

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	21
3 ANALISA	41
3.1 Analisis Data	41
3.1.1 Data Cleaning	41
3.1.2 Data Integration	41
3.1.3 <i>Data Selection</i>	41
3.1.4 <i>Data Transformation</i>	42
3.2 Analisis Perangkat Lunak	46
3.2.1 Diagram Use Case Perangkat Lunak <i>Data Mining Log Histori KIRI</i>	48
3.2.2 Diagram Kelas Perangkat Lunak <i>Data Mining Log Histori KIRI</i>	50
DAFTAR REFERENSI	51
A 100 DATA PERTAMA DARI <i>log HISTORI KIRI</i>	53

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>	18
3.1	<i>Classification</i> pada daerah Bandung	45
3.2	Diagram <i>Use Case</i> P-rangkaian Lunak <i>Data Mining Log Histori KIRI</i>	49
3.3	Diagram <i>Class</i> P-rangkaian Lunak <i>Data Mining Log Histori KIRI</i>	50

DAFTAR TABEL

2.1	tabl 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>	42
3.2	Contoh hasil data transformasi	44
3.3	Contoh hasil data transformasi latitud longitud	46
3.5	Sk nario M lakukan <i>load Data</i>	49
3.6	Sk nario M lakukan <i>Data Mining</i>	49
3.7	Sk nario M milih Algoritma yang Akan Digunakan	49

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 lebih 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

- Men cari 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

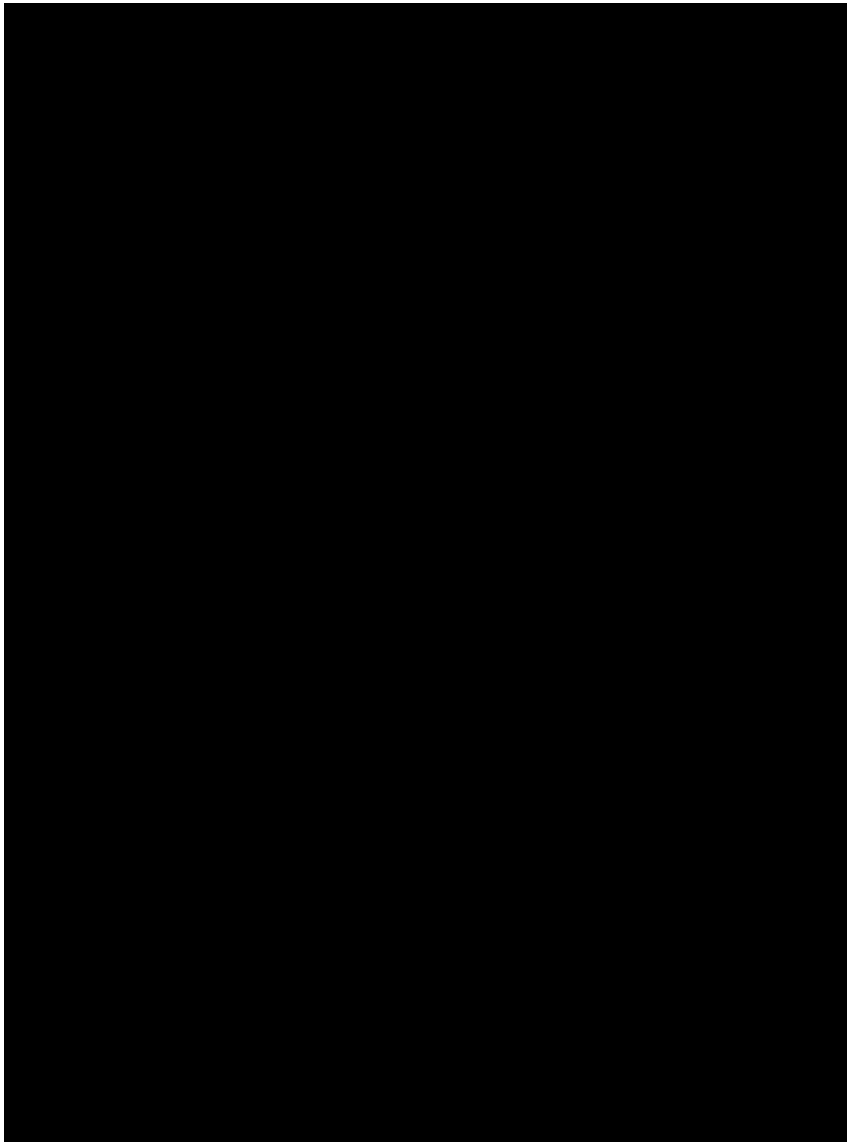
-
- BAB 2: Landasan Teori, teori dasar teori mengenai *data mining*, *data cleaning*, *data integration*, *data selection*, *data transform*, *decision tree*, *pattern evaluation*, *knowledge presentation* dan *log* histori KIRI
 - BAB 3: Teori analisa dasar teori yang akan digunakan, analisa data serta 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*.



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

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 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	Customer	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 *customer* 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 berbagai *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 mempersiapkan 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:

- NPM Mahasiswa
- Nama Mahasiswa
- Jenis Kelamin
- 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 pengubahan data agar siap dilakukan proses *data mining*. *Data transformation* bisa melibatkan:

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

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 memiliki 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

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

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

$$x' = \frac{x - \bar{A}}{A}$$

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 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.

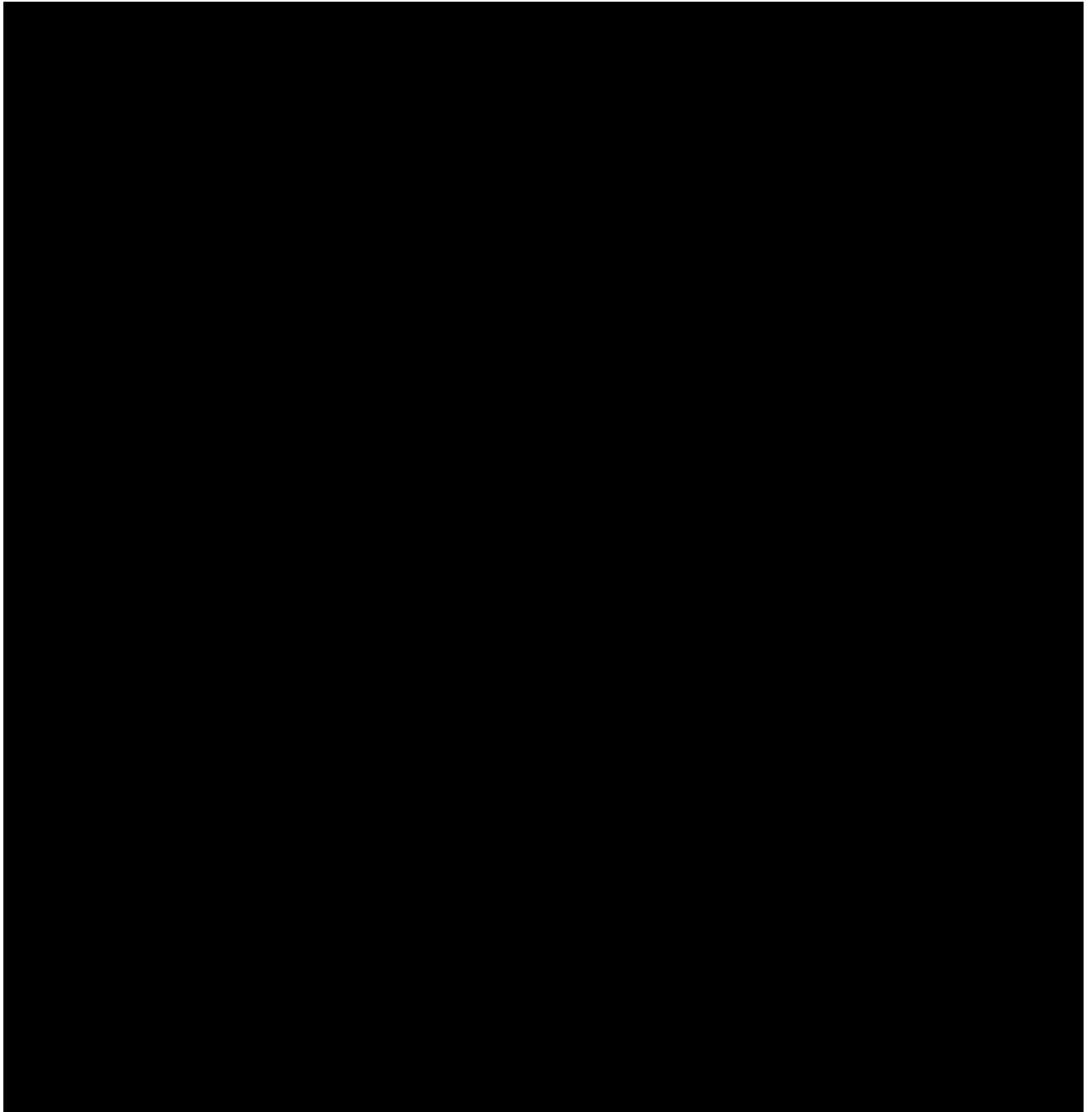
Require: Partisi data, D , merupakan subset data pelatihan dan kelas label

