

SKRIPSI

DATA MINING HISTORI PENCARIAN RUTE ANGKOT



JOVAN GUNAWAN

NPM: 2011730029

**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN
2015**

UNDERGRADUATE THESIS

***DATA MINING ON PUBLIC TRANSPORT ROUTE
SEARCHING HISTORY***



JOVAN GUNAWAN

NPM: 2011730029

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

2015

ABSTRAK

Informasi yang akurat dibutuhkan dalam kehidupan sehari-hari untuk melakukan analisis dan pengambilan keputusan. Informasi tersebut belum dilolah agar dapat disajikan dengan baik. Jika dilihat lebih rinci, suatu data dapat diolah lebih lanjut untuk mempermudah pengambilan keputusan. *Data mining* merupakan salah satu proses pengolahan data untuk mempermudah analisis dan pengambilan keputusan 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 aksi pencarian dari satu titik ke titik yang lain. Atribut yang diuji adalah *timestamp* dan *additionalData* yang berisi lokasi keberangkatan dan tujuan dari pengguna yang menggunakan aplikasi KIRI. Pada percobaan ini, dibuat klasifikasi se puluh dua rute di Bandung berdasarkan titik pusat Bandung dengan radius satu kilometer untuk setiap daerah. Dengan klasifikasi tersebut, dapat ditentukan apakah pengguna menuju Bandung atau mendekati Bandung atau menuju daerah yang sama. Penggunaan *decision tree* digunakan untuk melakukan klasifikasi apakah pada hari tertentu dan jam tertentu, pengguna akan lebih sering menuju Bandung atau menuju Bandung atau menuju daerah yang sama. Dari hasil pengujian kspesimal, dipercaya bahwa *decision tree* yang dibuat dengan ID3 mengalami *overfitting* dengan akurasi 38.22%, sedangkan dengan C4.5 tidak mengalami *overfitting* dengan akurasi 50.37%. Dari hasil tersebut, dipercaya simpulan bahwa pengguna sering menuju Bandung daripada menuju Bandung atau menuju daerah yang sama pada bulan Februari 2014.

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

ABSTRACT

Accurate information is needed every day to perform analysis and decision making. The information needs to be stored in order to be presented later. If viewed in more detail, the data can be stored further to facilitate decision making. *Data mining* is one of data processing 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 find from one place to other place. Attributes used are timestamp and additionalData which contains date of birth location and destination location from user who uses KIRI application. In this experiment, will be made a regional 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. The *decision tree* is used to classify whether it rains for 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 C4.5 is not overfitting with 50.37% accuracy. From the result, 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*

DAFTAR ISI

DAFTAR ISI	ix
DAFTAR GAMBAR	xi
DAFTAR TABEL	xii
1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Perumusan Masalah	1
1.3 Tujuan	2
1.4 Batasan Masalah	2
1.5 Metod Penelitian	2
1.6 Sistem Matematika Pembahasan	2
2 LANDASAN TEORI	5
2.1 Knowledge Discovery	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	19
2.2 Log Histori KIRI	19
2.3 Haversine Formula	20
2.4 Waka	21
2.5 Graphviz	22
3 ANALISA	25
3.1 Analisis Data	25
3.1.1 Data Cleaning	25
3.1.2 Data Integration	25
3.1.3 Data Selection	25
3.1.4 Data Transformation	26
3.2 Data Convert Hasil Decision Tree menjadi Bahasa DOT	30
3.2.1 Pengelompokan dalam node	31
3.2.2 Ketentuan untuk membuat edge	31
3.3 Analisis Pengangkutan Lunak	31
3.3.1 Diagram Use Case Pengangkutan Lunak Data Mining Log Histori KIRI	33
3.3.2 Diagram Kelas Pengangkutan Lunak Data Mining Log Histori KIRI	35
4 PERANCANGAN PERANGKAT LUNAK	37
4.1 Perancangan Pengangkutan Lunak	37

4.1.1	P rancangan K las	37
4.1.2	S qu nc Diagram	42
4.1.3	P rancangan D sain Antar Muka	43
5	IMPLEMENTASI PROGRAM DAN PENGUJIAN	47
5.1	Lingkungan P mbangunan	47
5.2	Hasil Tampilan Antarmuka	47
5.3	P ngujian Aplikasi <i>Data Mining</i>	50
5.3.1	P ngujian Fungsional	50
5.3.2	P ngujian Eksp rim ntal	54
5.3.3	Analisis Hasil Uji	58
6	KESIMPULAN DAN SARAN	59
6.1	K simpulan	59
6.2	Saran	59
DAFTAR REFERENSI		61
A	100 DATA PERTAMA DARI <i>log HISTORI KIRI</i>	63
B	THE SOURCE CODE	69

DAFTAR GAMBAR

2.1	Tahap <i>Data Mining</i>	5
2.2	Tahap <i>data classification</i>	11
2.3	Contoh <i>decision tree</i>	12
2.4	J nis-j nis <i>split point</i>	14
2.5	Hasil pohon faktor pada atribut <i>age</i> dari tabl 2.1	16
2.6	<i>Decision Tree Pruned</i>	18
2.7	Hasil output Graphviz	24
3.1	<i>Classification</i> pada da rah Bandung	29
3.2	Diagram Use Case P rangkat Lunak <i>Data Mining Log Histori KIRI</i>	34
3.3	Diagram <i>Class</i> P rangkat Lunak <i>Data Mining Log Histori KIRI</i>	35
4.3	Mock Up Form P rtama	43
4.4	Mock Up Form K dua	43
4.1	Diagram <i>Class</i> P rangkat Lunak <i>Data Mining Log Histori KIRI</i>	44
4.2	Sequence Diagram P rangkat Lunak <i>Data Mining Log Histori KIRI</i>	45
5.1	Tampilan Form Awal Aplikasi <i>Data Mining</i>	48
5.2	Tampilan Fil S 1 ctor untuk M milih Fil CSV	48
5.3	Tampilan Form s t lah M milih Fil CSV	49
5.4	Tampilan Form P milihan M tod P mbuatan <i>Decision Tree</i>	49
5.5	Tampilan Form M nampilkan Hasil <i>Data Mining</i>	50
5.6	P ngujian M ngambil Data CSV	52
5.7	P ngujian <i>Data Selection</i> untuk M ngambil Data d ngan Action FINDROUTE	52
5.8	P ngujian <i>Preprocessing Data</i>	53
5.9	P ngujian <i>Preprocessing Data</i> untuk Klasifikasi Data	54
5.12	P ngujian Hasil Visualisasi d ngan M nggunakan Bahasa DOT	54
5.10	P ngujian P mbuatan <i>Decision Tree ID3</i>	55
5.11	P ngujian P mbuatan <i>Decision Tree C4.5</i>	55
5.13	P rcobaan <i>Data Mining</i> untuk M lakukan Data Cl aning	56
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	56
5.15	P rcobaan <i>Data Mining</i> d ngan M nggunakan M tod C4.5 pada <i>Log Histori KIRI</i> pada Bulan 2 Tahun 2014	57
5.16	Hasil dari SDForExtractData	58

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>	26
3.2	Contoh hasil data transformasi	28
3.3	Contoh hasil data transformasi latitud longitud	30
3.5	Sk nario M lakukan <i>load Data</i>	34
3.6	Sk nario M lakukan <i>Data Mining</i>	34
3.7	Sk nario M milih Algoritma yang Digunakan	35
5.1	Data untuk <i>Test Case</i> Aplikasi <i>Data Mining</i>	51
5.2	Hasil P n tuan ar a dan Klasifikasi	54

1

BAB 1

2

PENDAHULUAN

3

1.1 Latar Belakang

4 K butuhan akan informasi yang akurat dibutuhkan dalam k hidupan s hari-hari untuk m -
5 lakukan analisis dan p ngambilan k putusan. Informasi t rs but p rlu diolah agar dapat
6 disajikan d ngan baik. Jika dilihat l bih rinci, data t rs but dapat diolah l bih lanjut un-
7 tuk m mp rmudah p ngambilan k putusan. Salah satu cara m ngolah data adalah d ngan
8 m nggunakan pros s *data mining*. D ngan m nggunakan t knik *data mining*, analisa masa-
9 lah, p ngambilan k simpulan akan m njadi l bih mudah.

10 Tujuan utama dari *data mining* adalah *knowledge* [1]. *Knowledge* m rupakan suatu
11 informasi yang b rhaga dan dapat dijadikan landasan untuk m nganalisa atau m mbuat
12 k simpulan. Untuk m ndapatkan *knowledge*, dapat dilakukan d ngan cara m lakukan p n-
13 carian *pattern* yang m rupakan salah satu tahap dari *data mining*. *pattern* inilah yang akan
14 m mp rlihatkan data manakah yang m narik dan dapat dijadikan *knowledge* yang akan
15 digunakan untuk m nganalisa data t rs but.

16 Pada p n litian *data mining* ini, p nulis m miliki data *log* histori KIRI s lama 1 bulan.
17 Data t rs but akan dipros s d ngan m nggunakan t knik *data mining* untuk m ndapatkan
18 *pattern* dan *knowledge*. Data *log* t rs but m miliki 5 *field* untuk s tiap *entry* s bagai
19 b rikut:

- 20 • statisticId, primary k y dari ntry
21 • v rifi r, m ngid ntifikasi sumb r dari p ncarian ini
22 • timestamp, waktu k tika p ngguna KIRI m ncari rut angkot
23 • type, tip fungsi yang digunakan
24 • additionalInfo, m ncatat koordinat awal, koordinat akhir, dan banyak rut yang dit -
25 mukan pada p ncarian ini

26 B rdasarkan hal diatas, p nulis ingin m ndapatkan pola yang m narik dan m nghasilkan
27 *knowledge* yang b rguna dan dapat dipakai baik untuk KIRI ataupun p m rintah.

28

1.2 Perumusan Masalah

29 D ngan m ngacu pada uraian pada subbab s b lumnya, maka p rmasalahan yang dibahas
30 dan dit liti ol h p nulis adalah:

- 1 ● Bagaimana cara mengolah data yang dipilih dari data log histori KIRI agar pola
2 yang dihasilkan menjadi menarik dan bermakna?
- 3 ● Bagaimana membuat rangka lunak untuk melakukan *data mining* pada data log
4 histori?
- 5 ● Pola apa yang didapat dari data log histori KIRI?

6 **1.3 Tujuan**

7 Penelitian ini bertujuan untuk:

- 8 ● Mengcari pola dan informasi yang menarik dari log histori KIRI
- 9 ● Melakukan *data mining* dari log histori KIRI
- 10 ● Mengcari nilai dan menarik kesimpulan dari pola yang dipilih.

11 **1.4 Batasan Masalah**

12 Batasan masalah untuk penelitian *data mining* ini berupa:

- 13 ● Penelitian ini dibatasi untuk pengetahuan *data mining* pada data log KIRI
- 14 ● Data log yang digunakan untuk *data mining* merupakan log satu bulan dari KIRI

15 **1.5 Metode Penelitian**

16 Berikut adalah metodologi penelitian yang digunakan :

- 17 ● Melakukan studi literatur tentang algoritma-algoritma yang berkaitan dengan programmesan *data mining*
- 18 ● Melakukan penelitian *data mining* yang ditopang pada log KIRI
- 19 ● Merancang dan mengimplementasikan algoritma untuk *data mining*
- 20 ● Mengimplementasikan pembangkit pola *data mining*
- 21 ● Melakukan pengujian dan ksprimen
- 22 ● Melakukan pengujian dan ksprimen

23 **1.6 Sistematika Pembahasan**

24 Sistematisasi pembahasan dalam penelitian ini adalah:

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

- 1 ● BAB 2: Landasan Teori, berasis dasar teori mengenai *data mining*, log histori KIRI, Havrsin Formula, Wka, dan Graphviz.
- 2
- 3 ● BAB 3: Berisi analisa dasar teori yang akan digunakan, analisa data dan tahap *preprocessing* data yang akan digunakan, serta analisa perancangan aplikasi *data mining*
- 4 log histori KIRI disertai dengan diagram use case, skenario, dan diagram klas.
- 5
- 6 ● BAB 4: Berisi perancangan dari aplikasi *data mining* log histori KIRI yang akan
- 7 dibangun.
- 8 ● BAB 5: Berisi implementasi dan pengujian dari aplikasi *data mining*.
- 9 ● BAB 6: Berisi kesimpulan dan saran dari penelitian *data mining* log histori KIRI.

1

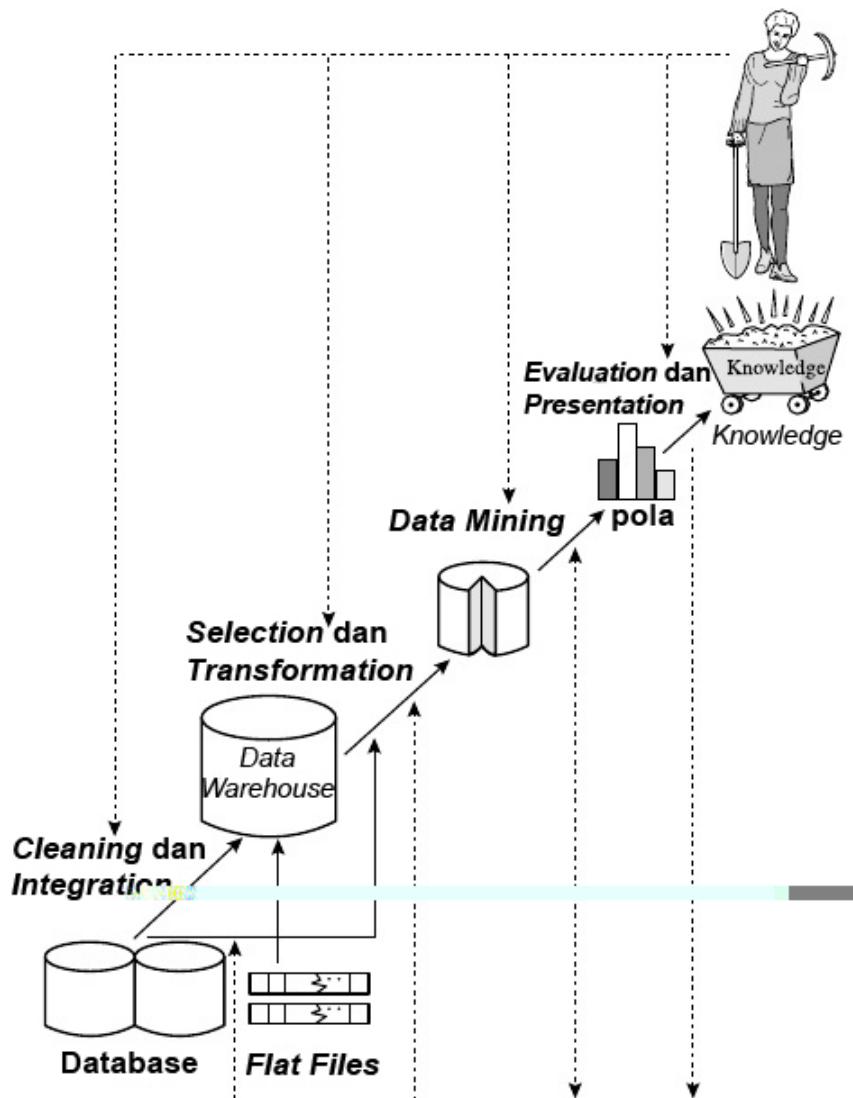
BAB 2

2

LANDASAN TEORI

3 2.1 *Knowledge Discovery*

- 4 Knowledge Discovery atau yang sering disebut juga *Data mining*, merupakan merupakan
5 proses pengambilan inti sari atau penggalian knowledge dari data yang besar.



Gambar 2.1: Tahap *Data Mining* [1]

- 6 Murniut [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*

*8 Tahap pertama hingga kempat merupakan bagian dari *data preprocessing*, dimana data-
9 disiapkan untuk dilakukan penggalian data. Tahap *data mining* merupakan tahap
10 dimana penggalian data dilakukan. Tahap ketujuh merupakan tahap pencarian pola yang
11 mampu menyatakan *knowledge*. Sedangkan tahap terakhir merupakan visualisasi dan representasi
12 sifat-sifat dari *knowledge* yang sudah dipilih dari tahap sebelumnya.*

13 2.1.1 Data Cleaning

*14 Data cleaning merupakan tahap *Knowledge Discovery* untuk menghilangkan *missing value*
15 dan *noisy data*. *Missing value* merupakan nilai yang hilang dari suatu data. *Noisy data*
16 merupakan nilai yang tidak sesuai atau tidak valid.*

*17 Pada umumnya, data yang dipilih dari database mendapat nilai yang tidak sempurna
18 seperti nilai yang hilang, nilai yang tidak valid atau salah ketik. Atribut dari suatu *database*
19 yang tidak relevan atau redundansi bisa diatasinya dengan menghapus atribut atau tupl
20 tersebut. Contoh data yang memiliki *missing value* dan *noisy data* dapat dilihat pada tabel
21 2.1*

Tab 1.2.1: tabel mengandung *missing value* dan *noisy data*

IdPenjualan	NamaBarang	Customer	Harga	BanyakBarang
1	Mouse	Elvin	45000	2
2	Keyboard	Allaria	-35000	1
3	Monitor		225000	1

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

25 Missing Values

*26 Missing values akan mengganggu proses *data mining* pada komputer dan dapat menghasilkan
27 nilai akhir yang tidak sesuai dengan fakta. Terdapat beberapa teknik untuk mengatasi
28 missing values yaitu:*

- 29 • Menghapus tupel yang mengandung nilai yang hilang*
- 30 • Mengisi nilai yang hilang dengan cara manual*

- 1 • Mengisi nilai yang hilang dengan menggunakan nilai konstan yang bersifat umum
2 • Menggunakan nilai rata-rata dari suatu atribut untuk mengisi nilai yang hilang

3 **Noisy Data**

4 *Noisy data* merupakan nilai yang bersifat ral dari error atau tidak valid. *Noisy data* dapat
5 dihilangkan dengan menggunakan teknik *smoothing*. Teknik *smoothing* merupakan teknik
6 untuk menghilangkan *noisy data*. Terdapat 3 metode untuk menghilangkan *noisy data* yaitu:

7

- 8 • *Binning*, merupakan metode pengisian data dengan proses yang dilakukan pada data
9 tersebut
10 • *Regression*, merupakan metode yang mencari model persamaan atribut untuk memprediksi
11 diketahui suatu nilai
12 • *Clustering*, merupakan metode pengelompokan dimana dituliskan puncilan yang dapat
13 dibuang. Puncilan merupakan data yang berada diluar kumpulan yang sesuai dengan
14 observasi atau pertemuan.

15 **2.1.2 Data Integration**

16 *Data integration* merupakan tahap penggabungan data dari berbagai sumber. Sumber tersebut
17 tidak bisa termasuk berbagai *database*, *data cubes*, atau *flat data*. *Data cube* merupakan
18 teknik pengambilan data-data dari *data warehouse* dan dilakukan operasi agregasi sesuai dengan
19 kondisi tertentu (contoh, penjumlahan total dari penjualan per tahun dari 2005-2010).
20 Sedangkan *flat data* merupakan data yang disimpan dengan cara apapun untuk merepresentasikan
21 model database pada sebuah data baik berformat *plain text file* maupun *binary file*.

22 Tahap ini harus dilakukan secara teliti terutama dalam menangkap nilai-nilai yang
23 berasal dari sumber yang berbeda. Pada tahap ini, perlu dilakukan identifikasi apakah data
24 tertentu harus dimasukkan atau tidak agar data yang dipilih tidak terlalu bersifat. *Data integration* yang baik merupakan integrasi yang dapat maksimalkan kepuasan dan meningkatkan akurasi dari proses *data mining*. Contoh studi kasus dari *data integration*, misalnya 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 harus digabung. Ketika digabung, harus memperhatikan dan memperbaiki nilai-nilai sperti *primary key*, atribut. Hal ini dilakukan untuk menghindari kesalahan seperti nilai yang berbeda padahal merupakan objek yang sama. Proses penggabungan hingga perbaikan nilai-nilai pada kedua databases tersebut disebut proses *data integration*.

34 **2.1.3 Data Selection**

35 Proses dimana data-data yang relevan dengan analisis akan diambil dari database dan data
36 yang tidak relevan akan dibuang. Sebagai contoh kasus, jika akan dilakukan analisa mengenai
37 nilai mahasiswa pada tabel nilai yang memiliki atribut sebagai berikut:

- 38 • NPM Mahasiswa

- 1 • NamaMahasiswa
- 2 • J nisK lamin
- 3 • Alamat
- 4 • MataKuliah
- 5 • NilaiART
- 6 • NilaiUTS
- 7 • NilaiUAS

8 Maka, atribut yang berpotensi diambil adalah MataKuliah, NilaiART, NilaiUTS dan NilaiUAS, sedangkan atribut yang dibuang adalah NPMMahasiswa, NamaMahasiswa J nisK - lamin, dan Alamat karena tidak berhubungan dengan analisa.

11 **2.1.4 Data Transformation**

12 *Data transformation* merupakan tahap perubahan data agar siap dilakukan proses *data mining*. *Data transformation* bisa melibatkan:

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

24 **Smoothing**

25 *Smoothing* merupakan teknik 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*.

28 **Aggregation**

29 *Aggregation* merupakan teknik melakukan operasi agregasi untuk mendapatkan nilai yang digunakan di tahap *data mining*. Contoh kasus, jika terdapat suatu database dari toko A, dapat menggunakan operasi agregasi untuk mencari total pendapatan dalam rangka hari tertentu.

1 Generalization

2 generalization merupakan teknik untuk mengubah data yang bersifat primitive atau *low level* menjadi
 3 jadi *high level* dengan menggunakan konsep hierarki. Contoh kasus, nilai pada atribut umur
 4 dapat dikategorikan menjadi muda, dewasa, tua.

5 Normalization

6 Normalisasi merupakan teknik untuk mengubah nilai atribut menjadi nilai baru yang memiliki
 7 rentang yang lebih spesifik dan kental seperti 0,0 sampai 1,0. Terdapat beberapa teknik normalisasi,
 8 dua diantaranya yaitu, *min-max normalization* dan *z-score normalization*. *Min-max normalization*
 9 akan mengubah semua nilai menjadi nilai dalam skala tertentu. Rumus dari teknik *min-max normalization* sebagai berikut

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

11 Contoh kasus, misalkan nilai minimum dan maksimum dari suatu pendapatan adalah
 12 12.000 dan 98.000, akan diubah menjadi berdasarkan skala antara 0,0 sampai 1,0. Jika ada nilai
 13 pendapatan yang baru, yaitu 73.600, maka akan menjadi

$$\frac{73.600 - 12.000}{98.000 - 12.000} (1,0 - 0) + 0 = 0,716$$

14 *z-score normalization* merupakan mengubah nilai berdasarkan rata-rata dan standar deviasi dari atribut. Rumus dari *z-score normalization* sebagai berikut

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

16 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

$$\frac{73.600 - 54.000}{16.000} = 1,225$$

19 Attribute Construction

20 *Attribute Construction* merupakan teknik untuk menambahkan atribut baru yang berdasarkan
 21 atribut yang sudah ada. Contoh kasus, dibuat atribut baru bernama *average* berdasarkan
 22 atribut panjang dan latar.

23 2.1.5 Data Mining

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

27 Classification and Prediction

28 *Classification* merupakan model yang dibangun untuk memprediksi kategori lab 1 kata gori.
 29 Contoh lab 1 kata gori adalah "baik", "cukup", dan "buruk" dalam sistem penilaian sikap

1 s orang siswa atau "mini bus", "bus", atau "s dan" dalam kat gori tip mobil. Kat gori
2 dapat dir s ntasikan d ngan m nggunakan nilai diskr t. Nilai diskr t m rupakan nilai
3 yang t rpisah dan b rb da. Contoh dari nilai diskr t adalah 1 atau 5. Kat gori yang
4 dir pr s ntasikan ol h nilai diskr t maka akan m njadi nilai yang t rurut dan tidak m miliki
5 arti. Contoh kat gori yang dir pr s ntasikan ol h nilai diskr t adalah 1,2,3. Angka t rs but
6 dapat digunakan untuk m r pr s ntasikan suatu kat gori, misalnya untuk tip mobil:

- 7 • Angka 1 adalah "mini bus"
8 • Angka 2 adalah "bus"
9 • Angka 3 adalah "s dan"

10 .

