BAB II
TINJAUAN PUSTAKA

Pada bab ini, akan dijelaskan tentang metode-metode yang akan digunakan dalam penelitian serta perangkat apa saja yang akan membantu dalam penelitian ini. Serta akan dibahas mengenai teori-teori yang akan digunakan untuk mendukung penelitian tentang sistem diagnosis penyakit covid-19 dengan menggunakan algoritma c5.0 yang menerapkan metode pruning didalamnya.

2.1. Penelitian Pendukung

Dasar acuan atau penelitian terdahulu merupakan sesuatu yang penting dalam sebuah penelitian yang dapat dijadikan sebagai data pendukung. Salah satu data pendukung yang dianggap perlu dijadikan bagian tersendiri adalah penelitian terdahulu yang relevan dengan permasalahan yang dibahas dalam penelitian ini. Sebagai referensi penulis, digunakan jurnal-jurnal dari penelitian terdahulu sebagai bahan acuan dalam proses pengerjaan tugas akhir ini, diantaranya adalah sebagai berikut.

Penelitian yang pertama, dalam jurnal yang ditulis oleh (Daniel, Memi, & Yuki, 2019), yang berjudul "Klasifikasi Lama Studi Mahasiswa Menggunakan Metode Algoritma C5.0 pada Studi Kasus Data Kelulusan Mahasiswa Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Mulawarman Tahun 2017" membahas tentang penelitian untuk mengetahui hasil klasifikasi lama studi kelulusan seluruh mahasiswa Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Mulawarman pada tahun 2017 dengan menggunakan algoritma C5.0. Berdasarkan hasil analisis dan pembahasan yang dilakukan, diperoleh kesimpulan bahwa hasil ketepatan klasifikasi pada metode algoritma C5.0, diperoleh nilai presentase ketepatan sebesar 84,21%. Kemudian kesalahan klasifikasi (APER) yang dihasilkan adalah sebesar 15,79%. Dengan demikian metode algoritma C5.0 merupakan metode yang baik dalam pengklasifikasian data masa studi kelulusan seluruh mahasiswa FMIPA UNMUL tahun 2017.

Penelitian kedua yang akan digunakan sebagai pendukung pada penelitian ini terdapat dalam jurnal yang ditulis oleh (Harani & Damayanti, 2021), yang

berjudul “Implementasi Algortima C5.0 Untuk Menentukan Pelanggan Potensial Di Kantor Pos Cimahi” didalam penelitiannya membahas tentang perancangan sebuah sistem untuk mengolah data penjualan korporat di Kantor Pos Cimahi. Sistem yang dibuat bertujuan agar dapat mengetahui pelanggan mana yang potensial dan yang mana yang tidak potensial. Dalam implementasi algoritma nya, digunakan data sebanyak 70% sebagai data latih dan 30% sebagai data uji dari total keseluruhan data. Hasil akurasi yang diperoleh dari sistem yang sudah dirancang mendapatkan nilai sebesar 96%.

Penelitian pendukung ketiga yaitu jurnal yang ditulis oleh (Nabillah, Yufis, & Gita, 2020), yang berjudul “Implementasi Algoritma C5.0 Untuk Menganalisa Gejala Prioritas Pada Anak yang Mengalami Bullying” membahas tentang penerapan algoritma C5.0 untuk menganalisa gejala prioritas pada anak yang mengalami bullying. Pada penelitian ini akurasi yang diperoleh dengan pengujian menggunakan algoritma C5.0 sebelum menggunakan fitur seleksi sebesar 92,77% dan setelah menggunakan fitur pilihan 93,33%.

Penelitian yang keempat terdapat dalam jurnal yang ditulis oleh (Kastawan, Wiharta, & Sudarma, 2018) yang berjudul “Implementasi Algoritma C5.0 pada Penilaian Kinerja Pegawai Negeri Sipil” dalam penelitian ini membahas tentang penerapan algoritma C5.0 untuk melakukan analisa klasifikasi terhadap data penilaian kinerja pegawai negeri sipil. Pada penelitian ini, pohon keputusan algoritma C5.0 memiliki akurasi sebesar 96,08% yang menggunakan data training sebanyak 184 data.

