

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 Waktu	21
2.5 Graphviz	40
3 ANALISA	43
3.1 Analisis Data	43
3.1.1 Data Cleaning	43
3.1.2 Data Integration	43
3.1.3 Data Selection	43
3.1.4 Data Transformation	44
3.2 Analisis Pengaruh Lunak	48
3.2.1 Diagram Use Case Pengaruh Lunak Data Mining Log Histori KIRI	50
3.2.2 Diagram Kelas Pengaruh Lunak Data Mining Log Histori KIRI	52
4 PERANCANGAN PERANGKAT LUNAK	55
4.1 Perancangan Pengaruh Lunak	55
4.1.1 Perancangan Klasifikasi	55
4.1.2 Skewness Diagram	62
4.1.3 Perancangan Dua Sain Antar Muka	64

5 IMPLEMENTASI PROGRAM DAN PENGUJIAN	65
5.1 Lingkungan Pengembangan	65
5.2 Hasil Tampilan Antarmuka	65
5.3 Pengujian Aplikasi <i>Data Mining</i>	68
5.3.1 Pengujian Fungsional	68
5.3.2 Pengujian Ekspresional	74
5.3.3 Analisis Hasil Uji	76
6 KESIMPULAN DAN SARAN	79
6.1 Kesimpulan	79
6.2 Saran	79
DAFTAR REFERENSI	81
A 100 DATA PERTAMA DARI <i>log HISTORI KIRI</i>	83
B THE SOURCE CODE	89

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	41
3.1	<i>Classification</i> pada da rah Bandung	47
3.2	Diagram Use Case P rangkat Lunak <i>Data Mining Log Histori KIRI</i>	51
3.3	Diagram <i>Class</i> P rangkat Lunak <i>Data Mining Log Histori KIRI</i>	52
4.1	Diagram <i>Class</i> P rangkat Lunak <i>Data Mining Log Histori KIRI</i>	61
4.2	Diagram <i>Class</i> P rangkat Lunak <i>Data Mining Log Histori KIRI</i>	63
4.3	Mock Up Form P rtama	64
4.4	Mock Up Form K dua	64
5.1	Tampilan Form Awal Aplikasi <i>Data Mining</i>	66
5.2	Tampilan Fil S 1 ctor untuk M milih Fil CSV	66
5.3	Tampilan Form s t lah M milih Fil CSV	67
5.4	Tampilan Form P milihan M tod P mbuatan <i>Decision Tree</i>	67
5.5	Tampilan Form M nampulkan Hasil <i>Data Mining</i>	68
5.6	P ngujian M ngambil Data CSV	70
5.7	P ngujian <i>Data Selection</i> untuk M ngambil Data d ngan Action FINDROUTE	70
5.8	P ngujian <i>Preprocessing Data</i>	71
5.9	P ngujian <i>Preprocessing Data</i> untuk Klasifikasi Data	72
5.10	P ngujian P mbuatan <i>Decision Tree ID3</i>	73
5.11	P ngujian P mbuatan <i>Decision Tree C4.5</i>	73
5.12	P ngujian Hasil Visualisasi d ngan M nggunakan Bahasa DOT	73
5.13	P rcobaan <i>Data Mining</i> untuk M lakukan Data Cl aning	74
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	75
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	75
5.16	Hasil dari SDForExtractData	76

DAFTAR TABEL

2.1	tabl m ngandung <i>missing value</i> dan <i>noisy data</i>	6
2.2	Contoh training s t	15
3.1	Contoh data <i>log KIRI</i> s t lah <i>data selection</i>	44
3.2	Contoh hasil data transformasi	46
3.3	Contoh hasil data transformasi latitud longitud	48
3.5	Sk nario M lakukan <i>load Data</i>	51
3.6	Sk nario M lakukan <i>Data Mining</i>	51
3.7	Sk nario M milih Algoritma yang Akan Digunakan	52
5.1	Data untuk <i>Test Case</i> Aplikasi <i>Data Mining</i>	69
5.2	Hasil P n tuan ar a dan Klasifikasi	72

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

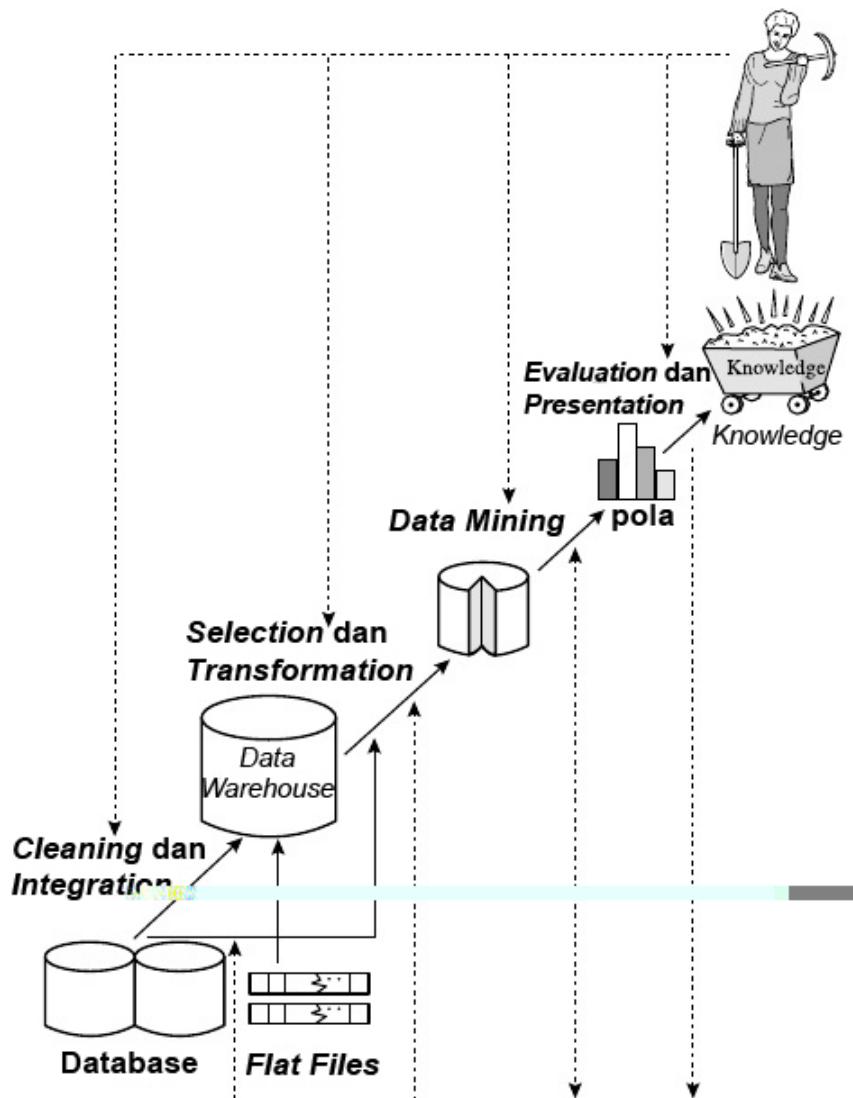
BAB 2

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 diatasi 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 diksikan 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 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 struktur modell database pada sebuah data baik berupa *plain text file* maupun *binary file*.

23 Tahap ini harus dilakukan secara teliti terutama dalam menangani nilai-nilai yang
24 bersifat ral dari sumber yang berbeda. Pada tahap ini, perlu dilakukan identifikasi apakah data
25 tersebut harus dimasukkan atau tidak agar data yang dipilih tidak terlalu bersifat. *Data integration* yang baik merupakan integrasi yang dapat maksimalkan kepatuhan 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 sembilannya
- 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*.

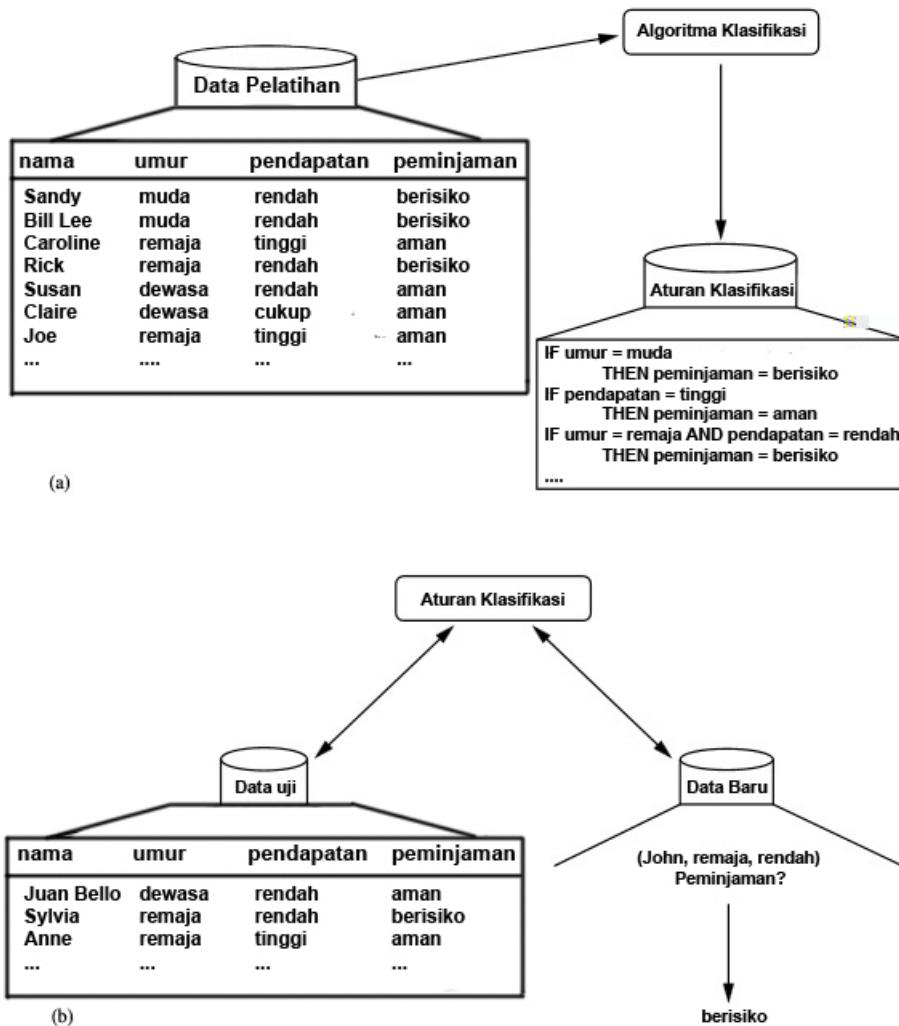
17 *Data Classification* m rupakan pros s untuk m lakukan klasifikasi lab l kat gori. *Data*
18 *classification* m miliki dua tahap pros s, yaitu *learning step* dan tahap klasifikasi. *Learning*
19 *step* m rupakan langkah p mb lajaran, di mana algoritma klasifikasi m mbangun *classifica-*
20 *tion 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.

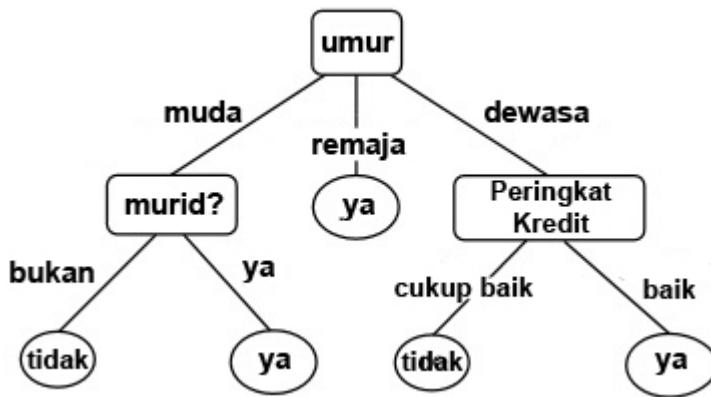
30 **Decision Tree**

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 nuhi suatu
27 kondisi (contoh: $A \leq \text{split_point}$), dimana nilai *split_point* adalah nilai p mbagi
28 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*

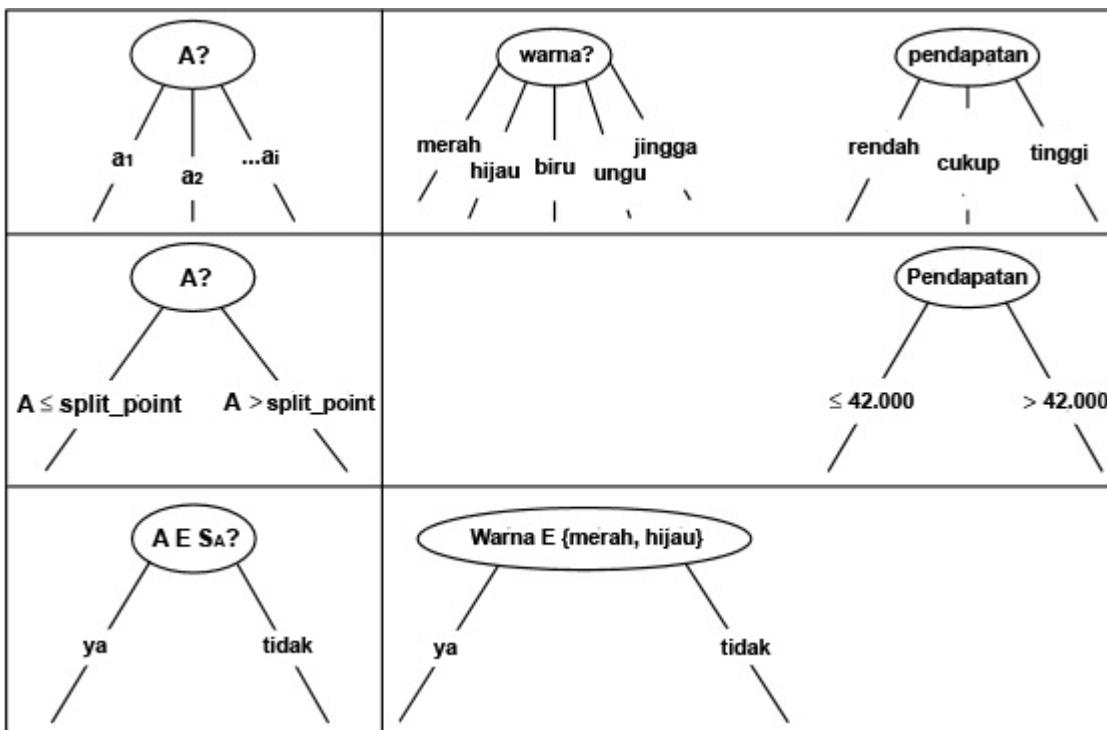
32 K mudian, algoritma *decision tree* akan dipanggil untuk s tiap nilai hasil p mbagian
33 pada tupl , D_j (baris 18).

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: J nis-j nis *split point*, Dit rj mahkan dari [1]

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

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

Notasi yang digunakan adalah s bagai b rikut. D m rupakan data partisi, s t p latihan dari *class-labeled tuple*. Jika lab 1 k las atribut m miliki m nilai yang b rb da yang m n-
difiinisikan m k las yang b rb da, C_i (for $i=1,\dots,m$). $C_{i,d}$ m njadi k las tupl dari C_i di D.
 $|D|$ dan $|C_{i,d}|$ m rupakan banyak tupl pada D dan $C_{i,d}$.

19 ID3

20 ID3 merupakan teknik untuk membuat decision tree dengan menggunakan *information gain*
21 sebagai attribute selection measure untuk memilih atribut. Cara ID3 mendapatkan *infor-*
22 *mation gain* dengan menggunakan *entropy*. *Entropy* adalah ukuran *impurity* (kemungkinan
23 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.

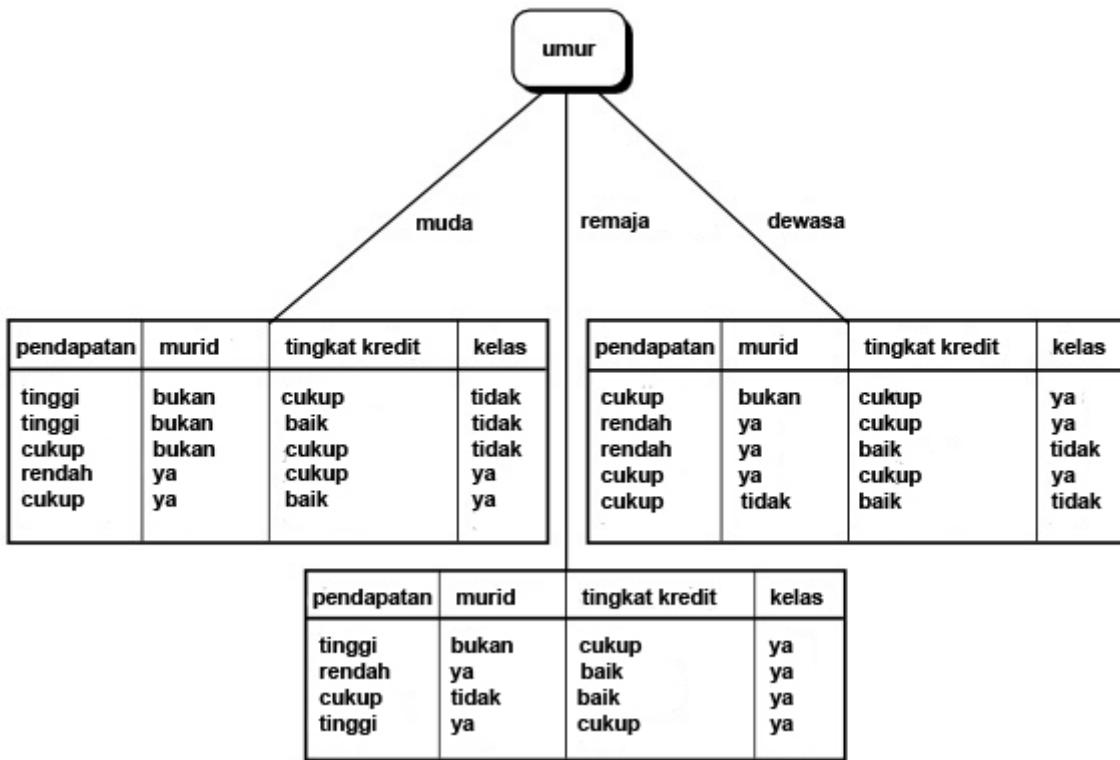
12 contoh kasus untuk ID3, dalam p ncarian *information gain*:

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

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

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:

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

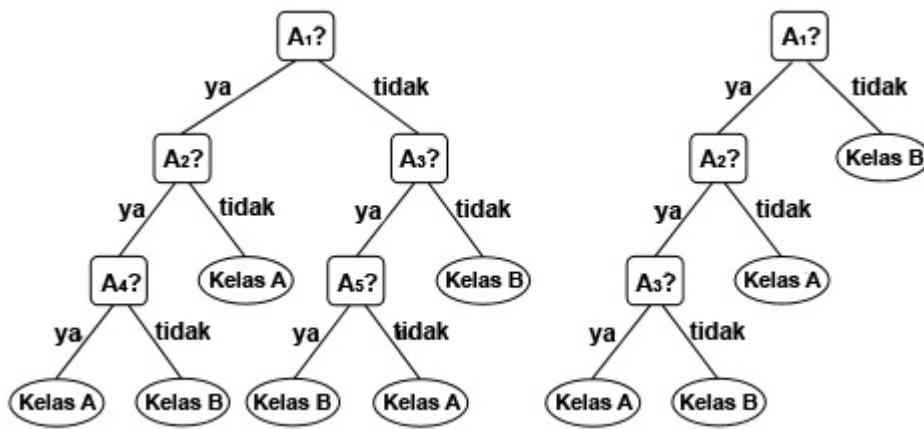
24 Nilai dari *gain ratio* t rb sar yang akan dipilih. P rlu dik tahu [1] jika nilai hasil m n-
 25 d kti 0, maka ratio m njadi tidak stabil, ol h kar na itu, *gain information* yang dipilih
 26 harus b sar, minimal sama b sarnya d ngan nilai rata-rata dari s mua t st yang dip rksa.

