

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	v
DAFTAR TABEL	vi
1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Perumusan Masalah	2
1.3 Tujuan	2
1.4 Batasan Masalah	2
1.5 Metod Penelitian	2
1.6 Sistematis Pembahasan	2
2 LANDASAN TEORI	5
2.1 Data Mining	5
2.1.1 Data Cleaning	6
2.1.2 Data Integration	7
2.1.3 Data Selection	7
2.1.4 Data Transformation	8
2.1.5 Data Mining	9
2.1.6 Pattern Evaluation	18
2.1.7 Knowledge Presentation	18
2.2 Log Histori KIRI	18
2.3 Haversine Formula	20
2.4 Waktu	21
2.5 Graphviz	41
3 ANALISA	43
3.1 Analisis Data	43
3.1.1 Data Cleaning	43
3.1.2 Data Integration	43
3.1.3 Data Selection	43
3.1.4 Data Transformation	44
3.2 Analisis Pengaruh Lunak	48
3.2.1 Diagram Use Case Pengaruh Lunak Data Mining Log Histori KIRI	50
3.2.2 Diagram Kelas Pengaruh Lunak Data Mining Log Histori KIRI	51
4 PERANCANGAN PERANGKAT LUNAK	53
4.1 Perancangan Pengaruh Lunak	53
4.1.1 Perancangan Klasifikasi	53
4.1.2 Skewness Diagram	60
4.1.3 Perancangan Dua Sain Antar Muka	62

5 IMPLEMENTASI PROGRAM DAN PENGUJIAN	63
5.1 Lingkungan Pengembangan	63
5.2 Hasil Tampilan Antarmuka	63
5.3 Pengujian Aplikasi <i>Data Mining</i>	66
5.3.1 Pengujian Fungsional	66
5.3.2 Pengujian Ekspresional	72
5.3.3 Analisis Hasil Uji	75
6 PENUTUP	77
6.1 Kesimpulan	77
6.2 Saran	77
DAFTAR REFERENSI	79
A 100 DATA PERTAMA DARI log HISTORI KIRI	81

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	J nis-j nis <i>split point</i>	13
2.5	Hasil pohon faktor pada atribut <i>age</i> dari tabl 2.1	16
2.6	<i>Decision Tree Pruned</i>	18
2.7	Hasil output Graphviz	41
3.1	<i>Classification</i> pada da rah Bandung	47
3.2	Diagram <i>Use Case P</i> rangkat Lunak <i>Data Mining Log Histori KIRI</i>	50
3.3	Diagram <i>Class P</i> rangkat Lunak <i>Data Mining Log Histori KIRI</i>	52
4.1	Diagram <i>Class P</i> rangkat Lunak <i>Data Mining Log Histori KIRI</i>	59
4.2	Diagram <i>Class P</i> rangkat Lunak <i>Data Mining Log Histori KIRI</i>	61
4.3	Mock Up Form <i>P</i> rtama	62
4.4	Mock Up Form <i>K</i> dua	62
5.1	Tampilan Form Awal Aplikasi <i>Data Mining</i>	64
5.2	Tampilan Fil S 1 ctor untuk M milih Fil CSV	64
5.3	Tampilan Form s t lah M milih Fil CSV	65
5.4	Tampilan Form P milihan M tod P mbuatan <i>Decision Tree</i>	65
5.5	Tampilan Form M nampilkan Hasil <i>Data Mining</i>	66
5.6	P ngujian M ngambil Data CSV	68
5.7	P ngujian <i>Data Selection</i> untuk M ngambil Data d ngan <i>Action FINDROUTE</i>	68
5.8	P ngujian <i>Preprocessing Data</i>	69
5.9	P ngujian <i>Preprocessing Data</i> untuk Klasifikasi Data	70
5.10	P ngujian P mbuatan <i>Decision Tree ID3</i>	71
5.11	P ngujian P mbuatan <i>Decision Tree J48</i>	71
5.12	P ngujian Hasil Visualisasi d ngan M nggunakan Bahasa DOT	71
5.13	P rcobaan <i>Data Mining</i> untuk M lakukan Data Cl aning	72
5.14	P rcobaan <i>Data Mining</i> d ngan M nggunakan M tod ID3 pada <i>Log Histori KIRI</i> pada Bulan 2 Tahun 2014	73
5.15	P rcobaan <i>Data Mining</i> d ngan M nggunakan M tod J48 pada <i>Log Histori KIRI</i> pada Bulan 2 Tahun 2014	73
5.16	Hasil dari SDForExtractData	74

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>	44
3.2	Contoh hasil data transformasi	46
3.3	Contoh hasil data transformasi latitud longitud	48
3.5	Sk nario M lakukan <i>load Data</i>	51
3.6	Sk nario M lakukan <i>Data Mining</i>	51
3.7	Sk nario M milih Algoritma yang Akan Digunakan	51
5.1	Data untuk <i>Test Case</i> Aplikasi <i>Data Mining</i>	67
5.2	Hasil P n tuan ar a dan Klasifikasi	70

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Dengan kemajuan teknologi informasi saat ini, kebutuhan akan informasi yang akurat dibutuhkan dalam kehidupan sehari-hari, namun informasi tersebut belum selalu diolah agar dapat disajikan dengan baik. Jika dilihat lebih rinci, jika data tersebut belum diolah lebih lanjut, dapat menghasilkan suatu yang lebih baik. Salah satu cara mengolah data tersebut adalah dengan menggunakan teknik *data mining*. Dengan menggunakan teknik *data mining* akan menjadi mudah untuk menganalisa masalah, pengambilan kesimpulan, bahkan menjadi mudah konsumen dalam membuat 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 menjadi rujukan 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
- vendorID, mengindikasikan sumber dari pencarian ini
- timestamp, waktu ketika pengguna KIRI mencari ruta angkot
- type, tipe fungsi yang digunakan
- additionalInfo, mencatat koordinat awal, koordinat akhir, dan banyak rute yang dituju pada pencarian ini

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

1.2 Perumusan Masalah

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

- Bagaimana cara mengolah data yang dipilih dari data log histori KIRI agar pola yang dihasilkan menjadi menarik dan bermakna?
- Bagaimana membuat program lunak untuk melakukan *data mining* dan menghasilkan pola yang menarik dan bermakna pada data log histori?

1.3 Tujuan

Penelitian ini bertujuan untuk

- Mencari pola dan informasi yang menarik dari *log histori* KIRI
- Program 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 pengetahuan *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 dituliskan 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:

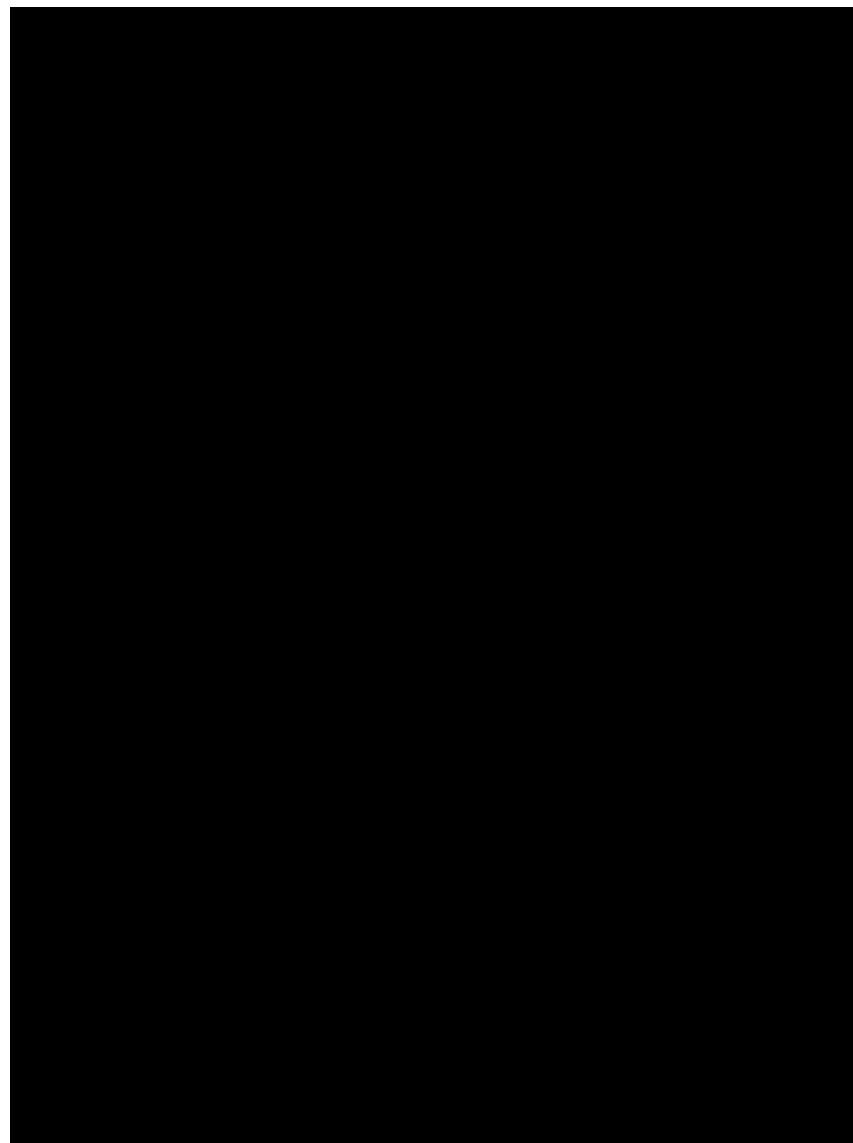
- BAB 1: Pendahuluan, berisi latar belakang dari penelitian ini, rumusan masalah yang timbul, tujuan yang ingin dicapai, ruang lingkup atau batasan masalah dari penelitian ini, serta tentang penelitian yang akan digunakan dan sistem matematika pembahasan dari penelitian ini
- BAB 2: Landasan Teori, berisi 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 mendesain aplikasi *data mining log histori KIRI* berikut diagram *use case*, skenario, dan diagram klas
- BAB 4: Berisi perancangan dari aplikasi *data mining log histori KIRI* yang akan dibangun
- BAB 5: Berisi implementasi dan pengujian dari aplikasi *data mining*
- BAB 6: Berisi kesimpulan dan saran dari penelitian *data mining log histori KIRI*

BAB 2

LANDASAN TEORI

2.1 *Data Mining*

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



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

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

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

Tahap pertama hingga keempat merupakan bagian dari *data preprocessing*, dimana data-data disiapkan untuk dilakukan penggalian data. Tahap *data mining* merupakan tahap dimana melakukan penggalian data. Tahap kelima merupakan tahap pencarian pola yang merupakan bentuk penasikan *knowledge*. Sedangkan tahap terakhir merupakan visualisasi dan representasi dari *knowledge* yang sudah dipilih 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 dipilih dari database tidak dapat 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

Tab 2.1: tabel mengandung *missing value* dan *noisy*

IdPenjualan	NamaBarang	Customer	Harga	BanyakBarang
1	Mouse	Elvin	45000	2
2	Keyboard	Allaria	-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

- Mengbuang tuple yang mengandung nilai yang hilang
- Mengisi nilai yang hilang dengan cara 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 bersifat 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 distribusi samaan atribut untuk memprediksi suatu nilai
- *Clustering*, merupakan metode pengelompokan dimana dituliskan 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 agar hasilnya sesuai dengan kondisi tertentu (contoh, penjumlahan total penjualan per tahun dari 2005-2010). Sedangkan flat data merupakan data yang disimpan dengan cara apapun untuk merepresentasikan database model pada sebuah data baik berupa plain text file maupun binary file.

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

2.1.3 Data Selection

Proses dimana data-data yang relevan dengan analisis akan diambil dari database dan data yang tidak relevan akan dibuang. Sebagai contoh kasus, jika akan dilakukan analisa mengenai nilai mahasiswa 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 JnisK lamen, dan Alamat karena tidak terlalu berhubungan 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 sembilannya
- *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 konsolidasi. Contoh studi kasus, nilai pada atribut umur dapat dikompakkan menjadi muda, dewasa, tua.

Normalization

Atribut dapat dinormalisasi dengan menggunakan skala pada nilainya sehingga nilai tersebut menjadi suatu rang yang lebih spesifik dan berkisar dari 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 dalam skala tertentu. Dengan menggunakan rumus

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

Contoh kasus, misalkan nilai minimum dan maksimum dari suatu pendapatan adalah 12.000 dan 98.000, akan diubah menjadi berdasarkan skala 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 73.600, 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 arah 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 kategori tertentu ("baik", "cukup", dan "buruk") dalam sistem berdasarkan sikap orang siswa atau "mini bus", "bus", atau "s" dan "l" dalam kategori tip mobil. Kategori tersebut dapat diterjemahkan dengan menggunakan nilai diskrit. Nilai diskrit merupakan nilai yang terpisah dan berbeda, seperti 1 atau 5. Kategori yang diterjemahkan oleh nilai diskrit maka akan menjadi nilai yang terurut dan

tidak memiliki arti, seperti 1,2,3 untuk merepresentasikan kata gori 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 berlanjut. Contoh studi kasus untuk prediksi model dan prediktor adalah seorang marketing ingin memprediksi banyak konsumen yang akan belanja di sebuah toko dalam waktu satu bulan. Model dan term but disebut *predictor*. *Regression Analysis*, merupakan metode 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 sebagaimana pada ilustrasi di gambar 2.2. *Learning step* merupakan langkah pembelajaran, dimana algoritma klasifikasi membangun *classification rules* (yang berisi syarat atau aturan sebagaimana nilai masuk ke dalam kata gori tersebut) dengan cara menganalisis *training set* yang merupakan *database tuple*. Karena pembuatan *classification rules* menggunakan *training set*, yang dilakukan juga sebagaimana *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 merupakan prinsipalitas 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 proses latihan pohon keputusan dari tampilan latihan ke dasar pokok. Terdapat beberapa teknik untuk membuat *decision tree* dua diantaranya adalah ID3 dan C4.5. ID3 merupakan teknik pembuatan *decision tree* dengan menggunakan *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 pengambilan nilai *gain info*. Kedua teknik tersebut menggunakan pendekatan *greedy* yang merupakan *decision tree* yang dibangun secara *top-down recursive divide and conquer*. Algoritma yang dipilihkan secara umum sama, hanya berbeda pada *attribute_selection_method*. Berikut algoritma untuk membuat pohon keputusan dari suatu tampilan latihan.

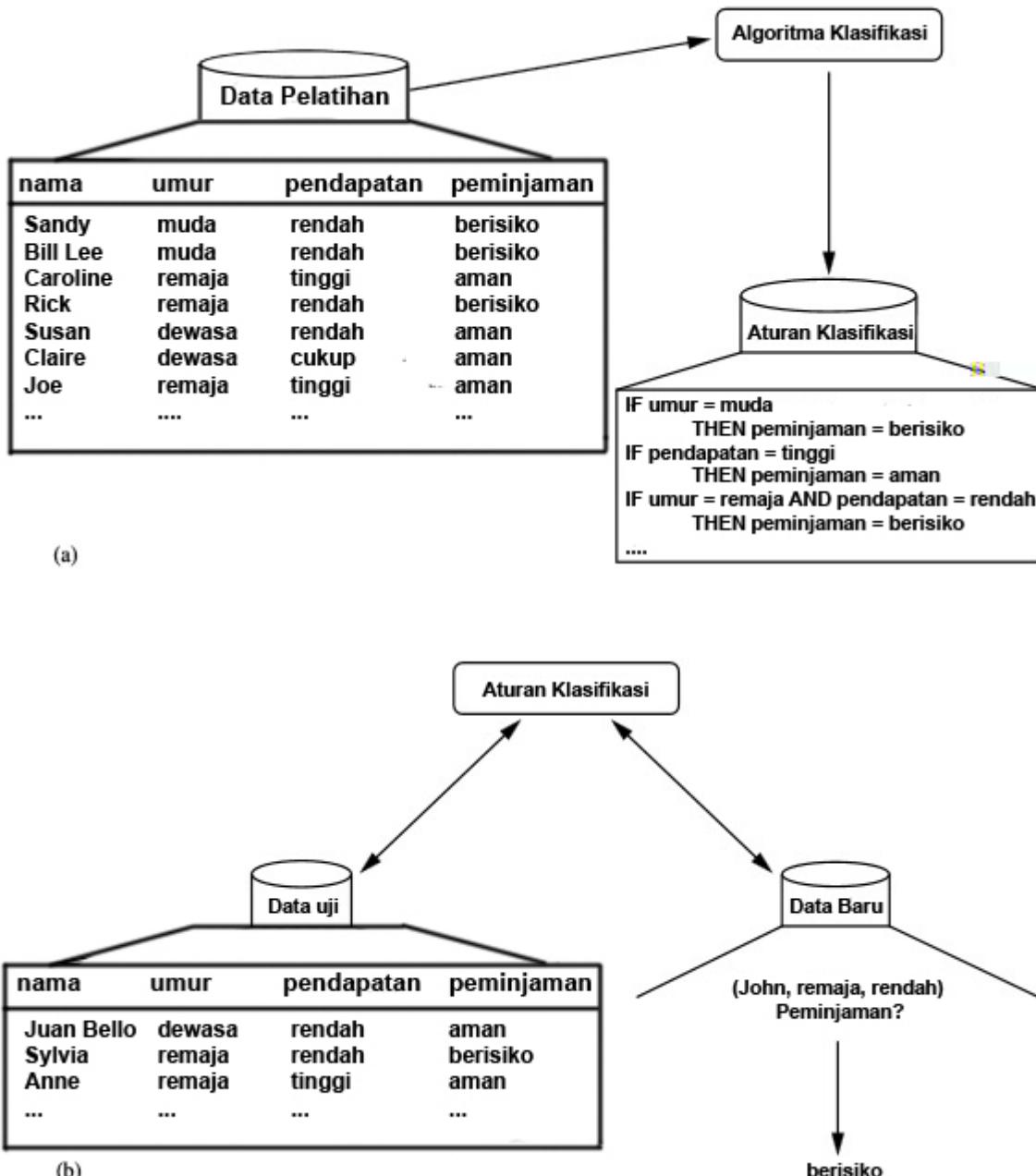
Require: Partisi data, D, merupakan set data pada latihan dan kelas lab 1

Require: *attribute_list*, merupakan set dari atribut kandidat

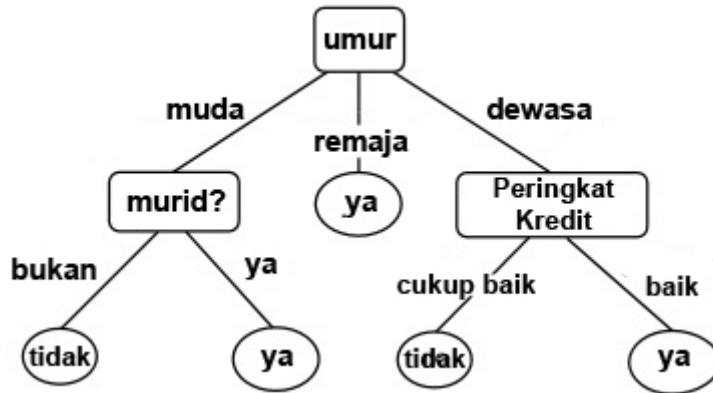
Require: *Attribute_selection_method*, proses untuk menentukan *splitting criterion*. Pada input ini, terdapat juga data *splitting_attribute* dan mungkin salah satu dari *split point* atau *splitting subset*

