

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

LEMBAR PENGESAHAN

DATA MINING HISTORI PENCARIAN RUTE ANGKOT

JOVAN GUNAWAN

NPM: 2011730029

Bandung, 24 April 2015

Menyetujui,

Pembimbing Tunggal

Pascal Alfadian, M.Com.

Ketua Tim Penguji

Anggota Tim Penguji

Chandra Wijaya, M.T.

Joanna Helga, M.Sc.

Mengetahui,

Ketua Program Studi

Thomas Anung Basuki, Ph.D.

PERNYATAAN

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

DATA MINING HISTORI PENCARIAN RUTE ANGKOT

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

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

Dinyatakan di Bandung,
Tanggal 24 April 2015

M. T. Rai

Jovan Gunawan
NPM: 2011730029

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*

*Dipersembahkan untuk keluarga, teman-teman serta dosen di
universitas Parahyangan jurusan Ilmu Komputer*

KATA PENGANTAR

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

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

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

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

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

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

Bandung, April 2015

P nulis

DAFTAR ISI

| | |
|--|-------------|
| KATA PENGANTAR | xv |
| DAFTAR ISI | xvii |
| DAFTAR GAMBAR | xix |
| DAFTAR TABEL | xx |
| 1 PENDAHULUAN | 1 |
| 1.1 Latar Belakang | 1 |
| 1.2 Perumusan Masalah | 1 |
| 1.3 Tujuan | 2 |
| 1.4 Batasan Masalah | 2 |
| 1.5 Metodologi Penelitian | 2 |
| 1.6 Sistem Matematika Pembahasan | 2 |
| 2 LANDASAN TEORI | 5 |
| 2.1 Knowledge Discovery | 5 |
| 2.1.1 Data Cleaning | 6 |
| 2.1.2 Data Integration | 7 |
| 2.1.3 Data Selection | 7 |
| 2.1.4 Data Transformation | 8 |
| 2.1.5 Data Mining | 9 |
| 2.1.6 Pattern Evaluation | 18 |
| 2.1.7 Knowledge Presentation | 19 |
| 2.2 Log Histori KIRI | 19 |
| 2.3 Haversine Formula | 20 |
| 2.4 Waka | 21 |
| 2.5 Graphviz | 23 |
| 3 ANALISA | 25 |
| 3.1 Analisis Data | 25 |
| 3.1.1 Data Cleaning | 25 |
| 3.1.2 Data Integration | 25 |
| 3.1.3 Data Selection | 25 |
| 3.1.4 Data Transformation | 26 |
| 3.2 Data Convert Hasil Decision Tree menjadi Bahasa DOT | 30 |
| 3.2.1 Pengambilan dalam node | 31 |
| 3.2.2 Ketentuan untuk membuat edge | 31 |
| 3.3 Analisis Pengaruh Lunak | 31 |
| 3.3.1 Diagram Use Case Pengaruh Lunak Data Mining Log Histori KIRI | 33 |
| 3.3.2 Diagram klas Pengaruh Lunak Data Mining Log Histori KIRI | 35 |

| | |
|--|-----------|
| 4 PERANCANGAN PERANGKAT LUNAK | 37 |
| 4.1 P rancangan P rangkat Lunak | 37 |
| 4.1.1 P rancangan K las | 37 |
| 4.1.2 S qu nc Diagram | 42 |
| 4.1.3 P rancangan D sain Antar Muka | 43 |
| 5 IMPLEMENTASI PROGRAM DAN PENGUJIAN | 47 |
| 5.1 Lingkungan P mbangunan | 47 |
| 5.2 Hasil Tampilan Antarmuka | 47 |
| 5.3 P ngujian Aplikasi <i>Data Mining</i> | 50 |
| 5.3.1 P ngujian Fungsional | 50 |
| 5.3.2 P ngujian Eksp rim ntal | 55 |
| 5.3.3 Analisis Hasil Uji | 58 |
| 6 KESIMPULAN DAN SARAN | 59 |
| 6.1 K simpulan | 59 |
| 6.2 Saran | 59 |
| DAFTAR REFERENSI | 61 |
| A 100 DATA PERTAMA DARI <i>log HISTORI KIRI</i> | 63 |
| B THE SOURCE CODE | 67 |

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 | 17 |
| 2.6 | <i>Decision Tree Pruned</i> | 18 |
| 2.7 | Hasil output Graphviz | 24 |
| 3.1 | <i>Classification</i> pada da rah Bandung | 29 |
| 3.2 | Diagram Use Case P rangkat Lunak <i>Data Mining Log Histori KIRI</i> | 34 |
| 3.3 | Diagram <i>Class</i> P rangkat Lunak <i>Data Mining Log Histori KIRI</i> | 35 |
| 4.3 | Mock Up Form P rtama | 43 |
| 4.4 | Mock Up Form K dua | 43 |
| 4.1 | Diagram <i>Class</i> P rangkat Lunak <i>Data Mining Log Histori KIRI</i> | 44 |
| 4.2 | Sequence Diagram P rangkat Lunak <i>Data Mining Log Histori KIRI</i> | 45 |
| 5.1 | Tampilan Form Awal Aplikasi <i>Data Mining</i> | 48 |
| 5.2 | Tampilan Fil S 1 ctor untuk M milih Fil CSV | 48 |
| 5.3 | Tampilan Form s t lah M milih Fil CSV | 49 |
| 5.4 | Tampilan Form P milihan M tod P mbuatan <i>Decision Tree</i> | 49 |
| 5.5 | Tampilan Form M nampilkan Hasil <i>Data Mining</i> | 50 |
| 5.6 | P ngujian M ngambil Data CSV | 52 |
| 5.7 | P ngujian <i>Data Selection</i> untuk M ngambil Data d ngan Action FINDROUTE | 52 |
| 5.8 | P ngujian <i>Preprocessing Data</i> | 53 |
| 5.9 | P ngujian <i>Preprocessing Data</i> untuk Klasifikasi Data | 54 |
| 5.12 | P ngujian Hasil Visualisasi d ngan M nggunakan Bahasa DOT | 54 |
| 5.10 | P ngujian P mbuatan <i>Decision Tree ID3</i> | 55 |
| 5.11 | P ngujian P mbuatan <i>Decision Tree C4.5</i> | 55 |
| 5.13 | P rcobaan <i>Data Mining</i> untuk M lakukan Data Cl aning | 56 |
| 5.14 | P rcobaan <i>Data Mining</i> d ngan M nggunakan M tod ID3 pada <i>Log Histori KIRI</i> pada Bulan 2 Tahun 2014 | 56 |
| 5.15 | P rcobaan <i>Data Mining</i> d ngan M nggunakan M tod C4.5 pada <i>Log Histori KIRI</i> pada Bulan 2 Tahun 2014 | 57 |
| 5.16 | Hasil dari SDForExtractData | 58 |

DAFTAR TABEL

| | | |
|-----|--|----|
| 2.1 | tabl m ngandung <i>missing value</i> dan <i>noisy data</i> | 6 |
| 2.2 | Contoh training s t | 15 |
| 3.1 | Contoh data <i>log KIRI</i> s t lah <i>data selection</i> | 26 |
| 3.2 | Contoh hasil data transformasi | 28 |
| 3.3 | Contoh hasil data transformasi latitud longitud | 30 |
| 3.5 | Sk nario M lakukan <i>load Data</i> | 34 |
| 3.6 | Sk nario M lakukan <i>Data Mining</i> | 34 |
| 3.7 | Sk nario M milih Algoritma yang Digunakan | 35 |
| 5.1 | Data untuk <i>Test Case</i> Aplikasi <i>Data Mining</i> | 51 |
| 5.2 | Hasil P n tuan ar a dan Klasifikasi | 54 |

BAB 1

PENDAHULUAN

1.1 Latar Belakang

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

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

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

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

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

1.2 Perumusan Masalah

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

- Bagaimana cara mengolah data yang diperoleh dari data log histori KIRI agar pola yang dihasilkan menjadi menarik dan bermakna?
- Bagaimana membuat perangkat lunak untuk melakukan *data mining* pada data log histori?
- Pola apa yang didapat dari data log histori KIRI?

1.3 Tujuan

Penelitian ini bertujuan untuk:

- Mencari pola dan informasi yang menarik dari *log histori* KIRI
- Melakukan *data mining* dari *log histori* KIRI
- Mencari nilai dan menarik kesimpulan dari pola yang diperoleh.