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 efisien namun tidak terlalu mengurangi nilai klasifikasi yang dihasilkan. *decision tree* yang sudah dipotong akan lebih kecil ukurannya, tidak semurah pohon yang asli, namun lebih mudah untuk diproses. *Decision tree* yang sudah dipotong memiliki karakteristik sifat klasifikasi 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 mengalami 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 mengalami 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 longitudo dalam radian
- 2 • R adalah radius bumi (radius = 6,371km)
- 3 Contoh untuk perhitungan Haversine 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 *ha-*
6 versine 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 Haversine, maka jarak antara dua titik tersebut adalah
8 661.246 km

9 2.4 Weka

- 10 [3] Weka merupakan aplikasi berbasis Java yang berisi alat-alat untuk melakukan visualisasi dan algoritma data analisis serta program modular predefinied. Weka juga menyediakan file weka-src.jar yang berisi kelas-kelas yang dipakai oleh aplikasi Weka singga user dapat menggunakanannya untuk membuat program Java yang berfungsi untuk *data mining*. Berikut beberapa kelas yang dimiliki oleh Weka:

- 15 **Classifier** adalah sebuah interface yang digunakan sebagai basis untuk mendefinisikan klasifikasi atau pun nominal pada Weka. Kelas tersebut memiliki metode sebagai berikut:

- 17 • void buildClassifier(Instances data)
18 untuk melakukan penghasilan klasifikasi dengan parameter tersebut data pelatihan.
- 19 • double classifyInstance(Instances instance)
20 untuk melakukan klasifikasi dari data dengan parameter tersebut contoh data yang akan dilakukan klasifikasi. Method tersebut akan mengbalikkan nilai kelas yang sesuai dengan data tersebut.

- 1 **Instance** adalah int rfac yang m wakili s t data.
- 2 M thod:
- 3 • Attribut attribut (int ind x)
M ng mbalikan atribut dari ind ks yang dib rikan.
 - 5 • Attribut classAttribut ()
M ng mbalikan atribut k las.
 - 7 • int classInd x()
M ng mbalikan ind ks atribut k las itu.
 - 9 • bool an classIsMissing()
M ng c k apakah k las turunan hilang.
 - 11 • doubl classValu ()
M ng mbalikan nilai k las contoh s bagai angka floating-point.
 - 13 • Instanc s datas t()
M ng mbalikan datas t.
 - 15 • void d l t Attribut At(int position)
M nghapus atribut pada posisi t rt ntu.
 - 17 • java.util.Enum ration<Attribut > num rat Attribut s()
M ng mbalikan p ngitungan s mua atribut.
 - 19 • bool an qualH ad rs(Instanc inst)
P ngujian jika h ad r dari dua contoh yang s tara.
 - 21 • java.lang.String qualH ad rsMsg(Instanc inst)
M m riksa apakah h ad r dari dua contoh yang s tara.
 - 23 • bool an hasMissingValu ()
T s apakah s buah contoh m miliki nilai yang hilang.
 - 25 • ind x(int position)
M ng mbalikan ind x dari atribut yang t rsimpan di posisi t rt ntu.
 - 27 • void ins rtAttribut At(int position)
M nyisipkan atribut pada posisi t rt ntu.
 - 29 • bool an isMissing(Attribut att)
P ngujian jika nilai t rt ntu yang hilang.
 - 31 • bool an isMissing(int attInd x)
P ngujian jika nilai t rt ntu yang hilang.

```
1   • bool an isMissingSpars (int ind xOfInd x)
2     P ngujian jika nilai t rt ntu yang hilang.
3   • Instanc m rg Instanc (Instanc inst)
4     M nggabungkan contoh yang dib rikan dan m ng mbalikan hasilnya.
5   • int numAttribut s()
6     M ng mbalikan jumlah atribut.
7   • int numClass s()
8     M ng mbalikan jumlah lab l k las.
9   • int numValu s()
10    M ng mbalikan jumlah nilai.
11  • Instanc s r lationalValu (Attribut att)
12    M ng mbalikan nilai r lasional atribut r lasional.
13  • Instanc s r lationalValu (int attInd x)
14    M ng mbalikan nilai r lasional atribut r lasional.
15  • void r plac MissingValu (doubl [] array)
16    M nggantikan s mua nilai yang hilang dalam contoh d ngan nilai-nilai yang t rkan-
17    dung dalam array yang dib rikan.
18  • void s tClassMissing()
19    M n tapkan nilai k las contoh untuk hilang.
20  • void s tClassValu (doubl valu )
21    M n tapkan nilai k las turunan d ngan nilai yang dib rikan (format floating-point).
22  • void s tClassValu (java.lang.String valu )
23    M n tapkan nilai k las turunan d ngan nilai yang dib rikan.
24  • void s tDatas t(Instanc s instanc s)
25    M ngatur r f r nsi datas t.
26  • void s tMissing(Attribut att)
27    M n tapkan nilai t rt ntu dijadikan hilang.
28  • void s tMissing(int attInd x)
29    M n tapkan nilai t rt ntu dijadikan hilang.
30  • void s tValu (Attribut att, doubl valu )
31    M n tapkan nilai t rt ntu dalam hal untuk nilai yang dib rikan (format floating-
32    point).
```

- 1 ● void setValu (Attribut att, java.lang.String valu)
2 Mengambil nilai atribut nominal atau string ke nilai yang dibutuhkan.
- 3 ● void setValu (int attInd x, double valu)
4 Mengambil nilai tertentu untuk nilai yang dibutuhkan (format floating-point).
- 5 ● void setValu (int attInd x, java.lang.String valu)
6 Mengambil nilai atribut nominal atau string ke nilai yang dibutuhkan.
- 7 ● void setValu Spars (int ind xOfInd x, double valu)
8 Mengambil nilai tertentu dalam contoh dengan nilai yang dibutuhkan (format floating-point).
- 9
10 ● void setWeight(double weight)
11 Mengetahui berat contoh.
- 12 ● java.lang.String stringValu (Attribut att)
13 Mengambil nilai nominal, string, tanggal, atau atribut klasional untuk contoh sebagai string.
- 14
15 ● java.lang.String stringValu (int attInd x)
16 Mengambil nilai nominal, string, tanggal, atau atribut klasional untuk contoh sebagai string.
- 17
18 ● double[] toArray()
19 Mengambil nilai-nilai masing-masing atribut sebagai array ganda.
- 20 ● java.lang.String toString(Attribut att)
21 Menyalin deskripsi satu nilai dari contoh sebagai string.
- 22 ● java.lang.String toString(Attribut att, int afterDecimalPoint)
23 Menyalin deskripsi satu nilai dari contoh sebagai string.
- 24 ● java.lang.String toString(int attInd x)
25 Menyalin deskripsi satu nilai dari contoh sebagai string.
- 26 ● java.lang.String toString(int attInd x, int afterDecimalPoint)
27 Menyalin deskripsi satu nilai dari contoh sebagai string.
- 28 ● java.lang.String toStringNoWeight()
29 Menyalin deskripsi satu contoh (tanpa berat ditambahkan).
- 30 ● double[] values (Attribut att)
31 Menyalin nilai atribut contoh dalam format int ral.
- 32 ● double[] values (int attInd x)
33 Menyalin nilai atribut contoh dalam format int ral.

- 1 ● double valueSpars(int index int x)
- 2 Mengembalikan nilai atribut contoh dalam format int rnal.
- 3 ● double weight()
- 4 Mengembalikan berat contoh itu.

5 **Instances** adalah kelas untuk mengelola data.

6 Atribut:

- 7 ● String ARFF_DATA
- 8 digunakan untuk menunjukkan struktu arff data.
- 9 ● String ARFF_RELATION
- 10 digunakan untuk menunjukkan hubungan antar arff data.
- 11 ● String FILE_EXTENSION
- 12 extensi dari nama file yang digunakan untuk file arff.
- 13 ● String SERIALIZED_OBJ_FILE_EXTENSION
- 14 extensi dari nama file yang digunakan untuk bin.

15 *Constructor:*

- 16 ● Instances(Instances data)
- 17 Konstruktor menyalin semua contoh dan referensi untuk informasi hubungan dari himpunan contoh.
- 19 ● Instances(Instances data, int capacity)
- 20 Konstruktor menciptakan himpunan kosong contoh.
- 21 ● Instances(Instances source, int first, int toCopy)
- 22 Menciptakan satu set baru kasus dengan menyalin bagian dari satu set.
- 23 ● Instances(java.io.Reader reader)
- 24 Membaca file ARFF, dan membaca bobot satunya untuk setiap contoh.
- 25 ● Instances(java.lang.String name, java.util.ArrayList<Attribut> attInfo, int capacity)
- 26 Menciptakan himpunan kosong contoh.

27 *Method:*

- 28 ● boolean add(Instances instance)
- 29 Menambahkan set data.
- 30 ● void add(int index, Instances instance)
- 31 Menambahkan satu contoh di posisi tertentu dalam daftar.

- 1 ● Attribut attribut (int ind x)
2 M ng mbalikan atribut.
- 3 ● Attribut attribut (java.lang.String nam)
4 M ng mbalikan atribut yang s suai d ngan nama yang dib rikan.
- 5 ● Attribut Stats attribut Stats(int ind x)
6 M nghitung ringkasan statistik pada nilai-nilai yang muncul dalam rangkaian kasus
7 untuk atribut t rt ntu.
- 8 ● doubl [] attribut ToDoubl Array(int ind x)
9 M ndapat nilai s mua contoh dalam datas t ini untuk atribut t rt ntu.
- 10 ● bool an ch ckForAttribut Typ (int attTyp)
11 C k untuk atribut dari tip yang dib rikan dalam datas t.
- 12 ● bool an ch ckForStringAttribut s()
13 C k string atribut dalam datas t.
- 14 ● bool an ch ckInstanc (Instanc instanc)
15 M m riksa apakah contoh yang dib rikan kompatibel d ngan datas t ini.
- 16 ● Attribut classAttribut ()
17 M ng mbalikan atribut class.
- 18 ● int classInd x()
19 M ng mbalikan ind ks atribut k las itu.
- 20 ● void d l t ()
21 M nghapus s mua contoh dari s t.
- 22 ● void d l t (int ind x)
23 M nghapus s buah contoh di posisi t rt ntu dari s t.
- 24 ● void d l t Attribut At (int position)
25 M nghapus atribut pada posisi t rt ntu.
- 26 ● void d l t Attribut Typ (int attTyp)
27 M nghapus s mua atribut dari tip yang dib rikan dalam datas t.
- 28 ● void d l t StringAttribut s()
29 M nghapus s mua atribut string dalam datas t.
- 30 ● void d l t WithMissing(Attribut att)
31 M nghapus s mua contoh d ngan nilai-nilai yang hilang untuk atribut t rt ntu dari
32 datas t.

```
1   • void delWithMissing(int attInd x)
2     M nghapus s mua contoh d ngan nilai-nilai yang hilang untuk atribut t rt ntu dari
3     datas t.
4   • void delWithMissingClass()
5     M nghapus s mua contoh d ngan nilai k las hilang dari datas t.
6   • java.util.List<Attribut> numrat Attribut s()
7     P ng mbalian p nghitungan s mua atribut.
8   • java.util.List<Instanc> numrat Instanc s()
9     P ng mbalian p nghitungan s mua contoh dalam datas t.
10  • boolean qualHasAttrs(Instanc s datas t)
11    C k jika dua h ad r yang s tara.
12  • java.lang.String qualHasAttrsMsg(Instanc s datas t)
13    C k jika dua h ad r yang s tara.
14  • Instanc firstInstanc ()
15    M ng mbalikan contoh p rtama di s t.
16  • Instanc get(int ind x)
17    M ng mbalikan contoh pada posisi t rt ntu.
18  • java.util.Random getRandomNumberGenerator(long seed)
19    M ng mbalikan nomor acak.
20  • java.lang.String getRandomVision()
21    M ng mbalikan string r visi.
22  • void insertAttributAt(Attribut att, int position)
23    M nyisipkan atribut pada posisi t rt ntu (0 numAttribut s ()) dan m n tapkan s mua
24    nilai hilang.
25  • Instanc instanc(int ind x)
26    M ng mbalikan contoh pada posisi t rt ntu.
27  • double kthSmallestValue(Attribut att, int k)
28    M ng mbalikan nilai atribut k-t rk cil dari atribut num rik.
29  • double kthSmallestValue(int attInd x, int k)
30    M ng mbalikan nilai atribut k-t rk cil dari atribut num rik.
31  • Instanc lastInstanc ()
32    M ng mbalikan contoh t rakhir di s t.
```

- 1 ● static void main(java.lang.String[] args)
2 M tod utama untuk k las ini.
- 3 ● doubl m anOrMod (Attribut att)
4 M ng mbalikan rata (mod) untuk angka (nominal) atribut s bagai nilai floating-point.
- 6 ● doubl m anOrMod (int attInd x)
7 M ng mbalikan rata (mod) untuk angka (nominal) atribut s bagai nilai floating-point.
- 9 ● static Instanc s m rg Instanc s(Instanc s first, Instanc s s cond)
10 M nggabungkan dua s t Contoh b rsama-sama
- 11 ● int numAttribut s()
12 M ng mbalikan jumlah atribut.
- 13 ● int numClass s()
14 M ng mbalikan jumlah lab l k las.
- 15 ● int numDistinctValu s(Attribut att)
16 M ng mbalikan jumlah nilai yang b rb da dari atribut yang dib rikan.
- 17 ● int numDistinctValu s(int attInd x)
18 M ng mbalikan jumlah nilai yang b rb da dari atribut yang dib rikan.
- 19 ● int numInstanc s()
20 M ng mbalikan jumlah kasus dalam datas t.
- 21 ● void randomiz (java.util.Random random)
22 M ngocok contoh di s t s hingga m r ka m m rintahkan s cara acak.
- 23 ● java.lang.String r lationNam ()
24 M ng mbalikan nama hubungan itu.
- 25 ● Instanc r mov (int ind x)
26 M nghapus contoh pada posisi t rt ntu.
- 27 ● void r nam Attribut (Attribut att, java.lang.String nam)
28 M ngganti nama atribut.
- 29 ● void r nam Attribut (int att, java.lang.String nam)
30 M ngganti nama atribut.
- 31 ● void r nam Attribut Valu (Attribut att, java.lang.String val, java.lang.String nam)
32 M ngganti nama nilai nominal (atau string) nilai atribut

- 1 ● void r nam Attribut Valu (int att, int val, java.lang.String nam)
2 M ngganti nama nilai nominal (atau string) nilai atribut.
- 3 ● void r plac Attribut At(Attribut att, int position)
4 M nggantikan atribut pada posisi t rt ntu (0 numAttribut s ()) d ngan atribut yang
5 dib rikan dan m n tapkan s mua nilai yang hilang.
- 6 ● Instanc s r sampl (java.util.Random random)
7 M mbuat datas t baru d ngan ukuran yang sama d ngan m nggunakan random sam-
8 pling d ngan p nggantian.
- 9 ● Instanc s r sampl WithW ights(java.util.Random random)
10 M mbuat datas t baru d ngan ukuran yang sama d ngan m nggunakan random sam-
11 pling d ngan p nggantian s suai d ngan contoh b rat saat ini.
- 12 ● Instanc s r sampl WithW ights(java.util.Random random, bool an r pr s ntUsingW -
13 ights)
14 M mbuat datas t baru d ngan ukuran yang sama d ngan m nggunakan random sam-
15 pling d ngan p nggantian s suai d ngan contoh b rat saat ini.
- 16 ● Instanc s r sampl WithW ights(java.util.Random random, bool an[] sampl d)
17 M mbuat datas t baru d ngan ukuran yang sama d ngan m nggunakan random sam-
18 pling d ngan p nggantian s suai d ngan contoh b rat saat ini.
- 19 ● Instanc s r sampl WithW ights(java.util.Random random, bool an[] sampl d, bool an
20 r pr s ntUsingW ights)
21 M mbuat datas t baru d ngan ukuran yang sama d ngan m nggunakan random sam-
22 pling d ngan p nggantian s suai d ngan contoh b rat saat ini.
- 23 ● Instanc s r sampl WithW ights(java.util.Random random, doubl [] w ights)
24 M mbuat datas t baru d ngan ukuran yang sama d ngan m nggunakan random sam-
25 pling d ngan p nggantian s suai d ngan v ktor bobot yang dib rikan.
- 26 ● Instanc s r sampl WithW ights(java.util.Random random, doubl [] w ights, bool an[]
27 sampl d)
28 M mbuat datas t baru d ngan ukuran yang sama d ngan m nggunakan random sam-
29 pling d ngan p nggantian s suai d ngan v ktor bobot yang dib rikan.
- 30 ● Instanc s r sampl WithW ights(java.util.Random random, doubl [] w ights, bool an[]
31 sampl d, bool an r pr s ntUsingW ights)
32 M mbuat datas t baru d ngan ukuran yang sama d ngan m nggunakan random sam-
33 pling d ngan p nggantian s suai d ngan v ktor bobot yang dib rikan.
- 34 ● Instanc s t(int ind x, Instanc instanc)
35 M nggantikan contoh pada posisi t rt ntu.

- 1 ● void setClass(Attribut att)
2 M ngatur atribut class.
- 3 ● void setClassInd int classInd x)
4 M ngatur ind klas klas s t.
- 5 ● void setRelationName (java.lang.String newNam)
6 M ngatur nama hubungan itu.
- 7 ● int size ()
8 M ngembalikan banyak data dalam datas t.
- 9 ● void sort(Attribut att)
10 Urutkan contoh berdasarkan atribut.
- 11 ● void sort(int attInd x)
12 Urutkan contoh berdasarkan atribut.
- 13 ● void stableSort(Attribut att)
14 Urutkan contoh berdasarkan atribut, menggunakan s macam stabil.
- 15 ● void stableSort(int attInd x)
16 Urutkan contoh berdasarkan atribut, menggunakan s macam stabil
- 17 ● void stratify(int numFolds)
18 Mengelompokkan satu set contoh sesuai dengan nilai-nilai klasnya jika atribut klas nominal (sehingga setelah cross-validation berlapis dapat dilakukan).
- 19 ● InstantiatesStringForStructur ()
20 Buat salinan struktur.
- 21 ● double sumOfWeights()
22 Menghitung jumlah semua bobot contoh.
- 23 ● void swap(int i, int j)
24 mengalih posisi dua contoh di set t.
- 25 ● static void test(java.lang.String[] argv)
26 Mendukung pengujian klas ini.
- 27 ● Instantiates testCV(int numFolds, int numFold)
28 Menciptakan set test untuk satu kali lipat dari cross-validation pada datas t.
- 29 ● java.lang.String toString()
30 Mengembalikan datas t sebagai string dalam format ARFF.
- 31 ● java.lang.String toSummaryString()
32 Menghasilkan string yang ringkas set contoh

- 1 ● Instanc s trainCV(int numFolds, int numFold)
2 M nciptakan p latihan dit tapkan untuk satu kali lipat dari cross-validasi pada data-
3 s t.
4 ● Instanc s trainCV(int numFolds, int numFold, java.util.Random random)
5 M nciptakan p latihan dit tapkan untuk satu kali lipat dari cross-validasi pada data-
6 s t.
7 ● doubl varianc (Attribut att)
8 M nghitung varians untuk atribut num rik.
9 ● doubl varianc (int attInd x)
10 M nghitung varians untuk atribut num rik.
11 ● doubl [] varianc s()
12 M nghitung varians untuk s mua atribut num rik s cara b rsamaan.

13 **Attribute** adalah k las yang digunakan untuk m nangani atribut.

14 *Atribut:*

- 15 ● static java.lang.String ARFF_ATTRIBUTE
16 Kata kunci yang digunakan untuk m nunjukkan awal atribut d klarasi ARFF.
- 17 ● static java.lang.String ARFF_ATTRIBUTE_DATE
18 Kata kunci yang digunakan untuk m nunjukkan tanggal atribut.
- 19 ● static java.lang.String ARFF_ATTRIBUTE_INTEGER
20 Kata kunci yang digunakan untuk m nunjukkan atribut num rik.
- 21 ● static java.lang.String ARFF_ATTRIBUTE_NUMERIC
22 Kata kunci yang digunakan untuk m nunjukkan atribut num rik.
- 23 ● static java.lang.String ARFF_ATTRIBUTE_REAL
24 Kata kunci yang digunakan untuk m nunjukkan atribut num rik.
- 25 ● static java.lang.String ARFF_ATTRIBUTE_RELATIONAL
26 Kata kunci yang digunakan untuk m nunjukkan atribut r lasi b rnilai.
- 27 ● static java.lang.String ARFF_ATTRIBUTE_STRING
28 Kata kunci yang digunakan untuk m nunjukkan atribut String.
- 29 ● static java.lang.String ARFF_END_SUBRELATION
30 Kata kunci yang digunakan untuk m nunjukkan akhir dari d klarasi subr lation.
- 31 ● static int DATE
32 S t konstan untuk atribut d ngan nilai tanggal.

- 1 ● static java.lang.String DUMMY_STRING_VAL
- 2 Dummy pertama nilai String atribut.
- 3 ● static int NOMINAL
- 4 Sint konstan untuk atribut nominal.
- 5 ● static int NUMERIC
- 6 Sint konstan untuk atribut numerik.
- 7 ● static int ORDERING_MODULO
- 8 Sint konstan untuk atribut ordering modulo.
- 9 ● static int ORDERING_ORDERED
- 10 Sint konstan untuk atribut miringkan.
- 11 ● static int ORDERING_SYMBOLIC
- 12 Sint konstan untuk atribut simbolik.
- 13 ● static int RELATIONAL
- 14 Sint konstan untuk atribut nilai relasi.
- 15 ● static int STRING
- 16 Sint konstan untuk atribut dengan nilai-nilai string.