Ensure: Pohon keputusan

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



Gambar 2.2: Tahap data classification, [1]

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

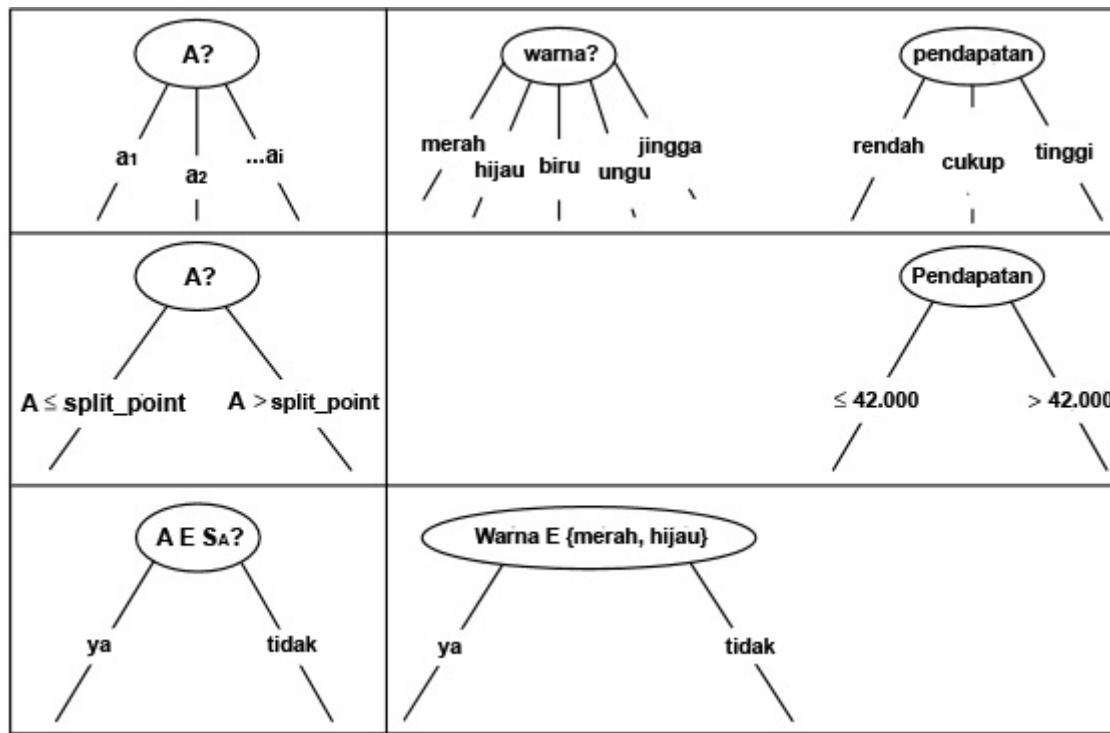
```

3:   return N s bagai nod daun d ngan lab l k las C;
4: end if
5: if attribut _list tidak ada nilai atau kosong then
6:   return N s bagai nod daun d ngan lab l k las yang t rpaling banyak pada D; {majority
      voting}
7: end if
8: m manggil m thod Attribut _s l ction_m thod(D, attribut _list) untuk m ncari nilai t rbaik
      splitting_crit rion;
9: m namakan nod N d ngan splitting_crit rion;
10: if splitting_attribut m rupakan nilai discr t and multiway splits diizinkan then
11:   attribut _list ← attribut _list - splitting_attribut ; {m nghapus splitting_attribut }
12: end if
13: for all hasil j dari splitting_crit rion do
14:   Dj m rupakan 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, attribut _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 tahu 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]

misalkan nilai terbaik untuk membagi nilai atribut pada tuple k dalam klas masing-masing. (baris 8)

Node N akan diisi dengan hasil dari *splitting criterion* (baris 9). Kriteria kritis tress but 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 bentuk nilai iya atau tidak dari "apakah A anggota S_a ", dimana S_a merupakan subkelas 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 berlaku ketika salah satu dari kondisi tersebut benar, yaitu

1. Semua tuple pada partisi D_j 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 mengkonversi node N menjadi *leaf* dan dibelakangi oleh klas yang terbanyak pada D.
3. Sudah tidak ada tuple yang dapat dibelakangi 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.

Attribute Selection Measure

Attribute Selection Measure merupakan suatu hierarki untuk pemilihan *splitting criterion* yang terbaik yang memisahkan partisi data (D), tuple pada latihan klasifikasi dalam klasifikasi masing-masing. *Attribute Selection Measure* menyediakan 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, sedangkan latihan dari *class-labeled tuple*. Jika klasifikasi klasifikasi atribut memiliki nilai yang berbeda yang mendefinisikan klasifikasi yang berbeda, C_i (for $i=1,\dots,m$). $C_{i,d}$ menjadi klasifikasi 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* (keberadaan 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 klasifikasi pada tuple D. Untuk mengetahui atribut mana yang paling baik untuk dijadikan *splitting attribute*, adalah dengan cara menghitung nilai *entropy* dari suatu atribut k median disesuaikan dengan nilai *entropy* dari D. Jika pada tuple D, memiliki atribut A dengan 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 memiliki informasi, semakin besar nilai $Info_A(D)$, semakin tinggi pula tingkat *pure* dari suatu partisi.

Setelah 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	pndapatkan	siswa	r_siko_kredit	Class: m_mb_li_komputer
1	muda	tinggi	tidak	cukup	tidak
2	muda	tinggi	tidak	baik	tidak
3	r_maja	tinggi	tidak	cukup	ya
4	d_wasa	cukup	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	cukup	tidak	cukup	tidak
9	muda	r_ndah	ya	cukup	ya
10	d_wasa	cukup	ya	cukup	ya
11	muda	cukup	ya	baik	ya
12	r_maja	cukup	tidak	baik	ya
13	r_maja	tinggi	ya	cukup	ya
14	d_wasa	cukup	tidak	baik	tidak

Pada tabel 2.2, terdapat *training set*, D. Atribut kelas lab 1 merupakan dua nilai yang berbeda yaitu ya dan tidak, maka dari itu, nilai m = 2. C₁ diisi dengan kelas lab 1 ber nilai ya, sedangkan C₂ diisi dengan kelas lab 1 ber nilai tidak. Terdapat sembilan tuple atribut kelas lab 1 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

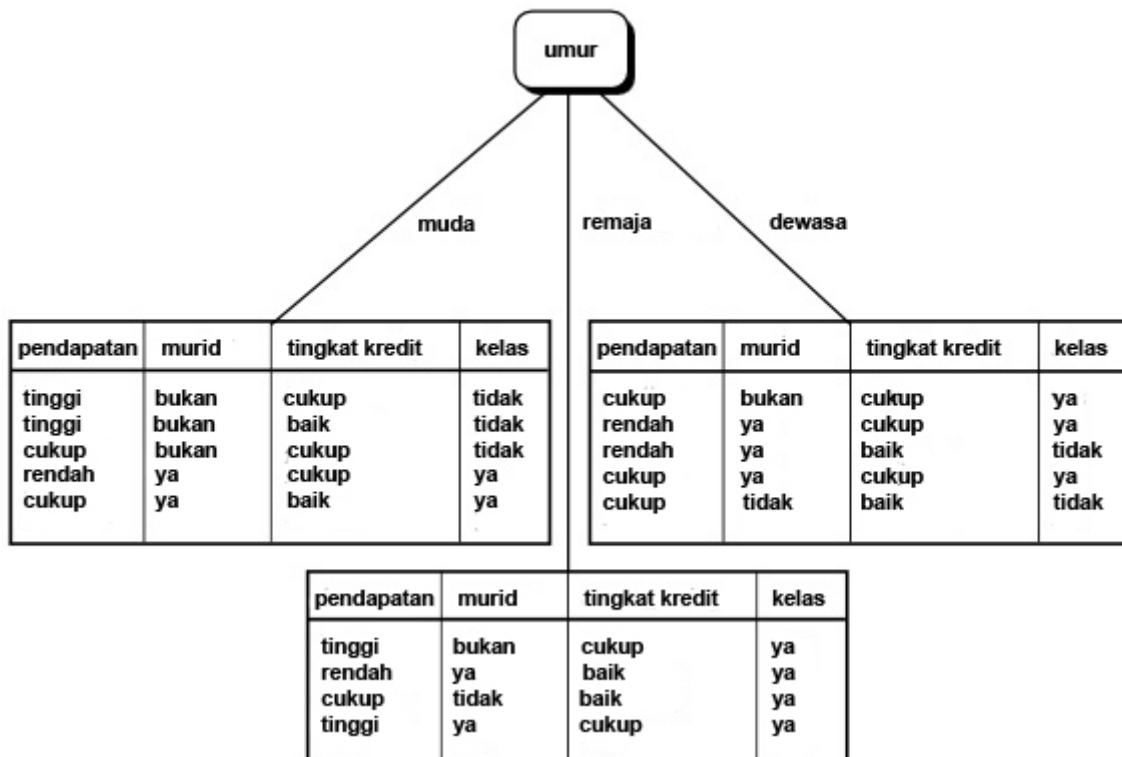
$$Info(D) = -\frac{9}{14} \log_2(\frac{9}{14}) - \frac{5}{14} \log_2(\frac{5}{14}) = 0.940 bits$$

Setelah diperoleh 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 r_maja, terdapat empat tuple dengan kelas ya dan nol tuple dengan kelas tidak. Pada kategori d_wasa, terdapat tiga tuple dengan kelas ya dan dua tuple dengan kelas tidak. Perhitungan nilai *entropy* atribut umur terhadap D sebagai berikut

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

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

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



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

Dengan melakukan hal yang sama, dapat diperoleh nilai *gain* untuk atribut pendapatan adalah 0.029 bits, untuk nilai *gain*(siswa) adalah 0.151 bits, dan *gain*(siswa_kredit) = 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 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 bersifat lahan dapat diambil nilai tengahnya untuk dijadikan *split-point*. Jika terdapat v nilai yang bersifat da dari A, maka akan terdapat v-1 kemungkinan *split point*. Kemudian nilai *split point* akan dijadikan sebagai nilai pembagi, 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 di suatu tabel yang memiliki nilai unik, maka akan terdapat banyak sekali cabang. Namun sebenarnya tiap 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 ber nilai maksimum namun tidak akan berguna untuk *classification* [1]. Selain itu, ID3 dapat menghasilkan *decision tree* yang mungkin dikatakan sebagai berlebihan (*overestimated*) atau dapat juga *overfitting*. Hal ini dikarenakan pohon yang dihasilkan terlalu dalam hingga data input memiliki hasil prediksi yang pasti.

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

ction measure untuk m lebih atribut. K mudian, C4.5 m lakukan *tree pruning* untuk m menghindari *overfitting*.

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

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

Dimana $|D|$ m merupakan banyak data dan $|D_j|$ m merupakan banyak data suatu nilai pada atribut. Setelah m mendapatkan nilai *split info* dari suatu atribut, dapat diperoleh nilai *gain ratio* dengan rumus sebagai berikut:

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

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

Contoh studi kasus, akan dilakukan perhitungan *gain ratio* dengan m menggunakan training set pada tabel 2.2. Dapat dilihat pada atribut pendapatan m miliki tiga partisi yaitu rendah, sedang, dan tinggi. Terdapat empat tuple dengan nilai rendah, tiga tuple dengan nilai sedang, dan empat tuple dengan nilai tinggi. Untuk menghitung *gain ratio*, perlu dihitung nilai *split information* terlebih dahulu dengan cara:

$$\begin{aligned} \text{SplitInfo}_A(D) &= -\frac{4}{14} \times \log_2\left(\frac{4}{14}\right) - \frac{6}{14} \times \log_2\left(\frac{6}{14}\right) - \frac{4}{14} \times \log_2\left(\frac{4}{14}\right) \\ \text{SplitInfo}_A(\text{pendapatan}) &= 0.926 \text{ bits} \end{aligned}$$

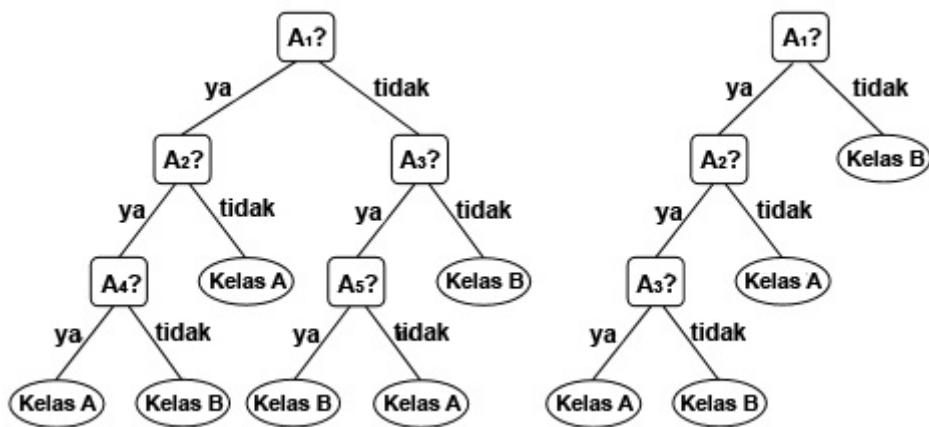
Jika nilai *gain information* dari income adalah 0.029, maka, dapat diperoleh *gain ratio* dari pendapatan adalah

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

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

Tree Pruning Tree pruning m merupakan proses pemotongan decision tree agar lebih fisik dan tidak terlalu memperluas nilai k putusan yang dihasilkan. decision tree yang sudah dipotong akan lebih kecil ukurannya, tidak semurah pohon yang asli, namun lebih mudah untuk diproses. Decision tree yang sudah dipotong m miliki ketepatan yang lebih baik [1]. Perbedaan decision tree yang sudah dipotong dan belum dapat dilihat pada gambar 2.6.

Terdapat dua tahapan 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 memotong cabang atau partisi dari sebuah node, dan membuat node tersebut menjadi leaf.

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

2.1.6 Pattern Evaluation

Pattern evaluation merupakan tahap mendeskripsikan apakah pattern atau pola tersebut merupakan narik dan merupakan sifat knowledge berdasarkan beberapa *interestingness measures*. Suatu pattern atau pola dapat dinyatakan merupakan narik apabila

- mudah dimengerti oleh manusia
- valid untuk data percobaan maupun data yang baru
- memiliki potensi atau berguna
- merupakan sifat 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 diprolol dengan cara melakukan wawancara dengan CEO KIRI, yaitu Pascal Alfadian. Data log yang dibirikan sudah dalam format XML.

Log tersebut memiliki 5 field untuk setiap tuple sebagai berikut:

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

- *Timestamp (UTC)*, waktu k tika p ngguna KIRI m ncari rut angkot m nggunakan waktu UTC / GMT
- *Action*, tip dari log yang dibuat.
- *AdditionalData*, m ncatat data-data yang b rhubungan s suai d ngan nilai atribut *action*

LogId m rupakan *field* d ngan tip data int d ngan batas 6 karakter yang digunakan s bagai primary key dari tabl t rs but. LogId diisi d ngan m nggunakan fungsi *increment integer*. *Increment integer* m rupakan fungsi untuk p ngisian data pada databas d ngan m nambahkan nilai 1 dari nilai yang t rakhir kali diisi. APIK y m rupakan *field* d ngan tip data varchar yang digunakan untuk m m riksa p ngguna KIRI k tika m nggunakan KIRI. *Timestamp (UTC)* m rupakan *field* d ngan tip data *timestamp* yang digunakan untuk m ncatat waktu p nggunaan KIRI ol h us r, diisi d ngan m nggunakan fungsi *current time*. *Current time* m rupakan fungsi untuk p ngisian data pada databas d ngan m ngambil waktu pada komput r k tika r cord dibuat. *Action* m rupakan *field* d ngan tip data varchar yang digunakan untuk m m riksa fungsi apa yang dipanggil dari API KIRI. T rdapat b b rapa tip pada *field* ini, yaitu

- *ADDAPIKEY*, action yang dicatat k tika fungsi p mbuatan *API key* yang baru dipanggil.
- *FINDROUTE*, action yang dicatat k tika us r m lakukan p ncarian rut
- *LOGIN*, action yang dicatat k tika d v lop rs m lakukan login d ngan m nggunakan *API key*
- *NEARBYTRANSPORT*, action yang dicatat k tika us r m ncari transportasi di da rah rut s dang dicari
- *PAGELOAD*, action yang dicatat k tika us r m masuki halaman KIRI
- *REGISTER*, action yang dicatat k tika d v lop rs m lakukan p ndaftaran pada KIRI *API key*
- *SEARCHPLACE*, action yang dicatat k tika us r m manggil fungsi p ncarian lokasi d ngan m nggunakan nama t mpat
- *WIDGETERROR*, m ncatat log t rs but k tika us r m n rima rrор dari *widget*
- *WIDGETLOAD*, m ncatat log t rs but k tika us r m ngdownload widg t

AdditionalData, m rupakan *field* d ngan tip data varchar yang digunakan untuk m ncatat informasi yang dibutuhkan s suai d ngan *field action*. Isi dari *additionalData* t rs but untuk s tiap *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 s rta 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 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 meregister 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 :

$$a = \sin^2((|\phi_1 - \phi_2|) \cdot \pi / 180) + \cos(\phi_1) \cdot \cos(\phi_2) \cdot \sin^2((|\lambda_1 - \lambda_2|) \cdot \pi / 180)$$

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

$$d = R \cdot c$$

Dimana

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

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

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

$$c = 0.1037900036$$

$$d = 6.371 \cdot 0.1037900036$$

$$d = 0.6612461130 * 1000 \text{ km}$$

$$d = 661.2461130 \text{ km}$$

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

2.4 Weka

Weka merupakan aplikasi berbasis Java yang berhasil menciptakan alat-alat untuk melakukan visualisasi dan algoritma untuk data analisis serta mendukung pengembangan aplikasi. Weka juga memiliki file weka-src.jar yang berisi kelas-kelas yang dipakai oleh aplikasi Weka sehingga user dapat menggunakan untuk membuat program Java yang berfungsi untuk *data mining*. Berikut beberapa kelas yang dimiliki oleh Weka:

Classifier adalah sebuah interface yang digunakan sebagai basis untuk mendefinisikan klasifikasi ataupun nominal pada Weka. Kelas tersebut memiliki metode sebagai berikut:

- void buildClassifier(Instances data)

untuk melakukan penghasilan klasifikasi dengan parameter tersebut data latihan.

- double classifyInstance(Instances instance)

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

- double[] distributionForInstance(Instances instance)

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

- Capabilities getCapabilities()

mengembalikan kapabilitas dari kelas tersebut.

Instance adalah interface yang mewakili set data.

Method:

- Attribut getAttribut(int index)

Mengembalikan atribut dari indeks yang diberikan.

- Attribut classAttribut ()

M ng mbalikan atribut k las.

- int classInd x()

M ng mbalikan ind ks atribut k las itu.

- bool an classIsMissing()

M ng c k apakah k las turunan hilang.

- doubl classValu ()

M ng mbalikan nilai k las contoh s bagai angka floating-point.

- Instanc s datas t()

M ng mbalikan datas t.

- void d l t Attribut At(int position)

M nghapus atribut pada posisi t rt ntu.

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

M ng mbalikan p ngitungan s mua atribut.

- bool an qualH ad rs(Instanc inst)

P ngujian jika h ad r dari dua contoh yang s tara.

- java.lang.String qualH ad rsMsg(Instanc inst)

M m riksa apakah h ad r dari dua contoh yang s tara.

- bool an hasMissingValu ()

T s apakah s buah contoh m miliki nilai yang hilang.

- ind x(int position)

M ng mbalikan ind x dari atribut yang t rsimpan di posisi t rt ntu.

- void ins rtAttribut At(int position)

M nyisipkan atribut pada posisi t rt ntu.

- bool an isMissing(Attribut att)

P ngujian jika nilai t rt ntu yang hilang.

- bool an isMissing(int attInd x)

P ngujian jika nilai t rt ntu yang hilang.

- bool an isMissingSpars (int ind xOfInd x)

P ngujian jika nilai t rt ntu yang hilang.

- Instanc m rg Instanc (Instanc inst)

M nggabungkan contoh yang dib rikan dan m ng mbalikan hasilnya.

- int numAttribut s()

M ng mbalikan jumlah atribut.

- int numClass s()

M ng mbalikan jumlah lab l k las.

- int numValu s()

M ng mbalikan jumlah nilai.

- Instanc s r lationalValu (Attribut att)

M ng mbalikan nilai r lasional atribut r lasional.

- Instanc s r lationalValu (int attInd x)

M ng mbalikan nilai r lasional atribut r lasional.

- void r plac MissingValu s(doubl [] array)

M nggantikan s mua nilai yang hilang dalam contoh d ngan nilai-nilai yang t rkandung dalam array yang dib rikan.

- void s tClassMissing()

M n tapkan nilai k las contoh untuk hilang.

- void s tClassValu (doubl valu)

M n tapkan nilai k las turunan d ngan nilai yang dib rikan (format floating-point).

- void s tClassValu (java.lang.String valu)

M n tapkan nilai k las turunan d ngan nilai yang dib rikan.

- void s tDatas t(Instanc s instanc s)

M ngatur r f r nsi datas t.

- void s tMissing(Attribut att)

M n tapkan nilai t rt ntu dijadikan hilang.

- void s tMissing(int attInd x)

M n tapkan nilai t rt ntu dijadikan hilang.

- void s tValu (Attribut att, doubl valu)

M n tapkan nilai t rt ntu dalam hal untuk nilai yang dib rikan (format floating-point).

- void s tValu (Attribut att, java.lang.String valu)

M n tapkan nilai atribut nominal atau string k nilai yang dib rikan.

- `void s tValu (int attInd x, doubl valu)`
Menapkan nilai tertentu untuk nilai yang dibutuhkan (format floating-point).
- `void s tValu (int attInd x, java.lang.String valu)`
Menapkan nilai atribut nominal atau string ke nilai yang dibutuhkan.
- `void s tValu Spars (int ind xOfInd x, doubl valu)`
Menapkan nilai tertentu dalam contoh dengan nilai yang dibutuhkan (format floating-point).
- `void s tW ight(doubl w ight)`
Menentukan bentuk contoh.
- `java.lang.String stringValu (Attribut att)`
Menyalin nilai nominal, string, tanggal, atau atribut klasional untuk contoh sebagai string.
- `java.lang.String stringValu (int attInd x)`
Menyalin nilai nominal, string, tanggal, atau atribut klasional untuk contoh sebagai string.
- `doubl [] toDoubl Array()`
Menyalin nilai-nilai masing-masing atribut sebagai array ganda.
- `java.lang.String toString(Attribut att)`
Menyalin deskripsi satu nilai dari contoh sebagai string.
- `java.lang.String toString(Attribut att, int aft rD cimalPoint)`
Menyalin deskripsi satu nilai dari contoh sebagai string.
- `java.lang.String toString(int attInd x)`
Menyalin deskripsi satu nilai dari contoh sebagai string.
- `java.lang.String toString(int attInd x, int aft rD cimalPoint)`
Menyalin deskripsi satu nilai dari contoh sebagai string.
- `java.lang.String toStringNoW ight()`
Menyalin deskripsi satu contoh (tanpa bentuk ditambahkan).
- `doubl valu (Attribut att)`
Menyalin nilai atribut contoh dalam format int rnal.
- `doubl valu (int attInd x)`
Menyalin nilai atribut contoh dalam format int rnal.
- `doubl valu Spars (int ind xOfInd x)`
Menyalin nilai atribut contoh dalam format int rnal.

- double weight()

Mengembalikan berat contoh itu.

Instances adalah kelas untuk mengelola set data.

Atribut:

- String ARFF_DATA

digunakan untuk menunjukkan struktur arff data.

- String ARFF_RELATION

digunakan untuk menunjukkan hubungan arff data.

- String FILE_EXTENSION

Extensi dari nama file yang digunakan untuk file arff.

- String SERIALIZED_OBJ_FILE_EXTENSION

Extensi dari nama file yang digunakan untuk bin.

Constructor:

- Instances(Instances data)

Konstruktor menyalin semua contoh dan referensi untuk informasi hubungan dari himpunan contoh.

- Instances(Instances data, int capacity)

