

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 ISI	iii
DAFTAR GAMBAR	iv
DAFTAR TABEL	v
1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Perumusan Masalah	2
1.3 Tujuan	2
1.4 Batasan Masalah	2
1.5 Metodologi Penelitian	2
1.6 Sistematika Pembahasan	2
2 LANDASAN TEORI	5
2.1 <i>Data Mining</i>	5
2.1.1 <i>Data Cleaning</i>	5
2.1.2 <i>Data Integration</i>	7
2.1.3 <i>Data Selection</i>	8
2.1.4 <i>Data Transformation</i>	8
2.1.5 <i>Data Mining</i>	10
2.1.6 <i>Pattern Evaluation</i>	18
2.1.7 <i>Knowledge Presentation</i>	18
2.2 Log Histori KIRI	18
2.2.1 Perhitungan Nilai Jarak Menggunakan <i>Euclidean</i>	20
3 ANALISA	21
3.1 Analisis Data	21
3.1.1 Data Cleaning	21
3.1.2 Data Integration	21
3.1.3 <i>Data Selection</i>	21
3.1.4 <i>Data Transformation</i>	22
3.2 Analisis Peringkat Lunak	25
3.2.1 Diagram Use Case Peringkat Lunak <i>Data Mining Log Histori KIRI</i>	27
3.2.2 Diagram Kelas Peringkat Lunak <i>Data Mining Log Histori KIRI</i>	27
DAFTAR REFERENSI	29
A 100 DATA PERTAMA DARI log HISTORI KIRI	31

DAFTAR GAMBAR

2.1	Tahap <i>Data Mining</i>	6
2.2	Tahap <i>data classification</i>	11
2.3	Contoh <i>decision tree</i>	12
2.4	Jenis-jenis <i>split point</i>	14
2.5	Hasil pohon faktor pada atribut <i>age</i> dari tabel 2.1	16
2.6	<i>Decision Tree Pruned</i>	18
3.1	<i>Classification</i> pada daerah Bandung	24
3.2	Diagram <i>Use Case</i> Perangkat Lunak <i>Data Mining Log Histori KIRI</i>	28

DAFTAR TABEL

2.1	Tab l m ngandung <i>missing value</i> dan <i>noisy</i>	7
2.2	Contoh training s t	15
3.1	Contoh data <i>log KIRI</i> s t lah <i>data selection</i>	22
3.2	Contoh hasil data transformasi	24
3.4	Contoh hasil data transformasi <i>latitud</i> <i>longitud</i>	25
3.6	Sk nario M lakukan <i>load Data</i>	28
3.7	Sk nario M lakukan <i>Data Mining</i>	28
3.8	Sk nario M milih Algoritma yang Akan Digunakan	28

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Pertumbuhan teknologi hingga saat ini telah menghasilkan banyak sekali data-data, namun seringkali kali pemilik data hanya menggunakan data tersebut sebagai pelengkap 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 konsumsi dalam membeli jasa atau barang.

Tujuan utama dari *data mining* adalah *knowledge* [1]. *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*, mengidentifikasi 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 di atas, 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* histori?

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 penelitian *data mining* pada *data log* KIRI
- *Data log* yang digunakan untuk mining merupakan log satu bulan dari KIRI

1.5 Metode Penelitian

Berikut adalah Metode Penelitian yang digunakan :

- Melakukan studi literatur tentang algoritma-algoritma yang berkaitan dengan permasalahan *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 kesimpulan

1.6 Sistematika Pembahasan

Sistematika pembahasan dalam penelitian ini adalah:

- BAB 1: Pendahuluan, berisi latar belakang dari penelitian ini, rumusan masalah yang timbul, tujuan yang ingin dicapai, ruang lingkup atau batasan masalah dari penelitian ini, serta metode penelitian yang akan digunakan dan sistematika pembahasan dari penelitian ini

-
- BAB 2: Landasan Teori, teori dasar teori mengenai *data mining*, *data cleaning*, *data integration*, *data selection*, *data transform*, *decision tree*, *pattern evaluation*, *knowledge presentation* dan *log* histori KIRI
 - BAB 3: Teori analisa dasar teori yang akan digunakan, analisa data serta tahap *preprocessing* data yang akan digunakan, serta analisa merancang aplikasi *data mining log* histori KIRI beserta diagram *use case*, skenario, dan diagram kelas
 - BAB 4: Teori perancangan dari aplikasi *data mining log* histori KIRI yang akan dibangun
 - BAB 5: Teori hasil yang diperoleh dan kesimpulan dari penelitian *data mining log* histori KIRI

BAB 2

LANDASAN TEORI

2.1 *Data Mining*

Data mining merupakan proses yang dilakukan pengambilan inti sari atau penggalan *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):

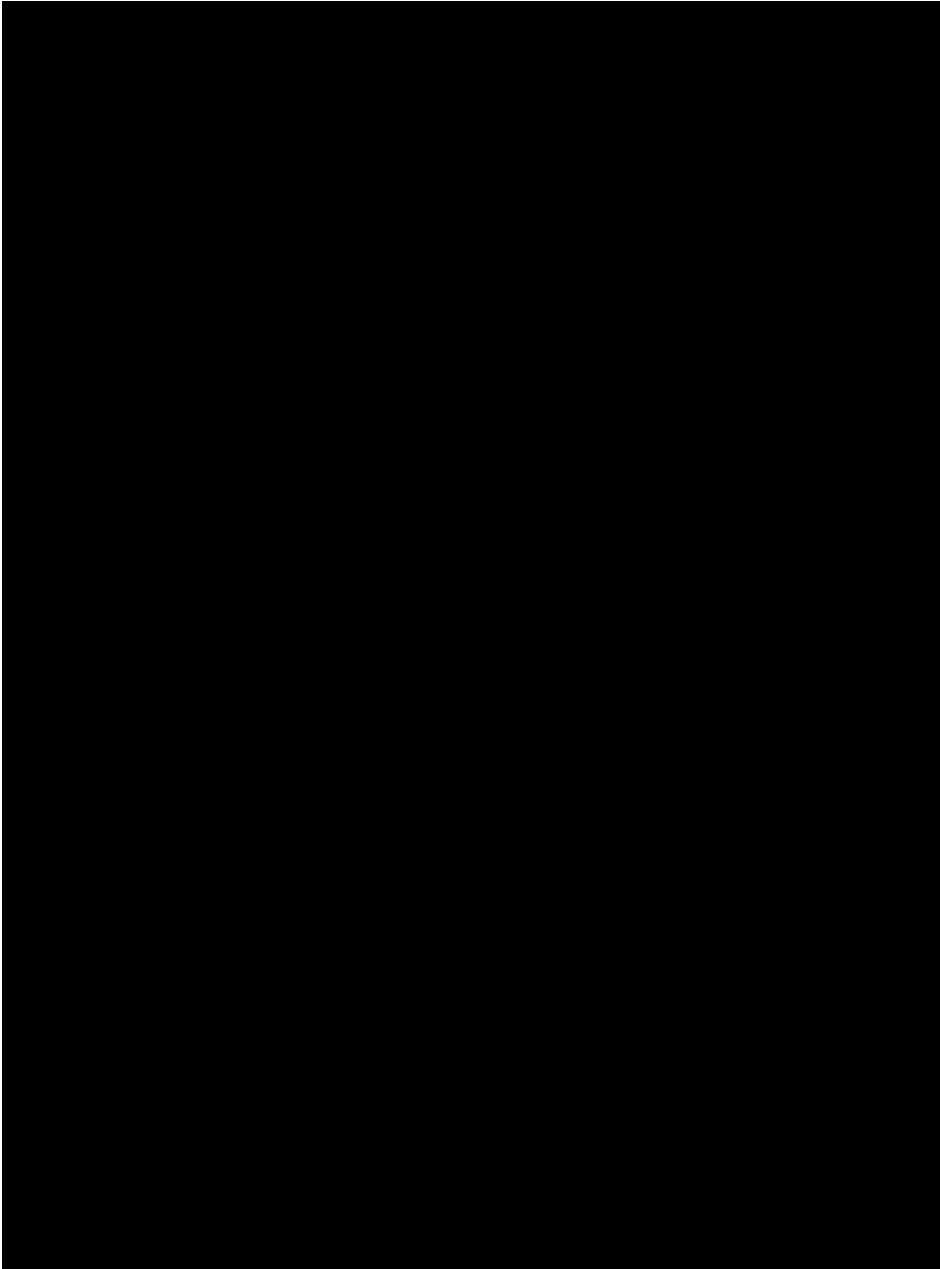
1. *Data cleaning*
2. *Data integration*
3. *Data selection*
4. *Data transformation*
5. *Data mining*
6. *Pattern Evaluation*
7. *Knowledge presentation*

Tahap pertama hingga keempat merupakan bagian dari *data preprocessing*, dimana data-data disiapkan untuk dilakukan penggalan data. Tahap *data mining* merupakan tahap dimana dilakukan penggalan data. Tahap kelima merupakan tahap pencarian pola yang representsasikan *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 redundansi bisa diatasi dengan menghapus atribut tersebut. Contoh studi data yang memiliki *missing value* dan *noisy data* dapat dilihat pada tabel 2.1

Dapat dilihat, pada idP njualan 2, harga dari keyboard adalah -35000, itu merupakan *noisy* karena tidak mungkin nilai harga suatu barang dibawah 0. Pada idP njualan 3, kolom *costumer* tidak memiliki nilai, dan itu merupakan *missing value*.



