

SKRIPSI

DATA MINING HISTORI PENCARIAN RUTE ANGKOT



JOVAN GUNAWAN

NPM: 2011730029

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

UNDERGRADUATE THESIS

DATA MINING HISTORY SEARCHING ROUTE



JOVAN GUNAWAN

NPM: 2011730029

**DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES
PARAHYANGAN CATHOLIC UNIVERSITY
2014**

LEMBAR PENGESAHAN

DATA MINING HISTORI PENCARIAN RUTE ANGKOT

JOVAN GUNAWAN

NPM: 2011730029

Bandung, 14 September 2014

Menyetujui,

Pembimbing Tunggal

Pascal Alfadian, M.Com.

Ketua Tim Penguji

Anggota Tim Penguji

Plato

Euclid

Mengetahui,

Ketua Program Studi

Thomas Anung Basuki, Ph.D.

PERNYATAAN

Dengan ini saya yang bertandatangan di bawah ini menyatakan bahwa skripsi dengan judul:

DATA MINING HISTORI PENCARIAN RUTE ANGKOT

adalah bukti narasi karya saya sendiri, dan saya tidak melakukan penjiplakan atau pengutipan dengan cara-cara yang tidak sesuai dengan tika ilmuan yang berlaku dalam masyarakat ilmuan.

Atas pernyataan ini, saya siap menggantung segera risiko dan sanksi yang dijatuahkan kepada saya, apabila dalam mudian hari dituliskan adanya pelanggaran terhadap tika ilmuan dalam karya saya, atau jika ada tuntutan formal atau non-formal dari pihak lain berikaitan dengan kesalahan karya saya ini.

Dinyatakan di Bandung,
Tanggal 14 September 2014

M. T. Rai

Jovan Gunawan
NPM: 2011730029

ABSTRAK

Dengan kajian 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, data tersebut belum dapat diolah lebih lanjut dan menghasilkan suatu yang lebih. *Data mining* merupakan salah satu proses untuk memproses data untuk mempermudah menganalisis dan mengambil kesimpulan dari data yang dimiliki. Salah satu teknik dari *data mining* adalah dengan membuat *decision tree* untuk melakukan klasifikasi suatu objek. Terdapat beberapa metode untuk memilih atribut pada pembuatan *decision tree*, dua diantaranya adalah ID3 dan C4.5. Metode ID3 merupakan metode yang menggunakan nilai *entropy* untuk memilih atribut sedangkan C4.5 menggunakan nilai *entropy* dan *gain ratio* untuk memilih atribut.

Pada penelitian ini, percobaan *data mining* dilakukan pada data log histori KIRI bulan Februari 2014, khususnya untuk action dengan nilai FINDROUTE. Atribut yang akan diuji adalah *timestamp* dan *additionalData* yang berisi lokasi keberangkatan dan tujuan dari user yang menggunakan aplikasi KIRI. Pada percobaan ini, akan dibuat klasifikasi sepuuh derah di Bandung berdasarkan titik pusat Bandung dengan radius satu kilometer untuk setiap derah. Dengan klasifikasi tersebut, dapat ditentukan apakah user meninggalki Bandung atau mendekati Bandung atau menuju daerah yang sama. Pembuatan *decision tree* digunakan untuk melakukan klasifikasi apakah pada hari tertentu dan jam tertentu, user akan lebih sering meninggalki Bandung atau menuju Bandung atau menuju daerah yang sama. Dari hasil pengujian experimental, diperoleh bahwa *decision tree* yang dibuat dengan ID3 mengalami overfitting dengan akurasi 38.22%, sedangkan dengan J48 (metode C4.5 dari weka) tidak mengalami overfitting dengan akurasi 50.37% dan dipercaya simpulan bahwa user sering meninggalki Bandung daripada menuju Bandung atau menuju daerah yang sama pada bulan Februari 2014.

Kata-kata kunci: Data Mining, Decision Tree, KIRI

ABSTRACT

With the advancement of technology these days, accurate information is needed every day, but the information needs to be treated in order to be presented better. If viewed in more detail, the data can be treated further and produce something better. *Data mining* is one of processes to process data to analyze and draw conclusions. One of the techniques of *data mining* is to make *decision tree* to classify object. There are several methods to choose attributes at *decision tree*, two of which are ID3 and C4.5. ID3 method is a method that uses entropy to choose attributes, while C4.5 uses entropy and gain ratio to do that.

In this study, *data mining* experiments performed on the log KIRI history in February 2014, in particular to the action with the value FINDROUTE. Attributes to be used are timestamp and additionalData which contains departure location and destination location from user who uses KIRI application. In this experiment, will be made territorial classification in Bandung based on central point of Bandung with radius of one kilometer for each region. With the regional classification, can be determined whether the user is away from Bandung or heading to Bandung or heading to same region. Making the *decision tree* is used to classify whether on certain days and certain hours, users will be more frequent to leave Bandung or heading to Bandung or heading to same region. From the results of experimental testing, obtained that the *decision tree* created with ID3 is overfitting with 38.22% accuracy, while J48 (method C4.5 without pruning) is not overfitting with 50.37% accuracy, and it is concluded that user more often away from Bandung than heading Bandung or heading to same region at February 2014.

Keywords: Data Mining, Decision Tree, KIRI

Dipersembahkan untuk diri sendiri

KATA PENGANTAR

Puji syukur k pada Tuhan yang Maha Esa atas p tunjuk dan rahmat-Nya, p nulis dapat m ny l - saikan laporan skripsi tanpa ada halangan apapun dan dapat s l sai t pat pada waktunya.

Laporan skripsi yang t lah saya susun ini dibuat dalam rangka m m nuhi salah satu syarat untuk m ndapatkan g lar sarjana (S1) Program Studi Ilmu Komput r, Fakultas T knologi Informasi dan Sains, Univ rsitas Katolik Parahyangan.

Saya m nyadari bahwa skripsi ini dapat dis l saikan kar na p ranan banyak pihak. Ol h kar na itu, saya ingin m ngucapkan t rima kasih s b sar-b sarnya k pada:

1. Papa, Mama, Koko, dan Cici yang t lah m mb rikan dukungan dalam m ng rjakan skripsi ini pada saya.
2. Pak Pascal s laku dos n wali atas bimbingannya s lama saya m ng rjakan skripsi ini dan s - laku d v lop r KIRI yang t lah m ngizinkan saya untuk m lakukan *data mining* s rta analisis pada data *log* histori KIRI.
3. Dos n-dos n p nguji yang t lah m luangkan waktu untuk m mp lajari skripsi ini dan m m- b rikan kritik dan saran.
4. Pak Lionov yang t lah m mb rikan informasi, tips, s rta dorongan dalam m laksanakan skripsi.
5. Clara, Antonio, St v n, Samu l, K vin, dan s luruh angkatan 2011 IT Univ rsitas Parahyangan atas s gala dukungan, bantuan, saran, p rsahabatan, s rta k rja samanya s lama ini.

S rta s luruh pihak yang tidak dapat dis butkan satu-p rsatu atas dukungan kalian yang sangat b arti bagi saya, Tuhan m mb rkati.

Saya m nyadari bahwa skripsi ini masih jauh dari s mpurna, ol h kar na itu, saya m n rima saran dan kritik agar bisa m nghasilkan yang l bih baik. Akhir kata, saya b rharap skripsi ini dapat b rmanfaat bagi p rk mbangan ilmu t knologi khususnya di bidang *data mining*, dan bagi KIRI s rta para p mbaca.

Bandung, S pt mb r 2014

P nulis

DAFTAR ISI

KATA PENGANTAR	xv
DAFTAR ISI	xvii
DAFTAR GAMBAR	xix
DAFTAR TABEL	xx
1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Perumusan Masalah	2
1.3 Tujuan	2
1.4 Batasan Masalah	2
1.5 Metode Penelitian	2
1.6 Sistematika Pembahasan	3
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	20
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	52
4 PERANCANGAN PERANGKAT LUNAK	53
4.1 Perancangan Pengaruh Lunak	53
4.1.1 Perancangan Klasifikasi	53

4.1.2	S ku nc Diagram	60
4.1.3	P rancangan D sain Antar Muka	62
5	IMPLEMENTASI PROGRAM DAN PENGUJIAN	63
5.1	Lingkungan P mbangunan	63
5.2	Hasil Tampilan Antarmuka	63
5.3	P ngujian Aplikasi <i>Data Mining</i>	66
5.3.1	P ngujian Fungsional	66
5.3.2	P ngujian Eksp rim ntal	72
5.3.3	Analisis Hasil Uji	75
6	PENUTUP	77
6.1	K simpulan	77
6.2	Saran	77
DAFTAR REFERENSI		79
A	100 DATA PERTAMA DARI <i>log HISTORI KIRI</i>	81
B	THE SOURCE CODE	87

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>	11
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>	17
2.7	Hasil output Graphviz	42
3.1	<i>Classification</i> pada da rah Bandung	47
3.2	Diagram Use Case P rangkat Lunak <i>Data Mining Log Histori KIRI</i>	51
3.3	Diagram <i>Class</i> P rangkat Lunak <i>Data Mining Log Histori KIRI</i>	52
4.1	Diagram <i>Class</i> P rangkat Lunak <i>Data Mining Log Histori KIRI</i>	59
4.2	Diagram <i>Class</i> P rangkat Lunak <i>Data Mining Log Histori KIRI</i>	61
4.3	Mock Up Form P rtama	62
4.4	Mock Up Form K 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 nampulkan 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 Action FINDROUTE	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	74
5.16	Hasil dari SDForExtractData	75

DAFTAR TABEL

2.1	tabl m ngandung <i>missing value</i> dan <i>noisy data</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	52
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, data tersebut belum dapat diolah lebih lanjut dan menghasilkan suatu yang lebih baik. Salah satu cara mengolah data tersebut adalah dengan menggunakan proses *data mining*. Dengan menggunakan teknik *data mining*, analisa masalah, pengambilan kesimpulan akan menjadi lebih mudah, bahkan dapat 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* pada data log histori?
- Pola apa yang didapat dari data log histori KIRI?

1.3 Tujuan

Penelitian ini bertujuan untuk

- Mencari pola dan informasi yang menarik dari *log histori* KIRI
- Pengembangan program lunak dapat melakukan *data mining* dari *log histori* KIRI
- Mencari nilai dan menarik kesimpulan dari pola yang dipilih.

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 diterapkan pada log KIRI
- Merancang dan mengimplementasikan algoritma untuk *data mining*
- Mengimplementasikan pembangkit pola *data mining*
- Melakukan pengujian dan kspesifikasi

1.6 Sistematika Pembahasan

Sistematika pembahasan dalam penelitian ini adalah:

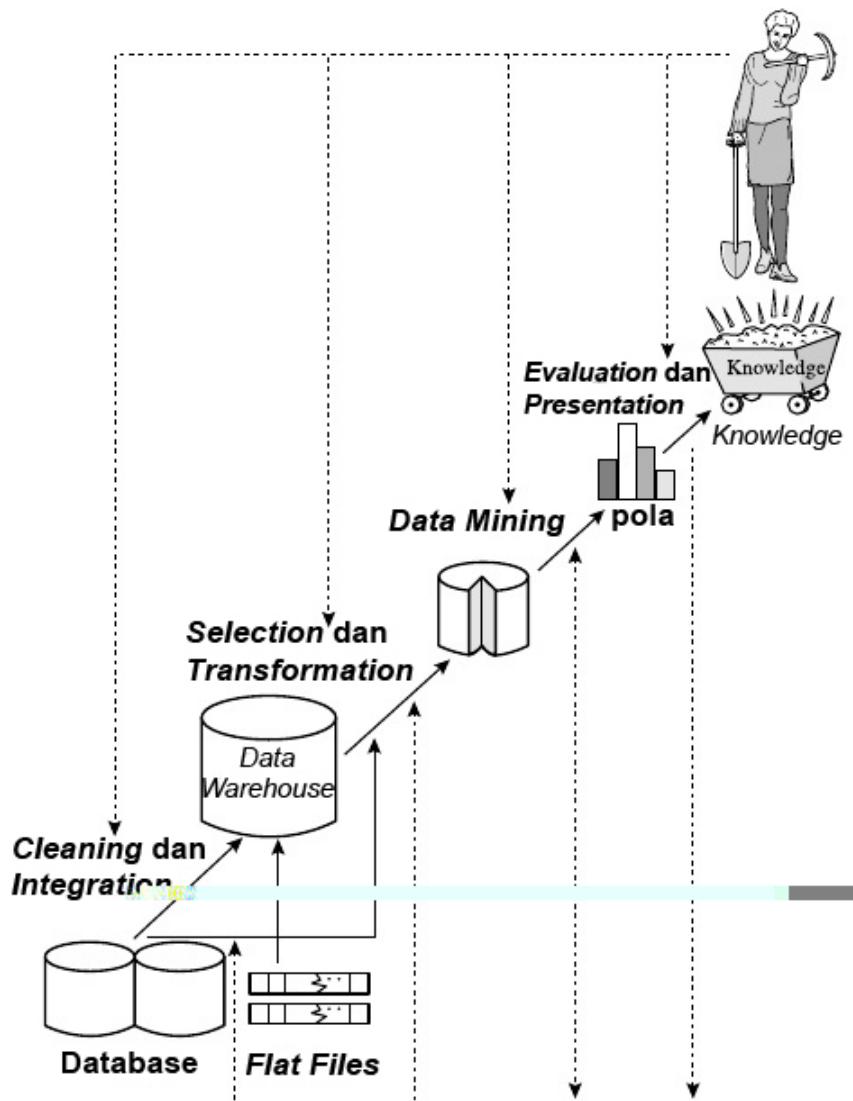
- BAB 1: Pendahuluan, berisi latar belakang dari penelitian ini, rumusan masalah yang timbul, tujuan yang ingin dicapain, ruang lingkup atau batasan masalah dari penelitian ini, serta tentang penelitian yang akan digunakan dan sistem matematika pembahasan dari penelitian ini
- BAB 2: Landasan Teori, berisi dasar teori mengenai *data mining*, *data cleaning*, *data integration*, *data selection*, *data transform*, *decision tree*, *pattern evaluation*, *knowledge presentation*, log histori KIRI, Havrsin Formula, Weka, dan Graphviz
- 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 dilakukan untuk mengambil inti sari atau pengetahuan dari data yang bersar dan merupakan salah satu langkah dari knowledge discovery.



Gambar 2.1: Tahap Data Mining, Ditiru mahkan dari [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 ketujuh merupakan bagian dari *data preprocessing*, dimana data-data disiapkan untuk dilakukan penggalian data. Tahap *data mining* merupakan tahap dimana melakukan penggalian data. Tahap ketujuh merupakan tahap pencarian pola yang merupakan bentuk penasikan *knowledge*. Sedangkan tahap terakhir merupakan visualisasi dan representasi dari *knowledge* yang sudah diproses 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 diproses dari database tidak selalu memiliki nilai yang lengkap atau sempurna. Ada beberapa nilai yang hilang, nilai yang tidak valid atau salah ketik. Atribut dari suatu database yang tidak relevan atau redundansi bisa diatasi dengan menghapus atribut atau record tersebut. Contoh studi data yang memiliki *missing value* dan *noisy data* dapat dilihat pada tabel 2.1

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

IdPenjualan	NamaBarang	Customer	Harga	BanyakBarang
1	Mous	Elvin	45000	2
2	Kyboard	Alleria	-35000	1
3	Monitor		225000	1

Dapat dilihat, pada idPenjualan 2, harga dari keyboard adalah -35000, itu merupakan *noisy data* karena tidak mungkin 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 secara manual
- Mengisi nilai yang hilang dengan menggunakan nilai konstan yang bersifat umum
- Menggunakan nilai rata-rata dari suatu atribut untuk mengisi nilai yang hilang

Noisy Data

Noisy data merupakan nilai yang berdasarkan 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 model pola yang samaan atribut untuk memprediksi suatu nilai
- *Clustering*, merupakan metode pengelompokan dimana dituliskan outliers yang dapat dibuang. Outliers merupakan data yang berada diluar kumpulan yang sesuai dengan observasi atau pertemuan.

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 hasil sesuai dengan kondisi tertentu (contoh, penjualan total per tahun dari 2005-2010). Sedangkan flat data merupakan data yang disimpan dengan cara apapun untuk merepresentasikan database modal pada sebuah data baik berbentuk *plain text file* maupun *binary file*.

Tahap ini harus dilakukan secara teliti terutama ketika dalam memasangkan nilai-nilai yang berdasarkan dari sumber yang berbeda. Pada tahap ini, perlu dilakukan identifikasi data apakah data tersebut harus dimasukkan 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 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 dipertahankan serta dipertahankan nilai-nilai seperti primary key, atribut, dan lain-lain agar tidak terjadi error pada database yang sudah digabung. Proses dari penggabungan hingga perbaikan nilai-nilai pada kedua 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:

- NPMMahasiswa
- NamaMahasiswa
- JenisKelamin
- Alamat

- MataKuliah
- NilaiART
- NilaiUTS
- NilaiUAS

Maka, atribut yang berpotensi diambil adalah MataKuliah, NilaiART, NilaiUTS, NilaiUAS, sedangkan atribut yang akan dibuang adalah NPMMahasiswa, NamaMahasiswa 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 konsep hierarki. Contoh studi kasus, nilai pada atribut umur dapat dikumpulkan menjadi muda, dewasa, tua.

Normalization

Atribut dapat dinormalisasi dengan menggunakan skala pada nilainya sehingga nilai tersebut menjadi suatu rang yang lebih spesifik dan kental seperti 0,0 sampai 1,0. Terdapat beberapa teknik normalisasi, dua diantaranya yaitu, *min-max normalization* dan *z-score normalization*. *Min-max normalization* akan mengubah semua nilai menjadi nilai dalam skala tertentu. Dengan menggunakan rumus

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

Contoh kasus, misalkan nilai minimum dan maximum dari suatu pendapatan adalah 12.000 dan 98.000, akan diubah menjadi 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

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

Contoh kasus, misal nilai rata-rata dan standar deviasi dari nilai-nilai atribut pendapatan adalah 54.000 dan 16.000. Dengan z-score, jika ada nilai pendapatan baru yaitu 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 penilaian sikap seseorang siswa atau "mini bus", "bus", atau "s" dalam kategori tip mobil. Kategori tersebut dapat diperkirakan dengan menggunakan nilai diskrit. Nilai diskrit merupakan nilai yang terpisah dan berbeda, seperti 1 atau 5. Kategori yang diperkirakan oleh nilai diskrit maka akan menjadi nilai yang terurut dan tidak

m miliki arti, s p rti 1,2,3 untuk m r pr s ntasikan kat gori tip mobil "mini bus", "bus", dan "s dan".

Prediction m rupakan mod l yang dibangun untuk m ramalkan fungsi nilai kontinu atau *ordered value*. *Ordered value* m rupakan nilai yang t rurut dan b rlanjut. Contoh studi kasus untuk p mod lan pr diction adalah s orang mark ting ingin m ramalkan s b rapa banyak konsum n yang akan b lanja di s buah toko dalam waktu satu bulan. P mod lan t rs but dis but *predictor*. *Regression Analysis*, m rupakan m todologi statistik yang digunakan untuk *numeric prediction*. *Classification* dan *numeric prediction* m rupakan dua j nis utama dalam masalah pr diksi.

Data Classification m rupakan pros s untuk m lakukan klasifikasi. *Data classification* m miliki dua tahap pros s, yaitu *learning step* dan tahap klasifikasi s p rti pada ilustrasi di gambar 2.2. *Learning step* m rupakan langkah p mb lajaran, di mana algoritma klasifikasi m mbangun *classification rules* (yang b risi syarat atau aturan s buah nilai masuk k dalam kat gori t rt ntu) d ngan cara m nganalisis *training set* yang m rupakan *database tuple*. Kar na p mbuatan *classification rules* m nggunakan *training set*, yang dik nal juga s bagai *supervised learning*. Pada tahap k dua, dilakukan pros s klasifikasi nilai b rdasarkan *classification rules* yang sudah dibangun dari tahap p rtama.

Decision Tree

Salah satu cara p mbuatan *classification rules* pada *Data Classification* adalah d ngan m mbuat *decision tree* (pohon k putusan). *Decision tree* m rupakan *flowchart* yang b rb ntuk pohon, dimana s tiap nod int rnal (*nonleaf nod*) m rupakan hasil t st dari atribut, s tiap cabang m - r pr s ntasikan output dari t st, dan s tiap nod daun m miliki *class label*. Bagian paling atas dari pohon dis but *root node*. Contoh studi kasus, pohon k putusan untuk m n ntukan apakah s orang konsum n akan m mb li komput r atau tidak (ilustrasi pohon k putusan pada gambar 2.3)

Decision Tree Induction *Decision tree induction* m rupakan p latihan pohon k putusan dari tup l p latihan k las b rlab l. T rdapat b b rapa t knik untuk m mbuat *decission tree* dua diantarnya adalah ID3 dan C4.5. ID3 m rupakan t knik p mbuatan *decision tree* d ngan m manfaatkan *entropy* dan *gain info* untuk m n ntukan atribut yang t rbaik untuk nod pada *decision tree*. S -dangkan C4.5 m rupakan t knik lanjutan dari ID3 yang m nggunakan *gain ratio* untuk m lakukan p ng c kan pada nilai *gain info*. K dua t knik t rs but m nggunakan p nd katan *greedy* yang m rupakan *decission tree* yang dibangun s cara *top-down recursive divide and conquer*. Algoritma yang dip rlukan s cara umum sama, hanya b rb da pada *attribute_selection_method*. B rikut algoritma untuk m mbuat pohon k putusan dari suatu tup l p latihan.

