BAB II
LANDASAN TEORI

2.1 Persyaratan Registrasi Dosen Baru (DIKTI)

Adapun persyaratan dari DIKTI untuk pengangkatan Dosen adalah sebagai berikut:

1. Diangkat sebagai Dosen Tetap maksimal berusia 58 tahun (Permenristekdikti No. 2 Tahun 2016)
2. Tidak berstatus sebagai Pegawai Tetap pada instansi lain, meliputi:
   a) PNS non-dosen (PNS Pemkot/Pemda, POLRI, TNI, PNS kementrian/lembaga negara selain PNS dosen)
   b) Guru tetap/tidak tetap
   c) Pegawai BUMN
   d) Anggota aktif partai politik dan legislatif (DPR/MPR/DPRD/DPD)
   e) Konsultan, pengacara, notaris, apoteker
3. Status kemahasiswaannya terdaftar di PD Dikti untuk lulusan setelah tahun 2002
4. KTP terbaru yang masih berlaku (berwarna/asli, bukan fotokopi)
5. Foto terbaru berwarna
6. Surat keterangan sehat dan bebas narkoba
7. Surat keputusan:
   a) Dosen Tetap dari ketua Yayasan/Ketua BPH yang memuat hak dan kewajiban antara calon dosen dan yayasan (bagi Dosen Tetap Yayasan)
   b) SK sebagai PNS/CPNS sebagai Dosen Tetap (bagi PNS Dpk)
8. Ijazah lengkap (mulai S-1/D4). Bagi lulusan PT luar negeri disertakan SK penyetaraan dari DIKTI/PTN yang ditunjuk DIKTI (LLDIKTI5, 2020).

2.2 Persyaratan Universitas Teknologi Digital Indonesia

UTDI sebagai perekrut memiliki beberapa kriteria di antaranya (Wawancara dengan Widyastuti Andriyani, 2020):

1. Mahasiswa kandidat harus memiliki paper di setiap semester
2. IPK tiap semester minimal 3.00

3. Pra Tesis tepat waktu
4. Tesis tepat waktu
5. Tidak ada matakuliah dengan nilai C
6. Bagaimana kerjasama saat studi
7. kedisiplinan guna meningkatkan nilai borang untuk menunjang akreditasi
8. Komunikasi
9. Berapa lama studi

2.3 *Rule Based System*

Rule jika diterjemahkan secara literatur berarti aturan. Banyak sekali aturan-aturan yang dapat kita temukan di sekitar kita, misalnya hukum, aturan penggunaan peralatan listrik, dan lain sebaGainya. Adanya aturan adalah untuk membatasi apa saja yang dilakukan oleh manusia. *Rule-based system* (RBS) digunakan sebagai cara untuk menyimpan dan memanipulasi pengetahuan untuk menafsirkan informasi dengan cara yang berguna untuk memecahkan masalah dengan aturan yang dibuat berdasarkan pengetahuan dari pakar. Jadi, di sini membuat sistem dengan aturan yang dibuat berdasarkan pengetahuan dari pakar untuk menangkap keahlian manusia dan pengambilan keputusan. Aturan tersebut memiliki kondisi (*if*) dan tindakan (*then*) (Grosan dan Abrahama, 2011). Peraturan-peraturan tersebut akan dimasukkan ke dalam mesin aplikasi untuk menyaring data layaknya ayakan pasir. Data yang cocok akan terus diproses hingga aturan terakhir. *Rule Based* semacam ini disebut *Rule Based Forward Chaining.*

Rule Based System memiliki tiga elemen (Grosan dan Abrahama, 2011), yaitu:

1. Kumpulan data dan fakta

   Kumpulan fakta di sini sebagai acuan dasar dan acuan pengetahuan yang kemudian akan diproses menjadi aturan.
2. Kumpulan *rule*/aturan

   Fakta-fakta yang sudah terkumpul akan dirangkai akan menjadi satu aturan yang bisa dimengerti mesin.

