SKRIPSI

DATA MINING HISTORI PENCARIAN RUTE ANGKOT

JOVAN GUNAWAN

NPM: 2011730029

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN
2014

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Pertumbuhan teknologi hingga saat ini telah menghasilkan banyak sekali data-data, namun sering kali pemilik data hanya menggunakan data tersebut seperlunya saja. Jika dilihat lebih rinci, sebenarnya jika data tersebut diolah lebih lanjut, dapat menghasilkan sesuatu yang lebih. Salah satu cara mengolah data tersebut adalah dengan menggunakan teknik *data mining*. Dengan menggunakan teknik *data mining* akan mempermudah menganalisa masalah, pengambilan kesimpulan, bahkan mempermudah konsumen dalam membeli jasa atau barang.

Tujuan utama dari *data mining* adalah *knowledge*. *Knowledge* merupakan suatu informasi yang berharga dan dapat dijadikan landasan untuk menganalisa atau membuat kesimpulan. Untuk mendapatkan *knowledge*, dapat dilakukan dengan cara melakukan pencarian *pattern* atau pola yang merupakan salah satu tahap dari *data mining*. Pola inilah yang akan memperlihatkan data manakah yang menarik dan dapat dijadikan *knowledge* yang akan digunakan untuk menganalisa data tersebut.

Pada penelitian *data mining* ini, penulis memiliki data *log* histori KIRI selama 1 bulan. Data tersebut akan diimplementasikan proses *data mining* untuk mendapatkan *pattern* dan *knowledge* yang terkandung pada data *log* KIRI. Data *log* tersebut memiliki 5 *field* untuk setiap *entry* sebagai berikut:

- statisticId, primary key dari entry
- verifier, mengidentifikasikan sumber dari pencarian ini
- *timestamp*, waktu ketika pengguna KIRI mencari rute angkot
- *type*, tipe fungsi yang digunakan
- additionalInfo, mencatat koordinat awal, koordinat akhir, dan banyak rute yang ditemukan pada pencarian ini

Berdasarkan hal diatas, penulis ingin mendapatkan pola yang menarik dan menghasilkan *knowledge* yang berguna dan dapat dipakai baik untuk KIRI ataupun pemerintah.

1.2 Perumusan Masalah

Dengan mengacu pada uraian diskripsi diatas, maka permasalahan yang dibahas dan diteliti oleh penulis adalah

- Bagaimana cara mengolah pola yang diperoleh dari *data log histori* KIRI agar pola menjadi menarik dan bermakna?
- Bagaimana membuat perangkat lunak untuk melakukan data mining pada data log history?

1.3 Tujuan

Penelitian ini bertujuan untuk

- Mencari pola dan informasi yang menarik dari *log histori* KIRI
- Perangkat lunak dapat melakukan data mining dari *log histori* KIRI

1.4 Batasan Masalah

Penelitian *data mining* yang diatas akan ditentukan batasan masalah yang diteliti berupa :

- Penelitian ini dibatasi hanya pada permasalahan pada penerapan *data mining* pada *data log* KIRI
- Data log yang merupakan masukan akan dibatasi sebanyak 10000 buah data

1.5 Metode Penelitian

Berikut adalah Metode Penelitian yang digunakan :

- Melakukan studi literatur tentang algoritma-algoritma yang berkaitan dengan pemrosesan *data mining*
- Melakukan penelitian *data mining* yang diterapkan pada *log* KIRI
- Merancang dan mengimplementasikan algoritma untuk *data mining*
- Mengimplementasikan pembangkit pola *data mining*
- Melakukan pengujian dan eksperimen

1.6 Sistematika Pembahasan

Sitematika pembahasan dalam penelitian ini adalah: Bab 1: Pendahuluan, berisi latar belakang dari penelitian ini, rumusan masalah yang timbul, tujuan yang ingin dicapain, ruang lingkup atau batasan masalah dari penelitian ini, serta metode penelitian yang akan digunakan dan sistematika pembahasan dari penelitian ini. Bab 2: Landasan Teori, berisi dasar teori mengenai *data mining*

dan *log* histori KIRI Bab 3: Analisa dasar teori yang akan digunakan untuk merancang aplikasi *data mining log* histori KIRI Bab 4: perancangan aplikasi *data mining log* histori KIRI Bab 5: kesimpulan

BAB 2

LANDASAN TEORI

2.1 *Data Mining*

Data mining merupakan merupakan proses yang melakukan pengambilan inti sari atau penggalian *knowledge* dari data yang besar dan merupakan salah satu langkah dari *knowledge discovery*.

Menurut [1], *knowledge discovery* dapat dibagi menjadi 7 tahap (gambar 2.1):

1. *Data cleaning*
2. *Data integration*
3. *Data selection*
4. *Data transformation*
5. *Data mining*
6. *Pattern Evaluation*
7. *Knowledge presentation*