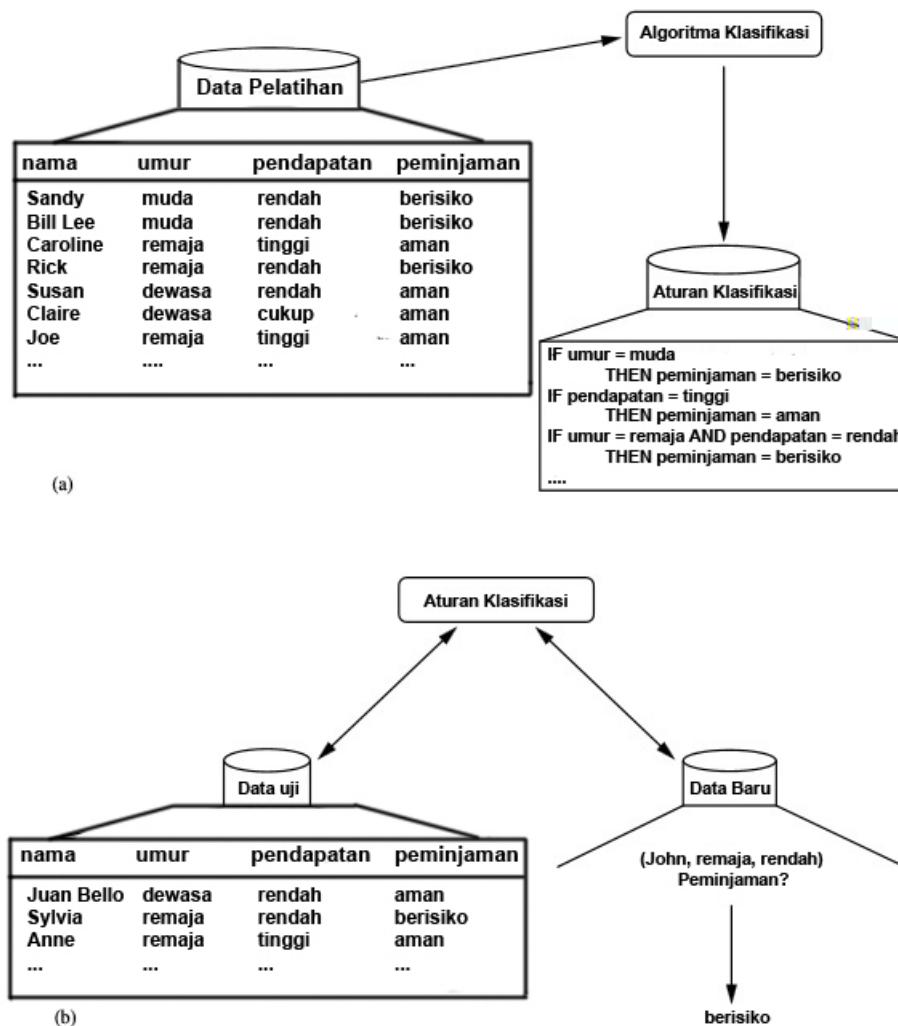
Require: Partisi data, D, m rupakan s t data p latihan dan k las lab l

Require: *attribute_list*, m rupakan s t dari atribut kandidat

Require: *Attribute_selection_method*, pros dur untuk m n ntukan *splitting criterion*. Pada in put ini, t rdapat juga data *splitting_attribute* dan mungkin salah satu dari *split point* atau *splitting subset*

Ensure: Pohon k putusan

- 1: M mbuat nod N;
- 2: **if** tupl pada D m rupakan k las yang sama, C **then**

Gambar 2.2: Tahap *data classification*, Dit rj mahkan dari [1]Gambar 2.3: Contoh *decision tree*, Dit rj mahkan dari [1]

```

3: return N s bagai nod daun d ngan lab 1 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 1 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  $\leftarrow$  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 1 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;

```

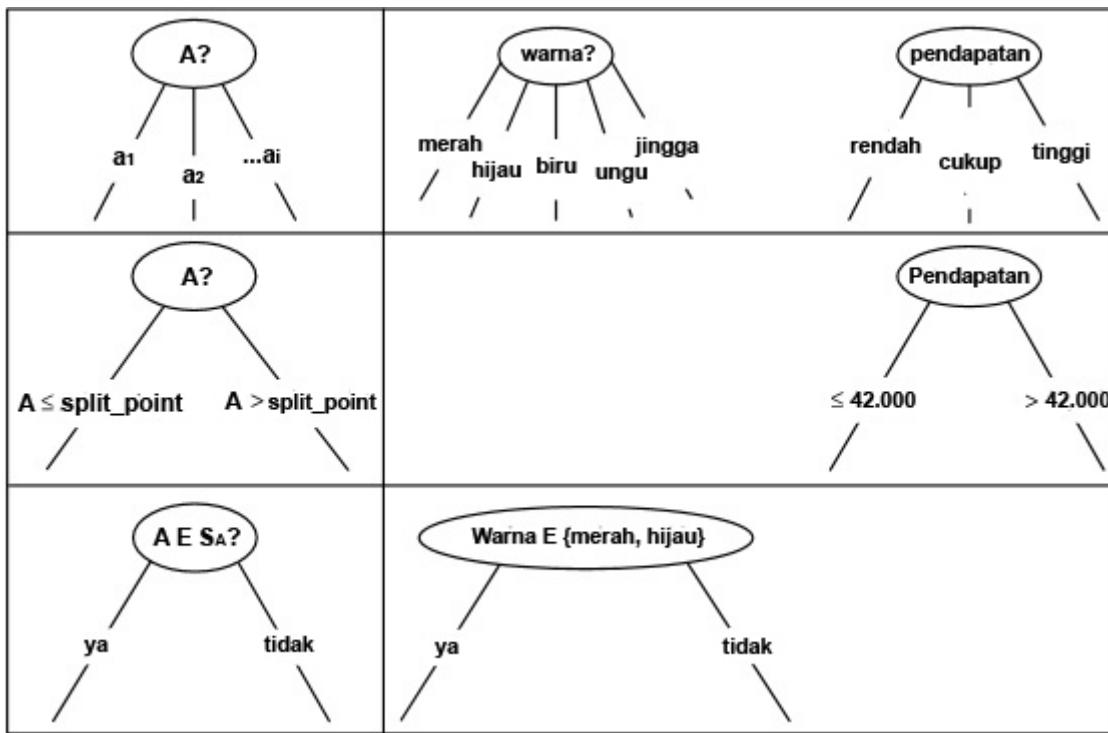
Pohon k putusan akan dimulai d ngan satu nod , yaitu N, m r pr s ntasikan tupl p latihan pada D (baris 1)

Jika tupl di D m miliki k las yang sama s mua, maka nod N akan m njadi daun dan dib ri lab 1 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 m rupakan nilai t rbaik untuk m m cah nilai atribut pada tupl k dalam k las masing-masing. (baris 8)

Nod N akan diisi d ngan hasil dari *splitting criterion* (baris 9). K mudian krit ria t rs but agak dib ntuk cabangnya masing-masing s suai pada baris 13 dan 14. T rdapat tiga k mungkin b ntuk krit ria jika A m rupakan *splitting_attribute* yang m miliki nilai unik s p rti $\{a_1, a_2, \dots, a_v\}$ s p rti pada gambar 2.4, yaitu,

1. *Discrete valued*: cabang yang dihasilkan m miliki k las d ngan nilai diskrit. Kar na k las yang dihasilkan diskrit dan hanya m miliki nilai yang sama pada cabang t rs but, maka attribut_list akan dihapus (baris 10 sampai 12)
2. *Continuous values*: cabang yang dihasilkan m miliki jarak nilai untuk m m nuhi suatu kondisi (contoh: $A \leq split_point$), dimana nilai *split_point* adalah nilai p mbagi yang dik mbalikan ol h *attribute_selection_method*



Gambar 2.4: Jenis-jenis split point, Ditiru mahkan dari [1]

3. *Discrete valued and a binary tree:* cabang yang dihasilkan adalah dua bentuk rupa nilai iya atau tidak dari “apakah A anggota S_a ”, dimana S_a merupakan subset dari A, yang dikembalikan oleh *Attribute_selection_method*

Kemudian, akan dipanggil kembali algoritma *decision tree* untuk setiap nilai hasil pembagian pada tupl D_j (baris 18).

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

1. Semua tupl pada partisi D merupakan bagian dari kelas yang sama.
2. Sudah tidak ada atribut yang dapat dilakukan pembagian lagi (dilakukan pengelompokan pada baris 4). Disini, akan dilakukan *majority voting* (baris 6) yang akan mengkonversi node N menjadi *leaf* dan dibuat label dengan kelas yang terbanyak pada D .
3. Sudah tidak ada tupl yang dapat dibagi cabang, D_j sudah kosong (baris 15) dan *leaf* akan dibuat dengan *majority class* pada D (baris 16).

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

Attribute Selection Measure merupakan suatu hierarki untuk memiliki *splitting criterion* yang terbaik yang memisahkan partisi data (D), tupl pada latihan kelas lab 1 k dalam kelas masing-masing. *Attribute Selection Measure* menyediakan peringkat untuk setiap atribut pada training tupl. 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, $s \in D$ merupakan latihan dari *class-labeled tuple*. Jika s memiliki atribut m miliki m nilai yang berbeda yang belum definisikan maka s yang memiliki nilai C_i (for $i=1,\dots,m$). $C_{i,d}$ menjadi kelas tuple dari C_i di D . $|D|$ dan $|C_{i,d}|$ merupakan banyak tuple pada D dan $C_{i,d}$.

ID3

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

Setelah mendapatkan nilai $Info(D)$ dan $Info_A(D)$, *information gain* dapat diperoleh dengan mengurangi $Info(D)$ dengan $Info_A(D)$

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

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

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

Pada tabel 2.2, terdapat *training set*, D . Atribut k kelas lab 1 merupakan dua nilai yang berbeda yaitu ya dan tidak, maka dari itu, nilai $m = 2$. C_1 diisi dengan kelas lab 1 ber nilai ya, sedangkan C_2 diisi dengan kelas lab 1 ber nilai tidak. Terdapat sembilan tuple atribut k kelas lab 1 dengan nilai ya dan lima tuple dengan nilai tidak. Untuk dapat menggunakan *splitting criterion*, *information gain* harus dihitung untuk setiap atribut terlebih dahulu. Perhitungan *entropy* untuk D adalah

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

Setelah diperoleh nilai *entropy* dari D , k median akan dihitung nilai *entropy* atribut dimulai dari atribut umur. Pada kategori muda, terdapat dua tuple dengan kelas ya dan tiga tuple dengan kelas tidak. Untuk kategori remaja, terdapat empat tuple dengan kelas ya dan nol tuple dengan kelas tidak. Pada kategori dewasa, terdapat tiga tuple dengan kelas ya dan dua tuple dengan kelas tidak.

Tab 1.2.2: Contoh training set

RID	umur	p ndapatkan	siswa	r siko_kr dit	Class: m mb li_komput r
1	muda	tinggi	tidak	cukup	tidak
2	muda	tinggi	tidak	baik	tidak
3	r maja	tinggi	tidak	cukup	ya
4	d wasa	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

P rhitungan nilai *entropy* atribut umur t rhadap D s bagai b rikut

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

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

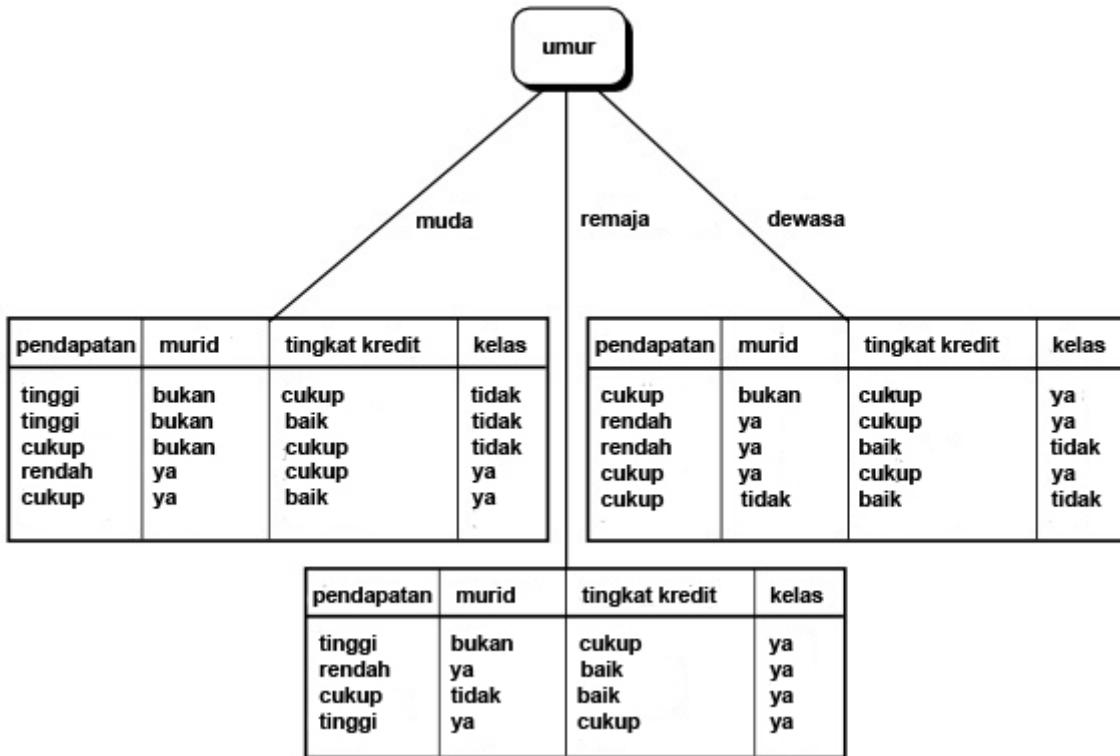
$$Gain_{(umur)} = Info(D) - Info_{age}(D) = 0.940 - 0.694 = 0.246 bits$$

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

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

C4.5

Information gain akan m miliki nilai yang baik jika suatu atribut m miliki banyak nilai yang b rb - da, namun hal itu tidak s lalu bagus. S bagai contoh kasus, jika nilai id suatu tabl yang m miliki nilai unik, maka akan t rdapat banyak s kali cabang. Namun s tiap cabang hanya akan b risi satu tupl dan b rsifat *pure*, maka nilai *entropy* yang dihasilkan adalah 0. Ol h kar na itu, informasi



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

yang dipilih pada atribut ini akan ber nilai maksimum namun tidak akan berguna untuk classification [1]. Selain itu, ID3 dapat menghasilkan decision tree yang memprediksi secara berlebihan (overestimated) atau dapat juga overfitting. Hal ini dikarenakan pohon yang dihasilkan terlalu mendekati set data input memiliki hasil prediksi yang pasti.

C4.5 merupakan teknik lanjutan dari ID3, yang menggunakan gain ratio sebagai attribute selection measure untuk memilih atribut. Kependekan, C4.5 melakukan tree pruning untuk menghindari overfitting.

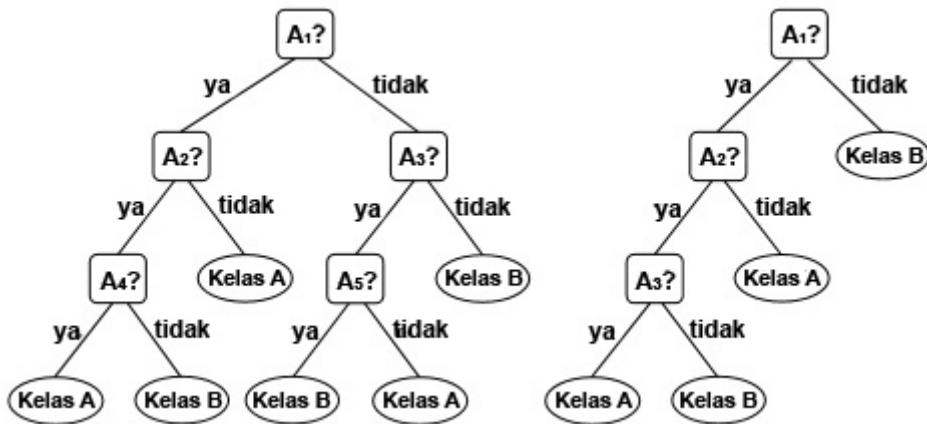
C4.5 menggunakan nilai tambahan dari information gain yaitu gain ratio, yang dapat mengatasi permasalahan information gain tentang nilai yang banyak. C4.5 melakukan teknik normalisasi terhadap gain information dengan menggunakan split information yang memiliki rumus sebagai berikut:

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

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

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

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



Gambar 2.6: Decision tree yang belum dipotong dan yang sudah dipotong, Ditiru mahkan dari [1]

sama bersarnya dengan nilai rata-rata dari semua tipe yang diperiksa.

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

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

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

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

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

Tree Pruning Tree pruning merupakan proses pemotongan decision tree agar lebih fisis dan tidak terlalu memengaruhi nilai klasifikasi 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 memiliki karakteristik seperti ketahanan klasifikasi 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.

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

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

2.1.6 *Pattern Evaluation*

Pattern evaluation merupakan tahap mendeklarasikan apakah *pattern* atau pola tersebut benar atau tidak dan memperbaikinya berdasarkan berbagai *interestingness measures*. Suatu *pattern* atau pola dapat dinyatakan benar jika apabila

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

2.1.7 *Knowledge Presentation*

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

2.2 Log Histori KIRI

KIRI memiliki log histori yang merupakan catatan untuk setiap usaha kota menggunakan KIRI. Data log tersebut dapat diolah dengan cara melakukan wawancara dengan developer KIRI, yaitu Pascal Alfadian. Data log yang dibuatkan 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 ketika pengguna KIRI mencari ruta angkot menggunakan waktu UTC / GMT
- Action, tipe dari log yang dibuat.
- AdditionalData, mencatat data-data yang berhubungan dengan nilai atribut action

LogId merupakan field dengan tip data int dengan batas 6 karakter yang digunakan sebagai primary key dari tabel tersebut. LogId diisi dengan menggunakan fungsi increment integer. Increment integer merupakan fungsi untuk pengisian data pada database dengan menambahkan nilai 1 dari nilai yang terakhir kali diisi. APIKey merupakan field dengan tip data varchar yang digunakan untuk mengidentifikasi pengguna KIRI ketika menggunakan KIRI. Timestamp (UTC) merupakan field dengan tip data timestamp yang digunakan untuk mencatat waktu penggunaan KIRI oleh user, diisi dengan menggunakan fungsi current time. Current time merupakan fungsi untuk pengisian data pada database dengan mengambil waktu pada komputer ketika record dibuat. Action merupakan field dengan tip data varchar yang digunakan untuk memeriksa fungsi apa yang dipanggil dari API KIRI. Terdapat beberapa tip pada field action, yaitu

- ADDAPIKEY, action yang dicatat ketika fungsi pembuatan API key yang baru dipanggil.

- *FINDROUTE*, action yang dicatat k tika us r m lakukan pencarian rut
- *LOGIN*, action yang dicatat k tika d v lop rs m lakukan login d ngan m menggunakan API key
- *NEARBYTRANSPORT*, action yang dicatat k tika us r m ncari transportasi di daerah 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 pendaftaran pada KIRI API key
- *SEARCHPLACE*, action yang dicatat k tika us r m panggil fungsi pencarian lokasi d ngan m menggunakan nama tempat
- *WIDGETERROR*, mncatat log t rs but k tika us r m nrima error dari *widget*
- *WIDGETLOAD*, mncatat log t rs but k tika us r m lakukan download widg t

AdditionalData, merupakan field d ngan tip data varchar yang digunakan untuk mncatat 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 m lakukan login s rta status apakah user berhasil login atau tidak
- Jika nilai atribut *action* adalah *NEARBYTRANSPORT*, maka isi dari additionalData adalah *latitude* dan *longitude* dari transportasi t rs but
- 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 mngistir 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 m lakukan download widg t

2.3 Haversine Formula

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

$$a = \sin^2((|\varphi_1 - \varphi_2|)/2) + \cos \varphi_1 \cdot \cos \varphi_2 \cdot \sin^2((|\lambda_1 - \lambda_2|)/2)$$

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

$$d = R.c$$

Dimana

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

Contoh untuk perhitungan Haversin sebagai berikut: Jika kita ingin menghitung jarak dua titik dari daerah Jakarta ke Surabaya, dengan titik pada Jakarta adalah -6.211544, 106.845172 dan titik pada Surabaya adalah -7.289166, 112.734398, maka perhitungan rumusnya akan menjadi

$$a = \sin^2((|-6.211544 - (-7.289166)|)/2) + \cos -6.211544 \cdot \cos -7.289166 \cdot \sin^2((|106.845172 - 112.734398|)/2)$$

$$a = 0.0026906745$$

$$c = 2.0.0026906745 \tan^2(\sqrt{0.0026906745}, \sqrt{1 - 0.0026906745})$$

$$c = 0.1037900036$$

$$d = 6.371 \times 0.1037900036$$

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

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

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

2.4 Weka

[3] Weka merupakan aplikasi berbasis java yang berisi alat-alat untuk melakukan visualisasi dan algoritma untuk data analisis serta program pengolahan. Weka juga menyediakan file weka-src.jar yang berisi kelas-kelas yang dipakai oleh aplikasi weka hingga saat ini 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 numerik ataupun nominal pada wakas. Klasifikasi but memiliki *method* sebagai berikut:

- void buildClassifier(Instances data)

untuk melakukan penghasilkan klasifikasi dengan parameter tertentu data latihan.