1.4 Batasan Masalah

Batasan masalah untuk penelitian *data mining* ini berupa:

- Penelitian ini dibatasi untuk penanganan *data mining* pada data log KIRI
- Data log yang digunakan untuk *data mining* merupakan log satu bulan dari KIRI

1.5 Metode Penelitian

Berikut adalah metodologi penelitian yang digunakan :

- Melakukan studi literatur tentang algoritma-algoritma yang berkaitan dengan programmesan *data mining*
- Melakukan penelitian *data mining* yang ditopang pada log KIRI
- Merancang dan mengimplementasikan algoritma untuk *data mining*
- Mengimplementasikan pembangkit pola *data mining*
- Melakukan pengujian dan kspesifikasi

1.6 Sistematika Pembahasan

Sistematika pembahasan dalam penelitian ini adalah:

- BAB 1: Pendahuluan, berisi latar belakang dari penelitian ini, rumusan masalah yang timbul, tujuan yang ingin dicapai, ruang lingkup atau batasan masalah dari penelitian ini, serta metodologi penelitian yang akan digunakan dan sistematisasi pembahasan dari penelitian ini.
- BAB 2: Landasan Teori, berisi dasar teori mengenai *data mining*, log histori KIRI, Havrsin Formula, Weka, dan Graphviz.

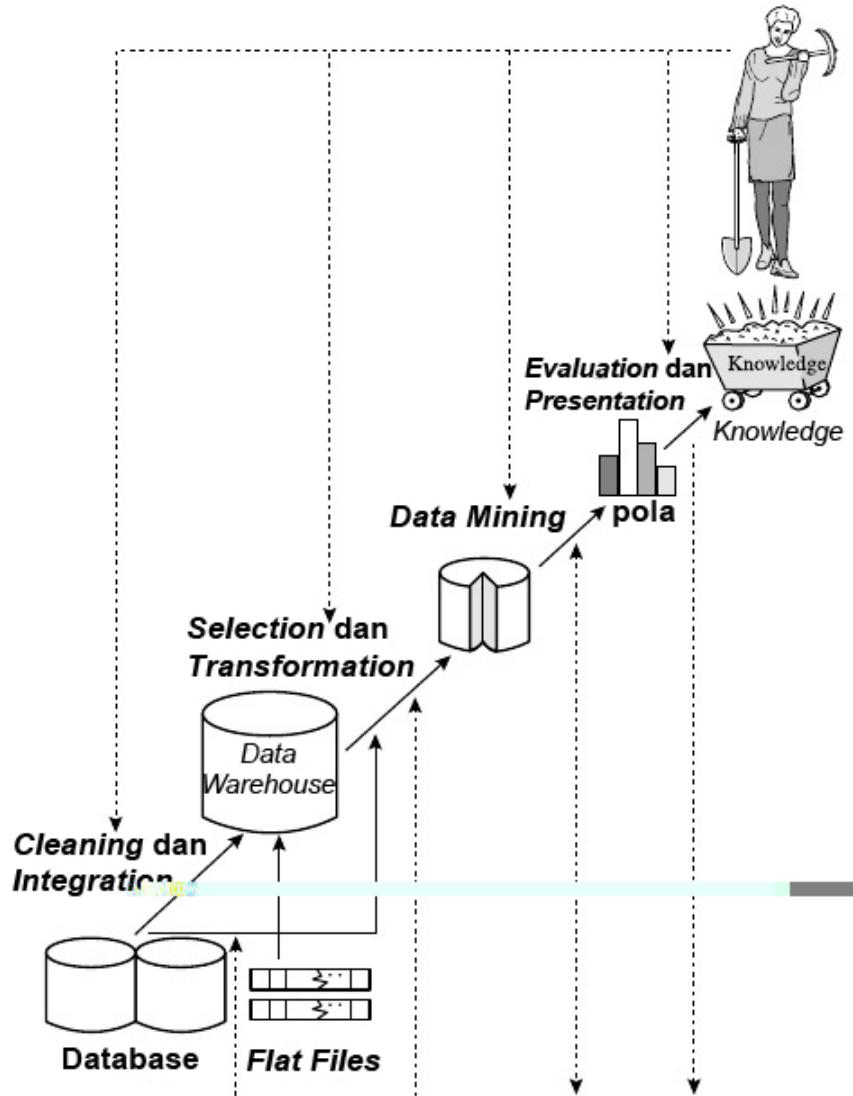
- BAB 3: Berisi analisa dasar teknologi yang akan digunakan, analisa data dan tahap *preprocessing* data yang akan digunakan, serta analisa perancangan aplikasi *data mining log histori KIRI* disertai dengan diagram *use case*, skenario, dan diagram klas.
- BAB 4: Berisi perancangan dari aplikasi *data mining log histori KIRI* yang akan dibangun.
- BAB 5: Berisi implementasi dan pengujian dari aplikasi *data mining*.
- BAB 6: Berisi kesimpulan dan saran dari penelitian *data mining log histori KIRI*.

BAB 2

LANDASAN TEORI

2.1 Knowledge Discovery

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



Gambar 2.1: Tahap Data Mining[1]

M nurut [1], *Knowledge Discovery* dapat dibagi m njadi 7 tahap (gambar 2.1):

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

Tahap p rtama hingga k mpat m rupakan bagian dari *data preprocessing*, dimana data-data disiapkan untuk dilakukan p nggalian data. Tahap *data mining* m rupakan tahap dimana p nggalian data dilakukan. Tahap k nam m rupakan tahap p ncarian pola yang m r pr s ntasikan *knowledge*. S dangkan tahap t rakhir m rupakan visualisasi dan r pr s ntasi dari *knowledge* yang sudah dip rol h dari tahap s b lumnya.

2.1.1 Data Cleaning

Data cleaning m rupakan tahap *Knowledge Discovery* untuk m nghilangkan *missing value* dan *noisy data*. *Missing value* m rupakan nilai yang hilang dari suatu data. *Noisy data* m rupakan nilai yang tidak s suai atau tidak valid.

Pada umumnya, *data* yang dip rol h dari *database* t rdapat nilai yang tidak s mpurna s p rti nilai yang hilang, nilai yang tidak valid atau salah k tik. Atribut dari suatu *database* yang tidak r l van atau r dudansi bisa diatasi d ngan m nghapus atribut atau tupl t rs but. Contoh data yang m miliki *missing value* dan *noisy data* dapat dilihat pada tabl 2.1

Tab 1 2.1: tabl m ngandung *missing value* dan *noisy data*

| IdP njualan | NamaBarang | Custom r | Harga | BanyakBarang |
|-------------|------------|----------|--------|--------------|
| 1 | Mous | Elvin | 45000 | 2 |
| 2 | K yboard | All ria | -35000 | 1 |
| 3 | Monitor | | 225000 | 1 |

Dapat dilihat, pada idP njualan 2, harga dari k yboard adalah -35000, itu m rupakan *noisy data* kar na tidak mungkin nilai harga suatu barang dibawah 0. Pada idP njualan 3, kolom *customer* tidak m miliki nilai, dan itu m rupakan *missing value*.

Missing Values

Missing values akan m nganggu pros s *data mining* pada komput r dan dapat m nghanilkan nilai akhir yang tidak s suai d ngan fakta. T rdapat b b rapa t knik untuk m ngatasi *missing values* yaitu:

- M mbuang tupl yang m ngandung nilai yang hilang

- Mengisi nilai yang hilang dengan cara manual
- Mengisi nilai yang hilang dengan menggunakan nilai konstan yang bersifat umum
- Menggunakan nilai rata-rata dari suatu atribut untuk mengisi nilai yang hilang

Noisy Data

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

- *Binning*, merupakan metode pengisian data dengan proses yang dilakukan pada data tersebut
- *Regression*, merupakan metode yang mencari distribusi yang samaan atribut untuk memprediksi suatu nilai
- *Clustering*, merupakan metode pengelompokan dimana pertemuan pada nilai yang dapat dibuang. Pencarian merupakan data yang berada diluar kumpulan yang sesuai dengan observasi atau penilitian.

2.1.2 Data Integration

Data integration merupakan tahap penggabungan data dari berbagai sumber. Sumber tersebut bisa termasuk berbagai database, data cubes, atau flat data. Data cube merupakan teknik pengambilan data-data dari data warehouse dan dilakukan operasi agar hasilnya sesuai dengan kondisi tertentu (contoh, penjumlahan total dari penjualan per tahun dari 2005-2010). Sedangkan flat data merupakan data yang disimpan dengan cara apapun untuk mempermudah pengetahuan model database pada sebuah data baik berbentuk plain text file maupun binary file.