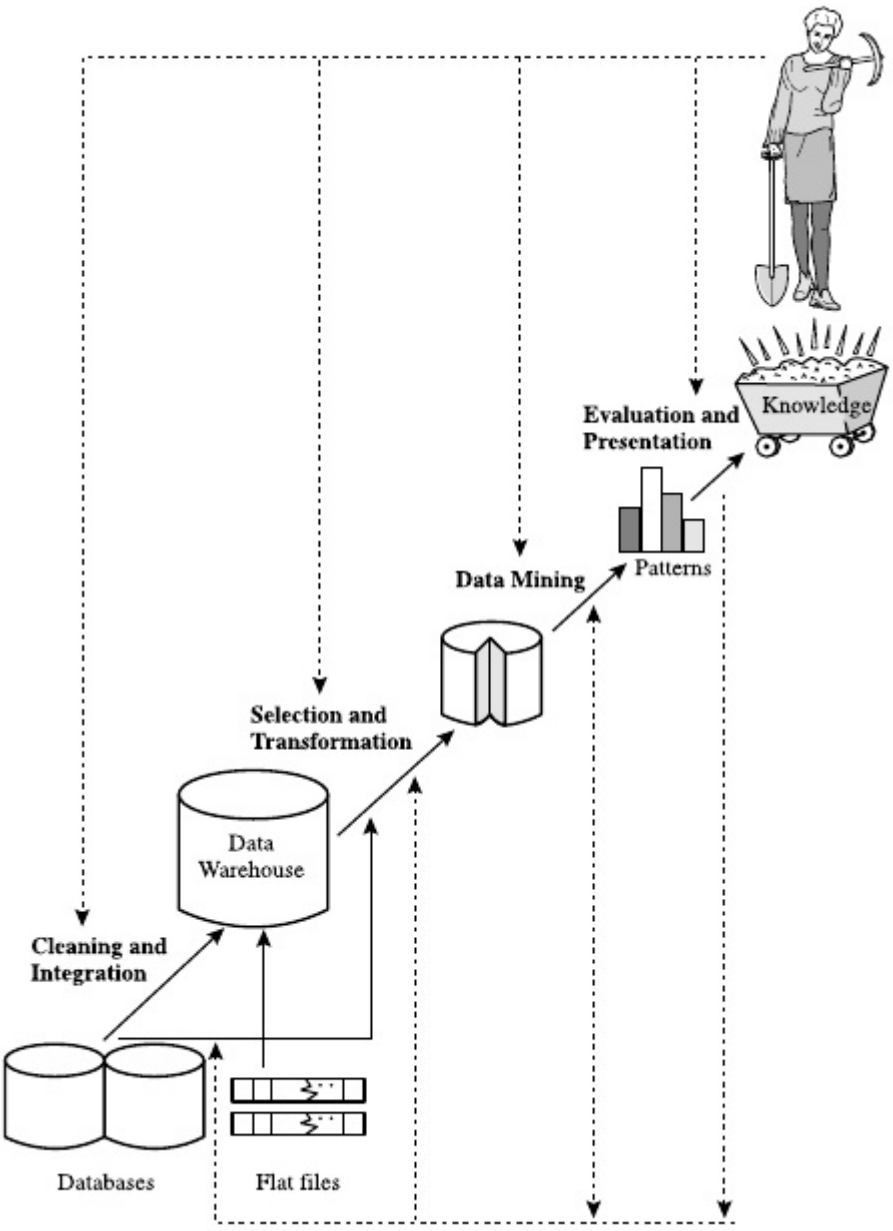
Tahap pertama hingga keempat merupakan bagian dari *data preprocessing*, dimana data-data disiapkan untuk dilakukan penggalian data. Tahap *data mining* merupakan tahap dimana melakukan penggalian data. Tahap keenam merupakan tahap pencarian pola yang merepresentasikan *knowledge*. Sedangkan tahap terakhir merupakan visualisasi dan representasi dari *knowledge* yang sudah diperoleh dari tahap sebelumnya.

2.1.1 *Data Cleaning*

Data cleaning merupakan tahap *data mining* untuk menghilangkan *missing value* dan *noisy data*. Pada umumnya, *data* yang diperoleh dari *database* terdapat nilai yang tidak sempurna seperti nilai yang hilang, nilai yang tidak valid atau salah ketik. Atribut dari suatu *database* yang tidak relevan atau redudansi bisa diatasi dengan menghapus atribut tersebut.

Missing Values

Missing values akan mengganggu proses *data mining* pada komputer dan dapat menghasilkan nilai akhir yang tidak sesuai. Terdapat beberapa teknik untuk mengatasi *missing values* yaitu



Gambar 2.1: Tahap *Data Mining*, [1]

- Membuang tuple yang mengandung nilai yang hilang
- Mengisi nilai yang hilang secara manual
- Mengisi nilai yang hilang dengan menggunakan nilai konstan yang bersifat umum
- Menggunakan nilai rata-rata dari suatu atribut untuk mengisi nilai yang hilang

Noisy Data

Noisy data merupakan nilai yang berasal dari error atau tidak valid. *Noisy data* dapat dihilangkan dengan menggunakan teknik *smoothing*. Terdapat 3 metode untuk menghilangkan *noisy data* yaitu

- *Binning*, merupakan metode pengisian data sesuai dengan proses yang dilakukan pada data tersebut
- *Regression*, merupakan metode yang mencari detail persamaan atribut untuk memprediksikan suatu nilai
- *Clustering*, merupakan metode pengelompokan dimana ditemukan *outliers* yang dapat dibuang

2.1.2 *Data Integration*

Data integration merupakan tahap menggabungkan data dari berbagai sumber. Sumber tersebut bisa termasuk beberapa *database*, *data cubes*, atau bahkan *flat data*. *Data cube* merupakan teknik pengambilan data-data dari *data warehouse* dan dilakukan operasi agregasi sesuai dengan kondisi tertentu (contoh, penjumlahan total penjualan per tahun dari 2005-2010). Sedangkan *flat data* merupakan data yang disimpan dengan cara apapun untuk merepresentasikan database model pada sebuah data baik berbentuk *plain text file* maupun *binary file*.