- double classifyInstance(Instances instance)

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

- double[] distributionForInstance(Instances instance)

untuk memprediksi anggota klasifikasi untuk contoh yang dibatasi dengan parameter contoh data yang akan dilakukan klasifikasi dan mengembalikan array yang berisi nilai k anggota dari contoh data.

- Capabilities getCapabilities()

mengembalikan kapabilitas dari klasifikasi tersebut.

Instance adalah interface yang mewakili set data.

Method:

- Attribut getAttribut(int index)

Mengembalikan atribut dari indeks yang dibatasi.

- Attribut getClassAttribut()

Mengembalikan atribut kelas.

- int getClassIndex()

Mengembalikan indeks atribut kelas itu.

- boolean classIsMissing()

Mengecek apakah kelas turunan hilang.

- double getClassValue()

Mengembalikan nilai kelas contoh sebagai angka floating-point.

- Instances getDataset()

Mengembalikan dataset.

- void deleteAttributAt(int position)

Menghapus atribut pada posisi tertentu.

- java.util.List<Attribut> numRateAttribut()

Mengembalikan penghitungan semua 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)

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

- void s tValu (int attInd x, java.lang.String valu)

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

- void s tValu Spars (int ind xOfInd x, doubl valu)

M n tapkan nilai t rt ntu dalam contoh d ngan nilai yang dib rikan (format floating-point).

- void s tW ight(doubl w ight)

M ngatur b rat contoh.

- java.lang.String stringValu (Attribut att)

M ng mbalikan nilai nominal, string, tanggal, atau atribut r lasional untuk contoh s bagai string.

- java.lang.String stringValu (int attInd x)

M ng mbalikan nilai nominal, string, tanggal, atau atribut r lasional untuk contoh s bagai string.

- doubl [] toDoubl Array()

M ng mbalikan nilai-nilai masing-masing atribut s bagai array ganda.

- `java.lang.String toString(Attribut att)`
Mengembalikan d skripsi satu nilai dari contoh s bagai string.
- `java.lang.String toString(Attribut att, int aft rD cimalPoint)`
Mengembalikan d skripsi satu nilai dari contoh s bagai string.
- `java.lang.String toString(int attInd x)`
Mengembalikan d skripsi satu nilai dari contoh s bagai string.
- `java.lang.String toString(int attInd x, int aft rD cimalPoint)`
Mengembalikan d skripsi satu nilai dari contoh s bagai string.
- `java.lang.String toStringNoW ight()`
Mengembalikan d skripsi satu contoh (tanpa berasal ditambahkan).
- `doubl valu (Attribut att)`
Mengembalikan nilai atribut contoh dalam format int rnal.
- `doubl valu (int attInd x)`
Mengembalikan nilai atribut contoh dalam format int rnal.
- `doubl valu Spars (int ind xOfInd x)`
Mengembalikan nilai atribut contoh dalam format int rnal.
- `doubl w ight()`
Mengembalikan berat contoh itu.

Instances adalah kelas untuk mengelola data.

Atribut:

- `String ARFF_DATA`
digunakan untuk menyimpan struktur arff data.
- `String ARFF_RELATION`
digunakan untuk menyimpan header arff data.
- `String FILE_EXTENSION`
extensi dari nama file yang digunakan untuk file arff.
- `String SERIALIZED_OBJ_FILE_EXTENSION`
ekstensi dari nama file yang digunakan untuk bin.

Constructor:

- `Instanc s(Instanc s datas t)`
Konstruktor menyimpan semua contoh dan referensi untuk informasi header dari himpunan contoh.

- `Instanc s(Instanc s datas t, int capacity)`

Membuat konstruktor yang menciptakan himpunan kosong contoh.

- `Instanc s(Instanc s sourc , int first, int toCopy)`

Membuat satu struktur baru kasus dengan menggunakan bagian dari struktur s.

- `Instanc s(java.io.R ad r r ad r)`

Membaca file ARFF, dan membaca bobot satunya untuk setiap contoh.

- `Instanc s(java.lang.String nam , java.util.ArrayList<Attribut > attInfo, int capacity)`

Membuat himpunan kosong contoh.

Method:

- `bool an add(Instanc instanc)`

Membuat tambahan struktur data.

- `void add(int ind x, Instanc instanc)`

Membuat satu contoh di posisi tertentu dalam daftar.

- `Attribut attribut (int ind x)`

Mengambil atribut.

- `Attribut attribut (java.lang.String nam)`

Mengambil 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.

- `bool an ch ckForAttribut Typ (int attTyp)`

Cek untuk atribut dari tip yang dibutuhkan dalam dataset.

- `bool an ch ckForStringAttribut s()`

Cek string atribut dalam dataset.

- `bool an ch ckInstanc (Instanc instanc)`

Meriksa apakah contoh yang dibutuhkan kompatibel dengan dataset ini.

- `Attribut classAttribut ()`

Mengambil atribut kelas.

- int classInd x()

Mengembalikan indeks atribut kelas itu.

- void del t ()

Menghapus semua contoh dari dataset t.

- void del t (int index x)

Menghapus satu buah contoh di posisi t-rt-ntu dari dataset t.

- void del t Attribut At (int position)

Menghapus atribut pada posisi t-rt-ntu.

- void del t Attribut Typ (int attTyp)

Menghapus semua atribut dari tip yang dibatasi dalam dataset t.

- void del t StringAttribut s()

Menghapus semua atribut string dalam dataset t.

- void del t WithMissing(Attribut att)

Menghapus semua contoh dengan nilai-nilai yang hilang untuk atribut t-rt-ntu dari dataset t.

- void del t WithMissing(int attInd x)

Menghapus semua contoh dengan nilai-nilai yang hilang untuk atribut t-rt-ntu dari dataset t.

- void del t WithMissingClass()

Menghapus semua contoh dengan nilai kelas hilang dari dataset t.

- java.util.List<Attribut> numRateAttribut s()

Pengembalian penghitungan semua atribut.

- java.util.List<Instanc> numRateInstanc s()

Pengembalian penghitungan semua contoh dalam dataset t.

- boolean qualHasAttribute(Instanc s dataset t)

Cek jika dua halaman yang sama.

- java.lang.String qualHasAttributeMsg(Instanc s dataset t)

Cek jika dua halaman yang sama.

- Instanc firstInstanc ()

Mengembalikan contoh pertama di dataset t.

- Instanc get(int index x)

Mengembalikan contoh pada posisi t-rt-ntu.

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

Mengembalikan nomor acak.

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

Mengembalikan string r visi.

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

Menyisipkan atribut pada posisi tertentu (0 numAttribut s ()) dan menutupkan semua nilai hilang.

- `Instanc instanc (int ind x)`

Mengembalikan contoh pada posisi tertentu.

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

Mengembalikan nilai atribut k-terkecil dari atribut numerik.

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

Mengembalikan nilai atribut k-terkecil dari atribut numerik.

- `Instanc lastInstanc ()`

Mengembalikan contoh terakhir di set.

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

Mendukung utama untuk kelas ini.

- `doubl meanOrMod (Attribut att)`

Mengembalikan rata (modus) untuk angka (nominal) atribut sebagai nilai floating-point.

- `doubl meanOrMod (int attInd x)`

Mengembalikan rata (modus) untuk angka (nominal) atribut sebagai nilai floating-point.

- `static Instanc s mrg Instanc s(Instanc s first, Instanc s s cond)`

Menggabungkan dua set Contoh bersama-sama

- `int numAttribut s()`

Mengembalikan jumlah atribut.

- `int numClass s()`

Mengembalikan jumlah kelas.

- `int numDistinctValu s(Attribut att)`

Mengembalikan jumlah nilai yang berbeda dari atribut yang diberikan.

- `int numDistinctValu s(int attInd x)`

Mengembalikan jumlah nilai yang berbeda dari atribut yang diberikan.

- `int numInstanc s()`

M eng balikan jumlah kasus dalam datas t.

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

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

- `java.lang.String relationNam ()`

M eng balikan nama hubungan itu.

- `Instanc r mov (int ind x)`

M enghapus contoh pada posisi t rt ntu.

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

M ngganti nama atribut.

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

M ngganti nama atribut.

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

M ngganti nama nilai nominal (atau string) nilai atribut

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

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

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

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

- `Instanc s r sampel (java.util.Random random)`

M mbuat datas t baru d engan ukuran yang sama d engan m enggunakan random sampling d engan p nggantian.

- `Instanc s r sampel WithW ights(java.util.Random random)`

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

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

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

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

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

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

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

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

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

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

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

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

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

- Instanc s t(int ind x, Instanc instanc)

M nggantikan contoh pada posisi t rt ntu.

- void s tClass(Attribut att)

M ngatur atribut class.

- void s tClassInd x(int classInd x)

M ngatur ind ks k las s t.

- void 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)

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

- Instantiates stringFromStructur()

Buat salinan struktur.

- double sumOfWeights()

Menghitung jumlah semua bobot contoh.

- void swap(int i, int j)

Mengukur posisi dua contoh di set.

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

Mendukung ujian kelas ini.

- Instantiates testCV(int numFolds, int numFold)

Menciptakan set tiga untuk satu kali lipat dari cross-validation pada dataset.

- java.lang.String toString()

Mengembalikan dataset sebagai string dalam format ARFF.

- java.lang.String toSummaryString()

Menghasilkan string yang ringkas tentang dataset.

- Instantiates trainCV(int numFolds, int numFold)

Menciptakan penerapan latihan ditampakkan untuk satu kali lipat dari cross-validation pada dataset.

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

Menciptakan penerapan latihan ditampakkan untuk satu kali lipat dari cross-validation pada dataset.

- double variance(Attribute attr)

Menghitung varians untuk atribut numerik.

- double variance(int attIndex)

Menghitung varians untuk atribut numerik.

- double[] variances()

Menghitung varians untuk semua atribut numerik secara bersamaan.

Attribute adalah klas yang digunakan untuk mengelola 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 numerik.

- static java.lang.String ARFF_ATTRIBUTE_NUMERIC

Kata kunci yang digunakan untuk menunjukkan atribut numerik.

- static java.lang.String ARFF_ATTRIBUTE_REAL

Kata kunci yang digunakan untuk menunjukkan atribut numerik.

- 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 subrelation.

- static int DATE

Skalar konstan untuk atribut dengan nilai tanggal.

- static java.lang.String DUMMY_STRING_VAL

Dummy pertama nilai String atribut.

- static int NOMINAL

Skalar konstan untuk atribut nominal.

- static int NUMERIC

Skalar konstan untuk atribut numerik.

- static int ORDERING_MODULO

Skalar konstan untuk atribut ordering modulo.

- static int ORDERING_ORDERED

Skalar konstan untuk atribut ordered.

- static int ORDERING_SYMBOLIC

S t konstan untuk atribut simbolik.

- static int RELATIONAL

S t konstan untuk atribut nilai r lazi.

- static int STRING

S t konstan untuk atribut d ngan nilai-nilai string.

Constructor:

- Attribut (java.lang.String attribut Nam)

Konstruktor untuk atribut num rik.

- Attribut (java.lang.String attribut Nam , Instanc s h ad r)

Konstruktor untuk atribut nilai r lazi.

- Attribut (java.lang.String attribut Nam , Instanc s h ad r, int ind x)

Konstruktor untuk atribut nilai r lazi d ngan ind ks t rt ntu.

- Attribut (java.lang.String attribut Nam , Instanc s h ad r, Prot ct dProp rti s m tadata)

Konstruktor untuk atribut nilai r lazi.

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

Konstruktor untuk atribut num rik d ngan ind ks t rt ntu.

- Attribut (java.lang.String attribut Nam , java.util.List<java.lang.String> attribut Valu s)

Konstruktor untuk atribut nominal dan atribut string.

- Attribut (java.lang.String attribut Nam , java.util.List<java.lang.String> attribut Valu s, int ind x)

Konstruktor untuk atribut nominal dan atribut string d ngan ind ks t rt ntu.

- 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.

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

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

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

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

- java.util.String formatDat (doubl dat)

M ng mbalikan milid tik s suai d ngan tanggal saat ini.

- java.util.String g tDat Format()

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

- doubl g tLow rNum ricBound()

P ng mbalian batas bawah dari atribut num rik.

- Prot ct dProp rti s g tM tadata()

M ng mbalikan prop rti s dis diakan untuk atribut ini.

- `java.lang.String getVision()`

Mengembalikan string r visi.

- `double getUpperNumericBoundary()`

Mengembalikan nilai dari atribut numrik.

- `int hashCode()`

Mengembalikan kod hash untuk atribut ini berdasarkan namanya.

- `boolean hasZeroPoint()`

Pengembalian apakah atribut memiliki titik desimal.

- `int indexOfX()`

Mengembalikan indeks x dari atribut ini.

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

Mengembalikan indeks dari nilai atribut tertentu.

- `boolean isAveragable()`

Pengembalian apakah atribut dapat dirata-ratakan bermakna.

- `boolean isDate()`

Pengujian jika atribut adalah jenis tanggal.

- `boolean isInRange(double value)`

Mengecek apakah suatu nilai termasuk dalam batas-batas atribut.

- `boolean isNominal()`

Mengejek apakah atribut nominal.

- `boolean isNumeric()`

Pengujian jika atribut numrik.

- `boolean isRelationshipValue()`

Pengujian jika atribut hubungan dihargai.

- `boolean isString()`

Pengujian jika atribut string.

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

Metoda utama yang sdh rhana untuk mengejek kasus ini.

- `java.lang.String getName()`

Mengembalikan nama atribut itu.

- int numValue()

Mengembalikan jumlah nilai atribut.

- int ordinal()

Mengembalikan posisi suatu atribut.

- int parseData (java.lang.String string)

Mengurai string yang dibaca seperti data, sesuai format saat ini dan mengembalikan sesuai dengan jumlah milidigit.

- Instantiation()

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

- Instantiation(int valIndex)

Mengembalikan nilai atribut nilai rasio.

- void setStringValue (java.lang.String value)

Mengosongkan nilai dan mengatur karakter yang hanya menyimpan nilai yang dibaca.

- void setWeight(double value)

Mengatur berat atribut baru.

- java.lang.String toString()

Pengembalian deskripsi atribut dalam format ARFF.

- int type()

Mengembalikan jenis atribut sebagai integer.

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

Mengembalikan representasi string dari jenis atribut.

- static java.lang.String typeToString(int type)

Mengembalikan representasi string dari jenis atribut.

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

Mengembalikan representasi string jenis atribut.

- static java.lang.String typeToStringShort(int type)

Mengembalikan representasi string jenis atribut.

- java.lang.String value (int valIndex)

Mengembalikan nilai atribut nominal atau tali.

- double weight()

Mengembalikan berat badan atribut itu

ID3 adalah klas yang digunakan untuk membuat 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.

- double classifyInstance(Instances instance)

Mengklasifikasikan satu data yang dibedakan dengan menggunakan pohon klasifikasi.

- double[] distributionForInstance(Instances instance)

Menghitung distribusi klas instanc menggunakan pohon klasifikasi.

- Capabilities getCapabilities()

Mengembalikan detail klasifikasi.

- java.lang.String getRevision()

Mengembalikan String revisi.

- TechnicalInformation getTechnicalInformation()

Mengembalikan sebuah instanc dari objek TechnicalInformation, yang berisi informasi rincian tentang latar belakang teknis klas ini.

- java.lang.String globalInfo()

Mengembalikan string yang menjelaskan klasifikasi.

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

Metoda utama untuk klas ini.

- java.lang.String toSource(Instances classNam)

Mengembalikan string yang menggambarkan klasifikasi.

- java.lang.String toString()

Mencetak pohon klasifikasi menggunakan metoda toString.

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

Constructor:

- ID3()

Method:

- java.lang.String binarySplitsTipText()

Mengembalikan tipe tip untuk properti ini.

- void buildClassifier(Instances instances)
Menghasilkan classifier.
- double classifyInstance(Instances instance)
Mengklasifikasikan satu data.
- java.lang.String confidenceFactorType()
Mengembalikan tipe faktor untuk properti ini.
- double[] distributionForInstance(Instances instance)
Pengembalian probabilitas kelas untuk sebuah data.
- java.util.List numRates()
Pengembalian penerapan ukuran.
- boolean getBinarySplits()
Dapatkan nilai binarySplits.
- Capabilities getCapabilities()
Mengembalikan kapabilitas dari kelas ini.
- float getConfidenceFactor()
Mengembalikan nilai *confident*.
- double getMallowsRate(java.lang.String additionalName)
Mengembalikan nilai bobot suai nama.
- int getMinNumObj()
Dapatkan nilai minNumObj.
- int getNumFolds()
Dapatkan nilai numFolds.
- java.lang.String getOptions()
Mendapat pengaturan saat ini.
- boolean getReducedErrorPruning()
Dapatkan nilai reducedErrorPruning.
- java.lang.String getRevision()
Mengembalikan string revisi.
- boolean getSaveInstancesData()
Periksa apakah contoh data disimpan.

- int g tS d()
Dapatkan nilai s d
- bool an g tSubtr Raising()
Dapatkan nilai subtr Raising.
- T chnicalInformation g tT chnicalInformation()
M ng mbalikan s buah instanc dari obj k T chnicalInformation, yang b risi informasi rinci tentang latar b lakang t knis k las ini.
- bool an g tUnprun d()
m ng c k apakah dilakukan *tree pruning*.
- bool an g tUs Laplac ()
Dapatkan nilai us Laplac .
- java.lang.String globalInfo()
M ng mbalikan string yang m nj laskan classifi r.
- java.lang.String graph()
P ng mbalian Grafik m nggambarkan pohon.
- int graphTyp ()
M ng mbalikan j nis grafik classifi r.
- static void main(java.lang.String[] argv)
M tod utama untuk m nguji k las ini.
- doubl m asur NumL av s()
M ng mbalikan jumlah daun.
- doubl m asur NumRul s()
M ng mbalikan s jumlah aturan.
- doubl m asur Tr Siz ()
M ng mbalikan 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 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()`
Mengembalikan string yang merupakan representasi klasifikasi.
- `java.lang.String unprintableTipText()`
Mengembalikan teks tip untuk properti ini.
- `java.lang.String useLaplaceTipText()`
Mengembalikan teks tip untuk properti ini.

NumericToNominal adalah kelas yang digunakan untuk mengubah nilai numerik menjadi nominal.

Constructor:

- `NumericToNominal()`

Method:

- `java.lang.String[] getOptions()`
Mengembalikan pengaturan dari filter.
- `java.lang.String getRevision()`
Mengembalikan revisi.
- `java.lang.String globalInfo()`
Mengembalikan string yang berisi deskripsi dari kelas tersebut.
- `static void main(java.lang.String[] args)`
Menjalankan filter dengan input parameter tertentu.
- `void setAttributIndices(java.lang.String value)`
Melakukan penyaringan untuk memilih atribut yang akan difilter.
- `void setAttributIndicesArray(int[] value)`
Melakukan penyaringan untuk memilih atribut yang akan difilter.
- `boolean setInputFormat(Instances instances)`
Melakukan penyaringan untuk input data.
- `void setOption(String[] options)`
Melakukan penyaringan pada pengaturan.

2.5 Graph iz

[4] Graphviz merupakan rangkat lunak *open source* untuk visualisasi grafik. Dengan menggunakan graphviz, visualisasi grafik dapat dibuat dengan menulis kod . Atribut gambar yang disediakan oleh graphviz adalah *node shapes* dan *labels*. Dengan memanfaatkan dua atribut gambar tersebut, gambar yang dihasilkan dapat diubah menjadi:

- Banyaknya bentuk untuk setiap node
- Banyaknya warna untuk setiap node dan edge
- Pemberian label pada setiap node dan edge

Bentuk umum untuk setiap node adalah lingkaran dengan diameter 0.75 dan tinggi 0.5 serta dibatasi oleh garis. Untuk bentuk umum yang lain tiga diantaranya adalah kotak, bulat, dan *plaintext*. Sedangkan untuk ukuran node, dapat dilakukan perubahan dengan cara mengubah nilai atribut dari diameter dan tinggi.

Warna untuk node dan edge secara umum adalah hitam. Node dan edge dapat diubah warnanya dengan cara mengubah nilai atribut dari warna. Sedangkan untuk memberi warnai bagian dalam dari node, dapat dibatasi oleh *style filled*.

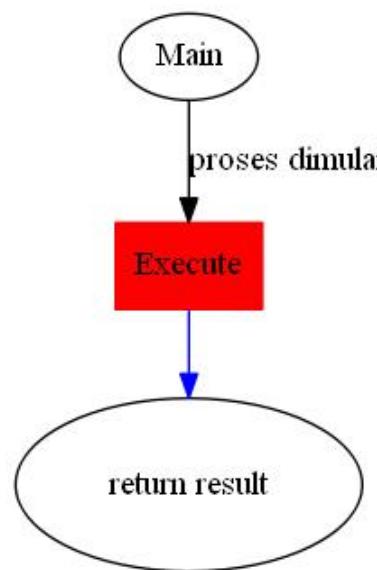
Pemberian label dapat dilakukan dengan cara mengisi nilai atribut label pada objek yang akan dibatasi oleh label.

Berikut contoh kod yang dapat dijadikan input untuk aplikasi graphviz:

```

1: digraph G{
2: Main
3: Ex cut [shape=box, color=r, style=filled]
4: Main -> Ex cut [label l="proses dimulai"]
5: Output [label l="result", width=2, height=1]
6: dg [color=blue]
7: Ex cut -> Output
8: }
```

Maka hasil yang diperoleh dari rangkat lunak graphviz dapat dilihat pada gambar 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 nai 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*, namun terdapat data-data yang berada diluar Bandung seperti Jakarta akan dibuang.