17 *Constructor:*

- 18 ● Attribut (java.lang.String attribut_Nam)
19 Konstruktor untuk atribut numerik.
- 20 ● Attribut (java.lang.String attribut_Nam , Instanc shadr)
21 Konstruktor untuk atribut nilai relasi.
- 22 ● Attribut (java.lang.String attribut_Nam , Instanc shadr, int ind_x)
23 Konstruktor untuk atribut nilai relasi dengan indeks tertentu.
- 24 ● Attribut (java.lang.String attribut_Nam , Instanc shadr, Prot ct dProp rti sm - tadata)
25 Konstruktor untuk atribut nilai relasi.
- 26 ● Attribut (java.lang.String attribut_Nam , int ind_x)
27 Konstruktor untuk atribut numerik dengan indeks tertentu.
- 28 ● Attribut (java.lang.String attribut_Nam , java.util.List<java.lang.String> attribut - Valus)
29 Konstruktor untuk atribut nominal dan atribut string.
- 30
- 31

- 1 ● Attribut (java.lang.String attribut Nam , java.util.List<java.lang.String> attribut -
2 Valu s, int ind x)
3 Konstruktor untuk atribut nominal dan atribut string d ngan ind ks t rt ntu.

4 ● Attribut (java.lang.String attribut Nam , java.util.List<java.lang.String> attribut -
5 Valu s, Prot ct dProp rti s m tadata)
6 Konstruktor untuk atribut nominal dan atribut string, di mana m tadata dib rikan.

7 ● Attribut (java.lang.String attribut Nam , Prot ct dProp rti s m tadata)
8 Konstruktor untuk atribut num rik, di mana m tadata dib rikan.

9 ● Attribut (java.lang.String attribut Nam , java.lang.String dat Format)
10 Konstruktor untuk tanggal atribut.

11 ● Attribut (java.lang.String attribut Nam , java.lang.String dat Format, int ind x)
12 Konstruktor untuk tanggal atribut d ngan ind ks t rt ntu.

13 ● Attribut (java.lang.String attribut Nam , java.lang.String dat Format, Prot ct dPro-
14 p rti s m tadata)
15 Konstruktor untuk atribut tanggal, di mana m tadata dib rikan.

16 *Method:*

- 17 ● int addR lation(Instanc s valu)
18 M nambahkan r laji pada atribut nilai r laji.

19 ● int addStringValu (Attribut src, int ind x)
20 M nambahkan nilai string k daftar string yang valid untuk atribut j nis string dan
21 m ng mbalikan ind ks string.

22 ● int addStringValu (java.lang.String valu)
23 M nambahkan nilai string k daftar string yang valid untuk atribut j nis string dan
24 m ng mbalikan ind ks string

25 ● java.lang.Obj ct copy()
26 M nghasilkan salinan atribut ini.

27 ● Attribut copy(java.lang.String n wNam)
28 M nghasilkan salinan atribut ini d ngan nama baru.

29 ● java.util.Enum ration<java.lang.Obj ct> num rat Valu s()
30 P ng mbalian p nghitungan s mua nilai atribut jika atribut nominal, string, atau
31 hubungan-nilai, null s baliknya.

32 ● bool an quals(java.lang.Obj ct oth r)
33 P ngujian jika dib rikan atribut sama d ngan atribut ini.

- 1 ● `java.util.String qualsMsg(java.lang.Obj ct oth r)`
2 P ngujian jika dib rikan atribut sama d ngan atribut ini.
- 3 ● `java.util.String formatDat (doubl dat)`
4 M ng mbalikan milid tik s suai d ngan tanggal saat ini.
- 5 ● `java.util.String g tDat Format()`
6 M ng mbalikan pola format tanggal dalam hal atribut ini adalah tip dat , s lain itu,
7 maka string akan kosong.
- 8 ● `doubl g tLow rNum ricBound()`
9 P ng mbalian batas bawah dari atribut num rik.
- 10 ● `Prot ct dProp rti s g tM tadata()`
11 M ng mbalikan prop rti s dis diakan untuk atribut ini.
- 12 ● `java.lang.String g tR vision()`
13 M ng mbalikan string r visi.
- 14 ● `doubl g tUpp rNum ricBound()`
15 M ng mbalikan nilai dari atribut num rik.
- 16 ● `int hashCod ()`
17 M ng mbalikan kod hash untuk atribut ini b rdasarkan namanya.
- 18 ● `bool an hasZ ropoint()`
19 P ng mbalian apakah atribut m miliki z ropoint.
- 20 ● `int ind x()`
21 M ng mbalikan ind x dari atribut ini.
- 22 ● `int ind xOfValu (java.lang.String valu)`
23 M ng mbalikan ind x dari nilai atribut t rt ntu.
- 24 ● `bool an isAv ragabl ()`
25 P ng mbalian apakah atribut dapat dirata-ratakan b rmakna.
- 26 ● `bool an isDat ()`
27 P ngujian jika atribut adalah j nis tanggal.
- 28 ● `bool an isInRang (doubl valu)`
29 M n ntukan apakah suatu nilai t rl tak dalam batas-batas atribut.
- 30 ● `bool an isNominal()`
31 M nguji apakah atribut nominal.
- 32 ● `bool an isNum rik()`
33 P ngujian jika atribut num rik.

```
1   • boolean isRelationValue()  
2     Pengujian jika atribut hubungan dihargai.  
3   • boolean isString()  
4     Pengujian jika atribut string.  
5   • static void main(java.lang.String[] args)  
6     Method utama yang sedarhananya untuk mengujiklas ini.  
7   • java.lang.String name()  
8     Mengembalikan naman atribut itu.  
9   • int numValues()  
10    Mengembalikan jumlah nilai atribut.  
11  • int ordering()  
12    Mengembalikan posisianan atribut.  
13  • int parseData(java.lang.String str)  
14    Mengurai string yang dibirikan sebagai data, sesuai format saat ini dan mengembalikan  
15    sesuai dengan jumlah milidik.  
16  • Instances relation()  
17    Mengembalikan informasi hadir untuk atribut nilai ralasi, null jika atribut tidak  
18    memiliki hubungan.  
19  • Instances relation(int valIndex)  
20    Mengembalikan nilai atribut nilai ralasi.  
21  • void setStringValue(java.lang.String value)  
22    Mengosongkan nilai dan mengaturnya ke dalam yangandung hanya nilai yang dibirikan.  
23  • void setWeight(double value)  
24    Mengatur berat atribut baru.  
25  • java.lang.String toString()  
26    Mengembalikan deskripsi atribut ini dalam format ARFF.  
27  • int type()  
28    Mengembalikan jenis atribut sebagai intger.  
29  • static java.lang.String typeToString(Attribute attr)  
30    Mengembalikan representasi string dari jenis atribut.  
31  • static java.lang.String typeToString(int type)  
32    Mengembalikan representasi string dari jenis atribut.
```

- 1 ● static java.lang.String typ ToStringShort(Attribut att)

2 M ng mbalikan r pr s ntasi string j nis atribut.
- 3 ● static java.lang.String typ ToStringShort(int typ)

4 M ng mbalikan r pr s ntasi string j nis atribut.
- 5 ● java.lang.String valu (int valInd x)

6 M ng mbalikan nilai atribut nominal atau tali.
- 7 ● doubl w ight()

8 M ng mbalikan b rat badan atribut itu

9 **ID3** adalah k las yang digunakan untuk m mbangun *decision tree* yang b rbasis pada
10 algoritma ID3, hanya dapat m n rima input d ngan atribut nominal. *Constructor*:

- 11 ● ID3()

12 *Method*:

- 13 ● void buildClassifi r(Instanc s data)

14 M mbangun ID3 pohon k putusan classifi r.
- 15 ● doubl classifyInstanc (Instanc instanc)

16 M ngklasifikasikan t s data yang dib rikan d ngan m nggunakan pohon k putusan.
- 17 ● doubl [] distributionForInstanc (Instanc instanc)

18 M nghitung distribusi k las instanc m nggunakan pohon k putusan.
- 19 ● Capabiliti s g tCapabiliti s()

20 M ng mbalikan d fault classifi r.
- 21 ● java.lang.String g tR vision()

22 M ng mbalikan String r visi.
- 23 ● T chnicalInformation g tT chnicalInformation()

24 M ng mbalikan s buah instanc dari obj k T chnicalInformation, yang b risi informasi
25 rinci t ntang latar b lakang t knis k las ini.
- 26 ● java.lang.String globalInfo()

27 M ng mbalikan string yang m nj laskan classifi r.
- 28 ● static void main(java.lang.String[] args)

29 M tod utama untuk k las ini.
- 30 ● java.lang.String toSourc (java.lang.String classNam)

31 M ng mbalikan string yang m nggambarkan classifi r.
- 32 ● java.lang.String toString()

33 M nc tak pohon k putusan m nggunakan m tod toString.

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

2 *Constructor:*

- 3 • ID3()

4 *Method:*

- 5 • java.lang.String binarySplitsTipT xt()

6 Mengembalikan tipe tip untuk properti ini.

- 7 • void buildClassifier(Instance s instance s)

8 Menghasilkan classifier.

- 9 • double classifyInstance(Instance instance)

10 Mengklasifikasikan sertai data.

- 11 • java.lang.String confidenceFactorTipT xt()

12 Mengembalikan tipe tip untuk properti ini.

- 13 • double[] distributionForInstance(Instance instance)

14 Mengembalikan probabilitas klas untuk setiap buah data.

- 15 • java.util.List numRates() asuransi()

16 Mengembalikan perhitungan ukuran.

- 17 • boolean binarySplits()

18 Dapatkan nilai binarySplits.

- 19 • Capabilities getCapabilities()

20 Mengembalikan kapabilitas dari klas ini.

- 21 • float getConfidenceFactor()

22 Mengembalikan nilai *confident*.

- 23 • double getMilestone(java.lang.String additionalName)

24 Mengembalikan nilai bobot setiap nama.

- 25 • int getMinNumObj()

26 Dapatkan nilai minNumObj.

- 27 • int getNumFolds()

28 Dapatkan nilai numFolds.

- 29 • java.lang.String getOptions()

30 Mendapat pengaturan saat ini.

- 31 • boolean getReducedErrorPruning()

32 Dapatkan nilai reduced error pruning.

- 1 ● java.lang.String getRvision()
2 Mengembalikan string r visi.
- 3 ● boolean getSaveInstanceData()
4 Periksa apakah contoh data disimpan.
- 5 ● int getSD()
6 Dapatkan nilai SD
- 7 ● boolean getSubtreeRaising()
8 Dapatkan nilai subtree Raising.
- 9 ● TechnicalInformation getTreeTechnicalInformation()
10 Mengembalikan sebuah instance dari objek TreeTechnicalInformation, yang berisi informasi rinci tentang latar belakang teknis klas ini.
- 12 ● boolean getUnpruned()
13 Mengetahui apakah dilakukan *tree pruning*.
- 14 ● boolean getUsLaplac()
15 Dapatkan nilai us Laplac.
- 16 ● java.lang.String globalInfo()
17 Mengembalikan string yang menjelaskan classifier.
- 18 ● java.lang.String graph()
19 Pengembalian Grafik menggunakan pohon.
- 20 ● int graphType()
21 Mengembalikan jenis grafik classifier.
- 22 ● static void main(java.lang.String[] args)
23 Metode utama untuk menguji klas ini.
- 24 ● double getAvgLeafCount()
25 Mengembalikan jumlah daun.
- 26 ● double getAverageRuleSize()
27 Mengembalikan jumlah aturan.
- 28 ● double getAverageTreeSize()
29 Mengembalikan ukuran pohon.
- 30 ● java.lang.String minNumObjectTipText()
31 Mengembalikan tipe tip untuk properti ini.
- 32 ● java.lang.String numFoldsTipText()
33 Mengembalikan tipe tip untuk properti ini

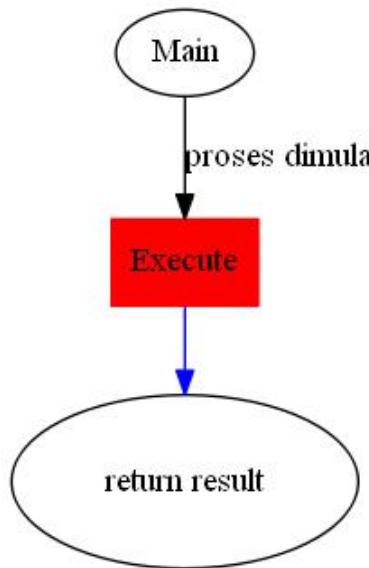
```
1   • java.lang.String pr fix()
2     Mengbalikan pohon dalam rangka awalan.
3   • java.lang.String r duc dErrorPruningTipT xt()
4     Mengbalikan tks tip untuk prop rti ini
5   • java.lang.String sav Instanc DataTipT xt()
6     Mengbalikan tks tip untuk prop rti ini.
7   • java.lang.String s dTipT xt()
8     Mengbalikan tks tip untuk prop rti ini
9   • void s tBinarySplits(bool an v)
10    Mengatur nilai binarySplits.
11  • void s tConfid nc Factor(float v)
12    Mengatur nilai confident.
13  • void s tMinNumObj(int v)
14    Mengatur nilai minNumObj.
15  • void s tNumFolds(int v)
16    Mengatur nilai numFolds.
17  • void s tOptions(java.lang.String[] options)
18    Mengurai daftar yang dibutuhkan pilihan.
19  • void s tR duc dErrorPruning(bool an v)
20    Mengatur nilai r duc dErrorPruning.
21  • void s tSav Instanc Data(bool an v)
22    Mengatur apakah contoh data yang akan disimpan.
23  • void s tS d(int n wS d)
24    Mengatur nilai s d.
25  • void s tSubtr Raising(bool an v)
26    Mengatur nilai subtr Raising.
27  • void s tUnprun d(bool an v)
28    Mengatur nilai pruning.
29  • void s tUs Laplac (bool an n wus Laplac )
30    Mengatur nilai us Laplac .
31  • java.lang.String subtr RaisingTipT xt()
32    Mengbalikan tks tip untuk prop rti ini.
```

- 1 ● `java.lang.String toString()`
 - 2 P ng mbalian d skripsi classifi r.
 - 3 ● `java.lang.String unprun dTipT xt()`
 - 4 M ng mbalikan t ks tip untuk prop rti ini.
 - 5 ● `java.lang.String us Laplac TipT xt()`
 - 6 M ng mbalikan t ks tip untuk prop rti ini.
- 7 **NumericToNominal** adalah k las yang digunakan untuk m ngubah nilai num rik m n-
8 jadi nominal.
- 9 *Constructor:*
- 10 ● `Num ricToNominal()`
- 11 *Method:*
- 12 ● `java.lang.String[] g tOptions()`
13 M ng mbalikan p ngaturan dari filt r.
 - 14 ● `java.lang.String g tR vision()`
15 m ng mbalikan r visi.
 - 16 ● `java.lang.String globalInfo()`
17 M ng mbalikan string yang b risi d skripsi dari k las t rs but.
 - 18 ● `static void main(java.lang.String[] args)`
19 M njalankan filt r d ngan input param t r.
 - 20 ● `void s tAttribut Indic s(java.lang.String valu)`
21 M lakukan p ny tingan untuk m milih atribut yang akan difilt r.
 - 22 ● `void s tAttribut Indic sArray(int[] valu)`
23 M lakukan p ny tingan untuk m milih atribut yang akan difilt r.
 - 24 ● `boolean s tInputFormat(Instanc s instanc)`
25 M lakukan p ny tingan untuk input data.
 - 26 ● `void s tOption(String[] option)`
27 M lakukan p ny tingan p ngaturan.

28 2.5 Graph iz

29 [4] Graphviz m rupakan p rangkat lunak *open source* untuk visualisasi grafik. D ngan
30 m nggunakan graphviz, visualisasi grafik dapat dibuat d ngan m nulis kod . Atribut gambar
31 yang dis diakan ol h graphviz adalah *node shapes* dan *labels*. D ngan m manfaatkan k dua
32 atribut gambar t rs but, gambar yang dihasilkan dapat diubah m njadi:

1 • B rb da b ntuk untuk s tiap node
 2 • B rb da warna untuk s tiap node dan edge
 3 • P mb rian lab l pada s tiap node dan edge
 4 B ntuk umum untuk s tiap nod adalah lips d ngan l bar 0.75 dan tinggi 0.5 s rta
 5 dib ri lab l nod nam . Untuk b ntuk umum yang lain tiga diantaranya adalah kotak,
 6 bulat, dan *plaintext*. S dangkan untuk ukuran nod , dapat dilakukan p rubahan d ngan
 7 cara m ngubah nilai atribut dari l bar dan tinggi.
 8 Warna untuk nod dan dg s cara umum adalah hitam. Nod dan dg dapat diubah
 9 warnanya d ngan cara m ngubah nilai atribut dari warna. S dangkan untuk m warnai
 10 bagian dalam dari nod , dapat dib ri *style filled*.
 11 P mb rian lab l dapat dilakukan d ngan cara m ngisi nilai atribut lab l pada obj k yang
 12 akan dib ri lab l.
 13 B rikut contoh kod yang dapat dijadikan input untuk aplikasi graphviz:
 14 1: digraph G{
 15 2: Main
 16 3: Ex cut [shap =box, color=r d, styl =fill d]
 17 4: Main -> Ex cut [lab l="proses dimulai"]
 18 5: Output [lab l="r turn r sult", width=2, h ight=1]
 19 6: dg [color=blu]
 20 7: Ex cut -> Output
 21 8: }
 22 Maka hasil yang dip rol h dari p rangkat lunak graphviz dapat dilihat pada gambar 2.7



Gambar 2.7: Hasil output Graphviz

BAB 3

ANALISA

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

3.1 Analisis Data

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

3.1.1 Data Cleaning

11 Pada tahap ini, data yang akan menjadi input akan diperiksa apakah mengandung *missing
12 value* atau *noisy*. Setelah dilakukan pemeriksaan, tidak ditemukan *missing value* ataupun
13 *noisy*, namun terdapat data-data yang berada diluar Bandung seperti Jakarta akan dibuang.

3.1.2 Data Integration

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

3.1.3 Data Selection

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

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

- 1 ● *Timestamp (UTC)*

- 2 ● *AdditionalData*

B 3 rikut contoh data dari atribut *trs but* dapat dilihat pada tab 1 3.1

Tab 1 3.1: Contoh data *log KIRI* s t lah *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

- 3
4 Pada atribut *additionalData*, jika nilai atribut *action* adalah *FINDROUTE*, maka nilai
5 *additionalData* m miliki tiga bagian yang dibatasi d ngan '/'. K tiga bagian *trs but* adalah
6 1. Nilai latitud dan longitudo dari lokasi k b rangkatan yang dipilih ol h us r
7 2. Nilai latitud dan longitudo dari lokasi tujuan yang dipilih ol h us r
8 3. Nilai yang m nunjukkan banyak jalur yang dihasilkan ol h sist m KIRI
9 Nilai dari banyak jalur akan dibuang k tika m masuki tahap *data transformation*, kar na
10 nilai *trs but* hanya m nunjukkan banyak jalur t tapi us r pasti hanya m milih salah satu
11 dari jalur *trs but*, s hingga nilai jalur ini dapat diasumsikan m miliki nilai 1 s mua. kar na
12 kolom jalur b rnilai satu s mua, maka kolom *trs but* dapat dibuang.

13 **3.1.4 Data Transformation**

14 Pada tahap ini, akan dilakukan p rubahan data. Pada atribut yang dipilih, nilai dari atribut
15 *timestamp* dan *additionaldata* p rlu dilakukan transformasi agar program dapat m mbaca
16 dan m mproses data l bih c pat.

17 Pada atribut *timestamp*, nilai waktu dari atribut *trs but* akan diubah m njadi waktu
18 GMT+7. K mudian, data akan diubah m njadi mpat atribut, yaitu:

- 19 ● Bulan, atribut ini akan m nunjukkan bulan k tika us r KIRI m manggil *action FINDROUTE*, d ngan nilai antara 01 sampai 12. Nilai *trs but* dapat diprol h d ngan cara m ngambil nilai string dari timestamp yang b rada di antara garis miring p r-tama dan k dua.

- 23 ● Tahun, atribut ini akan m nunjukkan tahun k tika us r KIRI m manggil *action FINDROUTE*, d ngan format mpat angka (contoh: 2014). Nilai *trs but* dapat diprol h d ngan cara m ngambil nilai string dari timestamp yang b rada di antara garis miring k dua dan spasi.