2.2. *Decision Tree*

Pohon keputusan merupakan salah satu teknik klasifikasi data yang menggunakan struktur pohon atau struktur hirarki. Konsepnya adalah dengan merubah data menjadi suatu bentuk pohon keputusan atau aturan-aturan keputusan. *decision tree* menggunakan metode *supervised machine learning* yaitu proses pembelajaran dimana data yang baru masuk akan diklasifikasikan berdasarkan *samples data* yang sudah ada (Wijaya, Bahtiar, Kaslani, & R, 2021). *Decission tree* mirip dengan pengambilan keputusan yang dilakukan manusia dan sangat mudah dipahami. Pengambilan keputusannya dapat mengatasi masalah

pada data diskrit maupun data *continuous* (Patel & Prajapati, 2018). *Decision tree* bekerja dengan cara membandingkan nilai-nilai atribut dari node internal kemudian menghasilkan kesimpulan dari *leaf node* pada pohon keputusan. Jadi, pengambilan keputusan dimulai dari *root node* dari pohon keputusan yang sudah terbentuk aturan-aturannya yang berjalan menuju *leaf node* sampai membentuk suatu kesimpulan (Dai, Zhang, & Wu, 2016).

2.3. Algoritma C5.0

Algoritma C5.0 adalah salah satu algoritma klasifikasi yang khsusunya diterapkan pada teknik decision tree. C5.0 merupakan penyempurnaan dari algoritma sebelumnya yang dibentuk oleh Ross Quinlan pada tahun 1987, yaitu ID3 dan C4.5. Dalam memilih atribut untuk pemecah objek dalam beberapa kelas harus dipilih atribut yang menghasilkan *gain ratio* paling besar. Atribut yang menghasilkan *gain ratio* paling besar tersebut akan dipilih sebagai *parent* pada *node* selanjutnya. Terdapat beberapa langkah yang harus dilakukan untuk mendapat nilai *gain ratio* diantaranya adalah menghitung nilai *entropy* dan *information gain*. Untuk menghitung nilai *entropy* dari keseluruhan sampel digunakan rumus :

$$Entropy\ (S) = \sum_{i=1}^{n} - p_i * \log_2(p_i) \qquad (2.\ 1)$$

Keterangan :

S = himpunan kasus

n = jumlah sampel

P_i = proporsi kelas

Sedangkan, untuk melakukan perhitungan *information gain* digunakan rumus berikut,

$$Gain\ (S, A) = Entropy(S) - \sum_{\mathrm{i=1}}^{\mathrm{n}} \frac{|Si|}{|S|} * Entropy(A) \qquad (2.\ 2)$$

Keterangan :

A = atribut

S = himpunan kasus

|Si| = jumlah kasus pada partisi ke-i

|S| = Jumlah kasus yang ada di S

Setelah hasil sudah didapat dari perhitungan *Entropy* dan *Information Gain*. Langkah terakhir sebelum menentukan atribut yang akan dipilih yaitu menghitung nilai *Gain Ratio* dari masing-masing atribut. Adapun, rumus dalam memperoleh nilai *Gain Ratio* adalah sebagai berikut,

$$Gain\ Ratio = \frac{Gain\ (S, A)}{\sum_{i=1}^{n} Entropy\ (Si)} \quad (2.\ 3)$$

Gain (S,A) = nilai gain dari *variable*

Entropy(Si) = banyaknya nilai *entropy* dalam suatu *variable*

Perhitungan dari *entropy* hingga *information gain* merupakan bagian dari C4.5, yang berbeda dari C5.0 adalah terdapat proses *boosting* yang nantinya akan digunakan untuk menentukan kelas berdasarkan hasil perhitungan kombinasi dari beberapa *tree* (Dyah, Imam, & Sutrisno, 2019). Agar dapat mempermudah dalam penjelasan teori algoritma C5.0 dalam penelitian ini, berikut akan ditunjukan contoh bagaimana proses pengolahan data pada algoritma C5.0 dengan menggunakan data pendapatan masyarakat dalam menentukan kelayakan seseorang untuk mendapatkan BLT (Bantuan Langsung Tunai).