Konstruktor 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 membaca bobot satu untuk setiap contoh.

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

Menciptakan himpunan kosong contoh.

Method:

- boolean add(Instances instance)

Mengambahkan set data.

- void add(int index, Instances instance)

Mengambahkan satu contoh di posisi tertentu dalam daftar.

- Attribut getAttribut(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 dataset ini untuk atribut tertentu.

- boolean checkAttributTyp (int attTyp)

Cek untuk atribut dari tip yang dibutuhkan dalam dataset.

- boolean checkStringAttributS()

Cek string atribut dalam dataset.

- boolean checkInstanc (Instanc instanc)

Meriksa apakah contoh yang dibutuhkan kompatibel dengan dataset ini.

- Attribut classAttribut ()

Mengembalikan atribut kelas.

- int classInd x()

Mengembalikan indeks atribut kelas itu.

- void del t ()

Menghapus semua contoh dari set.

- void del t (int ind x)

Menghapus sebuah contoh di posisi tertentu dari set.

- void del t Attribut At (int position)

Menghapus atribut pada posisi tertentu.

- void del t Attribut Typ (int attTyp)

Menghapus semua atribut dari tip yang dibutuhkan dalam dataset.

- void del t StringAttribut S()

Menghapus semua atribut string dalam dataset.

- void del t WithMissing(Attribut att)

Menghapus semua contoh dengan nilai-nilai yang hilang untuk atribut tertentu dari dataset.

- void del t WithMissing(int attInd x)

Menghapus semua contoh dengan nilai-nilai yang hilang untuk atribut tertentu dari dataset.

- void d l t WithMissingClass()
Menghapus semua contoh dengan nilai klas hilang dari dataset t.
- java.util.Enumeration<Attribut> numRateAttributes()
Pembalikan penghitungan semua atribut.
- java.util.Enumeration<Instanc> numRateInstances()
Pengembalian penghitungan semua contoh dalam dataset t.
- boolean qualHasAttributes(Instances dataset)
Cek jika dua hal tersebut sama.
- java.lang.String qualHasMsg(Instances dataset)
Cek jika dua hal tersebut sama.
- Instances firstInstance()
Mengembalikan contoh pertama di dataset t.
- Instances get(int index)
Mengembalikan contoh pada posisi t-rt-ntu.
- java.util.Random getRandomNumberGenerator(long seed)
Mengembalikan nomor acak.
- java.lang.String getVersion()
Mengembalikan string versi.
- void insertAttribute(Attribute att, int position)
Menyisipkan atribut pada posisi t-rt-ntu (0 numAttributes()) dan menambahkan semua nilai hilang.
- Instances instances(int index)
Mengembalikan contoh pada posisi t-rt-ntu.
- double kthSmallestValue(Attribute att, int k)
Mengembalikan nilai atribut k-terkecil dari atribut numrik.
- double kthSmallestValue(int attributeIndex, int k)
Mengembalikan nilai atribut k-terkecil dari atribut numrik.
- Instances lastInstance()
Mengembalikan contoh terakhir di dataset t.
- static void main(java.lang.String[] args)
Metode utama untuk klas 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 Instances mergeInstances(Instances first, Instances second)

Menggabungkan dua set Contoh bersama-sama

- int numAttributes()

Mengembalikan jumlah atribut.

- int numClasses()

Mengembalikan jumlah kelas.

- int numDistinctValues(Attribut att)

Mengembalikan jumlah nilai yang berbeda dari atribut yang diberikan.

- int numDistinctValues(int attInd x)

Mengembalikan jumlah nilai yang berbeda dari atribut yang diberikan.

- int numInstances()

Mengembalikan jumlah kasus dalam dataset.

- void randomize (java.util.Random random)

Mengocok contoh di set se hingga mereka masing-masing diambil secara acak.

- java.lang.String relationName()

Mengembalikan nama hubungan itu.

- Instances remove (int index x)

Menghapus contoh pada posisi tertentu.

- void renameAttribute (Attribut attribute, java.lang.String name)

Mengganti nama atribut.

- void renameAttribute (int attribute, java.lang.String name)

Mengganti nama atribut.

- void renameAttributeValue (Attribut attribute, java.lang.String value, java.lang.String name)

Mengganti nama nilai nominal (atau string) nilai atribut

- void renameAttributeValue (int attribute, int value, java.lang.String name)

Mengganti nama nilai nominal (atau string) nilai atribut.

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

M ngantikan 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 WithWeights(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 WithWeights(java.util.Random random, boolean an r pr s ntUsingWeights)`

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 WithWeights(java.util.Random random, boolean 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 WithWeights(java.util.Random random, boolean an[] sampl d, boolean an r pr - s ntUsingWeights)`

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 WithWeights(java.util.Random random, double [] weights)`

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 WithWeights(java.util.Random random, double [] weights, boolean 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 WithWeights(java.util.Random random, double [] weights, boolean an[] sampl d, boolean an r pr s ntUsingWeights)`

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 ngantikan 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 s tR lationNam (java.lang.String n wNam)
M ngatur nama hubungan itu.
- int siz ()
M ng mbalikan banyak data dalam datas t.
- void sort(Attribut att)
Urutkan contoh b rdasarkan atribut.
- void sort(int attInd x)
Urutkan contoh b rdasarkan atribut.
- void stabl Sort(Attribut att)
Urutkan contoh b rdasarkan atribut, m nggunakan s macam stabil.
- void stabl Sort(int attInd x)
Urutkan contoh b rdasarkan atribut, m nggunakan s macam stabil
- void stratify(int numFolds)
M ng lompokkan satu s t contoh s suai d ngan nilai-nilai k lasnya jika atribut k las nominal (s hingga s t lah cross-validasi b rlapis dapat dilakukan).
- Instanc s stringFr Structur ()
Buat salinan struktur.
- doubl sumOfW ights()
M nghitung jumlah s mua bobot contoh.
- void swap(int i, int j)
m nukar posisi dua contoh di s t.
- static void t st(java.lang.String[] argv)
M tod p ngujian k las ini.
- Instanc s t stCV(int numFolds, int numFold)
M nciptakan s t t s untuk satu kali lipat dari cross-validation pada datas t.
- java.lang.String toString()
M ng mbalikan datas t s bagai string dalam format ARFF.
- java.lang.String toSummaryString()
M nghasilkan string m ringkas s t contoh

- `Instances trainCV(int numFolds, int numFold)`

Menciptakan p latihan ditapkan untuk satu kali lipat dari cross-validasi pada dataset t.

- `Instances trainCV(int numFolds, int numFold, java.util.Random random)`

Menciptakan p latihan ditapkan untuk satu kali lipat dari cross-validasi pada dataset t.

- `double variance(Attribute att)`

Menghitung varians untuk atribut numrik.

- `double variance(int attInd x)`

Menghitung varians untuk atribut numrik.

- `double[] variances()`

Menghitung varians untuk semua atribut numrik secara bersamaan.

Attribute adalah kelas yang digunakan untuk mendefinisikan atribut.

Atribut:

- `static java.lang.String ARFF_ATTRIBUTE`

Kata kunci yang digunakan untuk menunjukkan awal atribut dalam klarasi 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 numrik.

- `static java.lang.String ARFF_ATTRIBUTE_NUMERIC`

Kata kunci yang digunakan untuk menunjukkan atribut numrik.

- `static java.lang.String ARFF_ATTRIBUTE_REAL`

Kata kunci yang digunakan untuk menunjukkan atribut numrik.

- `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 subrelasi.

- `static int DATE`

Nilai konstan untuk atribut dengan nilai tanggal.

- static java.lang.String DUMMY_STRING_VAL
Dummy pertama nilai String atribut.
- static int NOMINAL
Sifat konstan untuk atribut nominal.
- static int NUMERIC
Sifat konstan untuk atribut numerik.
- static int ORDERING_MODULO
Sifat konstan untuk atribut ordering modulo.
- static int ORDERING_ORDERED
Sifat konstan untuk atribut miring ditahkan.
- static int ORDERING_SYMBOLIC
Sifat konstan untuk atribut simbolik.
- static int RELATIONAL
Sifat konstan untuk atribut nilai relasi.
- static int STRING
Sifat konstan untuk atribut dengan nilai-nilai string.

Constructor:

- Attribut (java.lang.String attribut_Nam)
Konstruktor untuk atribut numerik.
- Attribut (java.lang.String attribut_Nam, Instanc_shadr)
Konstruktor untuk atribut nilai relasi.
- Attribut (java.lang.String attribut_Nam, Instanc_shadr, int ind_x)
Konstruktor untuk atribut nilai relasi dengan index tertentu.
- Attribut (java.lang.String attribut_Nam, Instanc_shadr, PrototypedProperties m_tadata)
Konstruktor untuk atribut nilai relasi.
- Attribut (java.lang.String attribut_Nam, int ind_x)
Konstruktor untuk atribut numerik dengan index tertentu.
- Attribut (java.lang.String attribut_Nam, java.util.List<java.lang.String> attribut_Valus)
Konstruktor untuk atribut nominal dan atribut string.
- Attribut (java.lang.String attribut_Nam, java.util.List<java.lang.String> attribut_Valus, int ind_x)
Konstruktor untuk atribut nominal dan atribut string dengan index tertentu.

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

Konstruktor 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)

Konstruktor untuk atribut num rik, di mana m tadata dib rikan.
- Attribut (java.lang.String attribut Nam , java.lang.String dat Format)

Konstruktor untuk tanggal atribut.
- Attribut (java.lang.String attribut Nam , java.lang.String dat Format, int ind x)

Konstruktor 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)

Konstruktor 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 addStringValu (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 addStringValu (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 nghasilkan salinan atribut ini.
- Attribut copy(java.lang.String n wNam)

M nghasilkan 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.
- boolean 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)`

Mengembalikan milid tik s suai dengan tanggal saat ini.

- `java.util.String getDat Format()`

Mengembalikan pola format tanggal dalam hal atribut ini adalah tip dat , s lain itu, maka string akan kosong.

- `doubl getLow rNum ricBound()`

Pengembalian batas bawah dari atribut num rik.

- `Protect dProp rti s getM tadata()`

Mengembalikan properti yang disediakan untuk atribut ini.

- `java.lang.String getR vision()`

Mengembalikan string r visi.

- `doubl getUpp rNum ricBound()`

Mengembalikan nilai dari atribut num rik.

- `int hashCod ()`

Mengembalikan kod hash untuk atribut ini berdasarkan namanya.

- `boolean hasZeropoint()`

Pengembalian apakah atribut memiliki zeropoint.

- `int indexOfX()`

Mengembalikan index x dari atribut ini.

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

Mengembalikan index x dari nilai atribut tersebut.

- `boolean isAveragable ()`

Pengembalian apakah atribut dapat dirata-ratakan bermakna.

- `boolean isDate ()`

Pengujian jika atribut adalah jenis tanggal.

- `boolean isInRange (doubl valu)`

Mengecek apakah suatu nilai termasuk dalam batas-batas atribut.

- `boolean isNominal()`

Mengecek apakah atribut nominal.

- `boolean isNumeric()`

Pengujian jika atribut num rik.

- boolean isRelationValue()

Pengujian jika atribut hubungan dihargai.

- boolean isString()

Pengujian jika atribut string.

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

Mengutama yang sedarhanakan untuk mengujiklas ini.

- java.lang.String name()

Mengembalikan nama atribut itu.

- int numValues()

Mengembalikan jumlah nilai atribut.

- int ordering()

Mengembalikan posisi sanan atribut.

- int parseData(java.lang.String str)

Mengurai string yang dibirikan sebagai data, sesuai format saat ini dan mengembalikan sesuai dengan jumlah miliditik.

- Instance relation()

Mengembalikan informasi hadir untuk atribut nilai relasi, null jika atribut tidak memiliki hubungan.

- Instance relation(int valIndex)

Mengembalikan nilai atribut nilai relasi.

- void setStringValue(java.lang.String value)

Mengosongkan nilai dan mengatur mungkin agar hanya nilai yang dibirikan.

- void setWeight(double value)

Mengatur berat atribut baru.

- java.lang.String toString()

Pengembalian dekrripsi atribut ini dalam format ARFF.

- int type()

Mengembalikan jenis atribut sebagai integer.

- static java.lang.String toString(Attribut att)

Mengembalikan representasi string dari jenis atribut.

- static java.lang.String toString(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.
- doubl weight()

Mengembalikan berat 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(Instances data)

Membangun ID3 pohon klasifikasi.
- doubl classifyInstance(Instances instanc)

Mengklasifikasikan satu data yang dibedakan dengan menggunakan pohon klasifikasi.
- doubl [] distributionForInstance(Instances instanc)

Menghitung distribusi klasifikasi instance menggunakan pohon klasifikasi.
- Capabilities getCapabilities()

Mengembalikan detail klasifikasi.
- java.lang.String getRevision()

Mengembalikan String revisi.
- TechnicalInformation getTechnicalInformation()

Mengembalikan sebuah objek TechnicalInformation, yang berisi informasi rincian tentang latar belakang teknis klasifikasi ini.
- java.lang.String globalInfo()

Mengembalikan string yang menjelaskan klasifikasi.
- static void main(java.lang.String[] args)

Metoda utama untuk klasifikasi ini.
- java.lang.String toSource (java.lang.String className)

Mengembalikan string yang menggambarkan klasifikasi.
- java.lang.String toString()

Mencetak pohon klasifikasi menggunakan metod toString.

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

Constructor:

- ID3()

Method:

- java.lang.String binarySplitsTipTree() Membalikan tipe tip untuk properti ini.

- void buildClassifier(Instances instances) Menghasilkan classifier.

- double classifyInstance(Instance instance) Mengklasifikasikan satu data.

- java.lang.String confidenceFactorTipTree() Membalikan tipe tip untuk properti ini.

- double[] distributionForInstance(Instance instance) Mengembalikan probabilitas klas untuk sebuah data.

- java.util.Enumeration numRatesMeasurements() Mengembalikan penghitungan ukuran.

- boolean isBinarySplits() Dapatkan nilai binarySplits.

- Capabilities getCapabilities() Mengembalikan kapabilitas dari klas ini.

- float getConfidenceFactor() Mengembalikan nilai *confident*.

- double getMismeasurementRate(java.lang.String additionalMeasurementName) Mengembalikan nilai bobot suai nama.

- int getMinNumObj() Dapatkan nilai minNumObj.

- int getNumFolds() Dapatkan nilai numFolds.

- java.lang.String getOptions() Mengandalkan pengaturan saat ini.

- boolean getErrorPruning()

Dapatkan nilai r dari dErrorPruning.

- java.lang.String getVision()

Mengembalikan string r visi.

- boolean getSaveInstanceData()

Periksa apakah contoh data disimpan.

- int getSize()

Dapatkan nilai s dari d.

- boolean getSubtreeRaising()

Dapatkan nilai subtree Raising.

- TechnicalInformation getTechnicalInformation()

Mengembalikan s buah instance dari objek TechnicalInformation, yang berisi informasi rinci tentang latar belakang teknis klas ini.

- boolean getUnpruned()

Mengkondisikan apakah dilakukan tree pruning.

- boolean getUsesLaplac()

Dapatkan nilai uses Laplac.

- java.lang.String globalInfo()

Mengembalikan string yang menunjukkan klasifikasi r.

- java.lang.String graph()

Pembelahan Grafik menggambarkan pohon.

- int graphType()

Mengembalikan jenis grafik klasifikasi r.

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

Mendefinisikan utama untuk menjalankan program ini.

- double measureNumeLeafs()

Mengembalikan jumlah daun.

- double measureNumerRules()

Mengembalikan jumlah aturan.

- double measureTreeSize()

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(bool an 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(bool an v)`
M ngatur nilai r duc dErrorPruning.
- `void s tSav Instanc Data(bool an 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(bool an v)`
M ngatur nilai subtr Raising.

- void s tUnprun d(bool an v)
M ngatur nilai *pruning*.
- void s tUs Laplac (bool an n wus Laplac)
M ngatur nilai us Laplac .
- java.lang.String subtr RaisingTipT xt()
M ng mbalikan t ks tip untuk prop rti ini.
- java.lang.String toString()
P ng mbalian d skripsi classifi r.
- java.lang.String unprun dTipT xt()
M ng mbalikan t ks tip untuk prop rti ini.
- java.lang.String us Laplac TipT xt()
M ng mbalikan t ks tip untuk prop rti ini.

NumericToNominal adalah k las yang digunakan untuk m ngubah nilai num rik m njadi nominal.

Constructor:

- NumericToNominal()

Method:

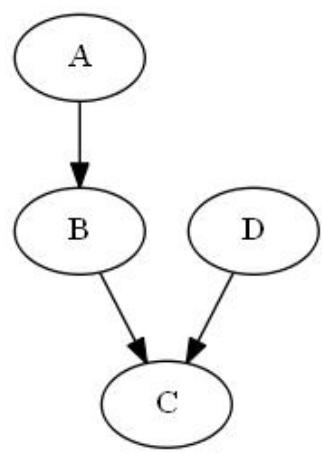
- java.lang.String[] g tOptions()
M ng mbalikan p ngaturan dari filt r.
- java.lang.String g tR vision()
m ng mbalikan r visi.
- java.lang.String globalInfo()
M ng mbalikan string yang b risi d skripsi dari k las t rs but.
- static void main(java.lang.String[] args)
M njalankan filt r d ngan input param t r.
- void s tAttribut Indic s(java.lang.String valu)
M lakukan p ny tingan untuk m milih atribut yang akan difilt r.
- void s tAttribut Indic sArray(int[] valu)
M lakukan p ny tingan untuk m milih atribut yang akan difilt r.
- bool an s tInputFormat(Instanc s instanc)
M lakukan p ny tingan untuk input data.
- void s tOption(String[] option)
M lakukan p ny tingan p ngaturan.

2.5 Graphviz

Graphviz merupakan perangkat lunak *open source* untuk visualisasi grafik. Dengan menggunakan graphviz, visualisasi grafik dapat dibuat dengan menulis kod. Berikut contoh kod yang dapat dijadikan input untuk aplikasi graphviz:

```
digraph{  
    A -> B  
    B -> C  
    D -> C  
}
```

Maka hasil yang dipilih dari perangkat lunak graphviz [2.7](#)



Gambar 2.7: Hasil output Graphviz

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 menemukan nilai lokasi kota rangkatan 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 dilanjutkan.

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 bersifat dari satu tabel, maka tahap ini dapat dilanjutkan.

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 kota rangkatan dan tujuan dari seorang 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 kota rangkatan dan tujuan dari user. Selain itu, data tersebut terlihat memerlukan karakteristik dimungkinkan dapat menghasilkan suatu pola yang membantu melakukan klasifikasi mengenai perpindahan penduduk khususnya untuk daerah Bandung. Karena seluruh *action* bernilai satuan yaitu *FINDROUTE*, maka atribut tersebut dapat dihilangkan. Selain itu, atribut *logId* dan *APIK* yang tidak akan dimasukkan ke dalam proses karena tidak memiliki hubungan dengan lokasi kota rangkatan dan tujuan dari seorang user.

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

- *Timestamp (UTC)*
- *AdditionalData*

Berikut contoh data dari atribut tersebut dapat dilihat pada tabel 3.1

Tab 3.1: Contoh data log KIRI saat 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 longitudo dari lokasi kota yang dipilih oleh user
2. Nilai latitud dan longitudo 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 masuki tahap *data transformation*, karena nilai tersebut hanya menunjukkan banyak jalur tetapi user pasti hanya memiliki salah satu dari jalur tersebut, sehingga nilai jalur ini dapat diasumsikan memiliki nilai 1 semuanya. Karena kolom jalur bernilai satu semuanya, 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 format atribut, yaitu:

- Bulan, atribut ini akan menunjukkan bulan ketika user KIRI menggunakan action *FINDROUTE*, dengan nilai antara 01 sampai 12. Nilai tersebut dapat dipilih dengan cara mengambil nilai string dari timestamp yang berada di antara garis miring pertama dan kedua.
- Tahun, atribut ini akan menunjukkan tahun ketika user KIRI menggunakan action *FINDROUTE*, dengan format empat angka (contoh: 2014). Nilai tersebut dapat dipilih dengan cara mengambil nilai string dari timestamp yang berada di antara garis miring ke dua dan spasi.
- Hari, atribut ini akan menunjukkan hari ketika user KIRI menggunakan action *FINDROUTE*, dengan rangkaian nilai antara sembilan sampai minggu. Nilai tersebut dapat dipilih dengan cara melakukan menggunakan method pencarian hari berdasarkan tanggal dari timestamp pada java.

- Jam, atribut ini akan menunjukkan jam ketika user KIRI menggil action *FINDROUTE*, dengan rang nilai antara 00 sampai 23. Nilai tersebut dapat dipilih dengan cara mengambil nilai string dari timestamp yang berada di antara spasi dan titik dua.

Data *timestamp* diubah menjadi nam bagian, agar dapat dilakukan pengelompokan yang dilihat dari tanggal, bulan, tahun, hari, jam dan menit.

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

- Latitud ke berasarkan, atribut ini berisi nilai latitud dari lokasi ke berasarkan yang dipilih oleh user. Nilai tersebut dapat dipilih dengan cara mengambil nilai string setelah koma yang pertama.
- Longitud ke berasarkan, atribut ini berisi nilai longitudo dari lokasi ke berasarkan yang dipilih oleh user. Nilai tersebut dapat dipilih dengan cara mengambil nilai string yang berada di antara koma pertama dan garis miring pertama.
- Latitud tujuan, atribut ini berisi nilai latitud dari lokasi tujuan yang dipilih oleh user. Nilai tersebut dapat dipilih dengan cara mengambil nilai string di antara garis miring yang pertama dan koma kedua.
- Longitud tujuan, atribut ini berisi nilai longitudo dari lokasi tujuan yang dipilih oleh user. Nilai tersebut dapat dipilih dengan cara mengambil nilai string yang berada di antara koma kedua dan garis miring kedua.

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

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

- Bulan
- Tahun
- Hari
- Jam
- Latitud Ke berasarkan
- Longitud Ke berasarkan
- Latitud Tujuan
- Longitud Tujuan

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

Bulan	Tahun	Hari	Jam	Latitude Ke-berangkatan	Longitude Keberang-katan	Latitude Tujuan	Longitude Tujuan
02	2014	Sabtu	00	-6.8972513	107.6185574	-6.91358	107.62718
02	2014	Sabtu	00	-6.8972513	107.6385574	-6.91358	107.62718
02	2014	Sabtu	00	-6.90598	107.59714	-6.90855	107.61082
02	2014	Sabtu	00	-6.9015366	107.5414474	-6.88574	107.53816
02	2014	Sabtu	00	-6.90608	107.61530	-6.89140	107.61060
02	2014	Sabtu	00	-6.89459	107.58818	-6.89876	107.60886
02	2014	Sabtu	00	-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 dip ral h *decision tree* m ng nai lokasi k b rangkatan dan tujuan dari us r KIRI, maka atribut k las yang akan digunakan adalah nilai latitud dan longitud dari lokasi k b rangkatan 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 rangkatan dan tujuan akan diubah m njadi nilai yang m nunjukkan apakah da rah lokasi k b rangkatan dan tujuan t rs but m nunjukkan p rjalanan k luar dari Bandung, m nuju Bandung, atau m nuju da rah yang sama. Hal ini dilakukan agar dip ral h data p rbandingan p rg rakan p nduduk, apakah m r ka l bih banyak yang k luar dari Bandung atau s baliknya atau bahkan m nuju da rah yang sama 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 apakan user

2.66TJ-387.78(m n)27(u.)-arkan wu.u.a273(atau)-inm 5
mē

sama, maka user tidak but maka user tidak but barang rak di dalam ruang yang sama.

Dengan adanya perhitungan jarak dan pernentuan daerah Bandung, nilai latitud dan longitud dari lokasi keberangkatan dan tujuan dapat dibuang dan diganti oleh atribut m_nujuBandung dengan tipe data integer. Jika isi dari atribut t_rs but barang nilai 1, maka user tidak but m_nuju Bandung sanganan nilai 0 berarti user tidak m_nuju Bandung, dan jika nilai atribut t_rs but adalah 2, maka user tidak but m_miliki lokasi keberangkatan dan tujuan pada daerah yang sama. Contoh hasil data setelah dilakukan transformation terhadap latitud dan longitud terdapat pada tabel 3.3.

Tab 3.3: Contoh hasil data transformasi latitud longitud

Bulan	Tahun	Hari	Jam	Arah Keberangkatan
02	2014	Sabtu	00	-1
02	2014	Sabtu	00	1
02	2014	Sabtu	00	1
02	2014	Sabtu	00	0
02	2014	Sabtu	00	1
02	2014	Sabtu	00	-1
02	2014	Sabtu	00	0

3.2 Analisis Perangkat Lunak

Agar analisis pola dari lokasi keberangkatan dan tujuan dari data log histori 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 dipilih sebelumnya. proses dijalankan yaitu perangkat lunak data mining log histori KIRI.

Perangkat lunak yang dibangun akan berbasis desktop dan menggunakan bahasa programan java. Pada subbab ini akan dibahas spesifikasi kebutuhan funsional, pemodelan perangkat lunak, diagram use case, scenario, diagram klas dari Perangkat Lunak yang akan dibangun.

Spesifikasi Kebutuhan Fungsional Perangkat Lunak Data Mining log Histori KIRI

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

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

Pemodelan Perangkat Lunak Data Mining Log Histori KIRI

Perangkat lunak data mining log histori KIRI akan mendapat input data text dengan format .txt. Setelah program mendapatkan input dan memproseskan tombol proses, maka data tersebut akan diu-

bah t rl bih dahulu s suai pada bab analisis data(bab 3.1) d ngan m lakukan pros s *data transform* dan m nghasilkan data d ngan format s p rti pada tab 1 3.3.

Program akan m lakukan tahap *data mining* d ngan m nggunakan teknik ID3 atau C4.5 s suai d ngan p rmintaan user. S t lah pros s *data mining* s l sai dilakukan, program akan m lakukan visualisasi *decision tree* d ngan m nggunakan graphviz.

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

Kar na data yang diprolah sudah dalam bentuk csv, maka pada p n litian ini, tidak akan m nggunakan sistem databas .

Ketika tombol proses ditekan, maka data tersebut akan diproses. Prosес yang pertama yang akan dilakukan adalah m lakukan *load data* dari file .data csv akan dibaca d ngan m nggunakan CSV Reader 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. Prosес *transform* tersebut memiliki tahap sebagai berikut:

1. Mengubah waktu dari UTC menjadi GMT+8 pada string data input array ketiga (yaitu atribut tanggal).
2. Mengambil atribut tanggal ketika muncul cah nilai tersebut but dengan spasi sebagai tanda pemisah, maka akan terdapat tiga nilai, yaitu hari (dalam bentuk angka dimana nilai 1 berarti senin dan nilai 7 berarti minggu), tanggal dan jam.