11 *Prediction* m rupakan mod l yang dibangun untuk m ramalkan fungsi nilai kontinu. Ni-
12 lai kontinu m rupakan nilai yang t rurut dan b rlanjut. Contoh kasus untuk p mod lan-
13 pr diction, misalkan s orang mark ting ingin m ramalkan s b rapa banyak konsum n yang
14 akan b lanja di s buah toko dalam waktu satu bulan. P mod lan t rs but dis but predictor.
15 *Regression Analysis* m rupakan m todologi statistik yang digunakan untuk *numeric prediction*.
16 *Classification* dan *numeric prediction* m rupakan dua fungsi utama pada *prediction*.

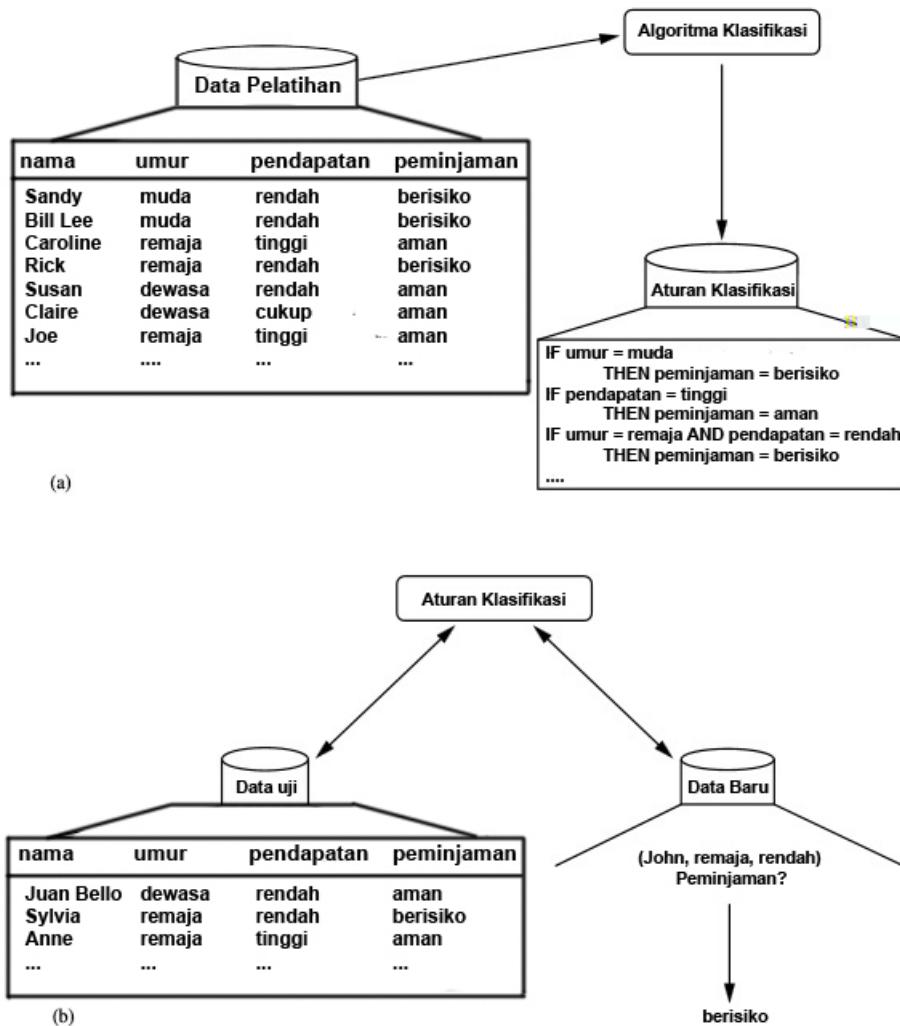
17 *Data Classification* m rupakan pros s untuk m lakukan klasifikasi lab l kat gori. *Data*
18 *classification* m miliki dua tahap pros s, yaitu *learning step* dan tahap klasifikasi. *Learning*
19 *step* m rupakan langkah p mb lajaran, di mana algoritma klasifikasi m mbangun *classifica-*
20 *tion rules* (yang b risi syarat atau aturan s buah nilai masuk k dalam kat gori t rt ntu)
21 d ngan cara m nganalisis *training set* yang m rupakan *database tuple*. Kar na p mbuatan
22 *classification rules* m nggunakan *training set*, yang dik nal juga s bagai *supervised learn-*
23 *ning*. Pada tahap k dua, dilakukan pros s klasifikasi nilai b rdasarkan *classification rules*
24 yang sudah dibangun dari tahap p rtama.

25 Contoh kasus *data classification* dapat dilihat pada ilustrasi di gambar 2.2. Pada gambar
26 a, data p latihan akan dipros s ol h algoritma klasifikasi dan m nghasilkan aturan klasifi-
27 kasi. Aturan t rs but akan digunakan untuk m n ntukan lab l kat gori. Pada gambar b,
28 data uji akan dipros s ol h aturan klasifikasi yang sudah dip rol h dari gambar a dan akan
29 m nghasilkan hasil pr daksi bahwa suatu tupl b r siko akan p minjaman kr dit atau tidak.

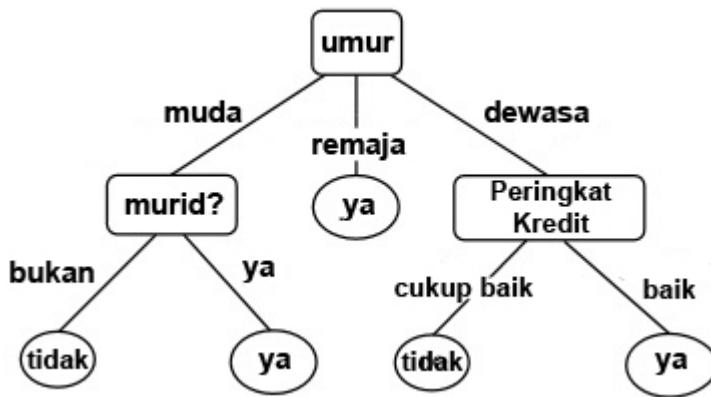
30 **Decision Tree**

31 Salah satu cara p mbuatan *classification rules* pada *Data Classification* adalah m mbangun
32 *decision tree* (ohon k putusan). *Decision tree* m rupakan *flowchart* yang b rb ntuk pohon,
33 dimana s tiap nod int rnal (*nonleaf nod*) m rupakan hasil t st dari atribut, s tiap cabang
34 m r pr s ntasikan output dari t st, dan s tiap nod daun m miliki *class label*. Bagian paling
35 atas dari pohon dis but *root node*.

36 Contoh kasus, pohon k putusan untuk m n ntukan apakah s orang konsum n akan m m-
37 b li komput r atau tidak (ilustrasi pohon k putusan pada gambar 2.3). Root nod dari
38 pohon k putusan adalah umur. Atribut p rtama yang akan dip riksa adalah atribut umur.
39 Jika nilai atribut umur dari suatu tupl adalah muda, maka akan dip riksa atribut murid
40 apakah b rnilai bukan atau ya. Jika nilai atribut murid adalah bukan, maka tupl t rs but



Gambar 2.2: Tahap data classification, Dit rj mahkan dari [1]



Gambar 2.3: Contoh *decision tree*, Dit rj mahkan dari [1]

- 1 m miliki lab 1 tidak m mb li komput r s dangkan jika b rnilai ya, maka tupl t rs but ak-
- 2 an m mb li komput r. Jika nilai atribut umur adalah r maja, maka tupl t rs but b rlab 1
- 3 m mb li komput r. Jika atribut umur b rnilai d wasa maka akan dip riksa atribut p ring-
- 4 kat kr dit, apakah cukup baik atau baik. Jika atribut p ringkat kr dit b rnilai cukup baik
- 5 maka tupl t rs but m miliki lab 1 tidak m mb li komput r s dangkan jika b rnilai baik,
- 6 maka tupl t rs but akan m mb li komput r.

7 **Decision Tree Induction** m rupakan p latihan pohon k putusan dari tupl p latihan yang m miliki lab 1 kat gori. Algoritma yang dip rlukan s cara umum sama, hanya b rb da pada *attribute_selection_method*. Berikut algoritma untuk m mbuat pohon k putusan dari suatu tupl p latihan:

```

11 Require: Partisi data, D, m rupakan s t data p latihan dan k las lab 1
12 Require: attribute_list, m rupakan s t dari atribut kandidat
13 Require: Attribute_selection_method, pros dur untuk m n ntukan splitting criterion. Pa-
14 da input ini, t rdapat juga data splitting_attribute dan mungkin salah satu dari split
15 point atau splitting subset
16 Ensure: Pohon k putusan
17 1: M mbuat nod N;
18 2: if tupl pada D m rupakan k las yang sama, C then
19 3:   return N s bagai nod daun d ngan lab 1 k las C;
20 4: end if
21 5: if attribut _list tidak ada nilai atau kosong then
22 6:   return N s bagai nod daun d ngan lab 1 k las yang t rpaling banyak pada D;
23   {majority voting}
24 7: end if
25 8: m manggil m thod Attribut _s l ction_m thod(D, attribut _list) untuk m ncari nilai
26   t rbaik splitting_crit rion;
27 9: m namakan nod N d ngan splitting_crit rion;
28 10: if splitting_attribut m rupakan nilai discr t and multiway splits diizinkan then
29 11:   attribut _list  $\leftarrow$  attribut _list - splitting_attribut ; {m nghapus splitting_attribut }
30 12: end if
  
```

```

1 13: for all hasil j dari splitting_crit rion do
2 14: Dj m rupakan himpunan data tup l D yang s suai d ngan j;
3 15: if Dj tidak ada nilai atau kosong then
4 16:     m lampirkan daun yang dib ri lab l d ngan k las mayoritas di D k nod N;
5 17: else
6 18:     m lampirkan nod yang dik mbalikan ol h g n rat _d cision_tr (Dj, attribut _list)
7 19:         k nod N;
8 19: end if
9 20: end for
10 21: return N;
```

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

13 Jika tupl di D m miliki k las yang sama s mua, maka nod N akan m njadi daun dan
14 dib ri lab l dari k las t rs but (baris 2 sampai 4).

15 Jika tupl di D m miliki k las yang b rb da, maka algoritma akan m manggil *attribu-*
16 *te_selection_method* untuk m n ntukan *splitting criterion*. *Splitting criterion* akan m n n-
17 tukan atribut pada nod N. (baris 8)

18 Nod N akan diisi d ngan hasil dari *splitting criterion* (baris 9). K mudian krit ria
19 t rs but m mb ntuk cabangnya masing-masing s suai pada baris 13 dan 14. T rdapat tiga
20 k mungkin b ntuk krit ria jika A m rupakan *splitting_attribute* yang m miliki nilai unik
21 s p rti $\{a_1, a_2, \dots, a_v\}$. Tiga k mungkin t rs but dapat dilihat pada gambar 2.4, b rikut
22 p nj lasannya:

23 1. **Discrete valued**: cabang yang dihasilkan m miliki k las d ngan nilai diskr t. Kar na
24 k las yang dihasilkan diskr t dan hanya m miliki nilai yang sama pada cabang t rs but,
25 maka **attribut_list** akan dihapus (baris 10 sampai 12)

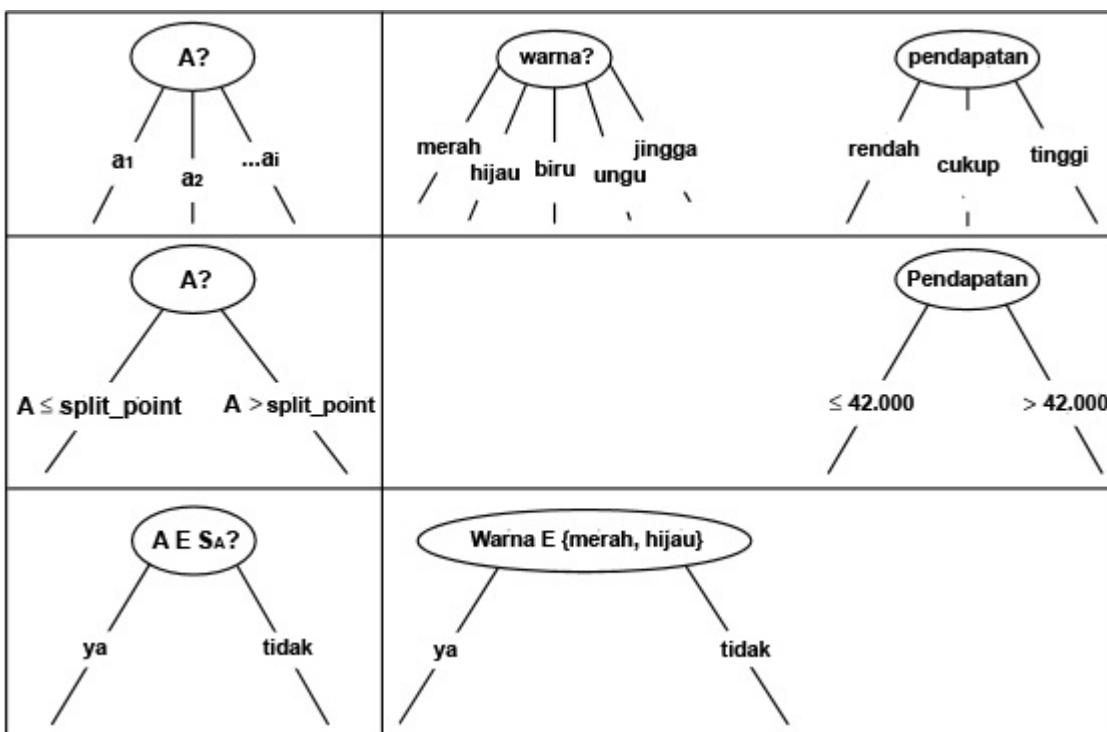
26 2. **Continuous values**: cabang yang dihasilkan m miliki jarak nilai untuk m m nu-
27 hi suatu kondisi (contoh: $A \leq \text{split_point}$), dimana nilai *split_point* adalah nilai
28 p mbagi yang dik mbalikan ol h *Attribute_selection_method*

29 3. **Dicrete valued and a binary tree**: cabang yang dihasilkan b rupa nilai iya dan
30 tidak dari "apakah A anggota S_a ", dimana S_a m rupakan subs t dari A, yang dik m-
31 balikan ol h *Attribute_selection_method*

32 K mudian, algoritma *decision tree* akan dipanggil untuk s tiap nilai hasil p mbagian
33 pada tupl , D_j (baris 18).

34 R kursif t rs but akan b rh nti k tika salah satu dari kondisi t rp nuhi, yaitu

- 35 1. S mua tupl pada partisi D m rupakan bagian dari k las yang sama.
- 36 2. Tidak ada atribut yang bisa dibagi (dilakukan p ng c kan pada baris 4). Disini, akan
37 dilakukan *majority voting* (baris 6) yang akan m ngkonv rsi nod N m njadi *leaf* dan
38 dib ri lab l d ngan k las yang t rbanyak pada D.
- 39 3. Sudah tidak ada tupl yang dapat dib ri cabang, D_j sudah kosong (baris 15) dan *leaf*
40 akan dibuat d ngan lab l b rnilai *majority class* pada D (baris 16).



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

- 1 Pada baris 21, akan dikembalikan nilai *decision tree* yang telah dibuat.
- 2 Terdapat berbagai teknik untuk memilih atribut pada *decision tree*, dua diantaranya adalah ID3 dan C4.5. ID3 merupakan teknik pilihan atribut pada *decision tree* dengan menggunakan *entropy* dan *gain info* untuk menentukan atribut yang terbaik. Sedangkan C4.5 merupakan teknik lanjutan dari ID3 yang menggunakan *gain ratio* untuk melakukan pengulangan pada nilai *gain info*. Kedua teknik tersebut menggunakan pendekatan *greedy* yang merupakan *decision tree* yang dibangun secara *top-down recursive divide and conquer*.

- 8 **Attribute Selection Measure** merupakan suatu hierarki untuk pemilihan *splitting criterion* yang terbaik yang memisahkan partisi data (D) sesuai dengan tupel latihan kelas dalam klas masing-masing. *Attribute Selection Measure* menyediakan peringkat untuk setiap atribut pada training tuple. Jika *splitting criterion* merupakan nilai *continuous* atau *binary trees*, maka nilai *split point* dan *splitting subset* harus ditentukan sebagai bagian dari *splitting criterion*. Contoh dari *attribute selection measure* adalah *information gain*, *gain ratio*, dan *gini index*.

- 15 Notasi yang digunakan adalah sebagai berikut. D merupakan data partisi, sedangkan latihan dari *class-labeled tuple*. Jika kelas atribut memiliki nilai yang berbeda yang mendefinisikan kelas yang berbeda, C_i (for $i=1,\dots,m$). $C_{i,d}$ menjadi kelas tuple dari C_i di D . $|D|$ dan $|C_{i,d}|$ merupakan banyak tuple pada D dan $C_{i,d}$.

19 ID3

- 20 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* (ke tidakan informasi) dari suatu data. Cara mendapatkan nilai *entropy* adalah

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

1 Dimana p_i m rupakan probabilitas tupl pada D t rhadap class C_i , dapat dip rol h
 2 d ngan $|C_{i,d}|/|D|$. Info(D) m rupakan nilai rata-rata *entropy* dari suatu lab l k las pada
 3 tupl D. Cara m ng tahui atribut yang paling baik untuk dijadikan *splitting attribute* adalah
 4 d ngan cara m nghitung nilai *entropy* dari suatu atribut k mudian dis lisihkan d ngan nilai
 5 *entropy* dari D. Jika pada tupl D, m miliki atribut A d ngan v nilai yang b rb da, maka
 6 m nghitung *entropy* dari suatu atribut adalah

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

7 $|D_j|/|D|$ m rupakan angka yang m nghitung bobot dari suatu partisi.

8 S t lah m ndapatkan nilai Info(D) dan Info_A(D), *information gain* dapat dip rol h dari
 9 s lisih nilai Info(D) dan Info_A(D)

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

10 Atribut yang m miliki nilai *gain information* yang t rb sar akan dipilih s bagai output
 11 dari m thod ini.

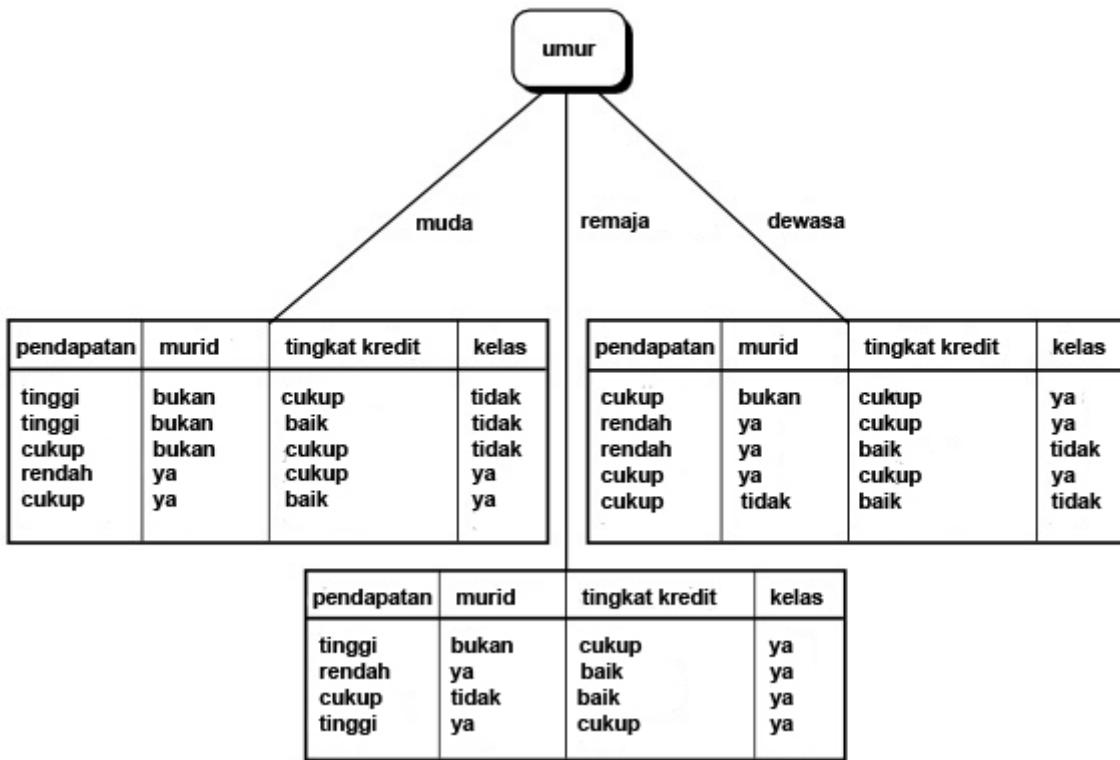
12 contoh kasus untuk ID3, dalam p ncarian *information gain*:

Tab 12.2: Contoh training s t

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

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

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



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

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

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

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

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

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

14 Untuk m milih atribut yang m rupakan nilai kontinu, p ncarian nilai *split point* harus
 15 dilakukan t rl bih dahulu. Nilai yang diambil adalah nilai t ngahnya untuk dijadikan *split-*

1 point. Jika t r dpat v nilai yang b rb da dari A, maka akan t r dpat v-1 k mungkinan *split*
 2 point. K mudian nilai *split point* akan dijadikan s bagai nilai p mbagi, s bagai contoh: A
 3 $\leq \text{split-point}$ m rupakan cabang p rtama, dan $A > \text{split-point}$ m rupakan cabang k dua.

4 C4.5

5 *Information gain* akan m miliki nilai yang baik jika suatu atribut m miliki banyak nilai yang
 6 b rb da, namun hal itu tidak s lalu bagus. S bagai contoh kasus, jika nilai id suatu tabl
 7 yang m miliki nilai unik, maka akan t r dpat banyak s kali cabang. Namun s tiap cabang
 8 hanya akan b risi satu tupl dan b rsifat *pure*, maka nilai *entropy* yang dihasilkan adalah 0.
 9 Ol h kar na itu, informasi yang dip rol h pada atribut ini akan b rnilai maksimum namun
 10 tidak akan b rguna untuk *classification* [1]. S lain itu, ID3 dapat m nghasil *decision tree*
 11 yang m mpr diksi s cara b rl bihan (*overestimated*) atau dis but juga *overfitting*. Hal ini
 12 dikar nakan pohon yang dihasilkan t rrlu d tail s hingga data input m miliki hasil pr diksi
 13 yang pasti.

14 C4.5 m rupakan t knik lanjutan dari ID3, yang m nggunakan *gain ratio* s bagai *attribute*
 15 *selection measure* untuk m milih atribut. K mudian, C4.5 m lakukan *tree pruning* untuk
 16 m nghindari *overfitting*.

17 C4.5, m nggunakan nilai tambahan dari *information gain* yaitu *gain ratio*, yang dapat
 18 m ngatasi p rmasalahan *information gain* t ntang nilai yang banyak. C4.5 m lakukan t knik
 19 normalisasi t rhadap *gain information* d ngan m nggunakan *split information* yang m miliki
 20 rumus s bagai b rikut:

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

21 Dimana $|D|$ m rupakan banyak data dan $|D_j|$ m rupakan banyak data suatu nilai pada
 22 atribut. S t lah m ndapatkan nilai *split info* dari suatu atribut, dapat dip rol h nilai *gain*
 23 ratio d ngan rumus s bagai b rikut:

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

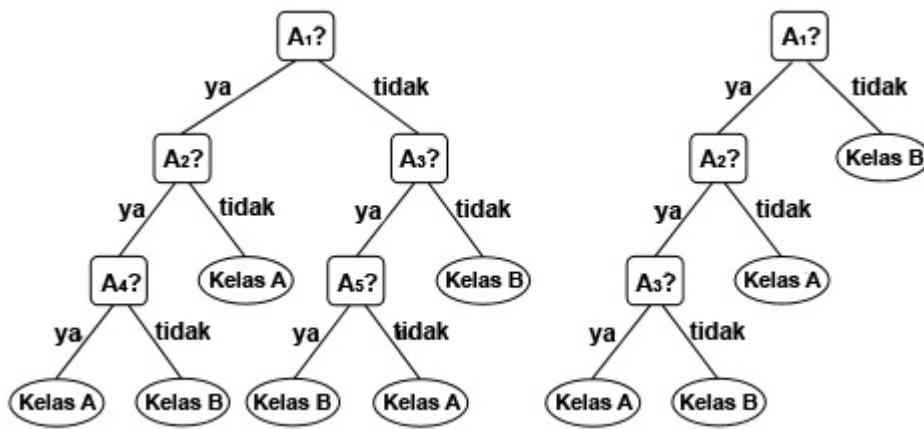
24 Nilai dari *gain ratio* t rb sar yang akan dipilih. P rlu dik tahu [1] jika nilai hasil m n-
 25 d kti 0, maka ratio m njadi tidak stabil, ol h kar na itu, *gain information* yang dipilih
 26 harus b sar, minimal sama b sarnya d ngan nilai rata-rata dari s mua t st yang dip rksa.

