BAB 2. LANDASAN TEORI

Pada bab ini dijelaskan mengenai semua teori yang digunakan di dalam pembuatan Aplikasi Pengenalan Pola Batik dengan Metode *Gray-Level Cooccurrence Matrix*. Berikut adalah hal-hal yang dijelaskan pada bab ini antara lain pengertian mengenai batik, *gray-level cooccurrence matrix*, *decision tree*. Berikut adalah penjelasan mengenai teori-teori tersebut.

2.1. Batik

Batik adalah kain bergambar yang pembuatannya secara khusus dengan menuliskan atau menerakan malam pada kain itu, kemudian pengolahannya diproses dengan cara tertentu (KBBI). Pada tanggal 2 Oktober 2009, UNESCO secara resmi mengakui batik sebagai warisan budaya Indonesia. Di Indonesia terdapat banyak sekali jenis batik, salah satunya adalah batik geringsing. (UNESCO)

2.1.1. Geringsing

Geringsing adalah ragam hias *isen-isen* latar yang dipakai dalam dunia perbatikan, merupakan stilasi sisik-sisik (ular) naga yang berbintik di tengahnya, bentuknya membundar menyegi empat, tertata berleret dengan rapi, dan menyirap karena pinggirnya saling menutupi (Tulistyantoro, p.23). Kain geringsing telah ada di jaman Majapahit, tercantum di kitab Pararaton dan Negarakertagama. Ada 10 macam i*sen-isen* geringsing, yaitu *buih, cacah gori, kelungsu, kerang-kerangan, kisi-kisi, merica bolong, oter-oter, polkadot, sisik,* dan *sisik melik.*

2.1.1.1. Isen buih

Isen buih adalah *isen-isen* latar berupa bulatan-bulatan tidak sama besar ukurannya dan tersebar tak teratur, setiap bulatannya diberi titik, garis rangkap pendek, atau lingkaran lebih kecil lagi yang ditempatkan secara asimetris jadi tidak tepat di titik tengah atau titik pusatnya, tak ubahnya bola-bola kosong atau transparan yang berisi udara seperti gelembung busa sabun. *Isen* ini adalah pelambang kemeriahan suasana.

2.1.1.2. Isen cacah gori

Isen cacah gori adalah *isen-isen* latar berupa bidang-bidang renik menyegi empat tanpa bintik dan tidak begitu teratur, sebagai hasil stilasi cacahan buah nangka muda (*gori*) untuk disayur, sepintas mitip dengan kotak kisi-kisi tetapi cakupan skalanya lebih sempit. *Isen* ini adalah pelambang kelimpahan diversifikasi sumber pangan.

2.1.1.3. Isen kelungsu

Isen kelungsu adalah *isen-isen* latar yang merupakan stilasi biji pohon asam, sepintas mirip sisik tetapi bidang-bidang membundar menyegi empat reniknya tidak berbintik dan penempatannya tidak tertata rapi seperti sisik. *Isen* ini adalah pelambang kesiapan menjaga kesehatan jasmani.

2.1.1.4. Isen kerang-kerangan

Isen kerang-kerangan adalah *isen-isen* latar berdasarkan stilasi deretan cangkang kerang, bentuk dan penempatannya mirip *isen-isen* sisik yang pinggirnya ditampilkan dengan garis lengkung berangkap, dan kadang-kadang permukaan bidangnya dipenuhi dengan hiasan berupa garis-garis yang menjari atau menyinar. *Isen* ini adalah pelambang kekayaan bahari dan kesejahteraan kehidupan nelayan.

2.1.1.5. Isen kisi-kisi

Isen kisi-kisi adalah *isen-isen* latar berupa jejaring garis-garis horisontal dan vertikal berjarak sama yang saling tegak lurus satu sama lainnya, sehingga menghasilkan bidang-bidang renik membujur sangkar sangat seragamnya yang tak bertitik di tengahnya. *Isen* ini adalah pelambang keteraturan tatanan lingkungan, serta ketegaran kemapanan dan pengamanan.