Tabel 2.1 Contoh Data Pendapatan Masyarakat

Umur	Pendidikan	Pekerjaan	Pemilikan Rumah	Jumlah Anggota Keluarga	Pendapatan	Keputusan
Muda	SMA	Penjahit	Numpang	Banyak	Rendah	Layak
Muda	SMP	Penjahit	Kontrak	Normal	Menengah	Tidak Layak
Tua	SMP	Supir	Milik Pribadi	Banyak	Menengah	Layak
Tua	SMA	Penjahit	Numpang	Banyak	Menengah	Layak
Muda	SD	Supir	Numpang	Normal	Menengah	Tidak Layak
Tua	SMP	Supir	Kontrak	Banyak	Menengah	Layak
Tua	SMA	Penjahit	Milik Pribadi	Banyak	Rendah	Layak
Muda	SMP	Supir	Numpang	Banyak	Rendah	Layak
Muda	SD	Supir	Numpang	Normal	Menengah	Tidak Layak
Muda	SD	Pedagang	Numpang	Normal	Menengah	Layak
Muda	SMP	Supir	Kontrak	Normal	Rendah	Tidak Layak
Muda	SMP	Penjahit	Milik Pribadi	Banyak	Menengah	Layak

Umur	Pendidikan	Pekerjaan	Pemilikan Rumah	Jumlah Anggota Keluarga	Pendapatan	Keputusan
Muda	SMA	Penjahit	Milik Pribadi	Normal	Rendah	Layak
Tua	SMA	Pedagang	Numpang	Banyak	Rendah	Layak
Tua	SMP	Supir	Kontrak	Normal	Menengah	Tidak Layak
Muda	SMP	Supir	Numpang	Normal	Menengah	Tidak Layak
Muda	SMA	Penjahit	Numpang	Normal	Menengah	Layak
Muda	SMA	Penjahit	Kontrak	Normal	Menengah	Tidak Layak
Tua	SMA	Supir	Kontrak	Banyak	Menengah	Layak
Muda	SD	Supir	Kontrak	Normal	Rendah	Tidak Layak

Dari tabel 2.1 diketahui terdapat 20 data *record* data pendapatan masyarakat yang memiliki beberapa kriteria, yakni umur, pendidikan, pekerjaan, pemilikan rumah, jumlah anggota keluarga dan pendapatan, juga terdapat kelasnya yang dilabeli dengan keputusan. Berdasarkan nilai kriteria yang diberikan, data tersebut dibagi menjadi dua kelas, yaitu layak dan tidak layak. Sebelum melakukan proses perhitungan algortima C5.0, perlu dilakukan pengelompokan data pada tabel tersebut. Pengelompokan dilakukan berdasarkan nilai dari kriteria pada *dataset* dan juga kelasnya. Lebih jelasnya, akan ditampilkan pada tabel dibawah hasil dari pengelompokan data pendapatan masyarakat.

Tabel 2.2 Pengelompokan Data

		Jumlah Data	Layak	Tidak Layak
Keputusan		20	12	8
Umur	Muda	13	6	7
	Tua	7	6	1
Pendidikan	SD	4	1	3
	SMP	8	4	4
	SMA	8	7	1
Pekerjaan	Supir	10	4	6
	Pedagang	2	2	0
	Penjahit	8	6	2

		Jumlah Data	Layak	Tidak Layak
Pemilik Rumah	Numpang	9	6	3
	Kontrak	7	2	5
	Milik Pribadi	4	4	0
Jumlah Anggota	Normal	11	3	8
	Banyak	9	9	0
Pendapatan	Rendah	7	5	2
	Menengah	13	7	6