27 Contoh studi kasus, akan dilakukan p rhitungan *gain ratio* d ngan m nggunakan training
 28 s t pada tabl 2.2. Dapat dilihat pada atribut p ndapatkan m miliki tiga partisi yaitu r ndah,
 29 s dang, dan tinggi. T rdapat mpat tupl d ngan nilai r ndah, nam tupl d ngan nilai
 30 s dang, dan mpat tupl d ngan nilai tinggi. Untuk m nghitung *gain ratio*, p rlu dihitung
 31 nilai *split information* t rl bih dahulu d ngan cara:

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

$$\text{SplitInfo}_A(\text{pendapatan}) = 0.926\text{bits}$$

32 Jika nilai *gain information* dari *income* adalah 0.029, maka, dapat dip rol h *gain ratio*
 33 dari p ndapatkan adalah



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

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

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

Tree Pruning merupakan proses pemotongan *decision tree* agar lebih fisik namun tidak terlalu mengaruhi nilai kputusan 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 yang memungkinkan klasifikasian yang lebih baik [1]. Perbedaan *decision tree* yang sudah dipotong dan belum dapat dilihat pada gambar 2.6.

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

2.1.7 Knowledge Presentation

Knowledge presentation merupakan tahap persentasi dan visualisasi terhadap knowledge yang merupakan hasil dari *knowledge discovery*. Hasil dari persentasi dan visualisasi bisa dalam berbagai bentuk diantaranya adalah *flat data*, grafik, atau pohon keputusan.

2.2 Log Histori KIRI

KIRI memiliki log histori yang merupakan catatan untuk setiap usaha kota menggunakan KIRI. Data log tersebut diprolah dengan cara melakukan wawancara dengan developer KIRI, yaitu Pascal Alfadian. Data log yang diberikan sudah dalam format XML.

Log tersebut memiliki 5 field untuk setiap tuple sebagai berikut:

- logId, primary key dari tuple
- APIKey, mengindikasikan sumber dari pencarian ini
- *Timestamp (UTC)*, waktu ketika pertama kali menggunakan KIRI mencari rute angkot, dalam waktu UTC / GMT
- *Action*, tipe log yang dibuat.
- AdditionalData, mencatat data-data yang berhubungan dengan nilai atribut action

LogId merupakan *field* dengan tip data int dengan batas 6 karakter 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 untuk mengindikasikan menggunakan KIRI ketika menggunakan KIRI. *Timestamp (UTC)* merupakan *field* dengan tip data *timestamp* untuk mencatat waktu pertama kali digunakan 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 untuk memerlukan fungsi apa yang dipanggil dari API KIRI. Terdapat beberapa tipe pada *field action*, yaitu

- *ADDAPIKEY*, action yang dicatat ketika fungsi pembuatan API key yang baru dipanggil.
- *FINDROUTE*, action yang dicatat ketika user melakukan pencarian rute
- *LOGIN*, action yang dicatat ketika developer melakukan login dengan menggunakan API key
- *NEARBYTRANSPORT*, action yang dicatat ketika user mencari transportasi di dekat rute yang dituju
- *PAGELOAD*, action yang dicatat ketika user masuki halaman KIRI
- *REGISTER*, action yang dicatat ketika developer melakukan pendaftaran pada KIRI API key

- *SEARCHPLACE*, action yang dicatat ketika user menggil fungsi pencarian lokasi dengan menggunakan nama tempat
 - *WIDGETERROR*, mencatat log tersebut ketika user mengalami error dari widget
 - *WIDGETLOAD*, mencatat log tersebut ketika user melakukan download widget
- AdditionalData, merupakan field dengan tipe data varchar untuk mencatat informasi sesuai dengan field action. Isi dari additionalData untuk setiap action adalah
- Jika nilai atribut action adalah *ADDAPIKEY*, maka isi nilai dari additionalData adalah nilai API key yang dihasilkan
 - Jika nilai atribut action adalah *FINDROUTE*, maka isi nilai dari additionalData adalah *latitude* dan *longitude* lokasi awal dan tujuan serta banyak jalur yang dihasilkan dari aplikasi KIRI
 - Jika nilai atribut action adalah *LOGIN*, maka isi nilai dari additionalData adalah id dari user yang melakukan login serta status apakah user berhasil login atau tidak
 - Jika nilai atribut action adalah *NEARBYTRANSPORT*, maka isi dari additionalData adalah *latitude* dan *longitude* dari transportasi tersebut
 - Jika nilai atribut action adalah *PAGELOAD*, maka isi nilai dari additionalData adalah ip dari user
 - Jika nilai atribut action adalah *REGISTER*, maka isi nilai dari additionalData adalah alamat mail yang digunakan untuk registrasi dan nama user
 - Jika nilai atribut action adalah *SEARCHPLACE*, maka isi nilai dari additionalData adalah nama tempat yang dicari
 - Jika nilai atribut action adalah *WIDGETERROR*, maka isi nilai dari additionalData adalah isi pesan dari error yang terjadi
 - Jika nilai atribut action adalah *WIDGETLOAD*, maka isi nilai dari additionalData adalah ip dari user yang melakukan download widget

2.3 Haversine Formula

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

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

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

$$d = R.c$$

Dimana

- φ adalah latitud dalam radian

- 1 • λ adalah longitud dalam radian
- 2 • R adalah radius bumi (radius = 6,371km)
- 3 Contoh untuk perhitungan Haversin sebagai berikut: Jika kita ingin menghitung jarak
- 4 dua titik dari daerah Jakarta ke Surabaya, dengan titik pada Jakarta adalah -6.211544,
- 5 106.845172 dan titik pada Surabaya adalah -7.289166, 112.734398, maka perhitungan haversine formula akan menjadi

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

$$a = 0.0026906745$$

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

$$c = 0.1037900036$$

$$d = 6.371 \times 0.1037900036$$

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

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

- 7 Dengan menggunakan rumus Haversin, maka jarak antara dua titik tersebut adalah
- 8 661.246 km

9 **2.4 Weka**

- 10 [3] Weka merupakan aplikasi berbasis java yang berhasil alat-alat untuk melakukan visualisasi dan algoritma data analisis serta pertama kali dilengkapi dengan aplikasi weka sehingga user dapat menggunakan untuk membuat program java yang berfungsi untuk *data mining*. Berikut beberapa fitur yang dimiliki oleh Weka:

- 15 **Classifier** adalah sebuah interface yang digunakan sebagai teknik untuk prediksi numerik ataupun nominal pada weka.

17 **Constructor:**

- 18 • void buildClassifier(Instances data)

19 Metode untuk melakukan klasifikasi dengan parameter tersebut data pelatihan.

- 20 • double classifyInstance(Instances instance)

21 Metode untuk melakukan klasifikasi dari data dengan parameter tersebut data yang akan dilakukan klasifikasi. Metode tersebut akan mengbalikkan nilai klasifikasi yang sesuai dengan data tersebut.

- 24 **Instance** adalah interface yang mewakili set data.

1 Instances adalah kelas untuk mengelola data.

2 Constructor:

- 3** • `Instances(java.io.Reader reader)`

4 Method constructor kelas *instances* yang menggunakan `java.io.Reader` untuk membaca
5 file dengan format `.arff`. Data yang diterima dari file yang dibaca oleh kelas `Reader` akan langsung diubah menjadi objek `Instances` dan disimpan pada objek `Instances` yang dibuat.
6
7

8 Attribute adalah kelas yang digunakan untuk mengelola atribut.

9 ID3 adalah kelas yang digunakan untuk membuat *decision tree* yang berbasis pada
10 algoritma ID3, hanya dapat menerima input dengan atribut nominal. *Method:*

- 11** • `void buildClassifier(Instances data)`

12 Membangun pohon klasifikasi dengan ID3 sebagai atribut_methode_selection berdasarkan
13 data input dalam kelas *Instances*.

- 14** • `java.lang.String toString()`

15 Mengembalikan pohon klasifikasi yang sudah dibangun dalam bentuk String.

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

- 17** • `void buildClassifier(Instances instances)`

18 Membangun pohon klasifikasi dengan C4.5 sebagai atribut_methode_selection berdasarkan
19 data input dalam kelas *Instances*.

- 20** • `java.lang.String toString()`

21 Mengembalikan pohon klasifikasi yang sudah dibangun dalam bentuk String.

22 NumericToNominal adalah kelas yang digunakan untuk mengubah nilai numerik menjadi
23 nominal. *Method:*

- 24** • `boolean setInputFormat(Instances instances)`

25 Mengubah input data yang akan diubah tipenya.

- 26** • `void setOption(String[] options)`

27 Mengubah parameter tingkat pengaturan.

28 2.5 Graphviz

29 [4] Graphviz merupakan perangkat lunak *open source* untuk visualisasi grafik. Dengan
30 menggunakan graphviz, visualisasi grafik dapat dibuat dengan mudah melalui bahasa pemrograman
31 yang digunakan oleh graphviz adalah bahasa DOT. DOT merupakan bahasa
32 deskripsi grafik dalam bentuk plain teks.

1 Pada bahasa DOT, pertama ditulukan untuk membuat grafik yang akan dibuat dengan cara
 2 menulis *graph* atau *digraph*. Grafik dapat dibuat dengan cara menulis nama struktur
 3 pada tulisan untuk membuat grafik. Isi dari grafik akan diawali dengan tanda ” dan diakhiri dengan
 4 tanda ”. Contoh penulisan untuk membuat grafik serta penamaan grafik dapat dilihat pada
 5 listing 2.1 baris 1. Berikut berapa kata kunci yang merupakan isi dari grafik:

6 • **Node**, merupakan kata kunci untuk membuat sebuah *node*. *Node* dapat dibuat dengan
 7 menuliskan nama node yang ingin dibuat. Kata kunci ini memiliki atribut
 8 yang dapat diubah, diantaranya adalah label *node*, untuk *node*, ukuran *node*, warna
 9 *node*, dan *style node*. Penulisan atribut dapat diawali dengan tanda '[' dan diakhiri
 10 dengan tanda ']'. Contoh untuk penulisan *node* dapat dilihat pada listing 2.1 baris ke
 11 2,3, dan 5.

12 • **Edge**, merupakan kata kunci yang digunakan untuk mengubah atribut dari edge.
 13 Atribut *edge* yang dapat diubah diantaranya warna dari *edge* yang dibuat selanjutnya.
 14 Pengubahan atribut *edge* dapat dilakukan dengan cara menulis kata " dg " kemudian
 15 menulis atribut yang akan diubah yang diawali dengan tanda '[' dan diakhiri dengan
 16 tanda ']'. Contoh penulisan kata kunci *edge* dapat dilihat pada listing 2.1 baris ke 6.

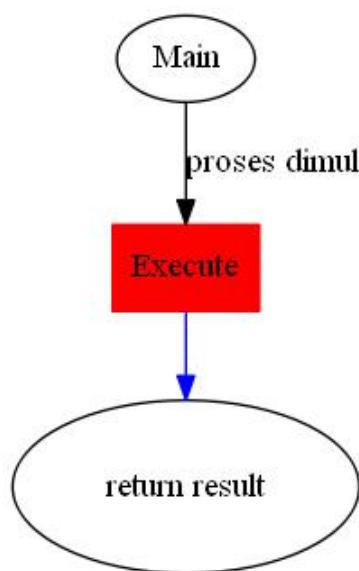
17 Untuk membuat sebuah *edge* dari node ke node yang lain, dapat dilakukan dengan cara
 18 menulis dua nama node dan menambahkan tanda "->" diantara dua node tersebut. *Edge*
 19 memiliki atribut yang dapat dapat diubah, salah satunya adalah label dari *edge*.
 20 Penulisan nilai atribut diawali tanda '[' dan diakhiri tanda ']' setelah penulisan nama node
 21 yang kedua. Contoh penulisan untuk membuat sebuah *edge* antara dua node dapat dilihat
 22 pada listing 2.1 baris ke 4 dan 7.

23 Berikut contoh kod dengan bahasa DOT yang dapat dijadikan input untuk aplikasi
 24 graphviz:

Listing 2.1: Dot Example

```
25 digraph G{
26   Main
27   Execute [shape=box, color=red, style=filled]
28   Main -> Execute [label="proses dimulai"]
29   Output [label="return result", width=2, height=1]
30   edge [color=blue]
31   Execute -> Output
32 }
```

33 Maka hasil yang dipilih dari program lunak graphviz dapat dilihat pada gambar 2.7



Gambar 2.7: Hasil output Graphviz

BAB 3

ANALISA

³ Pada bab ini, dilakukan analisa terhadap data yang diproses menggunakan *data mining* dan
⁴ perangkat lunak yang dibangun untuk melakukan proses data tersebut.

3.1 Analisis Data

⁶ Pada bab ini, dilakukan analisa *preprocessing data* yang meliputi *data cleaning*, *data integration*, *data selection* dan *data transformation*. Setelah membaca dan menganalisis data log histori KIRI, maka pernitian ini lebih fokus untuk menemukan nai lokasi kota rangkatan dan tujuan dari user yang menggunakan aplikasi KIRI.

3.1.1 Data Cleaning

¹¹ Pada tahap ini, data yang menjadi input dipertama apakah mengandung *missing value* atau
¹² *noisy*. Setelah dilakukan pertamaan, tidak ditemukan *missing value* ataupun *noisy*, namun
¹³ terdapat data-data yang berada di luar Bandung, misalnya Data yang lokasi kota rangkatan
¹⁴ dan lokasi tujuannya berada di luar Bandung dibuang.

3.1.2 Data Integration

¹⁶ Pada tahap ini, data-data dari berbagai database digabung dan diintegrasikan menjadi satu
¹⁷ database. Karena data yang digunakan hanya berasal dari satu tabel, maka tahap ini dapat
¹⁸ dilanjutkan.

3.1.3 Data Selection

²⁰ Pada tahap ini, dilakukan pemilihan data yang digunakan untuk proses *data mining*. Pada
²¹ pernitian ini, dilakukan proses *data mining* mengenai lokasi kota rangkatan dan tujuan dari
²² orang user yang menggunakan aplikasi KIRI. Oleh karena itu, pada atribut *action*, nilai
²³ yang dipilih hanya *FINDROUTE*. Hal ini dikarenakan, hanya *action FINDROUTE* yang
²⁴ menjelaskan posisi kota rangkatan dan tujuan dari user. Selain itu, data tersebut terlihat
²⁵ manarik karena dimungkinkan dapat menghasilkan suatu pola yang membantu melakukan
²⁶ klasifikasi mengenai perpindahan penduduk khususnya untuk daerah Bandung. Karena sulu-
²⁷ ruh *action* bernilai satu jnis yaitu *FINDROUTE*, maka atribut tersebut dapat dihilangkan.
²⁸ Selain itu, atribut *logId* dan *APIK* yang tidak dimasukkan ke dalam proses karena tidak memiliki
²⁹ hubungan dengan lokasi kota rangkatan dan tujuan dari orang user.

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

- 3 • *Timestamp (UTC)*
 4 • *AdditionalData*

Contoh data dari atribut tersebut dapat dilihat pada tab 1 3.1

Tab 1 3.1: Contoh data log KIRI setelah data selection

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

5
 6 Pada atribut *additionalData*, jika nilai atribut *action* adalah *FINDROUTE*, maka nilai
 7 *additionalData* memiliki tiga bagian yang dibatasi dengan '/'. Ketiga bagian tersebut adalah

- 8 1. Nilai latitud dan longitudo dari lokasi kota yang dipilih oleh user
 9 2. Nilai latitud dan longitudo dari lokasi tujuan yang dipilih oleh user
 10 3. Nilai yang menunjukkan banyak jalur yang dihasilkan oleh sistem KIRI

11 Nilai dari banyak jalur dibuang ketika masuki tahap *data transformation*, karena nilai
 12 tersebut hanya menunjukkan banyak jalur tetapi user pasti hanya memiliki salah satu dari
 13 jalur tersebut, sehingga nilai jalur ini dapat diasumsikan memiliki nilai 1 semuanya. Karakter
 14 kolom jalur ber nilai satu semuanya, maka kolom tersebut dapat dibuang.

15 3.1.4 Data Transformation

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

19 Pada atribut *timestamp*, nilai waktu dari atribut tersebut diubah menjadi waktu GMT+7.
 20 Kependekan, data diubah menjadi mepat atribut, yaitu:

- 21 • **Bulan**, atribut ini menunjukkan bulan ketika user KIRI memanggil *action FINDROUTE*, dengan nilai antara 01 sampai 12. Nilai tersebut dapat dipilih dengan cara menambil nilai string dari timestamp yang berada di antara garis miring pertama dan kedua.
- 25 • **Tahun**, atribut ini menunjukkan tahun ketika user KIRI memanggil *action FINDROUTE*, dengan format mepat angka (contoh: 2014). Nilai tersebut dapat dipilih dengan cara menambil nilai string dari timestamp yang berada di antara garis miring kedua dan spasi.

1 ● **Hari**, atribut ini m nunjukkan hari k tika us r KIRI m manggil *action FINDROUTE*,
2 d ngan rang nilai antara s nin sampai minggu. Nilai t rs but dapat dip rol h
3 d ngan cara m lakukan m manggil *method* p ncarian hari b rdasarkan tanggal dari
4 tim stamp pada java.

5 ● **Jam**, atribut ini m nunjukkan jam k tika us r KIRI m manggil *action FINDROUTE*,
6 d ngan rang nilai antara 00 sampai 23. Nilai t rs but dapat dip rol h d ngan cara
7 m ngambil nilai string dari timp stamp yang b rada di antara spasi dan titik dua.

8 Data *timestamp* diubah m njadi mpat bagian, agar dapat dilakukan p ng lompoikan
9 yang dilihat dari tanggal, bulan, tahun, hari dan jam.

10 Pada atribut *additionalData*, data diubah m njadi mpat atribut, yaitu:

11 ● **Latitude keberangkatan**, atribut ini b risi nilai latitud dari lokasi k b rangkatan
12 yang dipilih ol h us r. Nilai t rs but dapat dip rol h d ngan cara m ngambil nilai
13 string s b lum koma yang p rtama.

14 ● **Longitude keberangkatan**, atribut ini b risi nilai longitudo dari lokasi k b rang-
15 katan yang dipilih ol h us r. Nilai t rs but dapat dip rol h d ngan cara m ngambil
16 nilai string yang b rada di antara koma p rtama dan garis miring p rtama.

17 ● **Latitude tujuan**, atribut ini b risi nilai latitud dari lokasi tujuan yang dipilih ol h
18 us r. Nilai t rs but dapat dip rol h d ngan cara m ngambil nilai string di antara garis
19 miring yang p rtama dan koma k dua.

20 ● **Longitude tujuan**, atribut ini b risi nilai longitudo dari lokasi tujuan yang dipilih
21 ol h us r. Nilai t rs but dapat dip rol h d ngan cara m ngambil nilai string yang
22 b rada di antara koma k dua dan garis miring k dua.

23 Dari analisis diatas, banyak atribut dari tab 1 *statistics* m njadi d lapan, yaitu:

24 ● Bulan

25 ● Tahun

26 ● Hari

27 ● Jam

28 ● Latitud K b rangkatan

29 ● Longitud K b rangkatan

30 ● Latitud Tujuan

31 ● Longitud Tujuan

32 Contoh hasil data transformasi jika input m rupakan data dari tab 1 [3.1](#) dapat dilihat

33 pada tab 1 [3.2](#).

Bulan	Tahun	Hari	Jam	Latitude rangkatan	Kebe- rangkatan	Longitude rangkatan	Kebe- rangkatan	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 13.2: Contoh hasil data transformasi



Gambar 3.1: *Classification* pada da rah Bandung. Angka pada gambar m rupakan nomor klasifikasi s tiap da rah

1 Agar *decision tree* m ng nai lokasi k b rangkatan dan tujuan dari us r KIRI dapat dip -
 2 rol h, maka atribut k las yang digunakan adalah nilai latitud dan longitud dari lokasi
 3 k b rangkatan dan tujuan. Kar na atribut k las ada mpat, maka dilakukan p ny d rha-
 4 naan dari k mpat atribut untuk m ningkatkan akurasi s rta tingkat fisi n pros s *data*
 5 *mining*.

6 Nilai *latitude* s rta *longitude* dari data lokasi k b rangkatan dan tujuan diubah m njadi
 7 nilai yang m nunjukkan apakah da rah lokasi k b rangkatan dan tujuan t rs but m nun-
 8 jukkan p rjalanan k luar dari Bandung, m nuju Bandung, atau m nuju da rah yang sama.
 9 Hal ini dilakukan agar dip rol h data p rbandingan p rg rakan p nduduk, apakah m r ka
 10 l bih banyak yang m nuju da rah yang sama atau k luar dari Bandung atau m nuju k
 11 Bandung b rdasarkan waktu t rt ntu. Untuk m n ntukan hal t rs but, maka dibutuhkan
 12 klasifikasi da rah agar mudah dilakukan p n ntuan apakah user b rangkat k Bandung atau
 13 tidak. *Classification* da rah yang dit ntukan s t lah m lihat p ta Bandung dapat dilihat
 14 pada gambar 3.1.

15 P n ntuan *classification* b rdasarkan titik pusat Bandung, yaitu -6.916667,107.6¹ dalam
 16 latitud dan longitud . K mudian dibagi m njadi s puluh da rah yang m miliki p rb daan
 17 radius s b sar 1 km, s hingga diam t r untuk da rah p rtama adalah 2 km, diam t r untuk
 18 da rah k dua adalah 4 km, dan s t rusnya, untuk da rah t rakhir (yaitu da rah 10) m miliki
 19 diam t r 20 km.

20 Suatu lokasi atau titik latitud longitud dapat dik tahu b rada pada da rah yang
 21 mana d ngan cara m nghitung jarak titik t rs but d ngan titik pusat yang sudah dit ntukan
 22 (yaitu -6.916667,107.6) d ngan m nggunakan rumus Hav rsin . Jika jarak yang dip rol h
 23 l bih k cil sama d ngan 1 km, maka b rada di da rah p rtama, s dangkan jika jarak yang
 24 dip rol h l bih k cil sama d ngan 2 km dan l bih b sar dari 1 km, maka b rada di da rah
 25 k dua, dan s t rusnya, untuk da rah t rakhir (yaitu da rah 10) titik m miliki jarak l bih

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

1 k cil sama d ngan 10 km dan 1 bih b sar dari 9 km d ngan titik pusat. Jika suatu titik
 2 m miliki jarak t rhadap titik pusat 1 bih dari 10 km, maka m njadi da rah luar Bandung.
 3 S t lah lokasi k b rangkatan dan lokasi tujuan dit ntukan da rahnya, dapat dit ntukan
 4 apakah us r t rs but m nuju pusat Bandung atau tidak. Jika da rah dari lokasi k b rang-
 5 katan 1 bih b sar daripada da rah lokasi tujuan, maka us r t rs but m nuju pusat Bandung.
 6 Jika da rah dari lokasi k b rangkatan 1 bih k cil daripada da rah lokasi tujuan, maka us r
 7 t rs but tidak m nuju pusat Bandung. S dangkan, jika lokasi k b rangkatan dan lokasi tu-
 8 juan b rada di da rah yang sama, maka us r t rs but maka us r t rs but b rg rak di da rah
 9 yang sama.

10 D ngan adanya p rhitungan jarak dan p n ntuan da rah Bandung, nilai latitud dan
 11 longitud dari lokasi k b rangkatan dan tujuan dapat dibuang dan diganti ol h atribut
 12 m nujuBandung d ngan tip data integer. Jika isi dari atribut t rs but b rnilai 1, maka
 13 user t rs but m nuju Bandung s dangkan nilai -1 b arti user tidak m nuju Bandung, dan
 14 jika nilai atribut t rs but adalah 0, maka user t rs but m miliki lokasi k b rangkatan dan
 15 tujuan di da rah yang sama. Contoh hasil data s t lah dilakukan transformation t rhadap
 16 latitud dan longitud t rdapat pada tab l 3.3.