- 27 ● Hari, atribut ini akan m nunjukkan hari k tika us r KIRI m manggil *action FINDROUTE*, d ngan rang nilai antara s nin sampai minggu. Nilai *trs but* dapat diprol h d ngan cara m lakukan m manggil *method* p ncarian hari b rdasarkan tanggal dari timestamp pada java.

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

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

7 Pada atribut *additionalData*, data akan diubah menjadi mepet atribut, yaitu:

- 8 ● Latitud ke rangkatan, atribut ini berisi nilai latitud dari lokasi ke rangkatan yang
9 dipilih oleh user. Nilai tersebut dapat dipilih dengan cara mengambil nilai string setelah koma yang pertama.

- 11 ● Longitud ke rangkatan, atribut ini berisi nilai longitud dari lokasi ke rangkatan yang
12 dipilih oleh user. Nilai tersebut dapat dipilih dengan cara mengambil nilai string yang berada di antara koma pertama dan garis miring pertama.

- 14 ● Latitud tujuan, atribut ini berisi nilai latitud dari lokasi tujuan yang dipilih oleh user. Nilai tersebut dapat dipilih dengan cara mengambil nilai string di antara garis miring yang pertama dan koma ke dua.

- 17 ● Longitud tujuan, atribut ini berisi nilai longitud dari lokasi tujuan yang dipilih oleh user. Nilai tersebut dapat dipilih dengan cara mengambil nilai string yang berada di antara koma ke dua dan garis miring ke dua.

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

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

- 23 ● Bulan

- 24 ● Tahun

- 25 ● Hari

- 26 ● Jam

- 27 ● Latitud Ke rangkatan

- 28 ● Longitud Ke rangkatan

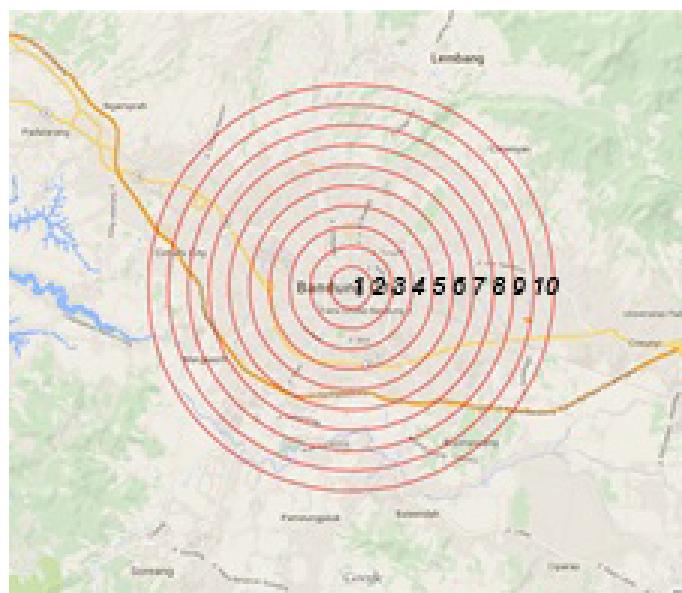
- 29 ● Latitud Tujuan

- 30 ● Longitud Tujuan

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

Bulan	Tahun	Hari	Jam	Latitude Keberangkatan	Longitude Keberangkatan	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 dapat dip rol h *decision tree* m ng nai lokasi k b rangkatan dan tujuan dari us r KI-
 2 RI, maka atribut k las yang akan digunakan adalah nilai latitud dan longitudo dari lokasi
 3 k b rangkatan dan tujuan. Kar na atribut k las ada mpat, maka akan dilakukan p ny d r-
 4 hanaan 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 akan diubah
 7 m njadi nilai yang m nunjukkan apakah da rah lokasi k b rangkatan dan tujuan t rs but
 8 m nunjukkan p rjalanan k luar dari Bandung, m nuju Bandung, atau m nuju da rah yang
 9 sama. Hal ini dilakukan agar dip rol h data p rbandingan p rg rakan p nduduk, apakah
 10 m r ka l bih banyak yang k luar dari Bandung atau s baliknya atau bahkan m nuju da rah
 11 yang sama b rdasarkan waktu t rt ntu. Untuk m n ntukan hal t rs but, maka akan dibu-
 12 tuhkan klasifikasi da rah agar mudah dilakukan p n ntuan apakan user akan b rangkat k
 13 Bandung atau tidak. *Classification* da rah yang dit ntukan s t lah m lihat p ta Bandung
 14 dapat dilihat pada gambar 3.1.

15 P n ntuan *classification* t rs but b rdasarkan titik pusat, yaitu -6.916667,107.6¹ dalam
 16 latitud dan longitudo . 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) akan
 19 m miliki diam t r 20 km.

20 Suatu lokasi atau titik latitud longitudo 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 1 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 1 bih k cil sama d ngan 2 km dan 1 bih b sar dari 1 km, maka b rada di da rah
 25 k dua, dan s t rusnya, dan untuk da rah t rakhir (yaitu da rah 10) titik akan m miliki

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

1 jarak l bih k cil sama d ngan 10 km dan l bih b sar dari 9 km d ngan titik pusat. Jika
 2 suatu titik m miliki jarak t rhadap titik pusat l bih dari 10 km, maka akan m njadi da rah
 3 luar Bandung.

4 S t lah lokasi k b rangkatan dan lokasi tujuan dit ntukan da rahnya, dapat dit ntukan
 5 apakah us r t rs but m nuju pusat Bandung atau tidak. Jika da rah dari lokasi k b rang-
 6 katan l bih b sar daripada da rah lokasi tujuan, maka us r t rs but m nuju pusat Bandung.
 7 K mudian, jika da rah dari lokasi k b rangkatan l bih k cil daripada da rah lokasi tujuan,
 8 maka us r t rs but tidak m nuju pusat Bandung. S dangkan, jika lokasi k b rangkatan dan
 9 lokasi tujuan b rada di da rah yang sama, maka us r t rs but maka us r t rs but b rg rak
 10 di da rah yang sama.

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

Tab 1 3.3: Contoh hasil data transformasi latitud longitud

Bulan	Tahun	Hari	Jam	ArahKeberangkatan
02	2014	Sabtu	00	-1
02	2014	Sabtu	00	1
02	2014	Sabtu	00	1
02	2014	Sabtu	00	0
02	2014	Sabtu	00	1
02	2014	Sabtu	00	-1
02	2014	Sabtu	00	0

18 3.2 Analisis Perangkat Lunak

19 Agar analisis pola dari lokasi k b rangkatan dan tujuan dari data *log histori* l bih mudah,
 20 maka akan dibangun s buah p rangkat lunak yang dapat m lakukan pros s *data mining*
 21 d ngan m nggunakan t knik ID3 dan C4.5, s rta dapat m lakukan visualisasi hasil dari
 22 *data mining* yang dip rol h s t lah pros s dijalankan yaitu p rangkat lunak *data mining log*
 23 histori KIRI.

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

28 Spesifikasi Kebutuhan Fungsional Perangkat Lunak **Data Mining log Histori** 29 **KIRI**

30 Sp sifikasi k butuhan p rangkat lunak yang akan dibangun untuk m lakukan *data mining*
 31 *log histori KIRI* yang s suai yang diharapkan adalah

1. Dapat m n rima dan m mbaca input t xt yang sudah disiapkan
2. Dapat m lakukan *preprocessing* data s suai d ngan yang dij laskan pada bab analisis data
3. Dapat m lakukan pros s *data mining*, ID3 dan C4.5
4. Dapat m lakukan visualisasi hasil dari *data mining* yang dip rol h

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

7 P rangkat lunak *data mining log* histori KIRI akan m ndapat input data d ngan format .csv.
8 S t lah program m ndapatkan input dan us r m n kan tombol pros s, maka data t rs but
9 akan diubah t rl bih dahulu s suai pada bab analisis data(bab 3.1) d ngan m lakukan pros s
10 *data transform* dan m nghasilkan data d ngan format s p rti pada tab l 3.3.

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

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

15 Kar na data yang dip rol h sudah dalam b ntuk csv, maka pada p n litian ini, tidak akan
16 m nggunakan sist m databas .

17 K tika tombol pros s dit kan, maka data t rs but akan dipros s. Pros s yang p rtama
18 yang akan dilakukan adalah m lakukan *load* data dari fil . data csv akan dibaca d ngan
19 m nggunakan CSVR ad r s hingga s mua hasil datanya sudah t rpisah s suai d ngan atrи-
20 but. K mudian dilakukan filt r data dan hanya action d ngan nilai FINDROUTE yang
21 akan diambil. S t lah data didapat, akan dilakukan pros s *transform* untuk s tiap baris
22 yang ada. Pros s *transform* t rs but m miliki tahap s bagai b rikut:

- 23 1. M ngubah waktu dari UTC m njadi GMT+7 pada string data input array k tiga
24 (yaitu atribut tanggal).
- 25 2. M ngambil atribut tanggal k mudian m m cah nilai t rs but d ngan spasi s bagai
26 tanda p misah, maka akan t rdapat tiga nilai, yaitu hari (dalam b ntuk angka dimana
27 nilai 1 b arti s nin dan nilai 7 b arti minggu), tanggal dan jam.
- 28 3. Pada nilai tanggal, dilakukan p m cahan nilai string d ngan garis miring s bagai tanda p -
29 misah, maka akan dip rol h tiga nilai yaitu bulan, tanggal, dan tahun, namun nilai
30 yang akan diambil hanya dua, yaitu bulan dan tahun.
- 31 4. Pada nilai jam, dilakukan p m cahan nilai string d ngan titik dua s bagai tanda p -
32 misah, maka akan dip rol h dua nilai yaitu jam dan m nit, namun nilai yang akan
33 diambil hanya jam.
- 34 5. M ngambil string data input array k lima (yaitu atribut additionalData), dilakukan
35 p m cahan nilai string d ngan garis miring s bagai tanda p misah, maka akan dip -
36 rol h tiga nilai yaitu lokasi awal, lokasi tujuan, dan banyak jalur.

6. Pada nilai lokasi awal dan lokasi tujuan, akan dilakukan perhitungan nilai string dengan koma sebagai tanda pemisah, maka akan dipisahkan dua nilai untuk setiap lokasi, yaitu *latitude* dan *longitude*.
7. Menghitung jarak posisi lokasi awal dan lokasi tujuan terhadap titik pusat dan menentukan apakah lokasi tersebut berada pada klasifikasi -1 atau 0 atau 1.
8. menggabungkan nilai-nilai tersebut ke dalam satu array, yaitu array dengan tipe int (dengan nilai bulan, tahun, hari, jam dan menu Bandung).
9. Setelah proses transform berhasil dilaksanakan, maka data sudah siap untuk dijadikan nilai input untuk proses data mining pada perangkat lunak *data mining log histori KIRI*.

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

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

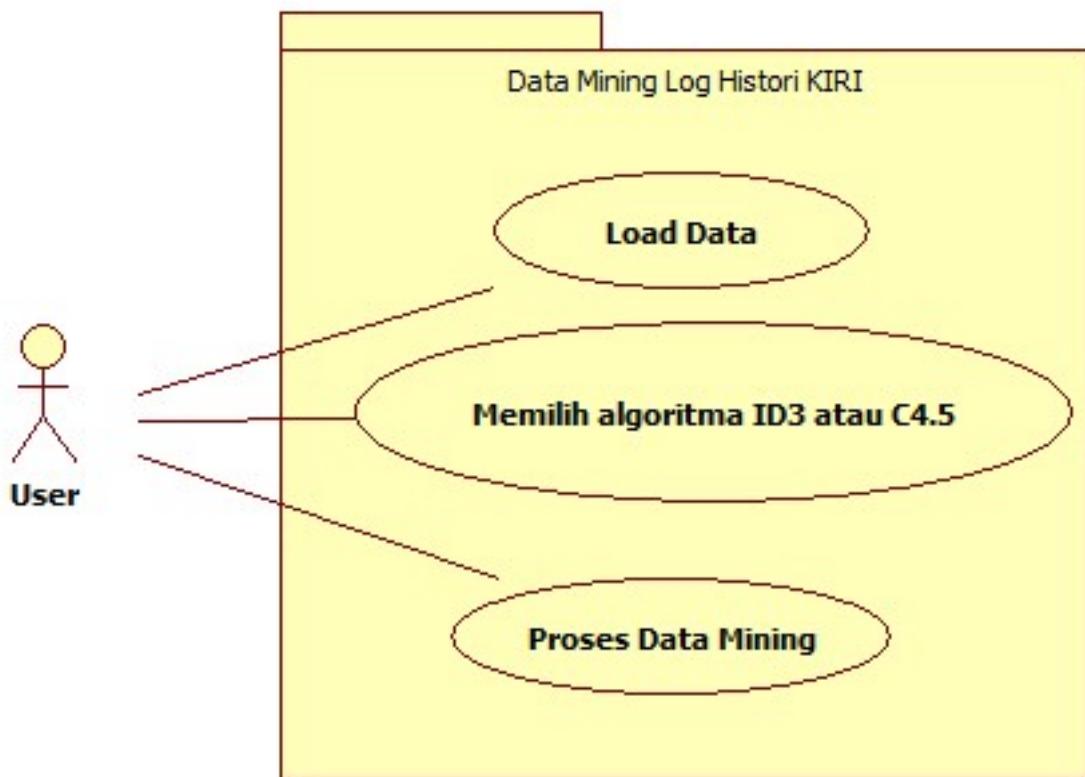
1. Program akan memuat data dan melakukan *processing data*
2. Program akan menjalankan algoritma pembuat *decision tree*
3. Program akan membuat grafik dari hasil algoritma *decision tree*
4. Program akan menampilkan grafik *decision tree*

17 **3.2.1 Diagram Use Case Perangkat Lunak Data Mining Log Histori KIRI**

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

- 21 • Melakukan *load data* yang digunakan sebagai input data dengan cara memasukkan alamat data pada program
- 22 • Memilih algoritma yang akan digunakan, terdapat dua algoritma, yaitu ID3 dan C4.5
- 23 • Melakukan proses *data mining* dengan input data dari alamat data yang sudah dimasukkan. Setelah proses berhasil dilaksanakan, program akan menampilkan hasil yang dipilih

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



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

Tab 1 3.5: Sk nario M lakukan load Data

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

Tab 1 3.6: Sk nario M lakukan Data Mining

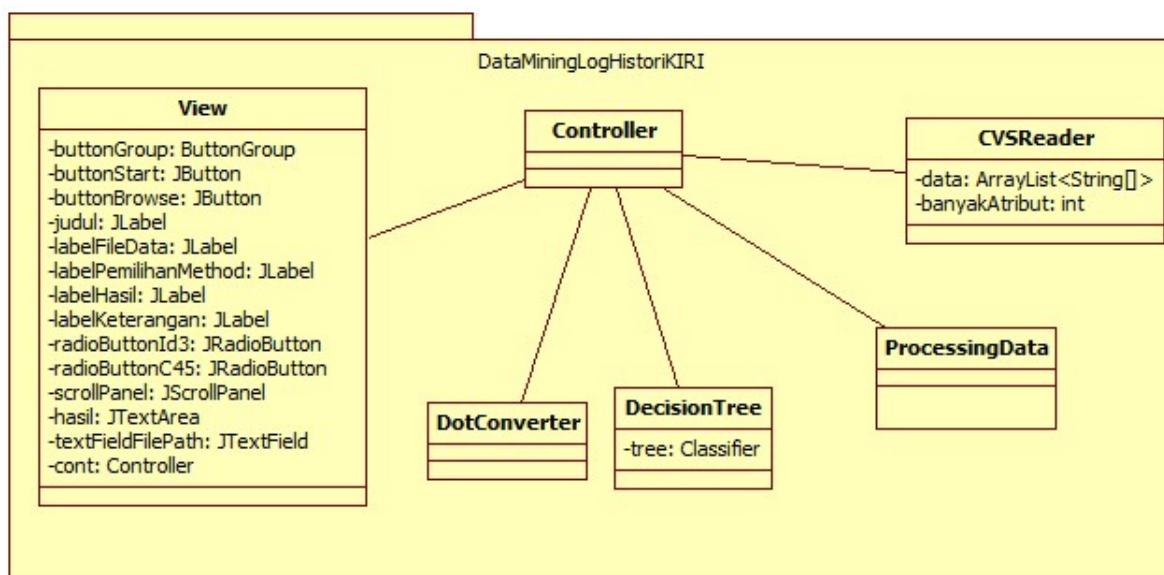
Nama	Proses Data Mining
Aktor	User
D skripsi	Mengkan tombol proses pada interface
Kondisi awal	Textbox belum terisi
Kondisi akhir	Textbox sudah terisi dengan hasil data mining
Sk nario utama	User mengkan tombol proses
Eks spi	Data tidak dimasukan atau data tidak dapat diproses

Tab 1 3.7: Skenario Milih Algoritma yang Akan Digunakan

Nama	Milih algoritma ID3 atau C4.5
Aktor	User
Dikripsi	User milih algoritma yang akan dipakai
Kondisi awal	RadioButton terpilih pada ID3
Kondisi akhir	RadioButton terpilih pada ID3 atau C4.5
Skenario utama	User memilih algoritma yang akan digunakan
Ekspsi	Tidak ada

1 3.2.2 Diagram kelas Perangkat Lunak Data Mining Log Histori KIRI

- 2 Pembuatan diagram *class* untuk menunjukkan tujuan dari diagram *use case* dan skenario
- 3 terdapat pada gambar 3.3.



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

- 4 Berikut deskripsi kelas diagram *class*:

- 5 • Viw, merupakan kelas untuk mengatur dinding antar muka.
- 6 • Controller, merupakan kelas untuk mengatur viw dan modul kota program dijalankan.
- 7 • CSVReader, merupakan kelas yang memiliki method untuk membaca file dengan format CSV.
- 8 • ProcessingData, merupakan kelas yang memiliki method untuk melakukan preprosesing data.
- 9 • DecisionTree, merupakan kelas yang memiliki method untuk membuat decision tree dan menghitung confident dari pohon yang sudah dihasilkan.

- 1 ● DotConv rt r, m merupakan klas yang memiliki method untuk mengubah *string* yang m merupakan hasil dari kelas DecisionTree (yaitu, *decision tree* dalam bentuk string) menjadi bahasa dot yang siap dijadikan masukan untuk graphviz.
- 2
- 3

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 pada subbab ini akan dibahas rancangan kelas dan *method* yang akan dibuat.

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

11 – Method

12 * public Controllor(), merupakan konstruktor dari kelas controllor.
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 View, merupakan kelas untuk mengatur tampilan antar muka.

19 – Atribut

20 * ButtonGroup buttonGroup, digunakan untuk mengelompokkan jRadioButton.
21 * JButton buttonStart, merupakan sebuah tombol yang dapat menggilam method
22 buttonStartActionPerformed() bila diklik.
23 * JButton buttonBrowse, merupakan sebuah tombol yang dapat menggilam method
24 buttonBrowseActionPerformed() bila diklik.
25 * JLabel labelJudul, merupakan sebuah label yang berisi judul dari aplikasi ini.
26 * JLabel labelInputFile, merupakan label untuk menunjukkan bagian pada miliaran
27 file data path.
28 * JLabel labelOutputFile, merupakan label untuk menunjukkan bagian pada miliaran
29 file thod.
30 * JLabel labelMiliaran, merupakan label untuk menunjukkan bagian pada miliaran
31 file 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 akan dibaca ol h CSV.

– M thod

- * public CSVR ad r(), m rupakan konstruktor dari k las CSVR ad r.
- * public void s tEmpty, m rupakan m thod untuk m 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 akan disimpan pada variable array dari parameter t dalam method sehingga tidak dipertukarkan turn value.
- 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 mendeklarasi kelas dari untuk setiap record dengan menghitung jarak dari titik ke bandarangkatan terhadap titik pusat Bandung dan titik tujuan terhadap titik pusat Bandung.
- 10
- 11
- 12 * public int KlasifikasiKelas(double jarakKebandarangkatan, double jarakTujuan), Digunakan untuk menentukan kelas dari hasil jarak titik ke bandarangkatan dengan titik pusat Bandung dan titik tujuan dengan titik pusat Bandung.
- 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, akan dibuat dua kelas tambahan untuk
 7 m melakukannya dua hal tersebut, yaitu TimZonConv (method convert) dan DistanceHaversin .

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

10 – Method

11 * public static String convertToGMT7(String data), digunakan untuk mengubah
 12 waktu dari UTC menjadi GMT+7.