Tahap ini harus dilakukan secara teliti terutama ketika dalam memasangkan nilai-nilai yang berasal dari sumber yang berbeda. Pada tahap ini, perlu dilakukan identifikasi data apakah data tersebut dapat diturunkan atau tidak agar data yang diperoleh tidak terlalu besar. *Data integration* yang baik merupakan integrasi yang dapat memaksimalkan kecepatan dan meningkatkan akurasi dari proses *data mining*. Contoh studi kasus dari *data integration*, jika suatu perusahaan sepatu A memiliki dua pabrik dengan *database* lokal pada masing-masing pabrik, jika akan dilakukan *data mining* pada kedua *database* tersebut, maka kedua *database* akan digabung dan perlu diperhatikan serta diperbaiki nilai-nilai seperti *primary key*, atribut, dan lain-lain agar tidak terjadi *error* pada *database* yang sudah digabung. Proses dari penggabungan hingga perbaikan nilai-nilai pada kedua database tersebut adalah proses *data integration*.

2.1.3 *Data Selection*

Proses dimana data-data yang relevan dengan analisis akan diambil dari database dan data yang tidak relevan akan dibuang. Sebagai contoh kasus, jika akan dilakukan analisa mengenai nilai mahasiswa dalam satu semester, atribut pada tabel nilai sebagai berikut

- NPMMahasiswa

- NamaMahasiswa
- JenisKelamin
- Alamat
- MataKuliah
- NilaiART
- NilaiUTS
- NilaiUAS

Maka, atribut yang berpotensi diambil adalah MataKuliah, NilaiART, NilaiUTS, NilaiUAS, sedangkan atribut yang akan dibuang adalah NPMMahasiswa, NamaMahasiswa JenisKelamin, dan Alamat karena tidak terlalu berhubungan dengan analisa.

2.1.4 *Data Transformation*

Data transformation merupakan tahap pengubahan data agar siap dilakukan proses *data mining*. *Data transformation* bisa melibatkan:

- *Smoothing*, proses untuk membuang *noise* seperti yang dilakukan pada tahap *data cleaning*
- *Aggregation*, proses mengganti nilai-nilai menjadi suatu nilai yang dapat mewakili nilai sebelumnya
- *Generalization*, proses dimana membuat suatu nilai yang bersifat khusus menjadi nilai yang bersifat umum
- *Normalization*, proses dimana suatu nilai dapat diubah skalanya menjadi nilai yang lebih kecil dan spesifik
- *Attribute construction*, proses membuat atribut baru yang berasal dari beberapa atribut untuk membantu proses data mining

Normalization

Atribut dapat dinormalisasi dengan memberi skala pada nilainya sehingga nilai tersebut menjadi suatu range yang lebih spesifik dan kecil seperti 0,0 sampai 1,0. Dua teknik nnormalisasi yaitu, *min-max normalization* dan *z-score normalization*. *Min-max normalization* akan mengubah semua nilai menjadi nilai dengan skala tertentu. Dengan menggunakan rumus

$$\nu' = \frac{\nu - min_A}{max_A - min_A}(newMax_A - newMin_A) + newMin_A$$

Contoh kasus, misalkan nilai minimun dan maximum dari suatu pendapatan adalah 12.000 dan 98.000, akan diubah menjadi berskala antara 0,0 sampai 1,0. Jika ada nilai pendapat yang baru, yaitu 73.600, maka akan menjadi

$$\frac{73.600 - 12.000}{98.000 - 12.000}(1,0 - 0) + 0 = 0,716$$

z-score normalization merupakan normalisasi berdasarkan nilai rata-rata dan standar deviasi dari nilai-nilai atribut dengan cara

$$\nu' = \frac{\nu - \overline{A}}{\sigma_A}$$

Contoh kasus, misal nilai rata-rata dan standar deviasi dari nilai-nilai atribut pendapatan adalah 54.000 dan 16.000. Dengan *z-score*, jika ada nilai pendapatan baru yaitu 73600, maka akan diubah menjadi

$$\frac{73.600 - 54.000}{16.000} = 1,225$$

Attribute Construction

Attribute Construction merupakan teknik menambahkan atribut baru yang berdasarkan dari atribut yang sudah ada guna menambah akurasi. Contoh kasus, dibuat atribut baru bernama area berdasarkan atribut panjang dan lebar.