Tahap ini harus dilakukan dengan cara teliti terutama dalam menambahkan nilai-nilai yang berdasarkan dari sumber yang berbeda. Pada tahap ini, perlu dilakukan identifikasi apakah data tersebut harus dimasukkan atau tidak agar data yang dipilih tidak terlalu besar. Data integration yang baik merupakan integrasi yang dapat memaksimalkan keakuratan dan meningkatkan akurasi proses data mining. Contoh studi kasus dari data integration, misalnya sebuah 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. Karena digabung, harus memperhatikan dan memperbaiki nilai-nilai seperti primary key, atribut. Hal ini dilakukan untuk menghindari kesalahan seperti nilai id yang berbeda padahal merupakan objek yang sama. Proses penggabungan hingga perbaikan nilai-nilai pada kedua database tersebut adalah proses data integration.

2.1.3 Data Selection

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

- NPM Mahasiswa

- NamaMahasiswa
- J_nisK_lamin
- Alamat
- MataKuliah
- NilaiART
- NilaiUTS
- NilaiUAS

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.

2.1.4 Data Transformation

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

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

Smoothing

Smoothing merupakan 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*.

Aggregation

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 dengan rentang hari tertentu.

Generalization

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

Normalization

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

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

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

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

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

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

Contoh kasus, misal nilai rata-rata dan standar deviasi dari nilai-nilai atribut pendapatan adalah 54.000 dan 16.000. Dengan *z-score*, jika ada nilai pendapatan baru yaitu 73.600, maka akan diubah menjadi

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

Attribute Construction

Attribute Construction merupakan teknik menambahkan atribut baru yang berdasarkan atribut yang sudah ada. Contoh kasus, dibuat atribut baru bernama *average* berdasarkan atribut panjang dan lebar.

2.1.5 Data Mining

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

Classification and Prediction

Classification merupakan model yang dibangun untuk memprediksi klasifikasi lab 1 kata gori. Contoh lab 1 kata gori adalah "baik", "cukup", dan "buruk" dalam sistem peringkat sikap seseorang siswa atau "mini bus", "bus", atau "s dan" dalam kategori tip mobil. Kategori dapat diperintahkan dengan menggunakan nilai diskrit. Nilai diskrit merupakan nilai yang terpisah dan berbeda. Contoh dari nilai diskrit adalah 1 atau 5. Kategori yang diperintahkan oleh nilai diskrit maka akan menjadi nilai yang terurut dan tidak memiliki arti. Contoh kategori yang diperintahkan oleh nilai diskrit adalah 1,2,3. Angka tersebut dapat digunakan untuk memprediksi suatu kategori, misalnya untuk tip mobil:

- Angka 1 adalah "mini bus"
- Angka 2 adalah "bus"
- Angka 3 adalah "s dan"

Prediction merupakan model yang dibangun untuk memramalkan fungsi nilai kontinu. Nilai kontinu merupakan nilai yang terurut dan berlanjut. Contoh kasus untuk prediksi, misalkan seorang maketing ingin memramalkan seberapa banyak konsumen yang akan belanja di sebuah toko dalam waktu satu bulan. Model lanjutannya disebut *predictor*. *Regression Analysis* merupakan metodologi statistik yang digunakan untuk *numeric prediction*. *Classification* dan *numeric prediction* merupakan dua fungsi utama pada *prediction*.

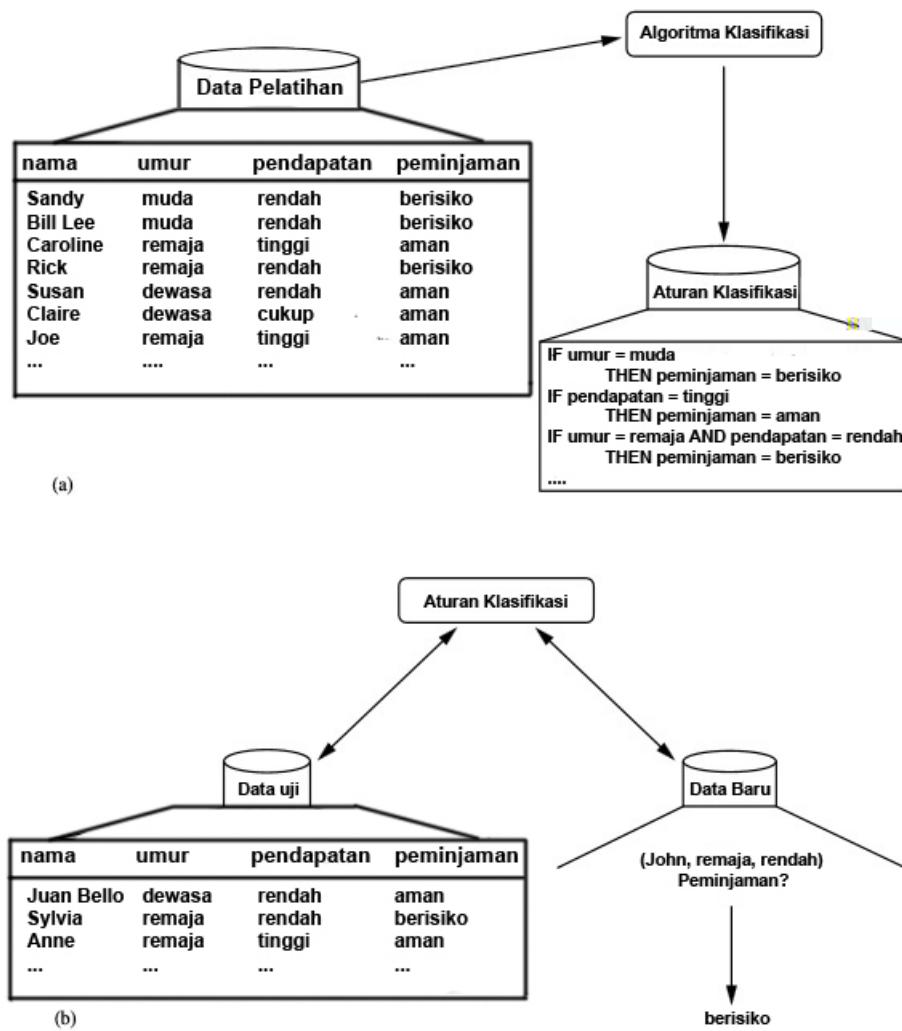
Data *Classification* merupakan proses untuk melakukan klasifikasi lab 1 kata gori. *Data classification* memiliki dua tahap proses, yaitu *learning step* dan tahap klasifikasi. *Learning step* merupakan langkah pembelajaran, dimana algoritma klasifikasi membangun *classification rules* (yang berisi syarat atau aturan sebuah nilai masuk ke dalam kategori tertentu) dengan cara menganalisis *training set* yang merupakan *database tuple*. Karakter pembuatan *classification rules* menggunakan *training set*, yang diketahui juga sebagai *supervised learning*. Pada tahap kedua, dilakukan proses klasifikasi nilai berdasarkan *classification rules* yang sudah dibangun dari tahap pertama.

Contoh kasus *data classification* dapat dilihat pada ilustrasi di gambar 2.2. Pada gambar a, data pelatihan akan diproses oleh algoritma klasifikasi dan menghasilkan aturan klasifikasi. Aturan tersebut akan digunakan untuk menentukan lab 1 kata gori. Pada gambar b, data uji akan diproses oleh aturan klasifikasi yang sudah dipilih dari gambar a dan akan menghasilkan hasil prediksi bahwa suatu tuple berisiko akan pinjam atau tidak.

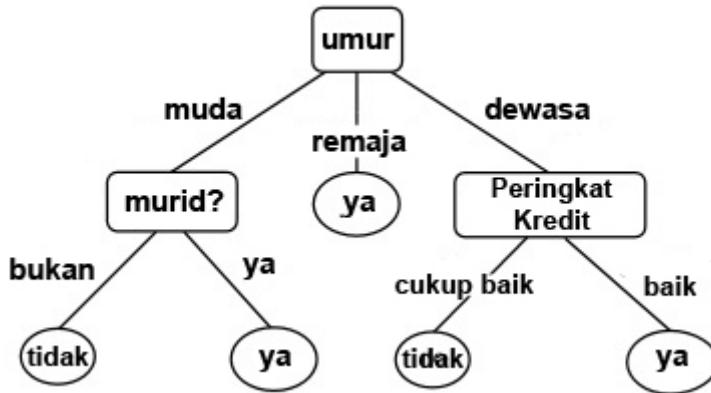
Decision Tree

Salah satu cara pembuatan *classification rules* pada *Data Classification* adalah membangun *decision tree* (pohon klasifikasi). *Decision tree* merupakan *flowchart* yang berfungsi sebagai pohon, dimana setiap node internal (*nonleaf node*) merupakan hasil test dari atribut, sedangkan cabang memprediksi output dari test, dan setiap node daun memiliki *class label*. Bagian paling atas dari pohon disebut *root node*.

