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 	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktifitas yang terjadi

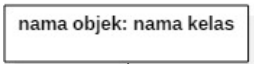
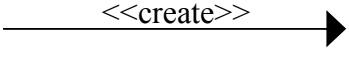
Sumber: A.S. dan Shalahuddin (2013: 162-163)

3. *sequence diagram*

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup (*life cycle*) objek dan *message* (pesan) yang

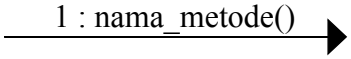
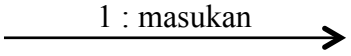
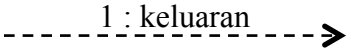
dikirimkan dan diterima antar objek. Jumlah *sequence diagram* yang harus digambar minimal sebanyak pendefinisian *use case* yang memiliki proses sendiri. Semakin banyak *use case* yang didefinisikan semakin banyak pula *sequence diagram* yang harus dibuat. Simbol-simbol yang digunakan pada *sequence 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
Pesan tipe *create*  	Menyatakan suatu objek membuat objek yang lain. Arah panah mengarah pada objek yang dibuat

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

Lanjutan Tabel 2.5

Simbol	Deskripsi
pesan tipe *call* 	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* 	Menyatakan bahwa suatu objek mengirimkan data/masukan/informasi ke objek lainnya. Arah panah mengarah pada objek yang dituju
pesan tipe *return* 	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)

2.4 Penelitian Terdahulu

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

Katen Lumbanbatu, Novriyeni (2014), Sistem pakar mendiagnosa gizi buruk pada balita dengan metode *fuzzy logic* (Studi kasus di rsud. Dr. Rm. Djoelham binjai), Pemenuhan kebutuhan gizi merupakan hal yang sangat penting dalam menjaga kondisi tubuh agar tetap sehat sering diabaikan, pada hal ini yang

sering membuat sejumlah anak tidak mendapatkan gizi yang baik dalam kesehatannya. Sedangkan dalam tubuh yang sehat dapat dinilai dari terpenuhinya kebutuhan gizi. Penilaian gizi dengan menggunakan metode logika fuzzy dengan menghitung nilai derajat keanggotaan sehingga menghasilkan diagnosa gizi buruk yang lebih akurat. Tahap pengembangan yang digunakan dalam peneliian ini, menggunakan model sequensial Trapesium yang diawali dengan tahap analisis sistem yaitu analisis depeneliian kebutuhan sistem, pembuatan diagram alir, data flow diagram, entity relationship diagram, dan tahap perancangan sistem yang meliputi spesifikasi proses, perancangan mapping tabel dan perancanggan menu antarmuka. Setelah tahap perancangan selesai maka dilanjutkan tahap implementasi dengan Pemrograman Visual bassic sebagai tool merancang desain sistem dan MySQL sebagai database. Hasil penelitian ini berupa aplikasi sistem pakar mendiagnosa gizi buruk pada balita di RSUD.Dr.R.M Djoelham Binjai. Aplikasi ini dapat dijadikan pengganti seorang pakar yang mana menghasilakan output berupa informasi diagnosa gizi buruk pada balita, yaitu: gizi baik, gizi kurang dan gizi buruk informasi tersebut dapat dijadikan acuan untuk mendapatkan diagnosa gizi yang baik berdasarkan saran yang ada pada aplikasi.

Dwi Aryanto, Ardi Pujiyanta (2013), Aplikasi sistem pakar penentuan asupan makanan bagi penderita penyakit gizi buruk dengan inferensi fuzzy, Asupan(konsumsi) makanan merupakan faktor utama untuk memenuhi kebutuhan gizi yang selanjutnya bertindak menyediakan energi bagi tubuh, mengatur proses metabolisme, memperbaiki jaringan tubuh serta untuk pertumbuhan. Tingkat konsumsi lebih banyak ditentukan oleh kualitas dan kuantitas pangan yang

dikonsumsi untuk mencapai keadaan gizi yang baik. Apabila kekurangan zat gizi khususnya energi dan protein menyebabkan berat badan menurun yang disertai produktivitas kerja, apabila kekurangan zat gizi berlanjut menyebabkan status gizi kurang dan gizi buruk. Gizi buruk atau malnutrisi dapat diartikan sebagai asupan gizi yang buruk. Hal ini bias diakibatkan oleh kurangnya asupan makanan, pemilihan jenis makanan yang tidak tepat ataupun karena sebab lain seperti adanya penyakit infeksi yang menyebabkan kurang terserapnya nutrisi dari makanan. Secara klinis gizi buruk ditandai dengan asupan protein, energi dan nutrisi mikro seperti vitamin yang tidak mencukupi ataupun berlebih sehingga menyebabkan terjadinya gangguan gizi. Penyakit gizi buruk terdiri atas tiga jenis penyakit menurut gejala klinisnya yaitu : gizi buruk marasmus, gizi buruk kwasiorkor, dan marasmus-kwasiorkor. Penanganan yang biasa dilakukan seorang pakar yaitu mengusulkan untuk pengaturan pola makan, termasuk jenis dan jumlah makanan. Maka dari itu penelitian ini bertujuan untuk membangun sebuah system pakar yang dapat digunakan untuk menentukan gangguan gizi dan jenis penyakit gizi buruk serta solusi asupan makananya. Subjek pada penelitian ini adalah “Aplikasi sistem pakar untuk penentuan asupan makanan bagi penderita penyakit gizi buruk”. Pada penelitian ini penelusuran faktanya menggunakan forward chaining dan logika yang digunakan adalah system inferensi fuzzy metode Tsukamoto. Tahap pengembangan aplikasi diawali dengan analisis data, perancangan system, pengkodean (Coding) dengan menggunakan Visual Basic 6.0 dan Testing (pengujian system dengan Black BoxTest dan Alfa Test). Dari hasil penelitian ini dihasilkan perangkat lunak yang mampu menentukan

makananya. Berdasarkan pengujian yang telah dilakukan terhadap responden status gangguan gizi dan jenis penyakit gizi buruk beserta solusi asupan telah layak untuk digunakan.