Pada tabel 2.2, setiap kriteria yang tersedia masing-masing dihitung nilai *entropy*, *information gain*, dan *gain ratio*. Langkah pertama untuk menentukan sebuah atribut yang akan dijadikan *parent node* adalah dengan menghitung nilai *entropy* masing-masing atribut. Hitung terlebih dahulu nilai *entropy* total yang mana atribut tersebut adalah atribut keputusan yang berperan sebagai atribut kelas. dapat dilihat terdapat total data sebanyak 20 data dan klasifikasi kedalam kelas layak sebanyak 12 dan tidak layak sebanyak 8. Kemudian, gunakan persamaan (2. 1) untuk menghitung nilai *entropy* keputusan.

$$\textit{Entropy Total (S)} = \left(-\frac{12}{20}\log\frac{12}{20}\right) + \left(-\frac{8}{20}\log\frac{8}{20}\right)$$
$$= 0{,}442 + 0{,}529$$
$$= 0{,}971$$

Dari perhitungan diatas, diperoleh nilai *entropy* total sebesar 0,971. Perhitungan *entropy* total yang dilakukan terhadap kelas atribut diperlukan untuk menghitung *information gain* nantinya. Setelah selesai menghitung nilai *entropy* total, dilanjutkan dengan menhitung nilai *entropy* semua atribut yang tersisa. Hasil dari nilai *entropy* seluruh atribut dapat dilihat seperti pada tabel dibawah.

Tabel 2.3 Nilai *Entopy* Seluruh Atribut

		Entropy
Keputusan		0,971
Umur	Muda	0,996
	Tua	0,592

		Entropy
Pendidikan	SD	0,811
	SMP	1
	SMA	0,544
Pekerjaan	Supir	0,971
	Pedagang	0
	Penjahit	0,811
Pemilik Rumah	Numpang	0,918
	Kontrak	0,863
	Milik Pribadi	0
Jumlah Anggota	Normal	0,845
	Banyak	0
Pendapatan	Rendah	0,863
	Menengah	0,996

Langkah selanjutnya yaitu menghitung nilai *information gain* dari masing-masing atribut, kecuali atribut kelas. Perhitungan untuk atribut kelas hanya dilakukan pada pencarian nilai *entropy* saja, karena atribut kelas akan digunakan sebagai *leaf node* pada pembentukan pohon keputusan pada akhirnya. Lanjut, mengacu pada tabel 2.2 dan 2.3 sebagai contoh ambil saja atribut umur, diketahui terdapat 2 cabang yaitu muda yang memiliki jumlah data sebanyak 13 data dan tua sebanyak 7 data. Serta dari perhitungan *entropy* yang telah dilakukan sebelumnya menghasilkan nilai sebesar 0,996 pada cabang muda dan 0,592 pada cabang tua. Setelah diketahui data yang diperlukan, dilakukan perhitungan *information gain* dengan menggunakan persamaan (2. 2).

$$IG\ (umur) = 0{,}971 - \left(\left(\frac{13}{20} * 0{,}996\right) + \left(\frac{7}{20} * 0{,}592\right)\right)$$

$$= 0{,}971 - (0{,}647 + 0{,}207)$$

$$= 0{,}117$$

Dari perhitungan diatas, didapat nilai *information gain* pada atribut umur sebesar 0,117. Perhitungan tersebut diterapkan pada seluruh atribut maka akan diperoleh hasil yang tampak seperti tabel dibawah.

Tabel 2.4 Hasil *Entropy* dan *Information Gain*

		Entropy	*Information Gain*
Keputusan		0,971	
Umur	Muda	0,996	0,117
	Tua	0,592	
Pendidikan	SD	0,811	0,191
	SMP	1	
	SMA	0,544	
Pekerjaan	Supir	0,971	0,161
	Pedagang	0	
	Penjahit	0,811	
Pemilik Rumah	Numpang	0,918	0,256
	Kontrak	0,863	
	Milik Pribadi	0	
Jumlah Anggota	Normal	0,845	0,506
	Banyak	0	
Pendapatan	Rendah	0,863	0,022
	Menengah	0,996	

Langkah terakhir untuk menentukan atribut mana yang akan dijadikan sebagai *parent node* yaitu menghitung nilai *gain ratio*. Nilai *gain ratio* diperoleh dengan melakukan pembagian *information gain* terhadap jumlah seluruh *entropy* yang rumusnya sesuai dengan persamaan (2. 3). Perhitungannya dapat dilihat pada contoh dibawah yang menggunakan atribut umur.

