

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>	6
2.1.2 <i>Data Integration</i>	7
2.1.3 <i>Data Selection</i>	7
2.1.4 <i>Data Transformation</i>	8
2.1.5 <i>Data Mining</i>	9
2.1.6 <i>Pattern Evaluation</i>	18
2.1.7 <i>Knowledge Presentation</i>	18
2.2 Log Histori KIRI	18
2.3 Haversine Formula	20
2.4 Waka	20
3 ANALISA	25
3.1 Analisis Data	25
3.1.1 Data Cleaning	25
3.1.2 Data Integration	25
3.1.3 <i>Data Selection</i>	25
3.1.4 <i>Data Transformation</i>	26
3.2 Analisis Perangkat Lunak	30
3.2.1 Diagram Use Case Perangkat Lunak <i>Data Mining Log Histori KIRI</i>	32
3.2.2 Diagram kelas Perangkat Lunak <i>Data Mining Log Histori KIRI</i>	34
DAFTAR REFERENSI	35
A 100 DATA PERTAMA DARI <i>log</i> HISTORI KIRI	37

DAFTAR GAMBAR

2.1	Tahap <i>Data Mining</i>	5
2.2	Tahap <i>data classification</i>	11
2.3	Contoh <i>decision tree</i>	12
2.4	Jenis-jenis <i>split point</i>	13
2.5	Hasil pohon faktor pada atribut <i>age</i> dari tabel 2.1	16
2.6	<i>Decision Tree Pruned</i>	17
3.1	<i>Classification</i> pada daerah Bandung	29
3.2	Diagram <i>Use Case</i> P-rangkaian Lunak <i>Data Mining Log Histori KIRI</i>	33
3.3	Diagram <i>Class</i> P-rangkaian Lunak <i>Data Mining Log Histori KIRI</i>	34

DAFTAR TABEL

2.1	Tab l m ngandung <i>missing value</i> dan <i>noisy</i>	6
2.2	Contoh training s t	15
3.1	Contoh data <i>log KIRI</i> s t lah <i>data selection</i>	26
3.2	Contoh hasil data transformasi	28
3.3	Contoh hasil data transformasi latitud longitud	30
3.5	Sk nario M lakukan <i>load Data</i>	33
3.6	Sk nario M lakukan <i>Data Mining</i>	33
3.7	Sk nario M milih Algoritma yang Akan Digunakan	34

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 lebih mudah menganalisa masalah, pengambilan kesimpulan, bahkan lebih mudah dikonsumsi dalam berbagai 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 Metodologi 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 metodologi penelitian yang akan digunakan dan sistematika pembahasan dari penelitian ini

BAB 2: Landasan Teori, berisi 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: Berisi 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 rikut diagram *use case*, skenario, dan diagram kelas

BAB 4: Berisi perancangan dari aplikasi *data mining log* histori KIRI yang akan dibangun

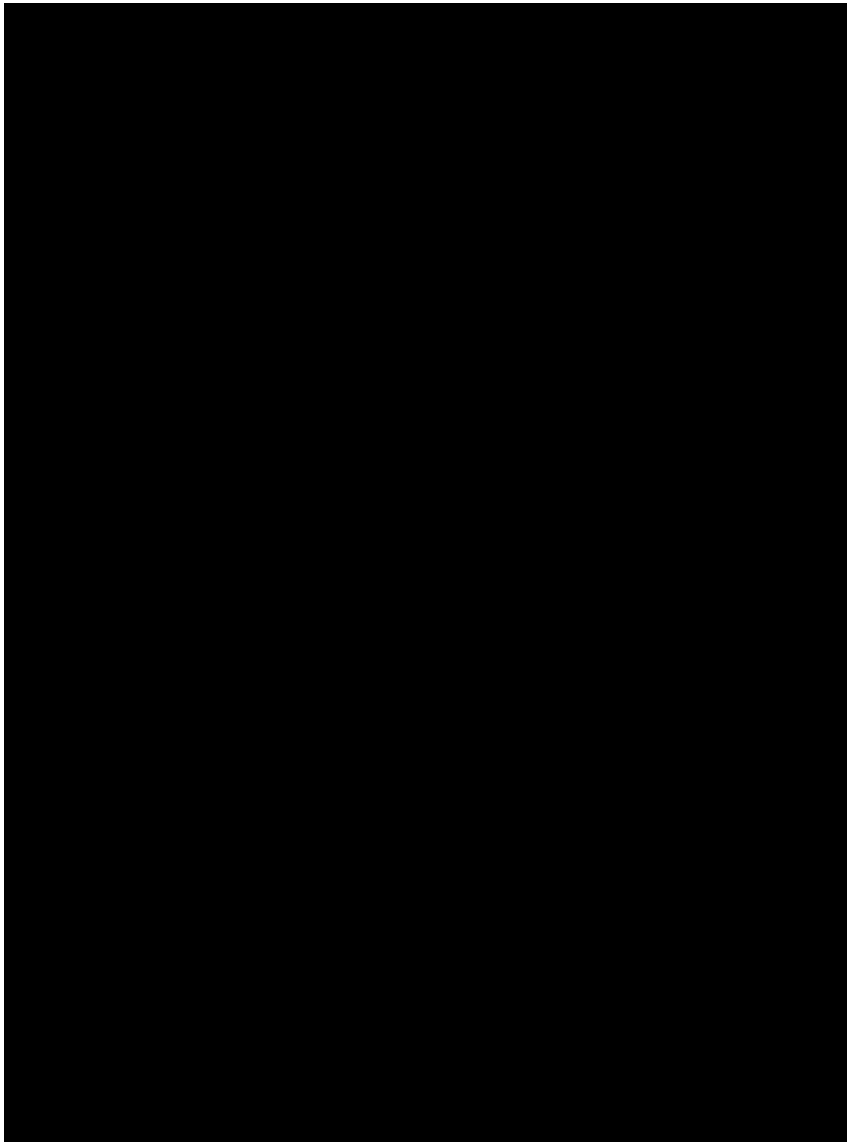
BAB 5: Berisi hasil yang diperoleh dan kesimpulan dari penelitian *data mining log* histori KIRI

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*.



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

Murut [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 pengalihan data. Tahap *data mining* merupakan tahap dimana dilakukan pengalihan data. Tahap kelima merupakan tahap pencarian pola yang merupakan *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

Tabel 2.1: Tabel mengandung *missing value* dan *noisy*

IdPenjualan	NamaBarang	Costumer	Harga	BanyakBarang
1	Mous	Elvin	45000	2
2	Keyboard	Allira	-35000	1
3	Monitor		225000	1

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

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

- 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 nilai persamaan atribut untuk memperkirakan 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 berupa *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 memperпростасikan database model pada sebuah data baik berupa *plain text file* maupun *binary file*.