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* ber nilai 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 lahan 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+7. Kemudian, data akan diubah menjadi empat atribut, yaitu:

- Bulan, atribut ini akan menunjukkan bulan ketika user KIRI memanggil *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 memanggil *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 kedua dan spasi.
- Hari, atribut ini akan menunjukkan hari ketika user KIRI memanggil *action FINDROUTE*, dengan rangkaian nilai antara sembilan sampai minggu. Nilai tersebut dapat dipilih dengan cara melakukan panggilan 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 temp stamp yang berada di antara spasi dan titik dua.

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

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

- Latitud ke bantuan, atribut ini berisi nilai latitud dari lokasi ke bantuan yang dipilih oleh user. Nilai tersebut dapat dipilih dengan cara mengambil nilai string setelah koma yang pertama.
- Longitud ke bantuan, atribut ini berisi nilai longitudo dari lokasi ke bantuan 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 mampat bagian, agar program dapat membaca data tersebut lebih mudah.

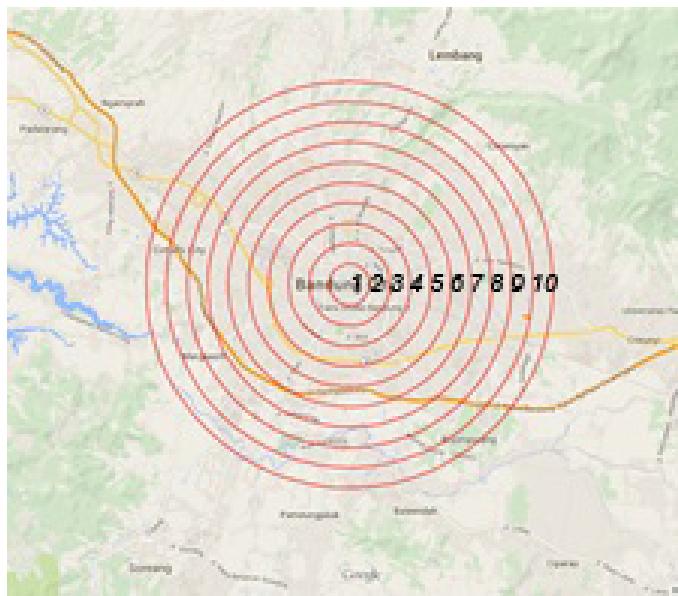
Dari analisis diatas, banyak atribut dari tabel *statistics* akan menjadi diperlukan, yaitu:

- Bulan
- Tahun
- Hari
- Jam
- Latitud Ke bantuan
- Longitud Ke bantuan
- 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	07	-6.8972513	107.6185574	-6.91358	107.62718
02	2014	Sabtu	07	-6.8972513	107.6385574	-6.91358	107.62718
02	2014	Sabtu	07	-6.90598	107.59714	-6.90855	107.61082
02	2014	Sabtu	07	-6.9015366	107.5414474	-6.88574	107.53816
02	2014	Sabtu	07	-6.90608	107.61530	-6.89140	107.61060
02	2014	Sabtu	07	-6.89459	107.58818	-6.89876	107.60886
02	2014	Sabtu	07	-6.89459	107.58818	-6.86031	107.61287

Tab 1 3.2: Contoh hasil data transformasi



Gambar 3.1: *Classification* pada daerah Bandung. Angka pada gambar merupakan nomor klasifikasi tiap daerah

Agar dapat dipilih *decision tree* mengenai lokasi keberangkatan dan tujuan dari user KIRI, maka atribut lokasi yang akan digunakan adalah nilai latitud dan longitud dari lokasi keberangkatan dan tujuan. Karena atribut lokasi ada empat, maka akan dilakukan pengurangan dari keempat atribut untuk meningkatkan akurasi serta tingkat fisik proses *data mining*.

Nilai *latitude* serta *longitude* dari data lokasi keberangkatan dan tujuan akan diubah menjadi nilai yang menunjukkan apakah daerah lokasi keberangkatan dan tujuan tersebut berada di jalanan ke luar dari Bandung, menuju Bandung, atau menuju daerah yang sama. Hal ini dilakukan agar dipilih data perbandingan dengan rakan penduduk, apakah mereka lebih banyak yang ke luar dari Bandung atau sebaliknya atau bahkan menuju daerah yang sama berdasarkan waktu tersebut. Untuk menentukan hal tersebut, maka akan dibutuhkan klasifikasi daerah agar mudah dilakukan pertemuan apakah user akan berangkat ke Bandung atau tidak. *Classification* daerah yang ditentukan sebagai berikut dapat dilihat pada gambar 3.1.

Pertemuan *classification* tersebut berdasarkan titik pusat, yaitu $-6.916667, 107.6$ ¹ dalam latitud dan longitud. Kedua titik dibagi menjadi sepuluh daerah yang memiliki perbedaan radius sebesar 1 km, sehingga diameter pertama adalah 2 km, diameter kedua adalah 4 km, dan seterusnya, untuk daerah terakhir (yaitu daerah 10) akan memiliki diameter 20 km.

Suatu lokasi atau titik latitud longitud dapat diketahui berada pada daerah yang mana dengan cara menghitung jarak titik tersebut dengan titik pusat yang sudah ditentukan (yaitu $-6.916667, 107.6$) dengan menggunakan rumus Haversine. Jika jarak yang dipilih lebih kecil sama dengan 1 km, maka berada di daerah pertama, sedangkan jika jarak yang dipilih lebih kecil sama dengan 2 km dan lebih besar dari 1 km, maka berada di daerah kedua, dan seterusnya, dan untuk daerah terakhir (yaitu daerah 10) titik akan memiliki jarak lebih kecil sama dengan 10 km dan lebih besar dari 9 km dengan titik pusat. Jika suatu titik memiliki jarak terhadap titik pusat lebih dari 10 km, maka akan menjadi daerah luar Bandung.

¹http://tools.wmflabs.org/geohack/geohack.php?pagename=Bandung¶ms=6_55_S_107_36_E_region:ID-JB_type:city

S tetapi lokasi kota berasarkan dan lokasi tujuan ditentukan dari arahnya, dapat ditentukan apakah user tersebut berada di pusat Bandung atau tidak. Jika arah dari lokasi kota berasarkan 1 berada di sebelah barat dari lokasi tujuan, maka user tersebut berada di pusat Bandung. Kecuali, jika arah dari lokasi kota berasarkan 1 berada di sebelah timur dari lokasi tujuan, maka user tersebut tidak berada di pusat Bandung. Sedangkan, jika lokasi kota berasarkan dan lokasi tujuan berada di arah yang sama, maka user tersebut berada di pusat Bandung.

Dengan adanya perhitungan jarak dan posisi tujuan dari arah Bandung, nilai latitud dan longitud dari lokasi kota berasarkan dan tujuan dapat dibuang dan diganti oleh atribut berada di pusat Bandung dengan tipe data integer. Jika isi dari atribut tersebut ber nilai 1, maka user tersebut berada di pusat Bandung sedangkan nilai -1 berarti user tidak berada di pusat Bandung, dan jika nilai atribut tersebut adalah 0, maka user tersebut berada di lokasi kota berasarkan dan tujuan pada arah yang sama. Contoh hasil data setelah dilakukan transformation terhadap latitud dan longitud dapat 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 kota berasarkan 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 selanjutnya. 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 fungsional, pemodelan perangkat lunak, diagram use case, scenario, diagram klasifikasi 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 menimba dan membaca input text yang sudah disiapkan
2. Dapat melakukan preprocessing data sesuai dengan yang dijelaskan pada bab analisis data
3. Dapat melakukan proses data mining, ID3 dan C4.5
4. Dapat melakukan visualisasi hasil dari data mining yang dipilih

Pemodelan Perangkat Lunak *Data Mining Log Histori KIRI*

Perangkat lunak *data mining log* histori KIRI akan mendapat input data dalam format .csv. Setelah program mendapatkan input dan user menekan tombol proses, maka data tersebut akan diubah terlebih dahulu sesuai pada bab analisis data(bab 3.1) dengan melakukan proses *data transform* dan menghasilkan data dalam format seperti pada tabel 3.3.

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

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

Karena data yang dipilih sudah dalam bentuk csv, maka pada pertemuan ini, tidak akan menggunakan sistem database.

Ketika tombol proses diklik, maka data tersebut akan diproses. Proses yang pertama yang akan dilakukan adalah melakukan *load* data dari file .data csv akan dibaca dengan menggunakan 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. Proses *transform* tersebut memiliki tahap sebagai berikut:

1. Mengubah waktu dari UTC menjadi GMT+7 pada string data input array ketiga (yaitu atribut tanggal).
2. Mengambil atribut tanggal k mudian memisahkan nilai tersebut dengan spasi sebagai tanda pengisian, 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 pemisahan nilai string dengan garis miring sebagai tanda pemisah, maka akan dipilih tiga nilai yaitu bulan, tanggal, dan tahun, namun nilai yang akan diambil hanya dua, yaitu bulan dan tahun.
4. Pada nilai jam, dilakukan pemisahan nilai string dengan titik dua sebagai tanda pemisah, maka akan dipilih 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 pemisahan nilai string dengan garis miring sebagai tanda pemisah, maka akan dipilih tiga nilai yaitu lokasi awal, lokasi tujuan, dan banyak jalur.
6. Pada nilai lokasi awal dan lokasi tujuan, akan dilakukan pemisahan nilai string dengan koma sebagai tanda pemisah, maka akan dipilih 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 -1 atau 0 atau 1.

8. menggabungkan nilai-nilai tersebut dalam satu array, yaitu array dengan tipe int (dengan nilai bulan, tahun, hari, jam dan menuju Bandung).

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

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

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

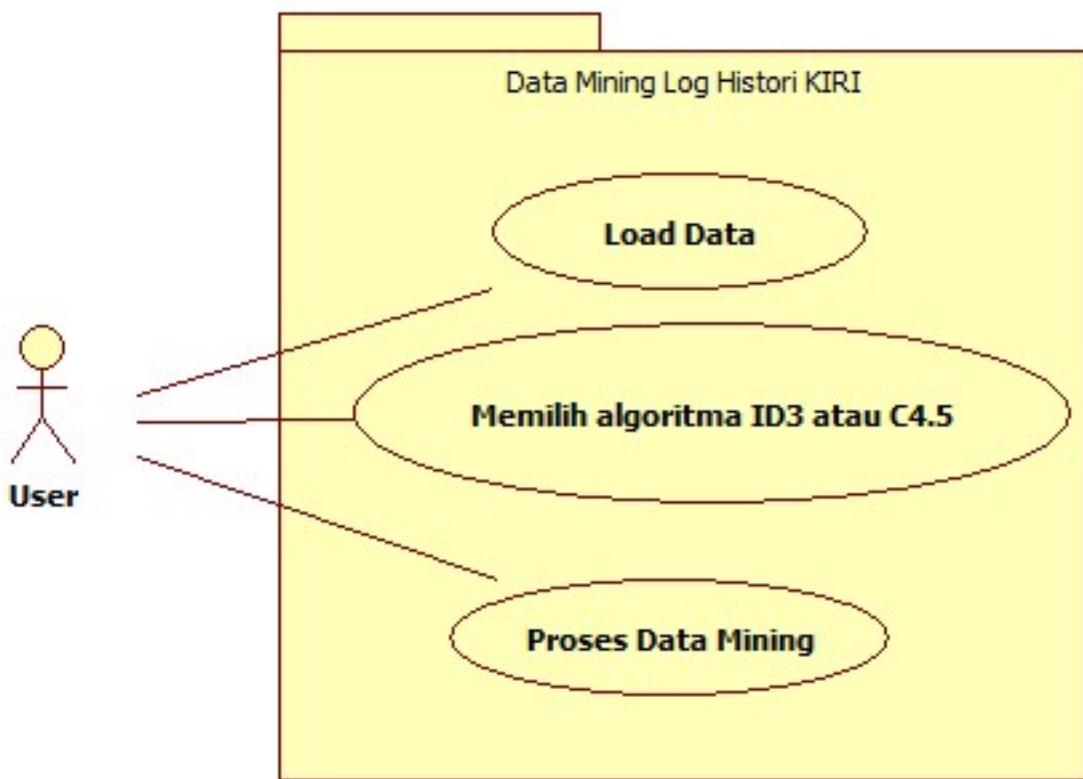
1. Program akan memuat data dan melakukan *processing data*
2. Program akan menjalankan algoritma pembuat *decision tree*
3. Program akan membuat grafik dari hasil algoritma *decision tree*
4. Program akan menampilkan grafik *decision tree*

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

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

- Melakukan load data yang digunakan sebagai input data dengan cara 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 P rangkat Lunak Data Mining Log Histori KIRI

Tab 1 3.5: Sk nario M lakukan load Data

Nama	Load data
Aktor	User
D skripsi	Masukan alamat data yang akan dijadikan sebagai input program
Kondisi awal	Textbox belum t risi
Kondisi akhir	Textbox sudah t risi dengan alamat data
Sk nario utama	User m masukan alamat data pada textbox
Eks spi	Data tidak dit mukan

Tab 1 3.6: Sk nario M lakukan Data Mining

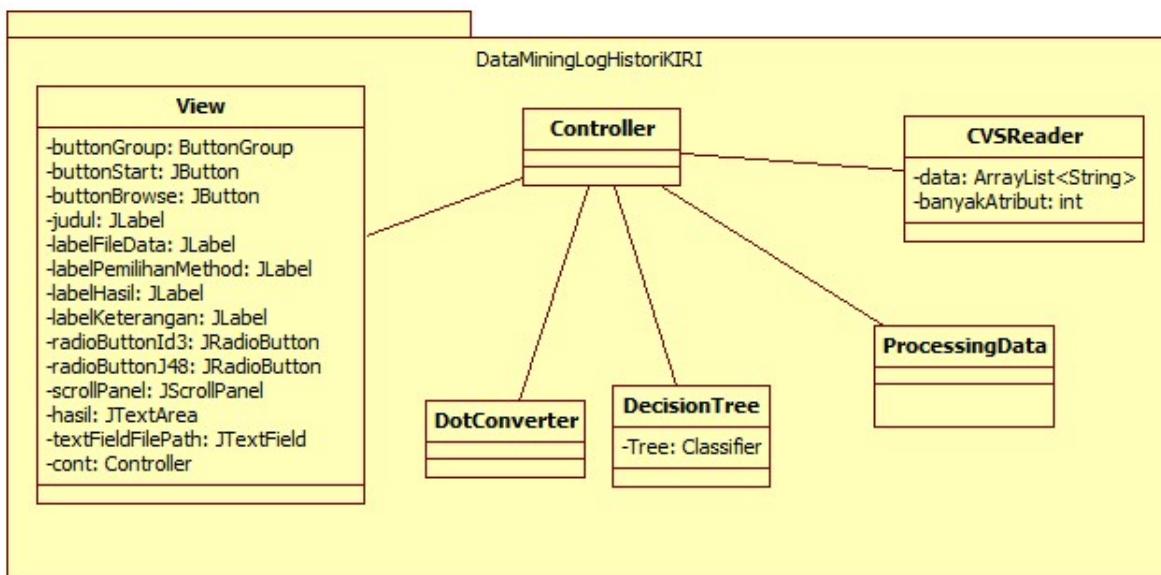
Nama	Pros s Data Mining
Aktor	User
D skripsi	M n kan tombol pros s pada interface
Kondisi awal	Textbox belum t risi
Kondisi akhir	Textbox sudah t risi dengan hasil data mining
Sk nario utama	User m n kan tombol pros s
Eks spi	Data tidak dit mukan atau data tidak dapat dipros s

Tab 13.7: Skenario Milih Algoritma yang Akan Digunakan

Nama	Milih algoritma ID3 atau C4.5
Aktor	User
D skripsi	Us r milih algoritma yang akan dipakai
Kondisi awal	<i>RadioButton</i> t rpilih pada ID3
Kondisi akhir	<i>RadioButton</i> t rpilih pada ID3 atau C4.5
Skenario utama	User milih algoritma yang akan digunakan
Ekspsi	Tidak ada

3.2.2 Diagram kelas Perangkat Lunak Data Mining Log Histori KIRI

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



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

Berikut deskripsi kelas diagram *class*:

- View, merupakan kelas untuk mengatur tampilan antar muka.
- Controller, merupakan kelas untuk mengatur viw dan modul k tika program dijalankan.
- CSVReader, merupakan kelas yang memiliki metode untuk membaca file dengan 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 buttonStartActionP rform d() bila diklik.
 - * JButton buttonBrows , merupakan sebuah tombol yang dapat menggil method buttonBrows ActionP rform d() 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 thod, 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.
- * JRadioButton 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 string 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 ProcesssingData, merupakan kelas yang memiliki method untuk melakukan *preprocessing data*.
 - Method
 - * public ProcesssingData(), merupakan konstruktor dari kelas ProcesssingData.
 - * 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 preprocessData(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 akurasi 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 calculatePrecision(Instances data), digunakan untuk mendapatkan nilai akurasi dari *decision tree* yang dihasilkan.
 - * public String id3(Instances data), digunakan untuk membuat *decision tree* dengan menggunakan metode ID3 dari API Weka.
 - * public String j48(Instances data), digunakan untuk membuat *decision tree* dengan menggunakan metode 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 isi dari 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.

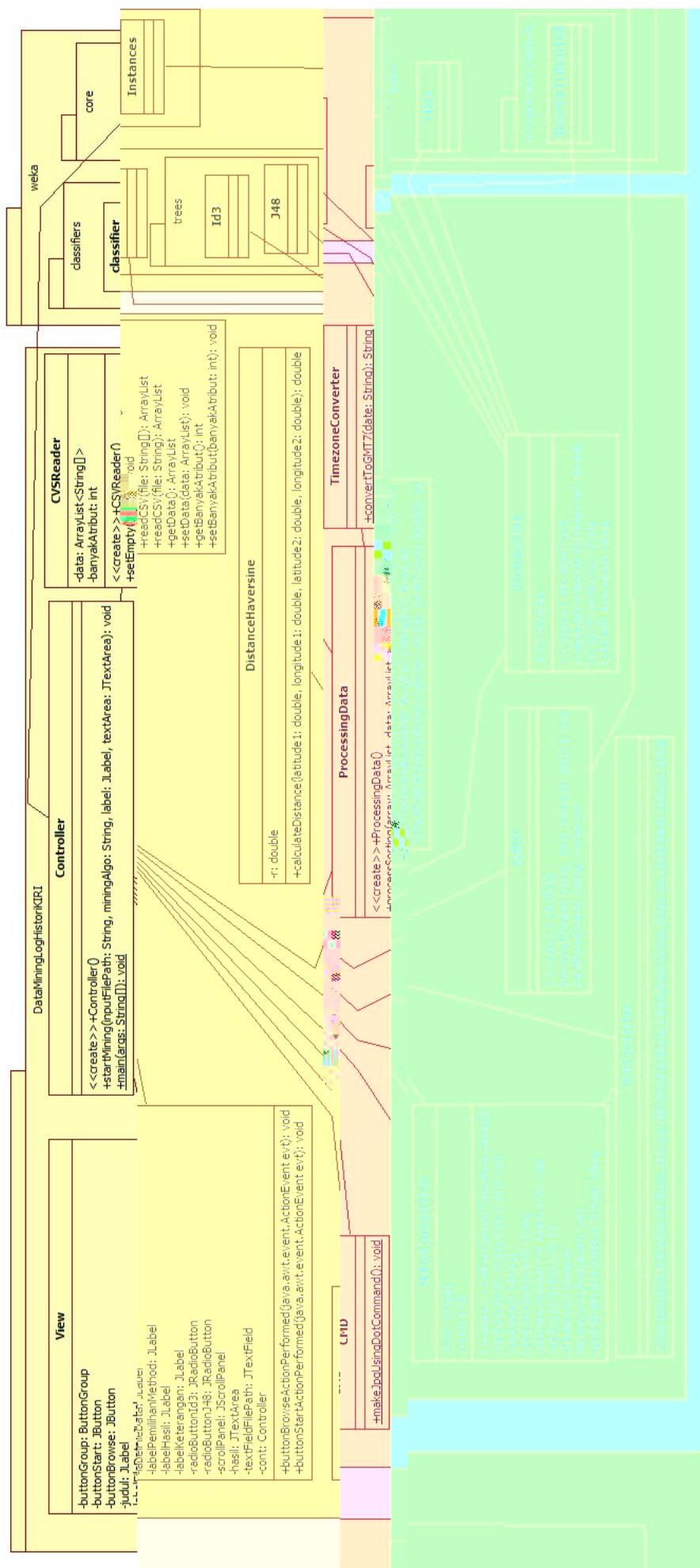
- k las CMD, m rupakan k las yang digunakan untuk m manggil *command prompt*.

– 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. Kedua, user akan menulis *file path* atau memilih (dengan menggunakan tombol *browse*) *input file* pada JXtFileSelector agar mampu membuat *decision tree* (tahap pertama). Setelah memilih file dan mode, user akan menekan tombol start, dan kelas Viw akan memanggil *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 dipisahkan dan mengambil *record* dengan cara memanggil *method proc ssSorting* pada kelas ProcssingData dan mengembalikan *ArrayList* dengan data yang sudah dipisahkan (tahap 7-8). Kemudian data tersebut akan dilakukan *preprocessing data* dengan cara memanggil *method processData* dari kelas ProcssingData (tahap 9).