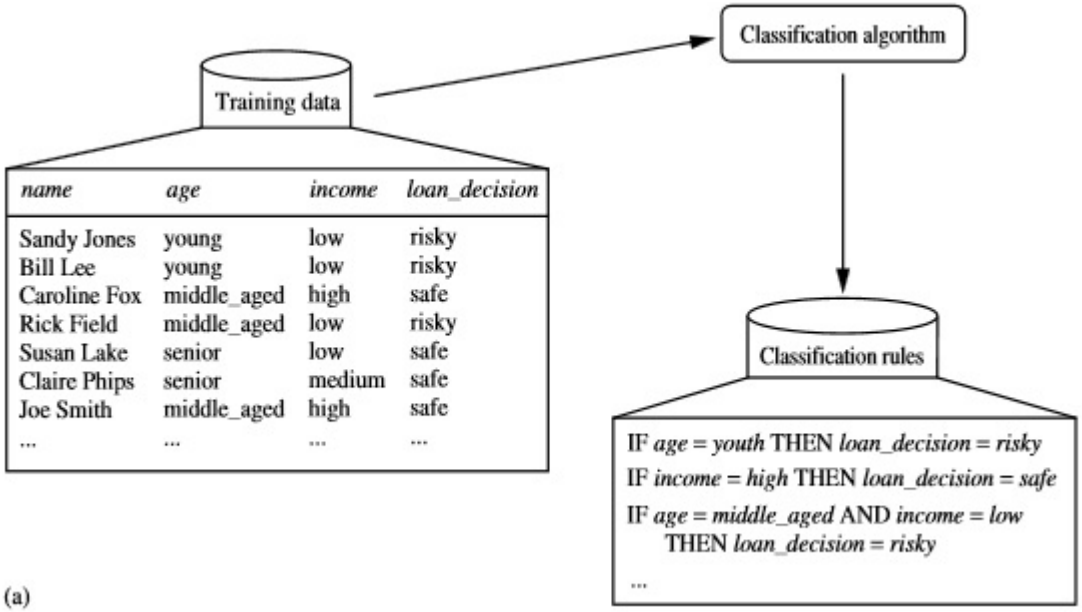
2.1.5 *Data Mining*

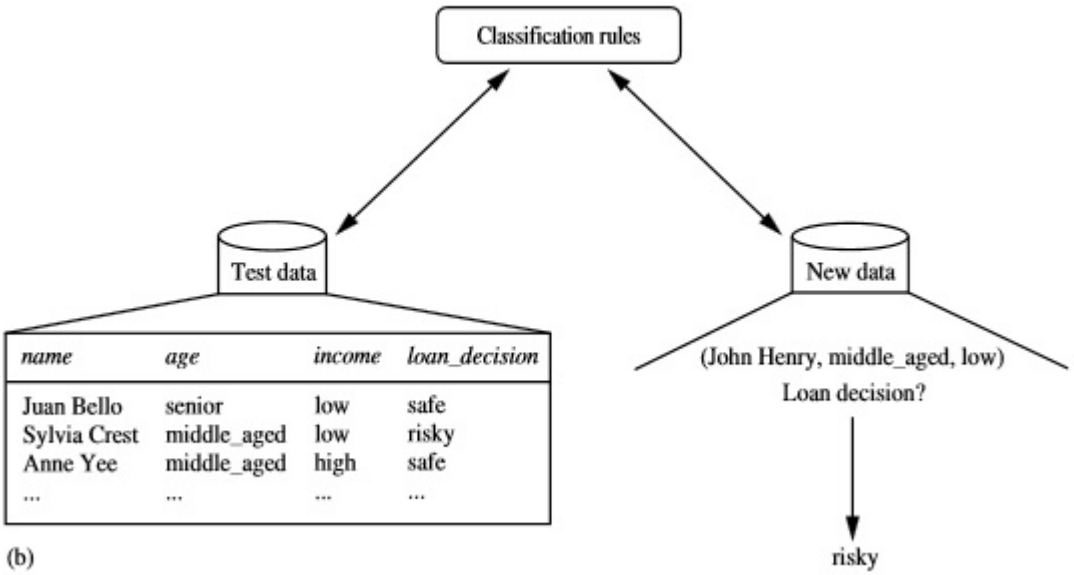
Classification and Prediction

Classification merupakan pemodelan yang dibangun untuk memprediksikan label kategori, seperti "'baik"', "'cukup"', dan "'buruk"' dalam sistem penilaian sikap seorang siswa atau "'mini bus"', "'bus"', atau "'sedan"' dalam kategori tipe mobil. Kategori tersebut dapat direpresentasikan dengan menggunakan nilai diskret. Nilai diskret merupakan nilai yang terpisah dan berbeda, seperti 1 atau 5. Kategori yang direpresentasikan oleh nilai diskret maka akan menjadi nilai yang terurut dan tidak memiliki arti, seperti 1,2,3 untuk merepresentasikan kategori tipe mobil "'mini bus"', "'bus"', dan "'sedan"'.

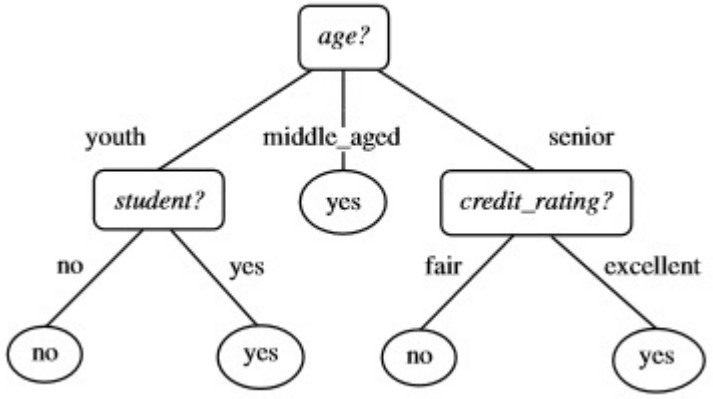
Prediction merupakan model yang dibangun untuk meramalkan fungsi nilai kontinu atau *ordered value*. *Ordered value* merupakan nilai yang terurut dan berlanjut. Contoh studi kasus untuk pemodelan prediction adalah seorang marketing ingin meramalkan seberapa banyak konsumen yang akan belanja di sebuah toko dalam waktu satu bulan. Pemodelan tersebut disebut *predictor*. *Regression Analysis*, merupakan metodologi statistik yang digunakan untuk *numeric prediction*. *Classification* dan *numeric prediction* merupakan dua jenis utama dalam masalah prediksi.