2.1.1.6. Isen merica bolong

Isen merica bolong adalah *isen-isen* latar berupa taburan bundaran atau lingkaran kecil-kecil, masing-masing dengan sebuah titik agak besar di tengahnya, secara sepintas menyerupai *isen-isen sisik*, tetapi bidang-bidang reniknya tidak menyegi empat melainkanberupa lingkaran (*circle*) sempurna, tertebar dan tidak berleret teratur sehingga latar belakangnya tidaklah merupakan garis-garis batas

yang bersinggungan melainkan bidang yang tersisa oleh lingkaran. *Isen* ini adalah pelambang kemandirian kewanitaan. Sinonimnya antara lain *mata iwak, mata pitik, mata dara,* dan juga *lobheng lewih*.

2.1.1.7. Isen oter-oter

Isen oter-oter adalah *isen-isen* latar berupa lingkaran-lingkaran utuh agak besar tanpa titik didalamnya. Sinonimnya adalah *mata keteran* (Madura).

2.1.1.8. Isen polkadot

Isen polkadot adalah *isen-isen* latar berupa bundaran-bundaran besar yang berwarna kontras dengan latar belakangnya, diletakkan bertaburan secara teratur yang umumnya agak berjauhan sehingga terlihat jarang-jarang, sepintas mirip dengan *isen-isen 'cecek'* yang ditampilkan dengan ukuran besar-besar. *Isen* ini adalah pelambang kegembiraan karena kebebasan dalam keteraturan berekspresi. Sinonimnya adalah '*bolleches*'.

2.1.1.9. Isen sisik

Isen sisik adalah *isen-isen* latar berdasarkan stilasi sisik ikan, berupa bidang membusur setengah lingkaran dengan ukuran lebar melebihi tingginya, tidak berbintik di tengahnya, tertata menyirap dengan pinggir saling menutupi, berleret mendaftar dan menyerong secara beraturan. *Isen* ini adalah pelambang kemakmuran dan kesejahteraan lahir batin.

2.1.1.10. Isen sisik melik

Isen sisik melik adalah *isen-isen* latar berdasarkan stilasi sisik ikan yang setiap sisiknya berbintik di tengahnya. *Isen* ini adalah pelambang kemakmuran dan kesejahteraan lahir batin yang cemerlang. Sinonimnya adalah *sesse' bai'* (Madura).

2.2. Gray-Level Cooccurrence Matrix

Gray-level cooccurrence matrix adalah sebuah metode ekstraksi fitur yang digunakan untuk mendapatkan fitur *texture*/pola pada suatu citra digital. Cara kerja metode ini adalah dengan mencatat intensitas dua pixel yang berdekatan pada suatu gambar *gray-scale*. Ukuran dari GLCM mengikuti banyaknya level intensitas sebuah gambar. Pada gambar *grayscale* 8-bit, terdapat

2^8 = 256 level intensitas, sehingga ukuran GLCM untuk gambar *grayscale* 8-bit adalah 256x256. Untuk mengurangi beban komputasi, gambar *grayscale* 8-bit diubah menjadi gambar 4-bit saja. Hal ini bisa didapat dengan mengambil *4 most significant bits* dari gambar 8-bit, sehingga ukuran GLCM menjadi 16x16 saja. Penjelasan dan contoh pada bagian ini diambil dari (Hall-Beyer, http://www.fp.ucalgary.ca/mhallbey).

GLCM menghitung seberapa sering pixel-pixel dengan *offset* jarak dan arah tertentu memiliki sebuah pasangan nilai. Sebagai contoh, Tabel 2.1 berisi gambar sederhana dengan 4 level intensitas.

Tabel 2.1. Contoh gambar sederhana

0	0	1	1
0	0	1	1
0	2	2	2
2	2	3	3

GLCM yang dihasilkan dengan *offset* (0,1) atau bersebelahan kiri-kanan dari Tabel 2.1 dapat dilihat pada Tabel 2.2.

Tabel 2.2. Hasil GLCM dengan *offset* (0, 1)

2	2	1	0
0	2	0	0
0	0	3	1
0	0	0	1

GLCM menggunakan *zero-based indexing*. Untuk mendapatkan angka pada Tabel 2.2, setiap pixel yang berdekatan pada Tabel 2.1 dihitung jumlah kemunculannya. Sebagai contoh, angka 2 pada baris ke-0 kolom ke-1 adalah seberapa banyak intensitas 0 muncul di sebelah kiri dan intensitas 1 muncul di sebelah kanannya, angka 1 pada baris ke-0 kolom ke-2 adalah seberapa banyak intensitas 0 muncul di sebelah kiri dan intensitas 2 muncul di sebelah kanannya, dsb.

