Penggunaan Pohon Keputusan untuk *Data Mining*

Indah Kuntum Khairina – NIM 13505088
Program Studi Teknik Teknik Informatika, Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10, Bandung 40132
email: if15088@students.if.itb.ac.id

Abstract – *Data mining* merupakan proses ekstraksi pengetahuan secara otomatis dari data berukuran besar dengan cara mencari pola-pola menarik yang terkandung di dalam data. *Data mining* memiliki banyak fungsionalitas di mana setiap fungsionalitas akan menghasilkan jenis pola yang berbeda satu sama lain. Klasifikasi adalah suatu fungsionalitas *data mining* yang akan menghasilkan model yang mampu memprediksi kelas atau kategori dari objek-objek di dalam basisdata. Model klasifikasi dibuat dengan cara menganalisis *training data*. Model yang dihasilkan nantinya dapat digunakan untuk memprediksi kelas dari *unknown data*. Model klasifikasi dapat digambarkan dalam berbagai bentuk, salah satunya adalah dengan menggunakan pohon keputusan. *Attribute selection measures* merupakan elemen yang sangat penting dalam pembangunan pohon keputusan. Jenis *attribute selection measures* yang paling banyak digunakan adalah *information gain*, *gain ratio*, dan gini *index*. *Sementara* itu, pemangkasan pohon dapat dilakukan untuk menghilangkan cabang-cabang tidak perlu yang terbentuk akibat adanya *noise* atau *outlier* pada *training data*.

Kata Kunci: pohon keputusan, *data mining*, klasifikasi.

1. PENDAHULUAN

Kemudahan penyimpanan dan pengaksesan data oleh suatu aplikasi menyebabkan membengkaknya jumlah data yang tersedia. Sudah banyak orang yang menyadari bahwa data yang berukuran besar tersebut sebenarnya mengandung berbagai jenis pengetahuan tersembunyi yang berguna untuk proses pengambilan keputusan. Akan tetapi, pengetahuan akan sangat sulit ditemukan dengan cara menganalisis data secara manual. Oleh karena itu, dilakukan *data mining* untuk mengekstraksi pengetahuan secara otomatis dari data berukuran besar dengan cara mencari pola-pola menarik yang terkandung di dalam data tersebut. *Data mining* memiliki banyak fungsionalitas, antara lain pembuatan ringkasan data, analisis asosiasi antar data, klasifikasi data, prediksi, dan pengelompokan data. Setiap fungsionalitas akan menghasilkan pengetahuan atau pola yang berbeda satu sama lain.

Pada klasifikasi, akan dihasilkan sebuah model yang dapat memprediksi kelas atau kategori dari objek-objek di dalam basisdata. Sebagai contoh, klasifikasi dapat digunakan oleh petugas peminjaman uang di sebuah bank untuk memprediksi pemohon mana yang aman dan mana yang beresiko untuk diberi pinjaman, oleh manajer pemasaran di sebuah toko elektronik untuk memprediksi apakah seorang pelanggan akan membeli komputer baru, atau oleh periset di bidang medis untuk memprediksi jenis pengobatan apa yang cocok diberikan kepada seorang pasien dengan penyakit tertentu. Pada kasus-kasus tersebut, model klasifikasi dibuat untuk memprediksi kelas ”aman” atau ”beresiko” untuk data permohonan pinjaman; ”beli” atau ”tidak” untuk data pemasaran; dan ”pengobatan-1”, ”pengobatan-2”, atau ”pengobatan-3” untuk data medis.

Model klasifikasi dibuat dengan cara menganalisis *training data* (terdiri dari objek-objek yang kelasnya sudah diketahui). Model yang dihasilkan kemudian akan digunakan untuk memprediksi kelas dari *unknown data* (terdiri dari objek-objek yang kelasnya belum diketahui). Model klasifikasi dapat digambarkan dalam beberapa bentuk, seperti aturan klasifikasi (`IF-THEN`), pohon keputusan, rumus matematika, atau jaringan saraf tiruan. Pohon keputusan banyak digunakan karena mudah dipahami oleh manusia serta mampu menangani data beratribut banyak.