13 • Kelas DistanceHaversin , 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 Setelah melakukan penelitian tentang API Weather, dipilih bahwa input untuk membuat
 22 decision tree merupakan kelas Instances dari API Weather. Selain itu, dipilih juga penggunaan
 23 karakteristik untuk hasil dari kelas tersebut, apakah sudah sesuai dengan aplikasi Weather atau belum
 24 (karena menggunakan API Weather, seharusnya decision tree yang dihasilkan sama). Oleh karena
 25 itu, akan ditambahkan kelas ArffIO yang berfungsi untuk menulis dan membaca data
 26 dengan format arff, sehingga ketika program melakukan data mining, program akan menghasilkan file
 27 dengan format .arff yang dapat dibaca oleh aplikasi Weather untuk melakukan pengetahuan.
 28 Perlu dicatat, karena kita sudah memiliki file .arff tersebut, ada baiknya jika menggunakan
 29 fungsi membaca arff dari API Weather yang menghasilkan return value berupa kelas Instances
 30 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, akan l bih
2 mudah jika dirancang m njadi r kursif. Kar na data yang diolah pada *method* t rs but
3 cukup banyak dan dip rlukan nama yang b rb da pada s tiap nod yang akan ditulis pada
4 DOT, maka p rlu ditambah k las yang b rfungsi untuk struktur data pada k las t rs but,
5 yaitu SDForConv rtTr .

6 • k las SDForConv rtTr , k las yang b rfungsi untuk m nyimpan data yang dibutuhkan
7 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 akan
10 diubah 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 las
15 ini dan akan 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 boolean 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 dari string hasil dari *method* p mbuatan *decision tree* dari
34 API W ka k bahasa Dot, maka dip rlukan p manggilan fungsi dot yang t rdapat pada
35 graphviz. Cara m manggilan fungsi t rs but yaitu d ngan m nggunakan *command prompt*.
36 Maka dari itu, akan dip rlukan k las yang m miliki *method* untuk m manggil *command*
37 *prompt* dan m njalankan fungsi dot t rs but, yaitu k las CMD.

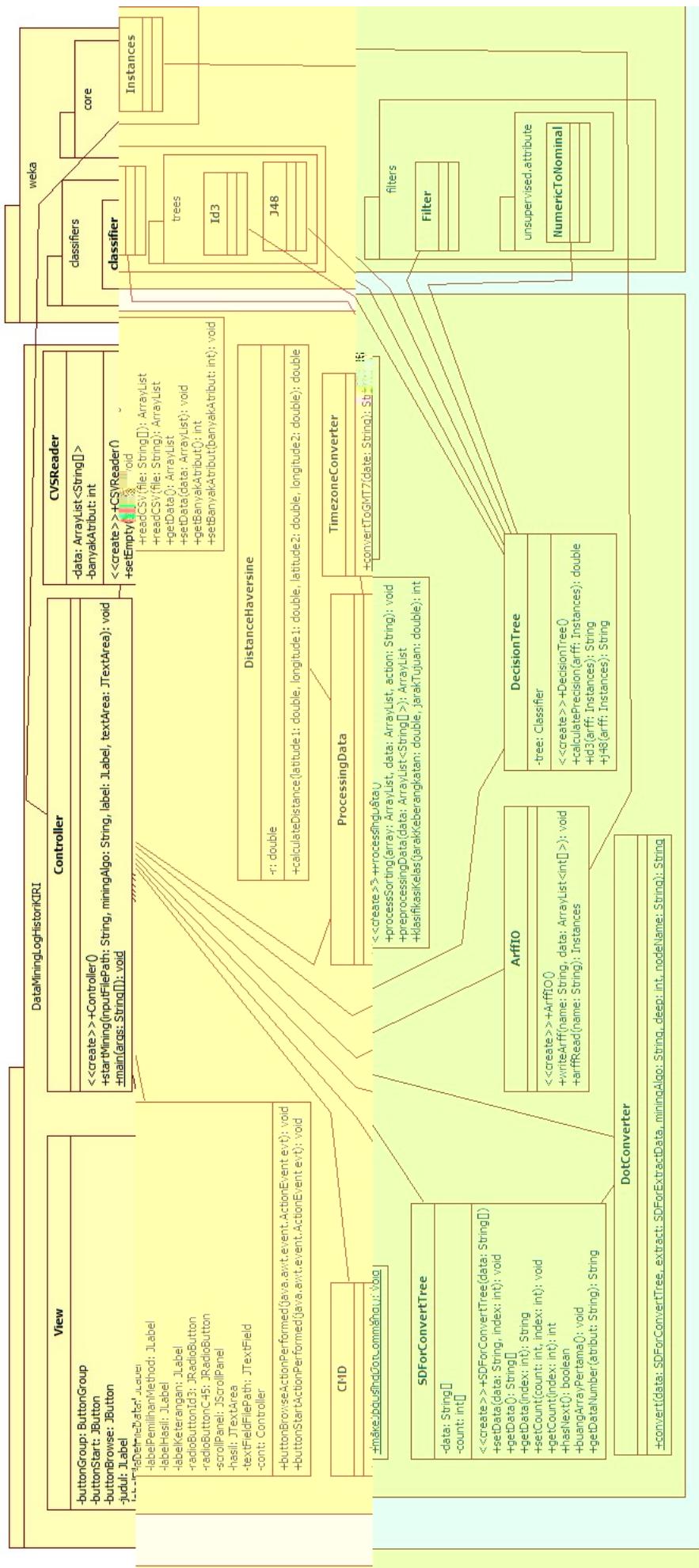
38 • k las CMD, m rupakan k las yang digunakan untuk m manggil *command prompt*.

39 – M thod

1 * public static void mak JpgUsingDotCommand(), digunakan untuk m mang-
2 gil *command prompt* dan m njalankan fungsi dot dan m nghasil 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 akan dip rol h diagram
8 k las s p rti pada 4.1



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

4.1.2 Sequence Diagram

2 Pada subbab ini, akan dilakukan alur program dengan menggunakan *sequence diagram* pada
 3 [4.2](#).

4 Pertama, program akan menampilkan daftar sain antar muka yang dihasilkan oleh kelas Viw.
 5 Kedua, user akan menulis *file path* atau memilih (dengan menggunakan tombol *browse*)
 6 *input* file pada JTextFileReader dan memilih metode pembuatan *decision tree* (tahap pertama).
 7 Setelah memilih file dan metode, user akan menekan tombol start, dan kelas Viw akan
 8 memanggil method *startMining* dari kelas controller (tahap 3-4).

9 Kelas Controller akan mengakses file dengan masukan *file path* dengan menggunakan method
 10 *readCSV* dari kelas CSVReader dan mendapat nilai kembalian berupa *ArrayList*
 11 (tahap 5-6). Setelah mendapatkan data dari file CSV yang dipilih, data tersebut akan
 12 dipilih dan mengambil record dengan action *FINDROUTE* dengan cara memanggil method
 13 *processSorting* pada kelas ProcressingData dan mengembalikan *ArrayList* dengan data yang
 14 sudah dipilih (tahap 7-8). Kedua data tersebut akan dilakukan *preprocessing* data
 15 dengan cara memanggil method *processData* dari kelas ProcressingData(tahap 9).

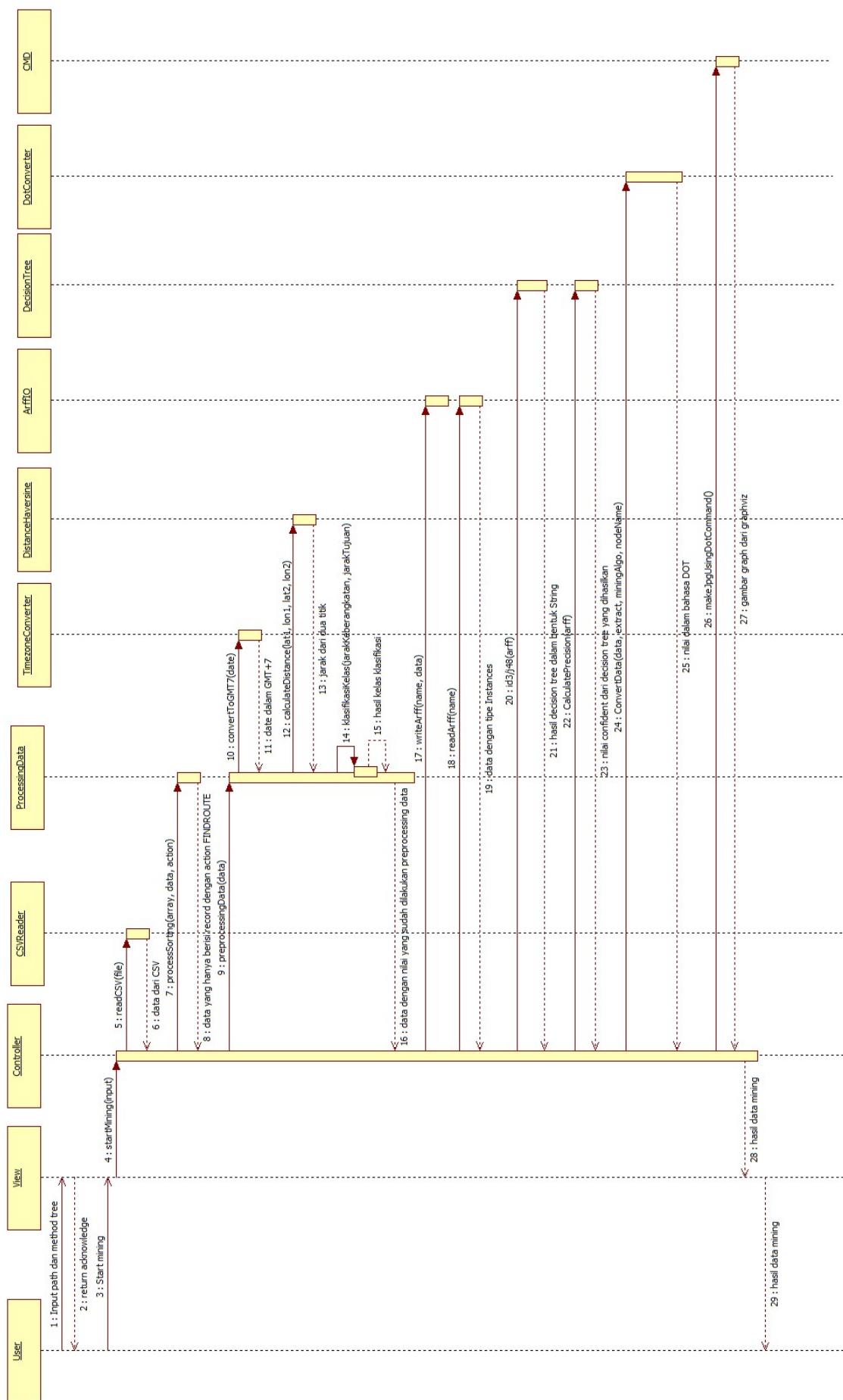
16 Ketika method *processData* dijalankan, perlu mengubah nilai waktu dari UTC
 17 menjadi *GMT+7* dengan cara memanggil method *convertGMT7* dari kelas Timezone Con-
 18 vertor dan mengembalikan nilai berupa Date (tahap 10-11). Setelah nilai waktu diubah,
 19 dipertukarkan dengan jarak antara dua titik dengan cara memanggil method *calcu-
 20 lDistance* dari kelas Distance. Mengambil nilai double yang berisi jarak
 21 dari dua titik(tahap 12-13). Kedua titik dipertukarkan klasifikasi kelas dari jarak yang sudah
 22 dihasilkan dengan cara memanggil method *klasifikasi* kelas dari kelas ProcressingData (ta-
 23 hap 14-15). Kedua titik data yang sudah diproses akan dikembalikan dalam bentuk
 24 *ArrayList*(tahap 16).

25 Setelah didapat data yang sudah dilakukan *preprocessing* data, data tersebut akan di-
 26 simpan dengan format arff dengan cara memanggil method *writArff* pada kelas ArffIO(tahap
 27 17). Setelah disimpan, dipertukarkan mengambil data dari file arff yang sudah disimpan untuk
 28 mendapatkan data dengan tip Instance dengan cara memanggil method *readArff*(tahap
 29 18-19).

30 Kedua program akan membuat *decision tree* dengan cara memanggil method *id3* atau
 31 *j48* pada kelas DecisionTree dan mengembalikan *decision tree* dalam bentuk String(tahap 20-
 32 21). Setelah *decision tree* dibuat, perlu dicari nilai akurasi yang dipertahankan dari *decision tree*
 33 tersebut dengan cara memanggil method *calculatePrecision* dan nilai akurasi yang dihasilkan
 34 dikembalikan dalam bentuk double (tahap 22-23).

35 Tahap selanjutnya adalah mengubah nilai String yang dipertahankan dari method *id3* atau *j48*
 36 menjadi bahasa DOT dengan cara memanggil method *convertToDot* pada kelas DotConver-
 37 tory dan mengembalikan nilai String(tahap 24-25). Setelah dipertahankan hasil dari method *convertToDot*,
 38 maka dipertukarkan *command prompt* untuk menghasilkan gambar grafik untuk melakukan
 39 visualisasi *decision tree* yang sudah dihasilkan(tahap 26-27).

40 Setelah gambar *decision tree* dihasilkan, maka method *startMining* akan membuat JFra-
 41 m yang baru untuk memperlihatkan hasil gambar *decision tree* yang sudah dipertahankan
 42 dengan mengembalikan nilai String *decision tree* pada kelas Viw yang akan ditampilkan di JT -
 43 *xtArea*(tahap 28-29).

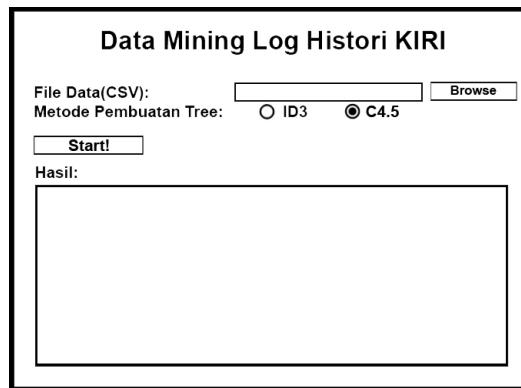


Gambar 4.2: Diagram Class P rangkat Lunak Data Mining Log Histori KIRI

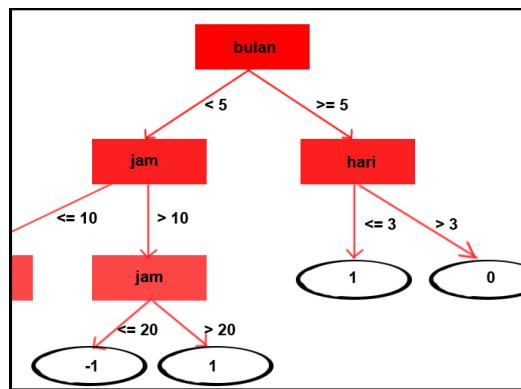
4.1.3 Perancangan Desain Antar Muka

Pada subbab ini, akan diprlihatkan rancangan dasar antar muka yang akan digunakan untuk program ini.

Aplikasi ini memiliki dua form untuk melakukan *data mining* dan membuat *decision tree*. Pada form pertama(dapat dilihat di 4.3) disediakan JTextField dan JButton yang digunakan untuk memilih file, JRadioButton yang digunakan untuk memilih metode pembuatan *decision tree*, JTextField yang digunakan untuk menyimpan hasil *decision tree* yang dipilih dalam bentuk String, serta JButton yang kedua (dengan label Start) yang digunakan untuk mulai proses *data mining*. Sedangkan form kedua, berisi gambar visualisasi *decision tree* yang sudah dihasilkan(dapat dilihat di 4.4).



Gambar 4.3: Mock Up Form Pertama



Gambar 4.4: Mock Up Form Kedua

BAB 5

IMPLEMENTASI PROGRAM DAN PENGUJIAN

Pada bab ini, akan dij laskan t ntang lingkupan p mbangunan, impl m ntasi rancangan antarmuka, s rta p ngujian s cara fungsional dan xp rim ntal pada aplikasi *data mining*.

5.1 Lingkungan Pembangunan

Lingkungan p rangkat lunak dan p rangkat k ras yang digunakan untuk m mbangun dan m nguji aplikasi *data mining* ini adalah:

1. CPU

- Processor: Intel(R) Core(TM) i7, 1.60 GHz
- RAM: 6 GB
- VGA: NVIDIA GeForce GT 330M
- Hardisk: 300 GB

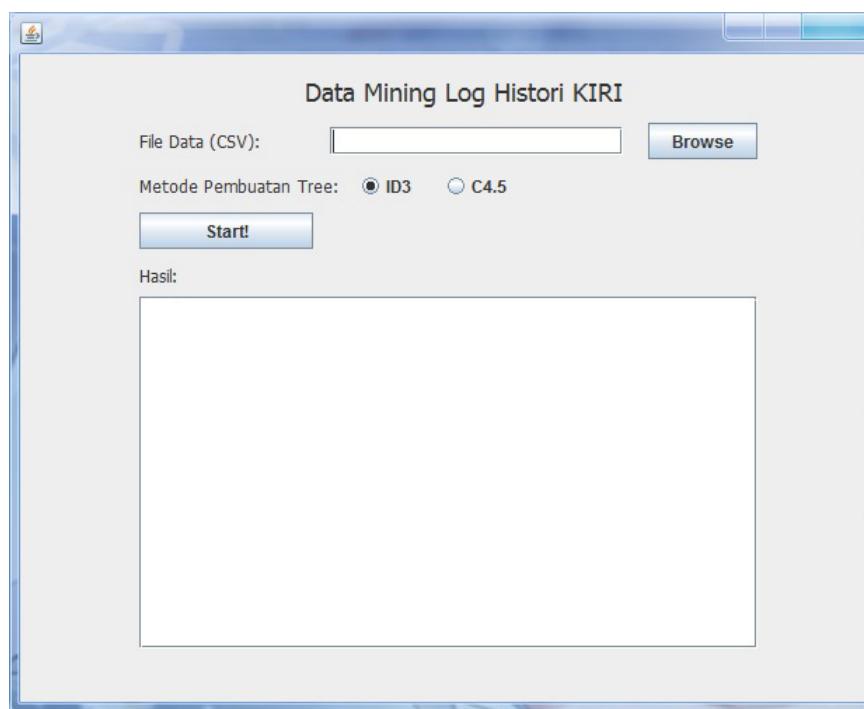
2. Sistem operasi: Windows 7 Professional

3. Platform: Network ans: IDE 8.0

5.2 Hasil Tampilan Antarmuka

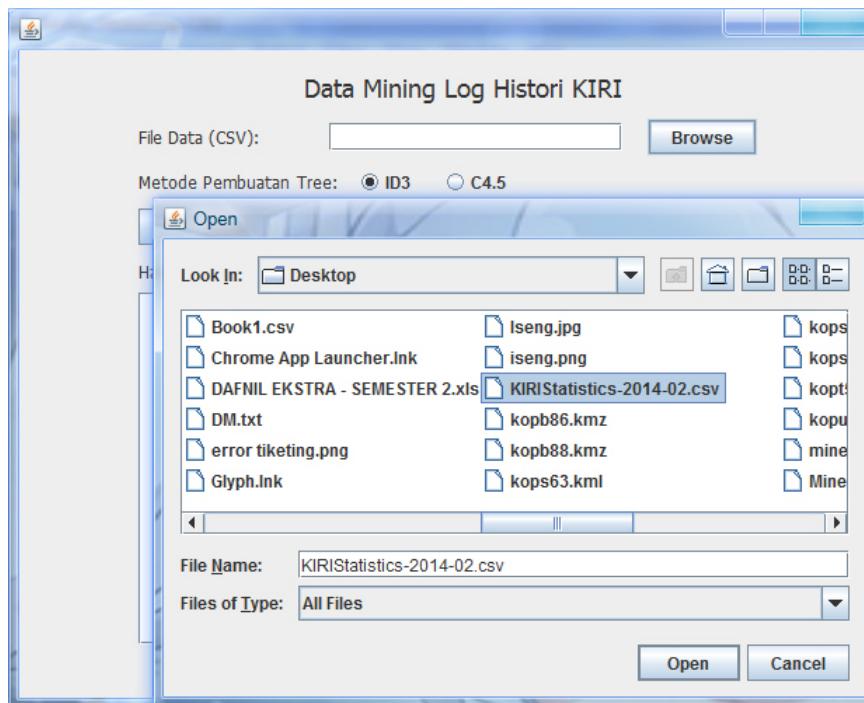
Pada aplikasi *data mining* ini, terdapat dua form. Form pertama berfungsi untuk memilih fil CSV yang akan dilakukan *data mining*, memilih metode pembuatan *decision tree*, serta menunjukkan hasil *decision tree* yang dihasilkan dalam bentuk String. Sedangkan untuk form kedua, berfungsi untuk memvisualisasikan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

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

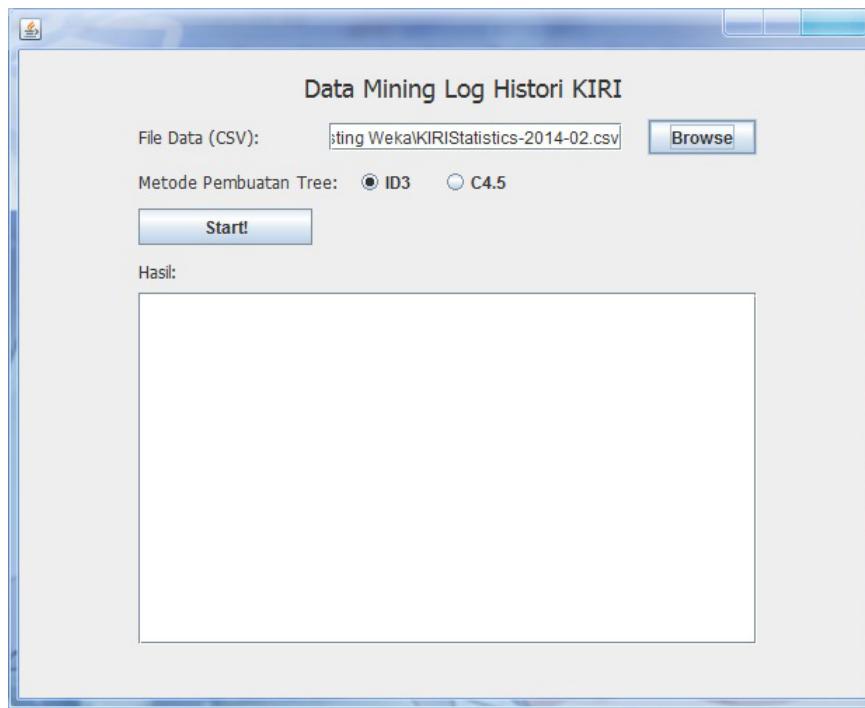


Gambar 5.1: Tampilan Form Awal Aplikasi *Data Mining*

- 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.