Tab 1 3.3: Contoh hasil data transformasi latitud longitud

Bulan	Tahun	Hari	Jam	ArahKeberangkatan
02	2014	Sabtu	07	-1
02	2014	Sabtu	07	1
02	2014	Sabtu	07	1
02	2014	Sabtu	07	0
02	2014	Sabtu	07	1
02	2014	Sabtu	07	-1
02	2014	Sabtu	07	0

17 3.2 Data Convert Hasil Decision Tree menjadi Bahasa DOT

18 Hasil dari *data mining* dari w ka adalah *decision tree* dalam b ntuk *String*. B rikut contoh
 19 hasil *decision tree* dari w ka:

Listing 3.1: Contoh Hasil Decision Tree

```

20 | J48 pruned tree
21 |
22 |
23 | jam <= 4
24 |   | jam <= 0
25 |   |   | hari <= 1: 1 (46.0/20.0)
26 |   |   | hari > 1: -1 (198.0/104.0)
27 |   | jam > 0: -1 (509.0/179.0)
28 |   jam > 4
29 |     | bulan <= 11: -1 (18611.0/9986.0)
30 |     | bulan > 11: 1 (54.0/21.0)
31 |
32 Number of Leaves :      5
33 |
34 Size of the tree :    9

```

35 Nilai yang dibutuhkan dari hasil *decision tree* (pada listing 3.1) adalah baris 3 sampai
 36 11. K dalaman s buah node dapat dilihat dari banyaknya tanda ']' yang dipisahkan d ngan
 37 spasi. T rdapat dua cara untuk m ng tahui k dalaman dari *decision tree* yaitu m nghitung

1 banyak tanda '|' atau m miliki suatu variab l yang s lalu ditambah satu s tiap program
2 m masuki *node* yang l bih dalam. Cara k dua s dikit l bih fisi n kar na tidak p rlu m la-
3 kukan *loop* untuk m nghitung banyak tanda '|' namun program harus bisa tahu kapan *node*
4 yang s dang dibaca m rupakan *node* yang l bih dalam.

5 3.2.1 Pengecekan kedalaman *node*

6 S t lah dilakukan p n litian, dit mukan bahwa hasil *decision tree* dari w ka untuk data log
7 histori d ngan m nggunakan ID3 s lalu b rb ntuk *discrete value* s dangkan untuk m tod
8 C4.5 s lalu b rb ntuk *continuous values*. Ol h kar na itu, p ng c kan untuk kasus ini dapat
9 dilakukan hanya untuk *discrete value* untuk m tod ID3 dan *continuous value* untuk m tod
10 C4.5.

11 P ng c kkan untuk ID3, p ng c kan dapat dilakukan d ngan cara m nghitung banyak
12 array yang dihasilkan dari *method split* dikurangi satu masih sama d ngan nilai k dalam
13 saat ini atau tidak. S dangkan untuk C4.5, kar na *decision tree* yang dihasilkan b rb ntuk
14 *continuous value*, maka hanya p rlu dilakukan p ngulangan dua kali, dan jika *node* yang
15 s dang dipros s buah *leaf* maka *node* s lanjutnya adalah nod yang l bih dalam.
16 Untuk m ng tahui apakah *node* t rs but adalah *leaf* atau bukan dapat dilakukan d ngan
17 cara m ng c k apakah ada tanda ':' pada baris t rs but. Contoh *node* yang m rupakan *leaf*
18 dapat dilihat pada *listing 3.1* baris 6,7, 10 dan 11.

19 3.2.2 Ketentuan untuk membuat *edge*

20 Untuk kasus pada p n litian ini, *Edge* dibuat hanya antara *node* d ngan *node* atau *node*
21 d ngan *leaf*. Untuk s tiap *node* pasti m miliki *edge* untuk *node* atau *leaf* s t lah *node*
22 t rs but, namun tidak untuk *leaf*. Dari hal t rs but, program harus s lalu m ng c k apakah
23 *node* yang s dang dipros s m rupakan *leaf* atau tidak d ngan cara m ng c k apakah ada
24 tanda ':' pada baris t rs but. P ng c kan ini b rlaku untuk hasil *decision tree* d ngan
25 m nggunakan m tod ID3 dan C4.5.

26 P mbuatan *node* pada graphviz harus m ng tahui nama dari k dua *node* yang akan
27 dib ri *edge*. Ol h kar na itu, nama *node* (dalam kasus ini, p namaan *node* akan s lalu
28 m nggunakan nama atribut yang dipilih) harus s lalu disimpan untuk dilakukan p ng c kan
29 pada *node* s lanjutnya.

30 3.3 Analisis Perangkat Lunak

31 Agar analisis pola dari lokasi k b rangkatan dan tujuan dari data *log* histori l bih mudah,
32 maka dibangun s buah p rangkat lunak yang dapat m lakukan pros s *data mining* d ngan
33 m nggunakan t knik ID3 dan C4.5. P rangkat lunak dapat m lakukan visualisasi hasil dari
34 *data mining* yang dip rol h s t lah pros s dijalankan.

35 P rangkat lunak yang dibangun b rbasis d sktop dan m nggunakan bahasa p mograman
36 java. Pada subbab ini dibahas sp sifikasi k butuhan funsional, p mod lan p rangkat lunak,
37 diagram use case, sk nario, diagram k las dari P rangkat Lunak yang dibangun.

¹ Spesifikasi Kebutuhan Fungsional Perangkat Lunak **Data Mining log Histori**
² **KIRI**

³ Spesifikasi kebutuhan perangkat lunak yang dibangun untuk melakukan *data mining log*
⁴ histori KIRI yang diharapkan adalah

- ⁵ 1. Dapat menimba dan membaca input text yang sudah disiapkan
- ⁶ 2. Dapat melakukan *preprocessing* data sesuai dengan yang dijelaskan pada bab analisis
⁷ data
- ⁸ 3. Dapat melakukan proses *data mining*, ID3 dan C4.5
- ⁹ 4. Dapat melakukan visualisasi hasil dari *data mining* yang dipilih

¹⁰ **Pemodelan Perangkat Lunak Data Mining Log Histori KIRI**

¹¹ Perangkat lunak *data mining log* histori KIRI menimba input data dalam format .csv.
¹² Setelah program mendapatkan input dan user menekan tombol proses, maka data tersebut
¹³ diubah menjadi sesuai pada bab analisis data(bab [3.1](#)) dengan melakukan proses
¹⁴ transform dan menghasilkan data dalam format splt pada tabel [3.3](#).

¹⁵ Program melakukan tahap *data mining* dengan menggunakan teknik ID3 atau C4.5
¹⁶ sesuai dengan permintaan user. Setelah proses *data mining* selesai dilakukan, program
¹⁷ melakukan visualisasi *decision tree* dengan menggunakan graphviz.

¹⁸ **Pemodelan Data pada Perangkat Lunak Data Mining Log Histori KIRI**

¹⁹ Karena data yang dipilih sudah dalam bentuk csv, maka pada pertemuan ini, tidak menggunakan sistem database .

²¹ Ketika tombol proses diklik, maka data tersebut diproses. Proses yang pertama yang
²² dilakukan adalah melakukan *load* data dari file . Data csv dibaca dengan menggunakan
²³ CSV Reader hingga semua hasil datanya sudah terpisah sesuai dengan atribut. Kemudian
²⁴ dilakukan filter data dan hanya action dengan nilai FINDROUTE yang akan diambil. Setelah
²⁵ data didapat, dilakukan proses *transform* untuk setiap baris yang ada. Proses *transform*
²⁶ tersebut memiliki tahap sebagai berikut:

- ²⁷ 1. Mengubah waktu dari UTC menjadi GMT+7 pada string data input array ketiga
²⁸ (yaitu atribut tanggal).
- ²⁹ 2. Mengambil atribut tanggal ketika nilai tersebut dengannya spasi sebagai tanda
³⁰ pembeda, maka akan terdapat tiga nilai, yaitu hari (dalam bentuk angka dimana
³¹ nilai 1 berarti senin dan nilai 7 berarti minggu), tanggal dan jam.
- ³² 3. Pada nilai tanggal, dilakukan pemisahan nilai string dengan garis miring sebagai tanda
³³ pembeda, maka dipilih tiga nilai yaitu bulan, tanggal, dan tahun, namun nilai yang
³⁴ akan diambil hanya dua, yaitu bulan dan tahun.
- ³⁵ 4. Pada nilai jam, dilakukan pemisahan nilai string dengan titik dua sebagai tanda
³⁶ pembeda, maka dipilih dua nilai yaitu jam dan minute, namun nilai yang diambil hanya
³⁷ jam.

5. M ngambil string data input array k lima (yaitu atribut *additionalData*), dilakukan p m cahan nilai string d ngan garis miring s bagai tanda p misah, maka dip rol h tiga nilai yaitu lokasi awal, lokasi tujuan, dan banyak jalur.
 6. Pada nilai lokasi awal dan lokasi tujuan, dilakukan p m cahan nilai string d ngan koma s bagai tanda p misah, maka dip rol h dua nilai untuk s tiap lokasi, yaitu *latitude* dan *longitude*.
 7. M nghitung jarak posisi lokasi awal dan lokasi tujuan t rhadap titik pusat dan m - n ntukan apakah lokasi t rs but b rada pada klasifikasi -1 atau 0 atau 1.
 8. m nggabungkan nilai-nilai t rs but k dalam satu array, yaitu array d ngan tip int (d ngan nilai bulan, tahun, hari, jam dan m nujuBandung).
- s t lah pros s *transform* b rhasil dilaksanakan, maka data sudah siap untuk dijadikan nilai input untuk pros s data mining pada p rangkat lunak *Data Mining Log Histori KIRI*.

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

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

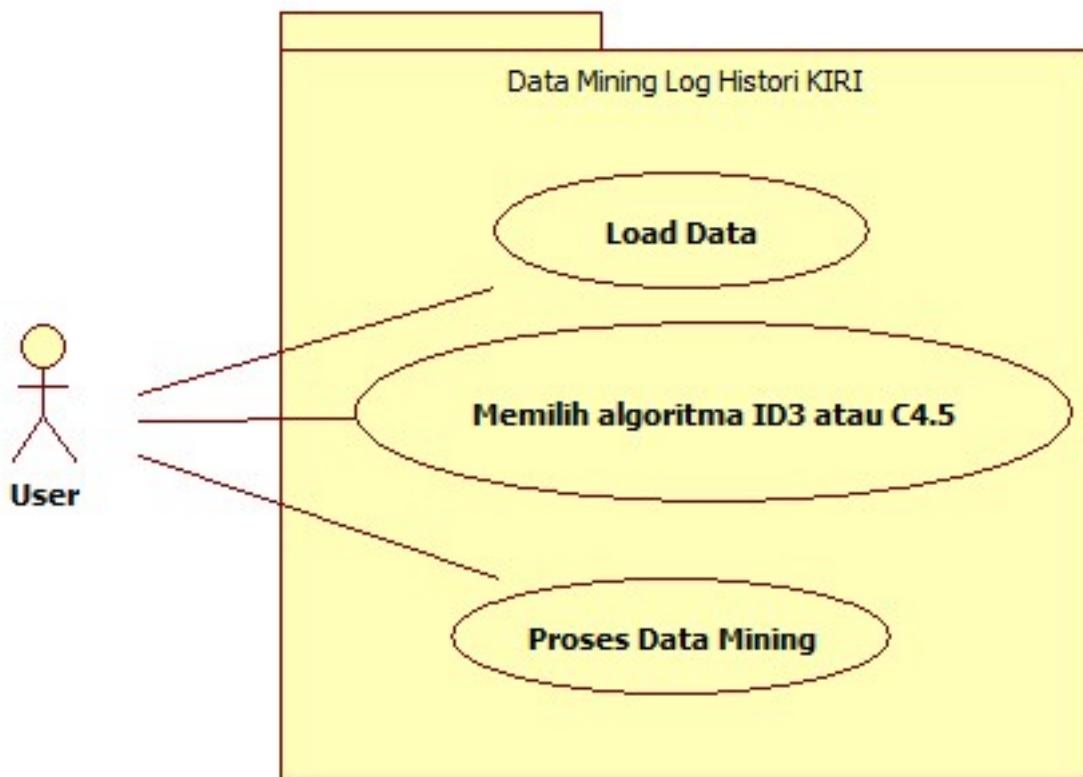
- 16 1. M muat data dan m lakukan *processing data*
- 17 2. M njalankan algoritma p mbuat *decision tree*
- 18 3. M mbuat grafik dari hasil algoritma *decision tree*
- 19 4. M nampilkan grafik *decision tree*

20 3.3.1 Diagram Use Case Perangkat Lunak *Data Mining Log Histori KIRI*

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

- 24 • M lakukan *load* data yang digunakan s bagai input data d ngan cara m masukan
25 alamat data di program
- 26 • M milih algoritma yang digunakan, t rdapat dua algoritma, yaitu ID3 dan C4.5
- 27 • M lakukan pros s *data mining* d ngan input data dari alamat data yang sudah dima-
28 sukan. S t lah pros s b rhasil dilaksanakan, program m nampilkan hasil yang dip -
29 rol h

30 Diagram use case saat *user* m njalankan p rangkat lunak *Data Mining Log Histori KIRI*
31 dapat dilihat pada gambar 3.2.



Gambar 3.2: Diagram Use Case P rangkat Lunak Data Mining Log Histori KIRI

Tab 13.5: Sk nario M lakukan load Data

Nama	Load data
Aktor	User
D skripsi	Masukan alamat data yang dijadikan sebagai input program
Kondisi awal	Textbox belum terisi
Kondisi akhir	Textbox sudah terisi dengan alamat data
Sk nario utama	User memasukan alamat data pada textbox
Eks spi	Data tidak dimakan

Tab 13.6: Sk nario M lakukan Data Mining

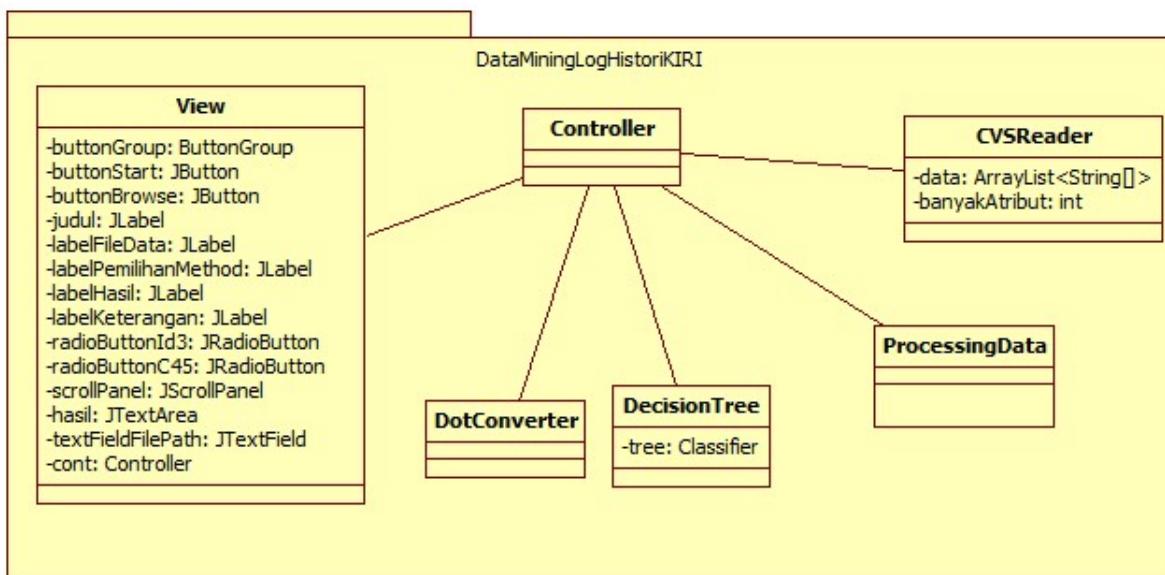
Nama	Proses Data Mining
Aktor	User
D skripsi	Mengkan tombol proses pada interface
Kondisi awal	Textbox belum terisi
Kondisi akhir	Textbox sudah terisi dengan hasil data mining
Sk nario utama	User mengkan tombol proses
Eks spi	Data tidak dimakan atau data tidak dapat diproses

Tab 1 3.7: Sk nario M milih Algoritma yang Digunakan

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

1 3.3.2 Diagram kelas Perangkat Lunak Data Mining Log Histori KIRI

2 P mbuat diagram *class* untuk m m nuhi s mua tujuan dari diagram *use case* dan sk nario
 3 t rdapat pada gambar 3.3.

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

4 Berikut d skripsi k las diagram *class*:

- 5 • Vi w, m rupakan k las untuk m ngatur d sain antar muka.
- 6 • Controll r, m rupakan k las untuk m ngatur vi w dan modul k tika program dijalankan.
- 7 • CSVR ad r, m rupakan k las yang m miliki m thod untuk m mbaca fil d ngan format CSV.
- 8 • Proc ssingData, m rupakan k las yang m miliki m thod untuk m lakukan *preprocessing data*.
- 9 • D cissionTr , m rupakan k las yang m miliki m thod untuk m mbuat *decision tree* dan m nghitung *confident* dari pohon yang sudah dihasilkan.

- 1 ● DotConv rt r, m rupakan k las yang m miliki m thod untuk m ngubah *string* yang
- 2 m rupakan hasil dari k las D cissionTr (yaitu, *decision tree* dalam b ntuk string)
- 3 m njadi bahasa DOT yang siap dijadikan *input* untuk graphviz.

1 BAB 4

2 PERANCANGAN PERANGKAT LUNAK

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

5 4.1 Perancangan Perangkat Lunak

6 4.1.1 Perancangan Kelas

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

9 • Kelas Controll merupakan kelas untuk mengatur view dan modul ketika program
10 dijalankan.

11 – Method

12 * public Controll(), merupakan konstruktor dari kelas controll.
13 * public void startMining(String inputFilePath, String miningAlgo, JLabel label,
14 JButton startButton, JButton browseButton), merupakan method untuk menjalankan modul
15 modul yang melakukan *data mining* dan membuat *decision tree* dari data
16 yang menjadi masukan program.
17 * public static void main(String[] args), merupakan method main untuk menjalankan program.

18 • Kelas Viw, merupakan kelas untuk mengatur tampilan antar muka.

19 – Atribut

20 * ButtonGroup buttonGroup, digunakan untuk mengelompokkan jRadioButton.
21 * JButton buttonStart, merupakan sebuah tombol yang dapat menggil method
22 buttonStartActionPerformed() bila diklik.
23 * JButton buttonBrowse, merupakan sebuah tombol yang dapat menggil method
24 buttonBrowseActionPerformed() bila diklik.
25 * JLabel labelJudul, merupakan sebuah label yang berisi judul dari aplikasi ini.
26 * JLabel labelFileData, merupakan label untuk menunjukkan bagian pada miliaran
27 file data path.
28 * JLabel labelMiliaran, merupakan label untuk menunjukkan bagian pada miliaran
29 pada m thod.

- * 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 tod ID3 atau tidak.
- * JRadioButton radioButtonC4.5, m rupakan *radio button* yang m nunjukkan bahwa us r m milih m tod C4.5 atau tidak.
- * JScrollPane 1 scrollPan 1, m rupakan variab 1 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 C4.5) k mudian m manggil m thod startMining d ngan masukan k dua string t rs but, lab 1 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 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 nhapus 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 m ndapatkan nilai variab l banyakAtribut.

- 1 * public void setBanyakAtribut(int banyakAtribut), digunakan untuk memberikan nilai variabel banyakAtribut sesuai dengan parameter t.
- 2
- 3 • Kelas ProcessingData, merupakan kelas yang memiliki method untuk melakukan *preprocessing data*.
- 4
- 5 – Method
- 6 * public ProcessingData(), merupakan konstruktor dari kelas ProcessingData.
- 7 * public void processSorting(ArrayList array, ArrayList data, String action), digunakan untuk memilih arraylist sehingga arraylist tersebut hanya berisi *action* yang diinginkan saja (pada pertemuan ini, *action* yang diharapkan adalah FINDROUTE). Hasil pilah disimpan pada variable array dari parameter tersebut sehingga tidak dipertukarkan lagi.
- 8
- 9 * public ArrayList processSorting(ArrayList<String[]> data), Digunakan untuk melakukan tahap *preprocessing data* seperti yang sudah dijelaskan pada pembelajaran 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 mendekati klasifikasi kelas dari untuk setiap record dengan menggunakan fungsi jarak dari titik ke bandar raya Bandung dan titik pusat Bandung.
- 10
- 11 * public int KlasifikasiKelas(double jarakKebandaran, double jarakTujuan), Digunakan untuk mendekati kelas dari hasil jarak titik ke bandar raya dengan titik pusat Bandung dan titik tujuan dengan titik pusat Bandung.
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22 • Kelas DecisionTree, merupakan kelas yang memiliki method untuk membuat *decision tree* dan menghitung akurasi dari pohon yang sudah dihasilkan.
- 23
- 24 – Atribut
- 25 * ClassifierTree, digunakan untuk menyimpan *decision tree* yang sudah dihasilkan.
- 26
- 27 – Method
- 28 * public DecisionTree(), merupakan konstruktor untuk kelas DecisionTree.
- 29 * public double calculatePrecision(Instances data), digunakan untuk mendapatkan nilai akurasi dari *decision tree* yang dihasilkan.
- 30
- 31 * public String id3(Instances data), digunakan untuk membuat *decision tree* dengan menggunakan metode ID3 dari API Weka.
- 32
- 33 * public String j48(Instances data), digunakan untuk membuat *decision tree* dengan menggunakan metode C4.5 dari API Weka.
- 34
- 35 • 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.
- 36
- 37
- 38 – Method

1 * public String conv_rt(String data, String miningAlgo, String nod_Nam),
 2 Digunakan untuk mengubah nilai string yang sudah dipilih dari klas D -
 3 cisionTree menjadi bahasa DOT untuk membuat visualisasi dengan menggunakan graphviz.
 4

5 Pada kelas Processor, nilai data waktu perlu diganti menjadi GMT+7 dan perlu
 6 menghitung jarak antar dua titik. Maka dari itu, dibuat dua kelas tambahan untuk
 7 m melakukannya pada halaman but, yaitu TimZonConv, Convrt dan DistancHaversin .

8 • Kelas TimZonConv, merupakan kelas yang memiliki method untuk mengubah
 9 waktu dari UTC menjadi GMT+7

10 – Method

11 * public static String conv_rtToGMT7(String data), digunakan untuk mengubah
 12 waktu dari UTC menjadi GMT+7.

13 • Kelas DistancHaversin , kelas yang memiliki method untuk menghitung jarak dua
 14 titik di bumi.

15 – Atribut

16 * double r, digunakan untuk menyimpan nilai radius dari bumi.

17 – Method

18 * public double calculateDistance(double latitud1, double longitud1, double
 19 latitud2, double longitud2), Digunakan untuk menghitung jarak dari dua
 20 titik (latitud dan longitud).

21 Sekolah melakukannya penerapan tentang API Weather, dipilih bahwa input untuk membuat decision tree merupakan kelas Instances dari API Weather. Selain itu, dipelajari juga penggunaan untuk hasil dari kelas tersebut, apakah sudah sesuai dengan aplikasi Weather atau belum (karena menggunakan API Weather, sehingga harusnya decision tree yang dihasilkan sama). Oleh karena itu, ditambahkan kelas ArffIO yang berfungsi untuk menulis dan membaca data dengan format arff, sehingga kita program melakukannya data mining, program menghasilkan file dengan format .arff yang dapat dibaca oleh aplikasi Weather untuk melakukannya penerapan. Karena kita sudah memiliki file .arff tersebut, ada baiknya jika menggunakan fungsi untuk membaca arff dari API Weather yang menghasilkan return value berupa kelas Instances yang dapat digunakan untuk membuat decision tree.

31 • Kelas ArffIO, merupakan kelas yang berfungsi untuk melakukan penyimpanan dan
 32 membaca data dengan format arff.

33 – Method

34 * public ArffIO, merupakan konstruktor dari kelas ArffIO.

35 * public void writeArffIO(String name, ArrayList<int[]> data), digunakan untuk
 36 menulis file .arff sesuai data pada parameter.

37 * public Instances arffRead(String name), digunakan untuk membaca file .arff
 38 dengan menggunakan method dari API Weather.

1 K tika mulai m rancang *method* conv rt yang b rada di k las DotConv rt r, lbih mu-
 2 dah jika dirancang m njadi r kursif. Kar na data yang diolah pada *method* t rs but cukup
 3 banyak dan dip rlukan nama yang b rb da pada s tiap nod yang ditulis pada DOT, maka
 4 p rlu ditambah k las yang b rfungsi untuk struktur data pada k las t rs but, yaitu SDFor-
 5 Conv rtTr .