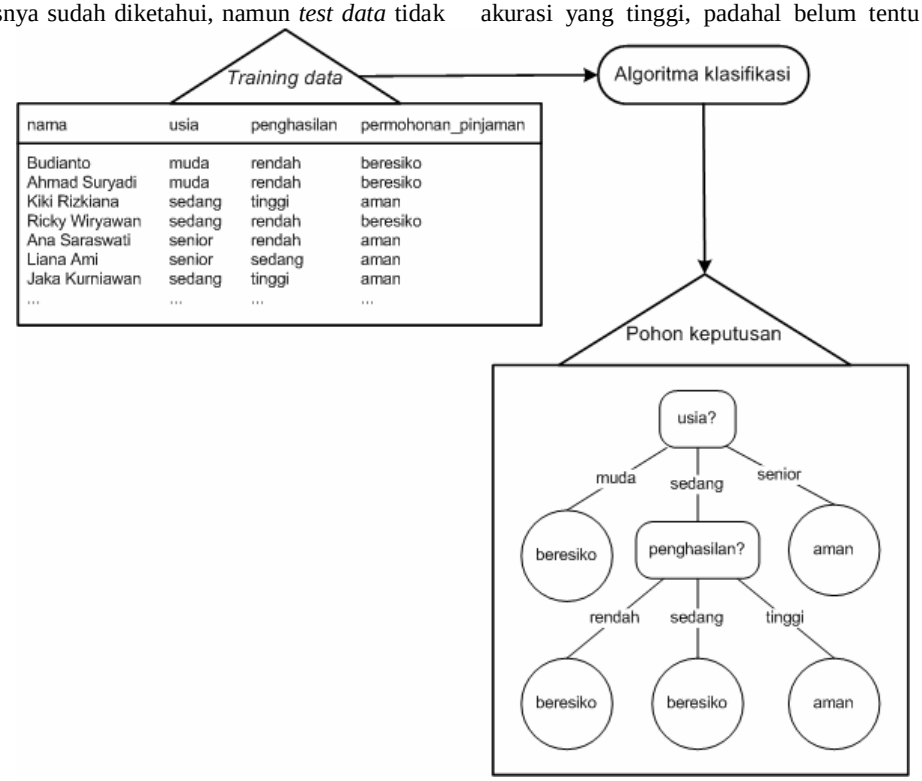
2. KLASIFIKASI

Klasifikasi adalah suatu fungsionalitas *data mining* yang akan menghasilkan model untuk memprediksi kelas atau kategori dari objek-objek di dalam basisdata. Klasifikasi merupakan proses yang terdiri dari dua tahap, yaitu tahap pembelajaran dan tahap pengklasifikasian.

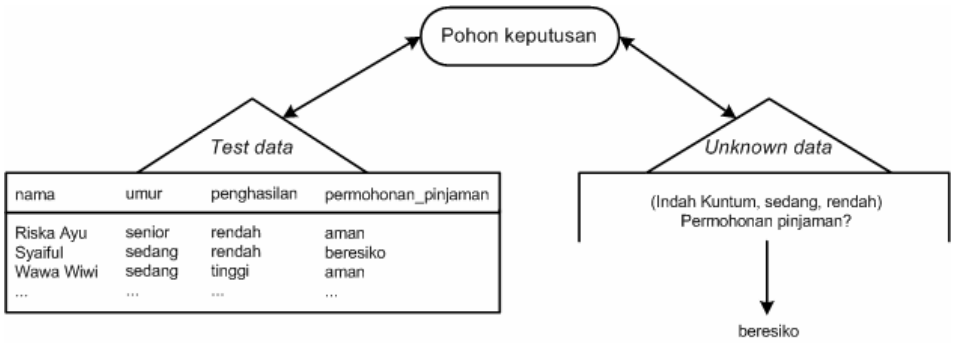
Pada tahap pembelajaran, sebuah algoritma klasifikasi akan membangun sebuah model klasifikasi dengan cara menganalisis *training data*. Tahap pembelajaran dapat juga dipandang sebagai tahap pembentukan fungsi atau pemetaan $y = f(X)$ di mana y adalah kelas hasil prediksi dan X adalah *tuple* yang ingin diprediksi kelasnya. Pada makalah ini, fungsi atau pemetaan tersebut akan digambarkan dalam bentuk pohon keputusan.