3. Kriteria untuk mengakhiri

   Kriteria di sini adalah sebuah kondisi yang dijadikan acuan pada sistem. Sehingga nantinya dapat ditentukan apakah sistem akan berhenti atau justru terus melakukan *looping*.

Tabel 2. 1 Contoh Transformasi Data Menjadi Rule

DATA	**CONDITIONS**	**RULES**
Season Winter temperature wind blushing road weather	<0, >0 strongly, gently slippery, not slippery cold, warm, hot	***Premises*** IF temperature < 0 AND IF wind blushing is strongly OR IF the road is slippery ***Conclusion*** THEN the weather is cold

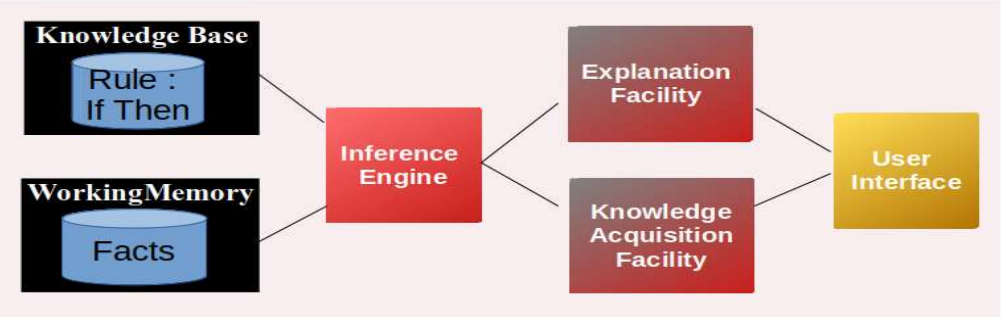
Keterangan:

1) Pada kolom data ditemukan sejumlah fakta dan data-data mengenai temperatur, kecepatan angin, kondisi jalan, dan cuaca.

2) Data-data yang tersedia kemudian diberikan sebuah kondisi yang mampu merujuk kepada sebuah konklusi.

3) Setelah selesai, data yang terkondisi tadi dimasukkan dalam bentuk *if-then,* di mana ***if*** berisikan premis (data + kondisi) dan ***Then*** berisikan konklusi.

Berikut ini adalah arsitektur dari Rule Based System (Devarapalli *et al*., 2013) yang dapat dilihat pada gambar 2.1



Gambar 2. 1 Arsitektur *Rule-Based System*

Keterangan:

1. *Knowledge Base:* menyimpan pengetahuan ahli sebagai aturan-tindakan-kondisiTerdapat aturan pengetahuan dari pakar atau ahli sebagai aturan yang berguna untuk pemecahan masalah. Pengetahuan direpresentasikan sebagai seperangkat aturan. Setiap aturan menentukan relasi, rekomendasi, arahan, strategi dan memiliki struktur IF(kondisi) dan THEN(tindakan)*, di mana* ***if*** *berisikan premis (data + kondisi) dan* ***Then*** *berisikan konklusi.* Ketika bagian kondisi dari suatu aturan terpenuh, maka bagian tindakan dieksekusi.
2. *Working Memory*: menyimpan fakta yang dihasilkan yang diturunkan oleh hasil inferensi (Abu-Nasser dan AbuNaser, 2018). Bagian dari system yang sedang berlangsung termasuk keputusan sementara. Berisi fakta-fakta dari masalah yang ditemukan dalam suatu proses (Henderi *et al.*, 2020).
3. *Inference Engine*: mencocokkan bagian kondisi aturan dengan fakta yang disimpan dalam memori kerja aturan dengan kondisi terpenuhi adalah aturan aktif dan ditempatkan dalam agenda, kemudian di antara aturan aktif dalam agenda, satu dipilih sebagai aturan berikutnya untuk dieksekusi.