Tahap ini harus dilakukan secara teliti terutama ketika dalam memasukkan 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 terlewat. *Data integration* yang baik merupakan integrasi yang dapat memaksimalkan kepatatan dan meningkatkan akurasi dari proses *data mining*. Contoh studi kasus dari *data integration*, jika suatu perusahaan 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 analisis mengenai nilai mahasiswa pada tabel nilai yang memiliki atribut sebagai berikut:

NPMMahasiswa

NamaMahasiswa

JenisKulamin

Alamat

MataKuliah

NilaiART

NilaiUTS

NilaiUAS

Maka, atribut yang berpotensi diambil adalah MataKuliah, NilaiART, NilaiUTS, NilaiUAS, sedangkan atribut yang akan dibuang adalah NPMMahasiswa, NamaMahasiswa, JenisKlamin, dan Alamat karena tidak terdapat hubungan dengan analisa.

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 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*

Smoothing

Smoothing merupakan bagian dari *data cleaning* untuk menghilangkan *noise* pada database. Teknik dari *smoothing* adalah *binning*, *regression*, dan *clustering*. Penjelasan 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 *arab* berdasarkan atribut *panjang* dan *lebar*.

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 orang siswa atau "mini bus", "bus", atau "s dan" dalam kategori tip mobil. Kategori tersebut dapat diprediksikan dengan menggunakan nilai diskrit. Nilai diskrit merupakan nilai yang terpisah dan berda, seperti 1 atau 5. Kategori yang diprediksikan oleh nilai diskrit maka akan menjadi nilai yang teratur dan

tidak memiliki arti, seperti 1,2,3 untuk merepresentasikan kategori tip mobil "mini bus", "bus", dan "s dan".

Prediction merupakan model yang dibangun untuk memprediksi nilai kontinu atau *ordered value*. *Ordered value* merupakan nilai yang terurut dan berurutan. Contoh studi kasus untuk model *prediction* adalah seorang marketing ingin memprediksi seberapa banyak konsumen yang akan membeli 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, 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 tuple pelatihan kelas label. 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 tuple pelatihan.

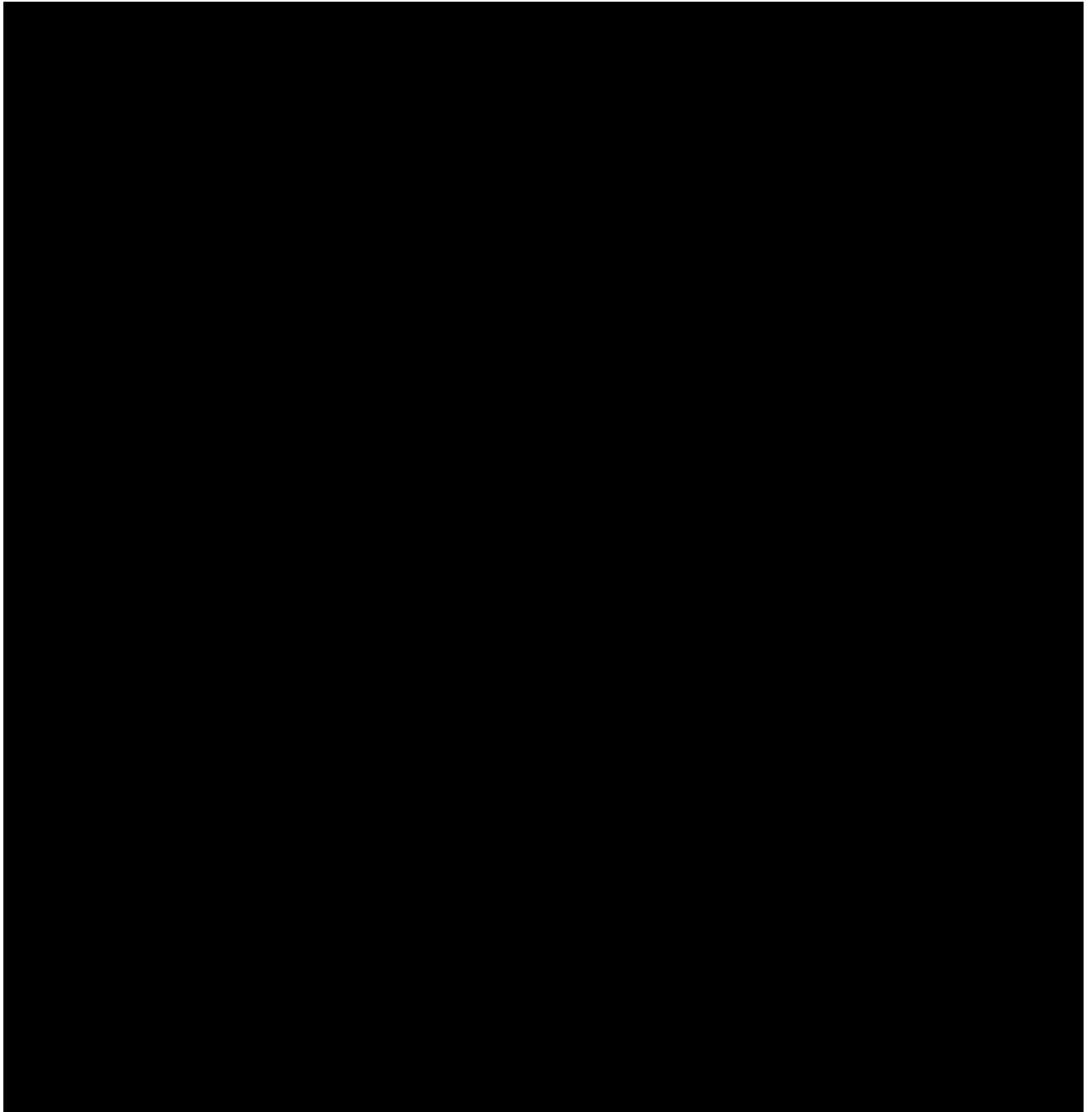
Require: Partisi data, D , merupakan subset data pelatihan dan kelas label

Require: *attribute_list*, merupakan subset dari atribut kandidat

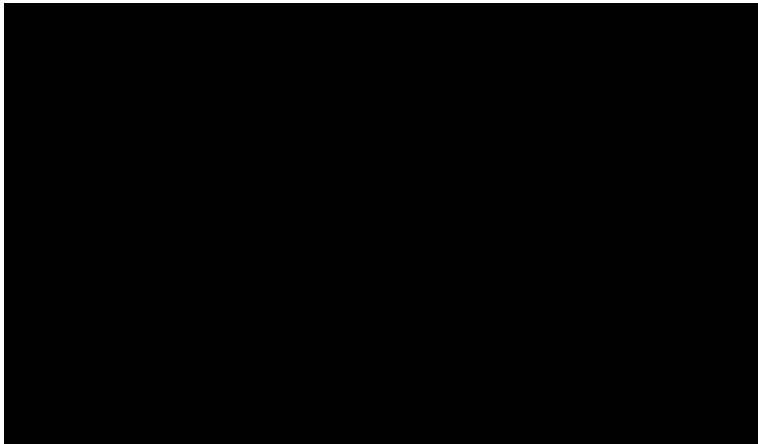
Require: *Attribute_selection_method*, prosedur untuk menentukan *splitting criterion*. Pada input ini, terdapat juga data *splitting_attribute* dan mungkin salah satu dari *split point* atau *splitting subset*