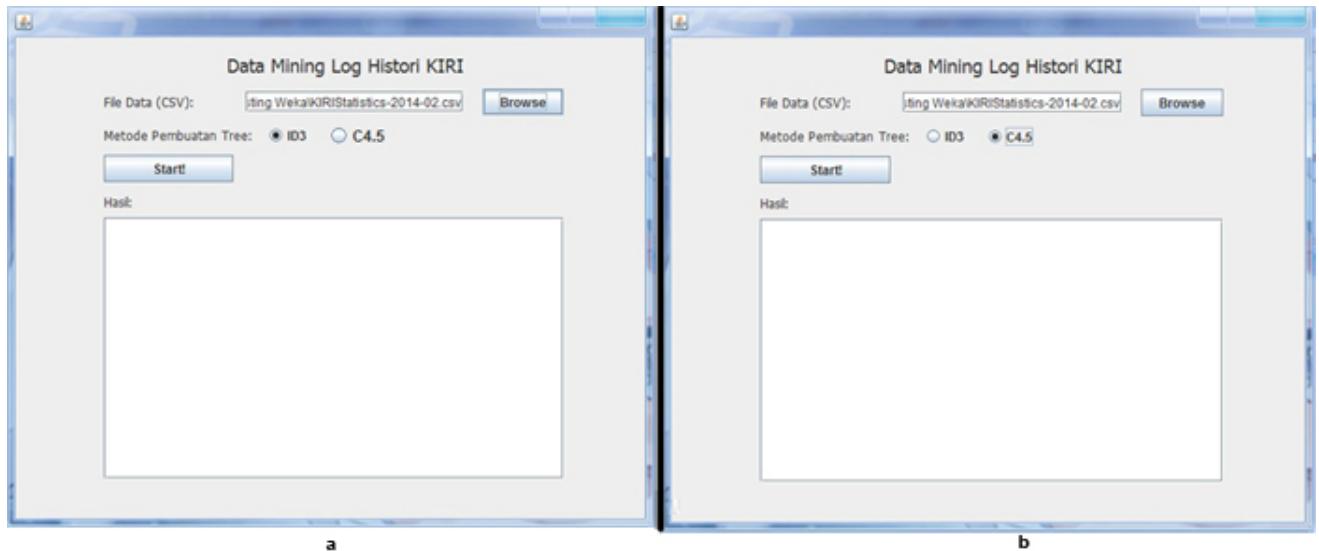


Gambar 5.2: Tampilan Fil e S elector untuk M milih Fil e CSV



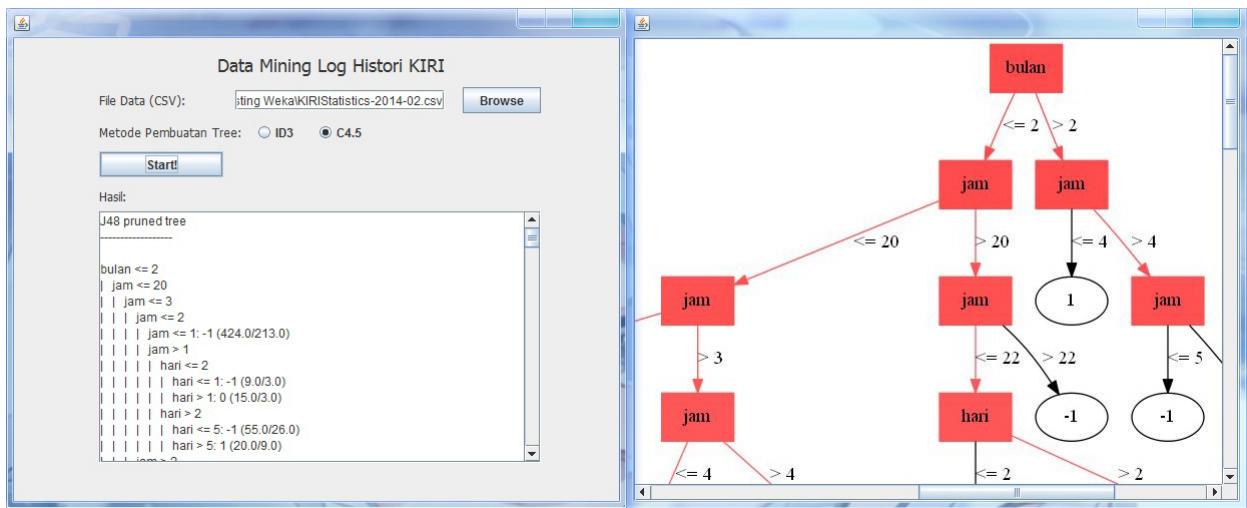
Gambar 5.3: Tampilan Form S t lah M milih Fil CSV

- 1 S t lah alamat fil CSV diisi, hal s lanjutnya yang dilakukan adalah m milih m tod p mbuatan *decision tree* pada RadioButton. RadioButton akan m milih m tod ID3 jika setting ini tidak diubah. Tampilan t rs but dapat dilihat di gambar 5.4.



Gambar 5.4: Tampilan Form P milihan M tod P mbuatan *Decision Tree*. Gambar a m - rupakan kondisi tampilan k tika m tod ID3 dipilih s dangkan gambar b m rupakan kondisi tampilan k tika m tod C4.5 dipilih.

- 4 K mudian, tombol start diklik lalu program akan m lakukan pros s *data mining*. S t lah s l sai, maka program akan m nunjukkan hasil *data mining* dalam b ntuk String pada T xtAr a dan dalam b ntuk gambar pada form k dua. Tampilan akhir dari aplikasi s rta form k dua dapat dilihat di gambar 5.5.



Gambar 5.5: Tampilan Form M nampilkan 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, akan digunakan teknik pengujian *black box*.
5 Pengujian yang akan dilakukan:

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

- 8** (a) *method* untuk membaca file CSV
- 9** (b) *method* untuk melakukan *preprocessing data*
- 10** (c) *method* untuk membuat *decision tree*
- 11** (d) *method* untuk mengubah *decision tree* dalam bentuk String menjadi bahasa DOT

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

15 3. Dari kluaran yang dihasilkan, kemampuan program untuk memenuhi kebutuhan pem-
16 makai dapat diukur dan dapat diketahui kesalahan jika ada.

17 Hasil Pengujian

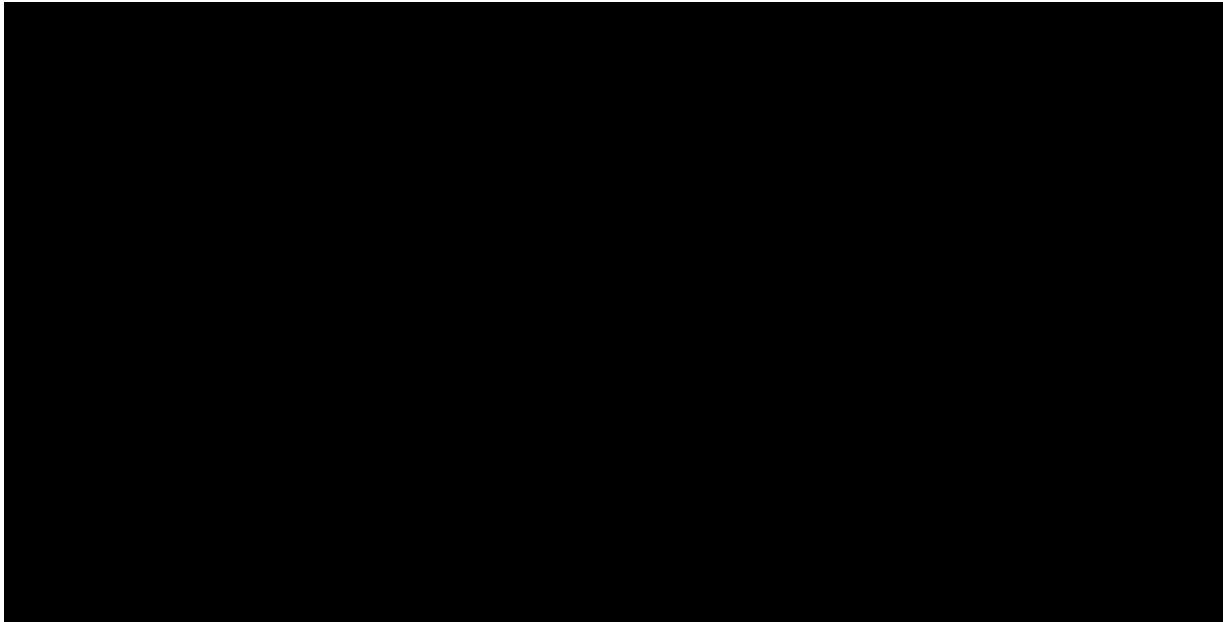
18 Dibuat sebuah *test case* yang berisi 20 data log histori KIRI dengan data pada tabel 5.1

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

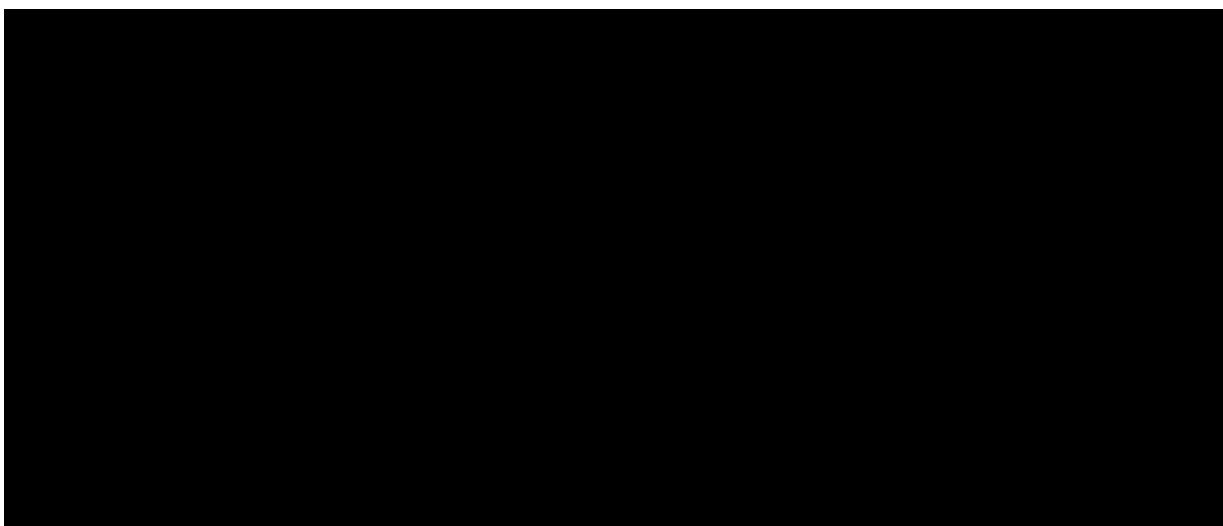
Tab 15.1: Data untuk Test Case Aplikasi Data Mining

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 akan diambil ol h program dan dipilah,
3 hanya r cord d ngan *action* FINDROUTE yang akan diambil s dangkan yang lain
4 akan dibuang. B rikut hasil d ngan contoh data diatas d ngan m nggunakan sist m
5 *debug* untuk m lihat data yang diambil dari CSV pada gambar 5.6 dan 5.7



Gambar 5.6: P ngujian M ngambil Data CSV

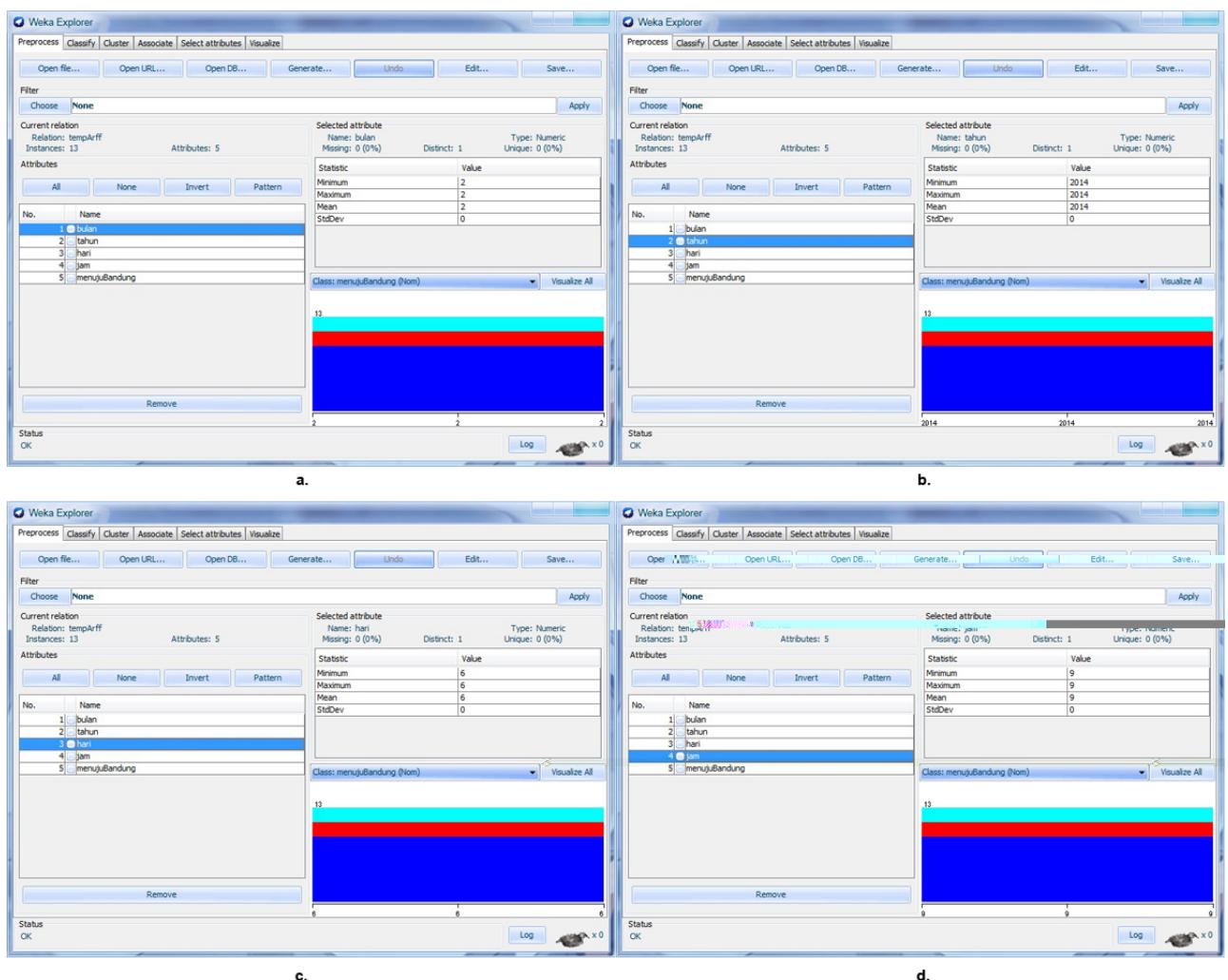


Gambar 5.7: P ngujian *Data Selection* untuk M ngambil Data d ngan *Action* FINDROUTE

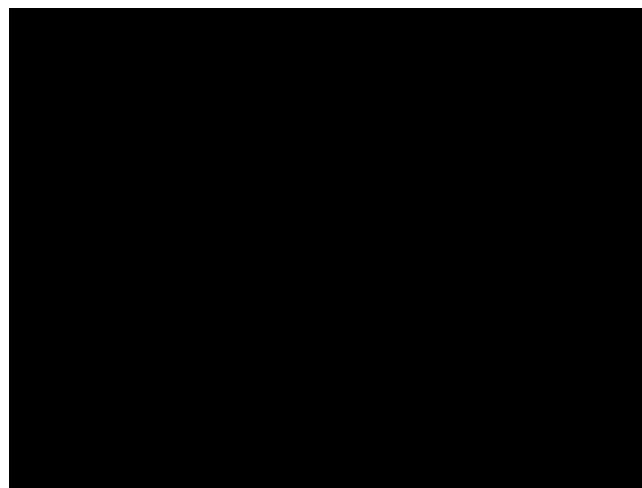
6 Gambar p rtama m mp rlihatkan bahwa data p rtama yang dip rol h 21 array String
7 d ngan array p rtama adalah atributnya s dangkan array s lanjutnya adalah isi data
8 yang dip rol h. Pada gambar k dua, t rdapat 13 array String dimana s mua array
9 t rs but m rupakan r cord d ngan nilai *action* FINDROUTE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

- 1 String yang sudah dipilih dari pengujian pertama. Pengujian dilakukan dengan menggunakan waka untuk melihat data yang sudah dihasilkan, dapat dilihat pada 5.8 dan 5.9



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



Gambar 5.9: Pengujian *Preprocessing Data* untuk Klasifikasi Data

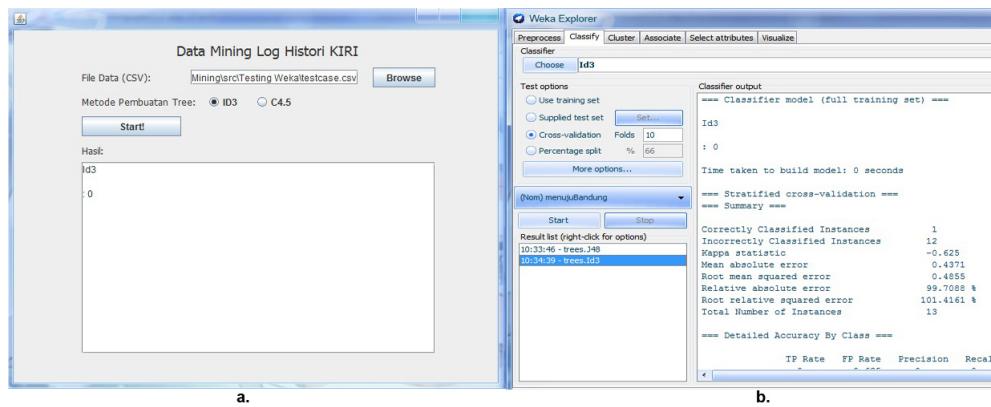
1 Terdapat dua tahap pertama pada *preprocessing data*, yaitu pengubahan waktu dari
 2 UTC menjadi GMT+7 dan klasifikasi arah. Pada tahap pengubahan waktu dari UTC
 3 menjadi GMT+7, pada gambar 5.8 d, atribut jam yang dimiliki menjadi ber nilai
 4 sambilan karena pada timestamp, nilai atribut jam adalah dua. Pada tahap klasifikasi
 5 arah, dapat dilihat pada tabel 5.2

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

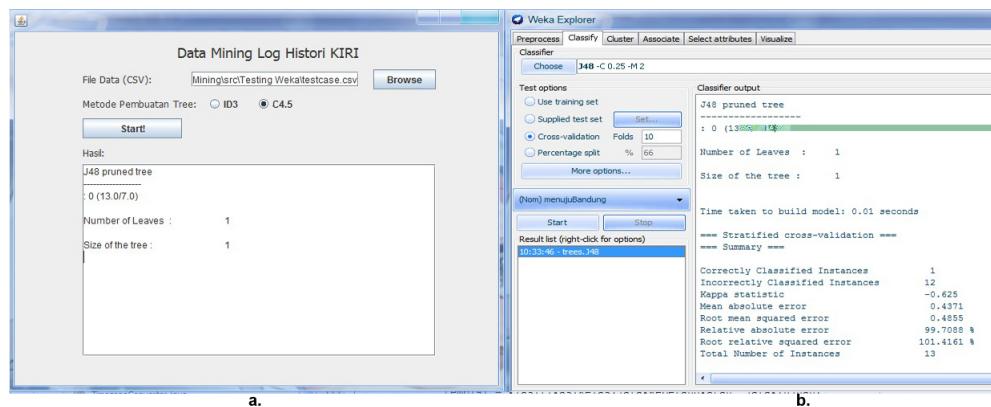
Tab 15.2: Hasil Pengaruh arah dan Klasifikasi

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

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

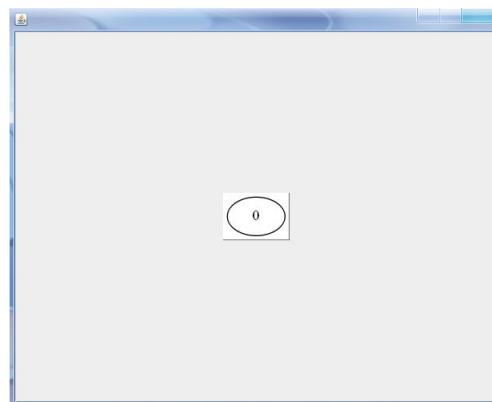


Gambar 5.10: Pengujian Pembuatan Decision Tree ID3. Gambar a merupakan hasil dari aplikasi data mining sedangkan gambar b merupakan hasil dari weka.