Selanjutnya, GLCM yang dihasilkan dibentuk menjadi simetris terhadap sumbu diagonal utama. Hal ini dikarenakan kemunculan 2 pixel yang berdekatan bersifat dua arah (misal angka 0 di sebelah angka 2, maka angka 2 juga berada di

sebelah angka 0). Proses ini dapat dilakukan dengan cara menjumlahkan GLCM dengan transposnya, sehingga GLCM menjadi seperti Tabel 2.3.

Tabel 2.3. Hasil GLCM yang simetris

4	2	1	0
2	4	0	0
1	0	6	1
0	0	1	2

Pada tahap ini, nilai angka dalam GLCM bisa bervariasi sesuai ukuran gambar yang dihitung. Gambar yang besar akan memiliki jumlah elemen GLCM yang besar. Agar dapat digunakan lebih lanjut, hasil perhitungan GLCM harus dinormalisasi agar jumlah seluruh elemen GLCM menjadi 1. Normalisasi dilakukan dengan cara membagi seluruh isi GLCM dengan total jumlah semua isi GLCM seperti rumus pada (1). GLCM yang sudah dinormalkan menyatakan prosentase suatu elemen terhadap jumlah keseluruhan. Contoh hasil GLCM yang sudah dinormalisasi dapat dilihat pada Tabel 2.4.

$$p_{i,j} = \frac{v_{i,j}}{\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} v_{i,j}} \quad (1)$$

Secara formal, $p_{i,j}$ menyatakan elemen GLCM yang sudah dinormalisasi, $v_{i,j}$ menyatakan elemen GLCM yang belum dinormalisasi, dan N menyatakan ukuran dari GLCM.

Tabel 2.4 Hasil GLCM yang sudah dinormalisasi

0.167	0.083	0.042	0
0.083	0.167	0	0
0.042	0	0.250	0.042
0	0	0.042	0.083

GLCM yang sudah dinormalisasi ini dapat digunakan untuk mendapatkan kalkulasi yang akan dibahas pada subbab berikutnya.

Sebuah *offset* terdiri dari 2 bagian, yaitu arah (θ) dan jarak (d). Jarak d=1 ternyata memiliki hasil yang lebih baik daripada jarak d=2 (Pathak, p.4212). Ada 4 arah *offset* yang umum dipakai, yaitu (0, 1), (1, 1), (0, 1), dan (-1, 1). Keempat *offset* ini masing-masing menggambarkan pixel yang bersebelahan dengan sudut

0^o, 45^o, 90^o, dan 135^o. Sudut 180^o, 225^o, 270^o, dan 315^o tidak dipakai karena sama dengan keempat *offset* umum tersebut.

GLCM sudah pernah digunakan dalam berbagai penelitian sebagai metode untuk mengevaluasi tekstur. Berbagai penelitian tersebut mencapai akurasi yang tinggi, yaitu 80% (Nurhaida, p.1), 91,57% (Pratiwi, p.8) dan 98.927% (Santoni, p.7).

2.2.1. Contrast

Nilai *contrast* menggambarkan seberapa besar perbedaan intensitas antar *pixel-pixel* yang berdekatan. Semakin kontras perbedaan antara *pixel-pixel* yang berdekatan, semakin tinggi nilai *contrast*. Nilai *contrast* merupakan penghitungan derajat kedua. Rumus perhitungan *contrast* dapat dilihat pada (2).

$$Contrast = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} p_{i,j}(i-j)^2 \quad (2)$$

2.2.2. Homogeneity (Inverse Difference Moment)

Nilai *homogeneity* menggambarkan seberapa homogen intensitas antar pixel-pixel yang berdekatan. Semakin kontras perbedaan antara pixel-pixel yang berdekatan, semakin rendah nilai *homogeneity*. Nilai *homogeneity* merupakan penghitungan derajat kedua. Rumus perhitungan *homogeneity* dapat dilihat pada (3).

$$Homogeneity = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \frac{p_{i,j}}{1+(i-j)^2} \quad (3)$$