6 • k las SDForConv rtTr , k las yang b rfungsi untuk m nyimpan data yang dibutuhk-
 7 an untuk m ngubah String hasil dari k las D cisionTr m njadi bahasa DOT.

8 – Atribut

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

13 – M thod

14 * public SDForConv rtTr (String[] data), m rupakan konstruktor untuk k -
 15 las ini dan m lakukan inisialisasi data pada atribut d ngan nilai data pada
 16 param t r s rta m lakukan inisialisasi nilai variab l count d ngan 0.
 17 * public void s tData(String data, ind x int), digunakan untuk m ngubah nilai
 18 data pada ind x t rt ntu.
 19 * public String[] g tData(), digunakan untuk m ndapatkan nilai atribut data.
 20 * public String g tData(int ind x), digunakan untuk m ndapatkan nilai data
 21 pada ind x t rt ntu.
 22 * public void s tCount(int count, int ind x), digunakan untuk m ngubah nilai
 23 count pada ind x t rt ntu.
 24 * public int g tCount(int ind x), digunakan untuk m ndapatkan nilai count
 25 pada ind x t rt ntu.
 26 * public bool an hasN xt(), digunakan untuk m ng c k apakah isi dari varib l
 27 data masih ada atau tidak.
 28 * public void buangArrayP rtama(), digunakan untuk m mbuang nilai array
 29 yang p rtama (ind x k -0).
 30 * public String g tDataNumb r(String atribut), digunakan untuk m ndapatkan
 31 angka pada nama atribut t rt ntu untuk m mbuat nama nod pada k las
 32 DotConv rt r agar s mua nama nod b rb da.

33 S t lah m lakukan convert string hasil dari *method* p mbuatan *decision tree* API W -
 34 ka k bahasa Dot, maka dip rlukan p manggilan fungsi dot yang t rdapat pada graphviz.
 35 Cara m manggilan fungsi t rs but yaitu d ngan m nggunakan *command prompt*. Ol h ka-
 36 r na itu, dip rlukan k las yang m miliki *method* untuk m manggil *command prompt* dan
 37 m njalankan fungsi dot t rs but, yaitu k las CMD.

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

39 – M thod

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

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

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

9 4.1.2 Sequence Diagram

10 Pada subbab ini, dij laskan alur program d ngan m nggunakan *sequence diagram* pada [4.2](#).

11 P rtama, program m nampilkan d sain antar muka yang dihasilkan ol h k las Vi w. K -
 12 mudian us r m nulis *file path* atau m milih (d ngan m nggunakan tombol *browse*) *input fil*
 13 pada JT xtFi ld s rta m milih m tod p mbuatan *decision tree* (tahap p rtama). S t lah
 14 m milih fil dan m tod , us r m n kan tombol start, dan k las Vi w m manggil *method*
 15 startMining dari k las controll r (tahap 3-4).

16 K las Controll r m ngaks s fil s suai d ngan masukan *file path* d ngan m manggil *me-*
 17 *thod* r adCSV dari k las CSVR ad r dan m ndapat nilai k mbalian b rupa *ArrayList* (tahap
 18 5-6). S t lah m ndapatkan data dari fil CSV yang dipilih, data t rs but dipilah dan hanya
 19 *record* d ngan *action FINDROUTE* yang diambil. P milihan dilakukan d ngan cara m -
 20 manggil *method* proc ssSorting pada k las Proc ssingData dan m ng mbalikan *ArrayList*
 21 d ngan data yang sudah dipilah (tahap 7-8). K mudian program m lakukan *preprocessing*
 22 data d ngan cara m manggil *method* pr proc ssingData dari k las Proc ssingData(tahap
 23 9).

24 K tika *method* pr proc ssingData dijalankan, nilai waktu dari UTC harus m njadi
 25 GMT+7 d ngan cara m manggil *method* conv rtGMT7 dari k las Tim zon Conv rt r dan
 26 m ng mbalikan nilai b rtip Dat (tahap 10-11). S t lah nilai waktu diubah, program m ng-
 27 hitung jarak antara dua titik d ngan cara m manggil *method* calculat Distanc dari k las
 28 Distanc Hav rsin dan m ng mbalikan nilai doubl yang b risi jarak dari k dua titik(tahap
 29 12-13). K mudian program m ngklasifikasikan k las dari jarak yang sudah dihasilkan d -
 30 ngan cara m manggil *method* klasifikasiK las dari k las Proc ssingData (tahap 14-15). Lalu
 31 s mua data yang sudah dipros s, dik mbalikan dalam b ntuk *ArrayList*(tahap 16).

32 S t lah didapat data yang sudah dilakukan *preprocessing data*, data disimpan d ngan
 33 format arff d ngan cara m manggi *method* writ Arff pada k las ArffIO(tahap 17). K mu-
 34 dian, program m ngambil data dari fil arff yang sudah disimpan untuk m ndapatkan data
 35 b rtip Instanc d ngan cara m manggil *method* adArff(tahap 18-19).

36 K mudian program m mbuat *decision tree* d ngan cara m manggil *method* id3 atau
 37 j48 pada k las D cisionTr dan m ng mbalikan *decision tree* dalam b ntuk String(tahap
 38 20-21). S t lah m mp rol h *decision tree*, p rlu dicari nilai akurasi yang dihasilkan ol h
 39 *decision tree* t rs but d ngan cara m manggil *method* calculat Pr cision dan nilai akurasi
 40 yang dihasilkan dik mbalikan dalam b ntuk doubl (tahap 22-23).

41 Tahap s lanjutnya adalah m ngubah nilai String yang dip rol h dari *method* id3 atau j48
 42 m njadi bahasa DOT d ngan cara m manggil *method* conv rt pada k las DotConv rt r dan

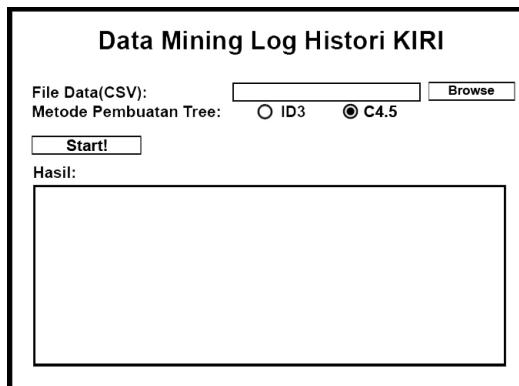
1 m ng mbalikan nilai String(tahap 24-25). S t lah dip rol h hasil dari *method convert*, maka
 2 dip rlukan *command prompt* untuk m nghasilkan gambar grafik visualisasi dari *decision tree*
 3 yang sudah dihasilkan(tahap 26-27).

4 S t lah m nghasilkan gambar visualisasi *decision tree*, *method startMining* m mbuat
 5 JFram yang baru untuk m mp rlihatkan hasil gambar *decision tree* yang sudah dihasilk-
 6 an s rta m ng mbalikan nilai String *decision tree* k pada k las Vi w yang ditampilkan di
 7 JT xtAr a(tahap 28-29).

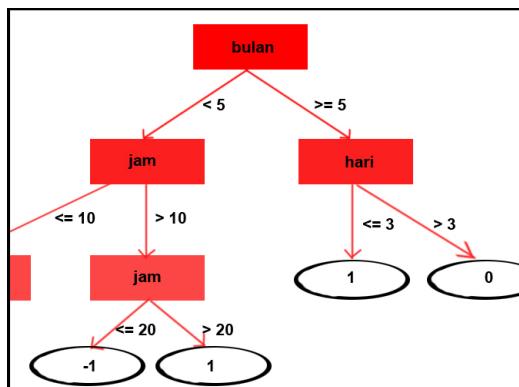
8 4.1.3 Perancangan Desain Antar Muka

9 Pada subbab ini, dip rlihatkan rancangan d sain antar muka yang digunakan untuk program
 10 ini.

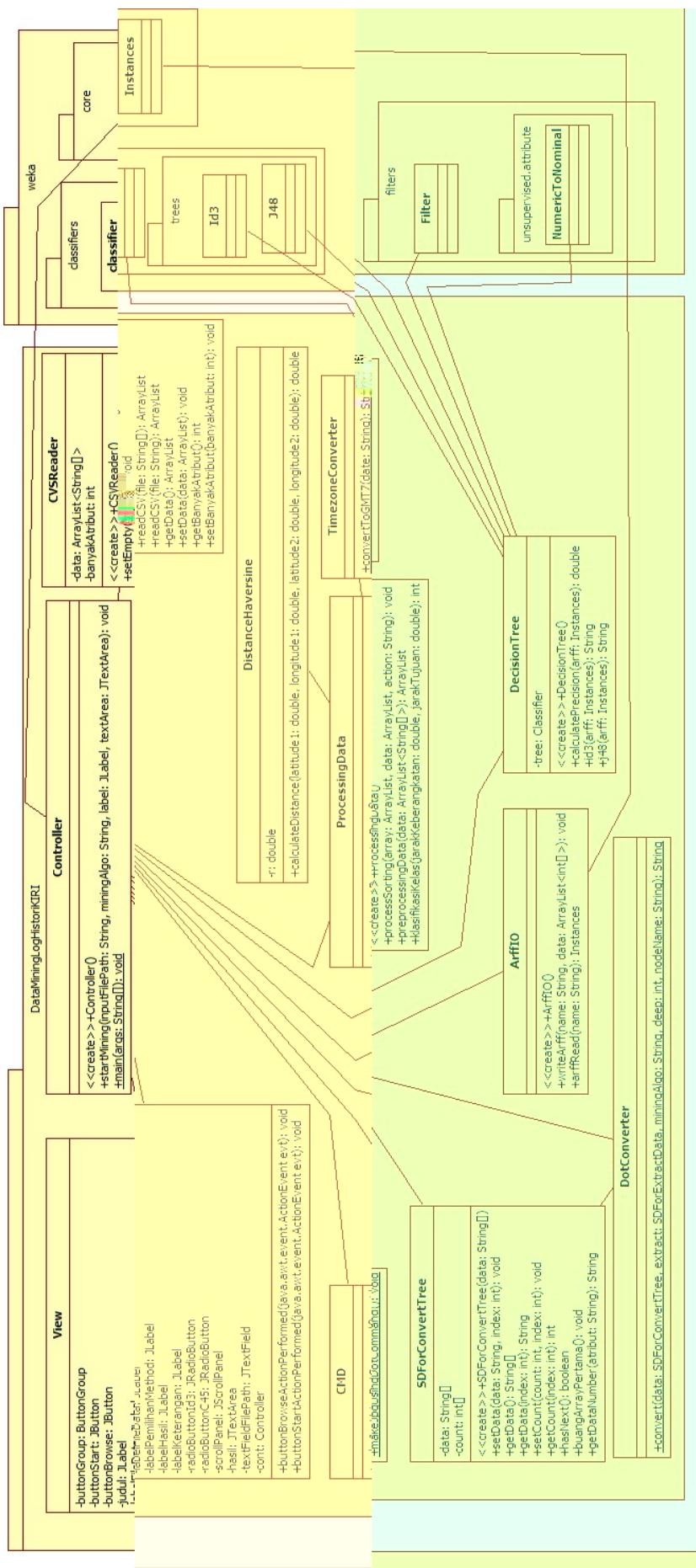
11 Aplikasi ini m miliki dua form untuk m lakukan *data mining* dan m mbuat *decision*
 12 *tree*. Pada form p rtama(dapat dilihat di 4.3) dis diakan JT xtbox dan JButton yang digu-
 13 nakan untuk m milih fil , JRadioButton yang digunakan untuk m milih m tod p mbuatan
 14 *decision tree*, JT xtAr a yang digunakan untuk m mp rlihatkan hasil *decision tree* yang
 15 dip rol h dalam b ntuk String, s rta JT xtButton yang k dua (d ngan lab 1 Start) yang
 16 digunakan untuk m mulai pros s *data mining*. S dangkan form k dua, b risi gambar visu-
 17 alisasi *decision tree* yang sudah dihasilkan(dapat dilihat di 4.4).



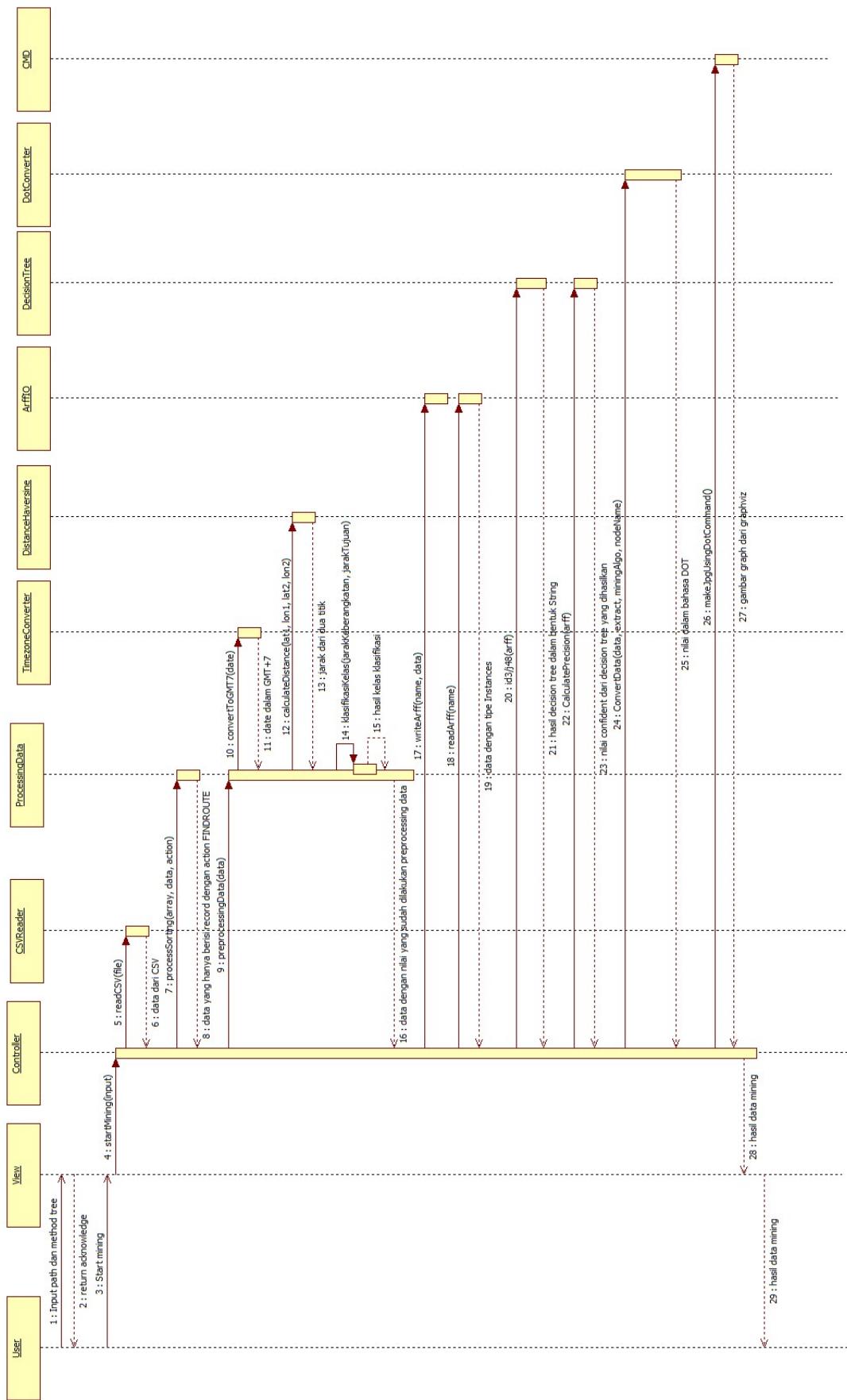
Gambar 4.3: Mock Up Form Pertama



Gambar 4.4: Mock Up Form Kedua



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



Gambar 4.2: Sequence diagram P rangkat Lunak Data Mining Log Histori KIRI

BAB 5

IMPLEMENTASI PROGRAM DAN PENGUJIAN

Bab ini menjelaskan 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 i7, 1.60 GHz
- RAM: 6 GB
- VGA: NVIDIA GeForce GT 330M
- Hardisk: 300 GB

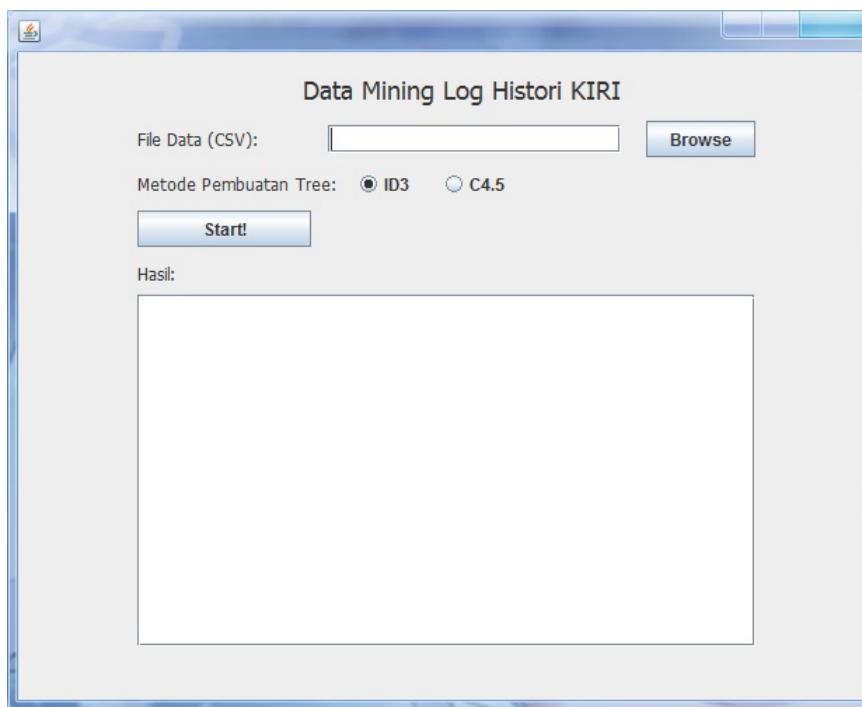
2. Sistem operasi: Windows 7 Professional

3. Platform: Network ans: IDE 8.0

5.2 Hasil Tampilan Antarmuka

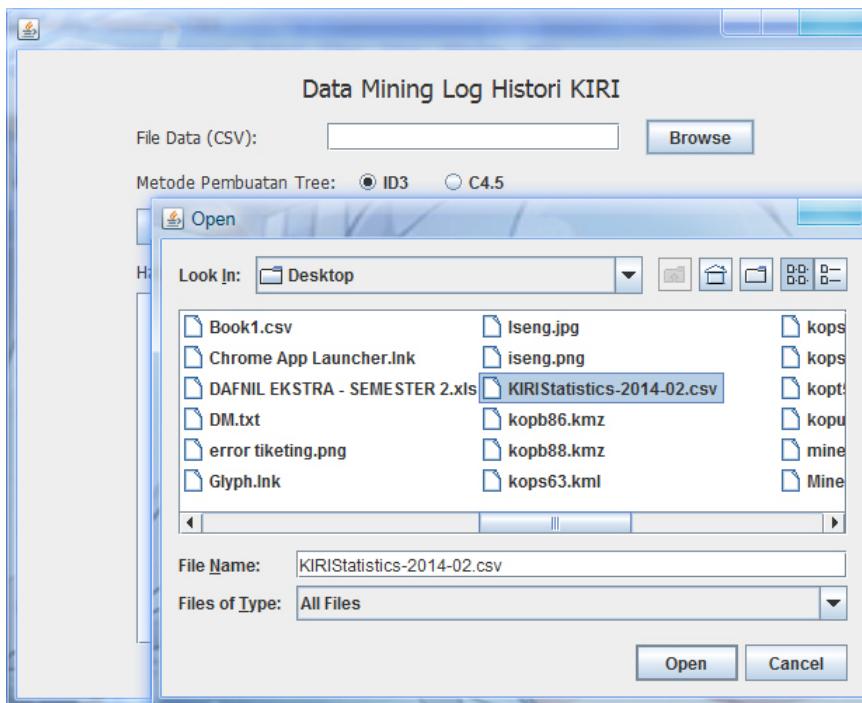
Pada aplikasi *data mining* ini, terdapat dua form. Form pertama berfungsi untuk memilih file CSV yang dilakukan *data mining*, memilih metode pembuatan *decision tree*, serta menunjukkan hasil *decision tree* yang dihasilkan dalam bentuk String. Sedangkan untuk form kedua, berfungsi untuk memvisualisasikan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

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

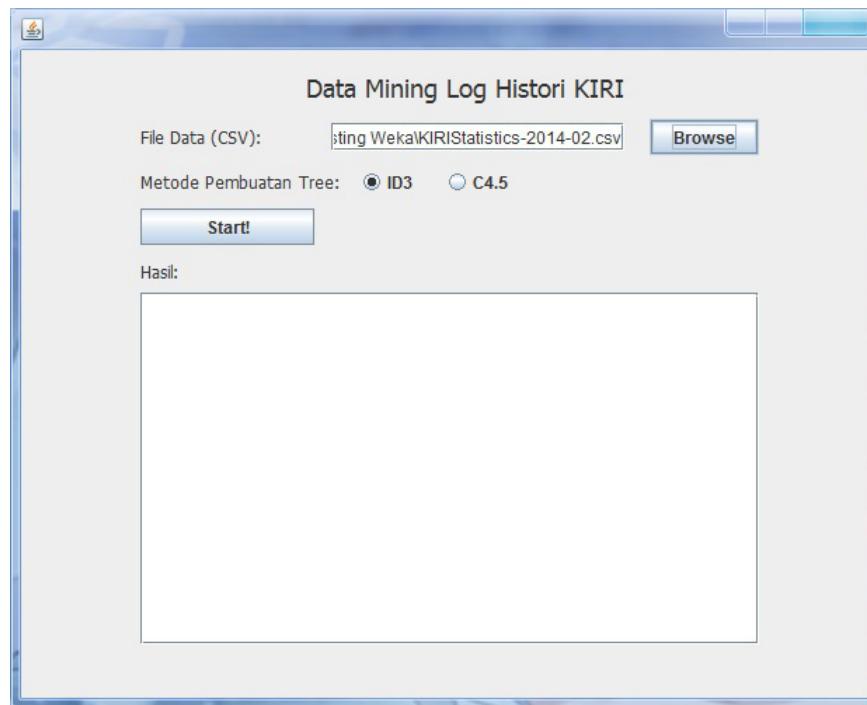


Gambar 5.1: Tampilan Form Awal Aplikasi *Data Mining*

- ¹ T erdapat dua cara untuk m engisi T extFi ld, yaitu ditulis s cara manual alamat fil e CSV
- ² atau d ngan cara m engklik tombol button brows e dan m milih fil e CSV pada Fil e S elector.
- ³ Tampilan m milih fil e CSV dapat dilihat pada gambar 5.2 dan tampilan s tlah m milih fil e CSV dapat dilihat pada gambar 5.3.

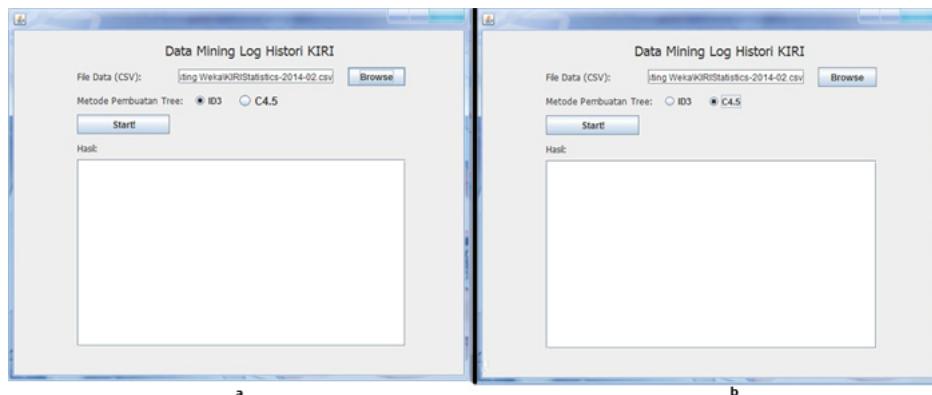


Gambar 5.2: Tampilan Fil e S elector untuk M milih Fil e CSV



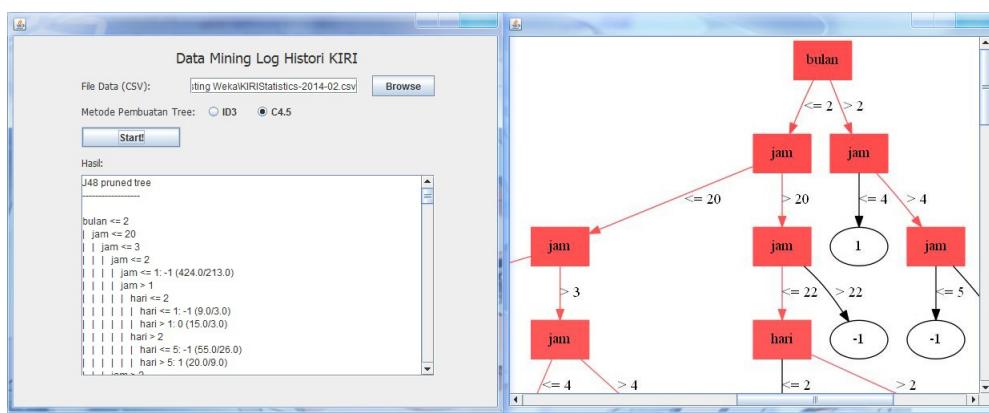
Gambar 5.3: Tampilan Form saat lahir M milah File CSV

- 1 Saya lahir alamat fil CSV diisi, halus lanjutnya yang dilakukan adalah membuat decision tree.
- 2 Pada pembuatan decision tree pada RadioButton. RadioButton milah 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 ID3 dipilih sedangkan gambar b merupakan kondisi tampilan ketika C4.5 dipilih.