Require: *attribute_list*, merupakan subset dari atribut kandidat

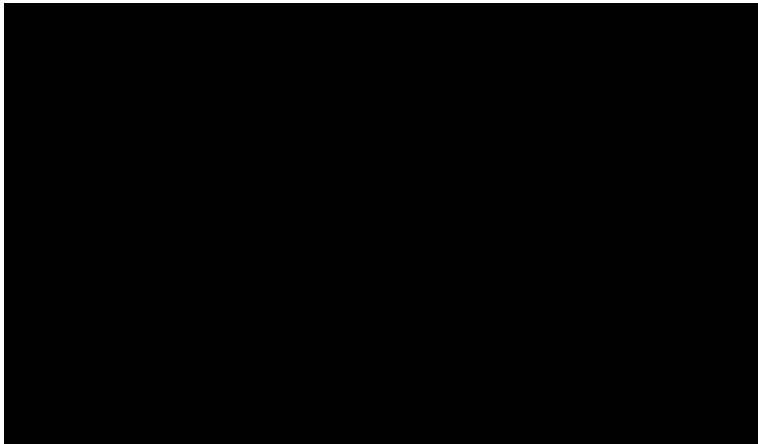
Require: *Attribute_selection_method*, prosedur untuk menentukan *splitting criterion*. Pada input ini, terdapat juga data *splitting_attribute* dan mungkin salah satu dari *split point* atau *splitting subset*