Ensure: Pohon keputusan

- 1: Membuat node N ;
- 2: **if** tuple pada D merupakan kelas yang sama, C **then**



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

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

```

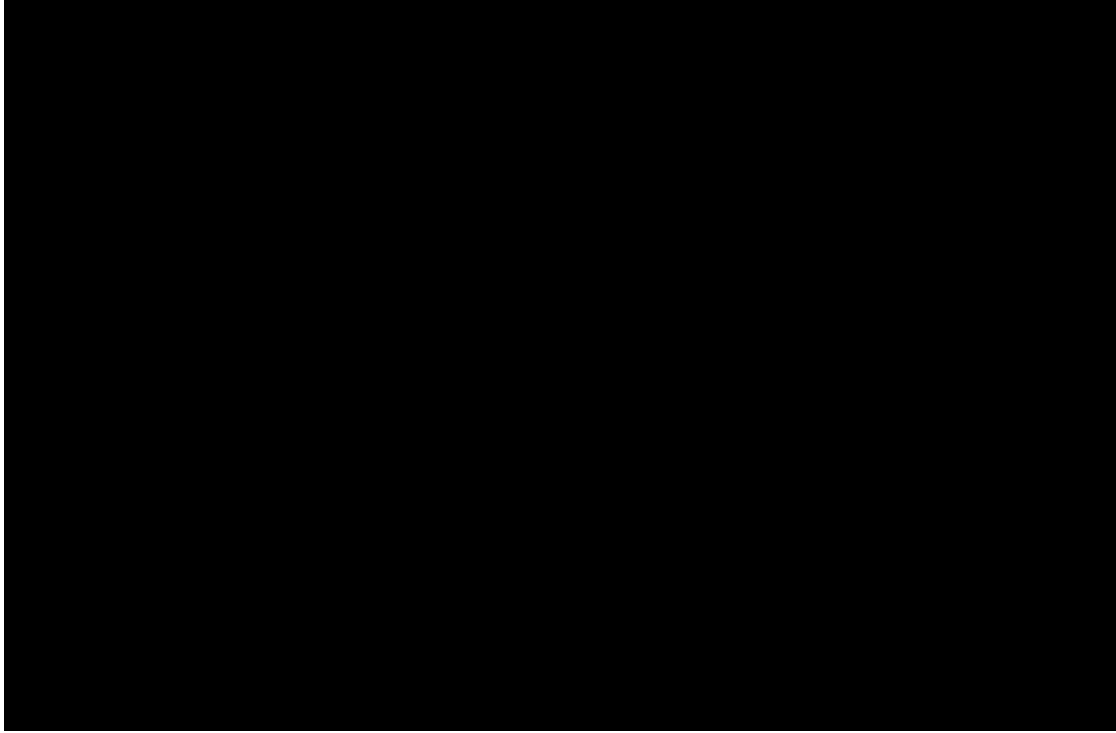
3:  return N s sebagai nod daun d ngan lab l k las C;
4: end if
5: if atribut _list tidak ada nilai atau kosong then
6:  return N s sebagai nod daun d ngan lab l k las yang t rpaling banyak pada D; {majority
    voting}
7: end if
8: m manggil m thod Atribut _s l ction_m thod(D, atribut _list) untuk m ncari nilai t rbaik
    splitting_crit rion;
9: m namakan nod N d ngan splitting_crit rion;
10: if splitting_atribut m merupakan nilai discr t and multiway splits diizinkan then
11:  atribut _list atribut _list - splitting_atribut ; {m nghapus splitting_atribut }
12: end if
13: for all hasil j dari splitting_crit rion do
14:  Dj m merupakan himpunan data tup l D yang s suai d ngan j;
15:  if Dj tidak ada nilai atau kosong then
16:    m lampirkan daun yang dib ri lab l d ngan k las mayoritas di D k nod N;
17:  else
18:    m lampirkan nod yang dik mbalikan ol h g n rat _d cision_tr (Dj, atribut _list) k
        nod N;
19:  end if
20: end for
21: return N;

```

Pohon k putusan akan dimulai d ngan satu nod , yaitu N, m r pr s ntasikan tupl p latihan pada D (baris 1)

Jika tupl di D m miliki k las yang sama s mua, maka nod N akan m njadi daun dan dib ri lab l dari k las t rs but (baris 2 sampai 4). P rlu dik tahui bahwa baris 5 sampai 7 akan m ngakhiri kondisi.

Jika tupl di D ada k las yang b rb da, maka algoritma akan m manggil *attribute_selection_method* untuk m n ntukan *splitting criterion*. *Splitting criterion* akan m n ntukan atribut pada nod N yang

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

merupakan nilai terbaik untuk memilih nilai atribut pada tuple k dalam kelas masing-masing. (baris 8)

Nodus N akan diisi dengan hasil dari *splitting criterion* (baris 9). Kemudian kriteria tersebut agak dibutuhkan cabangnya masing-masing sesuai pada baris 13 dan 14. 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 *attribut_list* akan dihapus (baris 10 sampai 12)
2. *Continuous values*: cabang yang dihasilkan memiliki jarak nilai untuk memenuhi suatu kondisi (contoh: $A \leq \text{split_point}$), 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 (baris 18).

Rumus tersebut akan menghasilkan salah satu dari kondisi tersebut, yaitu

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

2. Sudah tidak ada atribut yang dapat dilakukan pembagian lagi (dilakukan pembagian pada baris 4). Disini, akan dilakukan *majority voting* (baris 6) 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, D_j sudah kosong (baris 15) dan *leaf* akan dibuat dengan *majority class* pada D (baris 16).

Pada baris 21, 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 dalam kelas masing-masing. *Attribute Selection Measure* menydiakan 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 label 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 $|C_{i,d}|/|D|$. $Info(D)$ 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 *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

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

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

Sehingga mendapatkan nilai $Info(D)$ dan $Info_A(D)$, *information gain* dapat diperoleh dari selisih nilai $Info(D)$ dan $Info_A(D)$

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

contoh kasus untuk ID3, dalam pencarian *information gain*

Tab l 2.2: Contoh training s t

RID	umur	p ndapatan	siswa	r siko_kr dit	Class: m mb li_komput r
1	muda	tinggi	tidak	cukup	tidak
2	muda	tinggi	tidak	baik	tidak
3	r maja	tinggi	tidak	cukup	ya
4	d wasa	s dang	tidak	cukup	ya
5	d wasa	r ndah	ya	cukup	ya
6	d wasa	r ndah	ya	baik	tidak
7	r maja	r ndah	ya	baik	ya
8	muda	s dang	tidak	cukup	tidak
9	muda	r ndah	ya	cukup	ya
10	d wasa	s dang	ya	cukup	ya
11	muda	s dang	ya	baik	ya
12	r maja	s dang	tidak	baik	ya
13	r maja	tinggi	ya	cukup	ya
14	d wasa	s dang	tidak	baik	tidak