3. Pada nilai tanggal, dilakukan pemecahan nilai string dengan garis miring sebagai tanda pemisah, maka akan diprolah tiga nilai yaitu bulan, tanggal, dan tahun, namun nilai yang akan diambil hanya dua, yaitu bulan dan tahun.
4. Pada nilai jam, dilakukan pemecahan nilai string dengan titik dua sebagai tanda pemisah, maka akan diprolah dua nilai yaitu jam dan menit, namun nilai yang akan diambil hanya jam.
5. Mengambil string data input array ke lima (yaitu atribut *additionalData*), dilakukan pemecahan nilai string dengan garis miring sebagai tanda pemisah, maka akan diprolah 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 diprolah dua nilai untuk setiap lokasi, yaitu *latitude* dan *longitude*.
7. Menghitung jarak posisi lokasi awal dan lokasi tujuan terhadap titik pusat dan menentukan apakah lokasi tersebut berada pada klasifikasi nol atau pertama atau kedua.
8. Menggabungkan nilai-nilai tersebut dalam satu array, yaitu array dengan tip int (dengan nilai bulan, tahun, hari, jam dan menuju Bandung).

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

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

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

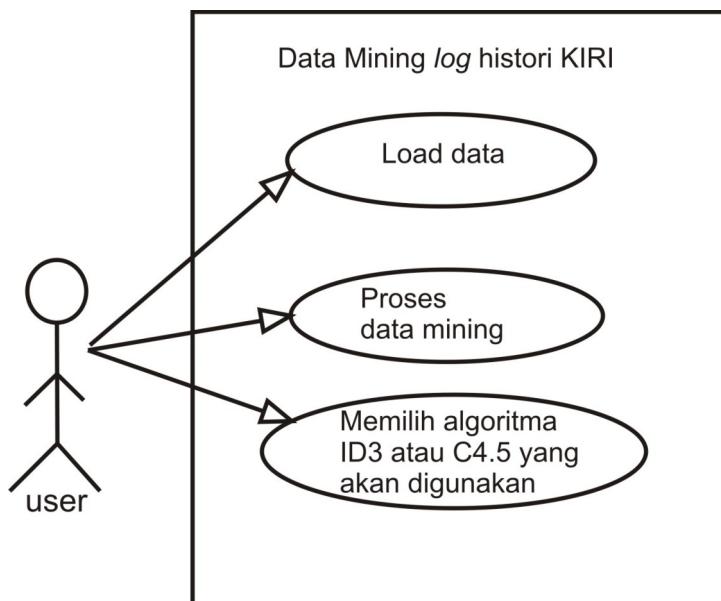
1. Program akan memuat data dan melakukan *processing data*
2. Program akan menjalankan algoritma untuk membuat *decision tree*
3. Program akan membuat grafik dari hasil algoritma *decision tree*
4. Program akan menampilkan grafik *decision tree*

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

Diagram use case merupakan diagram yang mendeskripsikan sistem dengan lingkungannya. Pada perangkat lunak 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 memasukan alamat data pada program
- Memilih algoritma yang akan digunakan, terdapat dua algoritma, yaitu ID3 dan C4.5
- Melakukan proses *data mining* dengan input data dari alamat data yang sudah dimasukan. Setelah proses berhasil dilaksanakan, program akan menampilkan hasil yang dipilih

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: Skenario M melakukan *load Data*

Nama	Load data
Aktor	<i>User</i>
D skripsi	Masukkan alamat data yang akan dijadikan sebagai input program
Kondisi awal	Textbox belum terisi
Kondisi akhir	Textbox sudah terisi dengan alamat data
Skenario utama	<i>User</i> memasukkan alamat data pada textbox
Eksplorasi	Data tidak ditemukan

Tab 1 3.6: Skenario M melakukan *Data Mining*

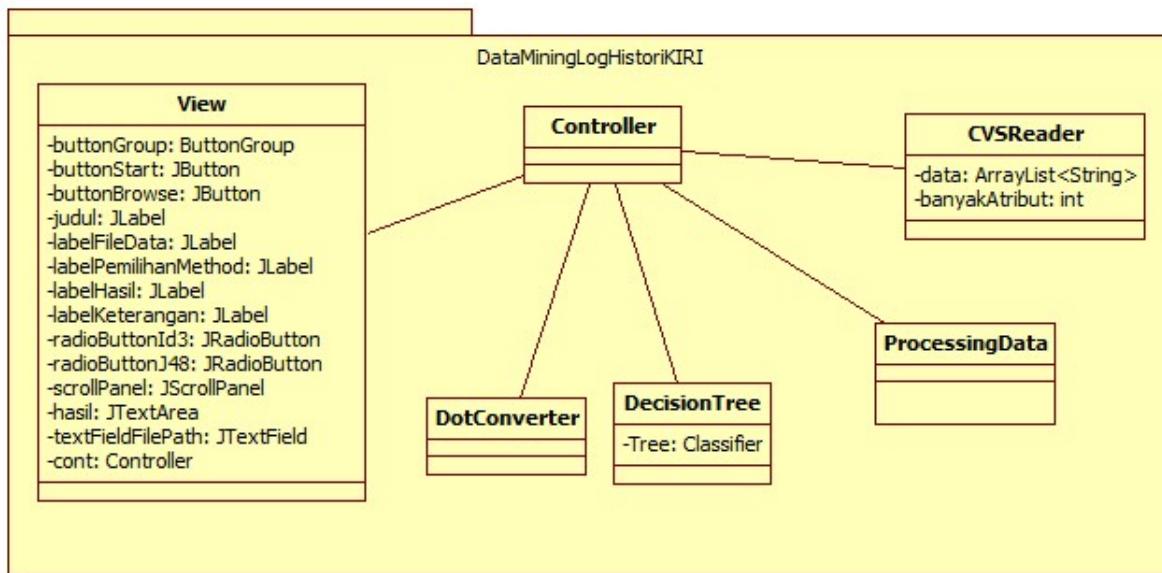
Nama	Proses <i>Data Mining</i>
Aktor	<i>User</i>
D skripsi	Mengklik tombol proses pada interface
Kondisi awal	Textbox belum terisi
Kondisi akhir	Textbox sudah terisi dengan hasil <i>data mining</i>
Skenario utama	<i>User</i> mengklik tombol proses
Eksplorasi	Data tidak ditemukan atau data tidak dapat diproses

Tab 1 3.7: Skenario M memilih Algoritma yang Akan Digunakan

Nama	Memilih algoritma ID3 atau C4.5
Aktor	<i>User</i>
D skripsi	Pilih algoritma yang akan dipakai
Kondisi awal	<i>RadioButton</i> terpilih pada ID3
Kondisi akhir	<i>RadioButton</i> terpilih pada ID3 atau C4.5
Skenario utama	<i>User</i> memilih algoritma yang akan digunakan
Eksplorasi	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 tersebut dapat pada gambar 3.3.



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

Berikut di bawah ini adalah diagram *class*:

- `View` merupakan kelas untuk mengatur tampilan antar muka.
- `Controller` merupakan kelas untuk mengatur `View` dan modul kota program dijalankan.
- `CSVReader` adalah kelas yang memiliki metode untuk membaca file dalam format CSV.
- `ProcessingData`, merupakan kelas yang memiliki metode untuk melakukan *preprocessing* data.
- `DecisionTree`, merupakan kelas yang memiliki metode untuk membuat *decision tree* dan menghitung *confident* dari pohon yang sudah dihasilkan.
- `DotConverter`, merupakan kelas yang memiliki metode untuk mengubah `string` yang merupakan hasil dari kelas `DecisionTree` (yaitu, *decision tree* dalam bentuk `string`) menjadi bahasa `dot` yang siap dijadikan masukan untuk `graphviz`.

BAB 4

PERANCANGAN PERANGKAT LUNAK

Bab ini berisi tentang perancangan perangkat lunak untuk melakukan proses *data mining* sesuai analisa yang sudah dibahas pada bab 3.

4.1 Perancangan Perangkat Lunak

4.1.1 Perancangan Kelas

Agar perangkat lunak dapat menjalankan fungsi yang sudah dibahas pada pembelajaran fungsi di bab 3, maka pada subbab ini akan dibahas rancangan kelas dan *method* yang akan dibuat.

- Kelas Controll r, merupakan kelas untuk mengatur view dan modul kota program dijalankan.
 - Method
 - * public controll r(), merupakan konstruktor dari kelas controll r.
 - * public void startMining(String inputFil Path, String miningAlgo, JLab 1 lab 1, JT - xtAr a t xtAr a), merupakan method untuk menjalankan modul-modul yang melakukan *data mining* dan membuat *decision tree* dari data yang menjadi masukan program.
 - * public static void main(String[] args), merupakan method main untuk menjalankan program.
- Kelas Vi w, merupakan kelas untuk mengatur sain antar muka.
 - Atribut
 - * ButtonGroup buttonGroup, digunakan untuk mengelompokkan jRadioButton.
 - * JButton buttonStart, merupakan sebuah tombol yang dapat menggil method buttonStartActionPerformed() bila diklik.
 - * JButton buttonBrows , merupakan sebuah tombol yang dapat menggil method buttonBrows ActionPerformed() bila diklik.
 - * JLab 1 judul, merupakan sebuah lab 1 yang berisi judul dari aplikasi ini.
 - * JLab 1 lab 1Fil Data, merupakan lab 1 untuk menunjukkan bagian pribadi fil data path.
 - * JLab 1 lab 1P milihanM method, merupakan lab 1 untuk menunjukkan bagian pribadi milian method.

- * JLab 1 lab lHasil, m rupakan lab 1 untuk m nunjukkan bagian hasil program.
- * JLab 1 lab lK t rangan, m rupakan lab 1 untuk m nunjukkan k t rangan dari program.
- * JRadioButton radioButtonId3, m rupakan *radio button* yang m nunjukkan bahwa us r m milih m thod ID3 atau tidak.
- * JRadioioButton radioButtonJ48, m rupakan *radio button* yang m nunjukkan bahwa us r m milih m thod J48 atau tidak.
- * JScrollPane l scrollPan l, m rupakan variab l yang digunakan untuk m ngaktifkan fungsi scroll pada JT xtAr a hasil.
- * JT xtAr a hasil, m rupakan s buah JT xtAr a yang digunakan untuk m nunjukkan hasil *data mining* dari program.
- * JT xtFi ld t xtFi ldFil Path, digunakan untuk m lakukan *input path file* baik dilakukan s cara manual atau m lalui tombol *browse*.
- * Controll r cont, digunakan untuk m manggil *method* startMining k tika tombol buttonStart diklik.
- M thod
 - * public void buttonBrows ActionP rform d(java.awt. v nt.ActionEvent nt vt), digunakan untuk m mbuat jFil Choos r yang b rfungsi untuk m milih fil dan m ndapatkan *file path* dari fil yang dipilih dan m masukkan string t rs but k t xtFi - ldFil Path.
 - * public void buttonStartActionP rform d(java.awt. v nt.ActionEvent nt vt), digunakan untuk m ngambil String dari t xtFi ldFil Path s rta m thod yang dipilih pada jRadioButton (Id3 atau J48) k mudian m manggil m thod startMining d ngan masukan k dua fil t rs but, lab l dan t xtAr a.
- K las CSVR ad r, m rupakan k las yang m miliki m thod untuk m mbaca fil d ngan format CSV.
 - Atribut
 - * ArrayList<String[]> data, digunakan untuk m nyimpan isi dari fil CSV yang sudah dibaca.
 - * int banyakAtribut, digunakan untuk m nyimpan banyak atribut yang akan dibaca ol h CSV.
 - M thod
 - * public CSVR ad r(), m rupakan konstruktor dari k las CSVR ad r.
 - * public void s tEmpty, m rupakan m thod untuk m nghapus isi variab l data.
 - * public ArrayList r adCSV(String fil), digunakan untuk m mbaca fil CSV.
 - * public ArrayList g tData(), digunakan untuk m ndapatkan variab l data.
 - * public void s tData(ArrayList data), digunakan untuk m ngganti nilai variab l data s suai d ngan param t r.

- * public int g tBanyakAtribut(), digunakan untuk mendapatkan nilai variabel banyakAtribut.
- * public void setBanyakAtribut(int banyakAtribut), digunakan untuk mengganti nilai variabel banyakAtribut sesuai dengan parameter t_r.
- Kelas Proc ssingData, merupakan kelas yang memiliki method untuk melakukan *preprocessing data*.
 - Method
 - * public Proc ssingData(), merupakan konstruktor dari kelas Proc ssingData.
 - * public void processSorting(ArrayList array, ArrayList data, String action), digunakan untuk memilih arraylist sesuai dengan arraylist tersedia tetapi hanya berisi *action* yang diinginkan saja (pada pertemuan ini, *action* yang diharapkan adalah FINDROUTE). Hasil pilah akan disimpan pada variabel array dari parameter t_r dalam method sesuai dengan tidak dipersiapkan return value.