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]

```

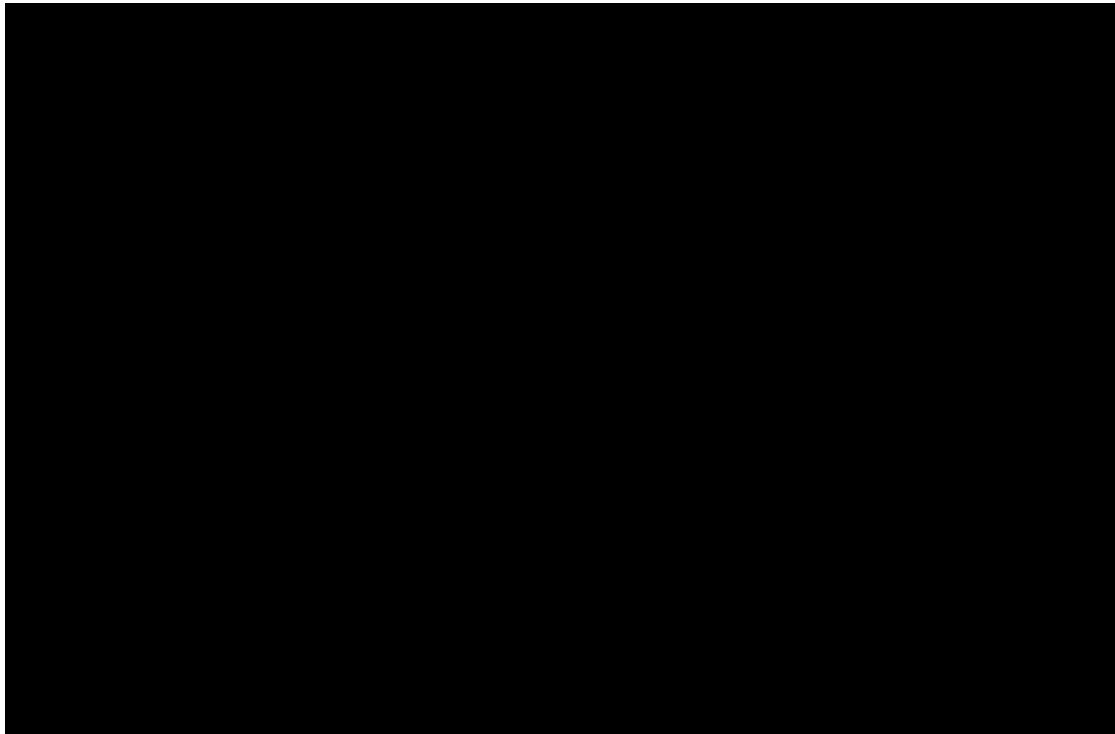
3:  return N s sebagai nod daun d ngan lab l k las C;
4: end if
5: if atribut _list tidak ada nilai atau kosong then
6:  return N s sebagai nod daun d ngan lab l k las yang t rpaling banyak pada D; {majority
    voting}
7: end if
8: m panggil m thod Atribut _s l ction_m thod(D, atribut _list) untuk m 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 panggil *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 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 ya 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 pada baris 4). Disini, akan dilakukan *majority voting* (baris 6) yang akan menghasilkan 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 (baris 15) dan *leaf* akan dibuat dengan *majority class* pada D (baris 16).

Pada baris 21, akan dikembalikan nilai *decision tree* yang telah dibuat.

subsubsc tion *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* menyediaan peringkat untuk setiap atribut pada training tuple. Jika *splitting criterion* merupakan nilai *continous* atau *binary trees*, maka nilai *split point* dan *splitting subset* harus ditentukan sebagai bagian dari *splitting criterion*. Contoh dari *attribute selection measure* adalah *information gain*, *gain ratio*, dan *gini index*.

Notasi yang digunakan adalah sebagai berikut. D merupakan data partisi, 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 $C_{i,d}$.

ID3

ID3 merupakan teknik untuk membuat *decision tree* dengan menggunakan *information gain* sebagai *attribute selection measure* untuk memilih atribut. Cara ID3 mendapatkan *information gain* dengan menggunakan *entropy*. *Entropy* adalah ukuran *impurity* (ketiadaan informasi) 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

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

$|D_j|/D$ 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 $Info_A(D)$

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

Atribut yang memiliki nilai *gain information* yang terbesar akan dipilih sebagai output dari metode ini.

contoh kasus untuk ID3, dalam pencarian *information gain*:

Tab 1 2.2: Contoh training set

RID	umur	pendapatan	siswa	tersikokur dit	Class: membership komputer
1	muda	tinggi	tidak	cukup	tidak
2	muda	tinggi	tidak	baik	tidak
3	remaja	tinggi	tidak	cukup	ya
4	dewasa	sedang	tidak	cukup	ya
5	dewasa	rendah	ya	cukup	ya
6	dewasa	rendah	ya	baik	tidak
7	remaja	rendah	ya	baik	ya
8	muda	sedang	tidak	cukup	tidak
9	muda	rendah	ya	cukup	ya
10	dewasa	sedang	ya	cukup	ya
11	muda	sedang	ya	baik	ya
12	remaja	sedang	tidak	baik	ya
13	remaja	tinggi	ya	cukup	ya
14	dewasa	sedang	tidak	baik	tidak

Pada tabel 2.2, terdapat *training set*, D. Atribut kelas label merupakan dua nilai yang berbeda yaitu ya dan tidak, maka dari itu, nilai $m = 2$. C_1 diisi dengan kelas label bernilai ya, sedangkan C_2 diisi dengan kelas label bernilai tidak. Terdapat sembilan tuple atribut kelas label dengan nilai ya dan lima tuple dengan nilai tidak. Untuk dapat menentukan *splitting criterion*, *information gain* harus dihitung untuk setiap atribut terlebih dahulu. Perhitungan *entropy* untuk D adalah

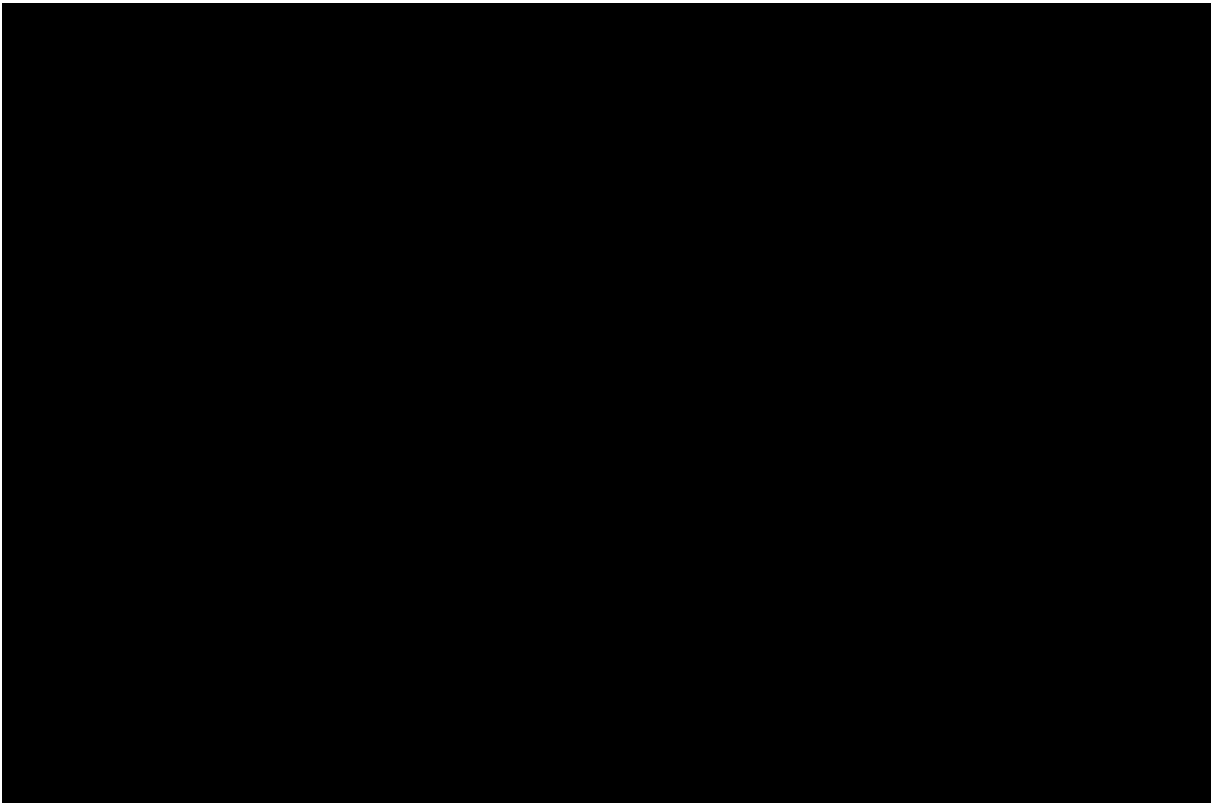
$$Info(D) = -\frac{9}{14} \log_2\left(\frac{9}{14}\right) - \frac{5}{14} \log_2\left(\frac{5}{14}\right) = 0.940bits$$

Selanjutnya dipilih nilai *entropy* dari D, kemudian akan dihitung nilai *entropy* atribut dimulai dari atribut umur. Pada kategori muda, terdapat dua tuple dengan kelas ya dan tiga tuple dengan kelas tidak. Untuk kategori remaja, terdapat empat tuple dengan kelas ya dan nol tuple dengan kelas tidak. Pada kategori dewasa, terdapat tiga dengan kelas ya dan dua dengan kelas tidak. Perhitungan nilai *entropy* atribut umur terhadap D sebagai berikut

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

Selanjutnya mendapatkan *entropy* dari atribut umur, maka nilai *gain information* dari atribut umur adalah

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



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

Dengan melakukan hal yang sama, dapat diperoleh nilai *gain* untuk atribut *p* didapatkan adalah 0.029 *bits*, untuk nilai *gain*(siswa) adalah 0.151 *bits*, dan $gain(r_{siko_kr\ dit}) = 0.048$ *bits*. Karena nilai *gain* dari atribut umur merupakan nilai terbesar diantara semua atribut, maka atribut umur dipilih menjadi *splitting attribute*. Setelah ditentukan, node *N* akan membagi untuk 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 bersebelahan dapat diambil nilai tengahnya untuk dijadikan *split-point*. Jika terdapat *v* nilai yang berbedanya dari *A*, maka akan terdapat *v-1* kemungkinan *split point*. Kemudian nilai *split point* akan dijadikan sebagai nilai pembagian, sebagai contoh: $A \leq split-point$ merupakan cabang pertama, dan $A > split-point$ merupakan cabang kedua.

C4.5

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]. Selain itu, ID3 dapat menghasilkan *decision tree* yang memprediksi cara berlebihan (*overestimated*) atau disebut juga *overfitting*. Hal ini dikarenakan pohon yang dihasilkan terlalu detail hingga data input memiliki hasil prediksi yang pasti.

C4.5 merupakan teknik lanjutan dari ID3, yang menggunakan *gain ratio* sebagai *attribute sele-*

ction measure untuk memilih atribut. Kemudian, C4.5 melakukan *tree pruning* untuk menghindari *overfitting*.

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:

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

Dimana $|D|$ merupakan banyak data dan $|D_j|$ merupakan banyak data suatu nilai pada atribut. Sehingga 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 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:

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

$$SplitInfo_A(pendapatan) = 0.926bits$$

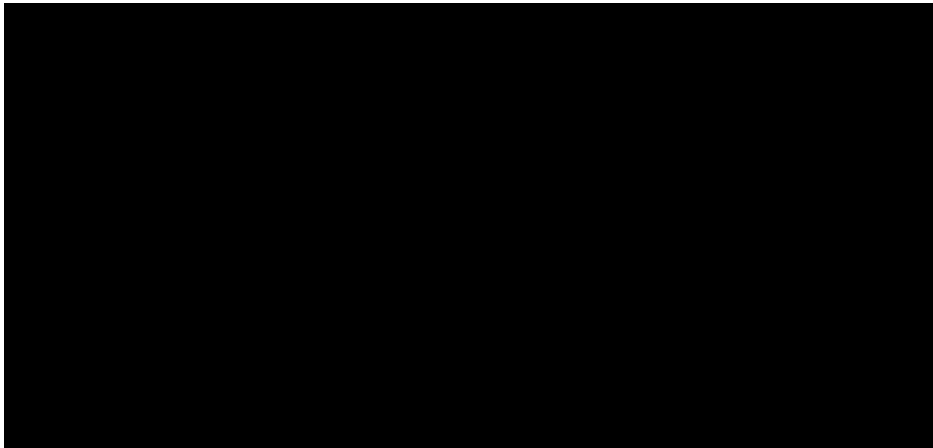
Jika nilai *gain information* dari *income* adalah 0.029, maka, dapat diperoleh *gain ratio* dari pendapatan adalah

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

Maka nilai *gain ratio* dari atribut pendapatan adalah 0.031 bits. Perhitungan tersebut dilakukan pada semua atribut, dan atribut yang memiliki nilai *gain ratio* yang terbesar adalah atribut yang dipilih.

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 serta kemampuan mengklasifikasikan 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*.



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

Pada *prepruning*, pemotongan pohon dilakukan dengan cara menghentikan dan tidak melanjutkan pembuatan cabang atau partisi dari sebuah node, dan membuat node tersebut menjadi *leaf*.

Pada *postpruning*, pemotongan pohon dilakukan ketika *decision tree* sudah selesai dibangun dengan cara mengubah 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 diprolah dengan cara melakukan wawancara dengan CEO KIRI, yaitu Pascal Alfadian. Data log yang diberikan sudah dalam format `excel`.