Data Classification merupakan proses untuk melakukan klasifikasi. *Data classification* memiliki dua tahap proses, yaitu *learning step* dan tahap klasifikasi seperti pada ilustrasi di gambar 2.2. *Learning step* merupakan langkah pembelajaran, di mana algoritma klasifikasi membangun *classification rules* (yang berisi syarat atau aturan sebuah nilai masuk ke dalam kategori tertentu) dengan cara menganalisis *training set* yang merupakan *database tuple*. Karena pembuatan *classification rules* menggunakan *training set*, yang dikenal juga sebagai *supervised learning*. Pada tahap kedua, dilakukan proses klasifikasi nilai berdasarkan *classification rules* yang sudah dibangun dari tahap pertama.





Gambar 2.2: Tahap *data classification*, [1]



Gambar 2.3: Contoh *decision tree*, [1]

2.1.6 *Decision Tree*

Salah satu cara pembuatan *classification rules* pada *Data Classification* adalah dengan membuat *decision tree* (pohon keputusan). *Decision tree* merupakan *flowchart* yang berbentuk pohon, dimana setiap node internal (*nonleaf* node) merupakan hasil test dari atribut, setiap cabang merepresentasikan output dari test, dan setiap node daun memiliki *class label*. Bagian paling atas dari pohon disebut *root node*. Contoh studi kasus, pohon keputusan untuk menentukan apakah seorang konsumen akan membeli komputer atau tidak (ilustrasi pohon keputusan pada gambar 2.3)

Decision Tree Induction

Decision tree induction merupakan pelatihan pohon keputusan dari tupel pelatihan kelas berlabel. Terdapat tiga teknik untuk membuat *decission tree* diantaranya adalah ID3 dan C4.5. Teknik tersebut menggunakan pendekatan *greedy* yang merupakan *decission tree* yang dibangun secara *top-down recursive divide and conquer*. Algoritma yang diperlukan secara umum sama, hanya berbeda pada *attribute_selection_method*. Berikut algoritma untuk membuat pohon keputusan dari suatu tupel pelatihan.

Input:

- Partisi data, D, merupakan set data pelatihan dan kelas label
- *attribute_list*, merupakan set dari atribut kandidat
- *Attribute_selection_method*, prosedur untuk menentukan *splitting criterion*. Pada input ini, terdapat juga data *splitting_attribute* dan mungkin salah satu dari *split point* atau *splitting subset*

Output: pohon keputusan
Method:
(1) create a 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 leaf node labeled with the majority class in D; //majority voting

(6) apply Attribute_selection_method(D, atribute_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 //not restricted to binary trees

(9) attribute_list <- attribute_list - splitting_attribute; //remove splitting_attribute

(10)for each outcome j of splitting_criterion // partition the tuples and from subtrees for each partition

(11)let Dj be the set of data tuples in D satisfying outcome j; //a partition

(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;

endfor

(15) return N;

Pohon keputusan akan dimulai dengan satu node, yaitu N, merepresentasikan tuple pelatihan pada D (langkah 1)

Jika tuple di D memiliki kelas yang sama semua, maka node N akan menjadi daun dan diberi label dari kelas tersebut (langkah 2 dan 3). Perlu diketahui bahwa langkah 4 dan 5 akan mengakhiri kondisi.

Jika tuple di D ada kelas yang berbeda, maka algoritma akan memanggil *attribute_selection_method* untuk menentukan *splitting criterion. Splitting criterion* akan menentukan atribut pada node N yang merupakan nilai terbaik untuk memecah nilai atribut pada tuple ke dalam kelas masing-masing. (langkah 6)

Node N akan diisi dengan hasil dari *splitting criterion* (langkah 7). Kemudian kriteria tersebut agak dibentuk cabangnya masing-masing sesuai pada langkah 10 dan 11. Terdapat tiga kemungkinan bentuk kriteria jika A merupakan *splitting_attribute* yang memiliki nilai unik seperti {a1, a2, ..., av} seperti pada gambar 2.4, yaitu,

1. *Discrete valued*: cabang yang dihasilkan memiliki kelas dengan nilai diskret. Karena kelas yang dihasilkan diskret dan hanya memiliki nilai yang sama pada cabang tersebut, maka *attribut_list* akan dihapus (langkah 8 dan 9)