Selanjutnya, pada tahap pengklasifikasian, model yang telah dihasilkan akan digunakan untuk melakukan klasifikasi terhadap *unknown data*. Akan tetapi, sebuah model hanya boleh digunakan untuk klasifikasi jika akurasi model tersebut cukup tinggi. Akurasi

dapat diketahui dengan cara menguji model tersebut dengan *test data*. *Test data* terdiri dari *tuple-tuple* yang kelasnya sudah diketahui, namun *test data* tidak boleh sama dengan *training data* karena akan menyebabkan pengujian tersebut menunjukkan akurasi yang tinggi, padahal belum tentu demikian.



Gambar 1. Tahap pembelajaran



Gambar 2. Tahap pengklasifikasian

3. POHON KEPUTUSAN UNTUK KLASIFIKASI

Pohon keputusan merupakan salah satu bentuk penggambaran model klasifikasi. Pada pohon keputusan, simpul dalam menyatakan pengujian terhadap suatu atribut (digambarkan dengan kotak), cabang menyatakan hasil dari suatu pengujian (digambarkan dengan panah yang memiliki label dan arah), sementara daun menyatakan kelas yang diprediksi (digambarkan dengan lingkaran). Contoh pohon keputusan untuk kasus permohonan pinjaman dapat dilihat pada gambar 1 di atas.

Pada saat membangun pohon keputusan, digunakan *attribute selection measures* dalam memilih kriteria terbaik untuk mempartisi *tuple-tuple* data ke dalam kelas-kelas berbeda. Kriteria tersebut meliputi *splitting attribute*, *split point*, maupun *splitting subset*. *Attribute selection measures* akan dibahas pada upabab 3.2. Sementara itu, terlalu banyak cabang pada suatu pohon keputusan mungkin mencerminkan adanya *noise* (kesalahan pencatatan nilai) atau *outlier* (penyimpangan nilai dari rentang seharusnya) pada *training data*. Pemangkasan pohon dapat dilakukan untuk mengenali dan menghapus cabang-cabang tersebut, sehingga diharapkan dapat meningkatkan akurasi model klasifikasi. Pemangkasan pohon akan dibahas pada upabab 3.3.

3.1. Algoritma

ID3 (*Iterative Dichotomiser* 3), C4.5 (suksesor ID3), dan CART (*Classification and Regression Trees*) merupakan beberapa contoh algoritma pembangunan pohon keputusan. Ketiga algoritma tersebut pada dasarnya memiliki karakteristik yang sama dalam membangun pohon keputusan, yaitu *top-down* dan *divide-and-conquer*. *Top-down* artinya pohon keputusan dibangun dari simpul akar ke daun, sementara *divide-and-conquer* artinya *training data* secara rekursif dipartisi ke dalam bagian-bagian yang lebih kecil saat pembangunan pohon. Biner tidaknya pohon keputusan yang dihasilkan ditentukan oleh *attribute selection measures* ataupun algoritma yang digunakan. Secara umum, algoritma pembangunan pohon keputusan dapat dirangkum sebagai berikut.

Tabel 1. Algoritma Pembangunan Pohon Keputusan

Algoritma: Generate_decision_tree

Input:

- `D` = partisi data yang mula-mula terdiri dari seluruh *tuples* pada *training data.*
- `attribute_list` = daftar atribut yang dimiliki oleh data.
- *`Attribute_selection_method`* = prosedur untuk menentukan kriteria terbaik dalam mempartisi data ke dalam kelas-kelas.