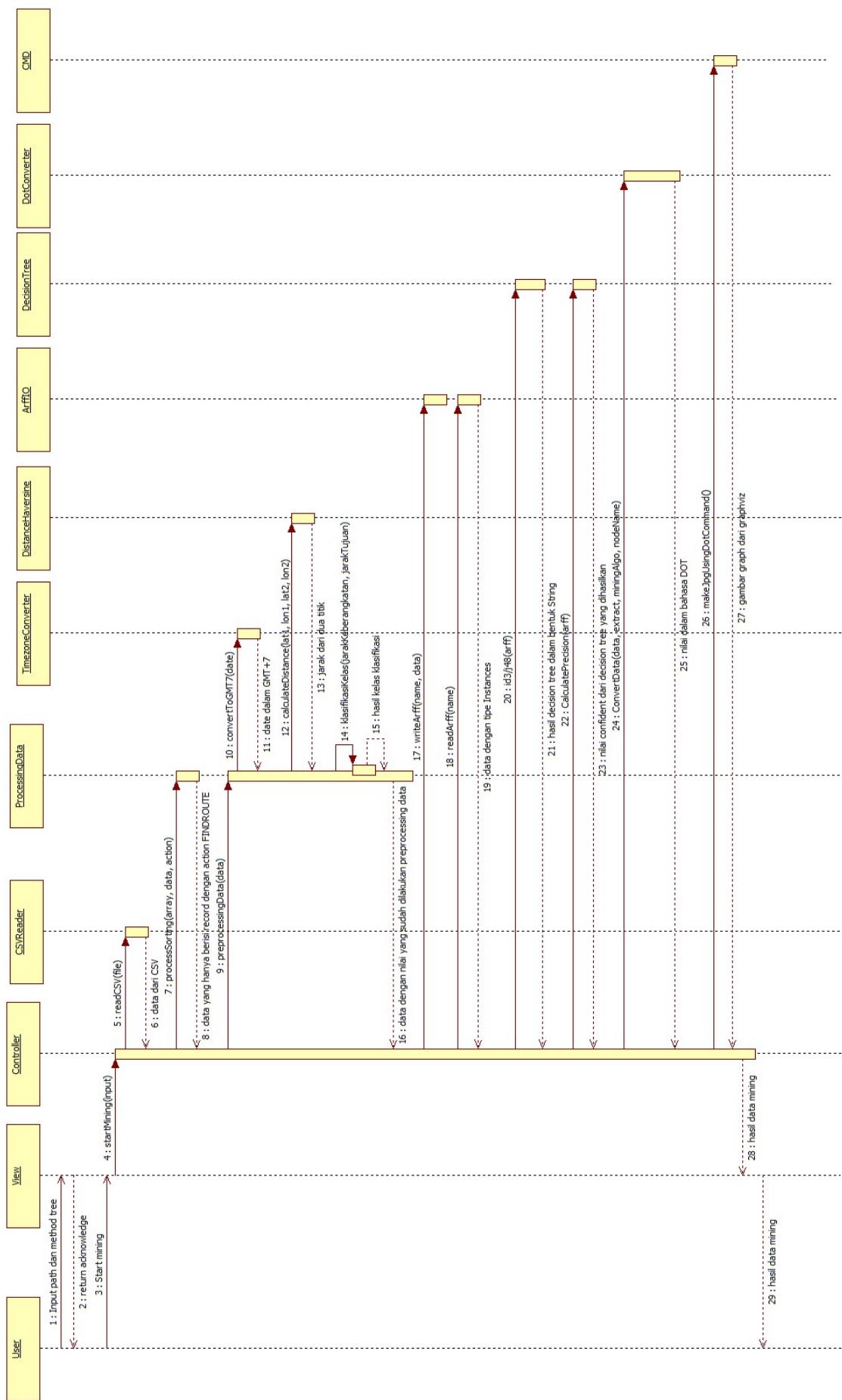
Ketika *method processData* dijalankan, perlu mengubah nilai waktu dari UTC menjadi $GMT+7$ dengan cara memanggil *method convertGMT7* dari kelas TimeZoneConverter dan mengembalikan nilai berupa Date (tahap 10-11). Setelah nilai waktu diubah, dipersiapkan perhitungan jarak antara dua titik dengan cara memanggil *method calculateDistance* dari kelas DistanceHaversine dan mengembalikan nilai double yang berisi jarak dari dua titik (tahap 12-13). Kemudian dipersiapkan klasifikasi kelas dari jarak yang sudah dihasilkan dengan cara memanggil *method classify* kelas 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 mengembalikan *decision tree* dalam bentuk String (tahap 20-21). Setelah *decision tree* dibuat, perlu dicari nilai akurasi yang dipilih dari *decision tree* tersebut dengan cara memanggil *method calculatePrecision* dan nilai akurasi 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 DotConverter dan mengembalikan 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 memperlihatkan hasil gambar *decision tree* yang sudah dipilih tersebut dengan mengembalikan nilai String *decision tree* pada kelas Viw yang akan ditampilkan di JTxtArea (tahap 28-29).

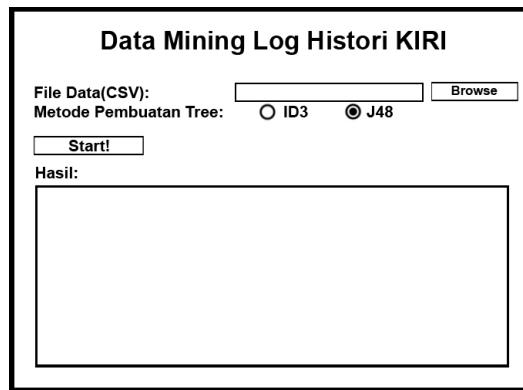


Gambar 4.2: Diagram Class Perangkat Lunak Data Mining Log Histori KIRI

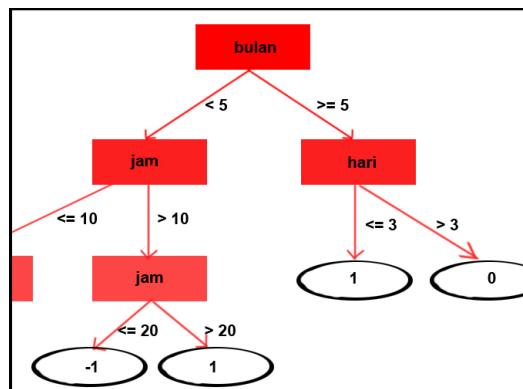
4.1.3 Perancangan Desain Antar Muka

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) disediakan JTextField dan JButton yang digunakan untuk memilih file, JRadioButton yang digunakan untuk memilih metode pembuatan *decision tree*, JTextField yang digunakan untuk memperlihatkan hasil *decision tree* yang dipilih dalam bentuk String, serta JButton yang kedua (dengan label Start) yang digunakan untuk mulai proses *data mining*. 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

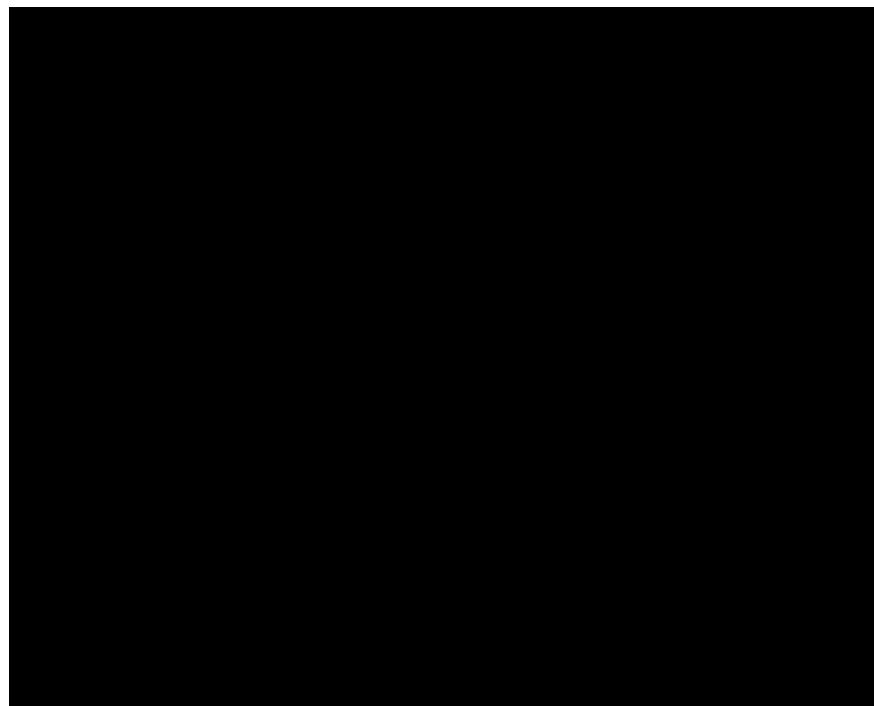
Lingkungan perangkat lunak dan perangkat keras yang digunakan untuk membangun dan menguji aplikasi *data mining* ini adalah:

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

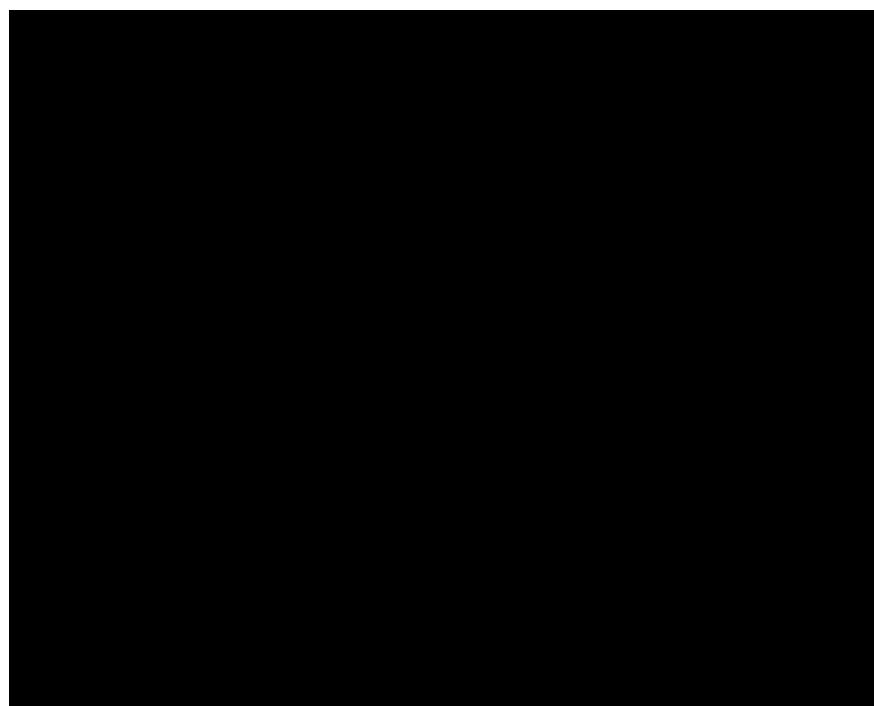
Pada aplikasi *data mining* ini, terdapat dua form. Form pertama berfungsi untuk memilih file CSV yang akan dilakukan *data mining*, memilih metode pembuatan *decision tree*, serta menunjukkan hasil *decision tree* yang dihasilkan dalam bentuk String. Sedangkan untuk form kedua, berfungsi untuk mvisualisasikan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

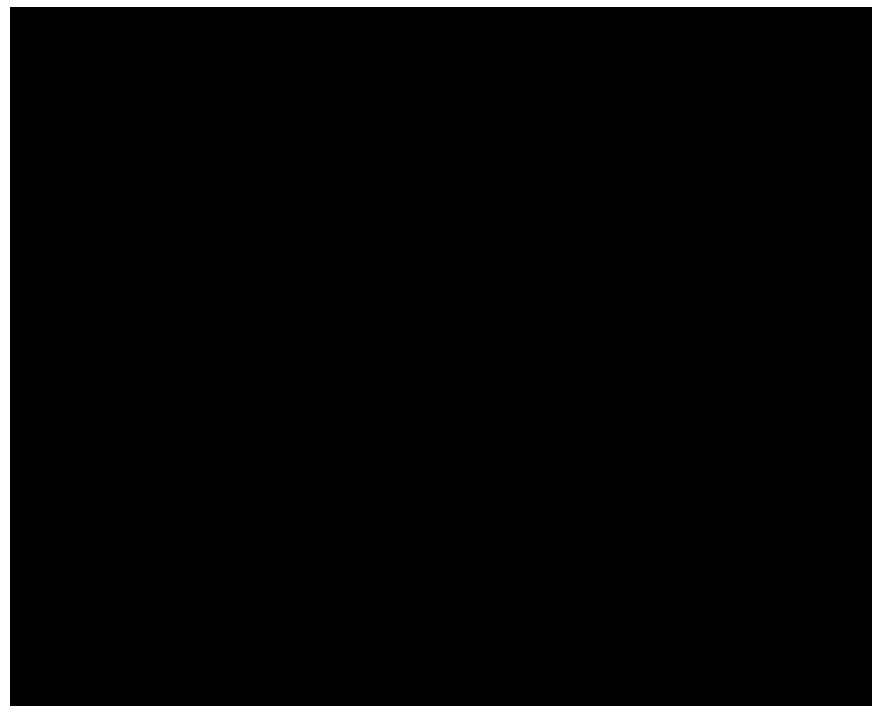


Gambar 5.1: Tampilan Form Awal Aplikasi *Data Mining*

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.

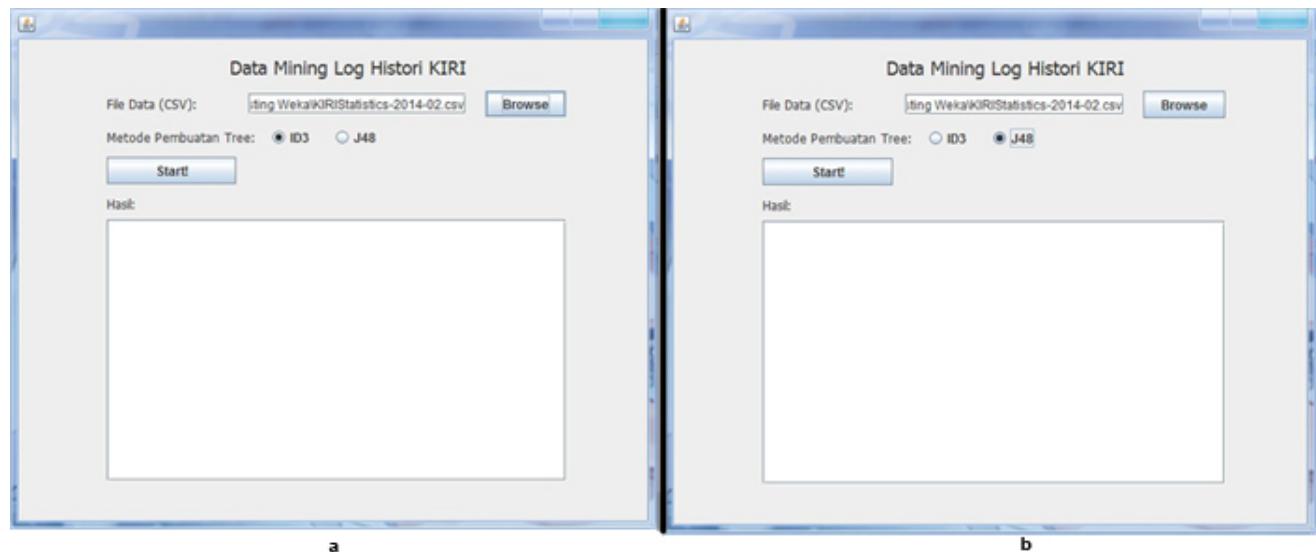


Gambar 5.2: Tampilan File Selector untuk Memilih File CSV



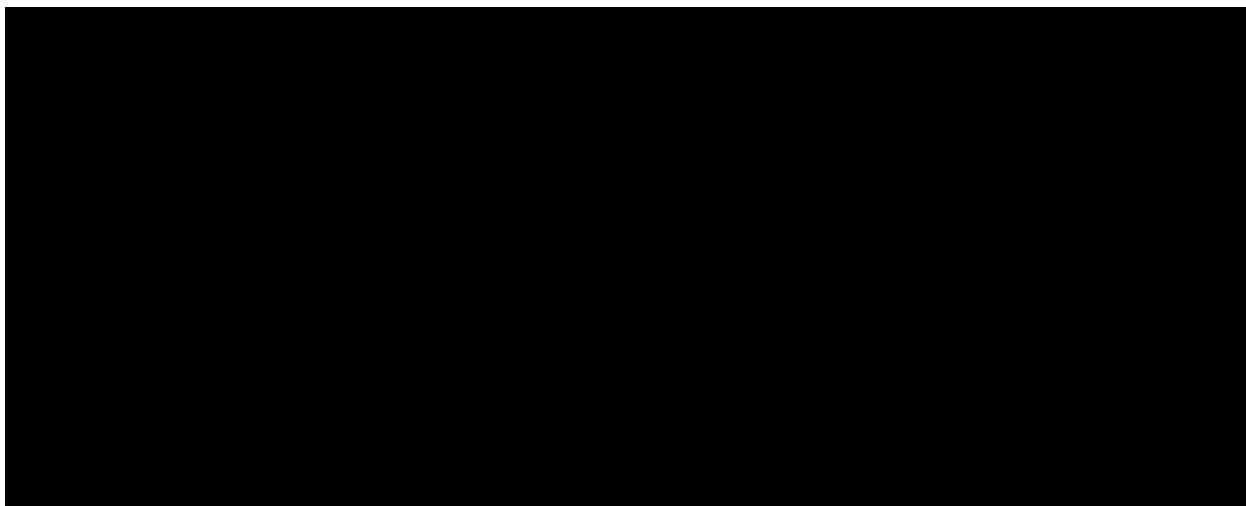
Gambar 5.3: Tampilan Form saat Membuat File CSV

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 ke salahannya jika ada.

Hasil Pengujian

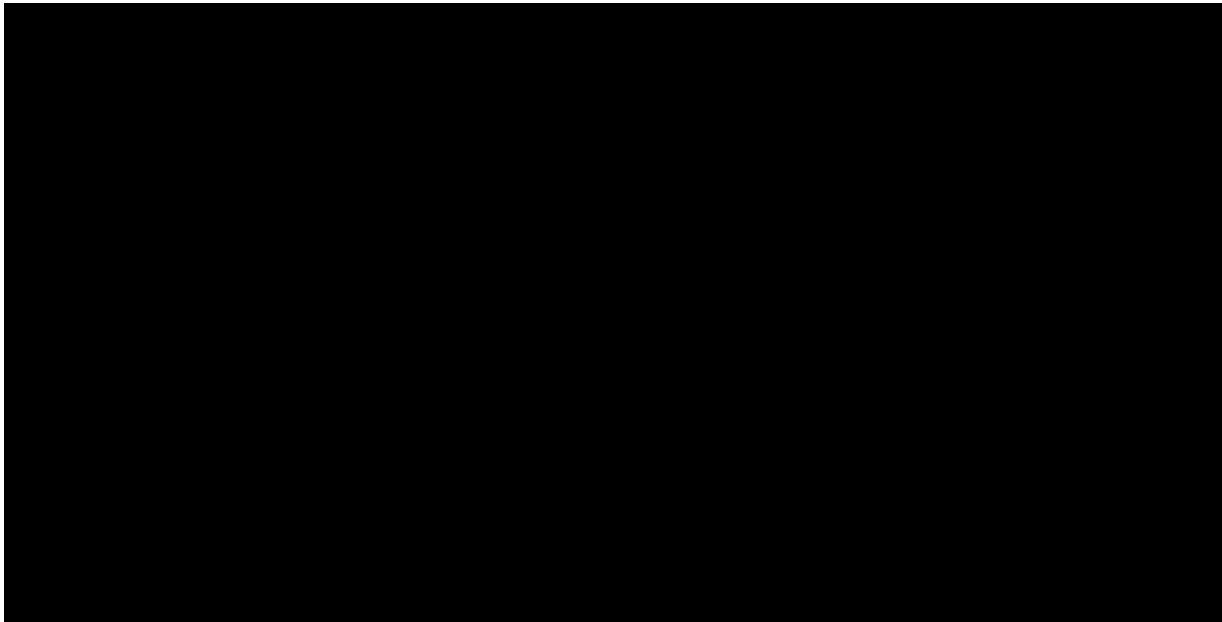
Dibuat sebuah *test case* yang berisi 20 data log histori KIRI dengan data pada tabel 5.1