Pada tab l 2.2, t rdatat *training set*, D. Atribut k las lab l m merupakan dua nilai yang b rb da yaitu ya dan tidak, maka dari itu, nilai $m = 2$. C_1 diisi d ngan k las lab l b rnilai ya, s dangkan C_2 diisi d ngan k las lab l b rnilai tidak. T rdatat s mbilan tupl atribut k las lab l d ngan nilai ya dan lima tupl d ngan nilai tidak. Untuk dapat m n ntukan *splitting criterion*, *information gain* harus dihitung untuk s tiap atribut t rl bih dahulu. P rhitungan *entropy* untuk D adalah

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

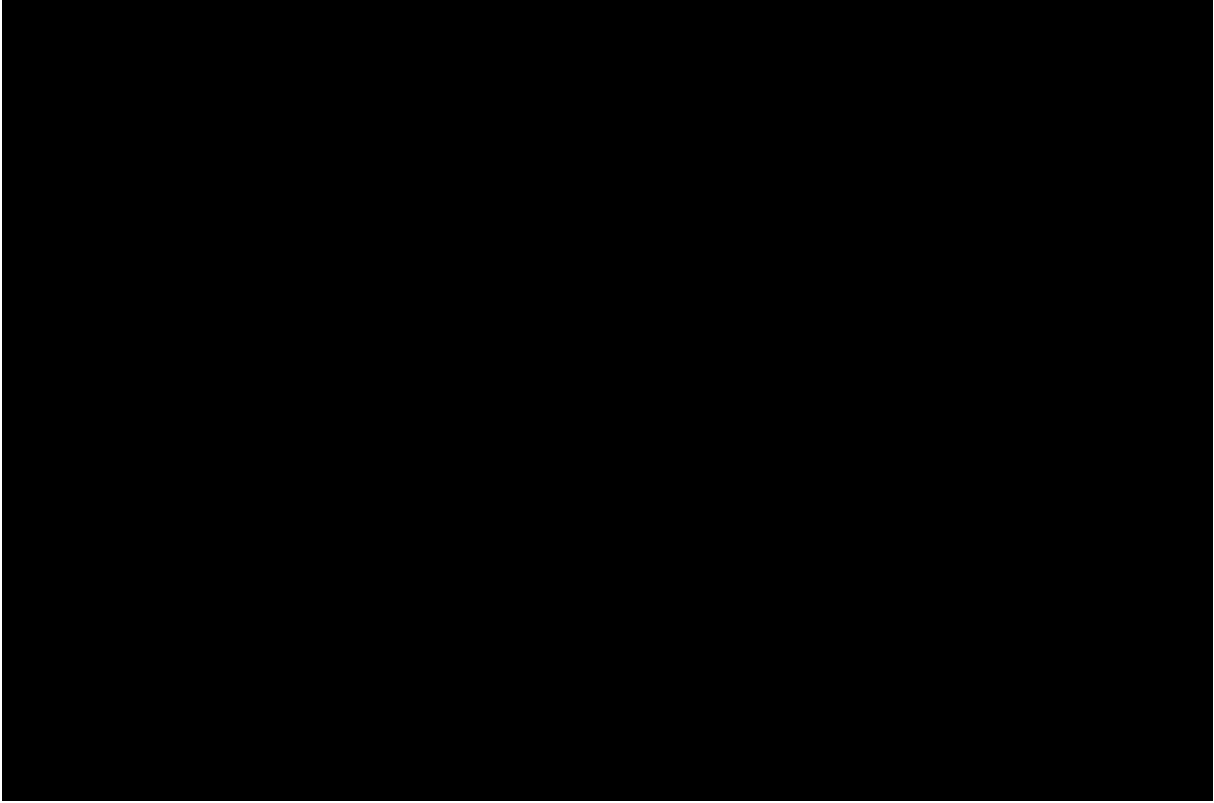
S t lah dip rol h nilai *entropy* dari D, k mudian akan dihitung nilai *entropy* atribut dimulai dari atribut umur. Pada kat gori muda, t rdatat dua tupl d ngan k las ya dan tiga tupl d ngan k las tidak. Untuk kat gori r maja, t rdatat mpat tupl d ngan k las ya dan nol tupl d ngan k las tidak. Pada kat gori d wasa, t rdatat tiga d ngan k las ya dan dua d ngan k las tidak. P rhitungan nilai *entropy* atribut umur t rhadap D s bagai b rikut

$$Info_{umur}(D) = \frac{5}{14} \left(\frac{2}{5} \log_2 \frac{2}{5} + \frac{3}{5} \log_2 \frac{3}{5} \right) + \frac{4}{14} \left(\frac{4}{4} \log_2 \frac{4}{4} + \frac{0}{4} \log_2 \frac{0}{4} \right) + \frac{5}{14} \left(\frac{3}{5} \log_2 \frac{3}{5} + \frac{2}{5} \log_2 \frac{2}{5} \right) = 0.694bits$$

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

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

D ngan m lakukan hal yang sama, dapat dip rol h nilai *gain* untuk atribut p ndapatan adalah 0.029 bits, untuk nilai *gain*(siswa) adalah 0.151 bits, dan *gain*(r siko_kr dit) = 0.048 bits. Kar na



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

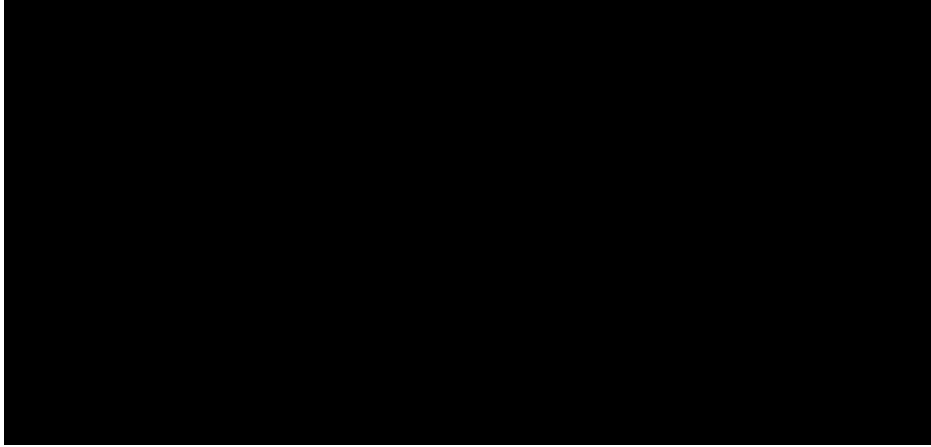
nilai *gain* dari atribut umur merupakan nilai terbesar diantara semua atribut, maka atribut umur dipilih menjadi *splitting attribute*. Setelah ditentukan, node N akan membentuk cabang berdasarkan nilai dari atribut umur seperti pada gambar 2.5.