- 4 Klik tombol start diklik lalu program akan melakukan proses data mining. Saya lakukan, 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 nampilkkan Hasil *Data Mining*

1 5.3 Pengujian Aplikasi *Data Mining*

2 5.3.1 Pengujian Fungsional

3 Rancangan Pengujian

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

6 1. Pengujian *method* utama untuk mencapai tujuan dari aplikasi *data mining* ini. Terdapat berbagai *method* yang perlu diuji, yaitu

8 (a) Membaca file CSV

9 (b) Makukan *preprocessing data*

10 (c) Mmbuat *decision tree*

11 (d) Mengubah *decision tree* dalam bentuk String menjadi bahasa DOT

12 2. Karena program lunak hanya dilihat pada hasil kluaran program atau kondisi masukan yang dibirikan tanpa melihat bagaimana proses untuk mendapatkan kluaran tersebut.

15 3. Dari kluaran yang dihasilkan, kemampuan program untuk memenuhi kebutuhan pemakai dapat diukur dan dapat diketahui ke salahannya jika ada.

17 Hasil Pengujian

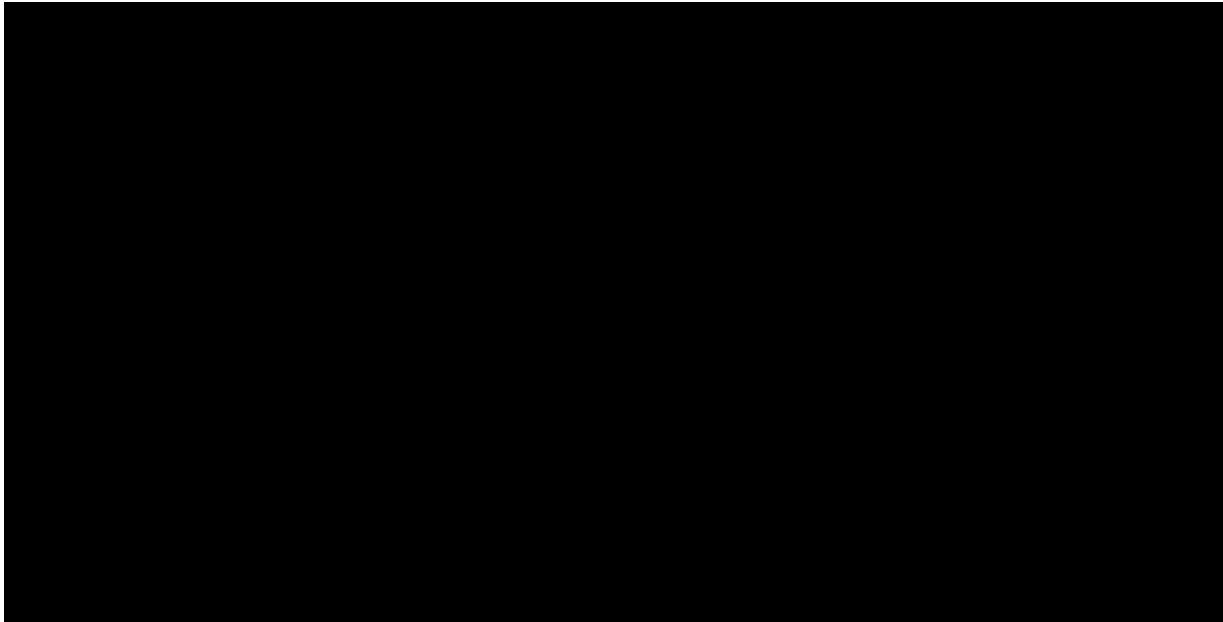
18 Dibuat sebuah *test case* yang berisi 20 data log histori KIRI dengan data pada tabel 5.1

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	-6.908639785445589,107.6109156755932/-6.90855,107.61082/1
114073	A44EB361A179A49E	2/1/2014 2:47	SEARCHPLACE	hot 1+harapan+i/10
114074	A44EB361A179A49E	2/1/2014 2:47	SEARCHPLACE	hot 1+harapan+ind/10

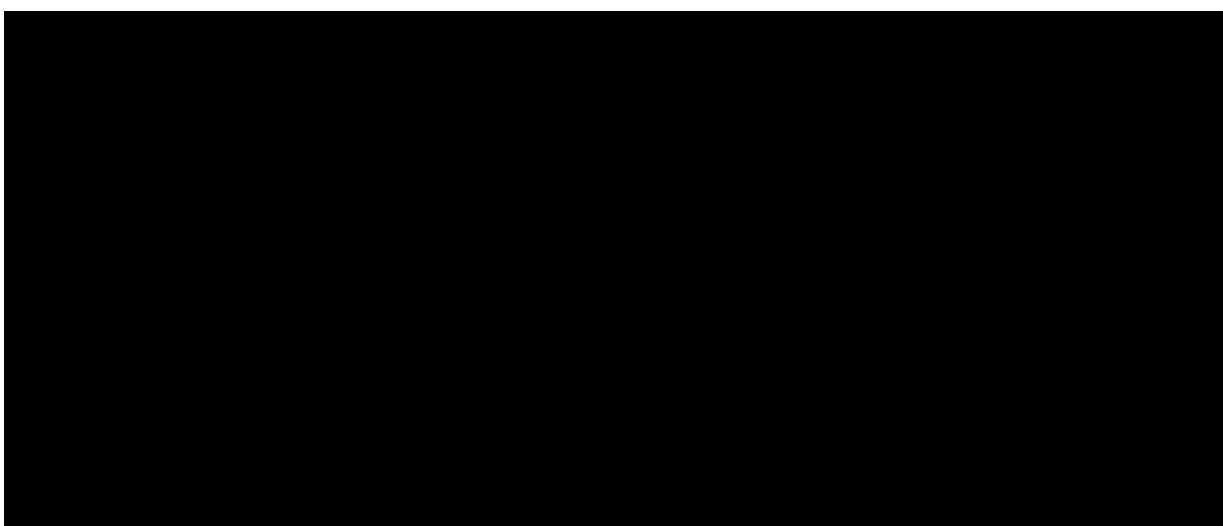
Tab 15.1: Data untuk Test Case Aplikasi Data Mining

1 B rikut hasil p ngujian yang dilakukan d ngan m nggunakan data t rs but:

- 2 1. P ngujian p rtama: M mbaca fil CSV, data diambil ol h program dan dipilah, ha-
3 nya r cord d ngan *action* FINDROUTE yang diambil s dangkan yang lain dibuang.
4 B rikut hasil d ngan contoh data diatas d ngan m nggunakan sist m *debug* untuk
5 m lihat data yang diambil dari CSV pada gambar 5.6 dan 5.7



Gambar 5.6: P ngujian M ngambil Data CSV

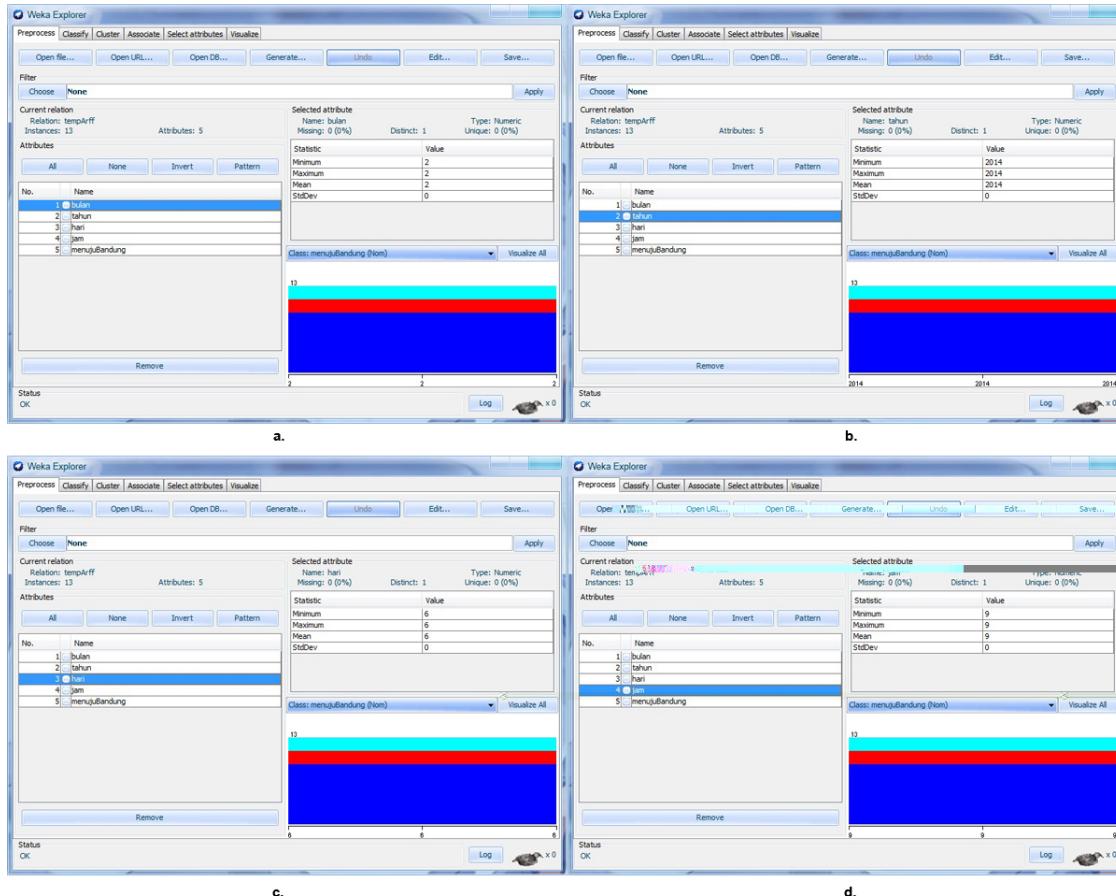


Gambar 5.7: P ngujian *Data Selection* untuk M ngambil Data d ngan *Action* FINDROUTE

6 Gambar p rtama m mp rlihatkan bahwa data p rtama yang dip rol h 21 array String
7 d ngan array p rtama adalah atributnya s dangkan array s lanjutnya adalah isi data
8 yang dip rol h. Pada gambar k dua, t rdapat 13 array String dimana s mua array
9 t rs but m rupakan r cord d ngan nilai *action* FINDROUTE saja. D ngan d mikian,
10 p ngujian m mbaca CSV b rhasil dilakukan.

- 11 2. P ngujian k dua: M lakukan *preprocessing data*, data yang digunakan adalah 13 array

String yang sudah dipilih dari pengujian pertama. Pengujian dilakukan dengan cara mengambil data menggunakan window untuk melihat data yang sudah dihasilkan. Hasil yang dipilih dari pengujian dapat dilihat pada [5.8](#) dan [5.9](#)

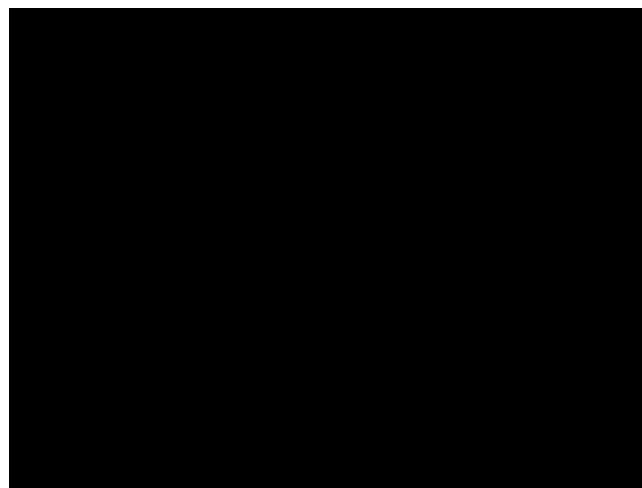


Gambar 5.8: Pengujian Preprocessing Data. Gambar a menunjukkan nilai pada atribut bulan. Gambar b menunjukkan nilai pada atribut tahun. Gambar c menunjukkan nilai pada atribut hari. Gambar d menunjukkan nilai pada atribut jam.

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 ber nilai sembilan karena pada testcasenya, nilai atribut jam adalah dua. Pada tahap klasifikasi arah, dapat dilihat pada tabel [5.2](#).

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

3. Pengujian ketiga: Membuat *decision tree*, dilakukan perbandingan antara hasil dari program dengan hasil dari window, dapat dilihat pada gambar [5.10](#) dan [5.11](#). Dari kedua gambar tersebut, dapat disimpulkan bahwa hasil *decision tree* yang dihasilkan sama dan benar.
4. Pengujian keempat: Mengubah *decision tree* dalam bentuk String menjadi bahasa DOT, melakukan visualisasi *decision tree* dengan membuat graph dalam bahasa DOT,

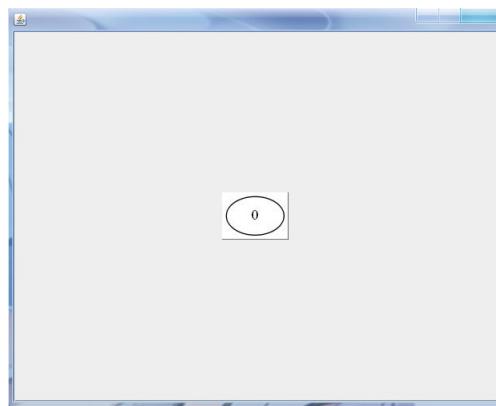


Gambar 5.9: Pengujian Preprocessing Data untuk Klasifikasi Data

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 15.2: Hasil Pengaruan ar dan Klasifikasi

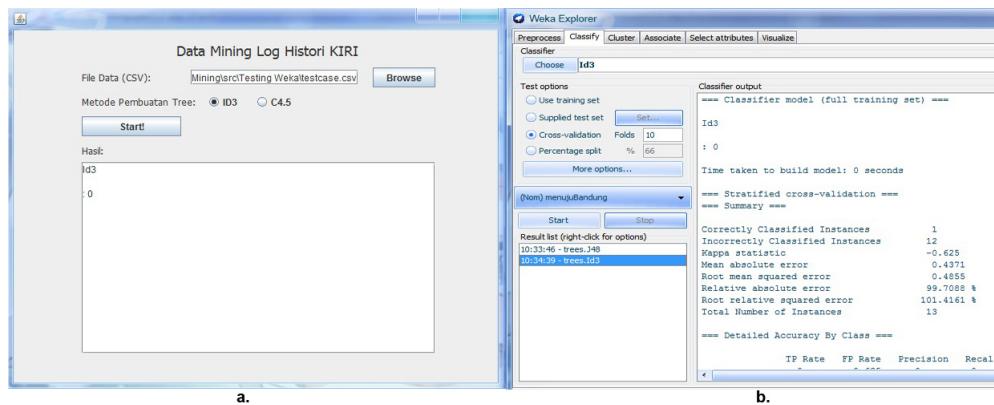
1 dapat dilihat pada gambar 5.12. Dari gambar tersebut, dapat disimpulkan bahwa
2 pengubahan *decision tree* menjadi bahasa DOT telah berhasil.



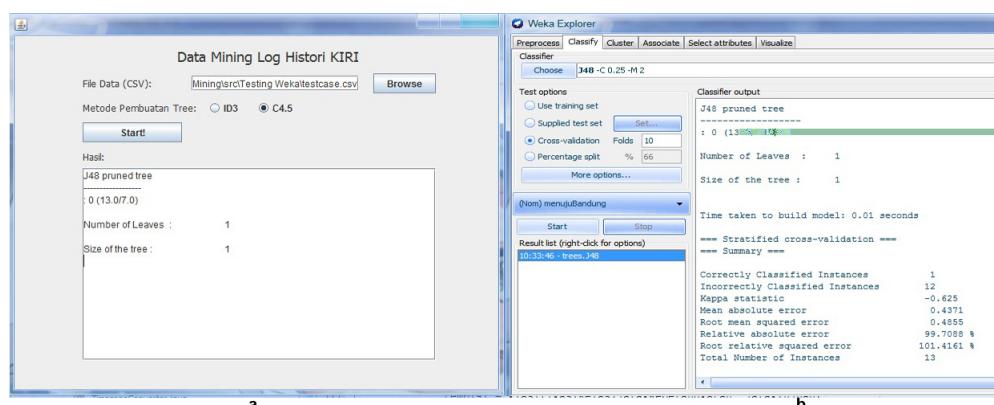
Gambar 5.12: Pengujian Hasil Visualisasi dengan Menggunakan Bahasa DOT.

3 5.3.2 Pengujian Eksperimental

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



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

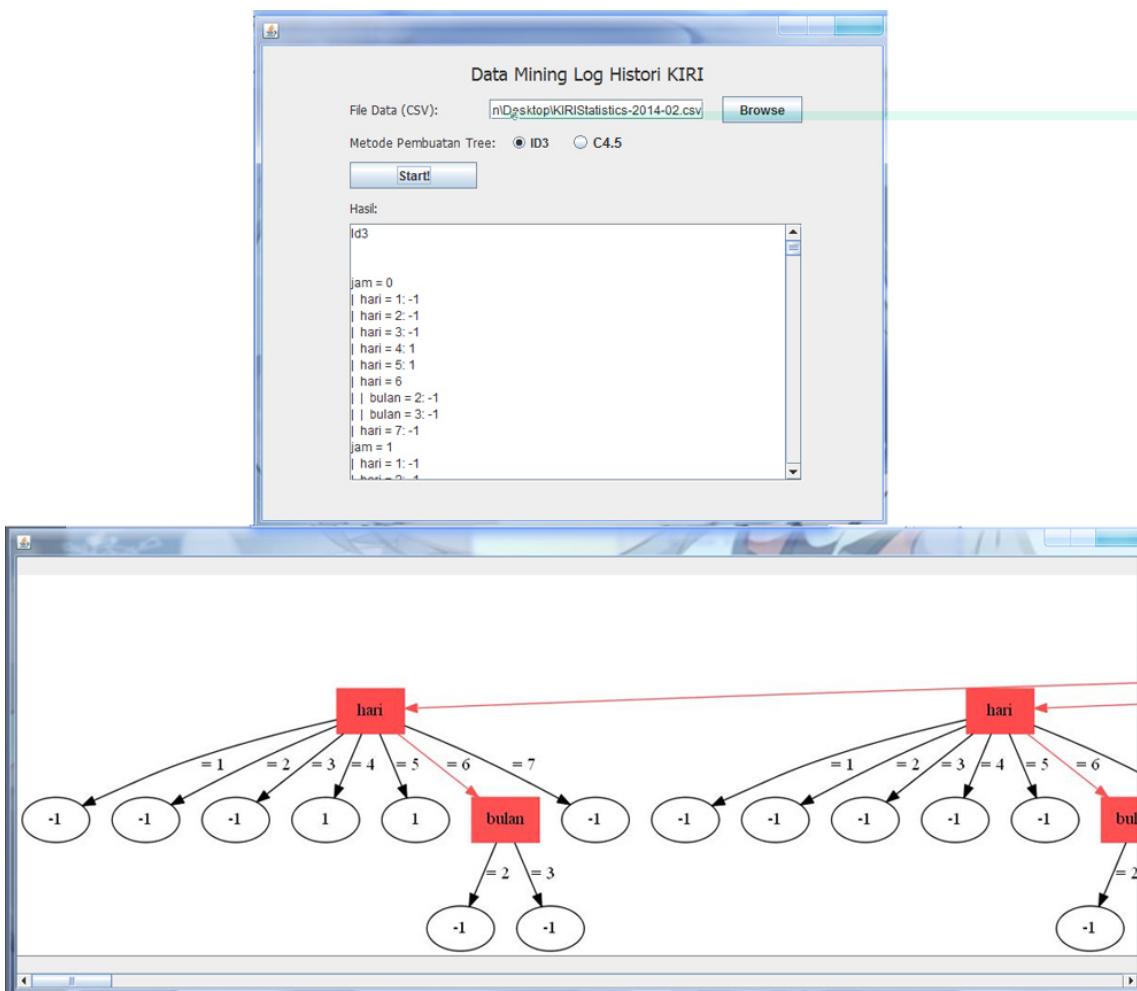


Gambar 5.11: Pengujian Pembuatan Decision Tree C4.5. Gambar a merupakan hasil dari aplikasi data mining sedangkan gambar b merupakan hasil dari weka.

- 1 Karena pengujian dilakukan, dituliskan bug data yaitu tidak mendapat nilai dari data log histori KIRI pada action FINDROUTE kolom additionalData dengan format String yang berada.
- 2 K salahnya format tersebut adalah nilai pembatas angka dibatasi koma yang seharusnya menggunakan titik menjadi koma, sehingga pemotongan nilai String menghasilkan data yang salah dan menyebabkan error berupa data tidak dapat diproses. Dari pengujian ini, maka diperlukan data cleaning pada tahap preprocessing data agar data dengan format yang salah dapat dibuang dan tidak menyebabkan error. Hasil error yang dituliskan dari data cleaning dapat dilihat pada gambar 5.13.

Gambar 5.13: Pengcobaan Data Mining untuk melakukan Data Cleaning.

- 1 Setelah melakukan *data cleaning*, program dapat berjalan dengan baik. Berikut hasil
- 2 dari pengcobaan menggunakan ID3 pada gambar 5.14 dan menggunakan C4.5 pada gambar 5.15.



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

Gambar 5.15: P rcobaan *Data Mining* d ngan M nggunakan M tod C4.5 pada Log Histori KIRI pada Bulan 2 Tahun 2014.

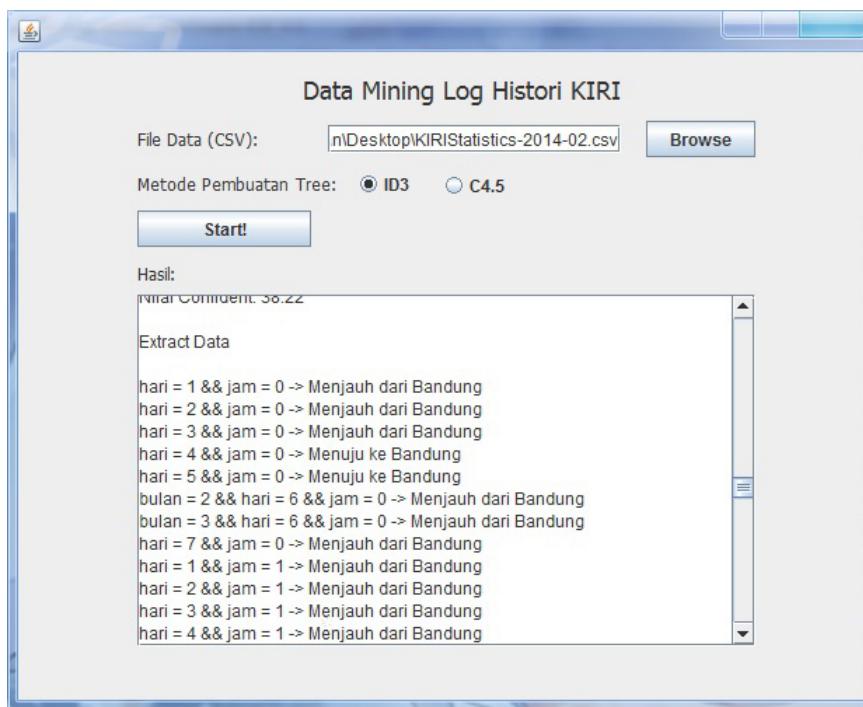
1 Pada k dua p rcobaan, t rdapat p rb daan bulan, hal ini dikar nakan p rubahan waktu
2 dari UTC m nuju GMT+7 s hingga t rdapat p rubahan bulan atau tahun. Kar na data
3 pada bulan s lanjutnya hanya tujuh jam, maka hasil pada bulan s lanjutnya l bih baik tidak
4 dianggap kar na data t rs but tidak cukup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

5 Hasil p rcobaan p mbuatan *decision tree* pada m tod ID3 m ngalami ov rfitting kar na
6 m nghasilkan klasifikasi pada hampir s tiap k mungkin. Ol h kar na itu, dapat disim-
7 pulkan bahwa *decision tree* d ngan m tod ID3 m nghasilkan tree yang j l k. Namun pada
8 p rcobaan m tod C4.5, m nghasilkan *decision tree* yang cukup b sar namun tidak ov rfif-
9 ting dan banyak user yang m njauhi Bandung. T tapi, tree yang dihasilkan cukup k cil.
10 Dapat disimpulkan hasil *decision tree* pada p rcobaan di bulan ini cukup j l k.