Contoh kasus, pohon klasifikasi untuk menentukan apakah seorang konsumen akan membeli komputer atau tidak (ilustrasi pohon klasifikasi pada gambar 2.3). Root node dari pohon klasifikasi



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



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

adalah umur. Atribut pertama yang akan dipertimbangkan adalah atribut umur. Jika nilai atribut umur dari suatu tuple adalah muda, maka akan dipertimbangkan atribut murid apakah bernilai bukan atau ya. Jika nilai atribut murid adalah bukan, maka tuple tersebut belum memiliki label tidak memungkinkan untuk dianalisis jika bernilai ya, maka tuple tersebut akan memiliki label komputer. Jika nilai atribut umur adalah remaja, maka tuple tersebut belum memiliki label belum memungkinkan untuk dianalisis. Jika atribut umur bernilai dewasa maka akan dipertimbangkan atribut peringkat kredit, apakah cukup baik atau baik. Jika atribut peringkat kredit bernilai cukup baik maka tuple tersebut belum memiliki label tidak memungkinkan untuk dianalisis jika bernilai baik, maka tuple tersebut akan memiliki label komputer.

Decision Tree Induction merupakan pertama kali pohon klasifikasi dari tuple pada latihan yang memiliki label klasifikasi. Algoritma yang dipelajari secara umum sama, hanya berbeda pada *attribute_selection_method*. Berikut algoritma untuk membuat pohon klasifikasi dari suatu tuple pada latihan:

Require: Partisi data, D, merupakan set data pada latihan dan klasifikasi

Require: *attribute_list*, merupakan set dari atribut kandidat

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

Ensure: Pohon klasifikasi

- 1: Membuat node N;
- 2: **if** tuple pada D merupakan klasifikasi yang sama, C **then**
- 3: **return** N sebagai node daun dengan label klasifikasi C;
- 4: **end if**
- 5: **if** *atribut_list* tidak ada nilai atau kosong **then**
- 6: **return** N sebagai node daun dengan label yang terpaling banyak pada D; {majority voting}
- 7: **end if**
- 8: memanggil method *Attribut_selection_method*(D, *atribut_list*) untuk mencari nilai terbaik *splitting_criterion*;
- 9: membagikan node N dengan *splitting_criterion*;

```

10: if splitting_attribut m merupakan nilai diskrit dan multiway splits diizinkan then
11:   attribut_list ← attribut_list - splitting_attribut ; {m menghapus splitting_attribut}
12: end if
13: for all hasil j dari splitting_criterion do
14:   Dj m merupakan himpunan data tuple D yang sesuai dengan j;
15:   if Dj tidak ada nilai atau kosong then
16:     m lampirkan daun yang dibelabilding pada klas mayoritas di D ke nod N;
17:   else
18:     m lampirkan nod yang dikembalikan oleh fungsi decision_tree(Dj, attribut_list) ke nod N;
19:   end if
20: end for
21: return N;

```

Pohon keputusan akan dimulai dengan satu node, yaitu N, yang merupakan representasi tuple pada D (baris 1)

Jika tuple di D memiliki kelas yang sama semuanya, maka node N akan menjadi daun dan dibelabilding dari kelas tersebut (baris 2 sampai 4).

Jika tuple di D memiliki kelas yang berbeda, maka algoritma akan memanggil *attribute_selection_method* untuk menentukan *splitting criterion*. *Splitting criterion* akan menentukan atribut pada node N. (baris 8)

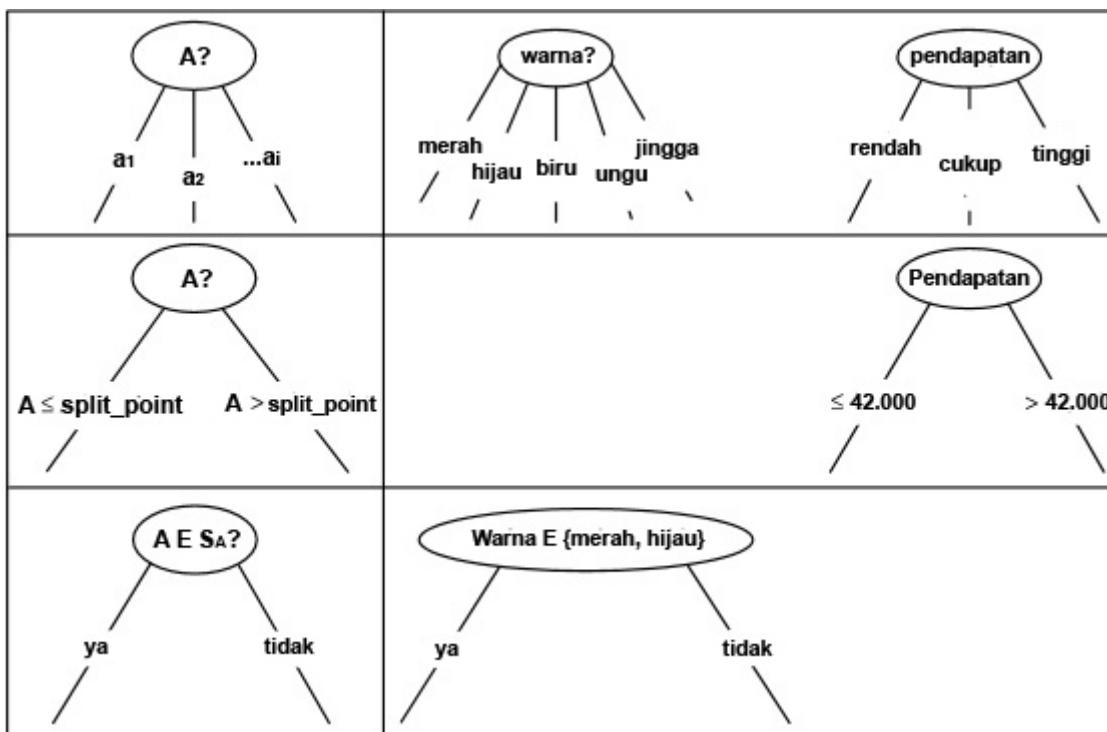
Node N akan diisi dengan hasil dari *splitting criterion* (baris 9). Kemandian kriteria tersebut membutuhkan untuk menentukan cabangnya masing-masing sesuai pada baris 13 dan 14. Terdapat tiga kemungkinan untuk kriteria jika A merupakan *splitting_attribute* yang memiliki nilai unik seperti $\{a_1, a_2, \dots, a_v\}$. Tiga kemungkinan tersebut dapat dilihat pada gambar 2.4, berikut penjelasannya:

1. **Discrete valued**: cabang yang dihasilkan memiliki kelas dengan nilai diskrit. Karena kelas yang dihasilkan diskrit dan hanya memiliki nilai yang sama pada cabang tersebut, maka *attribut_list* akan dihapus (baris 10 sampai 12)
2. **Continuous values**: cabang yang dihasilkan memiliki jarak nilai untuk memenuhi suatu kondisi (contoh: $A \leq \text{split_point}$), dimana nilai *split_point* adalah nilai pembagi yang dikembalikan oleh *attribute_selection_method*
3. **Discrete valued and a binary tree**: cabang yang dihasilkan berupa nilai iya dan tidak dari "apakah A anggota S_a ", dimana S_a merupakan sub-set dari A, yang dikembalikan oleh *Attribute_selection_method*

Kemandian, algoritma *decision tree* akan dipanggil untuk setiap nilai hasil pembagian pada tuple, D_j (baris 18).

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

1. Semua tuple pada partisi D memiliki bagian kelas yang sama.
2. Tidak ada atribut yang bisa dibagi (dilakukan pengambilan pada baris 4). Disini, akan dilakukan *majority voting* (baris 6) yang akan mengkonversi node N menjadi *leaf* dan dibelabilding dengan kelas yang terbanyak pada D.