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

- `logId`, primary key dari tuple
- APIKey, mengidentifikasi sumber dari pencarian ini

- *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

2.3 Haversine Formula

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

$$a = \sin^2\left(\frac{|\phi_1 - \phi_2|}{2}\right) + \cos \phi_1 \cdot \cos \phi_2 \cdot \sin^2\left(\frac{|\lambda_1 - \lambda_2|}{2}\right)$$

$$c = 2 \cdot \arctan^2(\sqrt{a}; \sqrt{1-a})$$

$$d = R \cdot c$$

Dimana

- ϕ adalah lintang dalam radian
- λ adalah longitud dalam radian
- R adalah radius bumi (radius = 6,371km)

Studi kasus untuk perhitungan Haversin sebagai berikut: Jika kita ingin menghitung jarak dua titik dari daerah Jakarta ke Surabaya, dengan titik pada Jakarta adalah -6.211544, 106.845172 dan

$$c = 2:0.0026906745 \tan^2(\sqrt{0.0026906745}; \sqrt{1 - 0.0026906745})$$

$$c = 0:1037900036$$

$$d = 6:371.0:1037900036$$

$$d = 0:6612461130 * 1000km$$

$$d = 661.2461130km$$

Dengan menggunakan rumus Haversin, maka jarak antara dua titik tersebut adalah 661.246 km

2.4 Weka

Weka merupakan aplikasi berbasis java yang berisi alat-alat untuk melakukan visualisasi dan algoritma untuk data analisis serta platform prediksi. Weka juga mempunyai file `weka-src.jar` yang berisi kelas-kelas yang dipakai oleh aplikasi weka sehingga user dapat menggunakannya untuk membuat program java yang berfungsi untuk *data mining*. Berikutnya kita akan membahas kelas-kelas yang dimiliki oleh Weka:

Classifier adalah sebuah *interface* yang digunakan sebagai skema untuk prediksi numerik ataupun nominal pada weka. Kelas-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 melakukan prediksi keanggotaan kelas untuk contoh yang diberikan dengan parameter contoh data yang akan dilakukan klasifikasi dan mengembalikan array yang berisi nilai keanggotaan dari contoh data.
- `Capabilities getCapabilities()`
mengembalikan capabilities dari kelas-kelas tersebut.

Instance adalah *interface* yang mewakili data.

Method:

- `Attribute[] attributes(int index)`
Mengembalikan atribut dari indeks yang diberikan.

- `Attribut classAttribut ()`
Mengembalikan atribut kelas.
- `int classInd x()`
Mengembalikan indeks atribut kelas itu.
- `boolean classIsMissing()`
Mengembalikan apakah kelas turunan hilang.
- `double classValu ()`
Mengembalikan nilai kelas contoh sebagai angka floating-point.
- `Instansi datas t()`
Mengembalikan datas t.
- `void deleteAttribut At(int position)`
Menghapus atribut pada posisi tertentu.
- `java.util.Enum ration<Attribut > num rat Attribut s()`
Mengembalikan penghitungan semua atribut.
- `boolean qualH ad rs(Instansi inst)`
Pengujian jika h ad r dari dua contoh yang setara.
- `java.lang.String qualH ad rsMsg(Instansi inst)`
Memeriksa apakah h ad r dari dua contoh yang setara.
- `boolean hasMissingValu ()`
Tersedia apakah sebuah contoh memiliki nilai yang hilang.
- `int x(int position)`
Mengembalikan indeks dari atribut yang tersimpan di posisi tertentu.
- `void insertAttribut At(int position)`
Menyisipkan atribut pada posisi tertentu.
- `boolean isMissing(Attribut att)`
Pengujian jika nilai tertentu yang hilang.
- `boolean isMissing(int attInd x)`
Pengujian jika nilai tertentu yang hilang.
- `boolean isMissingSpars (int ind xOfInd x)`
Pengujian jika nilai tertentu yang hilang.

- `Instance mergeInstance (Instance inst)`
Menggabungkan contoh yang diberikan dan mengembalikan hasilnya.
- `int numAttributes()`
Mengembalikan jumlah atribut.
- `int numClasses()`
Mengembalikan jumlah label kelas.
- `int numValues()`
Mengembalikan jumlah nilai.
- `Instance relationalValue (Attribute att)`
Mengembalikan nilai relational atribut relational.
- `Instance relationalValue (int attIndex)`
Mengembalikan nilai relational atribut relational.
- `void replaceMissingValues(double[] array)`
Menggantikan semua nilai yang hilang dalam contoh dengan nilai-nilai yang terkandung dalam array yang diberikan.
- `void setClassMissing()`
Menetapkan nilai kelas contoh untuk hilang.
- `void setClassValue (double value)`
Menetapkan nilai kelas turunan dengan nilai yang diberikan (format floating-point).
- `void setClassValue (java.lang.String value)`
Menetapkan nilai kelas turunan dengan nilai yang diberikan.
- `void setDataSet (Instance[] instances)`
Mengatur referensi data set.
- `void setMissing (Attribute att)`
Menetapkan nilai tertentu dijadikan hilang.
- `void setMissing (int attIndex)`
Menetapkan nilai tertentu dijadikan hilang.
- `void setValue (Attribute att, double value)`
Menetapkan nilai tertentu dalam hal untuk nilai yang diberikan (format floating-point).
- `void setValue (Attribute att, java.lang.String value)`
Menetapkan nilai atribut nominal atau string ke nilai yang diberikan.

- `void s tValu (int attInd x, doubl valu)`
M n tapkan nilai t rt ntu untuk nilai yang dib rikan (format floating-point).
- `void s tValu (int attInd x, java.lang.String valu)`
M n tapkan nilai atribut nominal atau string k nilai yang dib rikan.
- `void s tValu Spars (int ind xOfInd x, doubl valu)`
M n tapkan nilai t rt ntu dalam contoh d ngan nilai yang dib rikan (format floating-point).
- `void s tW ight(doubl w ight)`
M ngatur b rat contoh.
- `java.lang.String stringValu (Attribut att)`
M ng mbalikan nilai nominal, string, tanggal, atau atribut r lasional untuk contoh s bagai string.
- `java.lang.String stringValu (int attInd x)`
M ng mbalikan nilai nominal, string, tanggal, atau atribut r lasional untuk contoh s bagai string.
- `doubl [] toDoubl Array()`
M ng mbalikan nilai-nilai masing-masing atribut s bagai array ganda.
- `java.lang.String toString(Attribut att)`
M ng mbalikan d skripsi satu nilai dari contoh s bagai string.
- `java.lang.String toString(Attribut att, int aft rD cimalPoint)`
M ng mbalikan d skripsi satu nilai dari contoh s bagai string.
- `java.lang.String toString(int attInd x)`
M ng mbalikan d skripsi satu nilai dari contoh s bagai string.
- `java.lang.String toString(int attInd x, int aft rD cimalPoint)`
M ng mbalikan d skripsi satu nilai dari contoh s bagai string.
- `java.lang.String toStringNoW ight()`
M ng mbalikan d skripsi satu contoh (tanpa b rat ditambahkan).
- `doubl valu (Attribut att)`
M ng mbalikan nilai atribut contoh dalam format int rnal.
- `doubl valu (int attInd x)`
M ng mbalikan nilai atribut contoh dalam format int rnal.
- `doubl valu Spars (int ind xOfInd x)`
M ng mbalikan nilai atribut contoh dalam format int rnal.

- `double weight()`

Mengembalikan berat contoh itu.

Instances adalah kelas untuk menangani *set* data.

Atribut:

- `String ARFF_DATA`
digunakan untuk menunjukkan *section* *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 dataset)`
Konstruktor minimalis untuk informasi *header* dari himpunan contoh.
- `Instances(Instances dataset, int capacity)`
Konstruktor untuk menciptakan himpunan kosong contoh.