Gain Ratio $= 0{,}117/(0{,}996 + 0{,}592)$

$= 0{,}074$

Dari perhitungan diatas diperoleh nilai *gain ratio* dari atribut umur sebesar 0,074, dan jika dilakukan perhitungan terhadap seluruh atribut, maka akan tampak seperti tabel dibawah.

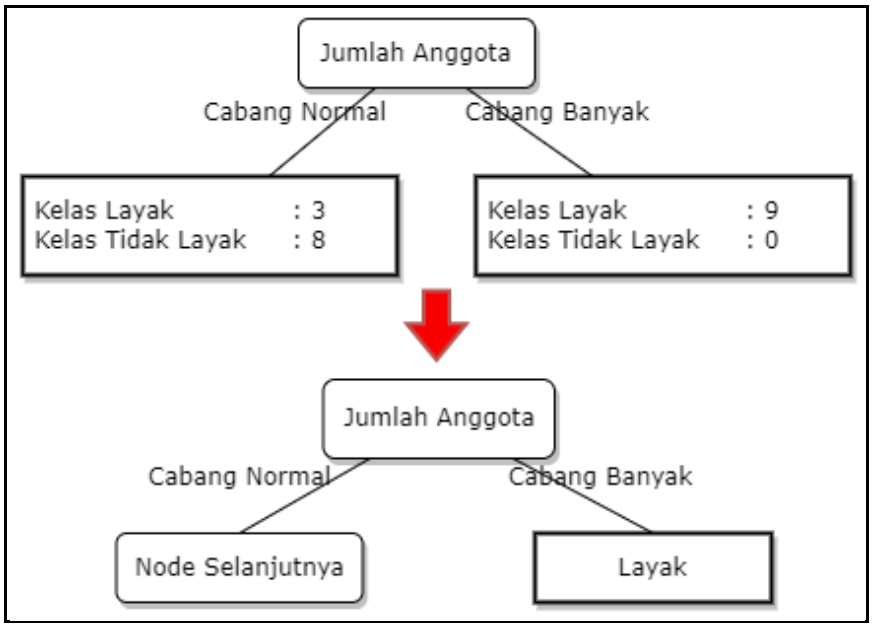
Tabel 2.5 Hasil *Gain Ratio*

		Gain Ratio
Umur	Muda	0,074
	Tua	
Pendidikan	SD	0,081
	SMP	
	SMA	
Pekerjaan	Supir	0,090
	Pedagang	
	Penjahit	
Pemilik Rumah	Numpang	0,144
	Kontrak	
	Milik Pribadi	
Jumlah Anggota	Normal	0,599
	Banyak	
Pendapatan	Rendah	0,012
	Menengah	

Berdasarkan tabel 2.5 yang menunjukkan hasil *gain ratio* seluruh atribut, nilai terbesarnya yaitu ada pada atribut jumlah anggota yang memiliki nilai *gain ratio* sebesar 0,599. Sehingga, yang dijadikan *parent node* pertama dalam pembentukan pohon keputusan adalah jumlah anggota. Setelah diketahui atribut mana yang dijadikan sebagai *parent node,* selanjutnya yaitu menentukan apakah cabang dari *node* tersebut dapat langsung menghasilkan *leaf node* atau perlu dilanjutkan pencarian *node* baru.

Untuk mengetahuinya, terdapat kondisi yang menentukan apakah cabang tersebut akan terbentuk suatu *leaf node* atau tidak yaitu jika cabang tersebut seluruh datanya terklasifikasi kedalam satu kelas saja, artinya kelas yang memenuhi data tersebut akan diguanakan sebagai *leaf node.* Sebagai contoh dari perhitungan yang telah dilakukan sebelumnya, dapat dilihat pada tabel 2.2 bahwa jumlah anggota pada cabang normal terklasifikasi layak sebanyak 3 dan tidak layak sebanyak 8, artinya cabang tidak menemukan kelas yang akan dijadikan

sebagai *leaf node*. Sedangkan, pada cabang banyak, terklasifikasi kedalam kelas layak sebanyak 9 data dan tidak layak sebanyak 0. Yang artinya, kelas layak akan dijadikan sebagai *leaf node* daripada cabang banyak. Untuk lebih jelasnya dapat dilihat gambar 2.1 dibawah.