4) Explanation Facility: memberikan pembenaran solusi kepada pengguna (reasoning chain)

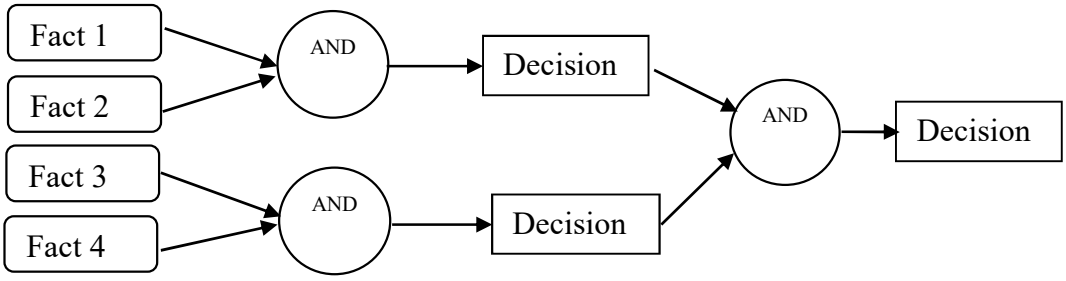
5) Knowledge Acrquisition Facility: membantu mengintegrasikan pengetahuan baru juga akuisisi pengetahuan otomatis.

6) User Interface (UI): memungkinkan pengguna untuk berinteraksi dengan Expert System, memasukkan fakta, presentasi solusi (Abu-Nasser dan AbuNaser, 2018).

2.4 *Forward Chaining*

Forward chaining adalah salah satu metode pada sistem pakar yang mencari solusi melalui masalah. Dengan kata lain, metode ini melakukan pertimbangan terhadap fakta-fakta yang kemudian berujung pada suatu kesimpulan berdasarkan fakta-fakta tersebut. Metode ini merupakan kebalikan dari metode *backward chaining* yang melakukan pencarian yang berawal dari hipotesis ke fakta-fakta untuk mendukung hipotesis tersebut.(Idris, Mustafid dan Suseno, 2019)

Dalam *forward chaining*, algoritma membandingkan data dengan kondisi setiap aturan dalam basis aturan, menggunakan pengurutan dari basis aturan itu. Jika informasi memberikan aturan yang diaktifkan, hasilnya ditempatkan di memori dan prosedur berlanjut ke aturan berikutnya.



Gambar 2. 2 Contoh *Forward Chaining*

2.5 Algoritma C4.5

Algoritma C4.5 merupakan salah satu algoritma yang digunakan melakukan proses klasifikasi data dengan membentuk pohon keputusan (*decision tree*) (Buulolo, Medan dan Utara, 2017). Dari fakta yang sangat besar, dengan metode pohon keputusan diubah menjadi pohon keputusan yang mewakili aturan. Algoritma C4.5 merupakan pengembangan dari algoritma ID3 dan menggunakan *descision tree* yang hampir sama. Proses pada pohon keputusan adalah mengubah bentuk data (tabel) menjadi model pohon, mengubah model pohon menjadi *rule*, dan menyederhanakan *rule* (Setio, Saputro dan Winarno, 2020). Pohon keputusan berguna untuk mengekspolari data, menemukan hubungan tersembunyi antara sejumlah calon variabel input dengan sebuah variabel target.

Berikut ini adalah beberapa kelebihan dari klasifikasi algoritma C4.5 atau pohon keputusan (Rahim *et al.*, 2018).

a) Hasil analisis diagram pohon yang mudah dipahami.
b) Mudah dibuat dan membutuhkan lebih sedikit data eksperimen dibandingkan dengan algoritma klasifikasi lainnya.
c) Modelnya dengan mudah diinterpretasikan dan diimplementasikan dengan nilai kontinu dan nilai diskrit (Siahaan *et al.*, 2019).
d) Model hasil dapat dengan mudah untuk dipahami.
e) Menggunaka teknik statistik yang dapat divalidasi.
f) Waktu komputasi relatif lebih cepat dibandingkan teknik klasifikasi lainnya.
g) Akurasi dapat menandingi teknik klasifikasi lainnya (Pah dan Utama, 2020).