2.2.3. Correlation

Nilai *correlation* menggambarkan dependensi / korelasi antar pixel-pixel yang berdekatan. Nilai *correlation* merupakan perhitungan derajat kedua. Rumus perhitungan *correlation* dapat dilihat pada (4).

$$Correlation = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} p_{i,j} \left[\frac{(i-\mu_i)(j-\mu_j)}{\sqrt{(\sigma_i^2)(\sigma_j^2)}} \right] \quad (4)$$

2.2.4. Energy

Nilai *energy* atau disebut juga *entropy* menggambarkan perubahan nilai intensitas dalam suatu gambar. Istilah *entropy* diambil dari istilah fisika

termodinamika, yaitu banyaknya energi yang hilang menjadi panas. Rumus perhitungan *energy* dapat dilihat pada (5).

$$Energy = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} {p_{i,j}}^2 \quad (5)$$

2.3. *Expert System*

Expert system adalah sebuah program komputer yang mampu melakukan suatu hal dengan kemampuan setingkat dengan seorang pakar dalam bidang tertentu. Seorang pakar (expert) sendiri didefinisikan sebagai orang yang memiliki kemampuan untuk melakukan sesuatu yang tidak dapat dilakukan orang lain (Negnevitsky, p.25). Secara umum, sistem pakar menggunakan pengetahuan dari seorang pakar dalam bentuk *rule-rule* yang spesifik.

Salah satu kesulitan utama dalam pembuatan sistem pakar adalah mendapatkan pengetahuan (*knowledge*) dan kepakaran dari seorang pakar. Seorang pakar seringkali menggunakan istilah yang tidak jelas dan ambigu, misal istilah "biasanya", "sangat sering", dan "hampir tidak pernah". Untuk mengatasi permasalahan ini, *fuzzy logic* dapat digunakan agar sistem pakar dapat mengolah data yang bersifat tidak jelas (*vague*).

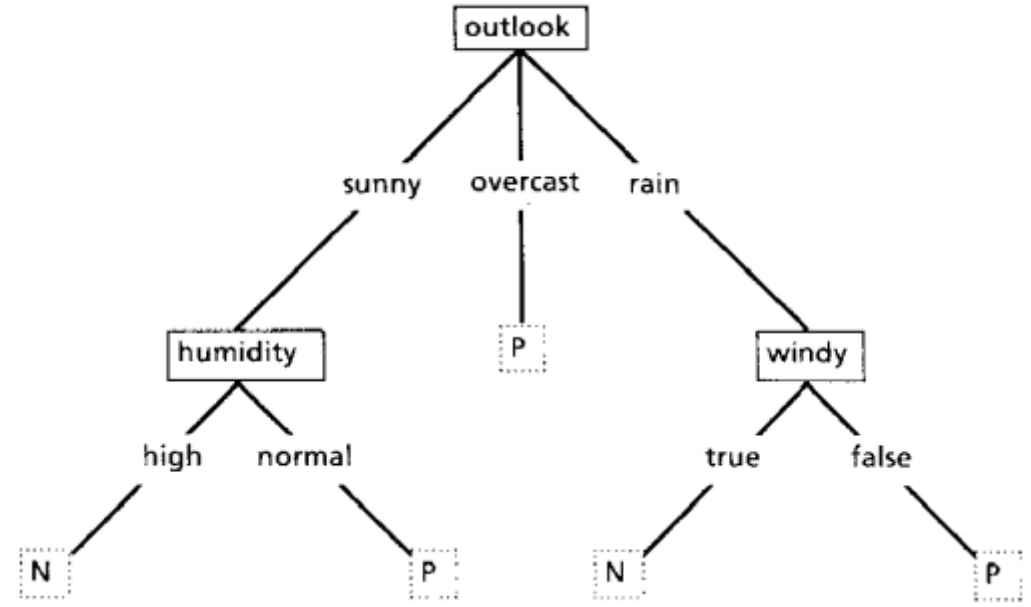
2.3.1. *Fuzzy Expert System*

Fuzzy expert system adalah sistem pakar yang tidak menggunakan *boolean logic*, tapi menggunakan *fuzzy logic*. Sebuah *fuzzy expert system* memiliki banyak *fuzzy rule* dan *membership function* yang dapat digunakan untuk menjelaskan data. Berbeda dengan sistem pakar biasa yang menggunakan label-label berupa kata-kata, *fuzzy expert system* lebih mengarah kepada pemrosesan numerik.