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

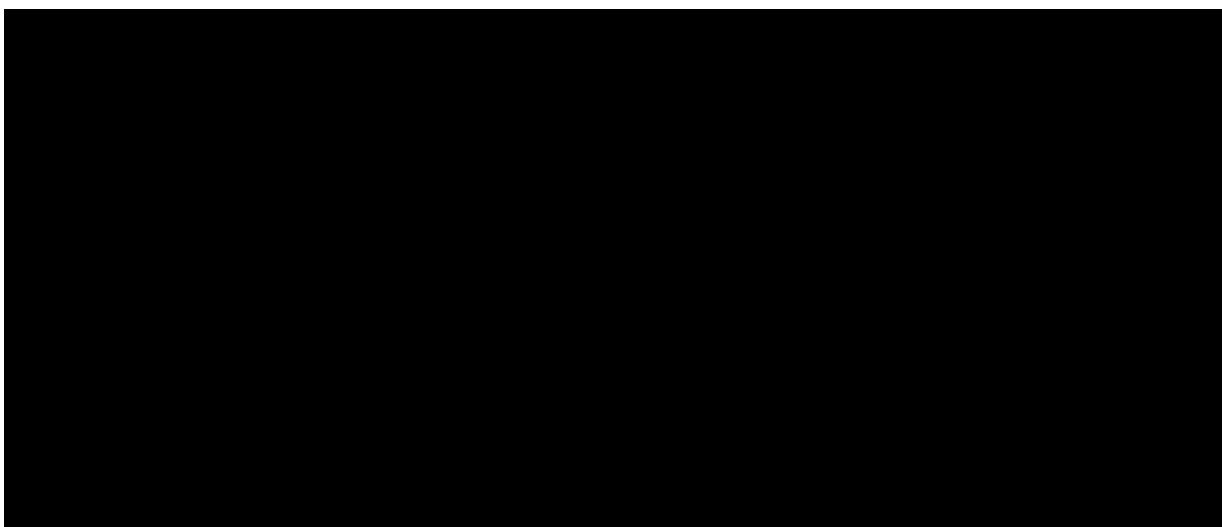
Tab 15.1: Data untuk Test Case Aplikasi Data Mining

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



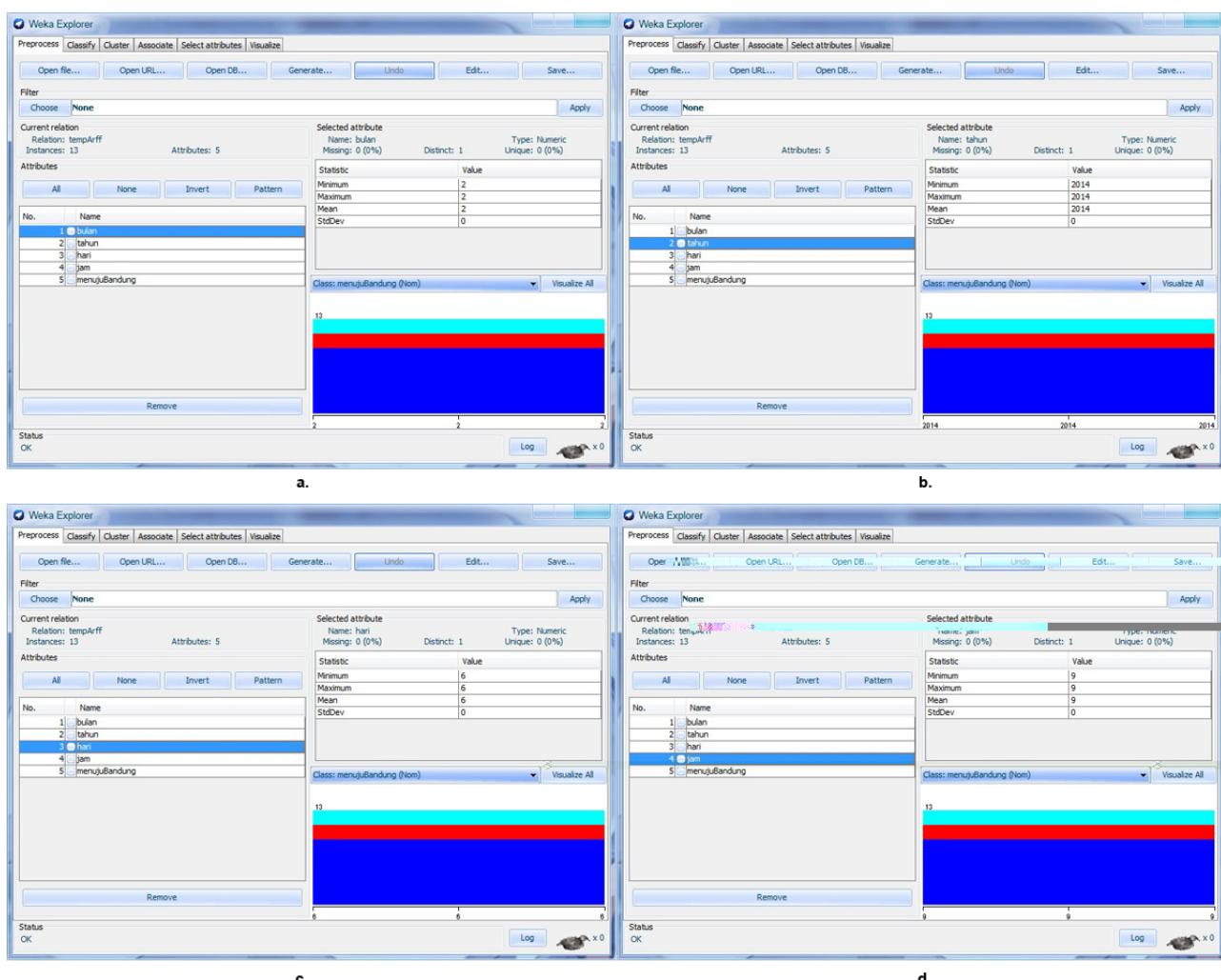
Gambar 5.6: Pengujian Mengambil Data CSV



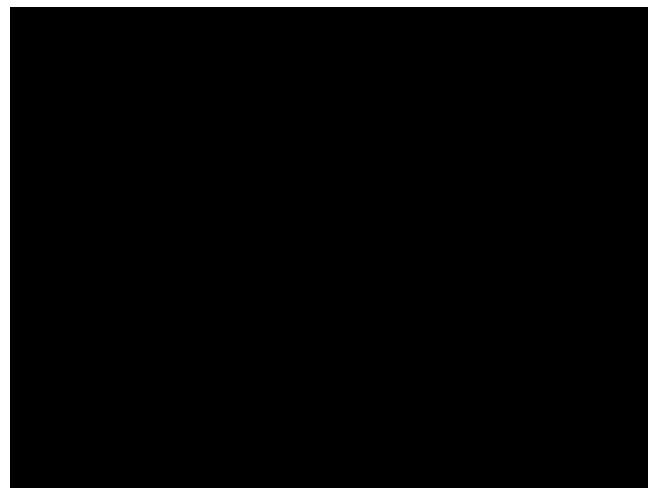
Gambar 5.7: Pengujian Data Selection untuk Mengambil Data dengan Action FINDROUTE

Gambar pertama memperlihatkan bahwa data pertama yang diproses oleh 21 array String dengan array pertama adalah atributnya sedangkan array selanjutnya adalah isi data yang diproses. 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

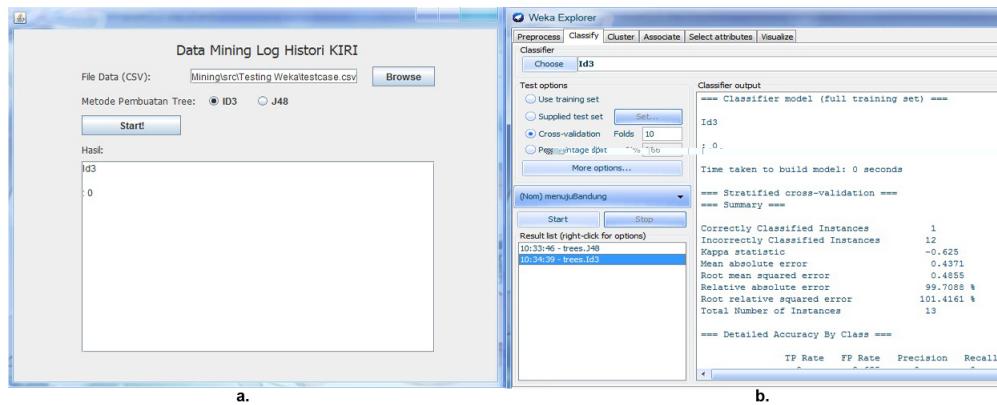
Terdapat dua tahap penting pada *preprocessing data*, yaitu pengubahan waktu dari UTC menjadi GMT+7 dan klasifikasi arah. Pada tahap pengubahan waktu dari UTC menjadi GMT+7, pada gambar 5.8 d, atribut jam yang dimiliki menjadi bernilai sembilan karena pada stcas, nilai atribut jam adalah dua. Pada tahap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

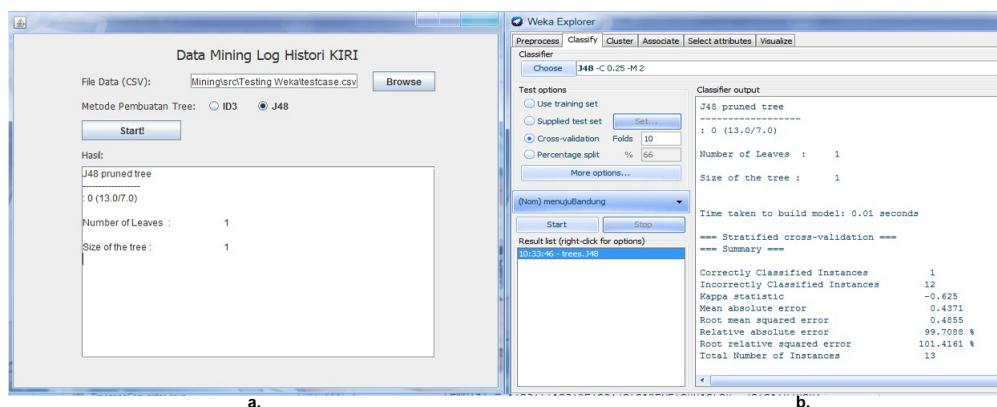
Tab 5.2: Hasil Pengaruh dan Klasifikasi

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.



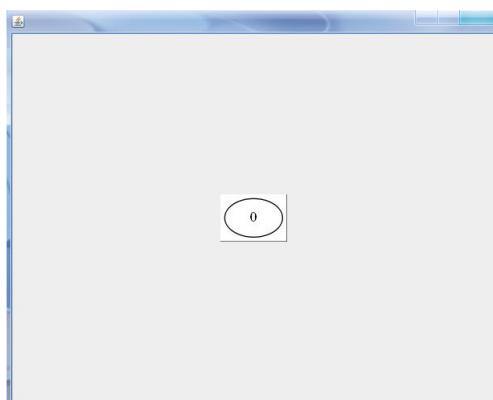
Gambar 5.10: Pengujian Pembuatan Decision Tree ID3. Gambar a merupakan hasil dari aplikasi data mining sedangkan gambar b merupakan hasil dari weka.



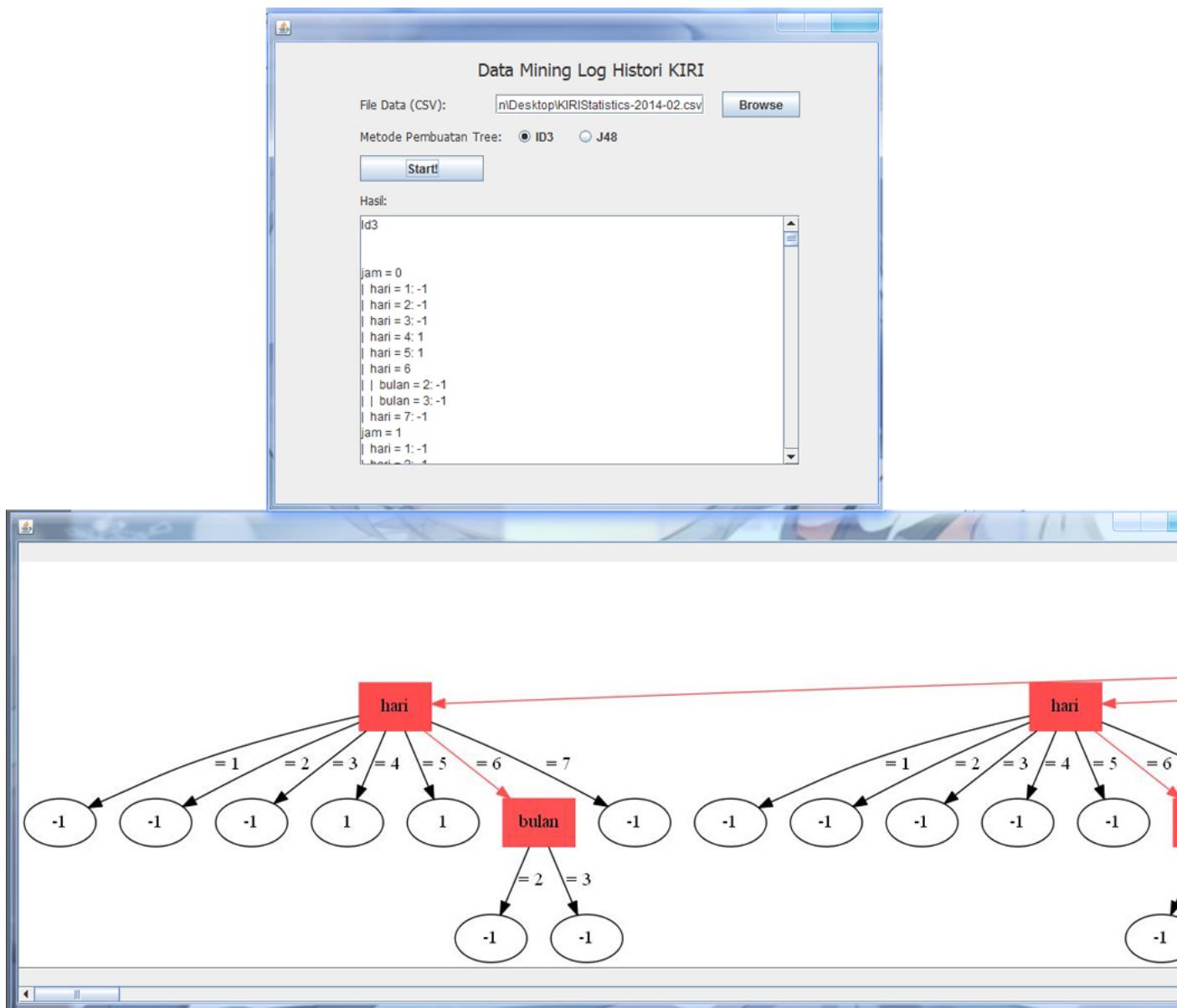
Gambar 5.11: Pengujian Pembuatan Decision Tree J48. Gambar a merupakan hasil dari aplikasi data mining sedangkan gambar b merupakan hasil dari weka.

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

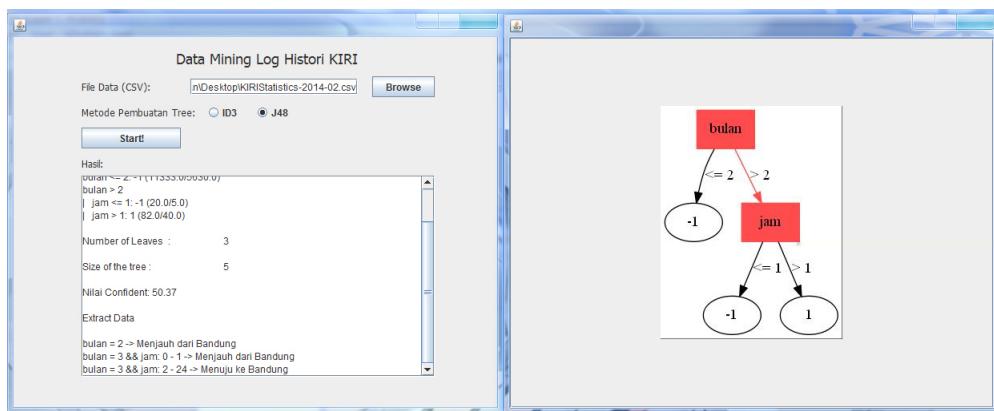
- Pengujian kompatibilitas: Mengubah *decision tree* dalam bentuk String menjadi bahasa DOT, akan dilakukan visualisasi *decision tree* dengan membuat graph dalam bahasa DOT, dapat dilihat pada gambar 5.12.



Gambar 5.12: Pengujian Hasil Visualisasi dengan Menggunakan Bahasa DOT.



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.

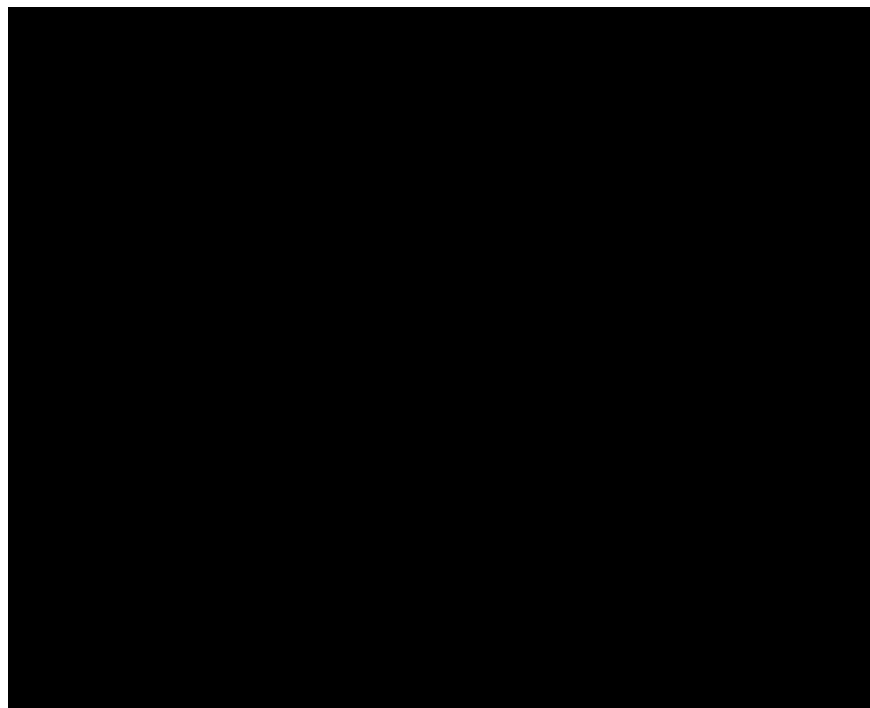
Pada kedua percobaan, terdapat perbedaan bulan, hal ini dikarenakan perubahan waktu dari UTC menuju GMT+7 sejak terealisasi hingga terdapat perubahan bulan atau tahun. Karena data pada bulan selanjutnya hanya tujuh jam, maka hasil pada bulan selanjutnya lebih baik tidak dianggap karena data tersebut tidak cukup untuk memperkirakan waktu satu bulan.

Hasil percobaan pembuatan decision tree pada metod ID3 mengalami overfitting karena menghasilkan klasifikasi pada hampir setiap kemungkinan, dapat disimpulkan hasil decision tree pada percobaan di bulan ini cukup jauh. Namun pada percobaan menggunakan J48, menghasilkan decision tree yang cukup berhasil namun tidak overfitting dan banyak user yang mengunjungi Bandung.

Nilai akurasi dari percobaan dengan metod ID3 adalah 38.22% sedangkan untuk percobaan dengan menggunakan J48 adalah 50.37%, dari sini dapat dinyatakan bahwa hasil decision tree dari J48 lebih tepat daripada ID3 dan menggunakan ID3 menghasilkan nilai akurasi yang belum memenuhi karena masih dibawah 50%.

Selain itu, dari kedua percobaan ini, diperoleh bahwa cukup sulit untuk membaca hasil decision tree terutama seperti hasil decision tree dengan menggunakan ID3. Oleh karena itu, akan ditambahkan fungsi agar decision tree lebih mudah dibaca. Program akan ditambah satu kelas baru yaitu SDForExtractData yang berfungsi untuk menyimpan data dan membuat simpulan dari setiap leaf yang dihasilkan.

Berikut hasil dari fungsi yang dibuat agar decision tree lebih mudah dibaca, dapat dilihat pada gambar 5.16.



Gambar 5.16: Hasil dari SDForExtractData

Dengan fungsi tersebut, diharapkan user dapat lebih mudah membaca *decision tree* yang dihasilkan.

5.3.3 Analisis Hasil Uji

Berdasarkan pengujian di atas, dapat disimpulkan bahwa

1. Menggunakan ID3 menghasilkan *decision tree* yang bersifat overfitting pada data log histori KIRI dengan *preprocessing data* dan klasifikasi yang sudah dilakukan pada bab 3.
2. Menggunakan 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 (J48 menghasilkan 50.37% sedangkan ID3 menghasilkan 38.22%).
3. Menggunakan ID3 belum menghasilkan nilai akurasi yang memuaskan, karena masih dibawah 50%.
4. Dari data log histori KIRI pada bulan Februari 2014, *decision tree* menggunakan J48 menyatakan bahwa user lebih suka mengunjungi Bandung daripada mendekati Bandung atau berangkat menuju daerah yang sama.

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 pengolahan rakan user KIRI di Bandung.

Pembuatan rangkatan lunak untuk melakukan *data mining* pada *log* histori KIRI dapat dilakukan.

Pola yang dipilih dari data *log* histori KIRI adalah *user* sering pergi menuju Bandung pada bulan Februari 2014.

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 lebih besar (jika dibandingkan dengan *decision tree* dengan metode J48) namun diharapkan lebih memiliki makna dan dapat menghasilkan nilai akurasi yang lebih besar.

DAFTAR REFERENSI

- [1] J. Han, M. Kamb r, and J. P i, *Data Mining : concepts and techniques, Third Edition*, p. 744. Els vi r, 2011.
- [2] C. V n ss, “Calculat distanc , b aring and mor b tw n latitud /longitud points,” 2002.
- [3] T. U. of Waikato, “W ka api,” 2009.
- [4] E. R. Gansn r, E. Koutsofios, and S. North, “Drawing graphs with dot,” January 2015.