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

3. Sudah tidak ada tupl yang dapat dibagi cabang, D_j sudah kosong (baris 15) dan leaf akan dibuat dengan lab 1 ber nilai *majority class* pada D (baris 16).

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

Terdapat beberapa teknik untuk memilih atribut pada *decision tree*, dua diantaranya adalah ID3 dan C4.5. ID3 merupakan teknik pemilihan atribut pada *decision tree* dengan menggunakan *entropy* dan *gain info* untuk menentukan atribut yang terbaik. Sedangkan C4.5 merupakan teknik lanjutan dari ID3 yang menggunakan *gain ratio* untuk melakukan pengambilan pada nilai *gain info*. Kedua teknik tersebut menggunakan prinsip *greedy* 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 tupl pada latihan kelas 1 k dalam klas masing-masing. **Attribute Selection Measure** menyediakan peringkat untuk setiap atribut pada training tupl. Jika *splitting criterion* merupakan nilai *continuous* atau *binary trees*, maka nilai *split point* dan *splitting subset* harus ditentukan sebagai bagian dari *splitting criterion*. Contoh dari *attribute selection measure* adalah *information gain*, *gain ratio*, dan *gini index*.

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

ID3

ID3 merupakan teknik untuk membuat *decision tree* dengan menggunakan *information gain* sebagai *attribute selection measure* untuk memilih atribut. Cara ID3 mendapatkan *information gain* dengan menggunakan *entropy*. *Entropy* adalah ukuran *impurity* (kemungkinan tidak ada informasi) dari suatu data. Cara mendapatkan nilai *entropy* adalah

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

Dimana p_i merupakan probabilitas tuple pada D terhadap class C_i , dapat diperoleh dengan $|C_{i,d}|/|D|$. $Info(D)$ merupakan nilai rata-rata *entropy* dari suatu labirin pada tuple D . Cara mengetahui atribut yang paling baik untuk dijadikan *splitting attribute* adalah dengan cara menghitung nilai *entropy* dari suatu atribut k mudian diselesaikan dengan nilai *entropy* dari D . Jika pada tuple D , memiliki atribut A dengan v nilai yang berbeda, maka menghitung *entropy* dari suatu atribut adalah

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

$|D_j|/|D|$ merupakan angka yang menghitung bobot dari suatu partisi.

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

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

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

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

Tab 1.2.2: Contoh training set

| RID | umur | pendapatan | siswa | risko_kredit | Class: mmbilikomputer |
|-----|--------|------------|-------|--------------|-----------------------|
| 1 | muda | tinggi | tidak | cukup | tidak |
| 2 | muda | tinggi | tidak | baik | tidak |
| 3 | remaja | tinggi | tidak | cukup | ya |
| 4 | dewasa | cukup | tidak | cukup | ya |
| 5 | dewasa | rendah | ya | cukup | ya |
| 6 | dewasa | rendah | ya | baik | tidak |
| 7 | remaja | rendah | ya | baik | ya |
| 8 | muda | cukup | tidak | cukup | tidak |
| 9 | muda | rendah | ya | cukup | ya |
| 10 | dewasa | cukup | ya | cukup | ya |
| 11 | muda | cukup | ya | baik | ya |
| 12 | remaja | cukup | tidak | baik | ya |
| 13 | remaja | tinggi | ya | cukup | ya |
| 14 | dewasa | cukup | tidak | baik | tidak |

Pada tabel 2.2, terdapat *training set*, D . Atribut kelas labirin merupakan dua nilai yang berbeda

yaitu ya dan tidak, maka dari itu, nilai $m = 2$. C_1 diisi dengan klas lab 1 ber nilai ya, sedangkan C_2 diisi dengan klas lab 1 ber nilai tidak. Terdapat sembilan tupel atribut klas lab 1 dengan nilai ya dan lima tupel dengan nilai tidak. Untuk dapat menentukan splitting criterion, information gain harus dihitung untuk setiap atribut terlebih dahulu. Perhitungan entropy untuk D adalah

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

Setelah diperoleh nilai entropy dari D, kemudian akan dihitung nilai entropy atribut dimulai dari atribut umur. Pada kategori muda, terdapat dua tupel dengan klas ya dan tiga tupel dengan klas tidak. Untuk kategori remaja, terdapat empat tupel dengan klas ya dan nol tupel dengan klas tidak. Pada kategori dewasa, terdapat tiga dengan klas ya dan dua dengan klas tidak. Perhitungan nilai entropy atribut umur terhadap D sebagai berikut

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

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

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

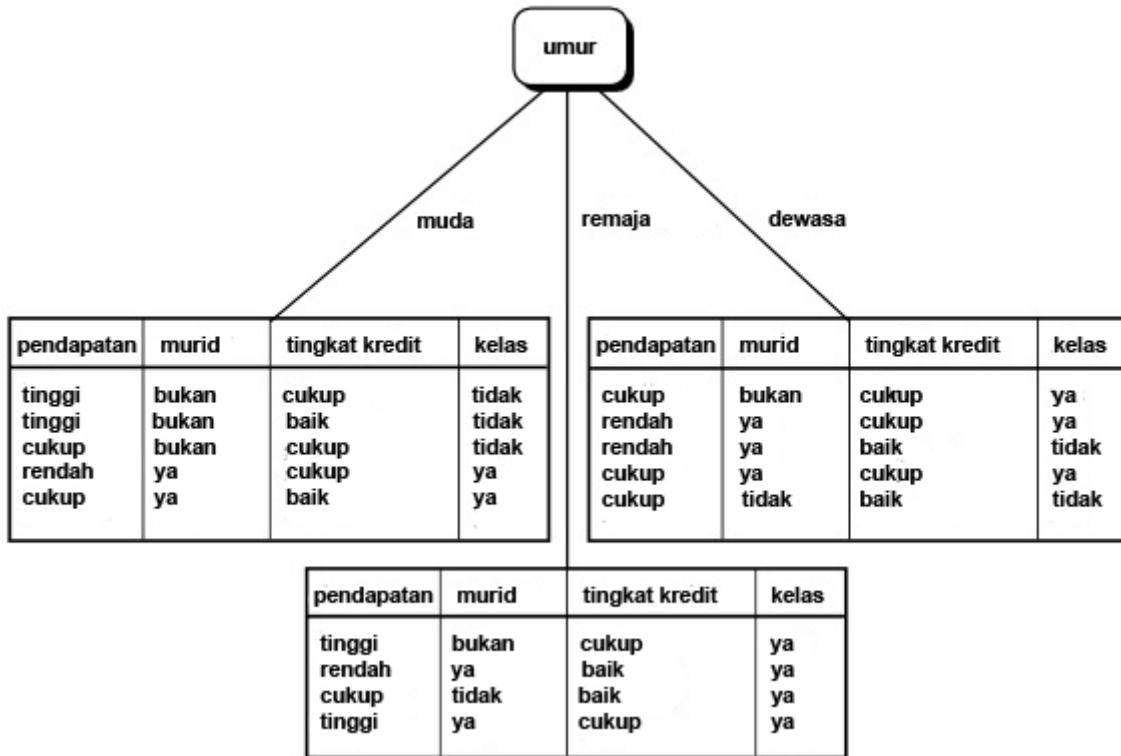
Dengan melakukan hal yang sama, dapat diperoleh nilai gain untuk atribut pendidikan adalah 0.029 bits, untuk nilai gain(siswa) adalah 0.151 bits, dan gain(rasiko_kredit) = 0.048 bits. Karena nilai gain dari atribut umur merupakan nilai terbesar diantara semua atribut, maka atribut umur dipilih menjadi splitting attribute. Kemudian, nod N akan membentuk cabang berdasarkan nilai dari atribut umur seperti pada gambar 2.5.

Untuk memilih atribut yang merupakan nilai kontinu, perencanaan nilai split point harus dilakukan terlebih dahulu. Nilai yang diambil adalah nilai tengahnya untuk dijadikan split-point. Jika terdapat nilai yang berada dari A, maka akan terdapat nilai mungkin split point. Kemudian nilai split point akan dijadikan sebagai nilai pembagi, sebagai contoh: $A \leq \text{split-point}$ merupakan cabang pertama, dan $A > \text{split-point}$ merupakan cabang kedua.