Gambar 2.1 Pohon Keputusan Node Pertama

2.4. *Reduce Error Pruning*(REP)

REP(*Reduce Error Pruning*) merupakan salah satu algoritma *pruning* yang dikemukakan oleh Quinlann pada tahun 1987. *Pruning* merupakan salah satu cara untuk memotong pohon keputusan sehingga proses generalisasi pada datanya akan menjadi lebih baik. Tahapan ini hanya mengolah data latih untuk membentuk pohon keputusan dan tidak membutuhkan data uji untuk mengukur tingkat galat selama proses *pruning* (Aric, Intan, & Fetty, 2019).

Terdapat 2 pendekatan yang dapat dilakukan dalam pemangkasan pohon keputusan, yaitu *pre-pruning* dan *post-pruning*. *Pre-pruning* yaitu menghentikan pembangunan *subtree* lebih awal, dimana pemangkasan dilakukan saat proses pohon keputusan dibuat. Ketika pembangunan *subtree* berhenti, maka *node* akan berubah menjadi *leaf(node* akhir*)*. Sedangkan *post-pruning* yaitu pemangakasan yang dilakukan ketika seluruh pohon selesai dibuat. Pengukuran tingkat galat dapat dihitung dengan rumus seperti berikut :

$$e = \frac{f + \frac{z^2}{2N} + \sqrt[z]{\frac{f}{N} - \frac{f^2}{N} + \frac{z^2}{4N^2}}}{1 + \frac{z^2}{N}} \quad (2.\ 4)$$

Keterangan :

e = tingkat galat pada *node* (*error rate*).

f = perbandingan jumlah data yang salah dalam klasifikasi dengan jumlah keseluruhan data pada *node* yaitu $\frac{E}{N}$, dengan E adalah jumlah galat.

N = jumlah data pada *node*.

z = nilai *invers* dari distribusi kumulatif normal baku dengan α sebagai parameter sesuai masukan dari pengguna (-0,67).

Cara kerja *pre-pruning* adalah dengan menghitung terlebih dahulu nilai *gain ratio* untuk menentukan *parent node* dan *child node*. Setelah *parent node* sudah diketahui kemudian dihitung nilai eror menggunakan persamaan (2. 4). Jika nilai eror *child node* lebih besar dibandingkan dengan *parent node* maka proses pemangkasan dilakukan dan perhitungan *gain ratio* untuk melakukan pencarian *node* yang selanjutnya dihentikan. Sebaliknya, jika nilai eror *child node* tidak lebih besar dibandingkan dengan *parent node* maka *parent node* akan membentuk *subtree* lagi dan proses perhitungan *gain ratio* dilanjutkan.

Berdasarkan hasil dari perhitungan sebelumnya yang terdapat pada gambar 2.1, menghasilkan *parent node* dengan atribut jumlah anggota yang memiliki dua cabang, yaitu cabang normal dan cabang banyak. Pada cabang banyak, dapat dilihat bahwa cabang tersebut sudah menghasilkan sebuah *leaf node* yang masuk kedalam kelas layak, dimana cabang tersebut tidak perlu dilakukan proses *pruning*. Sedangkan, pada cabang normal, masih belum menghasilkan *leaf node*, sehingga akan dilakukan pemangkasan atau pruning.