Output: Pohon keputusan.

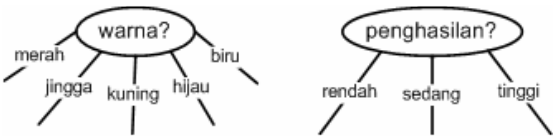
```
(1) create node N;
(2) if tuples in D are all of the same class,
    C, then
(3)     return N as a leaf node labeled with
        the class C;
(4) if attribute_list is empty then
(5)     return N as a leaf node labeled with
        the majority class in D;
(6) apply Attribute_selection_method(D,
    attribute_list) to find the best splitting
    criterion;
(7) label node N with splitting_criterion;
(8) if splitting_attribute is discrete-valued
    and multiway splits allowed then
(9)     attribute_list ← attribute_list –
                        splitting_attribute;
(10)for each outcome j of splitting_criterion
(11)    let Dj be the set of data tuples in D
        satisfying outcome j;
(12)    if Dj is empty then
(13)        attach a leaf labeled with the
            majority class in D to node N;
(14)    else attach the node returned by
            Generate_decision_tree(Dj,
            attribute_list) to node N;
(15)endfor
(16)return N
```

Algoritma di atas dapat dijelaskan sebagai berikut. Pada awalnya, pohon hanya memiliki sebuah simpul, N, yang mewakili seluruh *training data* di D. Jika seluruh *tuples* di D memiliki kelas yang sama, maka simpul N diubah menjadi daun dan dilabeli dengan nama kelas tersebut. Sebaliknya, jika *tuple-tuple* di D memiliki kelas yang berbeda-beda, maka dipanggil `Attribute_selection_method` untuk menentukan kriteria terbaik dalam mempartisi data dengan menggunakan *attribute selection measures*. Kemudian, simpul N dilabeli dengan *splitting attribute* yang diperoleh dari `Attribute_selection_method` dan sebuah cabang akan dibangkitkan untuk setiap hasil pengujian pada simpul N. Selanjutnya, *tuple-tuple* di D akan dipartisi sesuai dengan hasil pengujian tersebut. Terdapat tiga skenario yang mungkin dalam mempartisi D. Misalkan A adalah *splitting attribute* pada simpul N dan A memiliki sejumlah k nilai berbeda $\{a_1, a_2, ..., a_k\}$ pada *training data*.

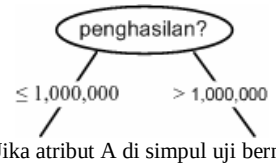
(i) Jika A memiliki nilai-nilai yang diskrit, maka sebuah cabang akan dibentuk untuk setiap nilai A (sehingga total akan terbentuk sebanyak k cabang). Partisi D_i terdiri dari *tuple-tuple* pada D yang memiliki nilai a_i untuk atribut A. Selanjutnya, atribut A dihapus dari `attribute_list`.

(ii) Jika A memiliki nilai-nilai yang kontinu, maka hasil pengujian pada simpul N akan menghasilkan dua cabang, yaitu untuk $A \leq$ *split point* dan $A >$ *split point*. *Split point* merupakan keluaran dari `Attribute_selection_method` sebagai bagian dari kriteria untuk melakukan partisi. Selanjutnya, D dipartisi sehingga D_1 terdiri dari *tuple-tuple* di mana $A \leq$ *split point* dan D_2 adalah sisanya.