2.4. *Decision Tree*

Decision tree adalah struktur *tree* yang menyerupai *flowchart*, dimana tiap *internal node* menandakan pengetesan terhadap suatu atribut, tiap cabang menyatakan hasil dari tes tersebut, dan *leaf node* menyatakan sebuah kelas-kelas atau distribusinya (Jiawei, p.284). Penggunaan *decision tree* adalah untuk menghasilkan dan memvisualisasikan *rule-rule* yang dihasilkan dari data, sehingga membuat hasil lebih mudah dipahami dan akurat (Sun dkk., p.303).

Dalam *decision tree*, tiap *node leaf* memiliki salah satu keputusan dan tiap *node branch* memiliki sebuah pilihan. Dimulai dari *root*, data akan diolah untuk menentukan pilihan di masing-masing *node*. Jalur yang diambil berbeda-beda, tergantung pilihan apa yang dipilih oleh data pada *node branch*. Data terus diuji dan pindah ke *node* selanjutnya hingga data mencapai suatu *node leaf*. Maka dapat dikatakan bahwa data itu akan menghasilkan keputusan sesuai *node leaf* yang dicapai data. Contoh decision tree dapat dilihat pada Gambar 2.1. Pada gambar tersebut *decision tree* digunakan untuk menentukan kelas untuk sebuah data cuaca di suatu hari. Terdapat 2 kelas, yaitu P dan N.



Gambar 2.1 Contoh *decision tree*

Sumber: (Quinlan, p.87)

Decision tree dapat dibuat secara manual ataupun otomatis. Namun pembuatan secara manual menjadi tidak efektif saat ukuran data yang digunakan untuk membuat *decision tree* menjadi semakin besar. Terdapat berbagai cara untuk membuat sebuah *decision tree* secara otomatis, salah satunya adalah dengan menggunakan algoritma ID3. Dengan menggunakan algoritma ini, *decision tree* yang dihasilkan akan menjadi optimal dan dapat dihasilkan secara otomatis.

2.4.1. ID3

ID3 adalah algoritma yang dapat digunakan untuk menghasilkan *decision tree* dari sebuah *dataset* (Bhatt dkk., p.4785). ID3 (*Iterative Dichotomizer* 3)

dikembangkan untuk menentukan dari pola saja apakah sebuah posisi tertentu dalam pertandingan catur antara raja dan benteng melawan raja dan kuda akan berakhir dengan kalahnya raja dan kuda (Quinlan, p.84). Dalam perkembangannya, algoritma ID3 cocok untuk pengambilan keputusan dengan dataset yang bersifat diskrit.

Atribut-atribut dataset yang digunakan dalam algoritma ID3 terdiri dari 2 macam, yaitu atribut penyebab (*antecedent*) dan atribut akibat (*consequent*). Atribut penyebab adalah atribut yang digunakan untuk menentukan nilai atribut akibat. Algoritma ini menggunakan kalkulasi entropi dalam data untuk menentukan atribut mana yang paling menentukan suatu kolom keputusan. Entropi adalah ukuran seberapa sebuah dataset itu tersebar (*random*). Nilai entropi dihitung sebagai relasi antara sebuah atribut penyebab dengan atribut akibat. Atribut dengan nilai entropi terrendah menandakan bahwa atribut tersebut paling menentukan atribut akibatnya.

Setiap atribut *consequent* memiliki *information need*, yaitu ukuran seberapa besar informasi yang dibutuhkan untuk dapat menentukan atribut *consequent* tersebut.