Proses perhitungan *pruning* dilakukan dengan menggunakan data yang sudah terklasifikasikan berdasarkan jumlah data dan jumlah kelas nya. Dapat diperhatikan pada tabel 2.2, untuk *parent node* sendiri yakni jumlah anggota memiliki dua cabang dengan masing cabang yaitu cabang normal yang memiliki jumlah data sebanyak 11 data dan cabang banyak sebanyak 9 data. Untuk mengetahui nilai dari f yang sesuai dengan persamaan (2. 4), maka akan diambil

jumlah data terbanyak sebagai kelasnya. Dalam hal ini, cabang normal memiliki jumlah terbanyak, sehingga nilai dari jumlah galat sebanyak 9. Kemudian untuk nilai *f* –nya didapat dari pembagian antara jumlah galat dan total data dari *parent node*. Agar mempermudah dalam pemahamannya, akan ditampilkan detail perhitungannya dibawah.

$$\textit{Parent Node} = \frac{\frac{9}{20}+\frac{(-0{,}67)^2}{2*20}+(-0.67)\sqrt{\frac{\frac{9}{20}}{20}-\frac{\frac{9}{20}^2}{20}+\frac{(-0{,}67)^2}{4*20^2}}}{1+\frac{(-0{,}67)^2}{20}}$$

$$= 0{,}377$$

Dari perhitungan diatas, diperoleh nilai *error rate* dari *parent node* sebesar 0,377. Selanjutnya, dilakukan perhitungan dari cabangnya yaitu cabang normal. Pada tabel 2.2 diketahui jumlah data terklasifikasi layak pada cabang normal dari atribut jumlah anggota sebanyak 3 data dan 8 data pada kelas tidak layak. Sehingga jumlah galat yang diperoleh sebanyak 3. Dari data yang telah diketahui tersebut dimasukkan kedalam persamaan (2. 4), dan didapat hasil perhitungan sesuai dengan perhitungan dibawah.

$$\text{Cabang Normal} = \frac{\frac{3}{11}+\frac{(-0{,}67)^2}{2*11}+(-0.67)\sqrt{\frac{\frac{3}{11}}{11}-\frac{\frac{3}{11}^2}{11}+\frac{(-0{,}67)^2}{4*11^2}}}{1+\frac{(-0{,}67)^2}{11}}$$

$$= 0{,}193$$

Dari perhitungan diatas, diperoleh nilai *error rate* sebesar 0,193. Nilai *error rate* tersebut ternyata lebih kecil daripada *error rate* pada *parent node*. Sehingga, pada cabang normal tidak dilakukan pemangkasan dan perhitungan untuk mencari *node* selanjutnya tetap dilakukan.

2.5. *Boosting*

Boosting adalah proses terakhir pada algoritma C5.0 dalam pembentukan pohon keputusan, dimana proses ini merupakan metode khusus atau spesial yang dimiliki algoritma C5.0. Sebuah teknik yang digunakan dengan tujuan untuk

meningkatkan nilai akurasi dengan cara menghitung bobot pada data *training* dengan tujuan untuk mengoptimalkan pengklasifikasian yang dihasilkan dari perhitungan algoritma C5.0 yang pada akhirnya akan digunakan sebagai acuan pada proses pengujian data (Meo, 2017).

Pada dasarnya proses *boosting* dilakukan dengan cara melakukan perulangan perhitungan algoritma C5 agar dapat memperoleh hasil klasifikasi terbaik. Dimana pada setiap perulangan perhitungan dilakukan pengacakan data *training*, yang kemudian data *training* acak yang didapat tersebut diolah pada proses pencarian nilai *entropy*, *information gain*, *gain ratio*, serta nilai *error* yang dihasilkan dari proses *pruning* agar mendapatkan pohon keputusan yang baru. Pada saat proses *boosting* dilakukan, bobot baru akan didapatkan pada setiap *iterasi boosting*. Langkah paling awal yang harus dilakukan yaitu menentukan bobot awal dengan menggunakan persamaan rumus dibawah.

$$w = \frac{1}{\mathrm{n}} \tag{2. 5}$$

Keterangan :

w = bobot awal

n = jumlah data *training*

Setelah menentukan nilai bobot awal, langkah selanjutnya yaitu menghitung nilai tengah atau *midpoint*, yang nantinya akan digunakan untuk menghitung bobot baru. Rumus yang digunakan agar mendapatkan nilai tengah atau *midpoint* adalah sebagai berikut.