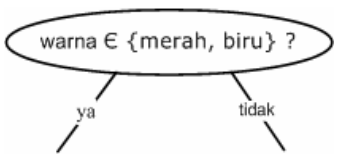
(iii) Jika A memiliki nilai-nilai yang diskrit dan pohon yang dihasilkan harus biner, maka bentuk pengujian di simpul N adalah "$A \in S_A$?" S_A adalah *splitting subset* berupa upahimpunan dari nilai-nilai A. *Splitting subset* diperoleh dari `Attribute_selection_method` sebagai bagian dari kriteria untuk melakukan partisi. Pada umumnya, cabang kiri dilabeli dengan "ya" dan akan menghasilkan D_1 berisi *tuple-tuple* yang memenuhi pengujian. Sebaliknya, cabang kanan dilabeli "tidak" dan menghasilkan D_2 berisi *tuple-tuple* yang tidak memenuhi pengujian.



Gambar 3. Jika atribut A di simpul uji bernilai diskrit.



Gambar 4. Jika atribut A di simpul uji bernilai kontinu.



Gambar 5. Jika atribut A di simpul uji bernilai diskrit dan pohon keputusan yang dihasilkan harus biner.

Algoritma akan melakukan proses yang sama secara rekursif terhadap setiap partisi yang dihasilkan. Proses ini berakhir hanya jika salah satu dari kondisi berikut dipenuhi.

(i) Seluruh *tuples* di D memiliki kelas yang sama.
(ii) Tidak ada lagi atribut yang tersisa di `attribute_list`. Pada kasus ini, simpul N akan diubah menjadi daun dan dilabeli dengan mayoritas kelas di D.
(iii) Tidak terdapat *tuple* di suatu cabang (D_i kosong). Pada kasus ini, sebuah daun dibuat dan dilabeli dengan mayoritas kelas di D.

3.2. *Attribute Selection Measures*

Attribute selection measure adalah sebuah pendekatan heuristik untuk memilih kriteria terbaik dalam mempartisi *training data* ke dalam kelas-kelas. Idealnya, setiap partisi yang dihasilkan harus bersifat *pure*, yang artinya, seluruh *tuples* yang berada di dalam suatu partisi harus memiliki kelas yang sama. Oleh karena itu, kriteria terbaik adalah kriteria yang mampu mempartisi data mendekati *pure*. *Attribute selection measure* akan membuat *ranking* dari atribut-atribut pada *training data*. Atribut yang berada pada peringkat paling ataslah yang dipilih sebagai *splitting attribute*. Jika *splitting attribute* bernilai kontinu, maka *split point* juga akan didefinisikan. Jika *splitting attribute* bernilai diskrit namun pohon keputusan yang dibentuk harus biner, maka *splitting subset* akan didefinisikan. Terdapat tiga jenis attribute selection measures yang banyak digunakan, yaitu information gain, gain ratio, dan gini index. Pada makalah ini, hanya akan dijelaskan mengenai konsep ketiga jenis *measures*, sementara contoh penggunaannya berada di luar lingkup makalah ini dan dapat dilihat pada [2].

Notasi yang digunakan dalam upabab ini adalah sebagai berikut. *D* merupakan partisi yang berisi *training data*. Sebuah atribut yang menyatakan kelas memiliki sejumlah *m* nilai berbeda, yang berarti bahwa terdapat sebanyak *m* kelas yang terdefinisi, C_i (i = 1, …, m). $C_{i,D}$ menyatakan *tuples* di *D* yang memiliki kelas C_i.

3.2.1 *Information Gain*

Attribute selection measure jenis ini digunakan pada ID3. Simpul *N* mewakili *tuples* di dalam *D*. Atribut dengan *information gain* tertinggi akan dipilih sebagai *splitting attribute* pada simpul *N*. Atribut seperti ini diharapkan mampu meminimalkan informasi yang dibutuhkan untuk mengklasifikasi seluruh *tuples* di *D* serta mencerminkan tingkat *impurity* yang rendah pada partisi-partisi yang dihasilkan. Dengan kata lain, jumlah pengujian yang dibutuhkan untuk mengklasifikasi sebuah tuple menjadi berkurang dan pohon keputusan yang dihasilkan pun menjadi lebih sederhana.