C4.5

Information gain akan memiliki nilai yang baik jika suatu atribut memiliki banyak nilai yang berbeda, namun hal itu tidak selalu bagus. Sebagai contoh kasus, jika nilai id suatu tabel yang memiliki nilai unik, maka akan terdapat banyak sekali cabang. Namun setiap cabang hanya akan berisi satu tupel dan bersifat pure, maka nilai entropy yang dihasilkan adalah 0. Oleh karena itu, informasi yang diperoleh pada atribut ini akan ber nilai maksimum namun tidak akan berguna untuk classification [1]. Selain itu, ID3 dapat menghasilkan decision tree yang memprediksi secara berulang (overestimated) atau dapat juga overfitting. Hal ini dikarenakan pohon yang dihasilkan terlalu dalam hingga data input memiliki hasil prediksi yang pasti.

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



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

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

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

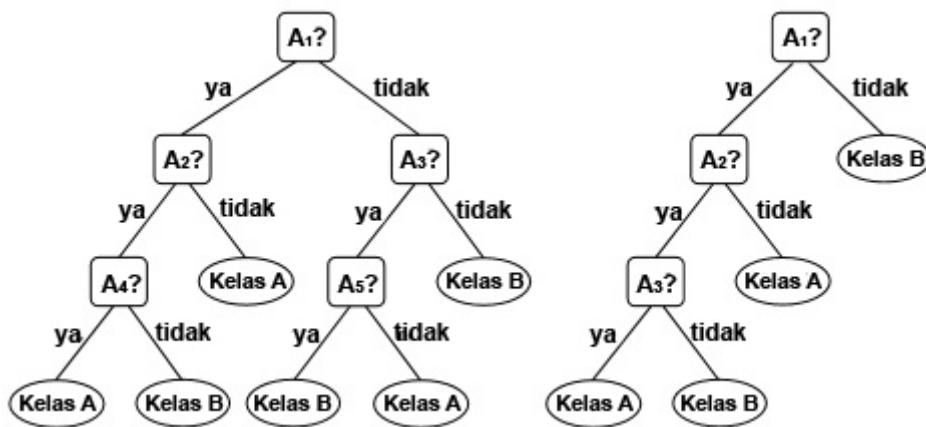
$$\text{SplitInfo}_A(D) = - \sum_{j=1}^v \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right)$$

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

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

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

Contoh studi kasus, akan dilakukan p rhitungan *gain ratio* d ngan m nggunakan training s t pada tabl 2.2. Dapat dilihat pada atribut p ndapatan m miliki tiga partisi yaitu r ndah, s dang, dan tinggi. T rdapat mpat tupl d ngan nilai r ndah, nam tupl d ngan nilai s dang, dan mpat tupl d ngan nilai tinggi. Untuk m nghitung *gain ratio*, p rlu dihitung nilai *split information*



Gambar 2.6: Decision tree yang belum dipotong dan yang sudah dipotong. Dituliskan dari [1]

telah diambil dahulu dengan cara:

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

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

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

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

Tree Pruning merupakan proses pemotongan decision tree agar lebih fisik namun tidak terlalu mengurangi nilai klasifikasi yang dihasilkan. decision tree yang sudah dipotong akan lebih kecil ukuran pohonnya, tidak se rumit dengan pohon yang asli, namun lebih mudah untuk diproses. Decision tree yang sudah dipotong memiliki karakteristik seperti tidak mungkin 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.

Prepruning, pemotongan pohon dilakukan dengan cara memotong dan tidak melanjutkan pembuatan cabang atau partisi dari sebuah node, dan membuat node tersebut menjadi leaf. Postpruning, pemotongan pohon dilakukan ketika decision tree sudah selesai dibangun dengan cara mengubah cabang pohon menjadi leaf.

2.1.6 Pattern Evaluation

Pattern evaluation merupakan tahap mengidentifikasi apakah pola tersebut menarik dan menunjukkan knowledge berdasarkan beberapa interestingness measures. Suatu pola dapat dinyatakan menarik apabila

- mudah dimengerti oleh manusia
- valid untuk data percobaan maupun data yang baru
- memiliki potensi atau fungsi
- merupakan bentuk *knowledge*

2.1.7 Knowledge Presentation

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

2.2 Log Histori KIRI

KIRI memiliki log histori yang merupakan catatan untuk setiap user ketika menggunakan KIRI. Data log tersebut dapat dilihat dengan cara melakukan wawancara dengan developer KIRI, yaitu Pascal Alfadian. Data log yang dibuat sudah dalam format XML.

Log tersebut memiliki 5 field untuk setiap tuple sebagai berikut:

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

LogId merupakan field dengan tip data integer 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 menyindikasi pengguna KIRI ketika menggunakan KIRI. *Timestamp* (UTC) merupakan field dengan tip data *timestamp* untuk mencatat waktu penggunaan KIRI oleh user, diisi dengan menggunakan fungsi *current time*. *Current time* merupakan fungsi untuk pengisian data pada database dengan mengambil waktu pada komputer ketika record dibuat. Action merupakan field dengan tip data varchar untuk memeriksa 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 k tika us r m ncari transportasi di da rah rut s dang dicari
- *PAGELOAD*, *action* yang dicatat k tika us r m masuki halaman KIRI
- *REGISTER*, *action* yang dicatat k tika d v lop rs m lakukan p ndaftaran pada KIRI *API key*
- *SEARCHPLACE*, *action* yang dicatat k tika us r m manggil fungsi p ncarian lokasi d ngan m nggunakan nama t mpat
- *WIDGETERROR*, m ncatat log t rs but k tika us r m n rima rrор dari *widget*
- *WIDGETLOAD*, m ncatat log t rs but k tika us r m lakukan download widg t

AdditionalData, m rupakan *field* d ngan tip data varchar untuk m ncatat informasi s suai d ngan *field action*. Isi dari *additionalData* untuk s tiap *action* adalah

- Jika nilai atribut *action* adalah *ADDAPIKEY*, maka isi nilai dari *additionalData* adalah nilai *API key* yang dihasilkan
- Jika nilai atribut *action* adalah *FINDROUTE*, maka isi nilai dari *additionalData* adalah *latitude* dan *longitude* lokasi awal dan tujuan s rta banyak jalur yang dihasilkan dari aplikasi KIRI
- Jika nilai atribut *action* adalah *LOGIN*, maka isi nilai dari *additionalData* adalah id dari us r yang m lakukan login s rta status apakah us r b rhasil login atau tidak
- Jika nilai atribut *action* adalah *NEARBYTRANSPORT*, maka isi dari *additionalData* adalah *latitude* dan *longitude* dari transportasi t rs but
- Jika nilai atribut *action* adalah *PAGELOAD*, maka isi nilai dari *additionalData* adalah ip dari us r
- Jika nilai atribut *action* adalah *REGISTER*, maka isi nilai dari *additionalData* adalah alamat mail yang digunakan untuk m r gis r dan nama us r
- Jika nilai atribut *action* adalah *SEARCHPLACE*, maka isi nilai dari *additionalData* adalah nama t mpat yang dicari
- Jika nilai atribut *action* adalah *WIDGETERROR*, maka isi nilai dari *additionalData* adalah isi p san dari rrор yang t rjadi
- Jika nilai atribut *action* adalah *WIDGETLOAD*, maka isi nilai dari *additionalData* adalah ip dari us r yang m lakukan download widg t

2.3 Haversine Formula

[2] *Haversine Formula* dapat m nghasilkan nilai jarak antar dua titik pada bola dari garis bujur dan garis lintang titik t rs but. Berikut rumus Havrsin :

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

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

$$d = R.c$$

Dimana

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

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

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

$$a = 0.0026906745$$

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

$$c = 0.1037900036$$

$$d = 6.371 \times 0.1037900036$$

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

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

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

2.4 Weka

[3] Weka merupakan aplikasi berbasis Java yang berisi alat-alat untuk melakukan visualisasi dan algoritma data analisis serta pembelajaran pr diksi. Weka juga menyediakan file weka-src.jar yang berisi klasifikasi yang dipakai oleh aplikasi Weka sertai hingga saat ini dapat menggunakan untuk membuat program Java yang berfungsi untuk *data mining*. Berikut beberapa klasifikasi yang dimiliki oleh Weka:

Classifier adalah sebuah interface yang digunakan sebagai alat untuk mendekripsi numerik ataupun nominal pada Weka.

Constructor:

- void buildClassifier(Instances data)

M thod untuk m lakukan klasifikasi d ngan param t r s t data p latihan.