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

LAMPIRAN B

THE SOURCE CODE

Listing B.1: Controll r.java

```
1 package DataMiningLogHistoriKIRIWithoutDateAndMinutes;
2
3 import java.awt.image.BufferedImage;
4 import java.io.BufferedReader;
5 import java.io.File;
6 import java.io.FileWriter;
7 import java.io.IOException;
8 import java.io.PrintWriter;
9 import java.util.ArrayList;
10 import javax.imageio.ImageIO;
11 import javax.swing.ImageIcon;
12 import javax.swing.JFrame;
13 import javax.swing.JLabel;
14 import javax.swing.JScrollPane;
15 import javax.swing.JTextArea;
16 import weka.core.Instances;
17
18 /**
19 *
20 * @author Jovan Gunawan
21 */
22 public class Controller
23 {
24     public Controller()
25     {
26     }
27
28     public void startMining(String inputFilePath, String miningAlgo, JLabel label, JTextArea textArea)
29     throws IOException
30     {
31         CSVReader csv = new CSVReader();
32         ArrayList<String[]> data = csv.readCSV(inputFilePath);
33         //ArrayList<String[]> data = csv.readCSV(file);
34
35         ArrayList<String[]> findRoute = new ArrayList<String[]>();
36         ProcessingData pd = new ProcessingData();
37         pd.processSorting(findRoute, data, "FINDROUTE");
38
39         //int maxMin digunakan untuk menyimpan nilai max dan min dari variable bulan dan tahun. Untuk
40         //ketentuan posisi array dapat dilihat di method preprocessing data
41         int[] maxMin = new int[4];
42         ArrayList<int[]> dataAfterPreprocessing = pd.preprocessingData(findRoute, maxMin);
43
44         ArffIO io = new ArffIO();
45         io.writeArff("tempArff", dataAfterPreprocessing);
46
47         Instances arff = io.readArff("temp.arff");
48         //arff.setClassIndex(arff.numAttributes() - 1);
49         DecisionTree dt = new DecisionTree();
50         String[] tempTreeDataResult;
51         System.out.println(miningAlgo);
52         if(miningAlgo.equals("id3"))
53         {
54             textArea.setText(dt.id3(arff));
55         }
56         else
57         {
58             textArea.setText(dt.j48(arff));
59         }
60         tempTreeDataResult = textArea.getText().split("\n");
61         textArea.setText(textArea.getText() + "\nNilai Confident: " + dt.calculatePrecision(arff) + "\n");
62         String[] treeDataResult;
63         System.out.println(tempTreeDataResult.length);
64         if(tempTreeDataResult.length < 8)
65         {
66             if(miningAlgo.equals("id3"))
67             {
68                 treeDataResult = new String[tempTreeDataResult.length - 2];
69             }
70             else
71             {
72                 treeDataResult = new String[tempTreeDataResult.length - 6];
73             }
74         }
75     }
76 }
```

```

71         }
72         System.arraycopy(tempTreeDataResult, 2, treeDataResult, 0, treeDataResult.length);
73     }
74     else
75     {
76         if(miningAlgo.equals("id3"))
77         {
78             treeDataResult = new String[tempTreeDataResult.length - 3];
79         }
80         else
81         {
82             treeDataResult = new String[tempTreeDataResult.length - 7];
83         }
84         System.arraycopy(tempTreeDataResult, 3, treeDataResult, 0, treeDataResult.length);
85     }
86     System.out.println(treeDataResult[0]);
87     SDForConvertTree dataTree = new SDForConvertTree(treeDataResult);
88 //System.out.println("TEST: " + dataTree.getData(0).length());
89
90     try {
91         PrintWriter out = new PrintWriter(new BufferedWriter(new FileWriter("tree.txt")));
92         SDForExtractData extract = new SDForExtractData(new String[]{"bulan", "tahun", "hari", "jam"}, 
93             new int[]{maxMin[0], maxMin[1], 7, 24}, new int[]{maxMin[2], maxMin[3], 1, 0});
94         out.println("digraph{" + DotConverter.convert(dataTree, extract, miningAlgo, 0, "") + "}");
95         out.close();
96
97         textArea.setText(textArea.getText());
98         ArrayList<String> extractData = extract.getList();
99
100        if(extractData.size() > 0)
101        {
102            textArea.setText(textArea.getText() + "\nExtractData\n");
103        }
104        for(int i = 0; i < extractData.size(); i++)
105        {
106            textArea.setText(textArea.getText() + "\n" + extractData.get(i));
107        }
108    } catch (IOException ex) {
109        System.out.println("Error ketika menulis file txt");
110    }
111
112    Cmd.makeJpgUsingDotCommand();
113
114    JFrame jf2 = new JFrame();
115
116    jf2.setVisible(true);
117    jf2.setSize(620, 500);
118    BufferedImage image = null;
119    image = ImageIO.read(new File("tree.jpg"));
120    ImageIcon image2 = new ImageIcon(image);
121    JLabel labels = new JLabel(image2);
122    JScrollPane pane = new JScrollPane(labels);
123    jf2.setContentPane(pane);
124}
125
126
127 public static void main (String [] args)
128 {
129     Controller cont = new Controller();
130
131     JFrame jf = new JFrame();
132     View v = new View(cont);
133
134     jf.setVisible(true);
135     jf.setSize(620, 500);
136     jf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
137
138     jf.add(v);
139 }
140

```

Listing B.2: Vi w.java

```

1 package DataMiningLogHistoriKIRIWithoutDateAndMinutes;
2
3 import java.io.File;
4 import java.io.IOException;
5 import java.util.logging.Level;
6 import java.util.logging.Logger;
7 import javax.swing.JFileChooser;
8
9 /**
10  * 
11  * @author Jovan Gunawan
12  */
13 public class View extends javax.swing.JPanel {
14
15     /**
16      * Creates new form View
17      */
18     public View(Controller cont) {
19         this.cont = cont;
20         initComponents();
21         buttonGroup1.add(radioButtonId3);
22         buttonGroup1.add(radioButtonJ48);
23     }
24

```



```

118         .addComponent(buttonStart, javax.swing.GroupLayout.PREFERRED_SIZE,
119                         124, javax.swing.GroupLayout.PREFERRED_SIZE)
120         .addComponent(labelHasil, javax.swing.GroupLayout.PREFERRED_SIZE, 89,
121                         javax.swing.GroupLayout.PREFERRED_SIZE)
122         .addComponent(scrollPanel, javax.swing.GroupLayout.DEFAULT_SIZE, 441,
123                         Short.MAX_VALUE)
124         .addComponent(labelKeterangan, javax.swing.GroupLayout.DEFAULT_SIZE,
125                         javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
126     .addGap(0, 0, Short.MAX_VALUE)))
127 );
128 );
129 layout.setVerticalGroup(
130     layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
131     .addGroup(layout.createSequentialGroup()
132         .addContainerGap()
133         .addComponent(judul, javax.swing.GroupLayout.PREFERRED_SIZE, 31, javax.swing.GroupLayout.PREFERRED_SIZE)
134         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
135         .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
136             .addComponent(labelFileData, javax.swing.GroupLayout.PREFERRED_SIZE, 26, javax.swing.GroupLayout.PREFERRED_SIZE)
137             .addComponent(textFieldFilePath, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
138             .addComponent(buttonBrowse))
139         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
140         .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
141             .addComponent(labelPemilihanMethod, javax.swing.GroupLayout.PREFERRED_SIZE, 26, javax.swing.GroupLayout.PREFERRED_SIZE)
142             .addComponent(radioButtonJ48)
143             .addComponent(radioButtonId3))
144         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
145         .addComponent(buttonStart)
146         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
147         .addComponent(labelHasil, javax.swing.GroupLayout.PREFERRED_SIZE, 26, javax.swing.GroupLayout.PREFERRED_SIZE)
148         .addGap(2, 2, 2)
149         .addComponent(scrollPanel, javax.swing.GroupLayout.PREFERRED_SIZE, 251, javax.swing.GroupLayout.PREFERRED_SIZE)
150         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 10, Short.MAX_VALUE)
151         .addComponent(labelKeterangan, javax.swing.GroupLayout.PREFERRED_SIZE, 26, javax.swing.GroupLayout.PREFERRED_SIZE)
152     )
153 );
154 }
155 // </editor-fold>
156
157 private void buttonBrowseActionPerformed(java.awt.event.ActionEvent evt) {
158     final JFileChooser fc = new JFileChooser();
159     int returnVal = fc.showOpenDialog(buttonStart);
160
161     if (returnVal == JFileChooser.APPROVE_OPTION) {
162         File file = fc.getSelectedFile();
163         textFieldFilePath.setText(file.getPath());
164     } else {
165         System.out.println("Error in capturing the path");
166         System.exit(1);
167     }
168 }
169
170 private void buttonStartActionPerformed(java.awt.event.ActionEvent evt) {
171     String inputPath = textFieldFilePath.getText();
172     String miningAlgo = "";
173     if(radioButtonId3.isSelected())
174     {
175         miningAlgo = "id3";
176     }
177     else
178     {
179         miningAlgo = "j48";
180     }
181
182     try {
183         cont.startMining(inputPath, miningAlgo, labelKeterangan, hasil);
184     } catch (IOException e) {
185         System.out.println("Error start mining");
186         System.exit(1);
187     }
188 }
189
190 // Variables declaration - do not modify
191 private javax.swing.JButton buttonBrowse;
192 private javax.swing.ButtonGroup buttonGroup1;
193 private javax.swing.JButton buttonStart;
194 private javax.swing.JTextArea hasil;
195 private javax.swing.JLabel judul;
196 private javax.swing.JLabel labelFileData;
197 private javax.swing.JLabel labelHasil;
198 private javax.swing.JLabel labelKeterangan;
199 private javax.swing.JLabel labelPemilihanMethod;
200 private javax.swing.JRadioButton radioButtonId3;
201 private javax.swing.JRadioButton radioButtonJ48;
202 private javax.swing.JScrollPane scrollPanel;
203 private javax.swing.JTextField textFieldFilePath;
204 // End of variables declaration
205 private Controller cont;

```

206 | }

Listing B.3: CSVR ad r.java

```

1 package DataMiningLogHistoriKIRIWithoutDateAndMinutes;
2
3 import java.io.BufferedReader;
4 import java.io.FileReader;
5 import java.io.IOException;
6 import java.util.ArrayList;
7 import java.util.Set;
8
9 /**
10  * 
11  * @author Jovan Gunawan
12  */
13 public class CSVReader
14 {
15     private ArrayList<String[]> data;
16     private int banyakAtribut;
17     public CSVReader()
18     {
19         data = new ArrayList<String[]>();
20         banyakAtribut = 0;
21     }
22
23     public void setEmpty()
24     {
25         getData().clear();
26     }
27     public ArrayList readCSV( String [] file )
28     {
29         for(int i = 0; i < file.length; i++)
30         {
31             readCSV( file [ i ] );
32         }
33         return getData();
34     }
35     public ArrayList readCSV( String file )
36     {
37         try
38         {
39             String temp;
40             String [] splited;
41             int i=0;
42             BufferedReader br = new BufferedReader(new FileReader(file));
43             while ((temp = br.readLine()) != null)
44             {
45                 splited = temp.split("\\\"");
46                 if(i==0)
47                 {
48                     //baca atributnya terlebih dahulu
49                     ArrayList al = new ArrayList<String>();
50                     String tempAtribut="";
51                     for(int j = 0; j < splited.length; j++)
52                     {
53                         if(j%2 == 0)
54                         {
55                             String [] splitedKoma = splited[j].split(",");
56                             for(int k = 0; k < splitedKoma.length; k++)
57                             {
58                                 if(!(k == 0 && splitedKoma[k].length() ==0) ||(k==splitedKoma.length-1 &&
59                                     splitedKoma[k].length() == 0))
60                                 {
61                                     al.add(splitedKoma[k]);
62                                 }
63                             }
64                         else
65                         {
66                             al.add(splited [ j ] );
67                         }
68                     }
69                     banyakAtribut = al.size();
70                     String [] tempDataAtribut = new String[banyakAtribut];
71                     for(int j = 0; j < banyakAtribut; j++)
72                     {
73                         tempDataAtribut[j] = (String)al.get(j);
74                     }
75                     getData().add(tempDataAtribut);
76                     i++;
77                 }
78             }
79         }
80         else
81         {
82             //baca untuk datanya
83             int index = 0;
84             String [] tempData = new String[banyakAtribut];
85             for(int j = 0; j < splited.length; j++)
86             {
87                 if(j%2 == 0)
88                 {
89                     String [] splitedKoma = splited[j].split(",");
90                     for(int k = 0; k < splitedKoma.length; k++)
91                     {
92                         if(!(k == 0 && splitedKoma[k].length() ==0) ||(k==splitedKoma.length-1 &&
93                             splitedKoma[k].length() == 0))
94                         {
95                             tempData[index] = splitedKoma[k];
96                         }
97                     }
98                 }
99             }
100            getData().add(tempData);
101        }
102    }
103 }
```

```

93             index++;
94         }
95     }
96   }  

97 else
98 {
99     // harusnya ada cek ada kutip lagi? (" merupakan " biasa)
100    tempData[index] = splitted[j];
101    index++;
102 }
103 getdata().add(tempData);
104 i++;
105 }
106 }
107 for(int j = 0; j < i; j++)
108 {
109     String [] temp2 = (String []) getData().get(j);
110 }
111 br.close();
112 }catch(IOException e)
113 {
114     System.out.println("Failed to read data");
115 }
116 }
117 return getData();
118 }
119
120 public ArrayList getData()
121 {
122     return data;
123 }
124 public void setData(ArrayList data)
125 {
126     this.data = data;
127 }
128 public int getBanyakAtribut()
129 {
130     return banyakAtribut;
131 }
132 public void setBanyakAtribut(int banyakAtribut)
133 {
134     this.banyakAtribut = banyakAtribut;
135 }
136 }

```

Listing B.4: ProcessingData.java

```

1 package DataMiningLogHistoriKIRIWithoutDateAndMinutes;
2
3 import java.util.ArrayList;
4
5 /**
6 *  

7 * @author Jovan Gunawan  

8 */
9 public class ProcessingData
10 {
11     ProcessingData()
12     {
13     }
14
15     public void processSorting(ArrayList addApiKey, ArrayList findRoute, ArrayList login, ArrayList
16         nearbyTransport, ArrayList pageLoad, ArrayList register, ArrayList searchPlace, ArrayList
17         widgetError, ArrayList widgetLoad, ArrayList data)
18     {
19         System.out.println("Banyak Data: " + data.size());
20         for(int i = 0; i < data.size(); i++)
21         {
22             String [] tempData = (String []) data.get(i);
23             if(tempData[3].equals("ADDAPIKEY"))
24             {
25                 addApiKey.add(tempData);
26             }
27             else if(tempData[3].equals("FINDROUTE"))
28             {
29                 findRoute.add(tempData);
30             }
31             else if(tempData[3].equals("LOGIN"))
32             {
33                 login.add(tempData);
34             }
35             else if(tempData[3].equals("NEARBYTRANSPORT"))
36             {
37                 nearbyTransport.add(tempData);
38             }
39             else if(tempData[3].equals("PAGELOAD"))
40             {
41                 pageLoad.add(tempData);
42             }
43             else if(tempData[3].equals("REGISTER"))
44             {
45                 register.add(tempData);
46             }
47             else if(tempData[3].equals("SEARCHPLACE"))
48             {
49                 searchPlace.add(tempData);
50             }
51             else if(tempData[3].equals("WIDGETERROR"))
52             {
53                 widgetError.add(tempData);
54             }
55         }
56     }
57 }

```

```

50         {
51             widgetError.add(tempData);
52         }
53     else if(tempData[3].equals("WIDGETLOAD"))
54     {
55         widgetLoad.add(tempData);
56     }
57 }
58 }
59 public void processSorting(ArrayList array, ArrayList data, String action)
60 {
61     for(int i = 0; i < data.size(); i++)
62     {
63         String [] tempData = (String []) data.get(i);
64         if(tempData[3].equals(action))
65         {
66             array.add(tempData);
67         }
68     }
69 }
70 public ArrayList preprocessingData(ArrayList<String[]> data, int [] maxMin)
71 {
72     // 2 int[] untuk mendapatkan nilai max dan min dari variabel bulan dan tahun yang digunakan untuk
73     // inisialisasi max min pada kelas SDForExtractData.
74     // array pertama untuk bulan, dan array kedua untuk tahun
75     int [] max = new int [2];
76     int [] min = new int [2];
77     max[0] = 0;
78     max[1] = 0;
79     min[0] = Integer.MAX_VALUE;
80     min[1] = Integer.MAX_VALUE;
81     ArrayList<int[]> result = new ArrayList<int[]>();
82
83     // tahap pertama: ubah waktu dari UTC ke GMT+7
84     for(int i = 0; i < data.size(); i++)
85     {
86         //System.out.println(data.get(i)[3]);
87         data.get(i)[2] = TimezoneConverter.convertToGMT7(data.get(i)[2]);
88     }
89
90     // tahap kedua sampai kesembilan
91     DistanceHaversine haversine = new DistanceHaversine();
92     for(int i = 0; i < data.size(); i++)
93     {
94         //cek apakah format sudah benar atau belum
95         if(data.get(i)[4].split(",").length == 3)
96         {
97             // tahap kedua: pecah string atribut tanggal
98             int [] temp = new int [5];
99             String [] splitted = data.get(i)[2].split("ω");
100            temp[2] = Integer.parseInt(splitted[0]);
101            // tahap ketiga: pecah nilai string yang tanggal
102            String [] splitted2 = splitted[1].split("/");
103            temp[0] = Integer.parseInt(splitted2[0]);
104            temp[1] = Integer.parseInt(splitted2[2]);
105            // tahap keempat: pecah nilai string yang jam
106            splitted2 = splitted[2].split(":");
107            temp[3] = Integer.parseInt(splitted2[0]);
108            // tahap kelima: pecah string atribut additional data
109            splitted = data.get(i)[4].split("/");
110            // tahap keenam: pecah lokasi keberangkatan dan lokasi tujuan untuk mendapatkan lat n lon
111            // dan (ini tahap ketujuh) menghitung jarak terhadap titik pusat
112            splitted2 = splitted[0].split(",");
113            double jarakKeberangkatan = haversine.calculateDistance(Double.parseDouble(splitted2[0]),
114                Double.parseDouble(splitted2[1]), -6.916667,107.6) * 1000;
115            splitted2 = splitted[1].split(",");
116            double jarakTujuan = haversine.calculateDistance(Double.parseDouble(splitted2[0]),
117                Double.parseDouble(splitted2[1]), -6.916667,107.6) * 1000;
118            // tahap kedelapan, set semua data ke array
119            temp[4] = klasifikasiKelas(jarakKeberangkatan, jarakTujuan);
120
121            if(temp[4] != -2)
122            {
123                result.add(temp);
124                //proses untuk mencatat nilai max dan min
125                if(max[0] < temp[0])
126                {
127                    max[0] = temp[0];
128                }
129                if(min[0] > temp[0])
130                {
131                    min[0] = temp[0];
132                }
133                if(max[1] < temp[1])
134                {
135                    max[1] = temp[1];
136                }
137                if(min[1] > temp[1])
138                {
139                    min[1] = temp[1];
140                }
141            }
142            else
143            {
144                System.out.println("ERROR: ωadditional ωdata ωtidak ωsesuai; ω" + data.get(i)[4]);
145            }
146        }
147    }
148 }
```

```

145     maxMin[0] = max[0];
146     maxMin[1] = max[1];
147     maxMin[2] = min[0];
148     maxMin[3] = min[1];
149     return result;
150 }
151
152 public int klasifikasiKelas(double jarakKeberangkatan, double jarakTujuan)
153 {
154     int regionKeberangkatan, regionTujuan;
155     int klasifikasi = 0; // 0 --> tidak menuju Bandung, 1 --> menuju Bandung, 2 --> menuju region yang sama
156
157     regionKeberangkatan = (int) jarakKeberangkatan;
158     regionTujuan = (int) jarakTujuan;
159
160     if (regionKeberangkatan >= 11 || regionTujuan >= 11)
161     {
162         return -2;
163     }
164     else
165     {
166         if (regionKeberangkatan > regionTujuan)
167         {
168             klasifikasi = -1;
169         }
170         else if (regionKeberangkatan < regionTujuan)
171         {
172             klasifikasi = 1;
173         }
174         else
175         {
176             klasifikasi = 0;
177         }
178     }
179     return klasifikasi;
180 }
181 }
```

Listing B.5: Tim zon Conv rt r.java

```

1 package DataMiningLogHistoriKIRIWithoutDateAndMinutes;
2
3 import java.text.ParseException;
4 import java.text.SimpleDateFormat;
5 import java.util.Date;
6 import java.util.TimeZone;
7 import java.util.logging.Level;
8 import java.util.logging.Logger;
9
10 /**
11 *
12 * @author Jovan Gunawan
13 */
14 public class TimezoneConverter
15 {
16     // Mengubah input waktu menjadi waktu GMT+7
17     // Jika ingin mengubah dari UTC ke GMT+7 maka komputer harus set timezone ke UTC
18     // input parameter string harus dalam format dd-MM-yyyy HH:mm:ss --> contoh 1/1/2014 3:51:15 AM
19     // output akan mengubah waktu menjadi GMT+8 dalam String dengan format EEE MM/dd/yyyy HH:mm:ss -->
20     // contoh Wed 01/01/2014 03:51:15
21     public static String convertToGMT7(String date)
22     {
23         Date d = null;
24         long milliseconds = 0;
25
26         try {
27             SimpleDateFormat f = new SimpleDateFormat("MM/dd/yyyy HH:mm");
28             d = (Date)f.parse(date);
29         } catch (ParseException ex) {
30             SimpleDateFormat f = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
31             try {
32                 d = (Date)f.parse(date);
33             } catch (ParseException ex1) {
34                 Logger.getLogger(TimezoneConverter.class.getName()).log(Level.SEVERE, null, ex);
35                 System.exit(1);
36             }
37         }
38         milliseconds = d.getTime();
39
40         Date currentTime = new Date(milliseconds);
41         // untuk hari --> 1 = senin, ..., 7 = minggu
42         SimpleDateFormat sdf = new SimpleDateFormat("u MM/dd/yyyy HH:mm:ss");
43
44         sdf.setTimeZone(TimeZone.getTimeZone("GMT+7"));
45         return sdf.format(currentTime);
46     }
}
```