Helson Mandala Putra, dkk (2016), Implementasi metode naïve bayes classifier dalam sistem pakar defisiensi nutrisi pada balita, Defisiensi nutrisi merupakan salah satu masalah yang dialami oleh balita di Indonesia. Masalah ini timbul karena kurangnya pengetahuan dari orang tua tentang zat gizi atau nutrisi yang dibutuhkan oleh balita. Melihat masalah ini, penulis membuat sebuah sistem yang mampu mendiagnosis defisiensi nutrisi pada balita, serta memberikan pengetahuan kepada masyarakat tentang defisiensi nutrisi, agar dapat dilakukan pencegahan sedinin mungkin. Sistem yang dibuat adalah sistem pakar defisiensi nutrisi pada balita menggunakan metode Naïve Bayes Classifier. Metode ini merupakan pengembangan dari Teorema Bayes, dimana nilai probabilitas dari sebuah objek diklasifikasikan berdasarkan syarat tertentu. Pada beberapa penelitian, disebutkan bahwa Naïve Bayes Classifier merupakan metode klasifikasi yang lebih baik, sehingga penulis menggunakan Naïve Bayes Classifier. Dalam sistem ini, penulis menentukan beberapa gejala defisiensi nutrisi berdasarkan tabel antropometri dengan memasukkan data tinggi badan, berat badan, dan umur. Kemudian memilih gejala tambahan untuk mempersempit hasil klasifikasinya. Setelah melakukan pengujian terhadap 20 data pasien, maka diperoleh nilai akurasi sistem sebesar 90%. Hal ini membuktikan bahwa sistem mampu mengklasifikasikan jenis kekurangan nutrisi dengan cukup baik, dan bisa diterapkan pada semua puskesmas dan rumah sakit.

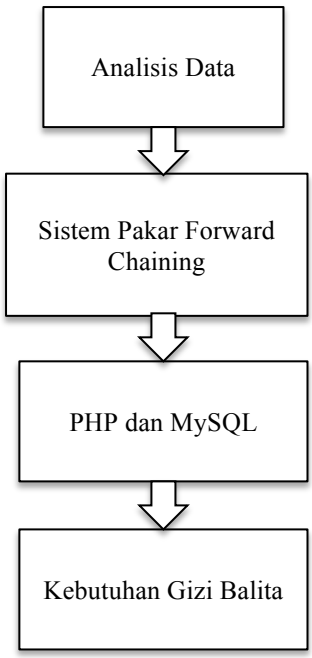
Suhati Novalia Rengganis (2015), Perancangan Sistem Pakar Klasifikasi Status Gizi Balita Berdasarkan Indeks Antropometri Berat Badan Terhadap Umur (BB/U) menggunakan Metode Forward Chaining, Status gizi merupakan ekspresi keadaan keseimbangan gizi seseorang dalam bentuk variable tertentu. Status gizi balita dapat diukur secara langsung menggunakan antropometri. Dengan perkembangan teknologi, mulai dikembangkan sistem pakar dengan berbagai metode, salah satunya forward chaining. Dalam penelitian ini, data masukan yang diperlukan diantaranya jenis kelamin, umur, serta berat badan balita. Sampel yang digunakan pada penelitian ini sebanyak dua puluh data balita. Hasil dari penelitian ini berupa perangkat lunak berbasis web yang memberikan hasil berupa status gizi balita, dan dari hasil pengujian, didapatkan sebanyak tiga data yang berbeda antara hasil analisis manual dengan hasil analisis sistem pakar.

Mukhammad *Shaid*, **dkk** (2016) Sistem Pakar Pertumbuhan Balita Berbasis *Web* Dengan Metode Case Based Reasoning, Sistem Pakar adalah salah satu bagian dari Kecerdasan Buatan yang mengandung pengetahuan dan pengalaman yang dimasukkan oleh banyak pakar ke dalam suatu area pengetahuan tertentu, sehingga setiap orang dapat menggunakannya untuk memecahkan berbagai masalah yang bersifat spesifik, dalam hal ini adalah penentuan gerakan motorik pada Pertumbuhan Balita.Pertumbuhan balita bisa terjadi berdasarkan beberapa factor, yaitu berdasarkan kelahirannya dan pertumbuhan gizi yang dikonsumsi. Dengan memanfaatkan metode Case-base Reasoning, dapat dihasilkan suatu aplikasi untuk mengidentifikasi pertumbuhan balita. Dengan harapan sistem ini nantinya dapat digunakan sebagai sarana atau

sebagai pengetahuan dalam menjaga kestabilan pertumbuhan balita dan membantu anda untuk mendapatkan hasil yang optimal dalam menjaga pertumbuhan setiap balita. Metode Case Based Reasoning (CBR) digunakan dalam aplikasi Pertumbuhan Balita dengan menggunakan Perhitungan Nearest Neighbor, Dimana data kasus baru akan dibandingkan perhitungannya dengan data kasus lama yang ada di database, dan kemudian dihitung kriteria kemiripannya berdasarkan rumus atau ketentuan yang berlaku.

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 2.14 Kerangka Pemikiran
(Sumber: Data Penelitian: 2016)

Data-data yang dibutuhkan berkaitan dengan kebutuhan gizi balita 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 *PHP* dan *database MySQL* sehingga menghasilkan sebuah sistem pakar untuk mendeteksi kebutuhan gizi pada balita dengan metode forward chaining berbasis *web.*