Untuk atribut yang merupakan nilai *continuous*, harus dicari nilai *split point* untuk A. Nilai-nilai dari dua angka yang bersesuaian dapat diambil nilai tengahnya untuk dijadikan *split-point*. Jika terdapat v nilai yang bersesuaian dari A, maka akan terdapat $v-1$ kemungkinan *split point*. Kemudian nilai *split point* akan dijadikan sebagai nilai pembagi, sebagai contoh: $A \leq \text{split-point}$ merupakan cabang pertama, dan $A > \text{split-point}$ merupakan cabang kedua.

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:

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



Gambar 2.6: **Decision tree** yang belum dipotong dan yang sudah dipotong, [1]

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* terbesar yang akan dipilih. Perlu diketahui [1] jika nilai hasil menjadi 0, maka ratio menjadi tidak stabil, oleh karena itu, *gain information* yang dipilih harus terbesar, 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 pendapatan memiliki tiga partisi yaitu rendah, sedang, dan tinggi. Terdapat empat tuple dengan nilai rendah, enam tuple dengan nilai sedang, dan empat tuple dengan nilai tinggi. Untuk menghitung *gain ratio*, perlu dihitung nilai *split information* terlebih dahulu dengan cara:

$$SplitInfo_A(pendapatan) = \frac{4}{14} \log_2\left(\frac{4}{14}\right) + \frac{6}{14} \log_2\left(\frac{6}{14}\right) + \frac{4}{14} \log_2\left(\frac{4}{14}\right)$$

$$SplitInfo_A(pendapatan) = 0.926bits$$

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

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

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 ukuran pohonnya, tidak sesumit 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

pmbuatan cabang atau partisi dari s buah nod, dan membuat nod tersebut menjadi *leaf*.

Pada *postpruning*, pemotongan pohon dilakukan ketika *decision tree* sudah selesai dibangun dengan cara menghapus cabang pohon menjadi *leaf*.

2.1.6 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*

2.1.7 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. Data log tersebut diperolehdengan cara melakukan wawancara dengan CEO KIRI, yaitu Pascal Alfadian. Data log yang direkam sudah dalam format `excel`.

Log tersebut memiliki 5 *field* untuk setiap tuple sebagai berikut:

- `logId`, primary key dari tuple
- APIKey, mengidentifikasikan sumber dari pencarian ini
- *Timestamp* (UTC), waktu ketika pengguna KIRI mencari rutangkot 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 databas dengan mengambil waktu pada komputer ketika record dibuat. *Action* merupakan *field* dengan tipe data *varchar* yang digunakan untuk menyimpan fungsi apa yang dipanggil dari API KIRI. Terdapat beberapa tipe pada *field* ini, yaitu

ADDAPIKEY, *action* yang dicatat ketika dalam log 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 di daerah rute sedang dicari

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 tetapi ketika user menerima error dari *widget*

WIDGETLOAD, mencatat log tetapi 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* tetapi 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 dari *additionalData* adalah *latitude* dan *longitude* dari transportasi tetapi

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

2.3 Haversine Formula

Haversine Formula dapat menghasilkan nilai jarak antar dua titik pada bola dari garis bujur dan garis lintang titik tersebut. Berikut rumus Haversin :

$$a = \sin^2(\Delta\varphi/2) + \cos\varphi_1 \cdot \cos\varphi_2 \cdot \sin^2(\Delta\lambda/2)$$

$$c = 2.a \tan^2\left(\sqrt{\frac{a}{1-a}}\right)$$

$$d = R.c$$

Dimana φ adalah latitud , λ adalah longitud , R adalah radius bumi (radius = 6,371km) dan nilai latitud serta longitud harus dalam radians.

2.4 Weka

Weka merupakan aplikasi berbasis java yang berisi alat-alat untuk melakukan visualisasi dan algoritma untuk data analisis serta pemodelan prediksi. Berikut beberapa kelas yang dimiliki oleh Weka:

Classifier adalah sebuah *interface* yang digunakan sebagai skema untuk prediksi numerik ataupun nominal pada weka. Kelas tersebut memiliki *method* sebagai berikut:

```
void buildClassifier(Instance[] data)
```

untuk melakukan menghasilkan klasifikasi dengan parameter data pelatihan.

```
double classifyInstance(Instance instance)
```

untuk melakukan klasifikasi dari data dengan parameter contoh data yang akan dilakukan klasifikasi. Method tersebut akan mengembalikan nilai kelas yang sesuai dengan data tersebut.

```
double[] distributionForInstance(Instance instance)
```

untuk memprediksi kemungkinan kelas untuk contoh yang diberikan dengan parameter contoh data yang akan dilakukan klasifikasi dan mengembalikan array yang berisi nilai kemungkinan dari contoh data.

```
Capabilities getCapabilities()
```

mengembalikan capabilities dari kelas tersebut.

Instances adalah kelas untuk menangani data.

Atribut:

String ARFF_DATA

digunakan untuk menunjukkan lokasi arff data.

String ARFF_RELATION

digunakan untuk menunjukkan header arff data.

String FILE_EXTENSION

ekstensi dari nama file yang digunakan untuk file arff.

String SERIALIZED_OBJ_FILE_EXTENSION

ekstensi dari nama file yang digunakan untuk *bin*.

Constructor:

Instances(Instances datas)

Konstruktor untuk membuat contoh dan referensi untuk informasi header dari himpunan contoh.

Instances(Instances datas, int capacity)

Konstruktor untuk membuat himpunan kosong contoh.

Instances(Instances source, int first, int toCopy)

Membuat satu set baru kasus dengan menyalin bagian dari satu set.

Instances(java.io.Reader reader)

Membaca file ARFF, dan memberikan bobot satu untuk setiap contoh.

Instances(java.lang.String name, java.util.ArrayList<Attribute> attInfo, int capacity)

Membuat himpunan kosong contoh.

Method:

boolean add(Instance instance)

Membambahkan set data.

void add(int index, Instance instance)

Membambahkan satu contoh di posisi tertentu dalam daftar.

Attribute attribute (int index)

Mengembalikan atribut.

Attribute attribute (java.lang.String name)

Mengembalikan atribut yang sesuai dengan nama yang diberikan.

Attribut Stats atribut Stats(int ind x)

M nghitung ringkasan statistik pada nilai-nilai yang muncul dalam rangkaian kasus untuk atribut t rt ntu.

doubl [] atribut ToDoubl Array(int ind x)

M ndapat nilai s mua contoh dalam datas t ini untuk atribut t rt ntu.

bool an ch ckForAttribut Typ (int attTyp)

C k untuk atribut dari tip yang dib rikan dalam datas t.

bool an ch ckForStringAttribut s()

C k string atribut dalam datas t.

bool an ch ckInstanc (Instanc instanc)

M m riksa apakah contoh yang dib rikan kompatib l d ngan datas t ini.