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

Tab l 2.1: Tab l m ngandung *missing value* dan *noisy*

serta diperbaiki nilai-nilai seperti *primary key*, atribut, dan lain-lain agar tidak terjadi *error* pada *database* yang sudah digabung. Proses dari penggabungan hingga memperbaiki 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 analisis 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 terlabuh hubungan dengan analisis.

2.1.4 Data Transformation

Data transformation merupakan tahap perubahan 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 sebenarnya
- *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

Smoothing

Smoothing merupakan bagian dari *data cleaning* untuk menghilangkan *noise* pada database. Teknik dari *smoothing* adalah *binning*, *regression*, dan *clustering*. Berikut adalah teknik *smoothing* dapat dilihat pada 2.1.1, bagian *noisy data*.

Aggregation

Aggregation, dimana suatu kesimpulan atau hasil dari *aggregation operation* yang disimpan dalam database. Contoh studi kasus, jika terdapat suatu database dari toko A, kita dapat menggunakan operasi *aggregation* untuk mencari total pendapatan dengan rentang hari tertentu.

Generalization

generalization, dimana suatu data yang memiliki nilai *primitive* atau *low level* diubah menjadi *high level* dengan menggunakan konsep hirarki. Contoh studi kasus, nilai pada atribut umur dapat dikelompokkan menjadi muda, dewasa, tua.

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 normalisasi yaitu, *min-max normalization* dan *z-score normalization*. *Min-max normalization* akan mengubah semua nilai menjadi nilai dengan skala tertentu. Dengan menggunakan rumus

$$\nu' = \frac{\nu - \min_A}{\max_A - \min_A}(\text{newMax}_A - \text{newMin}_A) + \text{newMin}_A$$

Contoh kasus, misalkan nilai minimum dan maximum dari suatu pendapatan adalah 12.000 dan 98.000, akan diubah menjadi berskala antara 0,0 sampai 1,0. Jika ada nilai pendapatan 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

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

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 *ar* berdasarkan atribut *panjang* dan *lbar*.

2.1.5 Data Mining

Pada tahap ini, akan dilakukan proses *data mining* dengan menggunakan input data yang sudah diproses pada tahap sebelumnya (*data cleaning*, *data selection*, *data integration*, dan */data transformation*).

Classification and Prediction

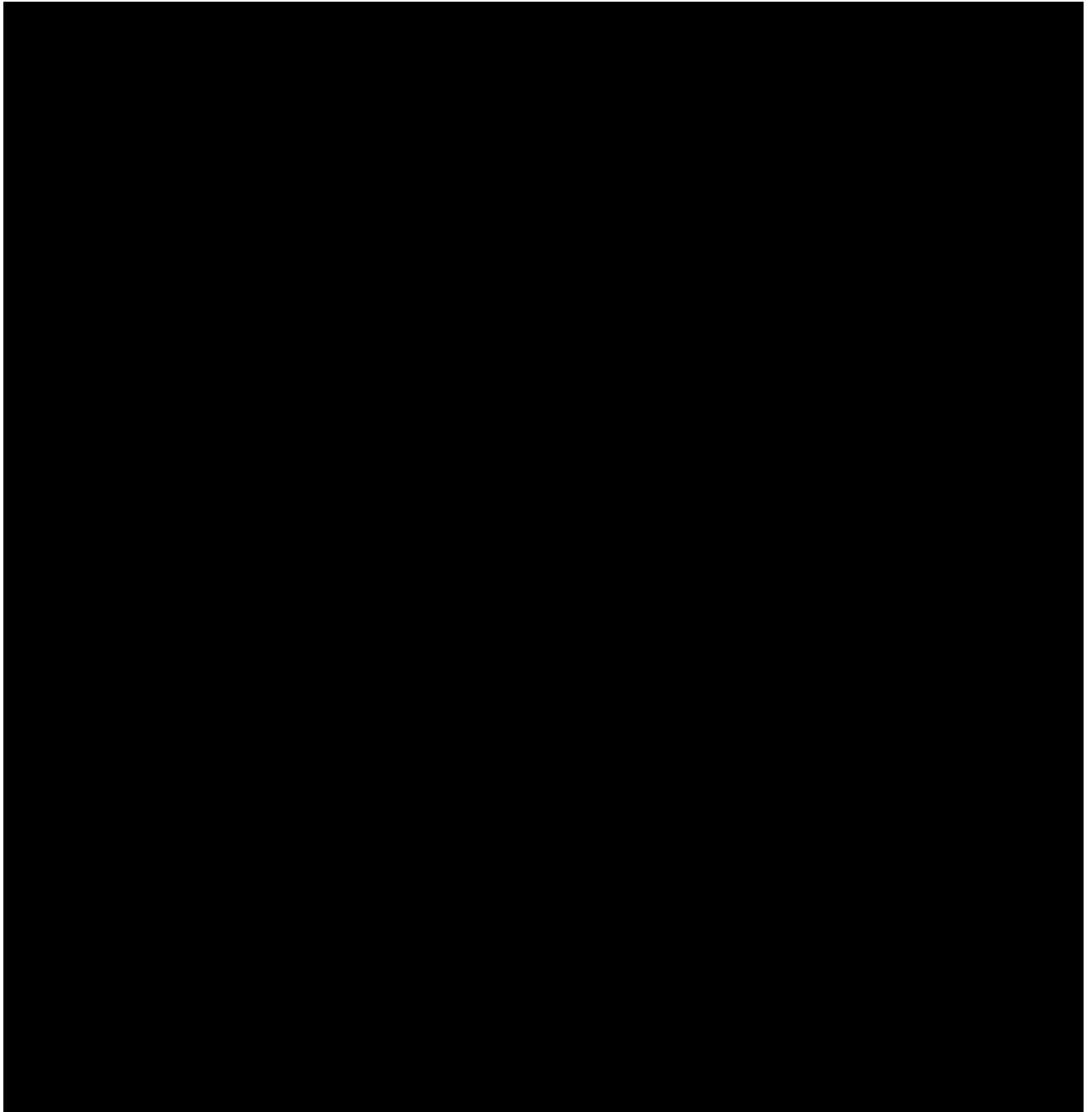
Classification merupakan model yang dibangun untuk memprediksi label kategori, seperti "baik", "cukup", dan "buruk" dalam sistem penilaian sikap seorang siswa atau "mini bus", "bus", atau "s dan" dalam kategori tip mobil. Kategori tersebut dapat diproses menjadi nilai yang menggunakan nilai diskrit. Nilai diskrit merupakan nilai yang terpisah dan berda, seperti 1 atau 5. Kategori yang diproses menjadi nilai diskrit maka akan menjadi nilai yang terurut dan tidak memiliki arti, seperti 1,2,3 untuk memprediksi kategori tip mobil "mini bus", "bus", dan "s dan".

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

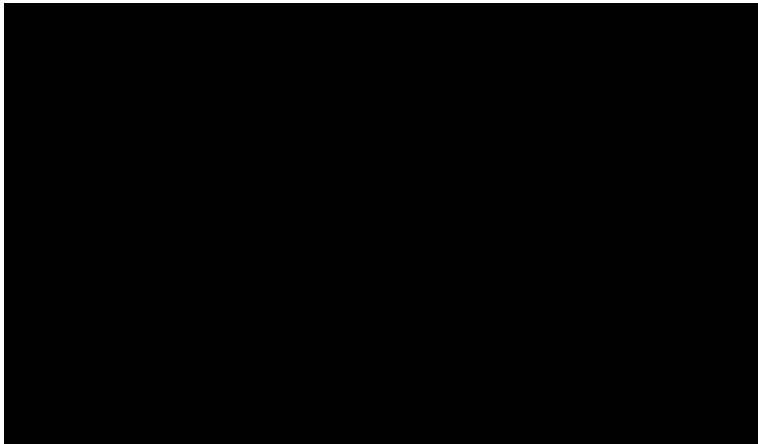
Data Classification merupakan proses untuk melakukan klasifikasi. *Data classification* memiliki dua tahap proses, yaitu *learning step* dan tahap klasifikasi seperti pada ilustrasi di gambar 2.2. *Learning step* merupakan langkah pembelajaran, di mana algoritma klasifikasi membangun *classification 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.

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, di mana setiap node internal (*nonleaf node*) merupakan hasil test dari atribut, setiap cabang merpresentsasikan 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)



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

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

Decision Tree Induction *Decision tree induction* merupakan pelatihan pohon keputusan dari tupel pelatihan klasifikasi. Terdapat beberapa teknik untuk membuat *decision tree* dua diantaranya adalah ID3 dan C4.5. ID3 merupakan teknik pembuatan *decision tree* dengan memanfaatkan *entropy* dan *gain info* untuk menentukan atribut yang terbaik untuk node pada *decision tree*. Sedangkan C4.5 merupakan teknik lanjutan dari ID3 yang menggunakan *gain ratio* untuk melakukan pengujian pada nilai *gain info*. Kedua teknik tersebut menggunakan pendekatan *greedy* yang merupakan *decision tree* yang dibangun secara *top-down recursive divide and conquer*. Algoritma yang dipelajari secara umum sama, hanya berbeda pada *attribute_selection_method*. Berikut algoritma untuk membuat pohon keputusan dari suatu tupel pelatihan.