- `Instances(Instances source, int first, int toCopy)`
Menciptakan 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)`
Menciptakan himpunan kosong contoh.

Method:

- `boolean add(Instances instance)`
Menambahkan *set* data.
- `void add(int index, Instances instance)`
Menambahkan satu contoh di posisi tertentu dalam daftar.
- `Attribute attribute(int index)`
Mengembalikan atribut.

- `Attribut attribut (java.lang.String nam)`
Mengembalikan atribut yang sesuai dengan nama yang diberikan.
- `Attribut Stats attribut Stats(int ind x)`
Menghitung ringkasan statistik pada nilai-nilai yang muncul dalam rangkaian kasus untuk atribut tertentu.
- `doubl [] attribut ToDoubl Array(int ind x)`
Mendapat nilai semua contoh dalam data ini untuk atribut tertentu.
- `boolean checkForAttribut Typ (int attTyp)`
Cek untuk atribut dari tipe yang diberikan dalam data.
- `boolean checkForStringAttribut s()`
Cek string atribut dalam data.
- `boolean checkInstanc (Instanc instanc)`
Memeriksa apakah contoh yang diberikan kompatibel dengan data ini.
- `Attribut classAttribut ()`
Mengembalikan atribut class.
- `int classInd x()`
Mengembalikan indeks atribut kelas itu.
- `void del t ()`
Menghapus semua contoh dari `s`.
- `void del t (int ind x)`
Menghapus semua contoh di posisi tertentu dari `s`.
- `void del t Attribut At (int position)`
Menghapus atribut pada posisi tertentu.
- `void del t Attribut Typ (int attTyp)`
Menghapus semua atribut dari tipe yang diberikan dalam data.
- `void del t StringAttribut s()`
Menghapus semua atribut string dalam data.
- `void del t WithMissing(Attribut att)`
Menghapus semua contoh dengan nilai-nilai yang hilang untuk atribut tertentu dari data.
- `void del t WithMissing(int attInd x)`
Menghapus semua contoh dengan nilai-nilai yang hilang untuk atribut tertentu dari data.

- `void deleteWithMissingClass()`
Menghapus semua contoh dengan nilai kelas hilang dari dataset.
- `java.util.Enumeration<Attribute> numRatAttributes()`
Mengembalikan penghitungan semua atribut.
- `java.util.Enumeration<Instance> numRatInstances()`
Mengembalikan penghitungan semua contoh dalam dataset.
- `boolean qualHeaders(Instances dataset)`
Cek jika dua header yang sama.
- `java.lang.String qualHeadersMsg(Instances dataset)`
Cek jika dua header yang sama.
- `Instance firstInstance()`
Mengembalikan contoh pertama di dataset.
- `Instance get(int index)`
Mengembalikan contoh pada posisi tertentu.
- `java.util.Random getRandomNumberGenerator(long seed)`
Mengembalikan nomor acak.
- `java.lang.String getRevision()`
Mengembalikan string revisi.
- `void insertAttributeAt(Attribute att, int position)`
Menyisipkan atribut pada posisi tertentu (0 numAttributes()) dan menaptkan semua nilai hilang.
- `Instance instance(int index)`
Mengembalikan contoh pada posisi tertentu.
- `double kthSmallestValue(Attribute att, int k)`
Mengembalikan nilai atribut k-terkecil dari atribut numerik.
- `double kthSmallestValue(int attIndex, int k)`
Mengembalikan nilai atribut k-terkecil dari atribut numerik.
- `Instance lastInstance()`
Mengembalikan contoh terakhir di dataset.
- `static void main(String[] args)`
Metode utama untuk kelas ini.

- `double meanOrMod (Attribut att)`
Mengembalikan rata (mod) untuk angka (nominal) atribut s sebagai nilai floating-point.
- `double meanOrMod (int attInd x)`
Mengembalikan rata (mod) untuk angka (nominal) atribut s sebagai nilai floating-point.
- `static Instance merge Instance s(Instance s first, Instance s s cond)`
Menggabungkan dua s t Contoh b rsama-sama
- `int numAttribut s()`
Mengembalikan jumlah atribut.
- `int numClass s()`
Mengembalikan jumlah lab l k las.
- `int numDistinctValu s(Attribut att)`
Mengembalikan jumlah nilai yang b rb da dari atribut yang dib rikan.
- `int numDistinctValu s(int attInd x)`
Mengembalikan jumlah nilai yang b rb da dari atribut yang dib rikan.
- `int numInstance s()`
Mengembalikan jumlah kasus dalam datas t.
- `void randomiz (java.util.Random random)`
Mengocok contoh di s t s hingga m r ka m m rintahkan s cara acak.
- `java.lang.String relationNam ()`
Mengembalikan nama hubungan itu.
- `Instance r mov (int ind x)`
Menghapus contoh pada posisi t rt ntu.
- `void r nam Attribut (Attribut att, java.lang.String nam)`
Mengganti nama atribut.
- `void r nam Attribut (int att, java.lang.String nam)`
Mengganti nama atribut.
- `void r nam Attribut Valu (Attribut att, java.lang.String val, java.lang.String nam)`
Mengganti nama nilai nominal (atau string) nilai atribut
- `void r nam Attribut Valu (int att, int val, java.lang.String nam)`
Mengganti nama nilai nominal (atau string) nilai atribut.

- `void replaceAttributeAt(Attribute att, int position)`
Menggantikan atribut pada posisi tertentu (`0 numAttributes()`) dengan atribut yang diberikan dan menaptakan semua nilai yang hilang.
- `Instance sample (java.util.Random random)`
Membuat data t baru dengan ukuran yang sama dengan menggunakan random sampling dengan penggantian.
- `Instance sample WithWeights(java.util.Random random)`
Membuat data t baru dengan ukuran yang sama dengan menggunakan random sampling dengan penggantian sesuai dengan contoh berat saat ini.
- `Instance sample WithWeights(java.util.Random random, boolean preserveUsingWeights)`
Membuat data t baru dengan ukuran yang sama dengan menggunakan random sampling dengan penggantian sesuai dengan contoh berat saat ini.
- `Instance sample WithWeights(java.util.Random random, boolean[] sample)`
Membuat data t baru dengan ukuran yang sama dengan menggunakan random sampling dengan penggantian sesuai dengan contoh berat saat ini.
- `Instance sample WithWeights(java.util.Random random, boolean[] sample, boolean preserveUsingWeights)`
Membuat data t baru dengan ukuran yang sama dengan menggunakan random sampling dengan penggantian sesuai dengan contoh berat saat ini.
- `Instance sample WithWeights(java.util.Random random, double[] weights)`
Membuat data t baru dengan ukuran yang sama dengan menggunakan random sampling dengan penggantian sesuai dengan vektor bobot yang diberikan.
- `Instance sample WithWeights(java.util.Random random, double[] weights, boolean[] sample)`
Membuat data t baru dengan ukuran yang sama dengan menggunakan random sampling dengan penggantian sesuai dengan vektor bobot yang diberikan.
- `Instance sample WithWeights(java.util.Random random, double[] weights, boolean[] sample, boolean preserveUsingWeights)`
Membuat data t baru dengan ukuran yang sama dengan menggunakan random sampling dengan penggantian sesuai dengan vektor bobot yang diberikan.
- `Instance set(int index, Instance instance)`
Menggantikan contoh pada posisi tertentu.
- `void setClass(Attribute att)`
Mengatur atribut class.

- `void setClassInd x(int classInd x)`
Mengatur indeks kelas `x`.
- `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 x)`
Urutkan contoh berdasarkan atribut.
- `void stableSort(Attribute att)`