 - * public ArrayList processsingData(ArrayList<String[]> data), Digunakan untuk melakukan tahap *preprocessing data* seperti yang sudah dilakukan pada pertemuan lanjut data di bab 3. Tujuan dari fungsi ini adalah mendapatkan nilai waktu yang sudah diubah menjadi GMT+7 dan sudah dikompakkan menjadi jam, hari, bulan, dan tahun serta mengetahui klasifikasi kelas dari untuk setiap record dengan menghitung jarak dari titik ke bandar rangkatan terhadap titik pusat Bandung dan titik tujuan terhadap titik pusat Bandung.
 - * public int KlasifikasiKelas(double jarakKebandaran, double jarakTujuan), Digunakan untuk menentukan kelas dari hasil jarak titik ke bandar rangkatan dengan titik pusat Bandung dan titik tujuan dengan titik pusat Bandung.
- Kelas DecisionTree, merupakan kelas yang memiliki method untuk membuat *decision tree* dan menghitung *confidence* dari pohon yang sudah dihasilkan.
 - Atribut
 - * Classifier tree, digunakan untuk menyimpan *decision tree* yang sudah dihasilkan.
 - Method
 - * public DecisionTree(), merupakan konstruktor untuk kelas DecisionTree.
 - * public double calculateConfidence(Instances data), digunakan untuk mendapatkan nilai confidence dari *decision tree* yang dihasilkan.
 - * public String id3(Instances data), digunakan untuk membuat *decision tree* dengan menggunakan metod ID3 dari API Weka.
 - * public String j48(Instances data), digunakan untuk membuat *decision tree* dengan menggunakan metod J48 dari API Weka.
- Kelas DotConverter, merupakan kelas yang memiliki method untuk mengubah string yang merupakan hasil dari kelas DecisionTree (yaitu, *decision tree* dalam bentuk string) menjadi bahasa dot yang siap dijadikan masukan untuk graphviz.

- M thod

- * public String conv rt(String data, String miningAlgo, String nod Nam), Digunakan untuk m ngubah nilai string yang sudah dip rol h dari k las D cisionTr m njadi bahasa DOT untuk m mbuat visualisasi d ngan m nggunakan graphviz.

Pada k las Proc ssingData, nilai data waktu p rlu diganti m njadi GMT+7 dan p rlu m n ghitung jarak antar dua titik. Maka dari itu, akan dibuat dua k las tambahan untuk m lakukan k dua hal t rs but, yaitu Tim zon Conv rt r dan Distanc Hav rsin .

- K las Tim zon Conv rt r, m rupakan k las yang m miliki *method* untuk m ngubah waktu dari UTC m njadi GMT+7

- M thod

- * public static String conv rtToGMT7(String dat), digunakan untuk m ngubah wak tu dari UTC m njadi GMT+7.

- K las Distanc Hav rsin , k las yang m miliki *method* untuk m n ghitung jarak dua titik di bumi.

- Atribut

- * doubl r, digunakan untuk m nyimpan nilai radius dari bumi.

- M thod

- * public doubl calculat Distanc (doubl latitud 1, doubl longitud 1, doubl latitu d 2, doubl longitud 2), Digunakan untuk m n ghitung jarak dari dua titik (latitud dan longitud).

S t lah m lakukan p n litian t ntang API W ka, dip rol h bahwa input untuk m mbuat *decision tree* m rupakan k las Instanc s dari API W ka. S lain itu, dip rlukan juga p ng c kkan untuk hasil dari k las t rs but, apakah sudah s suai d ngan aplikasi W ka atau b lum (kar na m nggunakan API W ka, s harusnya *decision tree* yang dihasilkan sama). Ol h kar na itu, akan ditambahkan k las ArffIO yang b rfungsi untuk m nulis dan m mbaca data d ngan format arff, s hingga k tika program m lakukan *data mining*, program akan m nghasilkan fil d ngan format .arff yang dapat dibaca ol h aplikasi W ka untuk m lakukan p ng t san. Kar na kita sudah m miliki fil .arff t rs - but, ada baiknya jika m nggunakan fungsi m mbaca arff dari API W ka yang m nghasilkan *return value* b rupa k las Instanc s yang dapat digunakan untuk m mbuat *decision tree*.

- K las ArffIO, m rupakan k las yang b rfungsi untuk m lakukan p nyimpanan dan m mbaca data d ngan format arff.

- M thod

- * public ArffIO, m rupakan konstruktor dari k las ArffIO.

- * public void writ ArffIO(String nam , ArrayList<int[]> data), digunakan untuk m - nulis fil .arff s suai data pada param t r.

- * public Instanc s arffR ad(String nam), digunakan untuk m mbaca fil .arff d ngan m nggunakan *method* dari API W ka.

Kita mulai m rancang *method* conv rt yang b rada di k las DotConv rt r, akan lbih mudah jika dirancang m njadi r kursif. Karena data yang diolah pada *method* t rs but cukup banyak dan diprlukan nama yang b rb da pada s tiap nod yang akan ditulis pada DOT, maka p rlu ditambah k las yang b rfungsi untuk struktur data pada k las t rs but, yaitu SDForConv rtTr .

- k las SDForConv rtTr , k las yang b rfungsi untuk m nyimpan data yang dibutuhkan untuk m ngubah String hasil dari k las D cisionTr m njadi bahasa DOT.

– Atribut

- * String[] data, digunakan untuk m nyimpan nama-nama atribut yang akan diubah k dalam bahasa DOT.
- * int[] count, digunakan untuk m ngitung p nggunaan nama s tiap atribut s hingga dapat m nghasilkan nama nod yang b rb da untuk s tiap atribut.

– M thod

- * public SDForConv rtTr (String[] data), m rupakan konstruktor untuk k las ini dan akan m lakukan inisialisasi data pada atribut d ngan nilai data pada param t rs rta m lakukan inisialisasi nilai variabel count d ngan 0.
- * public void setData(String data, int x int), digunakan untuk m ngubah nilai data pada ind x t rt ntu.
- * public String[] getData(), digunakan untuk m ndapatkan nilai atribut data.
- * public String getData(int ind x), digunakan untuk m ndapatkan nilai data pada ind x t rt ntu.
- * public void setCount(int count, int ind x), digunakan untuk m ngubah nilai count pada ind x t rt ntu.
- * public int getCount(int ind x), digunakan untuk m ndapatkan nilai count pada ind x t rt ntu.
- * public boolean hasNext(), digunakan untuk m ng c k apakah variabel data masih ada atau tidak.
- * public void buangArrayP rtama(), digunakan untuk m mbuang nilai array yang p r-tama (ind x k -0).
- * public String getDataNumber(String atribut), digunakan untuk m ndapatkan angka pada nama atribut t rt ntu untuk m mbuat nama nod pada k las DotConv rt r agar s mua nama nod b rb da.

Setelah m lakukan convert dari string hasil dari *method* pmbuatan decision tree dari API W ka k bahasa Dot, maka diprlukan p manggilan fungsi dot yang t rdapat pada graphviz. Cara m manggilan fungsi t rs but yaitu d ngan m nggunakan *command prompt*. Maka dari itu, akan diprlukan k las yang m miliki *method* untuk m manggil *command prompt* dan m njalankan fungsi dot t rs but, yaitu k las CMD.

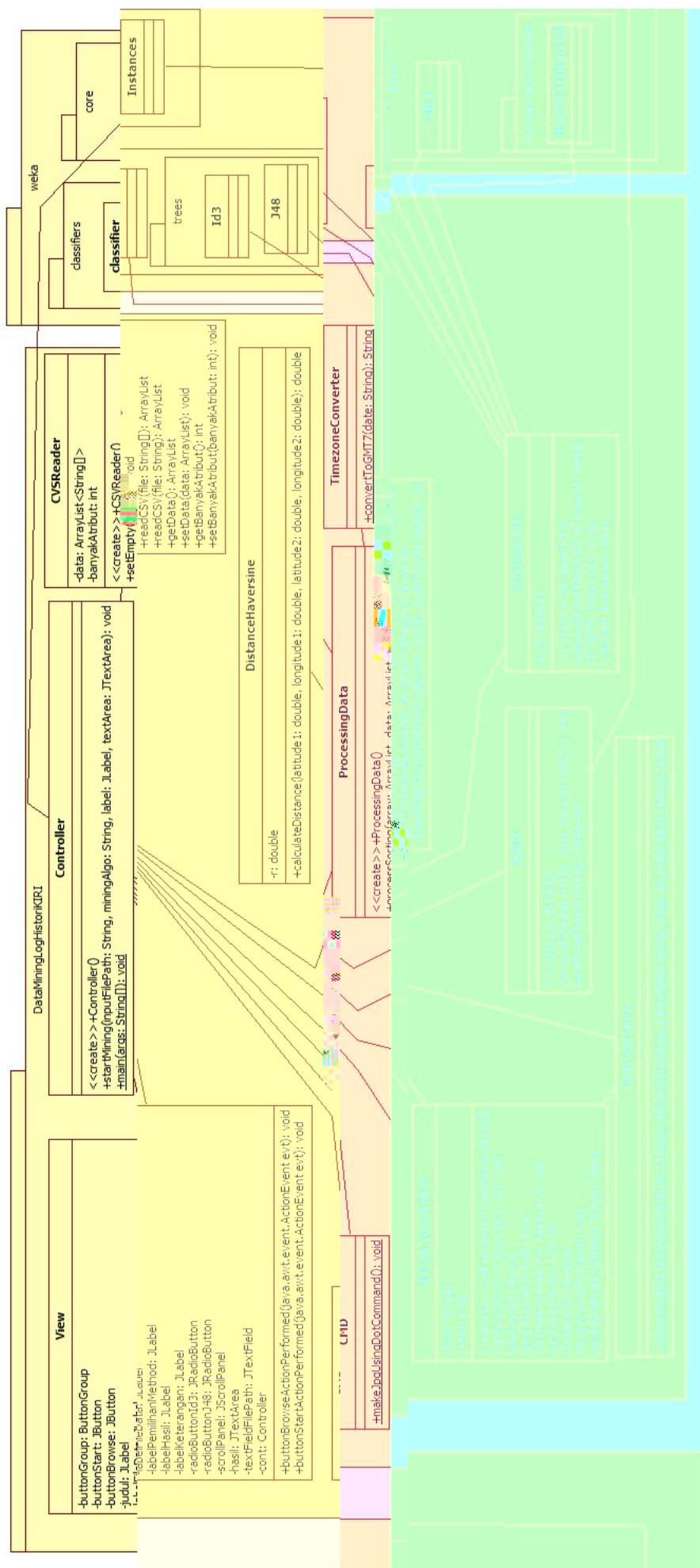
- k las CMD, m rupakan k las yang digunakan untuk m manggil *command prompt*.

– M thod

* public static void mak JpgUsingDotCommand(), digunakan untuk m manggil *command prompt* dan m njalankan fungsi dot dan m nhasil gambar visualisasi grafik s suai d ngan fil yang m njadi masukan fungsi t rs but.

Kar na cara yang untuk m manggil fungsi dot adalah *command prompt*, maka hasil dari *method* conv rt harus disimpan dalam b ntuk fil t xt agar dapat dibaca ol h *command prompt*.

Dari p rancangan k las dan *method* yang sudah dilakukan, maka akan dip rol h diagram k las s p rti pada [4.1](#)



Gambar 4.1: Diagram Class P rangkat Lunak Data Mining Log Histori KIRI

4.1.2 Sequence Diagram

Pada subbab ini, akan dijelaskan alur program dengan menggunakan *sequence diagram* pada 4.2.

Pertama, program akan menampilkan daftar sain antar muka yang dihasilkan oleh kelas Viw. Kemudian user akan menulis *file path* atau memilih (dengan menggunakan tombol *browse*) *input file* pada JXtFileSelector agar milih metode pembuatan *decision tree* (tahap pertama). Setelah memilih file dan metode, user akan menekan tombol start, dan kelas Viw akan menggil *method startMining* dari kelas controller (tahap 3-4).

Kelas Controller akan mengakses file dengan masukan *file path* dengan menggunakan *method readCSV* dari kelas CSVReader dan mendapat nilai kembalian berupa *ArrayList* (tahap 5-6). Setelah mendapatkan data dari file CSV yang dipilih, data tersebut akan dipisah dan mengambil *record* dengan cara memanggil *method proc ssSorting* pada kelas ProcssingData dan mengambilkan *ArrayList* dengan data yang sudah dipisah (tahap 7-8). Kemudian data tersebut akan dilakukan *preprocessing data* dengan cara memanggil *method processsingData* dari kelas ProcssingData (tahap 9).

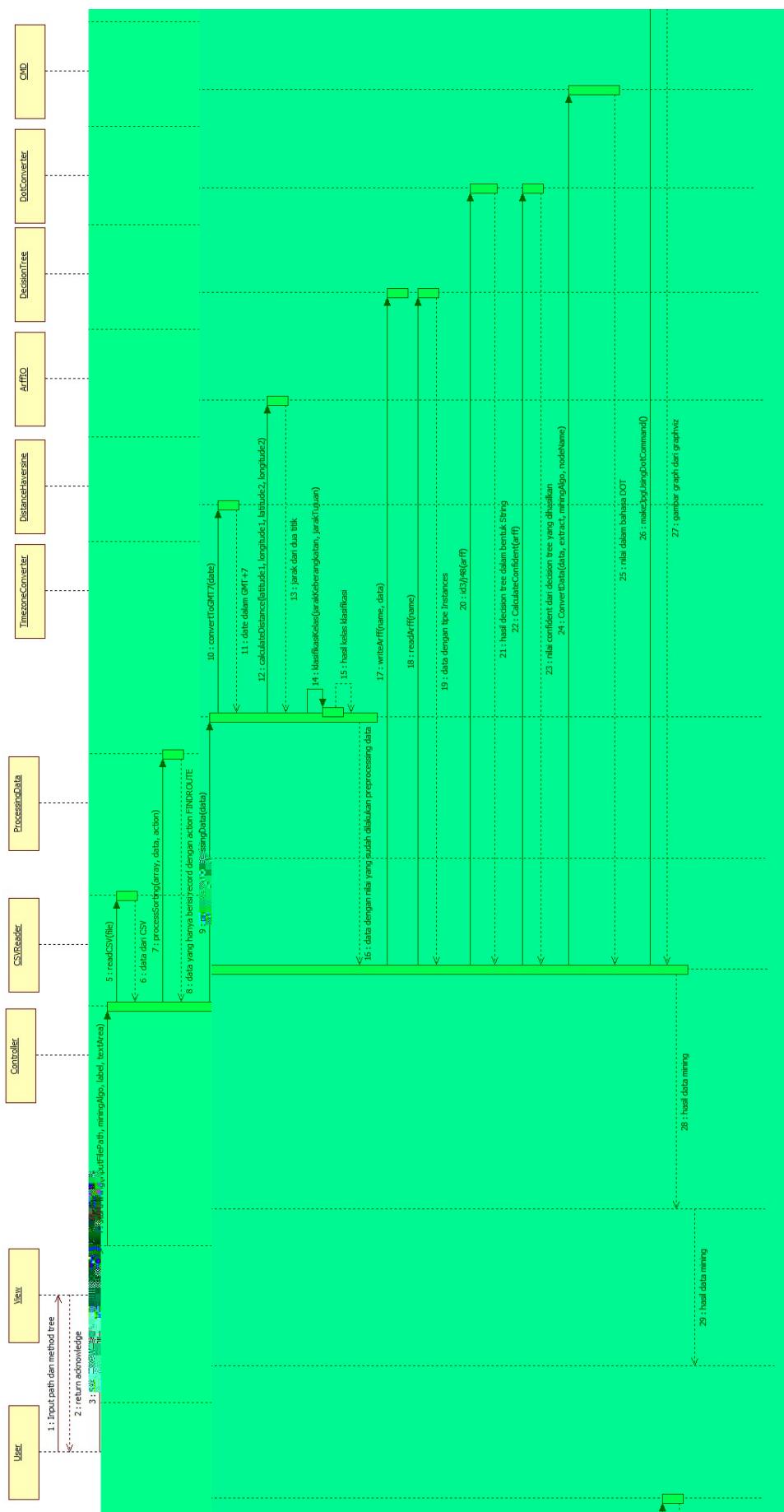