Dataset S didefinisikan sebagai berikut. Misal *m* adalah banyaknya macam nilai pada atribut *consequent* (sehingga ada *m* kelas pada atribut *consequent*), *s* adalah ukuran dataset, dan s_i adalah banyaknya data yang atribut *consequent*-nya bernilai c_i. Rumus untuk menghitung *information need* dapat dilihat pada (6) dan (7).

$$I(s_1, s_2, \dots, s_m) = -\sum_{i=1}^{m} p_i . log_2(p_i) \quad (6)$$

$$p_i = \frac{s_i}{s} \quad (7)$$

Misal suatu atribut *antecedent* mempunyai *v* macam nilai yang berbeda. s_{ij} didefinikan sebagai banyaknya data dengan atribut *antecedent* bernilai s_j dan atribut *consequent* bernilai c_i. *Entropy* dapat dihitung menggunakan (8), (9), dan (10).

$$E(A) = \sum_{j=1}^{v} \frac{s_j}{s} I(s_{1j}, s_{2j}, \dots, s_{mj}) \quad (8)$$

$$I(s_{1j}, s_{2j}, \dots, s_{mj}) = -\sum_{i=1}^{m} p_{ij} . log_2(p_{ij}) \quad (9)$$

$$p_{ij} = \frac{s_{ij}}{s_j} \quad (10)$$

Jika *entropy* mengukur persebaran nilai dalam sebuah *dataset*, kebalikannya adalah *information gain*. *Information gain* dapat dihitung menggunakan (11).

$$Gain(A) = I(s_1, s_2, \dots, s_m) - E(A) \quad (11)$$

2.4.2. *Cutting Point*

Salah satu kelemahan algoritma ID3 adalah atribut-atribut dalam *dataset* harus bersifat diskrit. Proses dapat dilakukan sampai dengan penghitungan *information gain* menggunakan data yang bersifat kontinu. Data yang bersifat kontinu berarti akan ada banyak percabangan pada suatu *node*, menyebabkan *decision tree* yang dibuat menjadi kurang optimal. Salah satu cara mengatasi permasalahan ini adalah dengan cara memetakan nilai-nilai kontinu pada atribut menjadi nilai-nilai diskrit. (Bahety, p.6)

Cara lain untuk mengatasi masalah ini adalah dengan mencari sebuah *cutting point*. Pilihan pada *node* ini menjadi apakah nilai atribut lebih besar atau lebih kecil dari *cutting point*. Untuk mendapatkan *cutting point* yang paling optimal dilakukan pendekatan *greedy*. Coba untuk semua nilai pada dataset menjadi cutting point, dan tentukan cutting point mana yang membuat entropy paling minimal. Dengan begitu, *decision tree* yang dihasilkan tetap menjadi optimal.

2.5. Sistem Logika *Fuzzy*

Sistem logika *fuzzy* adalah pengembangan dari sistem logika *boolean*, yaitu sistem logika dimana tiap pernyataan bernilai 1 (benar) atau 0 (salah). Dalam sistem logika *fuzzy*, kebenaran suatu pernyataan bisa bernilai sebuah bilangan riil antara 0 dan 1. Sistem logika *fuzzy* diciptakan untuk memberi definisi pada ketidakjelasan (*vagueness*). Sebagai contoh, derajat Celcius sering digunakan untuk mengukur panasnya suhu udara. Namun definisi ‘panas’ itu sendiri tidak jelas & tidak ada kesepakatan mengenai panasnya suhu udara.

Sistem logika *fuzzy* pertama kali diajukan oleh Lotfi Zadeh pada tahun 1965. Sistem logika *fuzzy* dapat diaplikasikan ke dalam berbagai konsep, misalnya konsep himpunan yang dikombinasikan dengan sistem logika *fuzzy* akan menghasilkan himpunan *fuzzy* (*fuzzy set*).

2.5.1. Fuzzy Set

Fuzzy set adalah himpunan dengan derajat keanggotaan yang bersifat tidak tegas (*fuzzy*). Berbeda dengan teori himpunan klasik dimana tiap objek hanya bisa merupakan anggota atau bukan anggota dari sebuah himpunan, *fuzzy set* menggambarkan derajat keanggotaan sebagai sebuah bilangan riil antara 0 dan 1. Sebagai contoh, diketahui suatu himpunan sekelompok manusia sebagai himpunan semesta. Misal dalam himpunan tersebut memiliki subhimpunan manusia yang gemuk. Dalam teori himpunan klasik, seorang manusia hanya bisa merupakan anggota himpunan manusia yang gemuk atau bukan anggota himpunan manusia yang gemuk. Konsep fuzzy set memungkinkan seseorang untuk hanya menjadi 70% gemuk atau 45% gemuk.