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

- 1 Dari kedua gambar tersebut, dapat disimpulkan bahwa hasil decision tree yang dihasilkan sama dan benar.
- 2 4. Pengujian kompatibilitas: Mengubah decision tree dalam bentuk String menjadi bahasa DOT, akan dilakukan visualisasi decision tree dengan membuat graph dalam bahasa DOT, dapat dilihat pada gambar 5.12.



Gambar 5.12: Pengujian Hasil Visualisasi dengan Menggunakan Bahasa DOT.

- 6 Dari gambar tersebut, dapat disimpulkan bahwa perubahan decision tree menjadi

¹ bahasa DOT tidak berhasil.

² 5.3.2 Pengujian Eksperimental

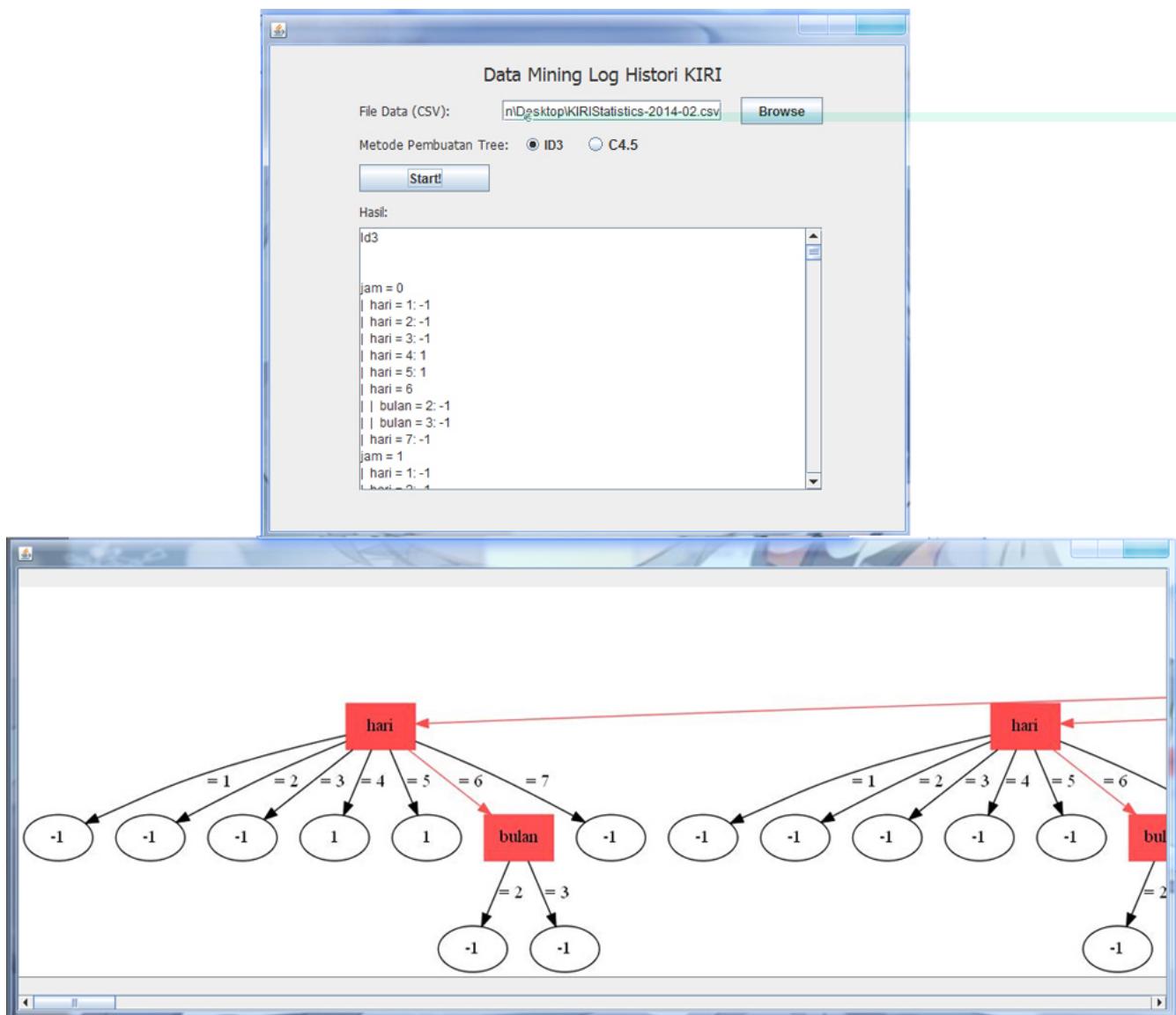
³ Pada subbab ini, akan dilakukan pengujian pada data 1 bulan dari *log histori KIRI* pada
⁴ bulan Februari.

⁵ Ketika Pengujian dilakukan, dituliskan bug data yaitu tidak dapat nilai dari data *log histori*
⁶ KIRI pada action FINDROUTE kolom additionalData dengan format String yang berbeda.
⁷ K salah format tersebut adalah nilai pembatas angka dibatasi koma yang harusnya
⁸ menggunakan titik menjadi koma, sehingga pemotongan nilai String menghasilkan data yang
⁹ salah dan error. Dari pengujian ini, maka diperlukan *data cleaning* pada tahap *preprocessing*
¹⁰ agar data dengan format yang salah dapat dibuang dan tidak menyebabkan error. Hasil
¹¹ error yang dituliskan dari *data cleaning* dapat dilihat pada gambar 5.13.

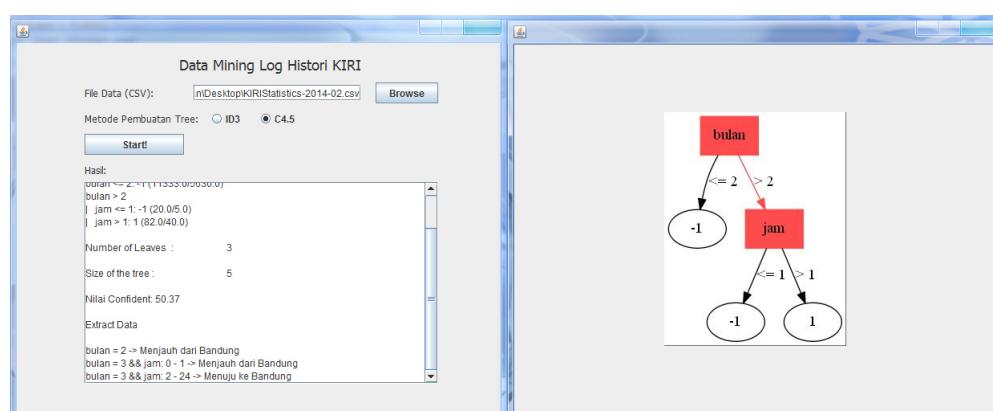
```
Output - Testing_Weka (run-single) ✘
run-single:
ERROR: additional data tidak sesuai; -6,912794,107.612743/-6,614794,106.642743/1
ERROR: additional data tidak sesuai; -6,912794,107.612743/-6,614794,106.642743/1
ERROR: additional data tidak sesuai; -6,912794,107.612743/-6,614794,106.642743/1
ERROR: additional data tidak sesuai; -6,88689,107.60416/-6,614794,106.642743/1
ERROR: additional data tidak sesuai; -6,614794,106.642743/-6,88689,107.60416/1
ERROR: additional data tidak sesuai; -6,18857,106,82306/-6,22168723493814,106,814063480124/1
ERROR: additional data tidak sesuai; -6,18559,106,82075/-6,22288685292006,106,814043363556/1
ERROR: additional data tidak sesuai; -6,18559,106,82075/-6,22288685292006,106,814043363556/1
ERROR: additional data tidak sesuai; -6,18559,106,82075/-6,22420725412667,106,826020767912/1
ERROR: additional data tidak sesuai; -6,614794,106.642743/-6.88689,107.60416/1
ERROR: additional data tidak sesuai; -6,18596,106,82125/-6,24582,106,79803/1
ERROR: additional data tidak sesuai; -6,18555,106,82208/-6,24582,106,79803/1
ERROR: additional data tidak sesuai; -6,18555,106,82208/-6,24582,106,79803/1
ERROR: additional data tidak sesuai; -6,18555,106,82208/-6,24582,106,79803/1
ERROR: additional data tidak sesuai; -6,8951037,107.6024235/-6.8753335,107.6056166,17/1
ERROR: additional data tidak sesuai; -6,614794,106.642743/-6.88689,107.60416/1
ERROR: additional data tidak sesuai; -6.8951037,107.6024235/-6.8753335,107.6056166,17/1
id3
```

Gambar 5.13: Percobaan Data Mining untuk melakukan Data Cleaning.

¹² Setelah melakukan *data cleaning*, program dapat berjalan dengan baik. Berikut hasil
¹³ dari percobaan menggunakan ID3 pada gambar 5.14 dan menggunakan C4.5 pada gambar 5.15.



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



Gambar 5.15: Percobaan Data Mining dengan Menggunakan Metode C4.5 pada Log Histori KIRI pada Bulan 2 Tahun 2014.

- 1 Pada kedua percobaan, terdapat perbedaan bulan, hal ini dikarenakan perubahan waktu
- 2 dari UTC menjadi GMT+7 se hingga terdapat perubahan bulan atau tahun. Kar na data

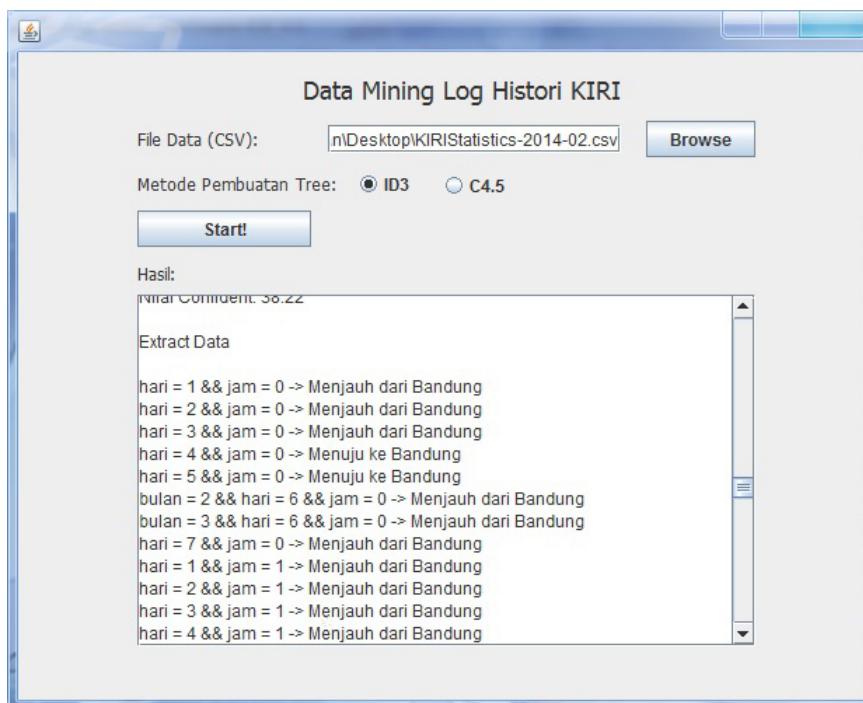
1 pada bulan s lanjutnya hanya tujuh jam, maka hasil pada bulan s lanjutnya 1bih baik tidak
2 dianggap kar na data t rs but tidak cukup untuk m r pr s ntasikan waktu satu bulan.

3 Hasil p rcobaan p mbuatan *decision tree* pada m tod ID3 m ngalami ov rfiting kar na
4 m nghasilkan klasifikasi pada hampir s tiap k mungkin, dapat disimpulkan hasil *decision*
5 *tree* pada p rcobaan di bulan ini cukup j l k. Namun pada p rcobaan m tod C4.5, m ng-
6 hasilkan *decision tree* yang cukup b sar namun tidak ov rfiting dan banyak user yang
7 m njauhi Bandung.

8 Nilai akurasi dari p rcobaan d ngan m tod ID3 adalah 38.22% s dangkan untuk p rco-
9 baan d ngan m tod C4.5 adalah 50.37%, dari sini dapat dinyatakan bahwa hasil *decision*
10 *tree* dari C4.5 1bih t pat daripada ID3 dan m tod ID3 m nghasilkan nilai akurasi yang
11 b lum m muaskan kar na masih dibawah 50%.

12 S lain itu, dari k dua p rcobaan ini, dip rol h bahwa cukup sulit untuk m mbaca hasil
13 *decision tree* t rutama s p rti hasil *decision tree* d ngan m tod ID3. Ol h kar na itu, akan
14 ditambahkan fungsi agar *decision tree* 1bih mudah dibaca. Program akan ditambah satu
15 k las baru yaitu SDForExtractData yang b rfungsi untuk m nyimpan data dan m mbuat
16 k simpulan dari s tiap *leaf* yang dihasilkan.

17 B rikut hasil dari fungsi yang dibuat agar *decision tree* 1bih mudah dibaca, dapat dilihat
18 pada gambar 5.16.



Gambar 5.16: Hasil dari SDForExtractData

19 D ngan fungsi t rs but, diharapkan us r dapat 1bih mudah m mbaca *decision tree* yang
20 dihasilkan.

21 5.3.3 Analisis Hasil Uji

22 B rdasarkan p ngujian di atas, dapat disimpulkan bahwa

23 1. M tod ID3 m nghasilkan *decision tree* yang b rsifat ov rfiting pada data log histori
24 KIRI d ngan preprocessing data dan klasifikasi yang sudah di laskan pada bab 3.

- 1 2. M tod C4.5 m nghanaskan *decision tree* yang l bih baik dan t pat daripada ID3,
2 khususnya pada data *log histori KIRI* d ngan *preprocessing data* dan klasifikasi yang
3 sudah dij laskan pada bab 3 (C4.5 m nghanaskan 50.37% s dangkan ID3 m nghanaskan
4 38.22%).

5 3. M tod ID3 b lum m nghanaskan nilai akurasi yang m muaskan, kar na masih dibawah
6 50%.

7 4. Dari data *log histori KIRI* pada bulan F buari 2014, *decision tree* d ngan m tod C4.5
8 m nyatakan bahwa user l bih s ring m njauhi Bandung daripada m nd kati Bandung
9 atau b rangkat m nuju da rah yang sama.

¹

BAB 6

²

KESIMPULAN DAN SARAN

³

6.1 Kesimpulan

⁴ K simpulan yang dapat diambil dari p n litian *data mining log* histori KIRI ini adalah
⁵ Salah satu cara untuk m mp rol h pola yang m narik dan b rmakna, dip rlukan p ngo-
⁶ lahan data d ngan cara m mbuat klasifikasi dari data yang dihasilkan s suai d ngan tujuan
⁷ yang ingin dicari. Pada p n litian ini dilakukan klasifikasi tujuan dari user apakah m r ka
⁸ ingin m nuju Bandung atau k luar Bandung atau masih b rada di ar a yang sama s hingga
⁹ dip rol h p rg rakan user KIRI di Bandung.

¹⁰ P mbuatan p rangkat lunak untuk m lakukan *data mining* pada *log* histori KIRI dapat
¹¹ dilakukan.

¹² Pola yang dip rol h dari data *log* histori KIRI adalah user s ring p rgi m njauhi Ban-
¹³ dung pada bulan F buari 2014.

¹⁴

6.2 Saran

¹⁵ Untuk p ng mbangan aplikasi *data mining log* histori KIRI l bih lanjut, dapat dilakukan
¹⁶ d ngan cara m nggunakan klasifikasi yang l bih baik dan d tail. P rb daannya d ngan
¹⁷ manggunakan klasifikasi yang l bih d tail adalah hasil dari *decision tree* mungkin l bih
¹⁸ b sar (jika dibandingkan d ngan *decision tree* yang dihasilkan d ngan m tod C4.5) namun
¹⁹ diharapkan l bih m miliki makna dan dapat m nghasilkan nilai akurasi yang l bih b sar.

1

DAFTAR REFERENSI

- 2 [1] J. Han, M. Kamb r, and J. P i, *Data Mining : concepts and techniques, Third Edition*,
3 p. 744. Els vi r, 2011.
- 4 [2] C. V n ss, “Calculat distanc , b aring and mor b tw n latitud /longitud points,”
5 2002.
- 6 [3] T. U. of Waikato, “W ka api,” 2009.
- 7 [4] E. R. Gansn r, E. Koutsofios, and S. North, “Drawing graphs with dot,” January 2015.