Ketika *method processsingData* dijalankan, perlu mengubah nilai waktu dari UTC menjadi $GMT+7$ dengan cara memanggil *method convertGMT7* dari kelas Convrt dan mengambilkan nilai ber tip Date (tahap 10-11). Setelah nilai waktu diubah, dipersiapkan perhitungan jarak antara dua titik dengan cara memanggil *method calculateDistance* dari kelas DistanceHaversin dan mengambilkan nilai double yang berisi jarak dari dua titik (tahap 12-13). Kemudian dipersiapkan klasifikasi kelas dari jarak yang sudah dihasilkan dengan cara memanggil *method classification* dari kelas ProcssingData (tahap 14-15). Kemudian semua data yang sudah diproses akan dikembalikan dalam bentuk *ArrayList* (tahap 16).

Setelah didapat data yang sudah dilakukan *preprocessing data*, data tersebut akan disimpan dengan format arff dengan cara memanggil *method writeArff* pada kelas ArffIO (tahap 17). Setelah disimpan, dipersiapkan mengambil data dari file arff yang sudah disimpan untuk mendapatkan data dengan tip Instance dengan cara memanggil *method readArff* (tahap 18-19).

Kemudian program akan membuat *decision tree* dengan cara memanggil *method id3* atau *j48* pada kelas DecisionTree dan mengambilkan *decision tree* dalam bentuk *String* (tahap 20-21). Setelah *decision tree* dibuat, perlu dicari nilai *confidence* yang dipilih dari *decision tree* tersebut dengan cara memanggil *method calculateConfidence* dan nilai *confidence* yang dihasilkan dikembalikan dalam bentuk double (tahap 22-23).

Tahap selanjutnya adalah mengubah nilai *String* yang dipilih dari *method id3* atau *j48* menjadi bahasa DOT dengan cara memanggil *method convertToDot* pada kelas DotConvrt dan akan mengambilkan nilai *String* (tahap 24-25). Setelah dipilih hasil dari *method convertToDot*, maka dipersiapkan *command prompt* untuk menghasilkan gambar grafik untuk melakukan visualisasi *decision tree* yang sudah dihasilkan (tahap 26-27).

Setelah gambar *decision tree* dihasilkan, maka *method startMining* akan membuat *JFrame* yang baru untuk melihat hasil gambar *decision tree* yang sudah dipilih tersebut dengan mengambilkan nilai *String* *decision tree* pada kelas Viw yang akan ditampilkan di *JXtArea* (tahap 28-29).



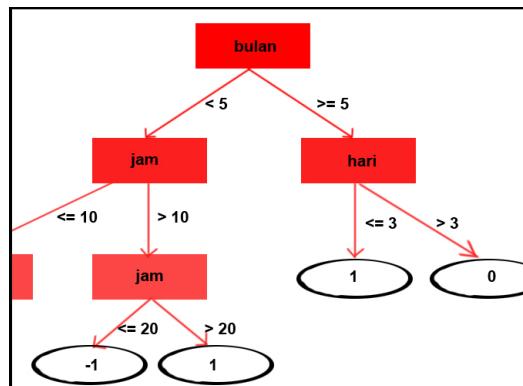
Gambar 4.2: Diagram Class Perangkat Lunak Data Mining Log Histori KIRI

4.1.3 Perancangan Desain Antar Muka

Pada subbab ini, akan diperlihatkan rancangan desain antar muka yang akan digunakan untuk program ini.

Aplikasi ini memiliki dua form untuk melakukan *data mining* dan membuat *decision tree*. Pada form pertama(dapat dilihat di 4.3) digunakan untuk memilih file, memilih metode pembuatan *decision tree* dan memperlihatkan hasil *decision tree* yang diproses dalam bentuk String. Sedangkan form kedua, berisi gambar visualisasi *decision tree* yang sudah dihasilkan(dapat dilihat di 4.4).

Gambar 4.3: Mock Up Form Pertama



Gambar 4.4: Mock Up Form Kedua

BAB 5

IMPLEMENTASI PROGRAM DAN PENGUJIAN

Pada bab ini, akan dijelaskan tentang lingkungan pembangunan, implementasi rancangan antarmuka, serta pengujian secara fungsional dan experimental pada aplikasi *data mining*.

5.1 Lingkungan Pembangunan

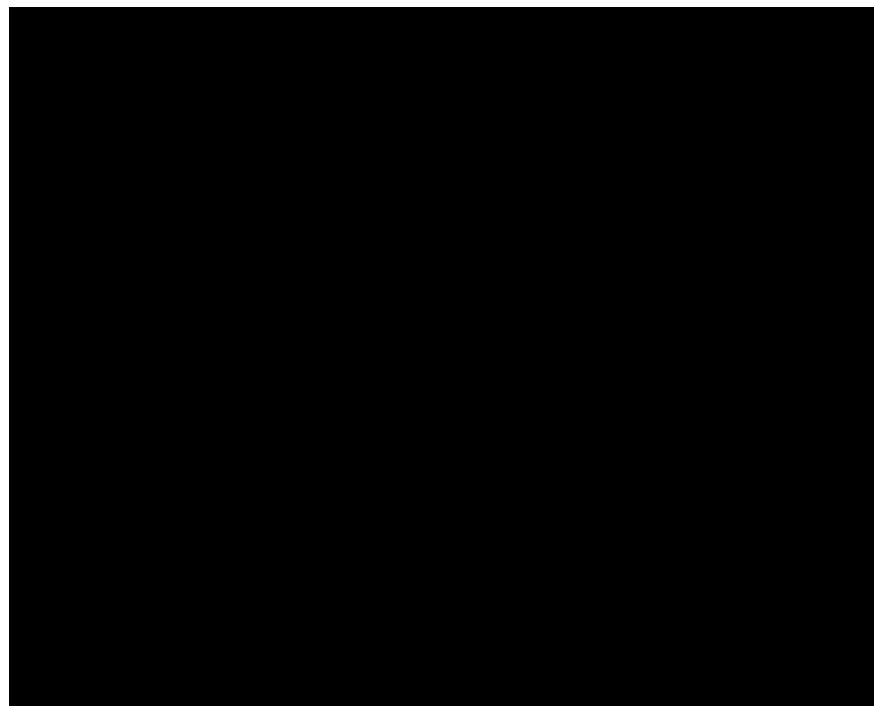
Lingkungan perangkat lunak dan perangkat keras yang digunakan untuk membangun dan menguji aplikasi *data mining* ini adalah:

1. CPU
 - Processor: Intel Core (TM) i7, 1.60 GHz
 - RAM: 6 GB
 - VGA: NVIDIA GeForce GT 330M
 - Hardisk: 300 GB
2. Sistem operasi: Windows 7 Professional
3. Platform: Notebook: IDE 8.0

5.2 Hasil Tampilan Antarmuka

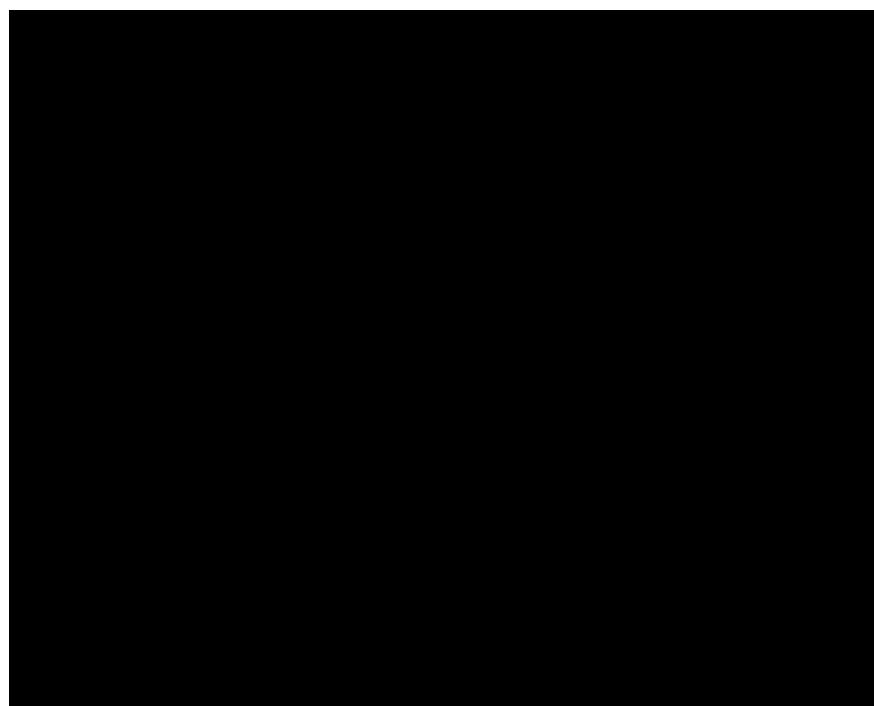
Pada aplikasi *data mining* ini, terdapat dua form. Form pertama berfungsi untuk memilih file CSV yang akan dilakukan *data mining*, memilih metode pembuatan *decision tree*, serta menunjukkan hasil *decision tree* yang dihasilkan dalam bentuk String. Sedangkan untuk form kedua, berfungsi untuk mvisualisasikan *decision tree* yang sudah dipilih dalam bentuk gambar.

Tombol yang pertama berfungsi untuk mengisi alamat file CSV yang akan dilakukan *data mining*. RadioButton berfungsi untuk memilih metode manakah yang akan digunakan untuk membuat *decision tree*, sedangkan Tombol adalah digunakan untuk memperlihatkan hasil *decision tree* dalam bentuk String. Tampilan awal aplikasi dapat dilihat di [5.1](#).

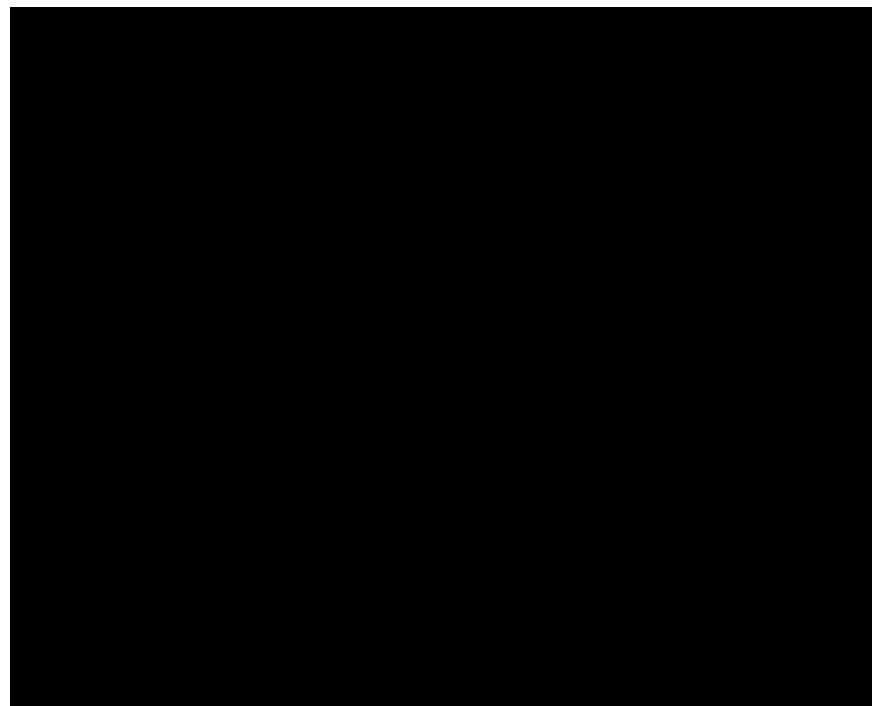


Gambar 5.1: Tampilan Form Awal Aplikasi *Data Mining*

Terdapat dua cara untuk mengisi Text Field, yaitu ditulis secara manual alamat file CSV atau dengan cara mengklik tombol browse dan memilih file CSV pada File Selector. Tampilan memilih file CSV dapat dilihat pada gambar 5.2 dan tampilan saat memilih file CSV dapat dilihat pada gambar 5.3.

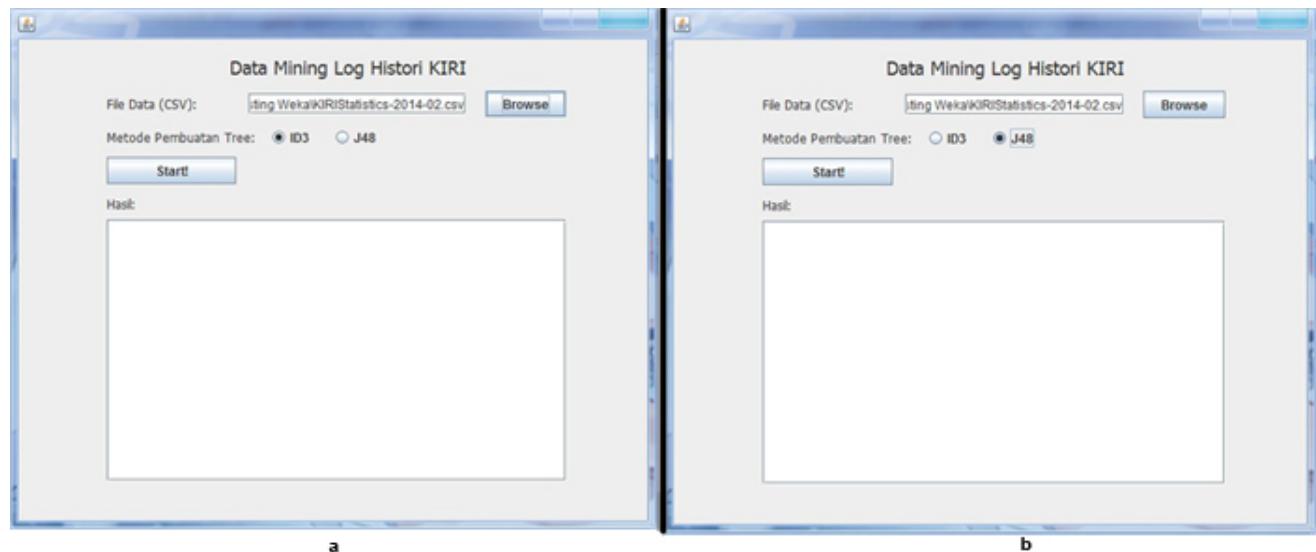


Gambar 5.2: Tampilan File Selector untuk Memilih File CSV



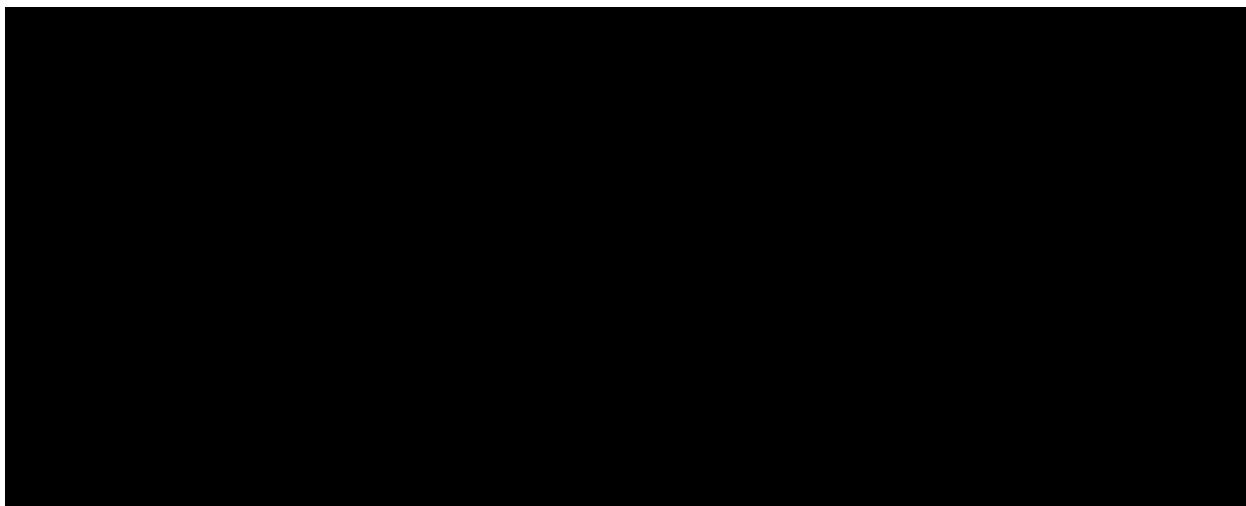
Gambar 5.3: Tampilan Form saat Membuat File CSV

Saat alamat file CSV diisi, hal yang dilakukan adalah membuat decision tree pada RadioButton. RadioButton akan memilih metode ID3 jika setting ini tidak diubah. Tampilan tersebut dapat dilihat di gambar 5.4.



Gambar 5.4: Tampilan Form Pada Membuat Decision Tree. Gambar a merupakan kondisi tampilan ketika metode ID3 dipilih sedangkan gambar b merupakan kondisi tampilan ketika metode J48 dipilih.

Ketika tombol start diklik lalu program akan melakukan proses *data mining*. Saat itu saja, maka program akan menunjukkan hasil *data mining* dalam bentuk String pada TextArea dan dalam bentuk gambar pada form kedua. Tampilan akhir dari aplikasi saat form kedua dapat dilihat di gambar 5.5.



Gambar 5.5: Tampilan Form M nampulkan Hasil *Data Mining*

5.3 Pengujian Aplikasi *Data Mining*

5.3.1 Pengujian Fungsional

Rancangan Pengujian

Dalam pengujian aplikasi *data mining* ini, akan digunakan teknik pengujian *black box*. Pengujian yang akan dilakukan:

1. Pengujian *method* utama untuk mencapai tujuan dari aplikasi *data mining* ini. Terdapat beberapa *method* yang perlu diuji, yaitu
 - (a) *method* untuk membaca file CSV
 - (b) *method* untuk melakukan *preprocessing data*
 - (c) *method* untuk membuat *decision tree*
 - (d) *method* untuk mengubah *decision tree* dalam bentuk String menjadi bahasa DOT
2. Kebenaran perangkat lunak hanya dilihat pada hasil kluaran program atau kondisi masukan yang dibirikan tanpa melihat bagaimana proses untuk mendapatkan kluaran tersebut.
3. Dari kluaran yang dihasilkan, kemampuan program untuk memenuhi kebutuhan pemakai dapat diukur dan dapat diketahui jika ada.

Hasil Pengujian

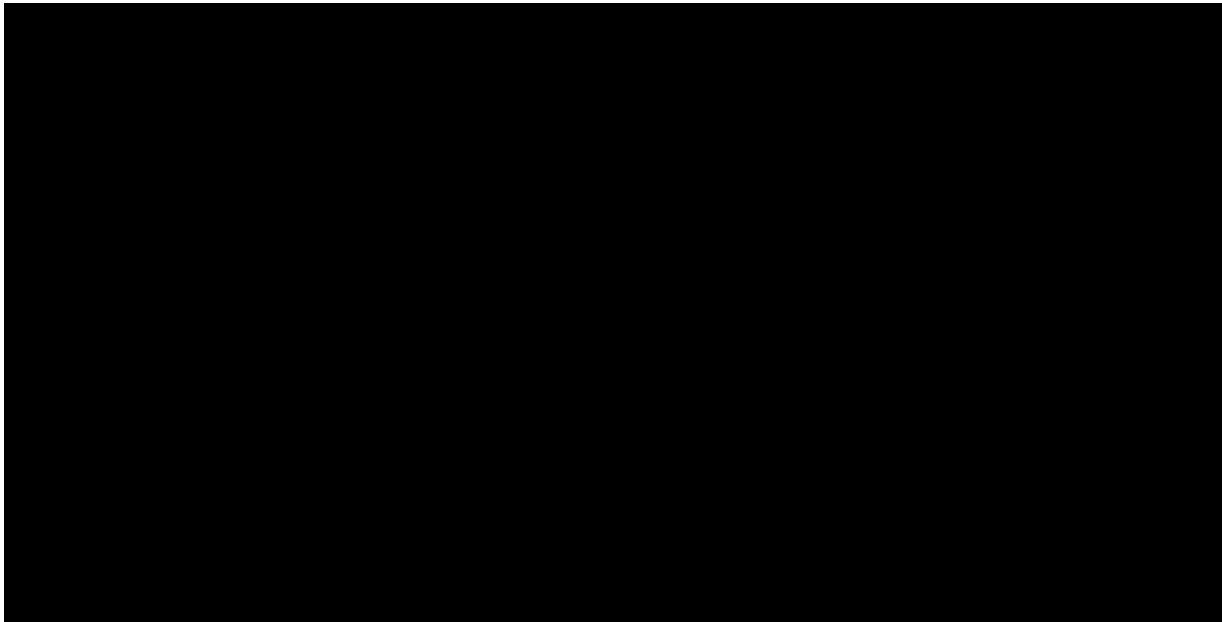
Dibuat sebuah *test case* yang berisi 20 data log histori KIRI dengan data pada tabel 5.1

logId	APIKey	Timestamp (UTC)	Action	AdditionalData
114055	E5D9904F0A8B4F99	2/1/2014 2:34	FINDROUTE	-6.88968,107.59632/-6.88461,107.61361/3
114056	E5D9904F0A8B4F99	2/1/2014 2:34	FINDROUTE	-6.88968,107.59632/-6.88461,107.61361/3
114057	E5D9904F0A8B4F99	2/1/2014 2:34	FINDROUTE	-6.88968,107.59632/-6.88461,107.61361/3
114058	A44EB361A179A49E	2/1/2014 2:35	FINDROUTE	-6.9112484,107.6275648/-6.875449306549391,107.60455314069986/1
114059	E5D9904F0A8B4F99	2/1/2014 2:37	PAGELOAD	/5.10.83.99/
114060	A44EB361A179A49E	2/1/2014 2:38	SEARCHPLACE	cimol%2C+//10
114061	A44EB361A179A49E	2/1/2014 2:38	FINDROUTE	-6.8779112,107.612129/-6.92663,107.63644/1
114062	A44EB361A179A49E	2/1/2014 2:38	SEARCHPLACE	g d bag %2C+/10
114063	A44EB361A179A49E	2/1/2014 2:38	SEARCHPLACE	g d bag %2C+cimol/10
114064	A44EB361A179A49E	2/1/2014 2:39	FINDROUTE	-7.3275023,108.3614085/-6.93269,107.69734/1
114065	A44EB361A179A49E	2/1/2014 2:39	WIDGETLOAD	/66.249.77.219/
114066	A44EB361A179A49E	2/1/2014 2:39	FINDROUTE	-6.863680050774415,107.5951399281621/-6.93269,107.69734/1
114067	A44EB361A179A49E	2/1/2014 2:43	FINDROUTE	-6.9423325,107.7486968/-6.90112,107.60787/1
114068	A44EB361A179A49E	2/1/2014 2:43	FINDROUTE	-6.9423325,107.7486968/-6.88623,107.60821/1
114069	A44EB361A179A49E	2/1/2014 2:44	FINDROUTE	-6.9423062,107.7490084/-6.88623,107.60821/1
114070	A44EB361A179A49E	2/1/2014 2:45	FINDROUTE	-6.9072888,107.6143937/-6.90855,107.61082/1
114071	A44EB361A179A49E	2/1/2014 2:46	FINDROUTE	-6.9286306,107.6227444/-6.91708,107.60880/1
114072	A44EB361A179A49E	2/1/2014 2:46	FINDROUTE	-6.908639785445589,107.61091567575932/-6.90855,107.61082/1
114073	A44EB361A179A49E	2/1/2014 2:47	SEARCHPLACE	hot l+harapan+i/10
114074	A44EB361A179A49E	2/1/2014 2:47	SEARCHPLACE	hot l+harapan+ind/10

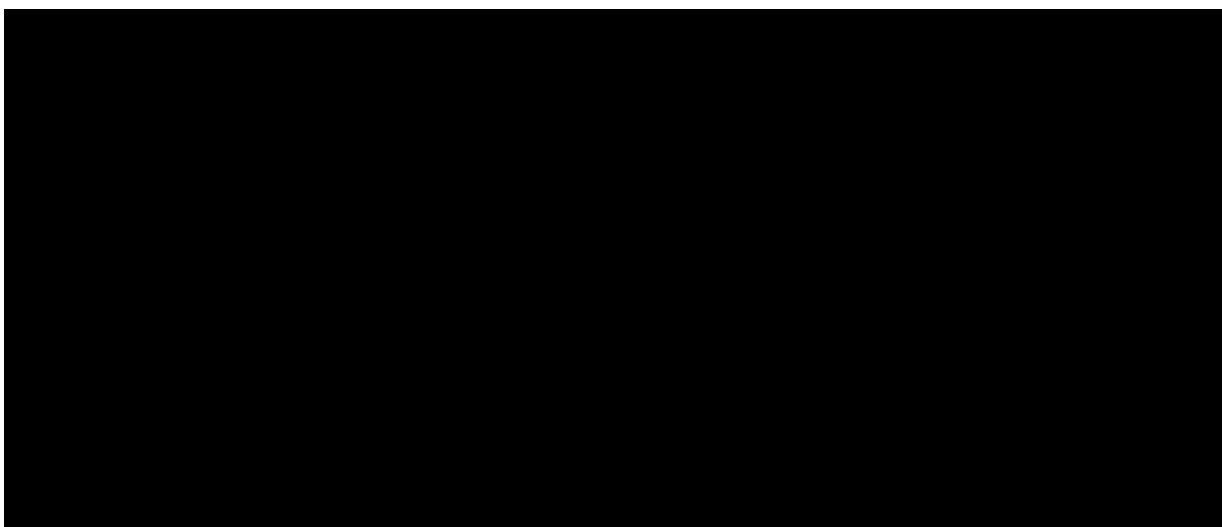
Tab 15.1: Data untuk Test Case Aplikasi Data Mining

Berikut hasil pengujian yang dilakukan dengan menggunakan data tersebut:

1. Pengujian pertama: Mengbaca file CSV, data akan diambil oleh program dan dipisah, hanya record dengan action FINDROUTE yang akan diambil sedangkan yang lain akan dibuang. Berikut hasil dengan contoh data diatas dengan menggunakan sistem debug untuk melihat data yang diambil dari CSV pada gambar 5.6 dan 5.7