2.5.2. *Fuzzy Decision Tree*

Penjelasan pada bagian ini diambil dari (Intan & Yuliana, 2006). *Fuzzy Decision Tree* adalah pengembangan dari *decision tree induction* yang dikombinasikan dengan sistem logika *fuzzy*. Pada *fuzzy decision tree*, *rule* yang dibuat menggunakan *fuzzy set* sebagai sebab (*antecedent*) dan akibat (*consequent*). Pada *decision tree* klasik percabangan pada nilai numerik membutuhkan adanya titik potong.

Sebagai contoh, sebuah sistem menggunakan data suhu untuk menentukan kecepatan kipas angin. Dalam teori *decision tree* klasik, *rule* yang mungkin terbentuk adalah ‘Jika suhu > 25^{o}C maka kecepatan = 20 putaran/detik’. Kelema *decision tree* klasik adalah jika suhu yang didapat adalah 24.9^{o}C maka rule tersebut tidak dijalankan. Dalam *fuzzy decision tree*, *rule* yang mungkin terbentuk adalah ‘Jika suhu = panas maka kecepatan = tinggi’, dengan *fuzzy set* panas untuk himpunan suhu udara dan *fuzzy set* tinggi untuk himpunan kecepatan putaran kipas.

Misal S menyatakan himpunan yang terdiri dari *s* sampel data. Himpunan atribut memiliki *m* nilai, yaitu v_i (untuk i=1, …, m), membentuk *m* kelas yang berbeda, C_i (untuk i=1, …, m). Terdapat pula *n fuzzy label*, F_j (untuk j=1, …, n) yang didefinisikan pada *m* atribut, v_i. $F_j(v_i)$ menyatakan derajat keanggotaan v_i pada *fuzzy set* F_j. Misal β_j adalah weighted sample pada fuzzy set F_j yang dapat

dihitung seperti pada (12). Informasi yang dibutuhkan untuk mengklasifikasi didapat dari (13) dan (14).

$$\beta_j = \sum_i^m \det(C_i) \times F_j(v_i) \quad (12)$$

$$I(\beta_1, \beta_2, \dots, \beta_n) = -\sum_{j=1}^n p_j log_2(p_j) \quad (13)$$

$$p_j = \frac{\beta_j}{s} \quad (14)$$

Misal atribut *A* memiliki *u* nilai yang berbeda, {a_1, a_2, …, a_u}, yang mendefinisikan *u* kelas yang berbeda, B_h (untuk *h*=1, …, *u*). Misal ada *r fuzzy label*, T_k (untuk *k*=1, …, *r*) yang terdefinisi pada *A* sehingga T_k memenuhi (15).

$$\sum_k^r T_k(a_h) = 1, \forall h \in \{1, \dots, u\} \quad (15)$$

Nilai *entropy* dari atribut *A* dapat dihitung menggunakan (16), (17), dan (18), sehingga nilai *information gain* juga dapat dihitung menggunakan (19).

$$E(A) = \sum_{k=1}^r \frac{\alpha_{1k} + \dots + \alpha_{nk}}{s} I(\alpha_{1k}, \dots, \alpha_{nk}) \quad (16)$$

$$\alpha_{jk} = \sum_h^u \sum_i^m \min(F_j(v_i), T_k(a_h)) \times \det(C_i \cap B_h) \quad (17)$$

$$I(\alpha_{1k}, \dots, \alpha_{nk}) = -\sum_{j=1}^n p_{jk} log_2(p_{jk}) \quad (18)$$

$$Gain(A) = I(\beta_1, \beta_2, \dots, \beta_n) - E(A) \quad (19)$$

2.6. *C#*

C# adalah bahasa pemrograman yang dikembangkan oleh *Microsoft*. Pengembang utama dari bahasa pemrograman ini adalah Anders Hejlsberg, Scott Wiltamuth dan Peter Goode. *C#* pertama kali dirilis oleh *Microsoft* pada Juli 2000 sebagai bagian dari *.NET Framework* (Ecma International, p.xix).

Bahasa C# adalah bahasa pemrograman multi-paradigma, berarti C# mendukung lebih dari satu paradigma pemrograman. C# dikembangkan sebagai bahasa pemrograman berbasis objek dan bahasa pemrograman prosedural, namun pada versi 3.0 C# juga mendukung pemrograman fungsional. (Hamilton).