Listing B.6: Distanc Hav rsin .java

```

1 package DataMiningLogHistoriKIRIWithoutDateAndMinutes;
2
3 /**
4 * @author Jovan Gunawan
5 */
6
```

```

7 public class DistanceHaversine
8 {
9     private double r;
10    public DistanceHaversine()
11    {
12        r = 6.371;
13    }
14
15    public double calculateDistance(double latitude1, double longitude1, double latitude2, double
16                                     longitude2)
17    {
18        latitude1 = Math.toRadians(latitude1);
19        longitude1 = Math.toRadians(longitude1);
20        latitude2 = Math.toRadians(latitude2);
21        longitude2 = Math.toRadians(longitude2);
22
23        double dlon = longitude2 - longitude1;
24        double dlat = latitude2 - latitude1;
25
26        double a = Math.pow((Math.sin(dlat/2)),2) + Math.cos(latitude1) * Math.cos(latitude2) * Math.pow(
27                           Math.sin(dlon/2),2);
28
29        double c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
30        return r * c;
31    }
32 }
```

Listing B.7: ArffIO.java

```

1 package DataMiningLogHistoriKIRIWithoutDateAndMinutes;
2
3 import java.io.BufferedReader;
4 import java.io.BufferedWriter;
5 import java.io.FileReader;
6 import java.io.FileWriter;
7 import java.io.IOException;
8 import java.io.PrintWriter;
9 import java.util.ArrayList;
10 import java.util.logging.Level;
11 import java.util.logging.Logger;
12 import weka.core.Instances;
13
14 /**
15  *
16  * @author Jovan Gunawan
17  */
18 public class ArffIO
19 {
20     public ArffIO()
21     {
22     }
23
24     // method writer ini dibuat KHUSUS untuk skripsi data log KIRI
25     // input berupa arraylist dengan objek int[]
26     // Setiap array int, terdapat 7 nilai yaitu tanggal, bulan, tahun, hari, jam, menit, menujuBandung
27     // disini nilai hari akan diubah menjadi string, 1 akan menjadi senin, ..., dan 7 akan menjadi minggu
28     public void writeArrf(String name, ArrayList<int[]> data)
29     {
30         String result = "@relation " + name + "\n\n@attribute bulan REAL\n@attribute tahun REAL\n"
31         + "@attribute hari REAL"
32         + "\n@attribute jam REAL\n@attribute menujuBandung {-1,0,1}\n\n@data";
33
34         for(int i = 0; i < data.size(); i++)
35         {
36             int[] temp = data.get(i);
37             result += "\n" + temp[0] + "," + temp[1] + "," + temp[2] + "," + temp[3] + "," + temp[4];
38         }
39
40         try {
41             PrintWriter out = new PrintWriter(new BufferedWriter(new FileWriter("temp.arff")));
42             out.println(result);
43             out.close();
44         } catch (IOException ex) {
45             System.out.println("Error ketika menulis file arff");
46         }
47     }
48
49     public Instances readArrf(String name) throws IOException
50     {
51         Instances data;
52         data = new Instances(new BufferedReader(new FileReader("temp.arff")));
53         data.setClassIndex(data.numAttributes() - 1);
54         return data;
55     }
56 }
```

Listing B.8: DecisionTree.java

```

1 package DataMiningLogHistoriKIRIWithoutDateAndMinutes;
2
3 import java.util.logging.Level;
4 import java.util.logging.Logger;
5 import weka.classifiers.Classifier;
6 import weka.classifiers.trees.Id3;
7 import weka.classifiers.trees.J48;
8 import weka.core.Instances;
9 import weka.filters.Filter;
```

```

10 import weka.filters.unsupervised.attribute.NumericToNominal;
11 /**
12 *
13 * @author Jovan Gunawan
14 */
15 public class DecisionTree
16 {
17     private Classifier tree;
18
19     public double calculatePrecision(Instances arff)
20     {
21         int nilaiBenar = 0, resultInt;
22         float result = 0;
23         for (int i = 0; i < arff.numInstances(); i++)
24         {
25             try {
26                 result = (float)tree.classifyInstance(arff.instance(i));
27                 resultInt = Math.round(result)-1;
28                 if(resultInt == Integer.parseInt(arff.instance(i).stringValue(4)))
29                 {
30                     nilaiBenar++;
31                 }
32             } catch (Exception ex) {
33                 System.out.println("CHECK: " + ex.getMessage());
34             }
35         }
36         double confident = Math.round(nilaiBenar * 1.0 / arff.numInstances() * 10000) / 100.0;
37         return confident;
38     }
39
40     public String id3(Instances arff)
41     {
42         tree = new Id3();
43         try {
44             NumericToNominal convert= new NumericToNominal();
45             String[] options= new String[2];
46             options[0] = "-R";
47             options[1] = "1-4";
48
49             convert.setOptions(options);
50             convert.setInputFormat(arff);
51
52             Instances newData=Filter.useFilter(arff, convert);
53
54             tree.buildClassifier(newData);
55         } catch (Exception ex) {
56             Logger.getLogger(Controller.class.getName()).log(Level.SEVERE, null, ex);
57         }
58
59         return tree.toString();
60     }
61
62     public String j48(Instances arff)
63     {
64         tree = new J48();
65         try {
66             tree.buildClassifier(arff);
67         } catch (Exception ex) {
68             Logger.getLogger(Controller.class.getName()).log(Level.SEVERE, null, ex);
69         }
70
71         return tree.toString();
72     }
73 }
74 }
```

Listing B.9: DotConv rt r.java

```

1 package DataMiningLogHistoriKIRIWithoutDateAndMinutes;
2
3 import DataMiningLogHistoriKIRI.*;
4
5 /**
6 *
7 * @author Jovan Gunawan
8 */
9 public class DotConverter
10 {
11     // converter dot khusus untuk skripsi data mining log histori KIRI --> output berupa tree dalam string
12     // dari weka
13     public static String convert(SDForConvertTree data, SDForExtractData extract, String miningAlgo, int
14     deep, String nodeName)
15     {
16         String result = "";
17         String [] temp;
18         String nodeName1="", nodeName2="";
19         //color 1 selalu 1.0 karena merah
20         double color;
21         if(deep == 0 && data.getData().length == 1 && data.getData()[0].charAt(0) == ':')
22         {
23             result = data.getData()[0].split("·") [1];
24         }
25         else
26         {
27             if(miningAlgo.equals("id3"))
28             {

```

```

29|         while( hasNext)
30|     {
31|         hasNext = false;
32|
33|         temp = data . getData(0) . split(";;");
34|         boolean iniName1 = false;
35|
36|         System.out.println("Deep:" + deep + data . getData(0));
37|         temp = temp[deep] . split(";");
38|
39|         color = 0.7 - deep * 0.02;
40|         if(color < 0.3)
41|         {
42|             color = 0.3;
43|         }
44|
45|         if(nodeName . equals(""))
46|         {
47|             if(loop == 0)
48|             {
49|                 nodeName1 = data . getDataNumber(temp[0]);
50|             }
51|             result += nodeName1 + "->-";
52|             iniName1 = true;
53|
54|         } else
55|         {
56|             result += nodeName + "->-";
57|         }
58|
59|         SDForExtractData tempExtract = new SDForExtractData(extract);
60|
61|         if(temp[2] . charAt(temp[2].length() - 1) == ':')
62|         {
63|             // masukin data buat extract
64|             tempExtract . setRules(temp[0], temp[1], Integer . parseInt(temp[2] . substring(0, temp[2].length() - 1)));
65|             try
66|             {
67|                 tempExtract . setKelas(Integer . parseInt(temp[3]));
68|             } catch(Exception a)
69|             {
70|                 if(temp[3] == null)
71|                 {
72|                     tempExtract . setKelas(-2);
73|                 }
74|             }
75|             tempExtract . extract();
76|             extract . addStringResult(tempExtract . getList());
77|             // menghasilkan daun
78|
79|             nodeName2 = data . getDataNumber(temp[3]);
80|             result += nodeName2 + "[label=\"" + temp[1] + "-" + temp[2] . substring(0, temp[2].length() - 1) + "\"]\n";
81|             if(iniName1 && loop == 0)
82|             {
83|                 result += nodeName1 + "[label=\"" + temp[0] + "\",shape=box,style=filled , color=\\"1.0\\"
84|                 color += "\\1.0\\"]\n";
85|             }
86|             result += nodeName2 + "[label=\"" + temp[3] + "\"]\n";
87|             data . buangArrayPertama();
88|         } else
89|         {
90|             // masukin data bwt extract
91|             tempExtract . setRules(temp[0], temp[1], Integer . parseInt(temp[2]));
92|
93|             // menghasilkan node
94|             String [] temp2;
95|             temp2 = data . getData(1) . split(";;");
96|             temp2 = temp2[deep + 1] . split(";");
97|             nodeName2 = data . getDataNumber(temp2[0]);
98|             result += nodeName2 + "[label=\"" + temp[1] + "-" + temp[2] + "\",shape=box , style=filled , color=\\"1.0\\"
99|             color += "\\1.0\\"]\n";
100|
101|             SDForExtractData newExtract = new SDForExtractData(tempExtract);
102|             result += DotConverter . convert(data, newExtract, miningAlgo, deep + 1, nodeName2);
103|
104|             if(iniName1 && loop == 0)
105|             {
106|                 result += nodeName1 + "[label=\"" + temp[0] + "\",shape=box,style=filled , color=\\"1.0\\"
107|                 color += "\\1.0\\"]\n";
108|             }
109|             result += nodeName2 + "[label=\"" + temp2[0] + "\",shape=box,style=filled , color=\\"1.0\\"
110|             color += "\\1.0\\"]\n";
111|             extract . addStringResult(newExtract . getList());
112|
113|             if(data . hasNext())
114|             {
115|                 if(data . getData(0) . split(";;") . length - 1 == deep)
116|                 {
117|                     hasNext = true;
118|                 }
119|             }
120|             loop++;
121|         } else

```

```

122
123     {
124         for(int i = 0; i < 2; i++)
125         {
126             temp = data.getData(0).split("~~~");
127             boolean iniName1 = false;
128
129             System.out.println("Deep:" + deep + data.getData(0));
130             temp = temp[deep].split("~~");
131
132             color = 0.7 - deep * 0.02;
133             if(color < 0.3)
134             {
135                 color = 0.3;
136             }
137
138             if(nodeName.equals(""))
139             {
140                 if(i == 0)
141                 {
142                     nodeName1 = data.getDataNumber(temp[0]);
143                 }
144                 result += nodeName1 + "~->~";
145                 iniName1 = true;
146             }
147             else
148             {
149                 result += nodeName + "~->~";
150             }
151
152             SDForExtractData tempExtract = new SDForExtractData(extract);
153
154             if(temp[2].charAt(temp[2].length()-1) == ':')
155             {
156                 // masukin data bwt extract
157                 tempExtract.setRules(temp[0], temp[1], Integer.parseInt(temp[2].substring(0, temp[2].length()-1)));
158                 tempExtract.setKelas(Integer.parseInt(temp[3]));
159                 tempExtract.extract();
160                 extract.addStringResult(tempExtract.getList());
161                 // menghasilkan daun
162                 nodeName2 = data.getDataNumber(temp[3]);
163                 result += nodeName2 + "[label=" + temp[1] + "~~" + temp[2].substring(0, temp[2].length()-1) + "\n";
164                 if(iniName1 && i == 0)
165                 {
166                     result += nodeName1 + "[label=" + temp[0] + "\n", shape=box, style=filled,
167                     color="1.0~~" + color + "1.0\n"];
168                 }
169                 result += nodeName2 + "[label=" + temp[3] + "\n";
170                 data.buangArrayPertama();
171             }
172             else
173             {
174                 // masukin data bwt extract
175                 tempExtract.setRules(temp[0], temp[1], Integer.parseInt(temp[2]));
176                 // menghasilkan node
177                 String [] temp2;
178                 temp2 = data.getData(1).split("~~~");
179                 temp2 = temp2[deep+1].split("~~");
180                 nodeName2 = data.getDataNumber(temp2[0]);
181                 result += nodeName2 + "[label=" + temp[1] + "~~" + temp[2] + "\n", shape=box,
182                 style=filled, color="1.0~~" + color + "1.0\n";
183                 data.buangArrayPertama();
184
185                 SDForExtractData newExtract = new SDForExtractData(tempExtract);
186                 result += DotConverter.convert(data, newExtract, miningAlgo, deep+1, nodeName2);
187
188                 if(iniName1 && i == 0)
189                 {
190                     result += nodeName1 + "[label=" + temp[0] + "\n", shape=box, style=filled,
191                     color="1.0~~" + color + "1.0\n"];
192                 }
193             }
194         }
195     }
196 }
197 }
```

Listing B.10: SDForConv rtTr .java

```

1 package DataMiningLogHistoriKIRIWithoutDateAndMinutes;
2
3 /**
4 *
5 * @author Jovan Gunawan
6 * Class ini dibuat untuk struktur data yang digunakan untuk mengubah hasil output dari weka menjadi
7 * gambar dengan bahasa DOT
8 * Class ini khusus untuk skripsi data mining log histori KIRI
9 *
10 * count akan digunakan untuk inisialisasi nomor data --> contoh tanggal1, tanggal2, .... etc, angka
11 * tersebut yang akan ditaruh pada array tersebut
12 * array count akan digunakan sebagai berikut --> [0] = tanggal, [1] = bulan, [2] = tahun, [3] = hari, [4]
13 * = jam, [5] = menit, [6] = nilai 0, [7] = nilai 1, [8] = nilai 2
```

```
11 */  
12 public class SDForConvertTree  
13 {  
14     private String [] data;  
15     private int [] count;  
16  
17     public SDForConvertTree(String [] data)  
18     {  
19         this.data = data;  
20         count = new int [9];  
21         for(int i = 0; i < 9; i++)  
22         {  
23             count[i] = 0;  
24         }  
25     }  
26  
27     public void setData(String data, int index)  
28     {  
29         this.data[index] = data;  
30     }  
31     public String [] getData()  
32     {  
33         return data;  
34     }  
35     public String getData(int index)  
36     {  
37         return this.data[index];  
38     }  
39     public void setCount(int count, int index)  
40     {  
41         this.count[index] = count;  
42     }  
43     public int getCount(int index)  
44     {  
45         int temp = this.count[index];  
46         this.count[index]++;  
47     }  
48     public boolean hasNext()  
49     {  
50         if(this.data.length > 0)  
51         {  
52             return true;  
53         }  
54     }
```

```

110     else if(atribut.equals("0"))
111     {
112         result += count[8];
113         count[8]++;
114     }
115     return result;
116 }
117 }
```

Listing B.11: SDForExtractData.java

```

1 package DataMiningLogHistoriKIRIWithoutDateAndMinutes;
2
3 import java.util.ArrayList;
4
5 /**
6 *
7 * @author Jovan Gunawan
8 */
9 public class SDForExtractData
10 {
11     private String[] atribut;
12     private boolean[] check;
13     private int[] batasAtas;
14     private int[] batasBawah;
15     private int[] maxBatasAtas;
16     private int[] minBatasBawah;
17     private int kelas;
18     private ArrayList<String> list;
19
20     public SDForExtractData(String[] atribut, int[] max, int[] min)
21     {
22         int length = atribut.length;
23         this.atribut = new String[length];
24         this.maxBatasAtas = new int[length];
25         this.minBatasBawah = new int[length];
26         this.batasAtas = new int[length];
27         this.batasBawah = new int[length];
28         this.check = new boolean[length];
29         list = new ArrayList<String>();
30
31         for(int i = 0; i < length; i++)
32         {
33             this.atribut[i] = atribut[i];
34             this.maxBatasAtas[i] = max[i];
35             this.batasAtas[i] = max[i];
36             this.minBatasBawah[i] = min[i];
37             this.batasBawah[i] = min[i];
38             check[i] = false;
39         }
40     }
41
42     public SDForExtractData(SDForExtractData data)
43     {
44         String[] atribut = data.getAtribut();
45         int[] max = data.getMaxBatasAtas();
46         int[] min = data.getMinBatasBawah();
47         int[] bmax = data.getBatasAtas();
48         int[] bmin = data.getBatasBawah();
49         boolean[] check = data.getCheck();
50         int length = data.getAtribut().length;
51         list = data.getList();
52
53         this.atribut = new String[length];
54         this.maxBatasAtas = new int[length];
55         this.minBatasBawah = new int[length];
56         this.batasAtas = new int[length];
57         this.batasBawah = new int[length];
58         this.check = new boolean[length];
59
60         for(int i = 0; i < length; i++)
61         {
62             this.atribut[i] = atribut[i];
63             this.maxBatasAtas[i] = max[i];
64             this.batasAtas[i] = bmax[i];
65             this.minBatasBawah[i] = min[i];
66             this.batasBawah[i] = bmin[i];
67             this.check[i] = check[i];
68         }
69     }
70
71     public void setKelas(int kelas)
72     {
73         this.kelas = kelas;
74     }
75
76     public void setRules(String atribut, String rulesOperation, int value)
77     {
78         int index = 0;
79         for(int i = 0; i < this.atribut.length; i++)
80         {
81             if(this.atribut[i].equals(atribut))
82             {
83                 index = i;
84                 break;
85             }
86         }
87         check[index] = true;
88     }
89 }
```

```

88
89     if(rulesOperation.equals("<="))
90     {
91         batasAtas[index] = value;
92     }
93     else if(rulesOperation.equals("<"))
94     {
95         batasAtas[index] = value - 1;
96     }
97     else if(rulesOperation.equals(">="))
98     {
99         batasBawah[index] = value;
100    }
101   else if(rulesOperation.equals(">"))
102   {
103     batasBawah[index] = value + 1;
104   }
105   else if(rulesOperation.equals("!="))
106   {
107     batasAtas[index] = value;
108     batasBawah[index] = value;
109   }
110 }
111
112 public void extract()
113 {
114     String[] hari = new String[]{"Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu", "Minggu"};
115     boolean in = false;
116     String result = "";
117     for(int i = 0; i < atribut.length; i++)
118     {
119         if(check[i])
120         {
121             if(in)
122             {
123                 result += "&&";
124             }
125             if(batasBawah[i] == batasAtas[i])
126             {
127                 result += atribut[i] + "= " + batasAtas[i];
128             }
129             else
130             {
131                 result += atribut[i] + ":" + batasBawah[i] + "- " + batasAtas[i];
132             }
133             in = true;
134         }
135     }
136
137     if(kelas == 1)
138     {
139         result += ">_Menju_ke_Bandung";
140     }
141     else if(kelas == 0)
142     {
143         result += ">_Menju_daerah_yang_sama";
144     }
145     else
146     {
147         result += ">_Menjauh_dari_Bandung";
148     }
149     list.add(result);
150 }
151
152 public void addStringResult(ArrayList<String> result)
153 {
154     list = result;
155 }
156
157 public String[] getAtribut()
158 {
159     return atribut;
160 }
161
162 public boolean[] getCheck()
163 {
164     return check;
165 }
166
167 public int[] getBatasAtas()
168 {
169     return batasAtas;
170 }
171
172 public int[] getBatasBawah()
173 {
174     return batasBawah;
175 }
176
177 public int[] getMaxBatasAtas()
178 {
179     return maxBatasAtas;
180 }
181
182 public int[] getMinBatasBawah()
183 {
184     return minBatasBawah;
185 }
186
187 public int getKelas()
188 {
189     return kelas;
190 }
191
192 public ArrayList<String> getList()
193 {
194     return list;
195 }
```

```
187 }
188 }
189 }
```

Listing B.12: CMD.java

```
1 package DataMiningLogHistoriKIRIWithoutDateAndMinutes;
2
3 import java.io.BufferedReader;
4 import java.io.IOException;
5 import java.io.InputStreamReader;
6
7 /**
8 *
9 * @author Jovan Gunawan
10 */
11 public class Cmd
12 {
13     public static void makeJpgUsingDotCommand()
14     {
15         try {
16             final String dir = System.getProperty("user.dir");
17
18             ProcessBuilder builder = new ProcessBuilder(
19                 "cmd.exe", "/c", "cd graphviz && cd bin && dot \" + dir + "\\tree.txt\" -o \" + dir + "\\"
20                 "tree.jpg\" -Tjpg");
21             builder.redirectErrorStream(true);
22             Process p = builder.start();
23             BufferedReader r = new BufferedReader(new InputStreamReader(p.getInputStream()));
24             String line;
25             while (true) {
26                 line = r.readLine();
27                 if (line == null)
28                 {
29                     break;
30                 }
31                 System.out.println(line);
32             }
33         } catch (IOException e) {
34             System.out.println("Error ketika proses CMD");
35             System.exit(1);
36         }
37     }
38 }
```