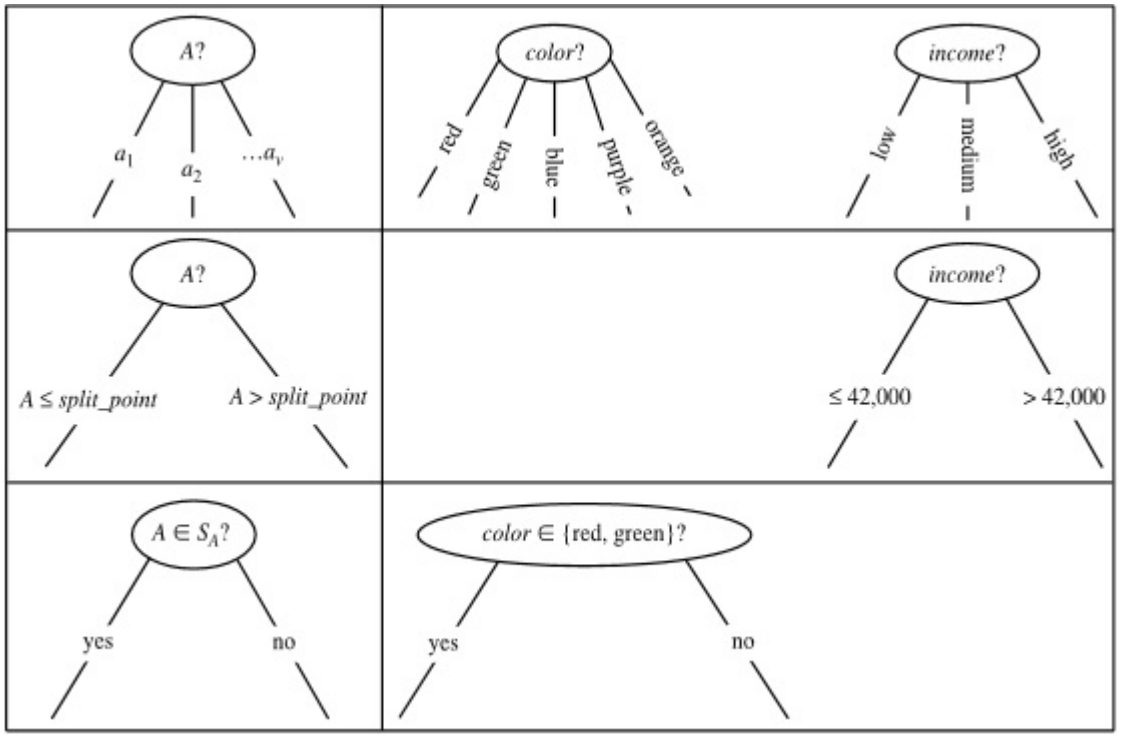
2. *Continuous values*: cabang yang dihasilkan memiliki jarak nilai untuk memenuhi suatu kondisi (contoh: A <= split_point), dimana nilai *split_point* adalah nilai pembagi yang dikembalikan oleh *attribute_selection_method*

3. *Dicrete valued and a binary tree*: cabang yang dihasilkan adalah dua berupa nilai iya atau tidak dari "'apakah A anggota Sa"', dimana Sa merupakan subset dari A, yang dikembalikan oleh *Attribute_selection_method*

Kemudian, akan dipanggil kembali algoritma *decision tree* untuk setiap nilai hasil pembagian pada tuple, Dj (langkah 14).

Rekursif tersebut akan berhenti ketika salah satu dari kondisi terpenuhi, yaitu

1. Semua tuple pada partisi D merupakan bagian dari kelas yang sama.



Gambar 2.4:]

Jenis-jenis *split point*, [1]

2. Sudah tidak ada atribut yang dapat dilakukan pembagian lagi (dilakukan pengecekan pada langkah 4). Disini, akan dilakukan *majority voting* (langkah 5) yang akan mengkonversi node N menjadi *leaf* dan diberi label dengan kelas yang terbanyak pada D.

3. Sudah tidak ada tuple yang dapat diberi cabang, Dj sudah kosong (langkah 12) dan *leaf* akan dibuat dengan *majority class* pada D (langkah 13).

Pada langkah 15, akan dikembalikan nilai *decision tree* yang telah dibuat.

subsubsection*Attribute Selection Measure*

Attribute Selection Measure merupakan suatu hirarki untuk pemilihan *splitting criterion* yang terbaik yang memisah partisi data (D), tuple pelatihan kelas label ke dalam kelas masing-masing. *Attribute Selection Measure* menyediakan peringkat untuk setiap atribut pada training tuple. Jika *splitting criterion* merupakan nilai *continous* atau *binary trees*, maka nilai *split point* dan *splitting subset* harus ditentukan sebagai bagian dari *splitting criterion*. Contoh dari *attribute selection measure* adalah *information gain*, *gain ratio*, dan *gini index*.

Notasi yang digunakan adalah sebagai berikut. D merupakan data partisi, set pelatihan dari *class-labeled* tuple. Jika label kelas atribut memiliki m nilai yang berbeda yang mendifinisikan m kelas yang berbeda, Ci (for i=1,...,m). Ci,d menjadi kelas tuple dari Ci di D. |D| dan |Ci,d| merupakan banyak tuple pada D dan Ci,d.

Information Gain

Information menurut Claude Shannon dalam *information theory* adalah ukuran *pure* dari suatu data. ID3 menggunakan *information gain* sebagai *attribute selection measure* yang melakukan

pemilihan atribut berdasarkan informasi yang terkandung dalam pesan. Cara ID3 mendapatkan *information gain* dengan menggunakan *entropy*. *Entropy* adalah ukuran *impurity* dari suatu data. Cara mendapatkan nilai *entropy* adalah

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

Dimana pi merupakan probabilitas tuple pada D terhadap class Ci, dapat diperoleh dengan |ci,d|/|D|. Info(D) merupakan nilai rata-rata *entropy* dari suatu label kelas pada tuple D. Untuk mengetahui atribut mana yang paling baik untuk dijadikan *splitting attribute*, adalah dengan cara menghitung nilai *entrophy* dari suatu atribut kemudian diselisihkan dengan nilai *entropy* dari D. Jika pada tuple D, memiliki atribut A dengan v nilai yang berbeda, maka menghitung *entropy* dari suatu atribut adalah

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

|Dj|/D merupakan angka yang menghitung bobot dari suatu partisi. Semakin kecil nilai dari InfoA(Dj), maka atribut tersebut masih memerlukan informasi, semakin besar nilai InfoA(Dj), semakin tinggi pula tingkat *pure* dari suatu partisi.