- double classifyInstance(Instances instance)

M thod untuk m lakukan klasifikasi dari data d ngan param t r data yang akan dilakukan klasifikasi. M thod t rs but akan m ng balikan nilai k las yang s suai d ngan data t rs but.

Instance adalah int rfac yang m wakili s t data.

Instances adalah k las untuk m nangani s t data.

Constructor:

- Instances(java.io.Reader reader)

M thod constructor k las instances yang m nggunakan java.io.Reader untuk m mbaca fil d ngan format .arff. Data yang dit rima dari fil yang dibaca ol h k las Reader akan langsung diubah m njadi k las Instances dan disimpan pada obj k Instances yang dibuat.

Attribute adalah k las yang digunakan untuk m nangani atribut.

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

- void buildClassifier(Instances data)

M mbangun pohon k putusan d ngan ID3 s bagai atribut_m thod_s l ction b rdasarkan data input dalam k las Instances.

- java.lang.String toString()

M ng mbalikan pohon k putusan yang sudah dibangun dalam b ntuk String.

J48 adalah k las yang digunakan untuk m mbuat decision tree c4.5. *Method:*

- void buildClassifier(Instances instance)

M mbangun pohon k putusan d ngan C4.5 s bagai atribut_m thod_s l ction b rdasarkan data input dalam k las Instances.

- java.lang.String toString()

M ng mbalikan pohon k putusan yang sudah dibangun dalam b ntuk String.

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

- boolean setInputFormat(Instances instance)

M ngubah input data yang akan diubah tip nya.

- void setOption(String[] option)

M ngubah p ny tingan p ngaturan.

2.5 Graph iz

[4] Graphviz merupakan rangkat lunak *open source* untuk visualisasi grafik. Dengan menggunakan graphviz, visualisasi grafik dapat dibuat dengan menggunakan kod . Bahasa program yang digunakan oleh graphviz adalah bahasa DOT. DOT merupakan bahasa deskripsi grafik dalam bentuk *plain teks*.

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

- **Node**, merupakan kata kunci untuk membuat sebuah *node*. *Node* dapat dibuat dengan cara menuliskan nama node yang ingin dibuat. Kata kunci ini memiliki atribut yang dapat diubah, diantaranya adalah label *node*, bentuk *node*, ukuran *node*, warna *node*, dan *style node*. Penulisan atribut dapat diawali dengan tanda '[' dan diakhiri dengan tanda ']'. Contoh untuk penulisan *node* dapat dilihat pada listing 2.1 baris ke 2,3, dan 5.
- **Edge**, merupakan kata kunci yang digunakan untuk mengubah atribut dari edge. Atribut *edge* yang dapat diubah diantaranya warna dari *edge* yang dibuat serta lanjutnya. Pengubahan atribut *edge* dapat dilakukan dengan cara menulis kata " dg " kemudian menulis atribut yang akan diubah yang diawali dengan tanda '[' dan diakhiri dengan tanda ']'. Contoh penulisan kata kunci *edge* dapat dilihat pada listing 2.1 baris ke 6.

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

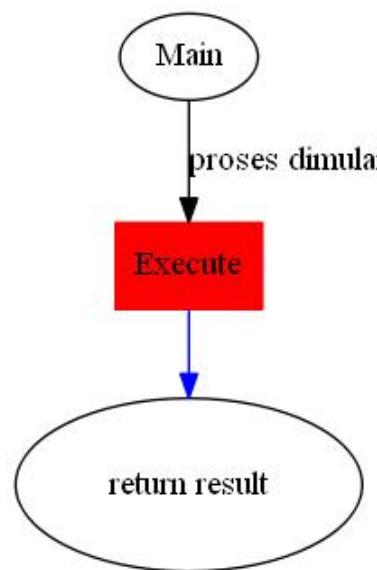
Berikut contoh kod dengan bahasa DOT yang dapat dijadikan input untuk aplikasi graphviz:

Listing 2.1: Dot Example

```

1 digraph G{
2 Main
3 Execute [shape=box, color=red, style=filled]
4 Main -> Execute [label="proses dimulai"]
5 Output [label="return result", width=2, height=1]
6 edge [color=blue]
7 Execute -> Output
8 }
```

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



Gambar 2.7: Hasil output Graphviz

BAB 3

ANALISA

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

3.1 Analisis Data

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

3.1.1 *Data Cleaning*

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

3.1.2 *Data Integration*

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

3.1.3 *Data Selection*

Pada tahap ini, dilakukan pemilihan data yang digunakan untuk proses *data mining*. Pada penelitian ini, dilakukan proses *data mining* mengenai lokasi kota rangkatan dan tujuan dari seorang user yang menggunakan aplikasi KIRI. Oleh karena itu, pada atribut *action*, nilai yang dipilih hanya *FINDROUTE*. Hal ini dikarenakan, hanya *action FINDROUTE* yang menjelaskan posisi kota rangkatan dan tujuan dari user. Selain itu, data tersebut terlihat memerlukan klasifikasi mengenai perpindahan penduduk khususnya untuk daerah Bandung. Karena seluruh *action* ber nilai satuan jenis yaitu *FINDROUTE*, maka atribut tersebut dapat dihilangkan. Selain itu, atribut *logId* dan *APIK* yang tidak dimasukkan ke dalam proses karena tidak memiliki hubungan dengan lokasi kota rangkatan dan tujuan dari seorang user.

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

- *Timestamp (UTC)*
- *AdditionalData*

Contoh data dari atribut tersebut dapat dilihat pada tabel 3.1

Tab 3.1: Contoh data log KIRI setelah data selection

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

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

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

Nilai dari banyak jalur dibuang ketika masuki tahap *data transformation*, karena nilai tersebut hanya menunjukkan banyak jalur tetapi user pasti hanya memiliki salah satu dari jalur tersebut, sehingga nilai jalur ini dapat diasumsikan memiliki nilai 1 saja. Karena kolom jalur berisi nilai satu saja, maka kolom tersebut dapat dibuang.

3.1.4 Data Transformation

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

Pada atribut *timestamp*, nilai waktu dari atribut tersebut diubah menjadi waktu GMT+7. Kemudian, data diubah menjadi input atribut, yaitu:

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

- **Hari**, atribut ini m nunjukkan hari k tika us r KIRI m manggil *action FINDROUTE*, d ngan rang nilai antara s nin sampai minggu. Nilai t rs but dapat dip rol h d ngan cara m lakukan m manggil *method* p ncarian hari b rdasarkan tanggal dari tim stamp pada java.
- **Jam**, atribut ini m nunjukkan jam k tika us r KIRI m manggil *action FINDROUTE*, d ngan rang nilai antara 00 sampai 23. Nilai t rs but dapat dip rol h d ngan cara m ngambil nilai string dari timp stamp yang b rada di antara spasi dan titik dua.

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

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

- **Latitude keberangkatan**, atribut ini b risi nilai latitud dari lokasi k b rangkatan yang dipilih ol h us r. Nilai t rs but dapat dip rol h d ngan cara m ngambil nilai string s b lum koma yang p rtama.
- **Longitude keberangkatan**, atribut ini b risi nilai longitudo dari lokasi k b rangkatan yang dipilih ol h us r. Nilai t rs but dapat dip rol h d ngan cara m ngambil nilai string yang b rada di antara koma p rtama dan garis miring p rtama.
- **Latitude tujuan**, atribut ini b risi nilai latitud dari lokasi tujuan yang dipilih ol h us r. Nilai t rs but dapat dip rol h d ngan cara m ngambil nilai string di antara garis miring yang p rtama dan koma k dua.
- **Longitude tujuan**, atribut ini b risi nilai longitudo dari lokasi tujuan yang dipilih ol h us r. Nilai t rs but dapat dip rol h d ngan cara m ngambil nilai string yang b rada di antara koma k dua dan garis miring k dua.