Input:

- Partisi data, D , merupakan subset data pelatihan dan klasifikasi
- *attribute_list*, merupakan subset 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 a leaf node labeled with the majority class in D ; //majority voting
- (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 //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 subsets for each partition

- (11) l_t is the set of data tuples in D satisfying outcome j ; // a partition
- (12) if D_j is empty then
- (13) attach a leaf labeled with the majority class in D to node N ;
- (14) else attach the node returned by $gain_ratio_decision_tree(D_j, attribute_list)$ to node N ;
- endfor
- (15) return N ;

Pohon keputusan akan dimulai dengan satu node, yaitu N , kemudian prosesitasikan 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 memisahkan nilai atribut pada tuple dalam kelas masing-masing. (langkah 6)

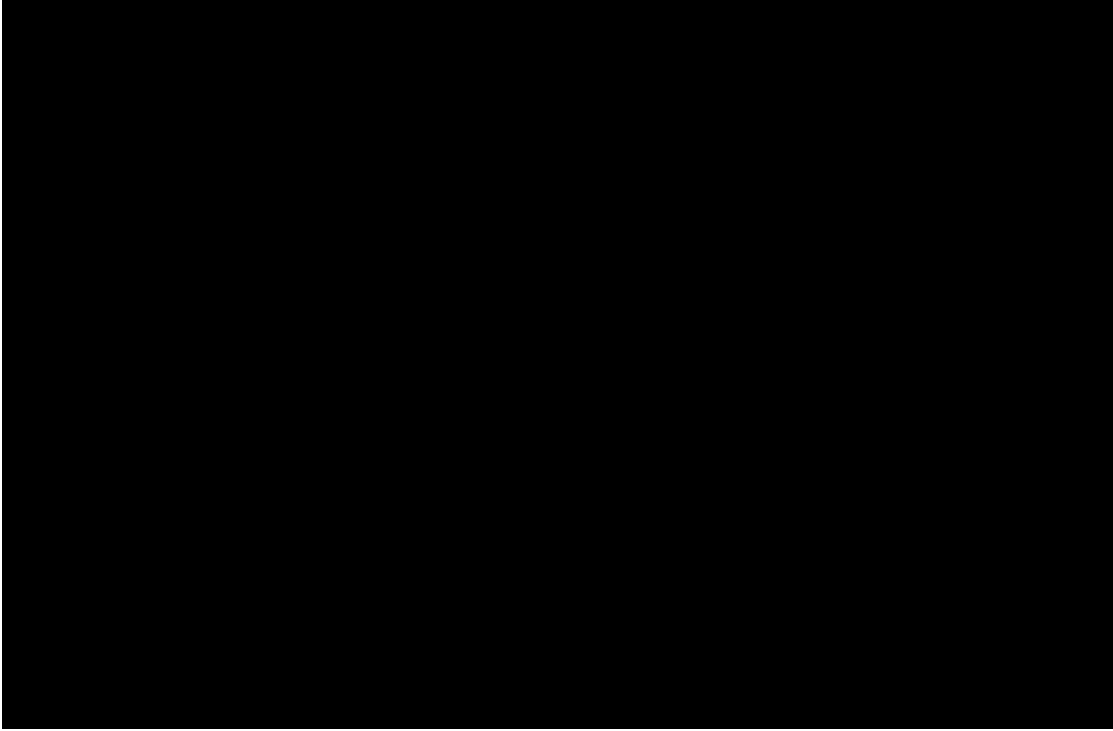
Node N akan diisi dengan hasil dari *splitting criterion* (langkah 7). Kemudian kriteria tersebut agak dibutuhkan cabangnya masing-masing sesuai pada langkah 10 dan 11. Terdapat tiga kemungkinan untuk kriteria jika A merupakan *splitting_attribute* yang memiliki nilai unik seperti $\{a_1, a_2, \dots, a_v\}$ seperti pada gambar 2.4, yaitu,

1. *Discrete valued*: cabang yang dihasilkan memiliki kelas dengan nilai diskrit. Karena kelas yang dihasilkan diskrit dan hanya memiliki nilai yang sama pada cabang tersebut, maka *attribute_list* akan dihapus (langkah 8 dan 9)
2. *Continuous values*: cabang yang dihasilkan memiliki jarak nilai untuk memenuhi suatu kondisi (contoh: $A \leq split_point$), dimana nilai *split_point* adalah nilai pembagi yang dikembalikan oleh *attribute_selection_method*
3. *Discrete valued and a binary tree*: cabang yang dihasilkan adalah dua berupa nilai iya atau tidak dari "apakah A anggota S_a ", dimana S_a merupakan subset dari A , yang dikembalikan oleh *Attribute_selection_method*

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

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

1. Semua tuple pada partisi D merupakan bagian dari kelas yang sama.
2. Sudah tidak ada atribut yang dapat dilakukan pembagian lagi (dilakukan pengujian 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 dibagi cabang, D_j sudah kosong (langkah 12) dan *leaf* akan dibuat dengan *majority class* pada D (langkah 13).

Gambar 2.4: Jenis-jenis *split point*, [1]

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 ke kelas-kelas dalam kelas masing-masing. *Attribute Selection Measure* menyediaan peringkat untuk setiap atribut pada training tuple. Jika *splitting criterion* merupakan nilai *continuous* 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, setiap pelatihan dari *class-labeled* tuple. Jika kelas atribut memiliki m nilai yang berbeda yang mendefinisikan kelas yang berbeda, C_i (for $i=1, \dots, m$). $C_{i,d}$ menjadi kelas tuple dari C_i di D . $|D|$ dan $|C_{i,d}|$ merupakan banyak tuple pada D dan $C_{i,d}$.

Information Gain *Information* menurut Claude Shannon dalam *information theory* adalah ukuran *pure* dari suatu data. Suatu data yang *pure* jika data tersebut memiliki tuple dengan *class* yang sama. 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

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

Dimana p_i merupakan probabilitas tuple pada D terhadap class C_i , dapat diperoleh dengan

$|c_i|/|D|$. $\text{Info}(D)$ 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 *entropy* dari suatu atribut kemudian disisihkan dengan nilai *entropy* dari D . Jika pada tuple D , memiliki atribut A dengan v nilai yang berbeda, maka menghitung *entropy* dari suatu atribut adalah

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

$|D_j|/D$ merupakan angka yang menghitung bobot dari suatu partisi. Semakin kecil nilai dari $\text{Info}_A(D_j)$, maka atribut tersebut masih memberikan informasi, semakin besar nilai $\text{Info}_A(D_j)$, semakin tinggi pula tingkat *pure* dari suatu partisi.

Sehingga mendapatkan nilai $\text{Info}(D)$ dan $\text{Info}_A(D_j)$, *information gain* dapat diperoleh dari selisih nilai $\text{Info}(D)$ dan $\text{Info}_A(D_j)$