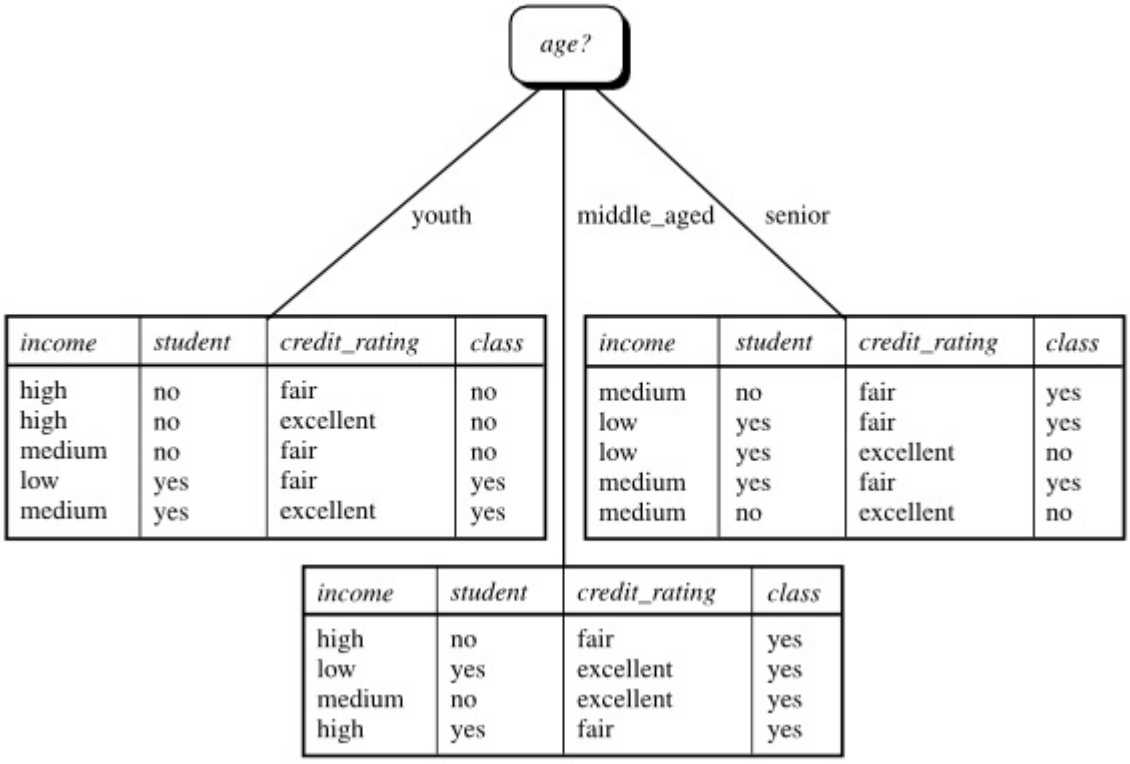
Setelah mendapatkan nilai Info(D) dan InfoA(Dj), *information gain* dapat diperoleh dari selisih nilai Info(D) dan InfoA(Dj)

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

contoh kasus untuk ID3, dalam pencarian*information gain*

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

Pada tabel 2.1, terdapat *training set*, D. Atribut kelas label merupakan dua nilai yang berbeda yaitu *yes* atau *no*, maka dari itu, nilai m = 2. C1 diisi dengan kelas label bernilai *yes*, sedangkan C2 diisi dengan kelas label bernilai *no*. Terdapat sembilan tuple atribut kelas label dengan nilai *yes* dan lima tuple dengan nilai *no*. Untuk dapat menentukan *splitting criterion*, *information gain* harus dihitung untuk setiap atribut terlebih dahulu. Perhitungan *entropy* untuk D adalah



income	*student*	*credit_rating*	*class*
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

income	*student*	*credit_rating*	*class*
medium	no	fair	yes
low	yes	fair	yes
low	yes	excellent	no
medium	yes	fair	yes
medium	no	excellent	no

income	*student*	*credit_rating*	*class*
high	no	fair	yes
low	yes	excellent	yes
medium	no	excellent	yes
high	yes	fair	yes

Gambar 2.5: Hasil cabang dari atribut *age*, [1]

$$Info(D) = -\frac{9}{14}\log 2(\frac{9}{14}) - \frac{5}{14}\log_2(\frac{5}{14}) = 0.940bits$$