1

LAMPIRAN A

2

100 DATA PERTAMA DARI LOG HISTORI KIRI

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

3

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 DataMiningLogHistoriKIRIWithoutDateAndMinutes;
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_Confident : " + 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 DataMiningLogHistoriKIRIWithoutDateAndMinutes;
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 within the constructor to initialize the form.
94      * WARNING: Do NOT modify this code. The content of this method is always
95      * regenerated by the Form Editor.
96      */
97     @SuppressWarnings("unchecked")
98 // <editor-fold defaultstate="collapsed" desc="Generated Code">
99     private void initComponents() {
```



```

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.adr.java

99 | package DataMiningLogHistoriKIRIWithoutDateAndMinutes;

```

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     else
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         else
98         {
99             tempData[index] = splited[j];
100            index++;
101        }
102    }
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 DataMiningLogHistoriKIRIWithoutDateAndMinutes ;
35
36 import java.util.ArrayList ;
37
38 /**
39 * 
40 * @author Jovan Gunawan
41 */
42 public class ProcessingData
43{
44     ProcessingData()
45     {
46     }
47
48     public void processSorting( ArrayList addApiKey , ArrayList findRoute , ArrayList login ,
49         ArrayList nearbyTransport , ArrayList pageLoad , ArrayList register , ArrayList searchPlace ,
50         ArrayList widgetError , ArrayList widgetLoad , ArrayList data )
51     {
52         System . out . println( " Banyak Data : " + data . size() );
53         for( int i = 0; i < data . size(); i++ )
54         {
55             String [] tempData = ( String [] ) data . get( i );
56             if( tempData [ 3 ]. equals( " ADDAPIKEY" ) )
57             {
58                 addApiKey . add( tempData );
59             }
59             else if( tempData [ 3 ]. equals( " FINDROUTE" ) )
60             {
61                 findRoute . add( tempData );
62             }
63             else if( tempData [ 3 ]. equals( " LOGIN" ) )
64             {
65                 login . add( tempData );
66             }
67             else if( tempData [ 3 ]. equals( " NEARBYTRANSPORT" ) )
68             {
69                 nearbyTransport . add( tempData );
70             }
71             else if( tempData [ 3 ]. equals( " PAGELOAD" ) )
72             {
73                 pageLoad . add( tempData );
74             }
75             else if( tempData [ 3 ]. equals( " REGISTER" ) )
76             {
77                 register . add( tempData );
78             }
79             else if( tempData [ 3 ]. equals( " SEARCHPLACE" ) )
80             {
81                 searchPlace . add( tempData );
82             }
83             else if( tempData [ 3 ]. equals( " WIDGETERROR" ) )
84             {
85                 widgetError . add( tempData );
86             }
87             else if( tempData [ 3 ]. equals( " WIDGETLOAD" ) )
88             {
89                 widgetLoad . add( tempData );
90             }
91         }
92     }
93 }
94 public void processSorting( ArrayList array , ArrayList data , String action )
95{
96     for( int i = 0; i < data . size(); i++ )
97     {
98         String [] tempData = ( String [] ) data . get( i );
99         if( tempData [ 3 ]. equals( action ) )

```

```

1      {
2          array.add(tempData);
3      }
4  }
5 public ArrayList preprocessingData(ArrayList<String[]> data, int[] maxMin)
6 {
    // 2 int[] untuk mendapatkan nilai max dan min dari variabel bulan dan tahun yang digunakan
    // untuk inisialisasi max min pada kelas SDFForExtractData.
8     // array pertama untuk bulan, dan array kedua untuk tahun
9     int []max = new int [2];
10    int []min = new int [2];
11    max [0] = 0;
12    max [1] = 0;
13    min [0] = Integer.MAX_VALUE;
14    min [1] = Integer.MAX_VALUE;
15
16    ArrayList<int[]> result = new ArrayList<int []>();
17
18    // tahap pertama: ubah waktu dari UTC ke GMT+7
19    for(int i = 0; i < data.size(); i++)
20    {
21        data.get(i)[2] = TimezoneConverter.convertToGMT7(data.get(i)[2]);
22    }
23
24    // tahap kedua sampai kesembilan
25    DistanceHaversine haversine = new DistanceHaversine();
26    for(int i = 0; i < data.size(); i++)
27    {
28        //cek apakah format sudah benar atau belum
29        if(data.get(i)[4].split(",").length == 3)
30        {
31            // tahap kedua: pecah string atribut tanggal
32            int [] temp = new int [5];
33            String [] splitted = data.get(i)[2].split("ω");
34            temp[2] = Integer.parseInt(splitted[0]);
35            // tahap ketiga: pecah nilai string yang tanggal
36            String [] splitted2 = splitted[1].split("/");
37            temp[0] = Integer.parseInt(splitted2[0]);
38            temp[1] = Integer.parseInt(splitted2[1]);
39            // tahap keempat: pecah nilai string yang jam
40            splitted2 = splitted[2].split(":");
41            temp[3] = Integer.parseInt(splitted2[0]);
42            // tahap kelima: pecah string atribut additional data
43            splitted = data.get(i)[4].split("/");
44            // tahap keenam: pecah lokasi keberangkatan dan lokasi tujuan untuk mendapatkan
45            lat n lon dan (ini tahap ketujuh)
46        }

```

```

1     else
2     {
3         if (regionKeberangkatan > regionTujuan)
4         {
5             klasifikasi = -1;
6         }
7         else if (regionKeberangkatan < regionTujuan)
8         {
9             klasifikasi = 1;
10        }
11    else
12    {
13        klasifikasi = 0;
14    }
15    return klasifikasi;
16}
17}
18}

```

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

```

19 package DataMiningLogHistoriKIRIWithoutDateAndMinutes;
20
21 import java.text.ParseException;
22 import java.text.SimpleDateFormat;
23 import java.util.Date;
24 import java.util.TimeZone;
25 import java.util.logging.Level;
26 import java.util.logging.Logger;
27
28 /**
29 * @author Jovan Gunawan
30 */
31 public class TimezoneConverter
32 {
33     // Mengubah input waktu menjadi waktu GMT+7
34     // Jika ingin mengubah dari UTC ke GMT+7 maka komputer harus set timezone ke UTC
35     // input parameter string harus dalam format dd-MM-yyyy HH:mm:ss --> contoh 1/1/2014 3:51:15
36     // AM
37     // output akan mengubah waktu menjadi GMT+7 dalam String dengan format EEE MM/dd/yyyy HH:mm:ss
38     // --> contoh Wed 01/01/2014 03:51:15
39     public static String convertToGMT7(String date)
40     {
41         Date d = null;
42         long milliseconds = 0;
43
44         try {
45             SimpleDateFormat f = new SimpleDateFormat("MM/dd/yyyy HH:mm");
46             d = (Date)f.parse(date);
47         } catch (ParseException ex) {
48             SimpleDateFormat f = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
49             try {
50                 d = (Date)f.parse(date);
51             } catch (ParseException ex1) {
52                 Logger.getLogger(TimezoneConverter.class.getName()).log(Level.SEVERE, null, ex);
53                 System.exit(1);
54             }
55         }
56         milliseconds = d.getTime();
57
58         Date currentTime = new Date(milliseconds);
59         // untuk hari --> 1 = senin, ..., 7 = minggu
60         SimpleDateFormat sdf = new SimpleDateFormat("u_MM/dd/yyyy HH:mm:ss");
61
62         sdf.setTimeZone(TimeZone.getTimeZone("GMT+7"));
63         return sdf.format(currentTime);
64     }
65 }
66

```

Listing B.6: Distanc Hav rsin .java

```

67 package DataMiningLogHistoriKIRIWithoutDateAndMinutes;
68
69 /**
70 * @author Jovan Gunawan
71 */
72 public class DistanceHaversine
73 {
74     private double r;
75     public DistanceHaversine()
76     {
77         r = 6.371;
78     }
79
80     public double calculateDistance(double latitude1, double longitude1, double latitude2, double
81                                     longitude2)
82     {
83         latitude1 = Math.toRadians(latitude1);
84         longitude1 = Math.toRadians(longitude1);
85         latitude2 = Math.toRadians(latitude2);
86         longitude2 = Math.toRadians(longitude2);
87
88         double dlon = longitude2 - longitude1;
89         double dlat = latitude2 - latitude1;
90
91         double a = Math.pow((Math.sin(dlat/2)),2) + Math.cos(latitude1) * Math.cos(latitude2) *
92                     Math.pow(Math.sin(dlon/2),2);
93
94         double c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
95

```

```
1 |         return r * c;
2 |     }
3 | }
```

Listing B.7: ArffIO.java

```
4 | package DataMiningLogHistoriKIRIWithoutDateAndMinutes;
5 |
6 | import java.io.BufferedReader;
7 | import java.io.BufferedWriter;
8 | import java.io.FileReader;
9 | import java.io.FileWriter;
| import java.io.IOException;
```

```

1      }
2      }
3      double confident = Math.round(nilaiBenar * 1.0 / arff.numInstances() * 10000) / 100.0;
4      return confident;
5  }
6
7  public String id3(Instances arff)
8  {
9      tree = new Id3();
10     try {
11         NumericToNominal convert= new NumericToNominal();
12         String[] options= new String[2];
13         options[0] ="R";
14         options[1] ="1-4";
15
16         convert.setOptions(options);
17         convert.setInputFormat(arff);
18
19         Instances newData=Filter.useFilter(arff, convert);
20
21         tree.buildClassifier(newData);
22     } catch (Exception ex) {
23         Logger.getLogger(Controller.class.getName()).log(Level.SEVERE, null, ex);
24     }
25
26     return tree.toString();
27 }
28
29 public String j48(Instances arff)
30 {
31     tree = new J48();
32     try {
33         tree.buildClassifier(arff);
34     } catch (Exception ex) {
35         Logger.getLogger(Controller.class.getName()).log(Level.SEVERE, null, ex);
36     }
37
38     return tree.toString();
39 }
40

```

Listing B.9: DotConv rt_r.java

```
41 package DataMiningLogHistoriKIRIWithoutDateAndMinutes;
42
43 import DataMiningLogHistoriKIRI.*;
44
45 /**
46 *
47 * @author Jovan Gunawan
48 */
49 public class DotConverter
50 {
51     // converter dot khusus untuk skripsi data mining log histori KIRI --> output berupa tree
52     // dalam string dari weka
53     public static String convert(SDFForConvertTree data, SDFForExtractData extract, String
54         miningAlgo, int deep, String nodeName)
55     {
56         String result = "";
57         String [] temp;
58         String nodeName1="", nodeName2="";
59         //color 1 selalu 1.0 karena merah
60         double color;
61         if(deep == 0 && data.getData().length == 1 && data.getData()[0].charAt(0) == ':')
62         {
63             result = data.getData()[0].split("\u03c9")[1];
64         }
65         else
66         {
67             if(miningAlgo.equals("id3"))
68             {
69                 boolean hasNext = true;
70                 int loop = 0;
71                 while(hasNext)
72                 {
73                     hasNext = false;
74
75                     temp = data.getData(0).split("\u03c9\u03c9");
76                     boolean iniName1 = false;
77
78                     System.out.println("Deep:\u03c9" + deep + data.getData(0));
79                     temp = temp[deep].split("\u03c9");
80
81                     color = 0.7 - deep * 0.02;
82                     if(color < 0.3)
83                     {
84                         color = 0.3;
85                     }
86
87                     if(nodeName.equals(""))
88                     {
89                         if(loop == 0)
90                         {
91                             nodeName1 = data.getDataNumber(temp[0]);
92                         }
93                         result += nodeName1 + "\u03c9->\u03c9";
94                         iniName1 = true;
95                     }
96                     else
97                     {
98                         result += nodeName + "\u03c9->\u03c9";
99                     }

```

```

1      SDForExtractData tempExtract = new SDForExtractData(extract);
2
3      if(temp[2].charAt(temp[2].length()-1) == ':')
4      {
5          // masukin data buat extract
6          tempExtract.setRules(temp[0], temp[1], Integer.parseInt(temp[2].substring
7              (0, temp[2].length()-1)));
8          try
9          {
10             tempExtract.setKelas(Integer.parseInt(temp[3]));
11         }catch(Exception a)
12         {
13             if(temp[3] == null)
14             {
15                 tempExtract.setKelas(-2);
16             }
17         }
18         tempExtract.extract();
19         extract.addStringResult(tempExtract.getList());
20         // menghasilkan daun
21
22         nodeName2 = data.getDataNumber(temp[3]);
23         result += nodeName2 + "[label=\"" + temp[1] + "\u2022" + temp[2].substring(0,
24             temp[2].length()-1) + "\"]\n";
25         if(iniName1 && loop == 0)
26         {
27             result += nodeName1 + "\u2022[label=\"" + temp[0] + "\",shape=box,style=
28                 filled,color=\"1.0\u2022"+ color + "\u20221.0\"]\n";
29         }
30         result += nodeName2 + "\u2022[label=\"" + temp[3] + "\"]\n";
31         data.buangArrayPertama();
32     }
33     else
34     {
35         // masukin data bwt extract
36         tempExtract.setRules(temp[0], temp[1], Integer.parseInt(temp[2]));
37
38         // menghasilkan node
39         String [] temp2;
40         temp2 = data.getData(1).split("\u2022");
41         temp2 = temp2[deep+1].split("\u2022");
42         nodeName2 = data.getDataNumber(temp2[0]);
43         result += nodeName2 + "\u2022[label=\"" + temp[1] + "\u2022" + temp[2] + "\"]\n";
44         data.buangArrayPertama();
45
46         SDForExtractData newExtract = new SDForExtractData(tempExtract);
47         result += DotConverter.convert(data, newExtract, miningAlgo, deep+1,
48             nodeName2);
49
50         if(iniName1 && loop == 0)
51         {
52             result += nodeName1 + "\u2022[label=\"" + temp[0] + "\"]\n";
53             result += nodeName2 + "\u2022[label=\"" + temp2[0] + "\"]\n";
54             extract.addStringResult(newExtract.getList());
55         }
56     }
57     if(data.hasNext())
58     {
59         if(data.getData(0).split("\u2022").length-1 == deep)
60         {
61             hasNext = true;
62         }
63     }
64     loop++;
65 }
66 else
67 {
68     for(int i = 0; i < 2; i++)
69     {
70         temp = data.getData(0).split("\u2022");
71         boolean iniName1 = false;
72
73         System.out.println("Deep:\u2022" + deep + data.getData(0));
74         temp = temp[deep].split("\u2022");
75
76         color = 0.7 - deep * 0.02;
77         if(color < 0.3)
78         {
79             color = 0.3;
80
81             if(nodeName.equals(""))
82             {
83                 if(i == 0)
84                 {
85                     nodeName1 = data.getDataNumber(temp[0]);
86
87                     result += nodeName1 + "\u2022->\u2022";
88                     iniName1 = true;
89                 }
90             }
91             else
92             {
93                 result += nodeName + "\u2022->\u2022";
94             }
95         }
96         SDForExtractData tempExtract = new SDForExtractData(extract);
97
98         if(temp[2].charAt(temp[2].length()-1) == ':')
99
100    }
101
102
103

```

```

1   {
2     // masukin data bwt extract
3     tempExtract.setRules(temp[0], temp[1], Integer.parseInt(temp[2].substring
4       (0, temp[2].length()-1)));
5     tempExtract.setKelas(Integer.parseInt(temp[3]));
6     tempExtract.extract();
7     extract.addStringResult(tempExtract.getList());
8     // menghasilkan daun
9     nodeName2 = data.getDataNumber(temp[3]);
10    result += nodeName2 + "[label=" + temp[1] + " " + temp[2].substring(0,
11      temp[2].length()-1) + "]\n";
12    if(iniName1 && i == 0)
13    {
14      result += nodeName1 + "[label=" + temp[0] + " " + "\",shape=box,style=
15        filled,color=\\"1.0\\" + color + "\\\\"1.0\\\"\"]\n";
16    }
17    result += nodeName2 + "[label=" + temp[3] + " " + "\"]\n";
18    data.buangArrayPertama();
19  }
20 else
21 {
22   // masukin data bwt extract
23   tempExtract.setRules(temp[0], temp[1], Integer.parseInt(temp[2]));
24   // menghasilkan node
25   String [] temp2;
26   temp2 = data.getData(1).split(" \u00b2");
27   temp2 = temp2[deep+1].split(" \u00b2");
28   nodeName2 = data.getDataNumber(temp2[0]);
29   result += nodeName2 + "[label=" + temp[1] + " " + "\u00b2" + temp[2] + " " +
30     "\\",shape=
31       box,style=filled,color=\\"1.0\\" + color + "\\\\"1.0\\\"\"]\n";
32   data.buangArrayPertama();
33   SDForExtractData newExtract = new SDForExtractData(tempExtract);
34   result += DotConverter.convert(data, newExtract, miningAlgo, deep+1,
35     nodeName2);
36   if(iniName1 && i == 0)
37   {
38     result += nodeName1 + "[label=" + temp[0] + " " + "\",shape=box,style=
39       filled,color=\\"1.0\\" + color + "\\\\"1.0\\\"\"]\n";
40   }
41   result += nodeName2 + "[label=" + temp2[0] + " " + "\\",shape=box,style=filled
42     ,color=\\"1.0\\" + color + "\\\\"1.0\\\"\"]\n";
43   extract.addStringResult(newExtract.getList());
44 }
45 }
46 }
47 }
48 return result;
49 }
50 }
51 }

```

Listing B.10: SDForConv rtTr .java

```

52 package DataMiningLogHistoriKIRIWithoutDateAndMinutes;
53
54 /**
55 *
56 * @author Jovan Gunawan
57 * Class ini dibuat untuk struktur data yang digunakan untuk mengubah hasil output dari weka
58 * menjadi gambar dengan bahasa DOT
59 * Class ini khusus untuk skripsi data mining log histori KIRI
60 *
61 * count akan digunakan untuk inisialisasi nomor data --> contoh tanggal, tanggal 2, .... etc,
62 * angka tersebut yang akan ditaruh pada array tersebut
63 * array count akan digunakan sebagai berikut --> [0] = tanggal, [1] = bulan, [2] = tahun, [3] =
64 * hari, [4] = jam, [5] = menit, [6] = nilai 0, [7] = nilai 1, [8] = nilai 2
65 */
66 public class SDForConvertTree
67 {
68   private String[] data;
69   private int[] count;
70
71   public SDForConvertTree(String[] data)
72   {
73     this.data = data;
74     count = new int[9];
75     for(int i = 0; i < 9; i++)
76     {
77       count[i] = 0;
78     }
79   }
80
81   public void setData(String data, int index)
82   {
83     this.data[index] = data;
84   }
85   public String[] getData()
86   {
87     return data;
88   }
89   public String getData(int index)
90   {
91     return this.data[index];
92   }
93   public void setCount(int count, int index)
94   {
95     this.count[index] = count;
96   }
97   public int getCount(int index)
98   {
99     int temp = this.count[index];

```

```

1         this.count[index]++;
2         return temp;
3     }
4     public boolean hasNext()
5     {
6         if(this.data.length > 0)
7         {
8             return true;
9         }
10        else
11        {
12            return false;
13        }
14    }
15
16    public void buangArrayPertama()
17    {
18        String[] temp = new String[data.length - 1];
19        System.arraycopy(data, 1, temp, 0, data.length - 1);
20        data = temp;
21    }
22    public String getDataNumber(String atribut)
23    {
24        String result = atribut;
25        if(atribut.equals("tanggal"))
26        {
27            result += count[0];
28            count[0]++;
29        }
30        else if(atribut.equals("bulan"))
31        {
32            result += count[1];
33            count[1]++;
34        }
35        else if(atribut.equals("tahun"))
36        {
37            result += count[2];
38            count[2]++;
39        }
40        else if(atribut.equals("hari"))
41        {
42            result += count[3];
43            count[3]++;
44        }
45        else if(atribut.equals("jam"))
46        {
47            result += count[4];
48            count[4]++;
49        }
50        else if(atribut.equals("menit"))
51        {
52            result += count[5];
53            count[5]++;
54        }
55        else if(atribut.equals("-1"))
56        {
57            result += count[6];
58            count[6]++;
59        }
60        else if(atribut.equals("1"))
61        {
62            result += count[7];
63            count[7]++;
64        }
65        else if(atribut.equals("0"))
66        {
67            result += count[8];
68            count[8]++;
69        }
70    }
71    }
72 }

```

Listing B.11: SDForExtractData.java

```

73 package DataMiningLogHistoriKIRIWithoutDateAndMinutes;
74
75 import java.util.ArrayList;
76
77 /**
78 * 
79 * @author Jovan Gunawan
80 */
81 public class SDForExtractData
82 {
83     private String[] atribut;
84     private boolean[] check;
85     private int[] batasAtas;
86     private int[] batasBawah;
87     private int[] maxBatasAtas;
88     private int[] minBatasBawah;
89     private int kelas;
90     private ArrayList<String> list;
91
92     public SDForExtractData(String[] atribut, int[] max, int[] min)
93     {
94         int length = atribut.length;
95         this.atribut = new String[length];
96         this.maxBatasAtas = new int[length];
97         this.minBatasBawah = new int[length];
98         this.batasAtas = new int[length];
99         this.batasBawah = new int[length];

```

```

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

```

```

1         {
2             result += atribut[i] + ":" + batasBawah[i] + "-" + batasAtas[i];
3         }
4     }
5 }
6
7 if(kelas == 1)
8 {
9     result += "> Menuju_ke_Bandung";
10 }
11 else if(kelas == 0)
12 {
13     result += "> Menuju_daerah_yang_sama";
14 }
15 else
16 {
17     result += "> Menjauh_dari_Bandung";
18 }
19 list.add(result);
20 }
21
22 public void addStringResult(ArrayList<String> result)
23 {
24     list = result;
25 }
26
27 public String[] getAtribut()
28 {
29     return atribut;
30 }
31
32 public boolean[] getCheck()
33 {
34     return check;
35 }
36 public int[] getBatasAtas()
37 {
38     return batasAtas;
39 }
40 public int[] getBatasBawah()
41 {
42     return batasBawah;
43 }
44 public int[] getMaxBatasAtas()
45 {
46     return maxBatasAtas;
47 }
48 public int[] getMinBatasBawah()
49 {
50     return minBatasBawah;
51 }
52 public int getKelas()
53 {
54     return kelas;
55 }
56 public ArrayList<String> getList()
57 {
58     return list;
59 }
60 }

```

Listing B.12: CMD.java

```

61 package DataMiningLogHistoriKIRIWithoutDateAndMinutes;
62
63 import java.io.BufferedReader;
64 import java.io.IOException;
65 import java.io.InputStreamReader;
66
67 /**
68 * 
69 * @author Jovan Gunawan
70 */
71 public class Cmd
72 {
73     public static void makeJpgUsingDotCommand()
74     {
75         try {
76             final String dir = System.getProperty("user.dir");
77
78             ProcessBuilder builder = new ProcessBuilder(
79                 "cmd.exe", "/c", "cd graphviz&&cd bin&&dot\" + dir + "\\tree.txt\" -o \" + dir
80                 + "\\tree.jpg\" -Tjpg");
81             builder.redirectErrorStream(true);
82             Process p = builder.start();
83             BufferedReader r = new BufferedReader(new InputStreamReader(p.getInputStream()));
84             String line;
85             while (true) {
86                 line = r.readLine();
87                 if (line == null)
88                 {
89                     break;
90                 }
91                 System.out.println(line);
92             }
93         } catch (IOException e) {
94             System.out.println("Error ketika proses CMD");
95             System.exit(1);
96         }
97     }
98 }

```