Urutkan contoh berdasarkan atribut, menggunakan macam stabil.
- `void stableSort(int attIndex x)`
Urutkan contoh berdasarkan atribut, menggunakan macam stabil.
- `void stratify(int numFolds)`
Mengelompokkan satu set contoh sesuai dengan nilai-nilai kelasnya jika atribut kelas nominal (sehingga setelah cross-validasi berlapis 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 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 meringkas set contoh.

- Instance trainCV(int numFolds, int numFold)

Menciptakan pelatihan dan tes untuk satu kali lipat dari cross-validasi pada dataset.
- Instance trainCV(int numFolds, int numFold, java.util.Random random)

Menciptakan pelatihan dan tes untuk satu kali lipat dari cross-validasi pada dataset.
- double variance (Attribute att)

Menghitung varians untuk atribut tertentu.
- double variance (int attIndex)

Menghitung varians untuk atribut tertentu.
- double [] variances()

Menghitung varians untuk semua atribut dengan cara berurutan.

Attribute adalah kelas yang digunakan untuk menangani atribut.

Attribute:

- static java.lang.String ARFF_ATTRIBUTE

Kata kunci yang digunakan untuk menunjukkan awal atribut dideklarasikan ARFF.
- static java.lang.String ARFF_ATTRIBUTE_DATE

Kata kunci yang digunakan untuk menunjukkan tanggal atribut.
- static java.lang.String ARFF_ATTRIBUTE_INTEGER

Kata kunci yang digunakan untuk menunjukkan atribut bernomor.
- static java.lang.String ARFF_ATTRIBUTE_NUMERIC

Kata kunci yang digunakan untuk menunjukkan atribut bernomor.
- static java.lang.String ARFF_ATTRIBUTE_REAL

Kata kunci yang digunakan untuk menunjukkan atribut bernomor.
- static java.lang.String ARFF_ATTRIBUTE_RELATIONAL

Kata kunci yang digunakan untuk menunjukkan atribut relasional.
- static java.lang.String ARFF_ATTRIBUTE_STRING

Kata kunci yang digunakan untuk menunjukkan atribut String.
- static java.lang.String ARFF_END_SUBRELATION

Kata kunci yang digunakan untuk menunjukkan akhir dari deklarasi subrelasi.
- static int DATE

Set konstan untuk atribut dengan nilai tanggal.

- `static java.lang.String DUMMY_STRING_VAL`
Dummy p rtama nilai String atribut.
- `static int NOMINAL`
S t konstan untuk atribut nominal.
- `static int NUMERIC`
S t konstan untuk atribut num rik.
- `static int ORDERING_MODULO`
S t konstan untuk atribut ord ring modulo.
- `static int ORDERING_ORDERED`
S t konstan untuk atribut m m rintahkan.
- `static int ORDERING_SYMBOLIC`
S t konstan untuk atribut simbolik.
- `static int RELATIONAL`
S t konstan untuk atribut nilai r lasi.
- `static int STRING`
S t konstan untuk atribut d ngan nilai-nilai string.

Constructor:

- Atribut (`java.lang.String` atribut `Nam`)
Konstruktor untuk atribut num rik.
- Atribut (`java.lang.String` atribut `Nam` , `Instanc s h ad r`)
Konstruktor untuk atribut nilai r lasi.
- Atribut (`java.lang.String` atribut `Nam` , `Instanc s h ad r`, `int ind x`)
Konstruktor untuk atribut nilai r lasi d ngan ind ks t rt nt u.
- Atribut (`java.lang.String` atribut `Nam` , `Instanc s h ad r`, `Prot ct dProp rti s m tadata`)
Konstruktor untuk atribut nilai r lasi.
- Atribut (`java.lang.String` atribut `Nam` , `int ind x`)
Konstruktor untuk atribut num rik d ngan ind ks t rt nt u.
- Atribut (`java.lang.String` atribut `Nam` , `java.util.List<java.lang.String>` atribut `Valu s`)
Konstruktor untuk atribut nominal dan atribut string.
- Atribut (`java.lang.String` atribut `Nam` , `java.util.List<java.lang.String>` atribut `Valu s`, `int ind x`)
Konstruktor untuk atribut nominal dan atribut string d ngan ind ks t rt nt u.

- `Attribut (java.lang.String attribut Nam , java.util.List<java.lang.String> attribut Valu s, Prot ct dProp rti s m tadata)`

Konstruktör untuk atribut nominal dan atribut string, di mana `m tadata` dib rikan.

- `Attribut (java.lang.String attribut Nam , Prot ct dProp rti s m tadata)`

Konstruktör untuk atribut num rik, di mana `m tadata` dib rikan.

- `Attribut (java.lang.String attribut Nam , java.lang.String dat Format)`

Konstruktör untuk tanggal atribut.

- `Attribut (java.lang.String attribut Nam , java.lang.String dat Format, int ind x)`

Konstruktör untuk tanggal atribut d ngan ind ks t rt ntu.

- `Attribut (java.lang.String attribut Nam , java.lang.String dat Format, Prot ct dProp rti s m tadata)`

Konstruktör untuk atribut tanggal, di mana `m tadata` dib rikan.

Method:

- `int addR lation(Instanc s valu)`

M nambahkan r lasi pada atribut nilai r lasi.

- `int addStringValue (Attribut src, int ind x)`

M nambahkan nilai string k daftar string yang valid untuk atribut j nis string dan m ng m-balikan ind ks string.

- `int addStringValue (java.lang.String valu)`

M nambahkan nilai string k daftar string yang valid untuk atribut j nis string dan m ng m-balikan ind ks string

- `java.lang.Obj ct copy()`

M nhasilkan salinan atribut ini.

- `Attribut copy(java.lang.String n wNam)`

M nhasilkan salinan atribut ini d ngan nama baru.

- `java.util.Enum ration<java.lang.Obj ct> num rat Valu s()`

P ng mbalian p nghitungan s mua nilai atribut jika atribut nominal, string, atau hubungan-nilai, null s baliknya.

- `bool an quals(java.lang.Obj ct oth r)`

P ngujian jika dib rikan atribut sama d ngan atribut ini.

- `java.util.String qualsMsg(java.lang.Obj ct oth r)`

P ngujian jika dib rikan atribut sama d ngan atribut ini.

- `java.util.String formatDat (doubl dat)`
M ng mbalikan milid tik s suai d ngan tanggal saat ini.
- `java.util.String g tDat Format()`
M ng mbalikan pola format tanggal dalam hal atribut ini adalah tip dat , s lain itu, maka string akan kosong.
- `doubl g tLow rNum ricBound()`
P ng mbalian batas bawah dari atribut num rik.
- `Prot ct dProp rti s g tM tadata()`
M ng mbalikan prop rti s dis diakan untuk atribut ini.
- `java.lang.String g tR vision()`
M ng mbalikan string r visi.
- `doubl g tUpp rNum ricBound()`
M ng mbalikan nilai dari atribut num rik.
- `int hashCod ()`
M ng mbalikan kod hash untuk atribut ini b rdasarkan namanya.
- `bool an hasZ ropoint()`
P ng mbalian apakah atribut m miliki z ropoint.
- `int ind x()`
M ng mbalikan ind x dari atribut ini.
- `int ind xOfValu (java.lang.String valu)`
M ng mbalikan ind x dari nilai atribut t rt ntu.
- `bool an isAv ragabl ()`
P ng mbalian apakah atribut dapat dirata-ratakan b rmakna.
- `bool an isDat ()`
P ngujian jika atribut adalah j nis tanggal.
- `bool an isInRang (doubl valu)`
M n ntukan apakah suatu nilai t rl tak dalam batas-batas atribut.
- `bool an isNominal()`
M nguji apakah atribut nominal.
- `bool an isNum ric()`
P ngujian jika atribut num rik.

- `boolean isRelationValue()`
Pengujian jika atribut hubungan dihargai.
- `boolean isString()`
Pengujian jika atribut string.
- `static void main(java.lang.String[] ops)`
Metode utama yang disediakan untuk menguji kelas ini.
- `java.lang.String name()`
Mengembalikan nama atribut itu.
- `int numValues()`
Mengembalikan jumlah nilai atribut.