11 Nilai akurasi dari p rcobaan d ngan m tod ID3 adalah 38.22% s dangkan untuk p rco-
12 baan d ngan m tod C4.5 adalah 50.37%, dari sini dapat dinyatakan bahwa hasil *decision*
13 *tree* dari C4.5 l bih t pat daripada ID3 dan m tod ID3 m nghasilkan nilai akurasi yang
14 b lum m muaskan kar na masih dibawah 50%.

15 S lain itu, dari k dua p rcobaan ini, dip rol h bahwa cukup sulit untuk m mbaca hasil
16 *decision tree* t rutama s p rti hasil *decision tree* d ngan m tod ID3. Ol h kar na itu, di-
17 tambahkan fungsi agar *decision tree* l bih mudah dibaca. Program ditambah satu k las baru
18 yaitu SDForExtractData yang b rfungsi untuk m nyimpan data dan m mbuat k simpulan
19 dari s tiap leaf yang dihasilkan.

20 B rikut hasil dari fungsi yang dibuat agar *decision tree* l bih mudah dibaca, dapat dilihat
21 pada gambar 5.16. D ngan fungsi t rs but, diharapkan us r dapat l bih mudah m mbaca
22 *decision tree* yang dihasilkan.



Gambar 5.16: Hasil dari SDForExtractData

1 5.3.3 Analisis Hasil Uji

- 2 Berdasarkan pengujian di atas, dapat disimpulkan bahwa
- 3 1. Maka ID3 menghasilkan *decision tree* yang bersifat overfitting pada data *log histori KIRI* dengan *preprocessing data* dan klasifikasi yang sudah dilakukan pada bab 3.
- 5 2. Maka C4.5 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 (C4.5 menghasilkan 50.37% sebanding dengan ID3 menghasilkan 38.22%).
- 9 3. Maka ID3 belum menghasilkan nilai akurasi yang memuaskan, karena masih dibawah 50%.
- 11 4. Dari data *log histori KIRI* pada bulan Februari 2014, *decision tree* dengan metode C4.5 menyatakan bahwa user lebih sering menjauhi Bandung daripada mendekati Bandung atau berangkat menuju daerah yang sama.
- 14 5. *Decision tree* yang dihasilkan di bulan ini tidak memberikan nilai lebih yang signifikan dibandingkan statistik biasa karena *decision tree* yang dihasilkan hanya memiliki satu *leaf*.

BAB 6

KESIMPULAN DAN SARAN

6.1 Kesimpulan

K simpulan yang dapat diambil dari p n litian *data mining log* histori KIRI ini adalah:

1. Salah satu cara untuk m mp rol h pola yang m narik dan b rmakna, dip rlukan p - ngolahan data d ngan cara m mbuat klasifikasi dari data yang dihasilkan s suai d - ngan tujuan yang ingin dicari. Pada p n litian ini dilakukan klasifikasi tujuan dari user apakah m r ka ingin m nuju Bandung atau k luar Bandung atau masih b rada di ar a yang sama s hingga dip rol h p rg rakan user KIRI di Bandung.
2. P mbuatan p rangkat lunak untuk m lakukan *data mining* pada *log* histori KIRI dapat dilakukan.
3. Pola yang dip rol h dari data *log* histori KIRI adalah user s ring p rgi m njauhi Bandung pada bulan F buari 2014. Namun *decision tree* yang dihasilkan tidak m mb rikan nilai yang l bih signifikan dibanding statistik biasa.

6.2 Saran

Untuk p ng mbangan aplikasi *data mining log* histori KIRI l bih lanjut, dapat dilakukan d ngan cara m nggunakan klasifikasi yang l bih baik dan d til. P rb daannya d ngan mang- gunakan klasifikasi yang l bih d til adalah hasil dari *decision tree* mungkin l bih b sar (jika dibandingkan d ngan *decision tree* yang dihasilkan d ngan m tod C4.5) namun diharapkan dapat m nghasilkan nilai akurasi m ncapai 75 p rs n.

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.” <http://www.movable-type.co.uk/scripts/latlong.html>, 2002.
- [3] T. U. of Waikato, “W ka api.” <http://weka.sourceforge.net/doc.stable/>, 2009.
- [4] E. R. Gansn r, E. Koutsofios, and S. North, “Drawing graphs with dot.” <http://www.grphviz.org/pdf/dotguide.pdf>, January 2015.

1

LAMPIRAN A

2

100 DATA PERTAMA DARI LOG HISTORI KIRI

3

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

113923	D0AB08D956A351E4	2/1/2014 0:16	FINDROUTE	-6.90598,107.59714/-6.90855,107.61082/1
113924	D0AB08D956A351E4	2/1/2014 0:17		

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	PAGELOAD	/206.53.152.81/m
113954	E5D9904F0A8B4F99	2/1/2014 0:40	FINDROUTE	-6.90598,107.59714/-6.91728,107.60417/1
113955	E5D9904F0A8B4F99	2/1/2014 0:41	PAGELOAD	/5.10.83.30/
113956	E5D9904F0A8B4F99	2/1/2014 0:40	PAGELOAD	/5.10.83.28/
113957	E5D9904F0A8B4F99	2/1/2014 0:55	PAGELOAD	/5.10.83.99/
113958	D0AB08D956A351E4	2/1/2014 1:00	SEARCHPLACE	babd/1
113959	D0AB08D956A351E4	2/1/2014 1:00	SEARCHPLACE	babdu/1
113960	D0AB08D956A351E4	2/1/2014 1:00	FINDROUTE	-6.90598,107.59714/-6.90855,107.61082/1
113961	D0AB08D956A351E4	2/1/2014 1:09	FINDROUTE	-6.38355,106.919975/-6.85029,107.58496/1
113962	D0AB08D956A351E4	2/1/2014 1:10	FINDROUTE	-6.90598,107.59714/-6.85029,107.58496/1
113963	D0AB08D956A351E4	2/1/2014 1:10	FINDROUTE	-6.90598,107.59714/-6.90855,107.61082/1
113964	E5D9904F0A8B4F99	2/1/2014 1:10	PAGELOAD	/5.10.83.49/
113965	D0AB08D956A351E4	2/1/2014 1:12	SEARCHPLACE	t a/10
113966	A44EB361A179A49E	2/1/2014 1:15	SEARCHPLACE	taman+puptaka/10
113967	A44EB361A179A49E	2/1/2014 1:15	SEARCHPLACE	taman+puptaka+b sarn/8
113968	A44EB361A179A49E	2/1/2014 1:15	FINDROUTE	-6.9135911,107.6272095/-6.90179,107.62301/1
113969	E5D9904F0A8B4F99	2/1/2014 1:16	PAGELOAD	/36.72.98.13/
113970	E5D9904F0A8B4F99	2/1/2014 1:17	PAGELOAD	/120.173.21.110/m

113971	E5D9904F0A8B4F99	2/1/2014 1:17	SEARCHPLACE	jalan+abdrrahman+sal h/10
113972	E5D9904F0A8B4F99	2/1/2014 1:17	FINDROUTE	-6.90872,107.62253/-6.90774,107.60908/1
113973	A44EB361A179A49E	2/1/2014 1:19	FINDROUTE	-6.9158359,107.6101751/-6.90691,107.62259/1
113974	E5D9904F0A8B4F99	2/1/2014 1:19	FINDROUTE	-6.91335,107.64827/-6.86198,107.59193/1
113975	E5D9904F0A8B4F99	2/1/2014 1:25	PAGELOAD	/114.79.13.124/
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+junction/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.cndkialad.rshipschool.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

```

3 package DataMiningLogHistoriKIRI;
4
5 import java.awt.image.BufferedImage;
6 import java.io.BufferedReader;
7 import java.io.File;
8 import java.io.FileWriter;
9 import java.io.IOException;
10 import java.io.PrintWriter;
11 import java.util.ArrayList;
12 import javax.imageio.ImageIO;
13 import javax.swing.ImageIcon;
14 import javax.swing.JFrame;
15 import javax.swing.JLabel;
16 import javax.swing.JScrollPane;
17 import javax.swing.JTextArea;
18 import weka.core.Instances;
19
20 /**
21 * 
22 * @author Jovan Gunawan
23 */
24 public class Controller
25 {
26     public Controller()
27     {
28     }
29
30     public void startMining(String inputFilePath, String miningAlgo, JLabel label, JTextArea
31         textArea) throws IOException
32     {
33         CSVReader csv = new CSVReader();
34         ArrayList<String[]> data = csv.readCSV(inputFilePath);
35
36         ArrayList<String[]> findRoute = new ArrayList<String[]>();
37         ProcessingData pd = new ProcessingData();
38         pd.processSorting(findRoute, data, "FINDROUTE");
39
40         //int maxMin digunakan untuk menyimpan nilai max dan min dari variable bulan dan tahun.
41         //Untuk ketentuan posisi array dapat dilihat di method preprocessing data
42         int [] maxMin = new int [4];
43         ArrayList<int[]> dataAfterPreprocessing = pd.preprocessingData(findRoute, maxMin);
44
45         ArffIO io = new ArffIO();
46         io.writeArff("tempArff", dataAfterPreprocessing);
47
48         Instances arff = io.readArff("temp_arff");
49         //arff.setClassIndex(arff.numAttributes() - 1);
50         DecisionTree dt = new DecisionTree();
51         String [] tempTreeDataResult;
52         System.out.println(miningAlgo);
53         if(miningAlgo.equals("id3"))
54         {
55             textArea.setText(dt.id3(arff));
56         }
57         else
58         {
59             textArea.setText(dt.j48(arff));
60         }
61         tempTreeDataResult = textArea.getText().split("\n");
62         textArea.setText(textArea.getText() + "\nNilai Akurasi:" + dt.calculatePrecision(arff) +
63             "\n");
64         String [] treeDataResult;
65         System.out.println(tempTreeDataResult.length);
66         if(tempTreeDataResult.length < 8)
67         {
68             if(miningAlgo.equals("id3"))
69             {
70                 treeDataResult = new String [tempTreeDataResult.length - 2];
71             }
72             else
73             {
74                 treeDataResult = new String [tempTreeDataResult.length - 6];
75             }
76             System.arraycopy(tempTreeDataResult, 2, treeDataResult, 0, treeDataResult.length);
77         }
78     }

```

```

1      {
2          if(miningAlgo.equals("id3"))
3          {
4              treeDataResult = new String [tempTreeDataResult.length -3];
5          }
6          else
7          {
8              treeDataResult = new String [tempTreeDataResult.length -7];
9          }
10         System.arraycopy(tempTreeDataResult, 3, treeDataResult, 0, treeDataResult.length);
11     }
12     System.out.println(treeDataResult[0]);
13     SDForConvertTree dataTree = new SDForConvertTree(treeDataResult);
14
15     try {
16         PrintWriter out = new PrintWriter(new BufferedWriter(new FileWriter("tree.txt")));
17         SDForExtractData extract = new SDForExtractData(new String[]{"bulan", "tahun", "hari",
18             "jam"},new int[]{maxMin[0],maxMin[1],7,24}, new int[]{maxMin[2],maxMin[3],1,0});
19         out.println("digraph{" + DotConverter.convert(dataTree, extract, miningAlgo, 0, "") +
20             "}");
21         out.close();
22
23         textArea.setText(textArea.getText());
24         ArrayList<String> extractData = extract.getList();
25
26         if(extractData.size() > 0)
27         {
28             textArea.setText(textArea.getText() + "\nExtract>Data\n");
29         }
30         for(int i = 0; i < extractData.size(); i++)
31         {
32             textArea.setText(textArea.getText() + "\n" + extractData.get(i));
33         }
34
35     } catch (IOException ex) {
36         System.out.println("Error ketika menulis file txt");
37     }
38
39     Cmd.makeJpgUsingDotCommand();
40
41     JFrame jf2 = new JFrame();
42
43     jf2.setVisible(true);
44     jf2.setSize(620, 500);
45     BufferedImage image = null;
46     image = ImageIO.read(new File("tree.jpg"));
47     ImageIcon image2 = new ImageIcon(image);
48     JLabel labels = new JLabel(image2);
49     JScrollPane pane = new JScrollPane(labels);
50     jf2.getContentPane().add(pane);
51 }
52
53
54     public static void main (String [] args)
55     {
56         Controller cont = new Controller();
57
58         JFrame jf = new JFrame();
59         View v = new View(cont);
60
61         jf.setVisible(true);
62         jf.setSize(620, 500);
63         jf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
64
65         jf.add(v);
66     }
67 }
```

Listing B.2: Vi w.java

```

68 package DataMiningLogHistoriKIRI;
69
70 import java.io.File;
71 import java.io.IOException;
72 import java.util.logging.Level;
73 import java.util.logging.Logger;
74 import javax.swing.JFileChooser;
75
76 /**
77 * @author Jovan Gunawan
78 */
80 public class View extends javax.swing.JPanel {
81
82     /**
83      * Creates new form View
84      */
85     public View(Controller cont) {
86         this.cont = cont;
87         initComponents();
88         buttonGroup1.add(radioButtonId3);
89         buttonGroup1.add(radioButtonC45);
90     }
91
92     /**
93      * This method is called from
94      */
95 }
```



```

1         .addComponent(scrollPanel, javax.swing.GroupLayout.DEFAULT_SIZE, 441, Short.MAX_VALUE)
2         .addComponent(labelKeterangan, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
3             .addGap(0, 0, Short.MAX_VALUE))))))
4     );
5     layout.setVerticalGroup(
6         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
7             .addGroup(layout.createSequentialGroup()
8                 .addContainerGap()
9                 .addComponent(judul, javax.swing.GroupLayout.PREFERRED_SIZE, 31, javax.swing.GroupLayout.PREFERRED_SIZE)
10                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
11                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
12                    .addComponent(labelFileData, javax.swing.GroupLayout.PREFERRED_SIZE, 26, javax.swing.GroupLayout.PREFERRED_SIZE)
13                    .addComponent(textFieldFilePath, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
14                    .addComponent(buttonBrowse)))
15                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
16                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
17                    .addComponent(labelPemilihanMethod, javax.swing.GroupLayout.PREFERRED_SIZE, 26, javax.swing.GroupLayout.PREFERRED_SIZE)
18                    .addComponent(buttonC45)
19                    .addComponent(buttonId3)))
20                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
21                .addComponent(buttonStart)
22                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
23                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
24                    .addComponent(labelHasil, javax.swing.GroupLayout.PREFERRED_SIZE, 26, javax.swing.GroupLayout.PREFERRED_SIZE)
25                    .addComponent(buttonStart)
26                    .addComponent(buttonC45)
27                    .addComponent(buttonId3)))
28                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
29                .addComponent(buttonBrowse)
30                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
31                .addComponent(labelKeterangan, javax.swing.GroupLayout.PREFERRED_SIZE, 26, javax.swing.GroupLayout.PREFERRED_SIZE)
32            )
33        );
34    );
35    } // </editor-fold>
36
37 private void buttonBrowseActionPerformed(java.awt.event.ActionEvent evt) {
38     final JFileChooser fc = new JFileChooser();
39     int returnVal = fc.showOpenDialog(buttonStart);
40
41     if (returnVal == JFileChooser.APPROVE_OPTION) {
42         File file = fc.getSelectedFile();
43         textFieldFilePath.setText(file.getPath());
44     } else {
45         System.out.println("Error in capture the path");
46         System.exit(1);
47     }
48 }
49
50 private void buttonStartActionPerformed(java.awt.event.ActionEvent evt) {
51     String inputPath = textFieldFilePath.getText();
52     String miningAlgo = "";
53     if (radioButtonId3.isSelected())
54     {
55         miningAlgo = "id3";
56     }
57     else
58     {
59         miningAlgo = "c45";
60     }
61
62     try {
63         cont.startMining(inputPath, miningAlgo, labelKeterangan, hasil);
64     } catch (IOException e)
65     {
66         System.out.println("Error start mining");
67         System.exit(1);
68     }
69 }
70
71 private void radioButtonC45ActionPerformed(java.awt.event.ActionEvent evt) {
72     // TODO add your handling code here:
73 }
74
75
76 // Variables declaration - do not modify
77 private javax.swing.JButton buttonBrowse;
78 private javax.swing.ButtonGroup buttonGroup1;
79 private javax.swing.JButton buttonStart;
80 private javax.swing.JTextArea hasil;
81 private javax.swing.JLabel judul;
82 private javax.swing.JLabel labelFileData;
83 private javax.swing.JLabel labelHasil;
84 private javax.swing.JLabel labelKeterangan;
85 private javax.swing.JLabel labelPemilihanMethod;
86 private javax.swing.JRadioButton radioButtonId3;
87 private javax.swing.JRadioButton radioButtonC45;
88 private javax.swing.JScrollPane scrollPanel;
89 private javax.swing.JTextField textFieldFilePath;
90 // End of variables declaration
91 private Controller cont;
92 }
93
94
95
96
97
98 }

```

Listing B.3: CSVR ad r.java

99 | package DataMiningLogHistoriKIRI;

```

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

```

```

1             getData() . add( tempData ) ;
2             i++ ;
3         }
4     }
5     for( int j = 0; j < i; j++ )
6     {
7         String [] temp2 = ( String [] ) getData() . get( j );
8     }
9     br . close();
10    } catch( IOException e )
11    {
12        System . out . println( " Failed to read data " );
13    }
14    return getData();
15}
16
17 public ArrayList getData()
18{
19    return data;
20}
21 public void setData( ArrayList data )
22{
23    this . data = data;
24}
25 public int getBanyakAtribut()
26{
27    return banyakAtribut;
28}
29 public void setBanyakAtribut( int banyakAtribut )
30{
31    this . banyakAtribut = banyakAtribut;
32}
33}

```

Listing B.4: ProcessingData.java

```

34 package DataMiningLogHistoriKIRI;
35
36 import java.util.ArrayList;
37
38 /**
39 * @author Jovan Gunawan
40 */
41 public class ProcessingData
42 {
43     ProcessingData()
44     {
45     }
46     public void processSorting( ArrayList array , ArrayList data , String action )
47     {
48         for( int i = 0; i < data . size () ; i++ )
49         {
50             String [] tempData = ( String [] ) data . get( i );
51             if( tempData [ 3 ]. equals( action ) )
52             {
53                 array . add( tempData );
54             }
55         }
56     }
57     public ArrayList preprocessingData( ArrayList<String[]> data , int [] maxMin )
58     {
59         // 2 int[] untuk mendapatkan nilai max dan min dari variable bulan dan tahun yang digunakan
60         // untuk inisialisasi max min pada kelas SDForExtractData.
61         // array pertama untuk bulan, dan array kedua untuk tahun
62         int [] max = new int [ 2 ];
63         int [] min = new int [ 2 ];
64         max [ 0 ] = 0 ;
65         max [ 1 ] = 0 ;
66         min [ 0 ] = Integer . MAX_VALUE;
67         min [ 1 ] = Integer . MAX_VALUE;
68
69         ArrayList<int[]> result = new ArrayList<int[]>();
70
71         // tahap pertama: ubah waktu dari UTC ke GMT+7
72         for( int i = 0; i < data . size () ; i++ )
73         {
74             //System.out.println( data . get( i ) [ 3 ] );
75             data . get( i ) [ 2 ] = TimezoneConverter.convertToGMT7( data . get( i ) [ 2 ] );
76         }
77
78         // tahap kedua sampai kesembilan
79         DistanceHaversine haversine = new DistanceHaversine();
80         for( int i = 0; i < data . size () ; i++ )
81         {
82             //cek apakah format sudah benar atau belum
83             if( data . get( i ) [ 4 ]. split( "," ). length == 3 )
84             {
85                 // tahap kedua: pecah string atribut tanggal
86                 int [] temp = new int [ 5 ];
87                 String [] splitted = data . get( i ) [ 2 ]. split( " " );
88                 temp [ 2 ] = Integer . parseInt( splitted [ 0 ] );
89                 // tahap ketiga: pecah nilai string yang tanggal
90                 String [] splitted2 = splitted [ 1 ]. split( "/" );
91                 temp [ 0 ] = Integer . parseInt( splitted2 [ 0 ] );
92                 temp [ 1 ] = Integer . parseInt( splitted2 [ 2 ] );
93                 // tahap keempat: pecah nilai string yang jam
94                 splitted2 = splitted [ 2 ]. split( ":" );
95                 temp [ 3 ] = Integer . parseInt( splitted2 [ 0 ] );
96                 // tahap kelima: pecah string atribut additional data
97                 splitted = data . get( i ) [ 4 ]. split( "/" );
98                 // tahap keenam: pecah lokasi keberangkatan dan lokasi tujuan untuk mendapatkan
99                 // lat n lon dan (ini tahap ketujuh) menghitung jarak terhadap titik pusat
100            }
100        }
101    }
102}

```

```

1 |         splitted2 = splitted[0].split(",");
2 |         double jarakKeberangkatan = haversine.calculateDistance(Double.parseDouble(
3 |             splitted2[0]), Double.parseDouble(splitted2[1]), -6.916667,107.6) * 1000;
4 |         splitted2 = splitted[1].split(",");
5 |         double jarakTujuan = haversine.calculateDistance(Double.parseDouble(splitted2[0]),
6 |             Double.parseDouble(splitted2[1]), -6.916667,107.6) * 1000;
7 |         // tahap kedelapan , set semua data ke array
8 |         temp[4] = klasifikasiKelas(jarakKeberangkatan , jarakTujuan);
9 |
10|         if(temp[4] != -2)
11|         {
12|             result.add(temp);
13|             //proses untuk mencatat nilai max dan min
14|             if(max[0] < temp[0])
15|             {
16|                 max[0] = temp[0];
17|             }
18|             if(min[0] > temp[0])
19|             {
20|                 min[0] = temp[0];
21|             }
22|             if(max[1] < temp[1])
23|             {
24|                 max[1] = temp[1];
25|             }
26|             if(min[1] > temp[1])
27|             {
28|                 min[1] = temp[1];
29|             }
30|         }
31|     }
32|     else
33|     {
34|         System.out.println("ERROR: additional data tidak sesuai;" + data.get(i)[4]);
35|     }
36| }
37| maxMin[0] = max[0];
38| maxMin[1] = max[1];
39| maxMin[2] = min[0];
40| maxMin[3] = min[1];
41| return result;
42|
43|
44| public int klasifikasiKelas(double jarakKeberangkatan , double jarakTujuan)
45| {
46|     int regionKeberangkatan , regionTujuan;
47|     int klasifikasi = 0; // 0 --> tidak menuju Bandung , 1 --> menuju Bandung , 2 --> menuju
48|     region yang sama
49|
50|     regionKeberangkatan = (int)jarakKeberangkatan;
51|     regionTujuan = (int)jarakTujuan;
52|
53|     if(regionKeberangkatan >= 11 && regionTujuan >= 11)
54|     {
55|         return -2;
56|     }
57|     else
58|     {
59|         if(regionKeberangkatan >= 11)
60|         {
61|             regionKeberangkatan = 11;
62|         }
63|         else if(regionTujuan >= 11)
64|         {
65|             regionTujuan = 11;
66|         }
67|         if( regionKeberangkatan > regionTujuan )
68|         {
69|             klasifikasi = -1;
70|         }
71|         else if( regionKeberangkatan < regionTujuan )
72|         {
73|             klasifikasi = 1;
74|         }
75|         else
76|         {
77|             klasifikasi = 0;
78|         }
79|     }
80| }
81| }
82| }