Algoritma C4.5 adalah pengembangan dari algoritma ID3, oleh karena pengembangan tersebut, algoritma C4.5 mempunyai prinsip dasar kerja yang sama dengan ID3.

Menurut (Lakshmi et al., 2013), tahapan dari proses algoritma C4.5 diuraikan sebagai berikut.

(1) Mempersiapkan data latih.

(2) Menghitung nilai *Entropy*.

Entropy merupakan ukuran ketidakpastian, yakni perbedaan keputusan terhadap nilai atribut tertentu. Semakin tinggi nilai Entropy, semakin tinggi perbedaan keputusan (ketidakpastian). Menghitung total kasus, jumlah kasus setiap kelas, kemudian *Entropy* semua kasus dan kasus dibagi dengan nilai atribut. *Entropy* digunakan untuk menentukan seberapa informatif atribut input untuk menghasilkan atribut *output*. Nilai Entopy dihitung dengan rumus yang ditulis sebagai berikut.

$$Entropy\ (S) = \sum_{i=1}^{n} - pi * log_2 pi \text{p} \quad \text{..... (1)}$$

Keterangan:

S = himpunan kasus

A = fitur

n = jumlah partisi S

pi = proporsi dari S_i terhadap S (probabilitas yang diperoleh dari jumlah (ya) dibagi dengan total kasus)

(3) Menghitung nilai *Gain* atau menghitung perolehan informasi untuk setiap atribut. *Gain* merupakan salah satu langkah pemilihan atribut yang digunakan untuk memilih atribut setiap simpul pada pohon keputusan atau dengan kata lain *Gain* merupakan tingkat pengaruh suatu atribut terhadap keputusan atau ukuran efektifitas suatu variabel dalam mengklasifikasikan data. *Gain* dihitung dengan rumus yang ditulis sebagai berikut.

$$Gain(S,A) = Entropy(S) - \Sigma_{i=1}^{n} \frac{S_i}{S} * Entropy(S_i) \quad \text{.....(2)}$$

Keterangan:

S = himpunan kasus

A = atribut

n = jumlah partisi atribut A

$|S_i|$ = jumlah kasus pada atribut partisi ke-i

$|S|$ = jumlah kasus dalam S

Pada algoritma C4.5, nilai *Gain* digunakan untuk menentukan variabel mana yang menjadi node dari suatu pohon keputusan. Suatu variabel yang memiliki *Gain* tertinggi akan dijadikan node di pohon keputusan.

(4) Menghitung nilai *Split Info* untuk setiap atribut dengan rumus

$$SplitInfo(S,A) = -\Sigma_{i=1}^{n} \frac{S_i}{S} \log_2 \frac{S_i}{S} \quad \text{..... (3)}$$

Keterangan:

S = himpunan kasus

A = atribut

S_i = jumlah sampel untuk atribut i

Split Info digunakan sebagai pembagi dari *Gain(*A) yang akan menghasilkan *Gain Ratio*

(5) Menentukan nilai *Gain Ratio* dengan rumus yang ditulis sebagai

$$GainRatio(A) = \frac{Gain(S,A)}{SplitInfo(S,A)} \quad \text{..... (4)}$$

Keterangan:

S = himpunan kasus

A = atribut

Gain (S, A) = info *Gain* pada atribut A

Split (S, A) = *Split Info* pada atribut A

Gain Ratio merupakan salah satu ukuran lain yang digunakan untuk mengatasi masalah pada atribut yang memiliki nilai sangat bervariasi. *Gain Ratio* tertinggi dipilih sebagai atribut sebagai akar simpul.

(6) Mengulangi proses ke-2 hingga semua atribut digunakan atau memenuhi kondisi atas sampai semua cabang memiliki kelas yang sama (Asidik, Kusrini dan Henderi, 2018).