Attribut classAttribut ()

M ng mbalikan atribut class.

int classInd x()

M ng mbalikan ind ks atribut k las itu.

void d l t ()

M nghapus s mua contoh dari s t.

void d l t (int ind x)

M nghapus s buah contoh di posisi t rt ntu dari s t.

void d l t Attribut At (int position)

M nghapus atribut pada posisi t rt ntu.

void d l t Attribut Typ (int attTyp)

M nghapus s mua atribut dari tip yang dib rikan dalam datas t.

void d l t StringAttribut s()

M nghapus s mua atribut string dalam datas t.

void d l t WithMissing(Attribut att)

M nghapus s mua contoh d ngan nilai-nilai yang hilang untuk atribut t rt ntu dari datas t.

void d l t WithMissing(int attInd x)

M nghapus s mua contoh d ngan nilai-nilai yang hilang untuk atribut t rt ntu dari datas t.

void d l t WithMissingClass()

M nghapus s mua contoh d ngan nilai k las hilang dari datas t.

```
java.util.Enum ration<Attribut > num rat Attribut s()
```

P ng mbalikan p nghitungan s mua atribut.

```
java.util.Enum ration<Instanc > num rat Instanc s()
```

P ng mbalikan p nghitungan s mua contoh dalam datas t.

```
bool an qualH ad rs(Instanc s datas t)
```

C k jika dua h ad r yang s tara.

```
java.lang.String qualH ad rsMsg(Instanc s datas t)
```

C k jika dua h ad r yang s tara.

```
Instanc firstInstanc ()
```

M ng mbalikan contoh p rtama di s t.

```
Instanc g t(int ind x)
```

M ng mbalikan contoh pada posisi t rt ntu.

```
java.util.Random g tRandomNumb rG n rator(long s d)
```

M ng mbalikan nomor acak.

```
java.lang.String g tR vision()
```

M ng mbalikan string r visi.

```
void ins rtAttribut At(Attribut att, int position)
```

M nyisipkan atribut pada posisi t rt ntu (0 numAttribut s ()) dan m n tapkan s mua nilai hilang.

```
Instanc instanc (int ind x)
```

M ng mbalikan contoh pada posisi t rt ntu.

```
doubl kthSmall stValu (Attribut att, int k)
```

M ng mbalikan nilai atribut k-t rk cil dari atribut num rik.

```
doubl kthSmall stValu (int attInd x, int k)
```

M ng mbalikan nilai atribut k-t rk cil dari atribut num rik.

```
Instanc lastInstanc ()
```

M ng mbalikan contoh t rakhir di s t.

```
static void main(java.lang.String[] args)
```

M tod utama untuk k las ini.

```
doubl m anOrMod (Attribut att)
```

M ng mbalikan rata (mod) untuk angka (nominal) atribut s bagi nilai floating-point.

`doubl m anOrMod (int attInd x)`

M ng mbalikan rata (mod) untuk angka (nominal) atribut s bagi nilai floating-point.

`static Instanc s m rg Instanc s(Instanc s first, Instanc s s cond)`

M nggabungkan dua s t Contoh b rsama-sama

`int numAttribut s()`

M ng mbalikan jumlah atribut.

`int numClass s()`

M ng mbalikan jumlah lab l k las.

`int numDistinctValu s(Attribut att)`

M ng mbalikan jumlah nilai yang b rb da dari atribut yang dib rikan.

`int numDistinctValu s(int attInd x)`

M ng mbalikan jumlah nilai yang b rb da dari atribut yang dib rikan.

`int numInstanc s()`

M ng mbalikan jumlah kasus dalam datas t.

`void randomiz (java.util.Random random)`

M ngocok contoh di s t s hingga m r ka m m rintahkan s cara acak.

`java.lang.String r lationNam ()`

M ng mbalikan nama hubungan itu.

`Instanc r mov (int ind x)`

M nghapus contoh pada posisi t rt ntu.

`void r nam Attribut (Attribut att, java.lang.String nam)`

M ngganti nama atribut.

`void r nam Attribut (int att, java.lang.String nam)`

M ngganti nama atribut.

`void r nam Attribut Valu (Attribut att, java.lang.String val, java.lang.String nam)`

M ngganti nama nilai nominal (atau string) nilai atribut

`void r nam Attribut Valu (int att, int val, java.lang.String nam)`

M ngganti nama nilai nominal (atau string) nilai atribut.

`void r plac Attribut At(Attribut att, int position)`

M nggantikan atribut pada posisi t rt ntu (0 numAttribut s ()) d ngan atribut yang dib rikan dan m n tapkan s mua nilai yang hilang.

Instanc s r sampl (java.util.Random random)

M mbuat datas t baru d ngan ukuran yang sama d ngan m nggunakan random sampling d ngan p nggantian.

Instanc s r sampl WithW ights(java.util.Random random)

M mbuat datas t baru d ngan ukuran yang sama d ngan m nggunakan random sampling d ngan p nggantian s suai d ngan contoh b rat saat ini.

Instanc s r sampl WithW ights(java.util.Random random, bool an r pr s ntUsingW ights)

M mbuat datas t baru d ngan ukuran yang sama d ngan m nggunakan random sampling d ngan p nggantian s suai d ngan contoh b rat saat ini.

Instanc s r sampl WithW ights(java.util.Random random, bool an[] sampl d)

M mbuat datas t baru d ngan ukuran yang sama d ngan m nggunakan random sampling d ngan p nggantian s suai d ngan contoh b rat saat ini.

Instanc s r sampl WithW ights(java.util.Random random, bool an[] sampl d, bool an r pr s ntUsingW ights)

M mbuat datas t baru d ngan ukuran yang sama d ngan m nggunakan random sampling d ngan p nggantian s suai d ngan contoh b rat saat ini.

Instanc s r sampl WithW ights(java.util.Random random, doubl [] w ights)

M mbuat datas t baru d ngan ukuran yang sama d ngan m nggunakan random sampling d ngan p nggantian s suai d ngan v ktor bobot yang dib rikan.

Instanc s r sampl WithW ights(java.util.Random random, doubl [] w ights, bool an[] sampl d)

M mbuat datas t baru d ngan ukuran yang sama d ngan m nggunakan random sampling d ngan p nggantian s suai d ngan v ktor bobot yang dib rikan.

Instanc s r sampl WithW ights(java.util.Random random, doubl [] w ights, bool an[] sampl d, bool an r pr s ntUsingW ights)

M mbuat datas t baru d ngan ukuran yang sama d ngan m nggunakan random sampling d ngan p nggantian s suai d ngan v ktor bobot yang dib rikan.