- `int ordinal()`
Mengembalikan posisinya atribut.
- `int parseDate(java.lang.String string)`
Mengurai string yang diberikan sebagai tanggal, sesuai format saat ini dan mengembalikan sesuai dengan jumlah milidetik.
- `Instance relation()`
Mengembalikan informasi header untuk atribut nilai relasi, null jika atribut tidak memiliki hubungan.
- `Instance relation(int valueIndex)`
Mengembalikan nilai atribut nilai relasi.
- `void setStringValue(java.lang.String value)`
Mengosongkan nilai dan mengatur merka mengandung hanya nilai yang diberikan.
- `void setWeight(double value)`
Mengatur berat atribut baru.
- `java.lang.String toString()`
Pengembalian deskripsi atribut ini dalam format ARFF.
- `int type()`
Mengembalikan jenis atribut sebagai integer.
- `static java.lang.String typeToString(Attribut att)`
Mengembalikan representasi string dari jenis atribut.
- `static java.lang.String typeToString(int type)`
Mengembalikan representasi string dari jenis atribut.

- static java.lang.String typ ToStringShort(Attribut att)
Mengembalikan representasi string jenis atribut.
- static java.lang.String typ ToStringShort(int typ)
Mengembalikan representasi string jenis atribut.
- java.lang.String valu (int valInd x)
Mengembalikan nilai atribut nominal atau tali.
- double weight()
Mengembalikan bobot badan atribut itu

ID3 adalah kelas yang digunakan untuk membangun *decision tree* yang berbasis pada algoritma ID3, hanya dapat menerima input dengan atribut nominal. *Constructor*:

- ID3()

Method:

- void buildClassifier(Instance data)
Membangun ID3 pohon keputusan classifier.
- double classifyInstance (Instance instance)
Mengklasifikasikan tes data yang diberikan dengan menggunakan pohon keputusan.
- double [] distributionForInstance (Instance instance)
Menghitung distribusi kelas instance menggunakan pohon keputusan.
- Capabilities getCapabilities()
Mengembalikan detail classifier.
- java.lang.String getRevision()
Mengembalikan String revisi.
- TechnicalInformation getTechnicalInformation()
Mengembalikan sebuah instance dari objek TechnicalInformation, yang berisi informasi rinci tentang latar belakang teknis kelas ini.
- java.lang.String globalInfo()
Mengembalikan string yang menjelaskan classifier.
- static void main(java.lang.String[] args)
Metode utama untuk kelas ini.
- java.lang.String toSource (java.lang.String className)
Mengembalikan string yang menggambarkan classifier.
- java.lang.String toString()
Menetak pohon keputusan menggunakan metode toString.

J48 adalah kelas yang digunakan untuk membuat *decision tree* c4.5.

Constructor:

- ID3()

Method:

- java.lang.String binarySplitsTipText()
Mengembalikan teks tip untuk properti ini.
- void buildClassifier(Instance instance)
Menghasilkan classifier.
- double classifyInstance(Instance instance)
Mengklasifikasikan data.
- java.lang.String confidenceFactorTipText()
Mengembalikan teks tip untuk properti ini.
- double[] distributionForInstance(Instance instance)
Mengembalikan probabilitas kelas untuk sebuah data.
- java.util.EnumRating numRatingMeasure()
Mengembalikan perhitungan ukuran.
- boolean getBinarySplits()
Dapatkan nilai binarySplits.
- Capabilities getCapabilities()
Mengembalikan capabilities dari kelas ini.
- float getConfidenceFactor()
Mengembalikan nilai *confident*.
- double getMeasure(java.lang.String additionalMeasureName)
Mengembalikan nilai bobot sesuai nama.
- int getMinNumObj()
Dapatkan nilai minNumObj.
- int getNumFolds()
Dapatkan nilai numFolds.
- java.lang.String getOptions()
Mendapatkan pengaturan saat ini.

- `boolean getReducedErrorPruning()`
Dapatkan nilai `reducedErrorPruning`.
- `java.lang.String getRevision()`
Mengembalikan string revisi.
- `boolean getSaveInstanceData()`
Periksa apakah contoh data disimpan.
- `int getSeed()`
Dapatkan nilai `seed`.
- `boolean getSubtreeRaising()`
Dapatkan nilai `subtreeRaising`.
- `TechnicalInformation getTechnicalInformation()`
Mengembalikan sebuah instance dari objek `TechnicalInformation`, yang berisi informasi rinci tentang latar belakang teknis kelas ini.
- `boolean getUnpruned()`
mengetahui apakah dilakukan *tree pruning*.
- `boolean getUseLaplace()`
Dapatkan nilai `useLaplace`.
- `java.lang.String globalInfo()`
Mengembalikan string yang menjelaskan classifer.
- `java.lang.String graph()`
Mengembalikan Grafik yang menggambarkan pohon.
- `int graphType()`
Mengembalikan jenis grafik classifer.
- `static void main(java.lang.String[] argv)`
Metode utama untuk menguji kelas ini.
- `double getMaxNumLeaves()`
Mengembalikan jumlah daun.
- `double getMaxNumRules()`
Mengembalikan jumlah aturan.
- `double getMaxTreeSize()`
Mengembalikan ukuran pohon.

- `java.lang.String minNumObjTipT xt()`
M ng mbalikan t ks tip untuk prop rti ini.
- `java.lang.String numFoldsTipT xt()`
M ng mbalikan t ks tip untuk prop rti ini
- `java.lang.String pr fix()`
P ng mbalian pohon dalam rangka awalan.
- `java.lang.String r duc dErrorPruningTipT xt()`
M ng mbalikan t ks tip untuk prop rti ini
- `java.lang.String sav Instanc DataTipT xt()`
M ng mbalikan t ks tip untuk prop rti ini.
- `java.lang.String s dTipT xt()`
M ng mbalikan t ks tip untuk prop rti ini
- `void s tBinarySplits(boolean v)`
M ngatur nilai `binarySplits`.
- `void s tConfid nc Factor(float v)`
M ngatur nilai *confident*.
- `void s tMinNumObj(int v)`
M ngatur nilai `minNumObj`.
- `void s tNumFolds(int v)`
M ngatur nilai `numFolds`.
- `void s tOptions(java.lang.String[] options)`
M ngurai daftar yang dib rikan pilihan.
- `void s tR duc dErrorPruning(boolean v)`
M ngatur nilai `r duc dErrorPruning`.
- `void s tSav Instanc Data(boolean v)`
M ngatur apakah contoh data yang akan disimpan.
- `void s tS d(int n wS d)`
M ngatur nilai `s d`.
- `void s tSubtr Raising(boolean v)`
M ngatur nilai `subtr Raising`.

- void setUnpruned(boolean v)
Mengatur nilai *pruning*.
- void setUsLaplac(boolean n_wusLaplac)
Mengatur nilai usLaplac.
- java.lang.String subtraisingTipText()
Mengembalikan teks tip untuk properti ini.
- java.lang.String toString()
Pengembalian skripsi classifikasi.
- java.lang.String unprunedTipText()
Mengembalikan teks tip untuk properti ini.
- java.lang.String usLaplacTipText()
Mengembalikan teks tip untuk properti ini.

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 menjelaskan 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 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 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 kebanyakan, atribut ini berisi nilai latitud dari lokasi kebanyakan yang dipilih oleh user
- Longitud kebanyakan, atribut ini berisi nilai longitud dari lokasi kebanyakan 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 Kebanyakan
- Longitud Kebanyakan
- 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