Setelah diperoleh nilai *entropy* dari D, kemudian akan dihitung nilai *entropy* atribut dimulai dari atribut age. Pada kategori *youth*, terdapat dua tuple dengan kelas *yes* dan tiga tuple dengan kelas *no*. Untuk kategori *middle_ age*, terdapat empat tuple dengan kelas *yes* dan nol tuple dengan kelas *no*. Pada kategori *senior*, terdapat tiga dengan kelas *yes* dan dua dengan kelas *no*. Perhitungan nilai *entropy* atribut *age* terhadap D sebagai berikut

$$Info_{age}(D) = \frac{5}{14}\times(-\frac{2}{5}\log_2\frac{2}{5}-\frac{3}{5}\log_2\frac{3}{5})+\frac{4}{14}\times(-\frac{4}{4}\log_2\frac{4}{4}-\frac{0}{4}\log_2\frac{0}{4})+\frac{5}{14}\times(-\frac{3}{5}\log_2\frac{3}{5}-\frac{2}{5}\log_2\frac{2}{5}) = 0.694bits$$

Setelah mendapatkan *entropy* dari atribut *age*, maka nilai *gain information* dari atribut *age* adalah

$$Gain_{(age)} = Info(D) - Info_{age}(D) = 0.940 - 0.694 = 0.246bits$$

Dengan melakukan hal yang sama, dapat diperoleh nilai *gain* untuk atribut *income* adalah 0.029 *bits*, untuk nilai *gain(student)* adalah 0.151 *bits*, dan *gain(credit_ rating)* = 0.048 *bits*. Karena nilai *gain* dari atribut *age* merupakan nilai terbesar diantara semua atribut, maka atribut *age* dipilih menjadi *splitting attribute*. Setelah ditentukan, node N akan membentuk cabang berdasarkan nilai dari atribut *age* seperti pada gambar 2.5.

Untuk atribut yang merupakan nilai *continuous*, harus dicari nilai *split point* untuk A. Nilai-nilai

dari dua angka yang bersebelahan dapat diambil nilai tengahnya untuk dijadikan *split-point*. Jika terdapat v nilai yang berbeda dari A, maka akan terdapat v-1 kemungkinan *split point*. Kemudian nilai *split point* akan dijadikan sebagai nilai pembagi, sebagai contoh: A <= *split-point* merupakan cabang pertama, dan A > *split-point* merupakan cabang kedua.

2.1.7 *Pattern Evaluation*

Pattern evaluation merupakan tahap mengidentifikasi apakah *pattern* atau pola tersebut menarik dan merepresentasikan *knowledge* berdasarkan beberapa *interestingness measures*. Suatu *pattern* atau pola dapat dinyatakan menarik apabila

- mudah dimengerti oleh manusia
- valid untuk data percobaan maupun data yang baru
- memiliki potensi atau berguna
- merepresentasikan *knowledge*

2.1.8 *Knowledge Presentation*

Knowledge presentation merupakan tahap representasi dan visualisasi terhadap *knowledge* yang merupakan hasil dari *knowledge discovery*.

2.2 Log Histori KIRI

KIRI memiliki log histori yang melakukan pencatatan untuk setiap user ketika menggunakan KIRI. *Log* tersebut memiliki 5 *field* untuk setiap *entry* sebagai berikut:

- logId, primary key dari entry
- APIKey, mengidentifikasikan sumber dari pencarian ini
- *Timestamp* (UTC), waktu ketika pengguna KIRI mencari rute angkot menggunakan waktu UTC / GMT
- *Action*, tipe dari log yang dibuat.
- AdditionalData, mencatat koordinat awal, koordinat akhir, dan banyak rute yang ditemukan pada pencarian ini

LogId merupakan *field* dengan tipe data int dengan batas 6 karakter yang digunakan sebagai *primary key* dari tabel tersebut. LogId diisi dengan menggunakan fungsi *increment integer*. *Increment integer* merupakan fungsi untuk pengisian data pada database dengan menambahkan nilai 1 dari nilai yang terakhir kali diisi. APIKey merupakan *field* dengan tipe data varchar yang digunakan untuk memeriksa pengguna KIRI ketika menggunakan KIRI. *Timestamp* (UTC) merupakan *field* dengan tipe data *timestamp* yang digunakan untuk mencatat waktu penggunaan KIRI oleh user, diisi dengan menggunakan fungsi *current time*. *Current time* merupakan fungsi untuk pengisian

data pada database dengan mengambil waktu pada komputer ketika record dibuat. *Action* merupakan *field* dengan tipe data varchar yang digunakan untuk memeriksa fungsi apa yang dipanggil dari API KIRI. Terdapat beberapa tipe pada *field* ini, yaitu /

- *ADDAPIKEY*, *action* yang dicatat ke dalam log ketika fungsi pembuatan *API key* yang baru dipanggil.
- *FINDROUTE*, *action* yang dicatat ketika user melakukan pencarian rute
- *LOGIN*, *action* yang dicatat ketika developers melakukan login dengan menggunakan *API key*
- *NEARBYTRANSPORT*, *action* yang dicatat ketika user mencari transportasi di daerah rute sedang dicari
- *PAGELOAD*, *action* yang dicatat ketika user memasuki halaman KIRI
- *REGISTER*,*action* yang dicatat ketika developers melakukan pendaftaran pada KIRI *API key*
- *SEARCHPLACE*, *action* yang dicatat ketika user memanggil fungsi pencarian lokasi dengan menggunakan nama tempat
- *WIDGETERROR*, mencatat log tersebut ketika user menerima error dari *widget*
- *WIDGETLOAD*, mencatat log tersebut ketika user mengdownload widget

AdditionalData, merupakan *field* dengan tipe data varchar yang digunakan untuk mencatat informasi yang dibutuhkan sesuai dengan *field action*.

DAFTAR REFERENSI