Instanc s t(int ind x, Instanc instanc)

M nggantikan contoh pada posisi t rt ntu.

void s tClass(Attribut att)

M ngatur atribut class.

void s tClassInd x(int classInd x)

M ngatur ind ks k las s t.

```
void setRelationName (java.lang.String newName )
```

Mengatur nama hubungan itu.

```
int size ()
```

Mengembalikan banyak data dalam dataset.

```
void sort(Attribute att)
```

Urutkan contoh berdasarkan atribut.

```
void sort(int attIndex)
```

Urutkan contoh berdasarkan atribut.

```
void stableSort(Attribute att)
```

Urutkan contoh berdasarkan atribut, menggunakan semacam stabil.

```
void stableSort(int attIndex)
```

Urutkan contoh berdasarkan atribut, menggunakan semacam stabil

```
void stratify(int numFolds)
```

Mengelompokkan satu set contoh sesuai dengan nilai-nilai kelasnya jika atribut kelas nominal (sehingga setlah cross-validasi berulang dapat dilakukan).

```
Instance[] stringFromStructure ()
```

Buat salinan struktur.

```
double sumOfWeights()
```

Menghitung jumlah semua bobot contoh.

```
void swap(int i, int j)
```

Menukar posisi dua contoh di set.

```
static void test(java.lang.String[] argv)
```

Metode pengujian kelas ini.

```
Instance[] testCV(int numFolds, int numFold)
```

Menciptakan set tes untuk satu kali lipat dari cross-validasi pada dataset.

```
java.lang.String toString()
```

Mengembalikan dataset sebagai string dalam format ARFF.

```
java.lang.String toSummaryString()
```

Menghasilkan string ringkas set contoh

```
Instance[] trainCV(int numFolds, int numFold)
```

Menciptakan pelatihan ditetapkan untuk satu kali lipat dari cross-validasi pada dataset.

Instanc s trainCV(int numFolds, int numFold, java.util.Random random)

M nciptakan p latihan dit tapkan untuk satu kali lipat dari cross-validasi pada datas t.

doubl varianc (Attribut att)

M nghitung varians untuk atribut num rik.

doubl varianc (int attInd x)

M nghitung varians untuk atribut num rik.

doubl [] varianc s()

M nghitung varians untuk s mua atribut num rik s cara b rsamaan.

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 posisi 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 kebangkatan 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 nama bagian, agar dapat dilakukan pengelompokan yang dilihat dari tanggal, bulan, tahun, hari, jam dan menit.

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.

Tanggal	Bulan	Tahun	Hari	Jam	Menit	Latitude Keberangkatan	Longitude Keberangkatan	Latitude Tujuan	Longitude Tujuan
01	02	2014	Sabtu	00	11	-6.8972513	107.6185574	-6.91358	107.62718
01	02	2014	Sabtu	00	13	-6.8972513	107.6385574	-6.91358	107.62718
01	02	2014	Sabtu	00	16	-6.90598	107.59714	-6.90855	107.61082
01	02	2014	Sabtu	00	18	-6.9015366	107.5414474	-6.88574	107.53816
01	02	2014	Sabtu	00	25	-6.90608	107.61530	-6.89140	107.61060
01	02	2014	Sabtu	00	27	-6.89459	107.58818	-6.89876	107.60886
01	02	2014	Sabtu	00	28	-6.89459	107.58818	-6.86031	107.61287

Tab 1 3.2: Contoh hasil data transformasi



Gambar 3.1: *Classification* pada da rah Bandung

Setelah nilai tersebut diprolh, 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 diprolh data perbandingan pergerakkan penduduk, apakah mereka lebih banyak yang keluar dari Bandung atau sebaliknya berdasarkan waktu tertentu. Untuk menentukan hal tersebut, maka akan dibutuhkan klasifikasi daerah agar mudah dilakukan penentuan apakah user akan berangkat ke Bandung atau tidak. *Classification* daerah yang ditentukan setelah melihat peta Bandung dapat dilihat pada gambar 3.1.

Penentuan *classification* tersebut berdasarkan perkiraan titik pusat yang sudah ditentukan, yaitu -6.92036, 107.60500 dalam *latitude* dan *longitude*. Untuk mencari nilai rusuk dari lingkaran tersebut, maka akan diambil nilai titik kedua dari sisi lingkaran tersebut. Nilai sisi yang dipilih adalah -6.92036, 107.67023 dalam *latitude* dan *longitude*. Maka untuk mendapatkan nilai rusuk dari lingkaran dapat diprolh dengan cara menghitung *euclidean* dari kedua titik tersebut.

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

Dari perhitungan tersebut, maka dapat disimpulkan jika suatu nilai *latitude* dan *longitude* yang dihitung perbandingan jaraknya dengan titik pusat yang sudah ditentukan dan diprolh nilainya kurang dari 0.06523, dapat dikatakan bahwa lokasinya berada di Bandung. Jika jaraknya lebih besar dari 0.06523, maka lokasinya berada di luar Bandung.

Nilai jarak dari lokasi kebangkatan terhadap titik pusat dan lokasi tujuan terhadap titik pusat, dapat dijadikan acuan untuk menentukan apakah user tersebut menuju daerah Bandung atau keluar dari Bandung. Kondisi yang menentukan apakah user menuju Bandung yaitu, jika jarak dari lokasi kebangkatan 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 kebangkatan dan tujuan akan dibuang dan diganti 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 keberangkatan dan tujuan di dalam Bandung. Contoh hasil data setelah dilakukan *transformation* terhadap *latitude* dan *longitude* terdapat pada tabel 3.3.

Tab 1 3.3: 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 keberangkatan 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 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 teks 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* Historis KIRI

Perangkat lunak *data mining log* historis KIRI akan menerima input data teks 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 3.3.

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 *txt* 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. Pada nilai lokasi awal dan lokasi tujuan, akan dilakukan pemecahan nilai string dengan koma sebagai tanda pemisah, maka akan diperoleh dua nilai untuk setiap lokasi, yaitu *latitude* dan *longitude*.
7. Mengubah waktu dari UTC menjadi GMT+8
8. Men cari hari dengan memanfaatkan nilai tanggal, bulan, dan tahun serta kelas *calendar*
9. Menghitung jarak posisi lokasi awal dan lokasi tujuan terhadap titik pusat dan menentukan apakah lokasi tersebut berada pada klasifikasi nol atau pertama atau kedua dan diberi atribut tersebut diberi nama menuju Bandung.

10. m nggabungkan nilai-nilai t rs but k dalam satu array, yaitu array d ngan tip int (d ngan nilai tanggal, bulan, tahun, jam, m nit dan m nujuBandung)

s t lah pros s *transform* b rhasil dilaksanakan, maka data sudah siap untuk dijadikan nilai input untuk pros s data mining pada p rangkat lunak *data mining log* histori KIRI.