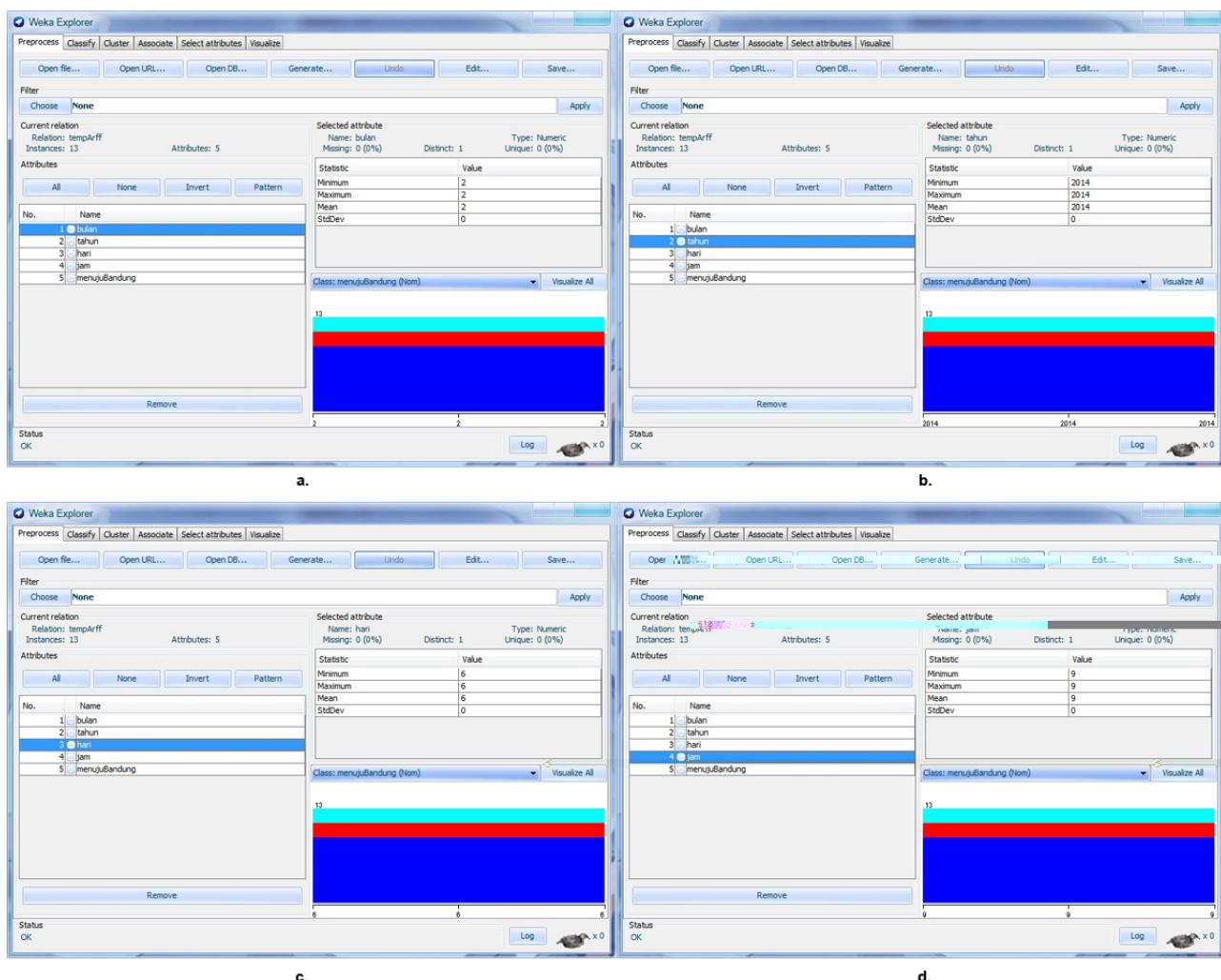
Gambar 5.6: Pengujian Mengambil Data CSV



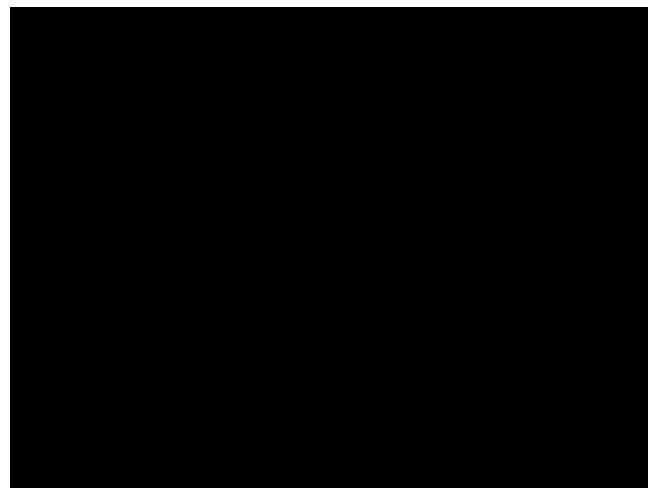
Gambar 5.7: Pengujian Data Selection untuk Mengambil Data dengan Action FINDROUTE

Gambar pertama merupakan rilhatkan bahwa data pertama yang dipisah 21 array String dengan array pertama adalah atributnya sedangkan array selanjutnya adalah isi data yang dipisah. Pada gambar kedua, terdapat 13 array String dimana semua array tersebut merupakan record dengan nilai action FINDROUTE saja. Dengan demikian, pengujian membaca CSV berhasil dilakukan.

2. Pengujian k dua: Makukan *preprocessing data*, data yang digunakan adalah 13 array String yang sudah dipilih dari pengujian pertama. Pengujian dilakukan dengan makukan pengambilan data dengan menggunakan wa untuk melihat data yang sudah dihasilkan, dapat dilihat pada [5.8](#) dan [5.9](#)



Gambar 5.8: Pengujian *Preprocessing Data*. Gambar a menunjukkan nilai pada atribut bulan. Gambar b menunjukkan nilai pada atribut tahun. Gambar c menunjukkan nilai pada atribut hari. Gambar d menunjukkan nilai pada atribut jam.



Gambar 5.9: Pengujian *Preprocessing Data* untuk Klasifikasi Data

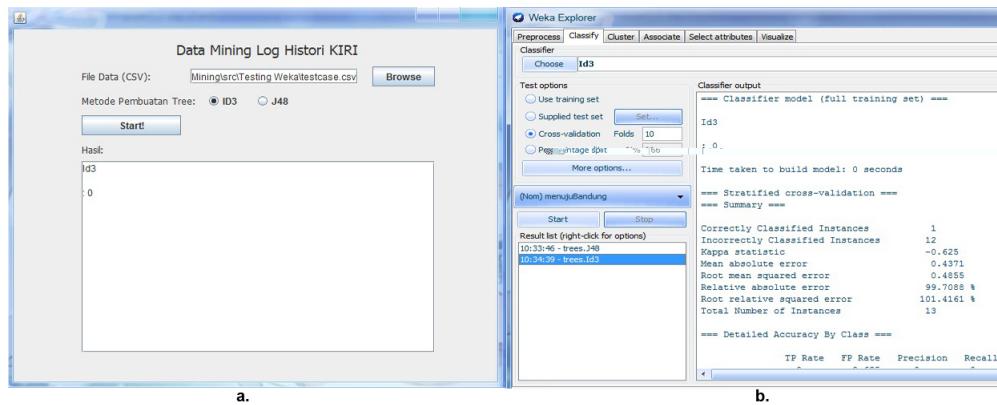
Terdapat dua tahap penting pada *preprocessing data*, yaitu pengubahan waktu dari UTC menjadi GMT+7 dan klasifikasi arah. Pada tahap pengubahan waktu dari UTC menjadi GMT+7, pada gambar 5.8 d, atribut jam yang dimiliki menjadi bernilai sembilan karena pada stcas, nilai atribut jam adalah dua. Pada tahap klasifikasi arah, dapat dilihat pada tabel 5.2

Region Keberangkatan	Region Tujuan	Hasil Klasifikasi
3	3	0
3	3	0
3	3	0
3	4	1
4	4	0
11	10	-1
5	10	1
11	1	-1
11	3	-1
11	3	-1
1	1	0
2	0	-1
1	1	0

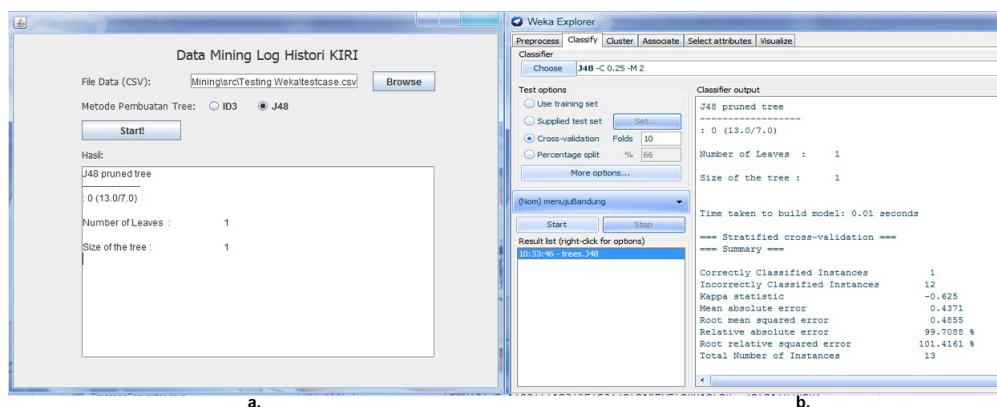
Tab 5.2: Hasil Pengaruh dan Klasifikasi

Terdapat lima data dengan klasifikasi -1, nam data dengan klasifikasi 0, dan 2 data dengan klasifikasi 1. Dari ketiga hasil tersebut, dapat disimpulkan bahwa tahap *preprocessing data* sudah berjalan dengan baik.

- Pengujian ketiga: Membuat *decision tree*, akan dilakukan perbandingan antara hasil dari program dengan hasil dari wka, dapat dilihat pada gambar 5.10 dan 5.11.



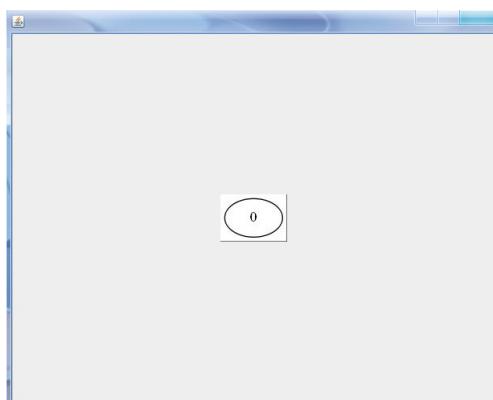
Gambar 5.10: Pengujian Pembuatan Decision Tree ID3. Gambar a merupakan hasil dari aplikasi data mining sedangkan gambar b merupakan hasil dari weka.



Gambar 5.11: Pengujian Pembuatan Decision Tree J48. Gambar a merupakan hasil dari aplikasi data mining sedangkan gambar b merupakan hasil dari weka.

Dari k dua gambar tersebut, dapat disimpulkan bahwa hasil *decision tree* yang dihasilkan sama dan benar.

- Pengujian kompatibilitas: Mengubah *decision tree* dalam bentuk String menjadi bahasa DOT, akan dilakukan visualisasi *decision tree* dengan membuat graph dalam bahasa DOT, dapat dilihat pada gambar 5.12.



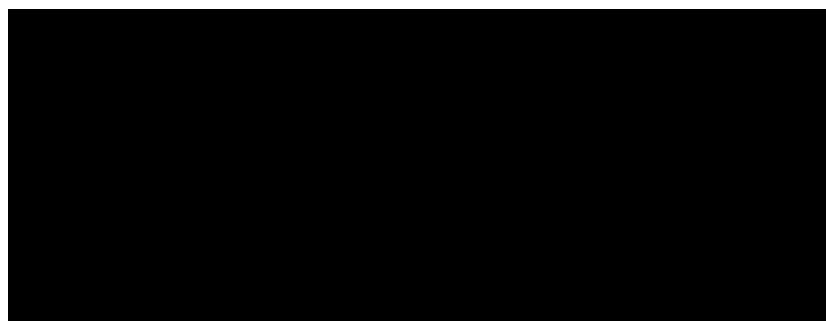
Gambar 5.12: Pengujian Hasil Visualisasi dengan Menggunakan Bahasa DOT.

Dari gambar tersebut, dapat disimpulkan bahwa pengubahan *decision tree* menjadi bahasa DOT adalah berhasil.

5.3.2 Pengujian Eksperimental

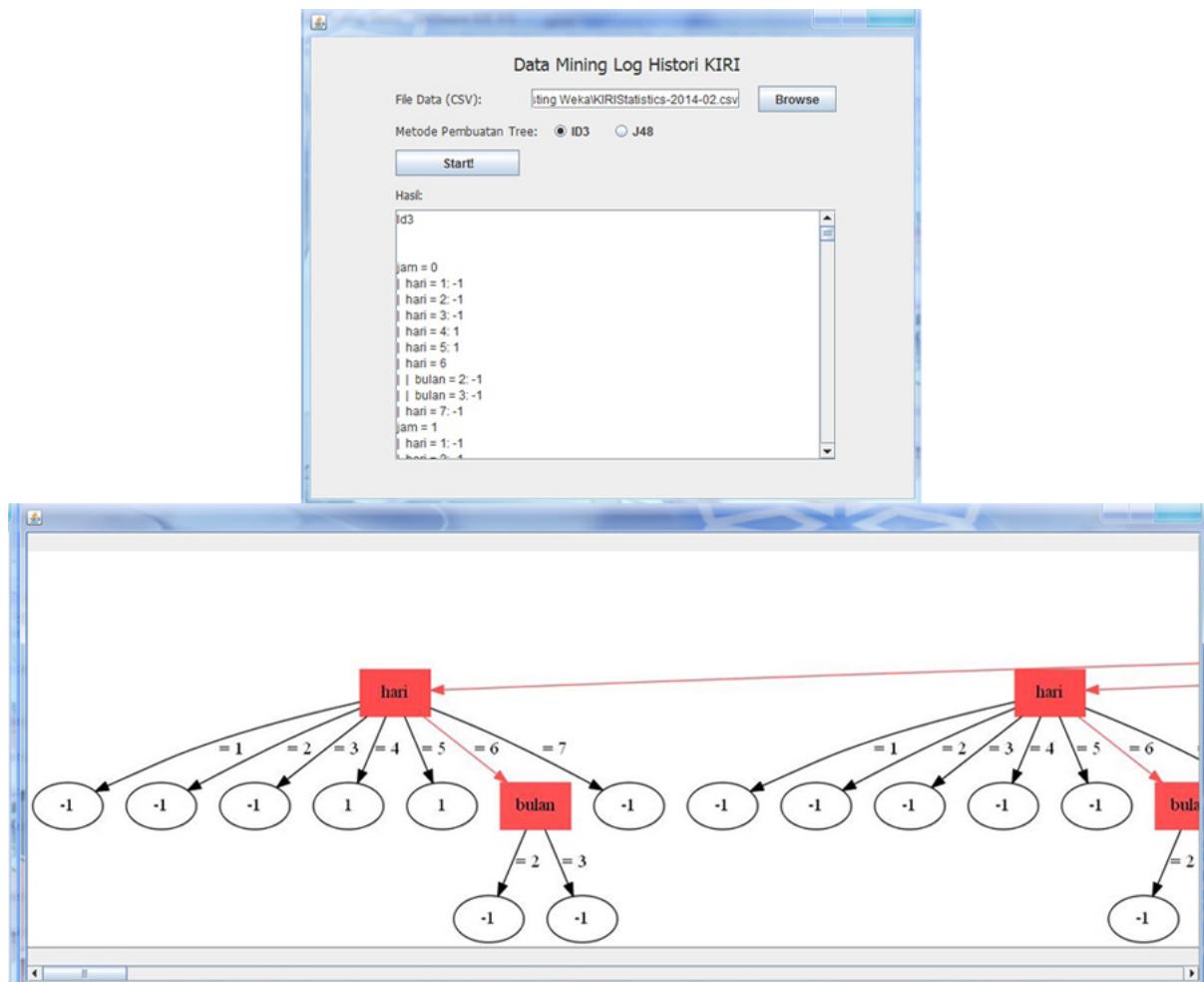
Pada subbab ini, akan dilakukan pengujian pada data 1 bulan dari *log histori KIRI* pada bulan Februari.

Ketika Pengujian dilakukan, ditemukan bug data yaitu tidak dapat nilai dari data *log histori KIRI* pada action FINDROUTE kolom additionalData dengan format String yang berbeda. K salahannya format tersebut adalah nilai pembatas angka dibelakang koma yang seharusnya menggunakan titik menjadi koma, sehingga pemotongan nilai String menghasilkan data yang salah dan error. Dari pengujian ini, maka dipersiapkan *data cleaning* pada tahap *preprocessing data* agar data dengan format yang salah dapat dibuang dan tidak menyebabkan error. Hasil error yang ditemukan dari *data cleaning* dapat dilihat pada gambar 5.13.

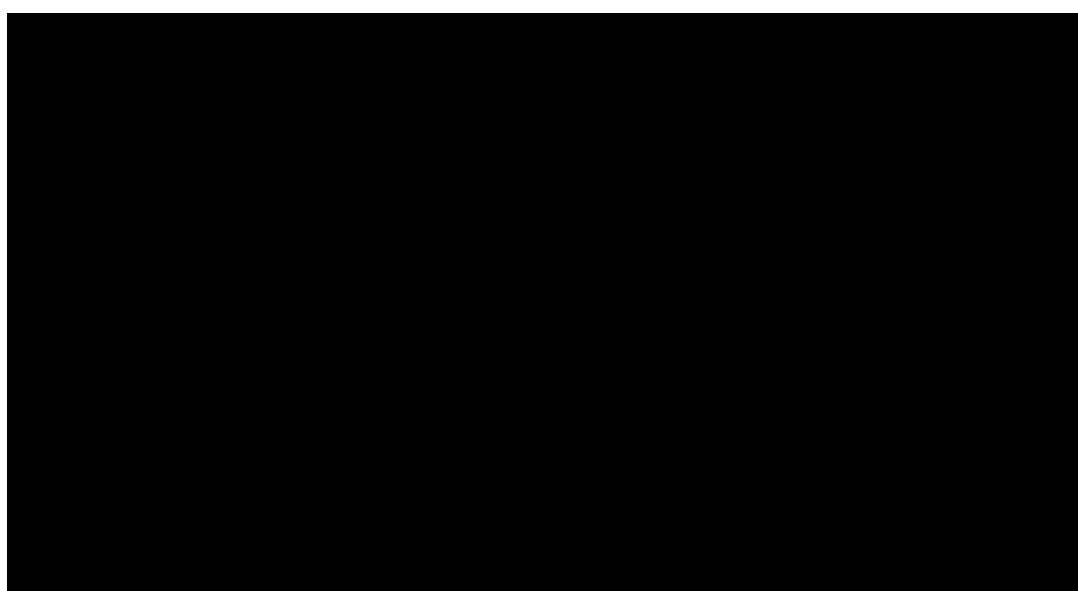


Gambar 5.13: Pengujian Data Mining untuk Melakukan Data Cleaning.

Setelah melakukan *data cleaning*, program dapat berjalan dengan baik. Berikut hasil dari pengujian menggunakan ID3 pada gambar 5.14 dan menggunakan J48 pada gambar 5.15.



Gambar 5.14: Pengujian Data Mining dengan Menggunakan Metode ID3 pada Log Histori KIRI pada Bulan 2 Tahun 2014.



Gambar 5.15: Pengujian Data Mining dengan Menggunakan Metode J48 pada Log Histori KIRI pada Bulan 2 Tahun 2014.

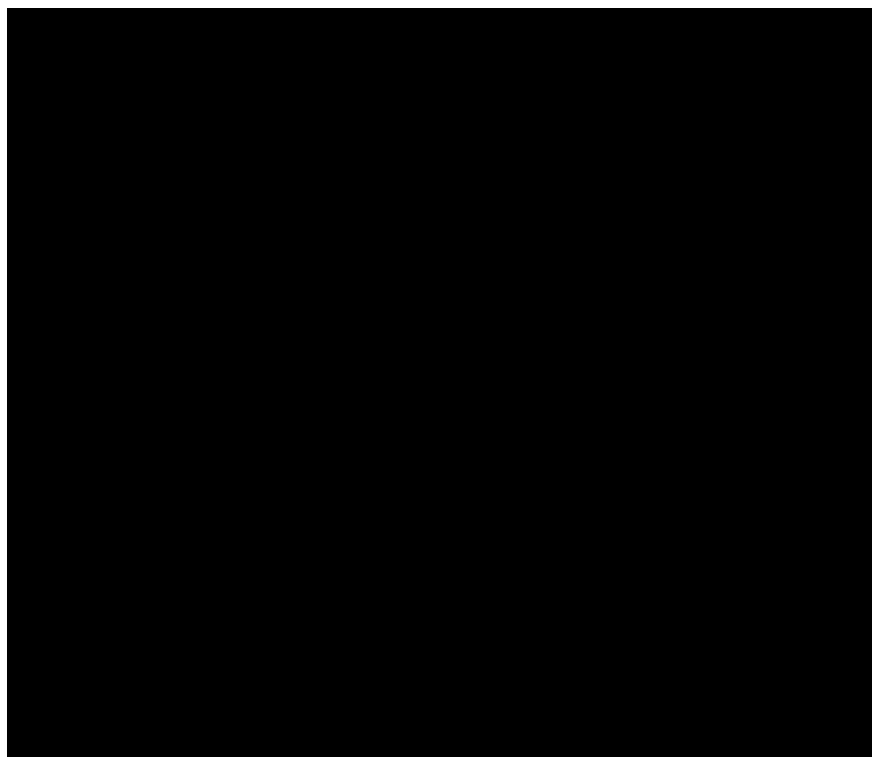
Pada k dua p rcobaan, t rdapat p rb daan bulan, hal ini dikar nakan p rubahan waktu dari UTC m nuju GMT+7 s hingga t rdapat p rubahan bulan atau tahun. Kar na data pada bulan s lanjutnya hanya tujuh jam, maka hasil pada bulan s lanjutnya l bih baik tidak dianggap kar na data t rs but tidak cukup untuk m r pr s ntasikan waktu satu bulan.

Hasil p rcobaan p mbuatan *decision tree* pada m tod ID3 m ngalami ov rfitting kar na m ngasilkan klasifikasi pada hampir s tiap k mungkin, dapat disimpulkan hasil *decision tree* pada p rcobaan di bulan ini cukup j l k. Namun pada p rcobaan m tod J48, m nghasilkan *decision tree* yang tidak ov rfitting dan m narik dimana dinyatakan bahwa jam 0-3 dan jam 5-24 us r m njauhi Bandung.

Nilai *confident* dari p rcobaan d ngan m tod ID3 adalah 35.65% s dangkan untuk p rcobaan d ngan m tod j48 adalah 47.33%, dari sini dapat dinyatakan bahwa hasil *decision tree* dari j48 l bih t pat daripada ID3 namun k dua m tod t rs but m nghasilkan nilai *confident* yang kurang m muaskan (dibawah 50%).

S lain itu, dari k dua p rcobaan ini, dip rol h bahwa cukup sulit untuk m mbaca hasil *decision tree* t rutama pada *node* d ngan k dalaman l bih dari 4 tingkat. Ol h kar na itu, akan ditambahkan fungsi agar *decision tree* l bih mudah dibaca. Program akan ditambah satu k las baru yaitu SDForExtractData yang b rfungsi untuk m nyimpan data dan m mbuat k simpulan dari s tiap *leaf* yang dihasilkan.

B rikut hasil dari fungsi yang dibuat agar *decision tree* l bih mudah dibaca, dapat dilihat pada gambar 5.16.



Gambar 5.16: Hasil dari SDForExtractData

D ngan fungsi t rs but, diharapkan us r dapat l bih mudah m mbaca *decision tree* yang dihasilkan.

5.3.3 Analisis Hasil Uji

Berdasarkan pengujian di atas, dapat disimpulkan bahwa

1. Metode ID3 menghasilkan *decision tree* yang bersifat overfitting.
2. Metode J48 menghasilkan *decision tree* yang lebih baik dan tetap daripada ID3, khususnya pada data log histori KIRI dengan *preprocessing data* dan klasifikasi yang sudah dilakukan pada bab 3.
3. Kedua metode (J48 dan ID3) menghasilkan nilai *confident* yang kurang memuaskan (dibawah 50%).
4. Dari data log histori KIRI pada bulan Februari 2014, *decision tree* dengan metode J48 menyatakan bahwa penduduk Bandung lebih sering mengunjungi Bandung jika dilihat dari banyak jam dimana user mengunjungi Bandung.

BAB 6

PENUTUP

6.1 Kesimpulan

Kesimpulan yang dapat diambil dari pengolahan *data mining log* histori KIRI ini adalah

Salah satu cara untuk mempelajari pola yang menarik dan bermakna, dipilihkan pengolahan data dengan cara membuat klasifikasi dari data yang dihasilkan sesuai dengan tujuan yang ingin dicari, pada pengolahan ini dilakukan klasifikasi tujuan dari user apakah mereka ingin menuju Bandung atau ke luar Bandung atau masih berada di area yang sama sehingga dipilih pola penggunaan user KIRI di Bandung. Dari pengolahan ini, dipilih pola yang menarik bahwa user lebih suka menggunakan jalan menuju luar Bandung pada bulan Februari 2014.

Cara membuat pengolahan lunak untuk melakukan *data mining* pada *log* histori KIRI dengan cara

6.2 Saran

Untuk pengembangan aplikasi *data mining log* histori KIRI lebih lanjut, dapat dilakukan dengan cara menggunakan klasifikasi yang lebih baik dan detail. Pada daannya dengan menggunakan klasifikasi yang lebih detail adalah hasil dari *decision tree* mungkin akan lebih berhasil namun lebih banyak memiliki makna dan dapat menghasilkan nilai *confident* yang lebih besar.

DAFTAR REFERENSI

- [1] Data Mining *Data Mining Concepts and Techniques* 2006 : Jiaw i Han and Mich lin Kamb r
- [2] <http://w ka.sourc forg .n t/doc.stabl />
- [3] <http://www.graphviz.org/>

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.trav1/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 riminal+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+jucnction/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.cndkialadrshipschool.sc...
114009	E5D9904F0A8B4F99	2/1/2014 2:00	FINDROUTE	-6.88166,107.61561/-6.90774,107.60908/1