$$\begin{aligned} midpoint &= \frac{1}{2}\left[\frac{1}{2}(S_+ + S_-) - S_-\right] \\ &= \frac{1}{4}(S_+ + S_-) \end{aligned} \tag{2. 6}$$

Keterangan :

S_- = jumlah bobot data yang salah diklasifikasi

S_+ = jumlah bobot data yang benar diklasifikasi

Nilai tengah yang didapatkan kemudian digunakan untuk menghitung bobot baru. Terdapat dua bobot baru yang harus dilakukan perhitungan, yaitu bobot baru pada data yang benar diklasifikasi dan bobot baru yang salah diklasifikasi.

Pertama, untuk mendapatkan bobot baru yang salah diklasifikasi, digunakan persamaan rumus sebagai berikut.

$$w_k = w_{k-1} \text{ x } \frac{S_+ - midpoint}{S_+} \quad (2.\ 7)$$

Dan selanjutnya, untuk menghitung nilai bobot baru yang salah diklasifikasikan menggunakan persamaan dibawah.

$$w_k = w_{k-1} + \frac{midpoint}{N_-} \quad (2.\ 8)$$

Keterangan :

w_k = bobot data pada iterasi *boosting* ke-$_k$

w_{k-1} = bobot data pada iterasi *boosting* ke-$_{k-1}$

N_- = jumlah data yang salah diklasifikasi

Setelah selesai dilakukan perhitungan *boosting* hingga menemukan bobot baru, terdapat suatu kondisi yang menentukan apakah suatu perulangan pada *boosting* dapat dihentikan atau harus dilanjutkan. Kondisi agar perulangan dapat dihentikan yaitu jika jumlah bobot data yang salah diklasifikasi (S_-) memiliki nilai kurang dari 0,1 atau rata-rata dari bobot jumlah data yang salah diklasifikasikan bernilai lebih dari 0,5.

2.6. *Confusion Matrix*

Confussion matrix merupakan suatu metode yang dapat digunakan untuk mengukur atau mengevaluasi kinerja suatu model dari proses klasifikasi. Perhitungan pada *confusion matrix* mengandung informasi yang membandingkan antara hasil klasifikasi yang terjadi pada sistem dengan klasifikasi yang seharusnya terjadi. *Confusion matrix* memiliki 4 istilah sebagai representasi hasil dari proses klasifikasi, diantaranya adalah *True Positive* (TP), *True Negative* (TN), *False Positive* (FP) dan *False Negative* (FN). Yang mana, nilai *true negative* merupakan jumlah data *negative* yang terdeteksi dengan benar dan untuk *false positive* adalah data *negative* yang terdeteksi sebagai data *positive* (Karsito & Susanti, 2019).

Tabel 2.6 Tabel *Confusion Matrix*

		Prediksi	
		Positif	Negatif
Aktual	Positif	TP	FN
	Negatif	FP	TN

Dengan dasar tabel *confusion matrix* pada tabel 2.6, maka dapat dilakukan perhitungan nilai akurasi, *precition* dan *recall*. Akurasi merupakan metode pengujian berdasarkan tingkat kedekatan antara nilai prediksi dengan nilai aktual/sebenarnya. Presisi merupakan metode pengujian dengan melakukan perbandingan jumlah informasi relevan yang didapatkan sistem dengan jumlah seluruh informasi yang terambil oleh sistem baik yang relevan maupun tidak. *Recall* merupakan metode pengujian yang membandingkan jumlah informasi relevan yang didapatkan sistem dengan jumlah seluruh informasi relevan yang ada dalam konteks informasi (baik yang terambil maupun yang tidak terambil oleh sistem). Untuk mendapatkan ketiga nilai tersebut, digunakan persamaan seperti dibawah ini.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} * 100\% \quad (2.\ 9)$$

$$precision = \frac{TP}{TP + FP} * 100\% \quad (2.\ 10)$$

$$recall = \frac{TP}{TP + FN} * 100\% \quad (2.\ 11)$$