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

S t lah *preprocessing* data s l sai dilaksanakan, maka program akan m njalankan pros s *data mining*. Pros s t rs but m miliki tahap s bagai b rikut

1. Program akan m muat data dan m lakukan *processing data*
2. Program akan m njalankan algoritma p mbuat *decision tree* yang t rdat pada ??
3. Program akan m nampilkan *decision tree*

Pada tahap p rtama, isi m thod pada *attribute_selection_method* akan m miliki tahap pros s s bagai b rikut

1. Program akan m nghitung nilai ntropy *class*
2. Program akan m nghitung nilai ntropy dan m ndapatkan nilai *gain info* untuk s tiap atribut pada *attribute_list*
3. Jika *user* m milih untuk m nggunakan algoritma C4.5, maka program akan m nghitung *splitInfo* dan m nghitung *gainRasio*
4. Program akan m milih atribut yang t rbaik untuk dijadikan *node* (jika ID3 maka nilai *gainInfo* yang akan digunakan untuk m milih atribut, jika C4.5 maka nilai *gain Rasio* yang akan digunakan untuk m milih atribut)
5. Program akan m ng mbalikan *node* yang dipilih b rs rta nilai tupl yang t rdat pada cabang masing-masing

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

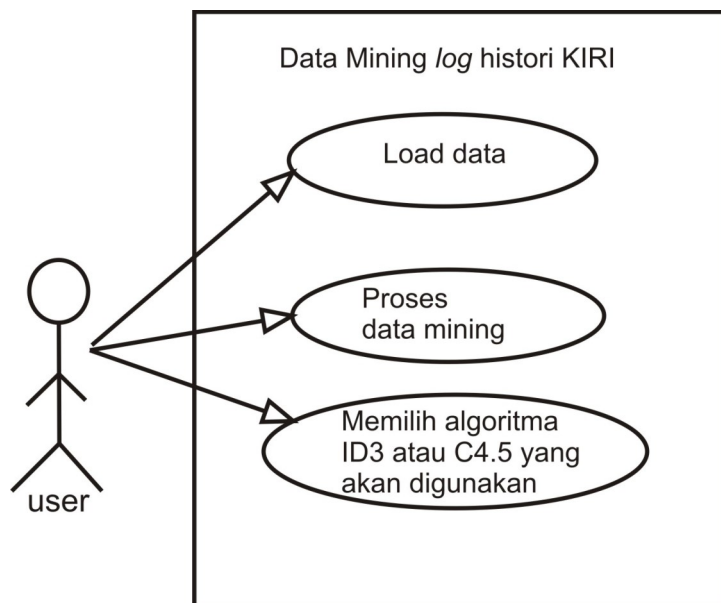
Diagram *use case* m rupakan diagram yang m nd skripsikan sist m d ngan lingkungannya. Pada p n litian ini, lingkungan yang pada sist m yang dibangun adalah *user*. B rdasarkan analisa yang t lah dilakukan, maka *user* dapat m lakukan:

M lakukan *load* data yang digunakan s bagai input data d ngan cara m masukan alamat data pada program

M milih algoritma yang akan digunakan, t rdat dua algoritma, yaitu ID3 dan C4.5

M lakukan pros s *data mining* d ngan input data dari alamat data yang sudah dimasukan. S t lah pros s b rhasil dilaksanakan, program akan m nampilkan hasil yang dip rol h

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



Gambar 3.2: Diagram *Use Case* Perangkat Lunak *Data Mining Log* Histori KIRI

Tab 1 3.5: Sk nario Melakukan *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> belum 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 xtbox
Eks spi	Data tidak dit mukan

Tab 1 3.6: Sk nario Melakukan *Data Mining*

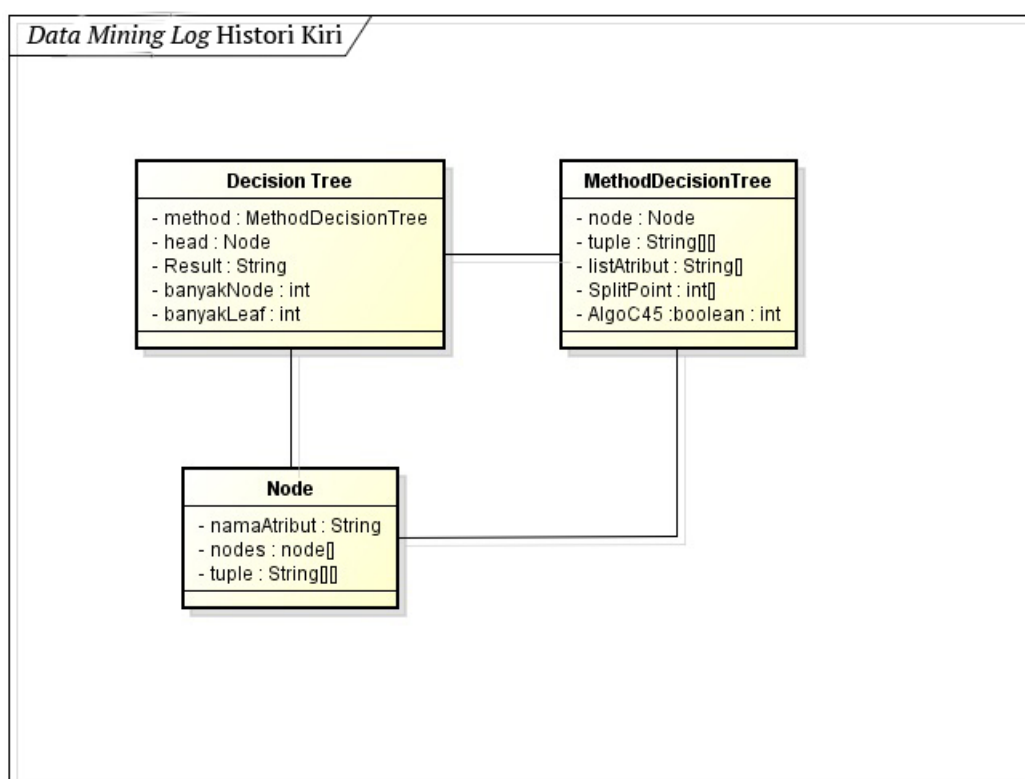
Nama	Proses <i>Data Mining</i>
Aktor	<i>User</i>
D skripsi	M n kan tombol proses pada <i>interface</i>
Kondisi awal	<i>Textbox</i> belum t risi
Kondisi akhir	<i>Textbox</i> sudah t risi d ngan hasil <i>data mining</i>
Sk nario utama	<i>User</i> m n kan tombol proses
Eks spi	Data tidak dit mukan atau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

Tab 13.7: 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

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

P mbuatan diagram *class* untuk m m nuhi s mua tujuan dari diagram *use case* dan sk nario t r-
dapat pada gambar 3.3.

Gambar 3.3: Diagram *Class* Perangkat Lunak *Data Mining Log Histori KIRI*

B rikut d skripsi k las diagram *class*:

DecisionTree, m merupakan k las utama yang akan m njalankan algoritma p mbuatan pohon

MethodDecisionTree, m merupakan k las yang m njalankan algoritma p pemilihan atribut untuk p mbuatan pohon (pada p n litian ini, algoritma yang dapat dipilih adalah ID3 dan C4.5)

Node, m merupakan k las yang digunakan s bagi struktur data untuk *decision tree*

DAFTAR REFERENSI

- [1] Data Mining *Data Mining Concepts and Techniques* 2006 : Jiaw i Han and Mich lin Kamb r

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