$$\text{Gain}(A) = \text{Info}(D) - \text{Info}_A(D)$$

contoh kasus untuk ID3, dalam pencarian *information gain*

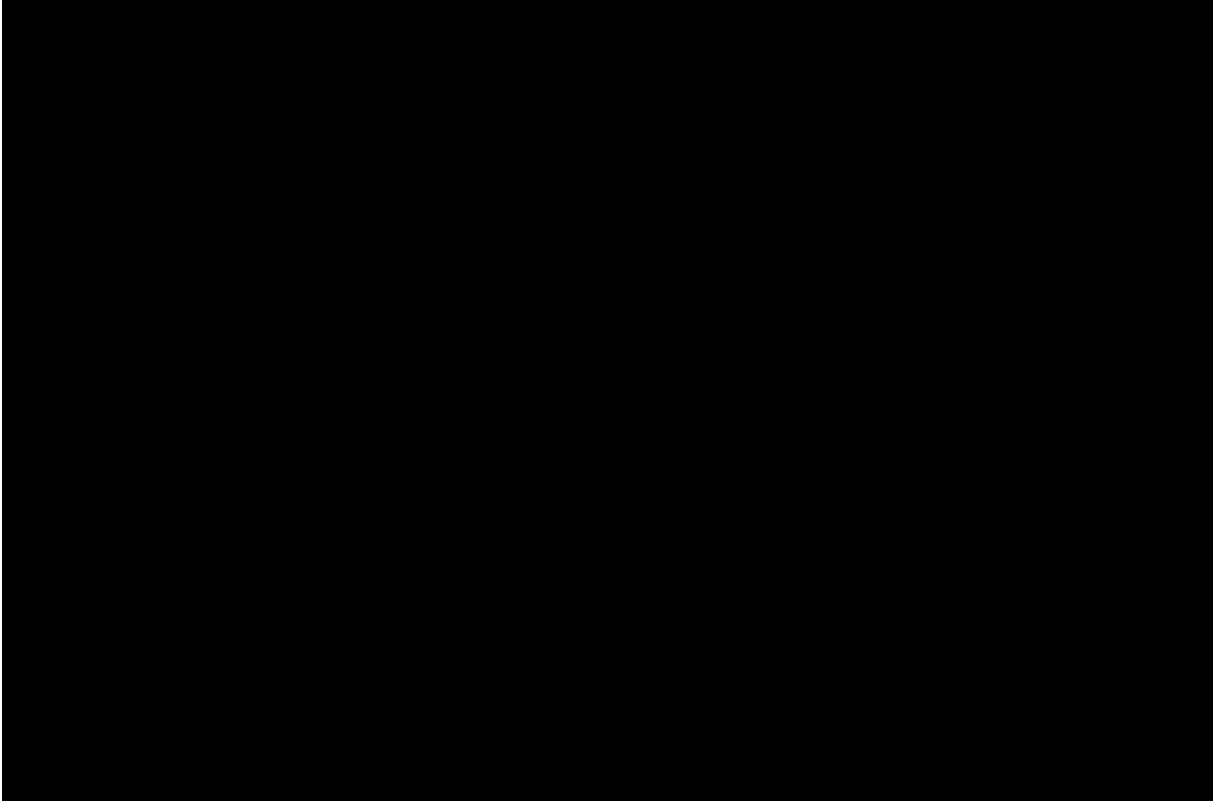
Tab 1 2.2: Contoh training set

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 tab 1 2.2, 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

$$\text{Info}(D) = -\frac{9}{14} \log_2\left(\frac{9}{14}\right) - \frac{5}{14} \log_2\left(\frac{5}{14}\right) = 0.940 \text{ bits}$$

Sehingga 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



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

k las *no*. Pada kat gori *senior*, t r dapat tiga d ngan k las *yes* dan dua d ngan k las *no*. P rhitungan nilai *entropy* atribut *age* t rhadap *D* s bagai b rikut

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

S t lah m ndapatkan *entropy* dari atribut *age*, maka nilai *gain information* dari atribut *age* adalah

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

D ngan m lakukan hal yang sama, dapat dip rol h nilai *gain* untuk atribut *income* adalah 0.029 *bits*, untuk nilai *gain(student)* adalah 0.151 *bits*, dan *gain(credit_rating)* = 0.048 *bits*. Kar na nilai *gain* dari atribut *age* m rupakan nilai t rb sar diantara s mua atribut, maka atribut *age* dipilih m njadi *splitting attribute*. S t lah dit ntukan, nod N akan m mb ntuk cabang b rdasarkan nilai dari atribut *age* s p rti pada gambar 2.5.

Untuk atribut yang m rupakan nilai *continuous*, harus dicari nilai *split point* untuk A. Nilai-nilai dari dua angka yang b rs b lahan dapat diambil nilai t ngahnya untuk dijadikan *split-point*. Jika t r dapat v nilai yang b rb da dari A, maka akan t r dapat v-1 k mungkin *split point*. K mudian nilai *split point* akan dijadikan s bagai nilai p mbagi, s bagai contoh: $A \leq split-point$ m rupakan cabang p rtama, dan $A > split-point$ m rupakan cabang k dua.

Gain Ratio *Information gain* akan memiliki nilai yang baik jika suatu atribut memiliki banyak nilai yang berbeda, namun hal itu tidak selalu bagus. Sebagai contoh kasus, jika nilai id suatu tabel yang memiliki nilai unik, maka akan terdapat banyak sekali cabang. Namun setiap cabang hanya akan berisi satu tuple dan bersifat *pure*, maka nilai *entropy* yang dihasilkan adalah 0. Oleh karena itu, informasi yang diperoleh pada atribut ini akan bernilai maksimum namun tidak akan berguna untuk *classification* [1].

C4.5, menggunakan nilai tambahan dari *information gain* yaitu *gain ratio*, yang dapat mengatasi permasalahan *information gain* tentang nilai yang banyak namun tidak baik untuk *classification*. C4.5 melakukan teknik normalisasi terhadap *gain information* dengan menggunakan *split information* yang memiliki rumus sebagai berikut:

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

Setelah mendapatkan nilai *split info* dari suatu atribut, dapat diperoleh nilai *gain ratio* dengan rumus sebagai berikut:

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

Nilai dari *gain ratio* tersebut yang akan dipilih. Perlu diketahui [1] jika nilai hasil mendekati 0, maka ratio menjadi tidak stabil, oleh karena itu, *gain information* yang dipilih harus sebesar, minimal sama besarnya dengan nilai rata-rata dari semua test yang diperiksa.

Contoh studi kasus, akan dilakukan perhitungan *gain ratio* dengan menggunakan training set pada tabel 2.2. Dapat dilihat pada atribut *income* memiliki tiga partisi yaitu *low*, *medium*, dan *high*. Terdapat empat tuple dengan nilai *low*, enam tuple dengan nilai *medium*, dan empat tuple dengan nilai *high*. Untuk menghitung *gain ratio*, perlu dihitung nilai *split information* terlebih dahulu dengan cara:

$$SplitInfo_A(D) = -\frac{4}{14} \times \log_2\left(\frac{4}{14}\right) - \frac{6}{14} \times \log_2\left(\frac{6}{14}\right) - \frac{4}{14} \times \log_2\left(\frac{4}{14}\right) = 0.926bits$$

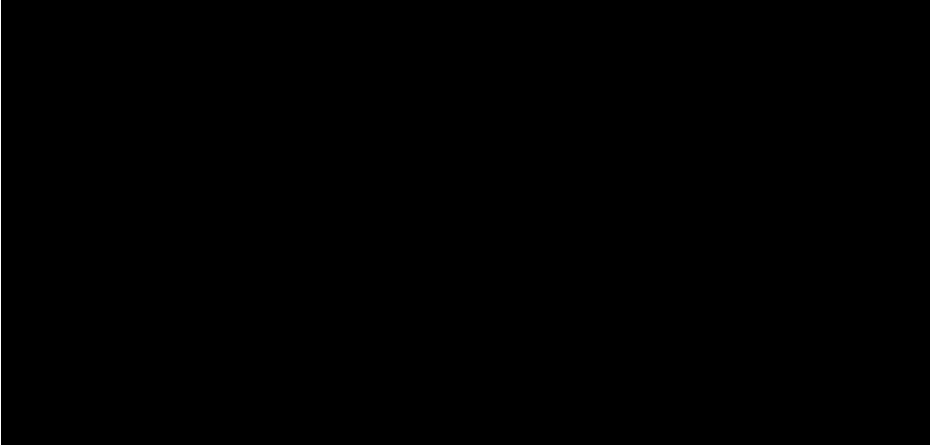
Jika nilai *gain information* dari *income* adalah 0.029, maka, dapat diperoleh *gain ratio* dari *income* adalah

$$GainRatio(D) = \frac{0.029}{0.926} = 0.031bits$$

Tree Pruning *Tree pruning* merupakan proses pemotongan *decision tree* agar lebih efisien dan tidak terlalu mempengaruhi nilai keputusan yang dihasilkan. *decision tree* yang sudah dipotong akan lebih kecil ukurannya, tidak rumit dengan pohon yang asli, namun lebih mudah untuk diproses. *Decision tree* yang sudah dipotong memiliki kemampuan klasifikasi yang lebih baik [1]. Perbandingan *decision tree* yang sudah dipotong dan belum dapat dilihat pada gambar 2.6.

Terdapat dua pendekatan dalam melakukan *pruning*, yaitu *prepruning* dan *postpruning*.

Pada *prepruning*, pemotongan pohon dilakukan dengan cara manual dan tidak dilanjutkan



Gambar 2.6: Decision tree

- *Timestamp* (UTC), waktu ketika pengguna KIRI mencari rute angkutan menggunakan waktu UTC / GMT
- *Action*, tipe dari log yang dibuat.
- *AdditionalData*, mencatat data-data yang berhubungan sesuai dengan nilai atribut *action*

LogId merupakan *field* dengan tipe data integer 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 ketika fungsi pembuatan *API key* yang baru dipanggil.
- *FINDROUTE*, *action* yang dicatat ketika user melakukan pencarian rute
- *LOGIN*, *action* yang dicatat ketika developer melakukan login dengan menggunakan *API key*
- *NEARBYTRANSPORT*, *action* yang dicatat ketika user mencari transportasi terdekat
- *PAGELOAD*, *action* yang dicatat ketika user memasuki halaman KIRI
- *REGISTER*, *action* yang dicatat ketika developer 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 mengunduh *widget*

AdditionalData, merupakan *field* dengan tipe data varchar yang digunakan untuk mencatat informasi yang dibutuhkan sesuai dengan *field action*. Isi dari *additionalData* tersebut untuk setiap *action* adalah

- Jika nilai atribut *action* adalah *ADDAPIKEY*, maka isi nilai dari *additionalData* adalah nilai *API key* yang dihasilkan
- Jika nilai atribut *action* adalah *FINDROUTE*, maka isi nilai dari *additionalData* adalah *latitude* dan *longitude* lokasi awal dan tujuan serta banyak jalur yang dihasilkan dari aplikasi KIRI