Berikut ini adalah *pseudocode* dari algoritma konstruksi pohon keputusan C4.5 (Ruggieri, 2002).

```
FormTree(T)
        (1) ComputeClassFrequency(T);
        (2) If OneClass or FewCases
                Return a leaf;
            Create a decision node N;
        (3) ForEach Attribute A
                ComputeGain (A);
        (4) N.test=AttributeWithBestGain;
        (5) If N.test is continuous
                find Threshold;
        (6) ForEach T' in the splitting of T
        (7) If T' is Empty
                Child of N is a leaf
            else
        (8)    Child of N = FormTree (T');
        (9) ComputeErrors oof N;
Return N;
```

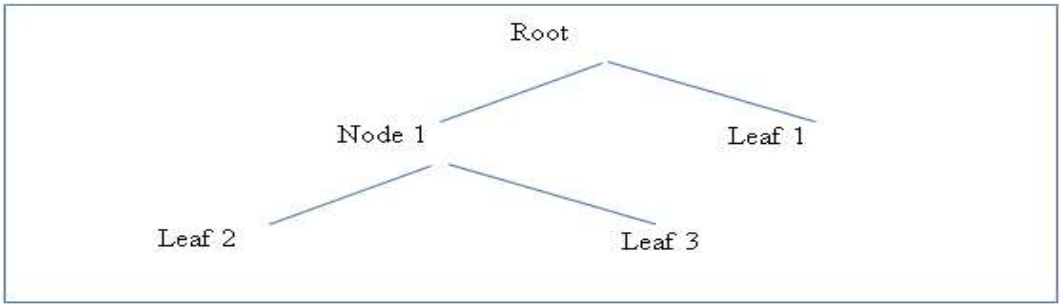
Merupakan *pseudocode* dari algoritma C4.5 yang berfungsi untuk pembentukan pohon keputusan. Perhitungan dimulai dari menghitung banyaknya jumlah atribut dan menentukan atribut mana yang akan digunakan sebagai akar dari pohon keputusan. Selanjutnya akan dilakukan perhitungan *Entropy*, *Gain*,

Split Info dan *Gain Ratio* untuk menentukan *leaf* dari pohon keputusan tersebut. Setelah semua perhitungan selesai dilakukan, pohon keputusan dapat dibentuk berdasarkan nilai *Gain Ratio* yang telah dihitung. Atribut dengan nilai *Gain Ratio* tertinggi akan terletak pada prioritas yang lebih tinggi dan memiliki kedudukan yang lebih tinggi juga pada pohon keputusan (Harryanto dan Hansun, 2017).

2.6 Pohon Keputusan

Pohon keputusan *(Decision Tree)* merupakan algoritma pengklasifikasian yang sering digunakan dan mempunyai struktur yang sederhana dan mudah untuk diimplementasikan, berbentuk menyerupai pohon terbalik, di mana simpul internal (bukan daun) menunjukkan pengujian pada atribut, setiap cabang sesuai dengan hasil pengujian, dan setiap simpul eksternal (daun) menunjukkan prediksi kelas. Pada setiap node, algoritma memilih atribut terbaik untuk mempartisi data ke dalam kelas-kelas individual. Ketika induksi pohon keputusan digunakan untuk pemilihan subset atribut, sebuah pohon disusun dari data yang diberikan. Semua atribut yang tidak muncul di pohon dianggap tidak relevan (Agarwal, 2014).

Berikut ini adalah contoh Pohon Keputusan.