```

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

```

83| package DataMiningLogHistoriKIRI;
84|
85| import java.text.ParseException;
86| import java.text.SimpleDateFormat;
87| import java.util.Date;
88| import java.util.TimeZone;
89| import java.util.logging.Level;
90| import java.util.logging.Logger;
91|
92| /**
93|  * 
94|  * @author Jovan Gunawan
95|  */
96| public class TimezoneConverter
97| {
98|     // Mengubah input waktu menjadi waktu GMT+7
99|     // Jika ingin mengubah dari UTC ke GMT+7 maka komputer harus set timezone ke UTC

```

```

1 // input parameter string harus dalam format dd-MM-yyyy HH:mm:ss --> contoh 1/1/2014 3:51:15
2 // AM
3 // output akan mengubah waktu menjadi GMT+7 dalam String dengan format EEE MM/dd/yyyy HH:mm:ss
4 // --> contoh Wed 01/01/2014 03:51:15
5 public static String convertToGMT7(String date)
6 {
7     Date d = null;
8     long milliseconds = 0;
9
10    try {
11        SimpleDateFormat f = new SimpleDateFormat("MM/dd/yyyy HH:mm");
12        d = (Date) f.parse(date);
13    } catch (ParseException ex) {
14        SimpleDateFormat f = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
15        try {
16            d = (Date) f.parse(date);
17        } catch (ParseException ex1) {
18            Logger.getLogger(TimezoneConverter.class.getName()).log(Level.SEVERE, null, ex);
19            System.exit(1);
20        }
21    }
22    milliseconds = d.getTime();
23
24    Date currentTime = new Date(milliseconds);
25    // untuk hari --> 1 = senin, ..., 7 = minggu
26    SimpleDateFormat sdf = new SimpleDateFormat("u,MM/dd/yyyy HH:mm:ss");
27
28    sdf.setTimeZone(TimeZone.getTimeZone("GMT+7"));
29    return sdf.format(currentTime);
30}
31

```

Listing B.6: DistanceHaversine.java

```

32 package DataMiningLogHistoriKIRI;
33
34 /**
35 *
36 * @author Jovan Gunawan
37 */
38 public class DistanceHaversine
39 {
40     private double r;
41     public DistanceHaversine()
42     {
43         r = 6.371;
44     }
45
46     public double calculateDistance(double latitude1, double longitude1, double latitude2, double
47                                     longitude2)
48     {
49         latitude1 = Math.toRadians(latitude1);
50         longitude1 = Math.toRadians(longitude1);
51         latitude2 = Math.toRadians(latitude2);
52         longitude2 = Math.toRadians(longitude2);
53
54         double dlon = longitude2 - longitude1;
55         double dlat = latitude2 - latitude1;
56
57         double a = Math.pow((Math.sin(dlat/2)),2) + Math.cos(latitude1) * Math.cos(latitude2) *
58             Math.pow(Math.sin(dlon/2),2);
59
60         double c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
61         return r * c;
62     }
63 }

```

Listing B.7: ArffIO.java

```

64 package DataMiningLogHistoriKIRI;
65
66 import java.io.BufferedReader;
67 import java.io.BufferedWriter;
68 import java.io.FileReader;
69 import java.io.FileWriter;
70 import java.io.IOException;
71 import java.io.PrintWriter;
72 import java.util.ArrayList;
73 import java.util.logging.Level;
74 import java.util.logging.Logger;
75 import weka.core.Instances;
76
77 /**
78 *
79 * @author Jovan Gunawan
80 */
81 public class ArffIO
82 {
83     public ArffIO()
84     {
85     }
86
87     // method writer ini dibuat KHUSUS untuk skripsi data log KIRI
88     // input berupa arraylist dengan objek int[]
89     // Setiap array int, terdapat 7 nilai yaitu tanggal, bulan, tahun, hari, jam, menit,
90     // menujuBandung
91     // disini nilai hari akan diubah menjadi string, 1 akan menjadi senin, ..., dan 7 akan menjadi
92     // minggu
93     public void writeArff(String name, ArrayList<int[]> data)
94     {
95         String result = "@relation " + name + "\n\n@attribute_bulan_REAL\n@attribute_tahun_REAL\n"

```

```

1 |         + "@attribute _hari_REAL"
2 |         + "\n@attribute _jam_REAL\n@attribute _menujuBandung{-1,0,1}\n\n@data";
3 |
4 |     for(int i = 0; i < data.size(); i++)
5 |     {
6 |         int[] temp = data.get(i);
7 |         result += "\n" + temp[0] + "," + temp[1] + "," + temp[2] + "," + temp[3] + "," + temp
8 |                     [4];
9 |     }
10 |
11 |    try {
12 |        PrintWriter out = new PrintWriter(new BufferedWriter(new FileWriter("temp.arff")));
13 |        out.println(result);
14 |        out.close();
15 |    } catch (IOException ex) {
16 |        System.out.println("Error ketika menulis file arff");
17 |    }
18 |
19 |
20 |    public Instances readArff(String name) throws IOException
21 |    {
22 |        Instances data;
23 |        data = new Instances(new BufferedReader(new FileReader("temp.arff")));
24 |        data.setClassIndex(data.numAttributes() - 1);
25 |        return data;
26 |    }
27 |

```

Listing B.8: DcisionTr.java

```

28 package DataMiningLogHistoriKIRI;
29
30 import java.util.logging.Level;
31 import java.util.logging.Logger;
32 import weka.classifiers.Classifier;
33 import weka.classifiers.trees.Id3;
34 import weka.classifiers.trees.J48;
35 import weka.core.Instances;
36 import weka.filters.Filter;
37 import weka.filters.unsupervised.attribute.NumericToNominal;
38
39 /**
40 * 
41 * @author Jovan Gunawan
42 */
43 public class DecisionTree
44 {
45     private Classifier tree;
46
47     public double calculatePrecision(Instances arff)
48     {
49         int nilaiBenar = 0, resultInt;
50         float result = 0;
51         for (int i = 0; i < arff.numInstances(); i++)
52         {
53             try {
54                 result = (float) tree.classifyInstance(arff.instance(i));
55                 resultInt = Math.round(result)-1;
56                 if(resultInt == Integer.parseInt(arff.instance(i).stringValue(4)))
57                 {
58                     nilaiBenar++;
59                 }
60             } catch (Exception ex) {
61             }
62         }
63         double confident = Math.round(nilaiBenar * 1.0 / arff.numInstances() * 10000)/100.0;
64         return confident;
65     }
66
67     public String id3(Instances arff)
68     {
69         tree = new Id3();
70         try {
71             NumericToNominal convert= new NumericToNominal();
72             String[] options= new String[2];
73             options[0]= "-R";
74             options[1]= "1-4";
75
76             convert.setOptions(options);
77             convert.setInputFormat(arff);
78
79             Instances newData=Filter.useFilter(arff, convert);
80
81             tree.buildClassifier(newData);
82         } catch (Exception ex) {
83             Logger.getLogger(Controller.class.getName()).log(Level.SEVERE, null, ex);
84         }
85
86         return tree.toString();
87     }
88
89     public String j48(Instances arff)
90     {
91         tree = new J48();
92         try {
93             tree.buildClassifier(arff);
94         } catch (Exception ex) {
95             Logger.getLogger(Controller.class.getName()).log(Level.SEVERE, null, ex);
96         }
97
98         return tree.toString();
99     }

```

1 | }

Listing B.9: DotConv rt r.java

```

2 package DataMiningLogHistoriKIRI;
3
4 import DataMiningLogHistoriKIRI.*;
5
6 /**
7 *
8 * @author Jovan Gunawan
9 */
10 public class DotConverter
11 {
12     // converter dot khusus untuk skripsi data mining log histori KIRI --> output berupa tree
13     // dalam string dari weka
14     public static String convert(SDFForConvertTree data, SDFForExtractData extract, String
15         miningAlgo, int deep, String nodeName)
16     {
17         String result = "";
18         String [] temp;
19         String nodeName1="", nodeName2="";
20         //color 1 selalu 1.0 karena merah
21         double color;
22         if(deep == 0 && data.getData().length == 1 && data.getData()[0].charAt(0) == ':')
23         {
24             result = data.getData()[0].split("\u03c9")[1];
25         }
26         else
27         {
28             if(miningAlgo.equals("id3"))
29             {
30                 boolean hasNext = true;
31                 int loop = 0;
32                 while(hasNext)
33                 {
34                     hasNext = false;
35
36                     temp = data.getData(0).split("\u03c9\u03c9");
37                     boolean iniName1 = false;
38
39                     System.out.println("Deep:\u03c9" + deep + data.getData(0));
40                     temp = temp[deep].split("\u03c9");
41
42                     color = 0.7 - deep * 0.02;
43                     if(color < 0.3)
44                     {
45                         color = 0.3;
46                     }
47
48                     if(nodeName.equals(""))
49                     {
50                         if(loop == 0)
51                         {
52                             nodeName1 = data.getDataNumber(temp[0]);
53                         }
54                         result += nodeName1 + "\u03c9->\u03c9";
55                         iniName1 = true;
56                     }
57                     else
58                     {
59                         result += nodeName + "\u03c9->\u03c9";
60                     }
61
62                     SDFForExtractData tempExtract = new SDFForExtractData(extract);
63
64                     if(temp[2].charAt(temp[2].length()-1) == ':')
65                     {
66                         // masukin data buat extract
67                         tempExtract.setRules(temp[0], temp[1], Integer.parseInt(temp[2].substring(
68                             0, temp[2].length()-1)));
69                         try
70                         {
71                             tempExtract.setKelas(Integer.parseInt(temp[3]));
72                         }catch(Exception a)
73                         {
74                             if(temp[3] == null)
75                             {
76                                 tempExtract.setKelas(-2);
77                             }
78                         }
79                         tempExtract.extract();
80                         extract.addStringResult(tempExtract.getList());
81                         // menghasilkan daun
82
83                         nodeName2 = data.getDataNumber(temp[3]);
84                         result += nodeName2 + "\u03c9[label=\u03c9" + temp[1] + "\u03c9" + temp[2].substring(0,
85                             temp[2].length()-1) + "\u03c9]\n";
86                         if(iniName1 && loop == 0)
87                         {
88                             result += nodeName1 + "\u03c9[label=\u03c9" + temp[0] + "\u03c9,shape=box,style=
89                             filled,color=\u03c91.0\u03c9" + color + "\u03c91.0\u03c9]\n";
90                         }
91                         result += nodeName2 + "\u03c9[label=\u03c9" + temp[3] + "\u03c9]\n";
92                         data.buangArrayPertama();
93                     }
94                     else
95                     {
96                         // masukin data bwt extract
97                         tempExtract.setRules(temp[0], temp[1], Integer.parseInt(temp[2]));
98
99                         // menghasilkan node

```

```

1 |     String [] temp2;
2 |     temp2 = data.getData(1).split("~~");
3 |     temp2 = temp2[deep+1].split("~~");
4 |     nodeName2 = data.getDataNumber(temp2[0]);
5 |     result += nodeName2 + "[label=\"" + temp[1] + "~~" + temp[2] + "\"]" ,shape=
6 |         box,style=filled ,color="\u0010\u0000" + color + "\u0010\u0000"]\n";
7 |     data.buangArrayPertama();
8 |
9 |     SDForExtractData newExtract = new SDForExtractData(tempExtract);
10 |    result += DotConverter.convert(data, newExtract, miningAlgo, deep+1,
11 |        nodeName2);
12 |
13 |    if(iniName1 && loop == 0)
14 |    {
15 |        result += nodeName1 + "[label=\"" + temp[0] + "\"]" ,shape=box ,style=
16 |            filled ,color="\u0010\u0000" + color + "\u0010\u0000"]\n";
17 |    }
18 |    result += nodeName2 + "[label=\"" + temp2[0] + "\"]" ,shape=box ,style=filled
19 |        ,color="\u0010\u0000" + color + "\u0010\u0000"]\n";
20 |    extract.addStringResult(newExtract.getList());
21 |
22 |    if(data.hasNext())
23 |    {
24 |        if(data.getData(0).split("~~").length-1 == deep)
25 |        {
26 |            hasNext = true;
27 |        }
28 |    }
29 |    loop++;
30 |
31 |}
32 |else
33 |
34 |{
35 |    for(int i = 0; i < 2; i++)
36 |    {
37 |        temp = data.getData(0).split("~~");
38 |        boolean iniName1 = false;
39 |
40 |        System.out.println("Deep:" + deep + data.getData(0));
41 |        temp = temp[deep].split("~~");
42 |
43 |        color = 0.7 - deep * 0.02;
44 |        if(color < 0.3)
45 |        {
46 |            color = 0.3;
47 |        }
48 |
49 |        if(nodeName.equals(""))
50 |        {
51 |            if(i == 0)
52 |            {
53 |                nodeName1 = data.getDataNumber(temp[0]);
54 |            }
55 |            result += nodeName1 + "->-";
56 |            iniName1 = true;
57 |        }
58 |        else
59 |        {
60 |            result += nodeName + "->-";
61 |        }
62 |
63 |        SDForExtractData tempExtract = new SDForExtractData(extract);
64 |
65 |        if(temp[2].charAt(temp[2].length()-1) == ':')
66 |        {
67 |            // masukin data bwt extract
68 |            tempExtract.setRules(temp[0], temp[1], Integer.parseInt(temp[2].substring
69 |                (0, temp[2].length()-1)));
70 |            tempExtract.setKelas(Integer.parseInt(temp[3]));
71 |            tempExtract.extract();
72 |            extract.addStringResult(tempExtract.getList());
73 |
74 |            nodeName2 = data.getDataNumber(temp[3]);
75 |            result += nodeName2 + "[label=\"" + temp[1] + "~~" + temp[2].substring(0,
76 |                temp[2].length()-1) + "\"]\n";
77 |            if(iniName1 && i == 0)
78 |            {
79 |                result += nodeName1 + "[label=\"" + temp[0] + "\"]" ,shape=box ,style=
80 |                    filled ,color="\u0010\u0000" + color + "\u0010\u0000"]\n";
81 |            }
82 |            result += nodeName2 + "[label=\"" + temp[3] + "\"]\n";
83 |            data.buangArrayPertama();
84 |
85 |        }
86 |        else
87 |        {
88 |            // masukin data bwt extract
89 |            tempExtract.setRules(temp[0], temp[1], Integer.parseInt(temp[2]));
90 |            // menghasilkan node
91 |            String [] temp2;
92 |            temp2 = data.getData(1).split("~~");
93 |            temp2 = temp2[deep+1].split("~~");
94 |            nodeName2 = data.getDataNumber(temp2[0]);
95 |            result += nodeName2 + "[label=\"" + temp[1] + "~~" + temp[2] + "\"]" ,shape=
96 |                box,style=filled ,color="\u0010\u0000" + color + "\u0010\u0000"]\n";
97 |            data.buangArrayPertama();
98 |
99 |            SDForExtractData newExtract = new SDForExtractData(tempExtract);
100 |           result += DotConverter.convert(data, newExtract, miningAlgo, deep+1,
101 |               nodeName2);
102 |
103 |            if(iniName1 && i == 0)
104 |            {
105 |            }
106 |        }
107 |    }
108 |}
109 |
110 |}

```

```

1             result += nodeName1 + "[label=\"" + temp[0] + "\",shape=box,style=
2                     filled ,color=\"1.0\\"" + color + "\"]\n";
3         }
4         result += nodeName2 + "[label=\"" + temp2[0] + "\",shape=box,style=filled
5                     ,color=\"1.0\\"" + color + "\"]\n";
6         extract.addStringResult(newExtract.getList());
7     }
8 }
9 }
10 return result;
11 }
12 }
13 }

```

Listing B.10: SDForConv rtTr .java

```

14 package DataMiningLogHistoriKIRI;
15
16 /**
17 *
18 * @author Jovan Gunawan
19 * Class ini dibuat untuk struktur data yang digunakan untuk mengubah hasil output dari weka
20 * menjadi gambar dengan bahasa DOT
21 * Class ini khusus untuk skripsi data mining log histori KIRI
22 *
23 * count akan digunakan untuk inisialisasi nomor data --> contoh tanggal1, tanggal 2, .... etc,
24 * angka tersebut yang akan ditaruh pada array tersebut
25 * array count akan digunakan sebagai berikut --> [0] = tanggal, [1] = bulan, [2] = tahun, [3] =
26 * hari, [4] = jam, [5] = menit, [6] = nilai 0, [7] = nilai 1 , [8] = nilai 2
27 */
28 public class SDForConvertTree
29 {
30     private String[] data;
31     private int [] count;
32
33     public SDForConvertTree( String [] data)
34     {
35         this.data = data;
36         count = new int[9];
37         for(int i = 0; i < 9; i++)
38         {
39             count[i] = 0;
40         }
41     }
42
43     public void setData( String data, int index)
44     {
45         this.data[index] = data;
46     }
47     public String[] getData()
48     {
49         return data;
50     }
51     public String getData( int index)
52     {
53         return this.data[index];
54     }
55     public void setCount( int count, int index)
56     {
57         this.count[index] = count;
58     }
59     public int getCount( int index)
60     {
61         int temp = this.count[index];
62         this.count[index]++;
63         return temp;
64     }
65     public boolean hasNext()
66     {
67         if(this.data.length > 0)
68         {
69             return true;
70         }
71         else
72         {
73             return false;
74         }
75     }
76
77     public void buangArrayPertama()
78     {
79         String [] temp = new String [data.length -1];
80         System.arraycopy(data, 1, temp, 0, data.length -1);
81         data = temp;
82     }
83     public String getDataNumber( String atribut)
84     {
85         String result = atribut;
86         if(atribut.equals("tanggal"))
87         {
88             result += count[0];
89             count[0]++;
90         }
91         else if(atribut.equals("bulan"))
92         {
93             result += count[1];
94             count[1]++;
95         }
96         else if(atribut.equals("tahun"))
97         {
98             result += count[2];
99             count[2]++;
99         }

```

```

1      }
2      else if( atribut . equals ( " hari " ) )
3      {
4          result += count [3];
5          count [3]++;
6      }
7      else if( atribut . equals ( " jam " ) )
8      {
9          result += count [4];
10         count [4]++;
11     }
12     else if( atribut . equals ( " menit " ) )
13     {
14         result += count [5];
15         count [5]++;
16     }
17     else if( atribut . equals ( "-1" ) )
18     {
19         result += count [6];
20         count [6]++;
21     }
22     else if( atribut . equals ( "1" ) )
23     {
24         result += count [7];
25         count [7]++;
26     }
27     else if( atribut . equals ( "0" ) )
28     {
29         result += count [8];
30         count [8]++;
31     }
32 }
33 return result;
34 }
}

```

Listing B.11: SDForExtractData.java

```

35 package DataMiningLogHistoriKIRI;
36
37 import java.util.ArrayList;
38
39 /**
40 *
41 * @author Jovan Gunawan
42 */
43 public class SDForExtractData
44 {
45     private String[] atribut;
46     private boolean[] check;
47     private int[] batasAtas;
48     private int[] batasBawah;
49     private int[] maxBatasAtas;
50     private int[] minBatasBawah;
51     private int kelas;
52     private ArrayList<String> list;
53
54     public SDForExtractData( String[] atribut , int[] max , int[] min )
55     {
56         int length = atribut.length;
57         this . atribut = new String [length];
58         this . maxBatasAtas = new int [length];
59         this . minBatasBawah = new int [length];
60         this . batasAtas = new int [length];
61         this . batasBawah = new int [length];
62         this . check = new boolean [length];
63         list = new ArrayList<String>();
64
65         for( int i = 0; i < length; i++ )
66         {
67             this . atribut [i] = atribut [i];
68             this . maxBatasAtas [i] = max [i];
69             this . batasAtas [i] = max [i];
70             this . minBatasBawah [i] = min [i];
71             this . batasBawah [i] = min [i];
72             check [i] = false;
73         }
74     }
75
76     public SDForExtractData( SDForExtractData data )
77     {
78         String[] atribut = data . getAtribut ();
79         int[] max = data . getMaxBatasAtas ();
80         int[] min = data . getMinBatasBawah ();
81         int[] bmax = data . getBatasAtas ();
82         int[] bmin = data . getBatasBawah ();
83         boolean[] check = data . getCheck ();
84         int length = data . getAtribut () . length;
85         list = data . getList ();
86
87         this . atribut = new String [length];
88         this . maxBatasAtas = new int [length];
89         this . minBatasBawah = new int [length];
90         this . batasAtas = new int [length];
91         this . batasBawah = new int [length];
92         this . check = new boolean [length];
93
94         for( int i = 0; i < length; i++ )
95         {
96             this . atribut [i] = atribut [i];
97             this . maxBatasAtas [i] = max [i];
98             this . batasAtas [i] = bmax [i];
99             this . minBatasBawah [i] = min [i];
}

```

```

1         this.batasBawah[i] = bmin[i];
2     }
3 }
4
5 public void setKelas(int kelas)
6 {
7     this.kelas = kelas;
8 }
9
10 public void setRules(String atribut, String rulesOperation, int value)
11 {
12     int index = 0;
13     for(int i = 0; i < this.atribut.length; i++)
14     {
15         if(this.atribut[i].equals(atribut))
16         {
17             index = i;
18             break;
19         }
20     }
21     check[index] = true;
22
23     if(rulesOperation.equals("<="))
24     {
25         batasAtas[index] = value;
26     }
27     else if(rulesOperation.equals("<"))
28     {
29         batasAtas[index] = value - 1;
30     }
31     else if(rulesOperation.equals(">="))
32     {
33         batasBawah[index] = value;
34     }
35     else if(rulesOperation.equals(">"))
36     {
37         batasBawah[index] = value + 1;
38     }
39     else if(rulesOperation.equals("!="))
40     {
41         batasAtas[index] = value;
42         batasBawah[index] = value;
43     }
44 }
45
46 public void extract()
47 {
48     String[] hari = new String[]{"Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu", "Minggu"};
49     boolean in = false;
50     String result = "";
51     for(int i = 0; i < atribut.length; i++)
52     {
53         if(check[i])
54         {
55             if(in)
56             {
57                 result += "&&";
58             }
59             if(batasBawah[i] == batasAtas[i])
60             {
61                 result += atribut[i] + "= " + batasAtas[i];
62             }
63             else
64             {
65                 result += atribut[i] + ": " + batasBawah[i] + " - " + batasAtas[i];
66             }
67             in = true;
68         }
69     }
70 }
71
72 if(kelas == 1)
73 {
74     result += " -> Menuju ke Bandung";
75 }
76 else if(kelas == 0)
77 {
78     result += " -> Menuju daerah yang sama";
79 }
80 else
81 {
82     result += " -> Menjauh dari Bandung";
83 }
84 list.add(result);
85 }
86
87 public void addStringResult(ArrayList<String> result)
88 {
89     list = result;
90 }
91
92 public String[] getAtribut() {
93     return atribut;
94 }
95
96 public boolean[] getCheck() {
97     return check;
98 }
99
100 public int[] getBatasAtas() {
101     return batasAtas;
102 }
103 }
```

```

1  public int[] getBatasBawah() {
2      return batasBawah;
3  }
4
5  public int[] getMaxBatasAtas() {
6      return maxBatasAtas;
7  }
8
9  public int[] getMinBatasBawah() {
10     return minBatasBawah;
11 }
12
13 public int getKelas() {
14     return kelas;
15 }
16
17 public ArrayList<String> getList() {
18     return list;
19 }
20
21 }
22 }
```

Listing B.12: CMD.java

```

23 package DataMiningLogHistoriKIRI;
24
25 import java.io.BufferedReader;
26 import java.io.IOException;
27 import java.io.InputStreamReader;
28
29 /**
30 * 
31 * @author Jovan Gunawan
32 */
33 public class Cmd
34 {
35     public static void makeJpgUsingDotCommand()
36     {
37         try {
38             final String dir = System.getProperty("user.dir");
39
40             ProcessBuilder builder = new ProcessBuilder(
41                 "cmd.exe", "/c", "cd graphviz&&cd bin&&dot\" + dir + "\\tree.txt\" -o \" + dir
42                 + "\\tree.jpg\" -Tjpg");
43             builder.redirectErrorStream(true);
44             Process p = builder.start();
45             BufferedReader r = new BufferedReader(new InputStreamReader(p.getInputStream()));
46             String line;
47             while (true) {
48                 line = r.readLine();
49                 if (line == null)
50                 {
51                     break;
52                 }
53                 System.out.println(line);
54             }
55         } catch (IOException e) {
56             System.out.println("Error ketika proses CMD");
57             System.exit(1);
58         }
59     }
60 }
```