- Jika nilai atribut *action* adalah *LOGIN*, maka isi nilai dari *additionalData* adalah id dari user yang melakukan login serta status apakah user berhasil login atau tidak
- Jika nilai atribut *action* adalah *NEARBYTRANSPORT*, maka isi nilai dari *additionalData* adalah *latitude* dan *longitude* dari transportasi tersebut
- Jika nilai atribut *action* adalah *PAGeload*, maka isi nilai dari *additionalData* adalah ip dari user
- Jika nilai atribut *action* adalah *REGISTER*, maka isi nilai dari *additionalData* adalah alamat mail yang digunakan untuk mendaftarkan dan nama user
- Jika nilai atribut *action* adalah *SEARCHPLACE*, maka isi nilai dari *additionalData* adalah nama tempat yang dicari
- Jika nilai atribut *action* adalah *WIDGETERROR*, maka isi nilai dari *additionalData* adalah isi pesan dari error yang terjadi
- Jika nilai atribut *action* adalah *WIDGETLOAD*, maka isi nilai dari *additionalData* adalah ip dari user yang melakukan download widget

2.2.1 Perhitungan Nilai Jarak Menggunakan *Euclidean*

Euclidean dapat menghasilkan nilai jarak antar dua objek. Misal kita memiliki dua objek (p dan q). Jika kedua objek tersebut merupakan objek dengan satu dimensi, maka rumus *euclidean* akan menjadi

$$\sqrt{(p - q)^2} = |p - q|$$

Jika objek p dan q merupakan objek dua dimensi, maka rumus *euclidean* akan menjadi

$$d(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2}$$

Jika objek p dan q merupakan objek tiga dimensi, maka rumus *euclidean* akan menjadi

$$d(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + (p_3 - q_3)^2}$$

Jika objek p dan q merupakan objek n dimensi, maka rumus *euclidean* akan menjadi

$$d(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \dots + (p_{n-1} - q_{n-1})^2 + (p_n - q_n)^2}$$

BAB 3

ANALISA

Pada bab ini, akan dilakukan analisa terhadap data yang akan diproses menggunakan *data mining* dan perangkat lunak yang akan dibangun untuk melakukan proses data tersebut.

3.1 Analisis Data

Pada bab ini, akan dilakukan analisa *preprocessing data* yang meliputi *data cleaning*, *data integration*, *data selection* dan *data transformation*. Setelah membaca dan menganalisis data log histori KIRI, maka penelitian ini akan lebih fokus untuk penelitian mengenai lokasi kebangkitan dan tujuan dari user yang menggunakan aplikasi KIRI.

3.1.1 Data Cleaning

Pada tahap ini, data yang akan menjadi input akan diperiksa apakah mengandung *missing value* atau *noisy*. Setelah dilakukan pemeriksaan, tidak ditemukan *missing value* ataupun *noisy*, sehingga tahap ini dapat diwat.

3.1.2 Data Integration

Pada tahap ini, data-data dari berbagai database akan digabung dan diintegrasikan menjadi satu database. Karena data yang digunakan hanya berasal dari satu tabel, maka tahap ini dapat diwat.

3.1.3 Data Selection

Pada tahap ini, akan dilakukan pemilihan data yang akan digunakan. Pada penelitian ini, akan dilakukan proses *data mining* mengenai lokasi kebangkitan dan tujuan dari seseorang user yang menggunakan aplikasi KIRI. Oleh karena itu, pada atribut *action*, nilai yang akan dipilih hanya *FINDROUTE*. Hal ini dikarenakan, hanya *action FINDROUTE* yang menunjukkan lokasi kebangkitan dan tujuan dari user. Selain itu, data tersebut terlihat menarik karena dimungkinkan dapat menghasilkan suatu pola yang membantu melakukan klasifikasi mengenai perpindahan penduduk khususnya untuk daerah Bandung. Karena seluruh *action* bernilai satu yaitu *FINDROUTE*, maka atribut tersebut dapat dihilangkan. Selain itu, atribut *logId* dan *APIK* tidak akan dimasukkan dalam proses karena tidak memiliki hubungan dengan lokasi kebangkitan dan tujuan dari seseorang user.

Dari analisis diatas, maka atribut yang dipilih untuk diproses dalam *data mining* adalah

- *Timestamp* (UTC)
- *AdditionalData*

Berikut contoh data dari atribut tersebut dapat dilihat pada tabel 3.1

Tab 1 3.1: Contoh data *log KIRI* setelah *data selection*

Timestamp (UTC)	AdditionalData
2/1/2014 0:11	-6.8972513,107.6385574/-6.91358,107.62718/1
2/1/2014 0:13	-6.8972513,107.6385574/-6.91358,107.62718/1
2/1/2014 0:16	-6.90598,107.59714/-6.90855,107.61082/1
2/1/2014 0:18	-6.9015366,107.5414474/-6.88574,107.53816/1
2/1/2014 0:25	-6.90608,107.61530/-6.89140,107.61060/2
2/1/2014 0:27	-6.89459,107.58818/-6.89876,107.60886/2
2/1/2014 0:28	-6.89459,107.58818/-6.86031,107.61287/2

Pada atribut *additionalData*, jika nilai atribut *action* adalah *FINDROUTE*, maka nilai *additionalData* memiliki tiga bagian yang dibatasi dengan '/'. Ketiga bagian tersebut adalah

1. Nilai latitud dan longitud dari lokasi keberangkatan yang dipilih oleh user
2. Nilai latitud dan longitud dari lokasi tujuan yang dipilih oleh user
3. Nilai yang menunjukkan banyak jalur yang dihasilkan oleh sistem KIRI

Nilai dari banyak jalur akan dibuang ketika memasuki tahap *data transformation*, karena nilai tersebut hanya menunjukkan banyak jalur tetapi user pasti hanya memilih salah satu dari jalur tersebut, sehingga nilai jalur ini dapat diasumsikan memiliki nilai 1 semula. Karena kolom jalur bernilai satu semula, maka kolom tersebut dapat dibuang.

3.1.4 Data Transformation

Pada tahap ini, akan dilakukan perubahan data. Pada atribut yang dipilih, nilai dari atribut *timestamp* dan *additionaldata* perlu dilakukan transformasi agar program dapat membaca dan memproses data lebih cepat.

Pada atribut *timestamp*, nilai waktu dari atribut tersebut akan diubah menjadi waktu GMT+8. Kemudian, data akan diubah menjadi nama atribut, yaitu:

- Tanggal, atribut ini akan menunjukkan tanggal ketika user KIRI memanggil *action FINDROUTE*, dengan nilai antara 01 sampai 31
- Bulan, atribut ini akan menunjukkan bulan ketika user KIRI memanggil *action FINDROUTE*, dengan nilai antara 01 sampai 12
- Tahun, atribut ini akan menunjukkan tahun ketika user KIRI memanggil *action FINDROUTE*, dengan format empat angka (contoh: 2014)
- Hari, atribut ini akan menunjukkan hari ketika user KIRI memanggil *action FINDROUTE*, dengan range nilai antara satu sampai minggu

- Jam, atribut ini akan menunjukkan jam ketika KIRI memanggil *action FINDROUTE*, dengan range nilai antara 00 sampai 23
- Menit, atribut ini akan menunjukkan menit ketika KIRI memanggil *action FINDROUTE*, dengan range nilai antara 00 sampai 59

Data *timestamp* diubah menjadi lima bagian, agar dapat dilakukan pengelompokan yang dilihat dari tanggal, bulan, tahun, hari, dan jam atau hasil dari *decision tree* dapat menghasilkan node yang menentukan tanggal, bulan, tahun, hari dan jam.

Pada atribut *additionalData*, data akan diubah menjadi empat atribut, yaitu:

- Latitud kebangkatan, atribut ini berisi nilai latitud dari lokasi kebangkatan yang dipilih oleh user
- Longitud kebangkatan, atribut ini berisi nilai longitud dari lokasi kebangkatan yang dipilih oleh user
- Latitud tujuan, atribut ini berisi nilai latitud dari lokasi tujuan yang dipilih oleh user
- Longitud tujuan, atribut ini berisi nilai longitud dari lokasi tujuan yang dipilih oleh user

Data *additionalData* diubah menjadi empat bagian, agar program dapat membaca data tersebut lebih mudah.