Informasi yg dibutuhkan untuk mengklasifikasi sebuah *tuple* di *D* diberikan dengan rumus berikut.

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

di mana p_i adalah peluang bahwa sebuah *tuple* di *D* memiliki kelas *C*. Nilai peluang ini dapat didekati dengan cara menghitung $|Ci,D|/|D|$. *Info*(*D*) hanyalah jumlah rata-rata informasi yang dibutuhkan utk memprediksi kelas dari sebuah *tuple*. Informasi seperti ini hanya bergantung pada jumlah dan proporsi *tuples* dari tiap kelas.

Pada saat akan mempartisi *tuple-tuple* di *D* terhadap atribut *A* yang memiliki *v* nilai berbeda, jika *A* diskrit, akan terbentuk sebanyak *v* hasil pengujian dan *v* partisi di mana D_j adalah partisi yang terdiri dari *tuple-tuple* di *D* yang memiliki nilai a_j untuk atribut *A*. Idealnya, setiap partisi yang dihasilkan akan bersifat *pure*. Namun pada kenyataannya, partisi yang dihasilkan sering *impure*. Oleh karena itu, setelah partisi dilakukan, masih dibutuhkan informasi untuk memperoleh klasifikasi yang *pure* yang dapat diukur dengan rumus berikut.

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

$Info_A(D)$ adalah informasi yang dibutuhkan untuk mengklasifikasi sebuah *tuple* di *D* berdasarkan hasil partisi di *A*. Semakin kecil jumlah informasi yang dibutuhkan ini, semakin tinggi tingkat *purity* dari partisi yang dihasilkan.

Information gain merupakan selisih antara kebutuhan informasi awal (yang hanya bergantung pada jumlah dan proporsi tiap kelas di dalam *D*) dan kebutuhan informasi baru (yang diperoleh setelah melakukan partisi terhadap atribut *A*).

$$Gain(A) = Info(D) - Info_A(D)$$

Gain(*A*) akan menginformasikan seberapa banyak informasi yang didapat dengan melakukan pembagian di *A*. Atribut dengan *Gain*(*A*) terbesar dipilih sebagai *splitting attribute* di simpul *N*. Dengan kata lain, atribut yang terbaik adalah yang meminimalkan jumlah informasi yang dibutuhkan untuk menyelesaikan klasifikasi dari seluruh *tuple* di *D*.

3.2.2 *Gain Ratio*

Pada uraian di atas, dapat dilihat bahwa *information gain* lebih mengutamakan pengujian yang menghasilkan banyak keluaran. Dengan kata lain, atribut yang memiliki banyak nilailah yang dipilih sebagai *splitting attribute*. Sebagai contoh, pembagian terhadap atribut yang berfungsi sebagai *unique identifier*, seperti *product_ID*, akan menghasilkan

keluaran dalam jumlah yang banyak, di mana setiap keluaran hanya terdiri dari satu *tuple*. Partisi semacam ini tentu saja bersifat *pure*, sehingga informasi yang dibutuhkan untuk mengklasifikasi D berdasarkan partisi seperti ini adalah sebesar $Info_{product\_ID}(D) = 0$. Sebagai akibatnya, *information gain* yang dimiliki atribut *product_ID* menjadi maksimal. Padahal, jelas sekali terlihat bahwa partisi semacam ini tidaklah berguna.

Algoritma C4.5 yang merupakan suksesor dari ID3 menggunakan *gain ratio* untuk memperbaiki *information gain*. Pendekatan ini menerapkan normalisasi pada *information gain* dengan menggunakan apa yang disebut sebagai *split information*.

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right)$$

Nilai ini menyatakan jumlah informasi yang dihasilkan akibat pembagian *training data* ke dalam partisi-partisi, berkaitan dengan pengujian yang dilakukan terhadap atribut *A*.