Agar dapat diprolh *decision tree* m ng nai lokasi k b rangkaian dan tujuan dari us r KIRI, maka atribut k las yang akan digunakan adalah nilai latitud dan longitud dari lokasi k b rangkaian dan tujuan. Kar na atribut k las ada mpat, maka akan dilakukan p ny d rhanaan dari k mpat atribut untuk m ningkatkan akurasi s rta tingkat fisi n pros s *data mining*.

Nilai *latitude* s rta *longitude* dari data lokasi k b rangkaian dan tujuan akan diubah m njadi nilai yang m nunjukkan apakah da rah lokasi k b rangkaian dan tujuan t rs but m nunjukkan p rjalanan k luar dari Bandung atau tidak. Hal ini dilakukan agar dip rol h data p rbandingan p rg rakan p nduduk, apakah m r ka l bih banyak yang k luar dari Bandung atau 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 . K mudian dibagi m njadi s puluh da rah yang m miliki p rb daan radius s b sar 1 km, s hingga diam t r untuk da rah p rtama adalah 2 km, diam t r untuk da rah k dua adalah 4 km, dan s t rusnya, untuk da rah t rakhir (yaitu da rah 10) akan m miliki diam t r 20 km.

Suatu lokasi atau titik latitud longitud dapat dik tahui b rada pada da rah yang mana d ngan cara m nghitung jarak titik t rs but d ngan titik pusat yang sudah dit ntukan (yaitu -6.92036,107.60500) d ngan m nggunakan rumus Hav rsin . Jika jarak yang dip rol h l bih k cil sama d ngan 1 km, maka b rada di da rah p rtama, s dangkan jika jarak yang dip rol h l bih k cil sama d ngan 2 km dan l bih b sar dari 1 km, maka b rada di da rah k dua, dan s t rusnya, dan untuk da rah t rakhir (yaitu da rah 10) titik akan m miliki jarak l bih k cil sama d ngan 10 km dan l bih b sar dari 9 km d ngan titik pusat. Jika suatu titik m miliki jarak t rhadap titik pusat l bih dari 10 km, maka akan m njadi da rah luar Bandung.

S t lah lokasi k b rangkaian dan lokasi tujuan dit ntukan da rahnya, dapat dit ntukan apakah us r t rs but m nuju pusat Bandung atau tidak. Jika da rah dari lokasi k b rangkaian l bih b sar daripada da rah lokasi tujuan, maka us r t rs but m nuju pusat Bandung. K mudian, jika da rah dari lokasi k b rangkaian l bih k cil daripada da rah lokasi tujuan, maka us r t rs but tidak m nuju pusat Bandung. S dangkan, jika lokasi k b rangkaian dan lokasi tujuan b rada di da rah yang sama, maka us r t rs but maka us r t rs but b rg rak di da rah yang sama.

Dengan adanya perhitungan jarak dan penentuan daerah Bandung, nilai latitud dan longitud dari lokasi kebangkatan dan tujuan dapat dibuang dan diganti oleh atribut menujuBandung 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 kebangkatan dan tujuan pada daerah yang sama. Contoh hasil data setelah dilakukan *transformation* terhadap latitud dan longitud terdapat pada tabel 3.3.

Tab 1 3.3: Contoh hasil data transformasi latitud longitud

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 kebangkatan 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 csv, maka pada penelitian ini, tidak akan menggunakan sistem database.

Ketika tombol proses ditekan, maka data tersebut akan diproses. Proses yang pertama yang akan dilakukan adalah melakukan *load* data dari file. data csv akan dibaca dengan menggunakan CSVReader hingga semua hasil datanya sudah terpisah sesuai dengan atribut. Kemudian dilakukan filter data dan hanya action dengan nilai FINDROUTE yang akan diambil. Setelah data didapat, akan dilakukan proses *transform* untuk setiap baris yang ada. Proses *transform* tersebut memiliki tahap sebagai berikut:

1. Mengambil string data input array kedua (yaitu atribut tanggal) kemudian memisah nilai tersebut dengan spasi sebagai tanda pemisah, maka akan terdapat dua nilai, yaitu tanggal dan jam.
2. Pada nilai tanggal, dilakukan pemecahan nilai string dengan garis miring sebagai tanda pemisah, maka akan diperoleh tiga nilai yaitu bulan, tanggal, dan tahun.
3. Pada nilai jam, dilakukan pemecahan nilai string dengan titik dua sebagai tanda pemisah, maka akan diperoleh dua nilai yaitu jam dan menit.
4. Mengambil string data input array keempat (yaitu atribut *additionalData*), dilakukan pemecahan nilai string dengan garis miring sebagai tanda pemisah, maka akan diperoleh tiga nilai yaitu lokasi awal, lokasi tujuan, dan banyak jalur.
5. 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*.
6. Mengubah waktu dari UTC menjadi GMT+8.
7. Mencari hari dengan memanfaatkan nilai tanggal, bulan, dan tahun serta kelas *calendar*.
8. 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 menujuBandung.
9. Menggabungkan nilai-nilai tersebut ke dalam satu array, yaitu array dengan tipe *int* (dengan nilai tanggal, bulan, tahun, jam, menit dan menujuBandung).

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 tahapan sebagai berikut

1. Program akan memuat data dan melakukan *processing data*
2. Program akan menjalankan algoritma untuk membuat *decision tree* yang terdapat pada ??
3. Program akan menampilkan *decision tree*

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

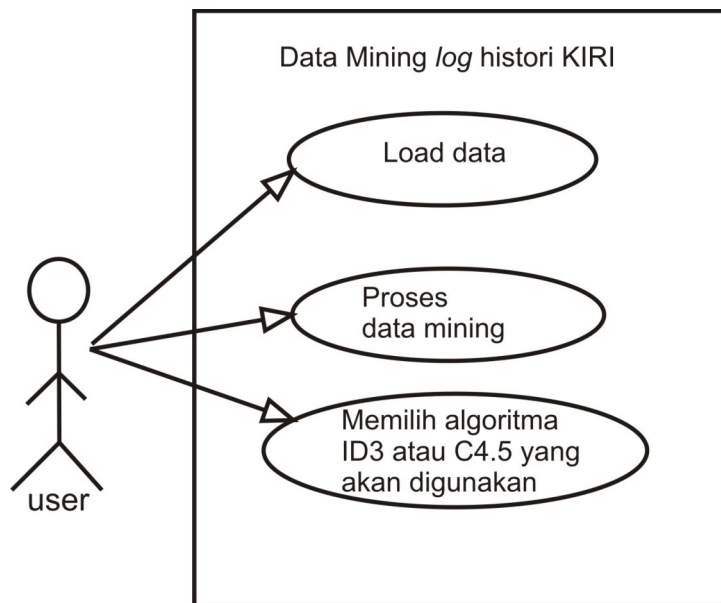
1. Program akan menghitung nilai entropi *class*
2. Program akan menghitung nilai entropi dan mendapatkan nilai *gain info* untuk setiap atribut pada *attribute_list*
3. Jika user memilih untuk menggunakan algoritma C4.5, maka program akan menghitung *splitInfo* dan menghitung *gainRatio*
4. Program 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 dimasukkan. 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.

Gambar 3.2: Diagram *Use Case* P perangkat Lunak *Data Mining Log Histori KIRI*Tab 1 3.5: Sk nario M lakukan *load Data*

Nama	Load data
Aktor	<i>User</i>
D skripsi	M masukan alamat data yang akan dijadikan s bagai 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 xtbox
Eks spi	Data tidak dit mukan

Tab 1 3.6: Sk nario M lakukan *Data Mining*

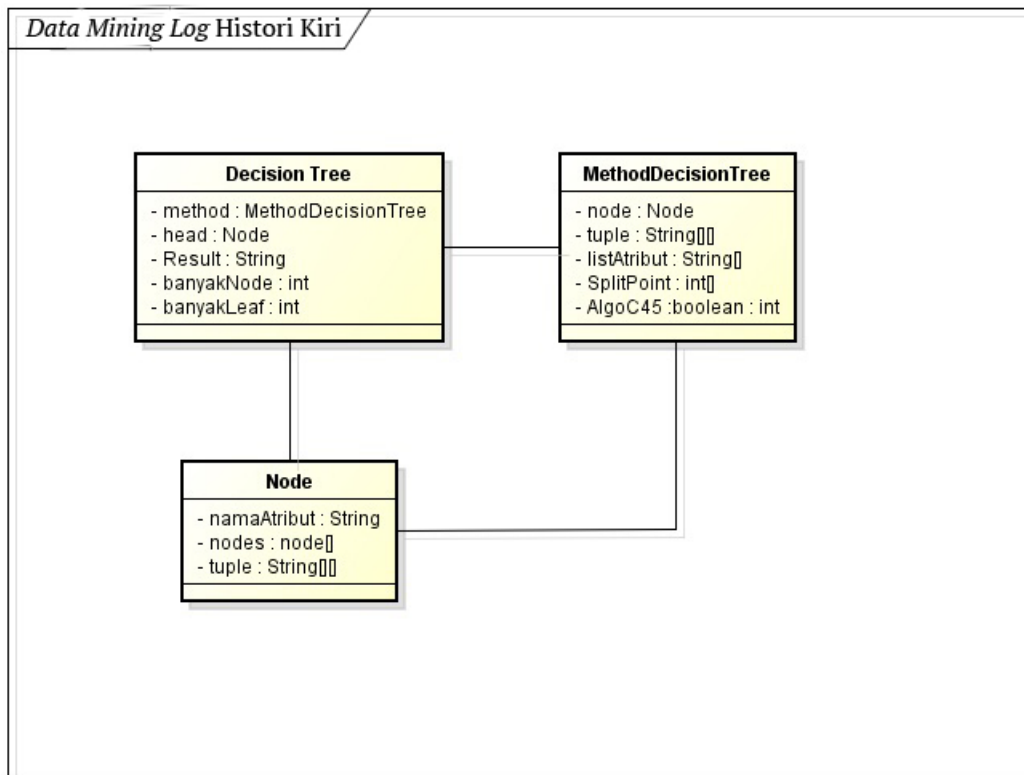
Nama	Pros s <i>Data Mining</i>
Aktor	<i>User</i>
D skripsi	M n kan tombol pros s pada <i>interface</i>
Kondisi awal	<i>Textbox</i> b lum t risi
Kondisi akhir	<i>Textbox</i> sudah t risi d ngan hasil <i>data mining</i>
Sk nario utama	<i>User</i> m n kan tombol pros s
Eks spi	Data tidak dit mukan atau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

Tab 1 3.7: 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*

Pembuatan diagram *class* untuk memenuhi semua tujuan dari diagram *use case* dan skenario terdapat pada gambar 3.3.



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

Berikut deskripsi kelas diagram *class*:

- *DecisionTree*, merupakan kelas utama yang akan menjalankan algoritma pembuatan pohon
- *MethodDecisionTree*, merupakan kelas yang menjalankan algoritma pemilihan atribut untuk pembuatan pohon (pada penelitian ini, algoritma yang dapat dipilih adalah ID3 dan C4.5)
- *Node*, merupakan kelas yang digunakan sebagai struktur data untuk *decision tree*

DAFTAR REFERENSI

- [1] Data Mining *Data Mining Concepts and Techniques* 2006 : Jiawei Han and Micheline Kamber
- [2] <http://www.kaisources.org.net/doc/stable/>

LAMPIRAN A

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