Dari analisis diatas, banyak atribut dari tabel *statistics* akan menjadi sepuluh, yaitu:

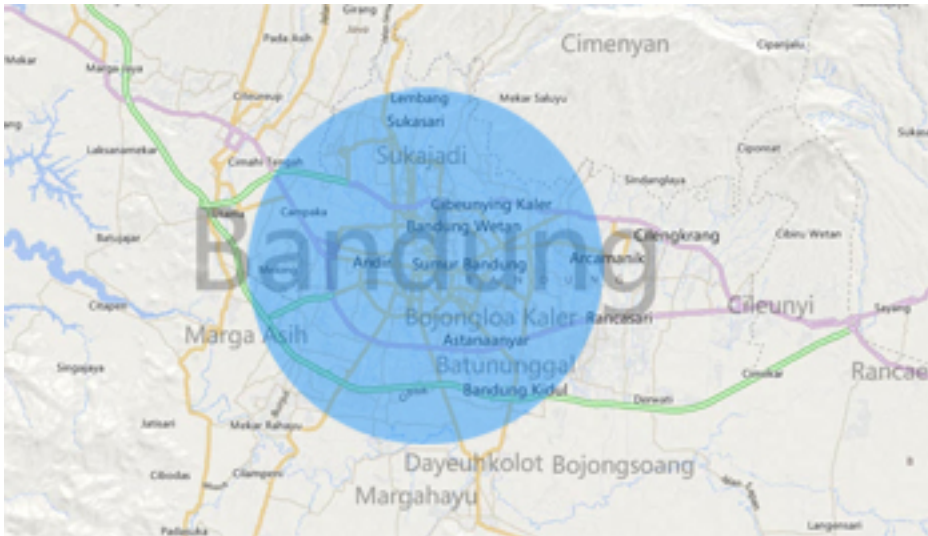
- Tanggal
- Bulan
- Tahun
- Hari
- Jam
- Menit
- Latitud Kebangkatan
- Longitud Kebangkatan
- Latitud Tujuan
- Longitud Tujuan

Contoh hasil data transformasi jika input merupakan data dari tabel 3.1 dapat dilihat pada tabel 3.2.

Setelah nilai tersebut diperolah, nilai *longitude* serta *latitude* dari data lokasi kebangkatan dan tujuan akan diubah sekali lagi menjadi nilai yang menunjukkan apakah daerah lokasi tersebut menunjukkan perjalanan keluar dari Bandung atau tidak. Hal ini dilakukan agar diperolah data perbandingan penduduk, apakah mereka lebih banyak yang keluar dari Bandung atau

Tab 1 3.2: Contoh hasil data transformasi

Tanggal	Bulan	Tahun	Hari	Jam	Menit	Latitude Keberangkatan	Longitude Keberangkatan
01	02	2014	Sabtu	00	11	-6.8972513	107.6185574
01	02	2014	Sabtu	00	13	-6.8972513	107.6385574
01	02	2014	Sabtu	00	16	-6.90598	107.59714
01	02	2014	Sabtu	00	18	-6.9015366	107.5414474
01	02	2014	Sabtu	00	25	-6.90608	107.61530
01	02	2014	Sabtu	00	27	-6.89459	107.58818
01	02	2014	Sabtu	00	28	-6.89459	107.58818

Gambar 3.1: *Classification* pada da rah Bandung

s baliknya b rdasarkan waktu t rt ntu. Untuk m n ntukan hal t rs but, maka akan dibutuhkan klasifikasi da rah agar mudah dilakukan p n ntuan apakah *user* akan b rangkat k Bandung atau tidak. *Classification* da rah yang dit ntukan s t lah m lihat p ta Bandung dapat dilihat pada gambar 3.1.

P n ntuan *classification* t rs but b rdasarkan p rkiraan titik pusat yang sudah dit ntukan, yaitu -6.92036,107.60500 dalam latitud dan longitud . Untuk m ncari nilai rusuk dari lingkaran t rs but, maka akan diambil nilai titik k dua dari sisi lingkaran t rs but. Nilai sisi yang dipilih adalah -6.92036,107.67023 dalam latitud dan longitud . Maka untuk m ndapatkan nilai rusuk dari lingkaran dapat dip rol h d ngan cara m nghitung *euclidean* dari k dua titik t rs but.

$$r = \sqrt{(-6.92036 - (-6.92036))^2 + (107.60500 - 107.67023)^2} r = 0.06523$$

Dari p rhitungan t rs but, maka dapat disimpulkan jika suatu nilai latitud dan longitud yang dihitung p rb daan jaraknya d ngan titik pusat yang sudah dit ntukan dan dip rol h nilainya kurang dari 0.06523, dapat dikatakan bahwa lokasinya b rada di Bandung. Jika jaraknya l bih b sar dari 0.06523, maka lokasinya b rada di luar Bandung.

Nilai jarak dari lokasi k b rangkatan t rhadap titik pusat dan lokasi tujuan t rhadap titik pusat, dapat dijadikan acuan untuk m n ntukan apakah *user* t rs but m nuju da rah Bandung atau k luar

dari Bandung. Kondisi yang menentukan apakah user menuju Bandung yaitu, jika jarak dari lokasi ke lokasi rangkatan dengan titik pusat lebih besar daripada 0.06523 (dari luar Bandung) dan jarak dari lokasi tujuan dengan titik pusat lebih kecil dari 0.06523 (di dalam Bandung), maka dapat ditentukan bahwa user tersebut menuju Bandung.

Maka dari itu, nilai *latitude* dan *longitude* dari lokasi ke lokasi rangkatan dan tujuan akan dibuang dan digantikan oleh atribut menuju Bandung dengan tipe data *integer*. Jika isi dari atribut tersebut bernilai 1, maka user tersebut menuju Bandung sedangkan nilai 0 berarti user tidak menuju Bandung, dan jika nilai atribut tersebut adalah 2, maka user tersebut memiliki lokasi ke lokasi rangkatan dan tujuan di dalam Bandung. Contoh hasil data setelah dilakukan *transformation* terhadap *latitude* dan *longitude* terdapat pada tabel 3.3.

Tab 3.4: Contoh hasil data transformasi *latitude* *longitude*

Tanggal	Bulan	Tahun	Hari	Jam	Menit	MenujuBandung
01	02	2014	Sabtu	00	11	2
01	02	2014	Sabtu	00	13	1
01	02	2014	Sabtu	00	16	1
01	02	2014	Sabtu	00	18	0
01	02	2014	Sabtu	00	25	1
01	02	2014	Sabtu	00	27	2
01	02	2014	Sabtu	00	28	0

3.2 Analisis Perangkat Lunak

Agar analisis pola dari lokasi ke lokasi rangkatan dan tujuan dari data *log* historis lebih mudah, maka akan dibangun sebuah perangkat lunak yang dapat melakukan proses *data mining* dengan menggunakan teknik ID3 dan C4.5, serta dapat melakukan visualisasi hasil dari *data mining* yang diperoleh setelah proses dijalankan yaitu perangkat lunak *data mining log* historis KIRI.

Perangkat lunak yang akan dibangun akan berbasis desktop dan menggunakan bahasa pemrograman java. Pada subbab ini akan dibahas spesifikasi kebutuhan fungsional, pemodelan perangkat lunak, diagram *use case*, skenario, diagram kelas dari Perangkat Lunak yang akan dibangun.

Spesifikasi Kebutuhan Fungsional Perangkat Lunak *Data Mining log* Historis KIRI

Spesifikasi kebutuhan perangkat lunak yang akan dibangun untuk melakukan *data mining log* historis KIRI yang sesuai yang diharapkan adalah

1. Dapat menerima dan membaca input text yang sudah disiapkan
2. Dapat melakukan *preprocessing* data sesuai dengan yang dijelaskan pada bab analisis data
3. Dapat melakukan proses *data mining*, ID3 dan C4.5
4. Dapat melakukan visualisasi hasil dari *data mining* yang diperoleh

Pemodelan Perangkat Lunak *Data Mining Log Histori KIRI*

Perangkat lunak *data mining log* histori KIRI akan mendapatkan input data dengan format .txt. Setelah program mendapatkan input dan user menekan tombol proses, maka data tersebut akan diubah terlebih dahulu sesuai pada bab analisis data (bab 3.1) dengan melakukan proses *data transform* dan menghasilkan data dengan format seperti pada tabel 3.2.

Program akan melakukan tahap *data mining* dengan menggunakan teknik ID3 atau C4.5 sesuai dengan permintaan user. Setelah proses *data mining* selesai dilakukan, program akan melakukan *visualisasi decision tree* dan nilai klasifikasi yang diperoleh.

Pemodelan Data pada Perangkat Lunak *Data Mining Log Histori KIRI*

Karena data yang diperoleh sudah dalam bentuk excel, maka pada penelitian ini, tidak akan menggunakan sistem database. Untuk mempermudah penelitian, data-data pada excel akan dipindahkan ke data dengan format .txt. Isi dari file txt tersebut merupakan nilai dari atribut *timestamp*(UTC) dan *additionalData* yang dipisahkan dengan spasi. Hal ini dapat dilakukan dengan menggunakan fungsi *CONCATENATE* dari excel untuk membuat format sesuai yang diharapkan kemudian melakukan *copy* pada kolom *CONCATENATE* lalu *paste* pada file txt yang masih kosong. Contoh data input untuk perangkat lunak *data mining log* histori KIRI adalah

2/1/2014 0:11 -6.8972513,107.6385574/-6.91358,107.62718/1

2/1/2014 0:13 -6.8972513,107.6385574/-6.91358,107.62718/1

2/1/2014 0:16 -6.90598,107.59714/-6.90855,107.61082/1

Setelah dipindahkan ke dalam format .txt, maka data sudah siap untuk menjadi input perangkat lunak *data mining log* histori KIRI.