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo(A)}$$

Atribut dengan *gain ratio* maksimal akan dipilih sebagai *splitting attribute*. Perlu diperhatikan bahwa jika *split information* mendekati 0, maka perbandingan tersebut menjadi tidak stabil. Oleh karena itu, perlu ditambahkan batasan untuk memastikan bahwa *information gain* dari sebuah pengujian haruslah besar, dan minimal sama besar dengan *information gain* rata-rata dari seluruh pengujian.

3.2.4 Gini *Index*

Attribute selection measure jenis ini digunakan pada algoritma CART. Gini *index* akan menghasilkan pembagian yang bersifat biner pada setiap atribut, baik yang memiliki nilai diskrit ataupun kontinu.

Gini *index* mengukur *impurity* dari suatu partisi, D, dengan rumus berikut.

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2$$

di mana p_i adalah peluang bahwa sebuah *tuple* di *D* berada pada kelas C_i. Peluang tersebut dapat didekati dengan hasil perhitungan $|C_i,D|/|D|$ di mana. $|C_i,D|$ merupakan jumlah *tuple* pada *D* yang memiliki kelas C_i dan $|D|$ adalah jumlah seluruh *tuple* di D. Perhitungan ini dilakukan untuk setiap kelas.

Misalkan *A* merupakan atribut bernilai diskrit yang memiliki sejumlah *v* nilai berbeda, $\{a_1, a_2, ..., a_n\}$. Untuk menentukan kriteria pembagian terbaik terhadap *A*, seluruh upahimpunan dari *A* harus diperiksa. Setiap upahimpunan, S_A, dapat dijadikan sebagai *splitting subset* untuk pengujian dalam bentuk "$A \in S_A$?". Sebuah *tuple* memenuhi pengujian jika nilai untuk atribut *A* pada *tuple* tersebut merupakan bagian dari S_A. Dengan tidak mempertimbangkan himpunan kuasa dan himpunan kosong, maka akan terdapat $2^v - 2$ cara untuk melakukan pembagian biner dari *D*.

Pemeriksaan sebuah pembagian biner dilakukan dengan cara menjumlahkan *impurity* dari setiap partisi yang dihasilkan oleh pembagian tersebut. Misalkan sebuah pembagian yang dilakukan terhadap atribut *A* mempartisi *D* menjadi D_1 dan D_2. Gini *index* dari *D* dapat dihitung dengan rumus berikut.

$$Gini_A(D) = \frac{|D_1|}{|D|}Gini(D_1) + \frac{|D_2|}{|D|}Gini(D_2)$$

Untuk atribut bernilai diskrit, upahimpunan yang memberikan nilai gini *index* terkecil untuk atribut *A* akan dipilih sebagai *splitting subset*. Seluruh pembagian biner yang mungkin terjadi pada suatu atribut harus diperiksa.

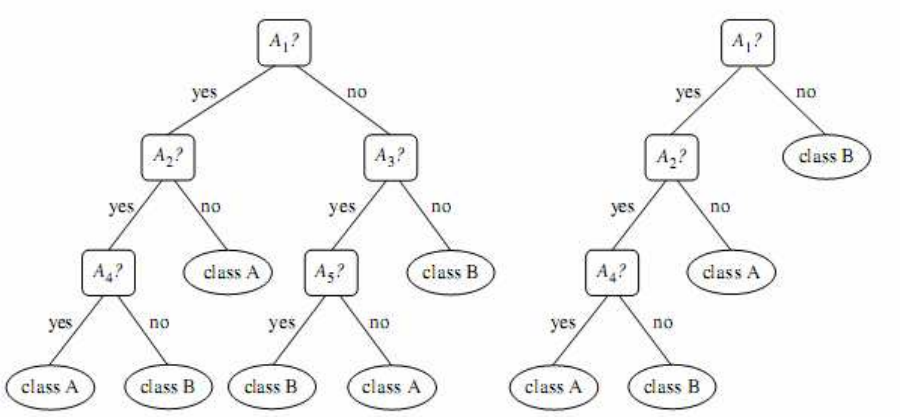
Sementara untuk atribut bernilai kontinu, setiap *split point* yang mungkin harus diperiksa. Untuk nilai-nilai suatu atribut yang telah diurutkan, titik tengah di antara setiap pasangan nilai yang saling berseberangan dapat diambil sebagai sebuah *split point*. Titik yang memberikan nilai gini *index* terkecil untuk suatu atributlah yang akhirnya diambil sebagai *split point*.