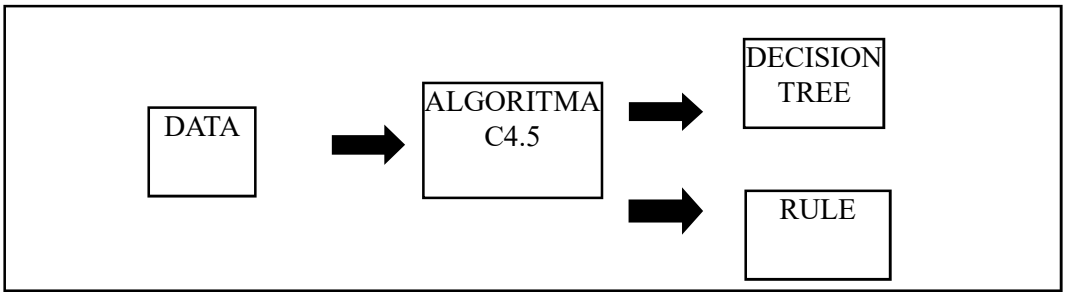


Gambar 2. 3 Contoh Pohon Keputusan

Cabang-cabang dari pohon keputusan merupakan pertanyaan klasifikasi, sedangkan untuk daun-daunnya merupakan kelas-kelas atau kelompoknya. Tujuan dari algoritma C4.5 adalah untuk melakukan klasifikasi sehingga hasil dari pengolahan dataset berupa pengelompokan data ke dalam kelas-kelas tertentu (Setio, Saputro dan Winarno, 2020). Kemampuan pohon keputusan adalah untuk membuat proses pengambilan keputusan yang kompleks menjadi lebih sederhana

sehingga pengambilan keputusan akan lebih menginterpretasikan solusi dari permasalahan.

Berikut proses pengolahan data menggunakan algoritma C4.5 untuk membentuk pohon keputusan dan *rule* (Rismayana dan Rosdiana, 2019).



Gambar 2. 4 Konsep Pohon Keputusan

Proses pada pohon keputusan adalah mengubah data dengan menggunakan algoritma C4.5 menjadi pohon keputusan, dan menjadi *rule* juga menyederhanakan *rule*.

2.7 *Confusion Matrix*

Untuk permasalahan dalam klasifikasi, pengukuran yang biasa digunakan adalah *precision*, *recall* dan *accuracy*. Nilai tersebut dapat dihitung dengan *confusion matrix*. *Confusion matrix* yang juga sering disebut *error matrix* adalah sebuah tabel yang terdiri atas banyaknya baris data uji yang diprediksi benar (*positive*) dan tidak benar (*negative*) oleh model klasifikasi (Idris, Mustafid dan Suseno, 2019). Berikut ini tabel *confusion matrix* dapat dilihat pada tabel 2.2.

Tabel 2. 2 *Confusion Matrix*

	Actually Positive (1)	*Actually Negative* (0)
Predicted Positive (1)	*True Positive* (TP)	*False Positive* (FP)
Predicted Negative (1)	*False Negative* FN)	True Negative (TN)

Confusion matrix yang berbentuk tabel matriks yang menggambarkan kinerja model klasifikassi pada serangkaian data uji yang nilai sebenarnya diketahui. Gambar di atas menunjukkan dengan 4 nilai kombinasi nilai prediksi dan nilai aktual yang berbeda.

Adapun terdapat 4 istilah sebagai representasi hasil proses pada *confusion matrix*, sebagai berikut.

True Positive (TP) merupakan data positif yang diprediksi benar.
True Negative (TN) merupakan data negatif yang diprediksi benar.
False Positive (TP) merupakan data negatif namun diprediksi sebagai data positif.
False Negative (TN) merupakan data positif namun diprediksi sebagai data negatif.

Berikut adalah beberapa manfaat dari *confusion matrix.*

1. Menunjukkan bagaimana model ketika membuat prediksi.
2. Tidak hanya memberi informasi tentang kesalahan yang dibuat oleh model, tetapi juga jenis kesalahan yang dibuat.
3. Setiap kolom dari *confusion matrix* merepresentasikan *instance* dari kelas prediksi.
4. Setiap baris dari *confusion matrix* mewakili *instance* dari kelas aktual.

Adapun beberapa *performance metrics* populer yang umum dan sering digunakan adalah:

Accuracy menggambarkan seberapa akurat model dalam mengklasifikasikan dengan benar.

$$accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (5)$$

Precision menggambarkan tingkat keakuratan antara data yang diminta dengan hasil yang diberikan oleh model.

$$precision = \frac{TP}{TP+FP} \quad (6)$$

Recall menggambarkan keberhasilan model dalam menemukan kembali sebuah informasi.

$$recall = \frac{TP}{TP+FN} \qquad (7)$$