Ketika tombol proses ditekan, maka data tersebut akan diproses. Proses yang pertama yang akan dilakukan adalah melakukan *load* data dari file. Setelah data didapat, akan dilakukan proses *transform* untuk setiap baris yang ada. Proses *transform* tersebut memiliki tahap sebagai berikut:

1. Mengambil nilai string pada baris tersebut
2. Memisah nilai string yang didapat dengan spasi sebagai tanda pemisah, maka akan terdapat tiga nilai, yaitu tanggal, jam, dan *additionalData*
3. Pada nilai tanggal, dilakukan pemecahan nilai string dengan garis miring sebagai tanda pemisah, maka akan diperoleh tiga nilai yaitu bulan, tanggal, dan tahun
4. Pada nilai jam, dilakukan pemecahan nilai string dengan titik dua sebagai tanda pemisah, maka akan diperoleh dua nilai yaitu jam dan menit
5. Pada *additionalData*, dilakukan pemecahan nilai string dengan garis miring sebagai tanda pemisah, maka akan diperoleh tiga nilai yaitu lokasi awal, lokasi tujuan, dan banyak jalur
6. Mengubah waktu dari UTC menjadi GMT+8
7. Mencari hari dengan memanfaatkan nilai tanggal, bulan, dan tahun serta kelas *calendar*

8. menggabungkan nilai-nilai tersebut dalam dua array, yaitu array dengan tipe `int` (dengan nilai tanggal, bulan, tahun, jam, dan menit) dan array `double` (dengan nilai `loss` sesuai dengan urutan yang diharapkan)

Setelah proses *transform* berhasil dilaksanakan, maka data sudah siap untuk dijadikan nilai input untuk proses data mining pada perangkat lunak *data mining log* histori KIRI.

Pemodelan Fungsi pada Perangkat Lunak *Data Mining Log* Histori KIRI

Setelah *preprocessing* data selesai dilaksanakan, maka program akan menjalankan proses *data mining*. Proses tersebut memiliki tahap sebagai berikut

1. Program akan menjalankan algoritma untuk membuat *decision tree* yang terdapat pada 2.1.5
2. Program akan menampilkan *decision tree*

Pada tahap pertama, isi metode pada *attribute_selection_method* akan memiliki tahap proses sebagai berikut

1. Program akan menghitung nilai *entropy class*
2. Program akan menghitung nilai *entropy* dan mendapatkan nilai *gain info* untuk setiap atribut pada *attribute_list*
3. Jika *user* memilih untuk menggunakan algoritma C4.5, maka program akan menghitung *splitInfo* dan menghitung *gainRatio*
4. Program akan memilih atribut yang terbaik untuk dijadikan *node* (jika ID3 maka nilai *gainInfo* yang akan digunakan untuk memilih atribut, jika C4.5 maka nilai *gain Ratio* yang akan digunakan untuk memilih atribut)
5. Program akan mengembalikan *node* yang dipilih beserta nilai tuple yang terdapat pada cabang masing-masing

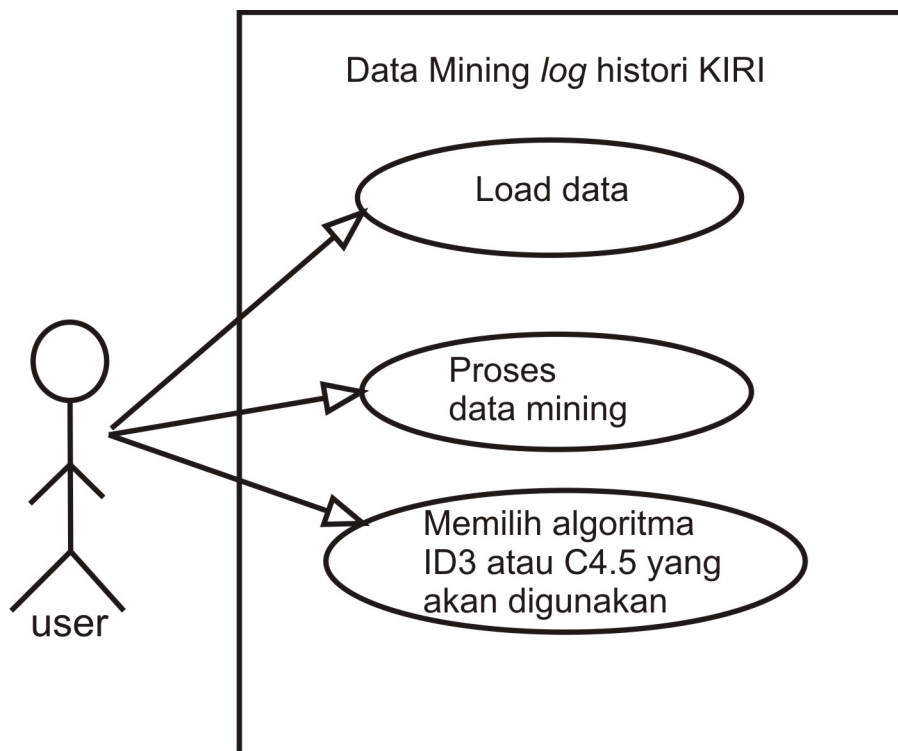
3.2.1 Diagram Use Case Perangkat Lunak *Data Mining Log* Histori KIRI

Diagram *use case* merupakan diagram yang mendeskripsikan sistem dan lingkungannya. Pada penelitian ini, lingkungan yang pada sistem yang dibangun adalah *user*. Berdasarkan analisa yang telah dilakukan, maka *user* dapat melakukan:

- Melakukan *load* data yang digunakan sebagai input data dengan cara memasukkan alamat data pada program
- Memilih algoritma yang akan digunakan, terdapat dua algoritma, yaitu ID3 dan C4.5
- Melakukan proses *data mining* dengan input data dari alamat data yang sudah dimasukan. Setelah proses berhasil dilaksanakan, program akan menampilkan hasil yang diperoleh

Diagram *use case* saat *user* menjalankan perangkat lunak *data mining log* histori KIRI dapat dilihat pada gambar 3.2

3.2.2 Diagram kelas Perangkat Lunak *Data Mining Log* Histori KIRI

Gambar 3.2: Diagram *Use Case* P perangkat Lunak *Data Mining Log Histori KIRI*Tab 1 3.6: Sk nario M lakukan *load* Data

Nama	Load data
Aktor	<i>User</i>
D skripsi	M masukan alamat data yang akan dijadikan s sebagai input program
Kondisi awal	<i>Textbox</i> b lum t risi
Kondisi akhir	<i>Textbox</i> sudah t risi d ngan alamat data
Sk nario utama	<i>User</i> m masukan alamat data pada t xtbody
Eks spi	Data tidak dit mukan

Tab 1 3.7: Sk nario M lakukan *Data Mining*

Nama	Pros s <i>Data Mining</i>
Aktor	<i>User</i>
D skripsi	M n kan tombol pros s pada <i>interface</i>
Kondisi awal	<i>Textbox</i> b lum t risi
Kondisi akhir	<i>Textbox</i> sudah t risi d ngan hasil <i>data mining</i>
Sk nario utama	<i>User</i> m n kan tombol pros s
Eks spi	Data tidak dit mukan atau data tidak dapat dipros s

Tab 1 3.8: Sk nario M milih Algoritma yang Akan Digunakan

Nama	M milih algoritma ID3 atau C4.5
Aktor	<i>User</i>
D skripsi	Us r m milih algoritma yang akan dipakai
Kondisi awal	<i>Radiobutton</i> t rpilih pada ID3
Kondisi akhir	<i>Radiobutton</i> t rpilih pada ID3 atau C4.5
Sk nario utama	<i>User</i> m milih algoritma yang akan digunakan
Eks spi	Tidak ada

DAFTAR REFERENSI

LAMPIRAN A

100 DATA PERTAMA DARI LOG HISTORI KIRI

LogId	APIKey	Timestamp (UTC)	Action	AdditionalData
113909	E5D9904F0A8B4F99	2/1/2014 0:07	PAGELOAD	/5.10.83.30/
113910	E5D9904F0A8B4F99	2/1/2014/ 0:07	PAGELOAD	/5.5.83.49/
113911	E5D9904F0A8B4F99	2/1/2014/ 0:09	PAGELOAD	/5.10.83.30/
113912	E5D9904F0A8B4F99	2/1/2014 0:10	PAGELOAD	/5.10.83.88/
113913	E5D9904F0A8B4F99	2/1/2014 0:10	PAGELOAD	/5.10.83.58/
113914	A44EB361A179A49E	2/1/2014 0:11	SEARCHPLACE	taman+fot/10
113915	A44EB361A179A49E	2/1/2014 0:11	FINDROUTE	-6.8972513,107.6385574/-6.91358,107.62718/1
113916	E5D9904F0A8B4F99	2/1/2014 0:12	PAGELOAD	/5.10.83.24/
113917	81CC9E4AD224357E	2/1/2014 0:13	WIDGETLOAD	/192.95.25.92/
11318	A44EB361A179A49E	2/1/2014 0:13	SEARCHPLACE	taman+f/10
113919	A44EB361A179A49E	2/1/2014 0:13	FINDROUTE	-6.8972513,107.6385574/-6.91358,107.62718/1
113920	D0AB08D956A351E4	2/1/2014 0:15	FINDROUTE	-6.90598,107.59714/-6.90855,107.61082/1
113921	D0AB08D956A351E4	2/1/2014 0:16	SEARCHPLACE	istanta/0
113922	D0AB08D956A351E4	2.1.2014 0:16	SEARCHPLACE	istaba/0
113923	D0AB08D956A351E4	2/1/2014 0:16	FINDROUTE	-6.90598,107.59714/-6.90855,107.61082/1
113924	D0AB08D956A351E4	2/1/2014 0:17	FINDROUTE	-6.90598,107.59714/-6.90855,107.61082/1