Penurunan tingkat *impurity* yang diperoleh dari sebuah pembagian biner terhadap atribut *A* dapat dihitung dengan rumus berikut.

$$\Delta Gini(A) = Gini(D) - Gini_A(D)$$

Atribut yang memaksimalkan penurunan tingkat *impurity* inilah yang dipilih sebagai *splitting attribute*. Atribut ini, bersama dengan *splitting subset* (jika atribut tersebut bernilai diskrit) atau *split point* (jika atribut tersebut bernilai kontinu) akan membentuk kriteria pembagian.

3.3. Pemangkasan Pohon



Gambar 6. Pohon keputusan sebelum dan setelah dipangkas.

Pada saat pembangunan pohon keputusan, banyaknya cabang mungkin mencerminkan adanya *noise* atau *outlier* pada *training data*. Pemangkasan pohon dapat dilakukan untuk mengenali dan menghapus cabang-cabang tersebut. Pohon yang dipangkas akan menjadi lebih kecil dan lebih mudah dipahami. Pohon semacam itu biasanya juga menjadi lebih cepat dan lebih baik dalam melakukan klasifikasi terhadap *unknown data*. Terdapat dua pendekatan utama dalam pemangkasan pohon: *prepruning* dan *postpruning*.

Pada pendekatan *prepruning*, sebuah pohon dipangkas dengan cara menghentikan pembangunannya jika partisi yang akan dibuat dianggap berada di bawah batasan tertentu. Kesulitan terbesar pada pendekatan ini adalah dalam menentukan batasan.

Pada pendekatan *postpruning*, upapohon dipangkas dari pohon dewasa. Upapohon dipangkas dengan cara menghapus cabang-cabangnya serta mengubah upapohon tersebut menjadi sebuah simpul yang dilabeli dengan kelas mayoritas pada upapohon tersebut.

4. KESIMPULAN

Kesimpulan yang dapat diambil dari pembahasan di dalam makalah ini adalah:

1. Klasifikasi adalah suatu fungsional *data mining* yang menghasilkan model untuk memprediksi kelas dari objek-objek pada basisdata.
2. Model klasifikasi dapat digambarkan dalam bentuk pohon keputusan di mana simpul dalam menyatakan atribut penguji, panah menyatakan hasil pengujian, dan daun menyatakan kelas hasil prediksi.
3. Pada dasarnya, algoritma pembangun pohon keputusan memiliki karakteristik yang sama, yaitu *top-down* (pohon keputusan dibangun dari simpul akar ke daun) dan *divide-and-conquer* (*training data* secara rekursif dipartisi ke dalam bagian-bagian yang lebih kecil).
4. *Attribute selection measures* digunakan untuk menentukan kriteria terbaik dalam membagi *training data* ke dalam kelas-kelas. Beberapa contoh *attribute selection measures* antara lain *information gain*, *gain ratio*, dan gini *index*.
5. Pemangkasan pohon dapat dilakukan untuk menghilangkan cabang-cabang tidak perlu yang terbentuk akibat adanya *noise* atau *outlier* pada *training data*.

DAFTAR REFERENSI

[1] Han, Jiawei, Micheline Kamber, *Data Mining Concepts and Techniques* (*2nd edition*), Morgan Kaufmann, 2006.

[2] Munir, Rinaldi, *Diktat Kuliah IF2151 Matematika Diskrit* (*edisi keempat*), Institut Teknologi Bandung, 2004.