113925	A44EB361A179A49E	2/1/2014 0:18	SEARCHPLACE	kantor+po/10
113926	A44EB361A179A49E	2/1/2014 0:18	SEARCHPLACE	kantor+pos/10
113927	A44EB361A179A49E	2/1/2014 0:18	SEARCHPLACE	kantor+pos+ci/10
113928	A44EB361A179A49E	2/1/2014 0:18	SEARCHPLACE	kantor+pos+cimahi/10
113929	A44EB361A179A49E	2/1/2014 0:18	FINDROUTE	-6.7185828,107.0150728/- 6.918881548242062,107.60667476803064/1
113930	A44EB361A179A49E	2/1/2014 0:18	FINDROUTE	-6.9015366,107.5414474/-6.88574,107.53816/1
113931	E5D9904F0A8B4F99	2/1/2014 0:22	PAGELOAD	/5.10.83.49/
113932	E5D9904F0A8B4F99	2/1/2014 0:22	PAGELOAD	/180.253.140.219/
113933	E5D9904F0A8B4F99	2/1/2014 0:24	PAGELOAD	/180.253.140.219/
113934	E5D9904F0A8B4F99	2/1/2014 0:25	PAGELOAD	/180.253.140.219/
113935	E5D9904F0A8B4F99	2/1/2014 0:25	FINDROUTE	-6.90608,107.61530/-6.89140,107.61060/2
113936	E5D9904F0A8B4F99	2/1/2014 0:26	PAGELOAD	/118.137.96.28/
113937	E5D9904F0A8B4F99	2/1/2014 0:26	FINDROUTE	-6.89459,107.58818/-6.89876,107.60886/2
113938	E5D9904F0A8B4F99	2/1/2014 0:27	FINDROUTE	-6.90608,107.61530/-6.89140,107.61060/2
113939	E5D9904F0A8B4F99	2/1/2014 0:28	FINDROUTE	-6.89977,107.62706/-6.89140,107.61060/2
113940	E5D9904F0A8B4F99	2/1/2014 0:28	FINDROUTE	-6.89459,107.58818/-6.86031,107.61287/2
113941	D0AB08D956A351E4	2/1/2014 0:28	FINDROUTE	-6.90598,107.59714/-6.90855,107.61082/1
113942	A44EB361A179A49E	2/1/2014 0:29	FINDROUTE	-6.9172304,107.6042556/-6.92663,107.63644/1
113943	A44EB361A179A49E	2/1/2014 0:29	FINDROUTE	-6.9172448,107.6042255/-6.92663,107.63644/1
113944	D0AB08D956A351E4	2/1/2014 0:30	FINDROUTE	-6.90598,107.59714/-6.90855,107.61082/1
113945	D0AB08D956A351E4	2/1/2014 0:32	FINDROUTE	-6.90598,107.59714/-6.90855,107.61082/1
113946	D0AB08D956A351E4	2/1/2014 0:33	FINDROUTE	-6.90598,107.59714/-6.90855,107.61082/1
113947	A44EB361A179A49E	2/1/2014 0:35	SEARCHPLACE	jalan+asia+af/8
113948	A44EB361A179A49E	2/1/2014 0:35	FINDROUTE	-6.9172448,107.6042255/-6.92163,107.61046/1
113949	A44EB361A179A49E	2/1/2014 0:35	SEARCHPLACE	taman+fotog/10

113950	A44EB361A179A49E	2/1/2014 0:36	FINDROUTE	-6.917321,107.6043132/- 6.921568846707516,107.61015225201845/1
113951	E5D9904F0A8B4F99	2/1/2014 0:38	PAGELOAD	/5.10.83.68/
113952	E5D9904F0A8B4F99	2/1/2014 0:38	PAGELOAD	/5.10.83.28/
113953	E5D9904F0A8B4F99	2/1/2014 0:40		

113976	E5D9904F0A8B4F99	2/1/2014 1:25	PAGELOAD	/5.10.83.24/
113977	E5D9904F0A8B4F99	2/1/2014 1:25	FINDROUTE	-6.91485,107.59123/-6.91593,107.65588/1
113978	E5D9904F0A8B4F99	2/1/2014 1:26	PAGELOAD	/5.10.83.82/
113979	E5D9904F0A8B4F99	2/1/2014 1:28	FINDROUTE	-6.91593,107.65588/-6.91485,107.59123/1
113980	A44EB361A179A49E	2/1/2014 1:29	FINDROUTE	-6.9250709,107.6204635/-6.91728,107.60417/1
113981	A44EB361A179A49E	2/1/2014 1:35	FINDROUTE	-6.9252132,107.6200288/-6.91728,107.60417/1
113982	A44EB361A179A49E	2/1/2014 1:36	FINDROUTE	-6.922427886995373,107.61768691241741/-6.91728,107.60417/1
113983	E5D9904F0A8B4F99	2/1/2014 1:36	FINDROUTE	-6.91431,107.63921/-6.94024,107.71550/1
113984	E5D9904F0A8B4F99	2/1/2014 1:37	PAGELOAD	/5.10.83.98/
113985	A44EB361A179A49E	2/1/2014 1:37	FINDROUTE	-6.921635413232821,107.61909071356058/-6.91728,107.60417/1
113986	E5D9904F0A8B4F99	2/1/2014 1:38	FINDROUTE	-6.88936,107.57533/-6.92600,107.63628/1
113987	E5D9904F0A8B4F99	2/1/2014 1:39	PAGELOAD	http://www.kiri.trav l/m/r/?qs=trans+studi...
113988	E5D9904F0A8B4F99	2/1/2014 1:39	FINDROUTE	-6.92600,107.63628/-6.88936,107.57533/1
113989	A44EB361A179A49E	2/1/2014 1:41	SEARCHPLACE	t rminal+ta/10
113990	A44EB361A179A49E	2/1/2014 1:41	FINDROUTE	-6.9158359,107.6101751/-6.90658,107.61623/1
113991	A44EB361A179A49E	2/1/2014 1:42	FINDROUTE	-6.9158359,107.6101751/-6.90658,107.61623/1
113992	D0AB08D956A351E4	2/1/2014 1:50	FINDROUTE	-6.38355,106.919975/-7.08933734335005,107.562576737255/1
113993	A44EB361A179A49E	2/1/2014 1:51	SEARCHPLACE	taman+ci/10
113994	A44EB361A179A49E	2/1/2014 1:51	SEARCHPLACE	taman+cilaki/10
113995	E5D9904F0A8B4F99	2/1/2014 1:52	PAGELOAD	/206.53.152.33/m
113996	E5D9904F0A8B4F99	2/1/2014 1:52	FINDROUTE	-6.90598,107.59714/-6.91728,107.60417/1
113997	A44EB361A179A49E	2/1/2014 1:54	FINDROUTE	-6.901306,107.6214169/-6.90336,107.62235/1
113998	A44EB361A179A49E	2/1/2014 1:54	FINDROUTE	-6.901306,107.6214169/-6.90336,107.62235/1
113999	E5D9904F0A8B4F99	2/1/2014	PAGELOAD	/5.10.83.27/

114000	308201BB30820124	2/1/2014 1:15	SEARCHPLACE	riau+juction/10
114001	308201BB30820124	2/1/2014 1:56	FINDROUTE	-6.90687,107.61239/-6.89032,107.57961/2
114002	E5D9904F0A8B4F99	2/1/2014 1:57	PAGELOAD	/118.99.112.66/
114003	308201BB30820124	2/1/2014 1:57	FINDROUTE	-6.90687,107.61239/-6.90159,107.60442/1
114004	308201BB30820124	2/1/2014 1:57	FINDROUTE	-6.90687,107.61239/-6.89032,107.57961/2
114005	E5D9904F0A8B4F99	2/1/2014 1:58	FINDROUTE	-6.88211,107.60378/-6.90774,107.60908/1
114006	A44EB361A179A49E	2/1/2014 1:59	FINDROUTE	-6.9212516,107.6196466/-6.91728,107.60417/1
114007	308201BB30820124	2/1/2014 1:59	FINDROUTE	-6.90687,107.61239/-6.91486,107.60824/1
114008	687C44EB2424285D	2/1/2014 1:59	WIDGETLOAD	http://www.c nd kial ad rshipschool.sc...
114009	E5D9904F0A8B4F99	2/1/2014 2:00	FINDROUTE	-6.88166,107.61561/-6.90774,107.60908/1