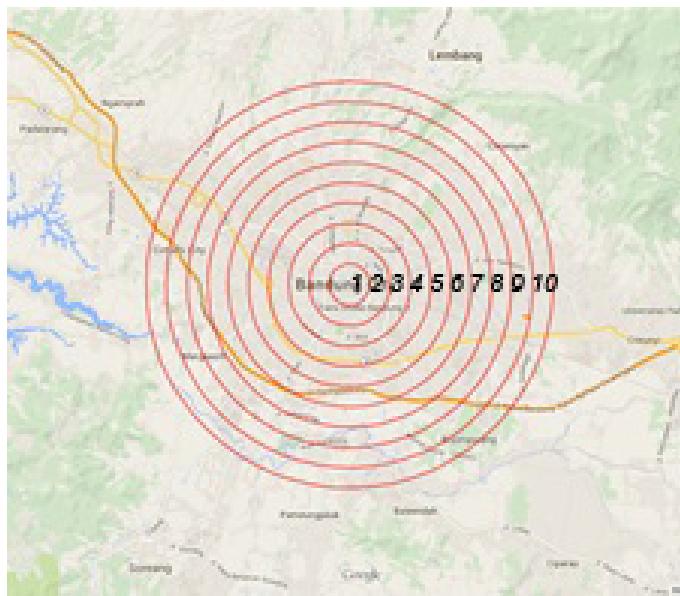
Dari analisis diatas, banyak atribut dari tab 1 *statistics* m njadi d lapan, yaitu:

- Bulan
- Tahun
- Hari
- Jam
- Latitud K b rangkatan
- Longitud K b rangkatan
- Latitud Tujuan
- Longitud Tujuan

Contoh hasil data transformasi jika input m rupakan data dari tab 1 [3.1](#) dapat dilihat pada tab 1 [3.2](#).

| Bulan | Tahun | Hari | Jam | Latitude rangkatan | 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 1 3.2: Contoh hasil data transformasi



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

Agar *decision tree* mengambil lokasi sebagai rangkatan dan tujuan dari user KIRI dapat dipilih, maka atribut lokasi yang digunakan adalah nilai latitud dan longitudo dari lokasi sebagai rangkatan dan tujuan. Ketika atribut lokasi ada empat, maka dilakukan perhitungan rincianan dari kategori atribut untuk meningkatkan akurasi serta tingkat fisik proses *data mining*.

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

Pentingnya *classification* berdasarkan titik pusat Bandung, yaitu -6.916667,107.6¹ dalam latitud dan longitudo. Kedua titik dibagi menjadi sepuluh daerah yang memiliki perbedaan radius sebesar 1 km, sehingga diameter untuk daerah pertama adalah 2 km, diameter untuk daerah kedua adalah 4 km, dan seterusnya, untuk daerah terakhir (yaitu daerah 10) memiliki diameter 20 km.

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

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

S tetapi lokasi k b rangkatan dan lokasi tujuan ditentukan dari arahnya, dapat ditentukan apakah user t rs but m menuju pusat Bandung atau tidak. Jika arah dari lokasi k b rangkatan lbih b saran daripada arah lokasi tujuan, maka user t rs but m menuju pusat Bandung. Jika arah dari lokasi k b rangkatan lbih kcil daripada arah lokasi tujuan, maka user t rs but m menuju pusat Bandung. Sedangkan, jika lokasi k b rangkatan dan lokasi tujuan berada di arah yang sama, maka user t rs but m menuju user t rs but m menuju arah yang sama.

Dengan adanya perhitungan jarak dan pernentuan arah Bandung, nilai latitud dan longitud dari lokasi k b rangkatan dan tujuan dapat dibuang dan diganti oleh atribut m menuju Bandung dengan tipe data integer. Jika isi dari atribut t rs but m memiliki nilai 1, maka user t rs but m menuju Bandung sedangkan nilai -1 berarti user tidak menuju Bandung, dan jika nilai atribut t rs but adalah 0, maka user t rs but m memiliki lokasi k b rangkatan dan tujuan di arah yang sama. Contoh hasil data setelah dilakukan transformation terhadap latitud dan longitud dapat pada tabel 3.3.

Tab 3.3: Contoh hasil data transformasi latitud longitud

| Bulan | Tahun | Hari | Jam | Arah Keberangkatan |
|-------|-------|-------|-----|--------------------|
| 02 | 2014 | Sabtu | 07 | -1 |
| 02 | 2014 | Sabtu | 07 | 1 |
| 02 | 2014 | Sabtu | 07 | 1 |
| 02 | 2014 | Sabtu | 07 | 0 |
| 02 | 2014 | Sabtu | 07 | 1 |
| 02 | 2014 | Sabtu | 07 | -1 |
| 02 | 2014 | Sabtu | 07 | 0 |

3.2 Data Convert Hasil Decision Tree menjadi Bahasa DOT

Hasil dari data mining dari wka adalah decision tree dalam bentuk String. Berikut contoh hasil decision tree dari wka:

Listing 3.1: Contoh Hasil Decision Tree

```

1 J48 pruned tree
2
3
4 jam <= 4
5 | jam <= 0
6 | | hari <= 1: 1 (46.0/20.0)
7 | | hari > 1: -1 (198.0/104.0)
8 | jam > 0: -1 (509.0/179.0)
9 jam > 4
10 | bulan <= 11: -1 (18611.0/9986.0)
11 | bulan > 11: 1 (54.0/21.0)
12
13 Number of Leaves : 5
14
15 Size of the tree : 9

```

Nilai yang dibutuhkan dari hasil decision tree (pada listing 3.1) adalah baris 3 sampai 11. Kedalamannya sebuah node dapat dilihat dari banyaknya tanda '|' yang dipisahkan dengan spasi. Terdapat dua cara untuk mengetahui kedalamannya dari decision tree yaitu menghitung banyak tanda '|' atau memiliki suatu variabel yang selalu ditambah satu setiap program memasuki node yang lebih dalam. Cara kedua sedikit lebih fisik namun tidak perlu melakukan loop untuk menghitung

banyak tanda '|' namun program harus bisa tahu kapan *node* yang s d ang dibaca m rupakan *node* yang l bih dalam.

3.2.1 Pengecekan kedalaman *node*

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

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

3.2.2 Ketentuan untuk membuat *edge*

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

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

3.3 Analisis Perangkat Lunak

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

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

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

Sp sifikasi k butuhan p rangkat lunak yang dibangun untuk m lakukan *data mining log histori KIRI* 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

Pemodelan Perangkat Lunak *Data Mining Log Histori KIRI*

P rangkat lunak *data mining log* histori KIRI m n rima input data d ngan format .csv. S t lah program m ndapatkan input dan us r m n kan tombol pros s, maka data t rs but diubah t rl bih dahulu s suai pada bab analisis data(bab 3.1) d ngan m lakukan pros s *transform* dan m nghasilkan data d ngan format s p rti pada tab 1 3.3.

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

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

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

K tika tombol pros s dit kan, maka data t rs but dipros s. Pros s yang p rtama yang dilakukan adalah m lakukan *load* data dari fil . Data csv dibaca d ngan m nggunakan CSVReader s hingga s mua hasil datanya sudah t rpisah s suai d ngan atribut. K mudian dilakukan filter data dan hanya action d ngan nilai FINDROUTE yang akan diambil. S t lah data didapat, dilakukan pros s *transform* untuk s tiap baris yang ada. Pros s *transform* t rs but m miliki tahap s bagai b rikut:

1. M ngubah waktu dari UTC m njadi GMT+7 pada string data input array k tiga (yaitu atribut tanggal).
2. M ngambil atribut tanggal k mudian m m cah nilai t rs but d ngan spasi s bagai tanda p misah, maka akan t rdapat tiga nilai, yaitu hari (dalam b ntuk angka dimana nilai 1 b arti s nin dan nilai 7 b arti minggu), tanggal dan jam.
3. Pada nilai tanggal, dilakukan p m cahan nilai string d ngan garis miring s bagai tanda p misah, maka dip rol h tiga nilai yaitu bulan, tanggal, dan tahun, namun nilai yang akan diambil hanya dua, yaitu bulan dan tahun.
4. Pada nilai jam, dilakukan p m cahan nilai string d ngan titik dua s bagai tanda p misah, maka dip rol h dua nilai yaitu jam dan m nit, namun nilai yang diambil hanya jam.
5. M ngambil string data input array k lima (yaitu atribut *additionalData*), dilakukan p m - cahan nilai string d ngan garis miring s bagai tanda p misah, maka dip rol h tiga nilai yaitu lokasi awal, lokasi tujuan, dan banyak jalur.
6. Pada nilai lokasi awal dan lokasi tujuan, dilakukan p m cahan nilai string d ngan koma s bagai tanda p misah, maka dip rol h dua nilai untuk s tiap lokasi, yaitu *latitude* dan *longitude*.
7. M nghitung jarak posisi lokasi awal dan lokasi tujuan t rhadap titik pusat dan m n ntukan apakah lokasi t rs but b rada pada klasifikasi -1 atau 0 atau 1.

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

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

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

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

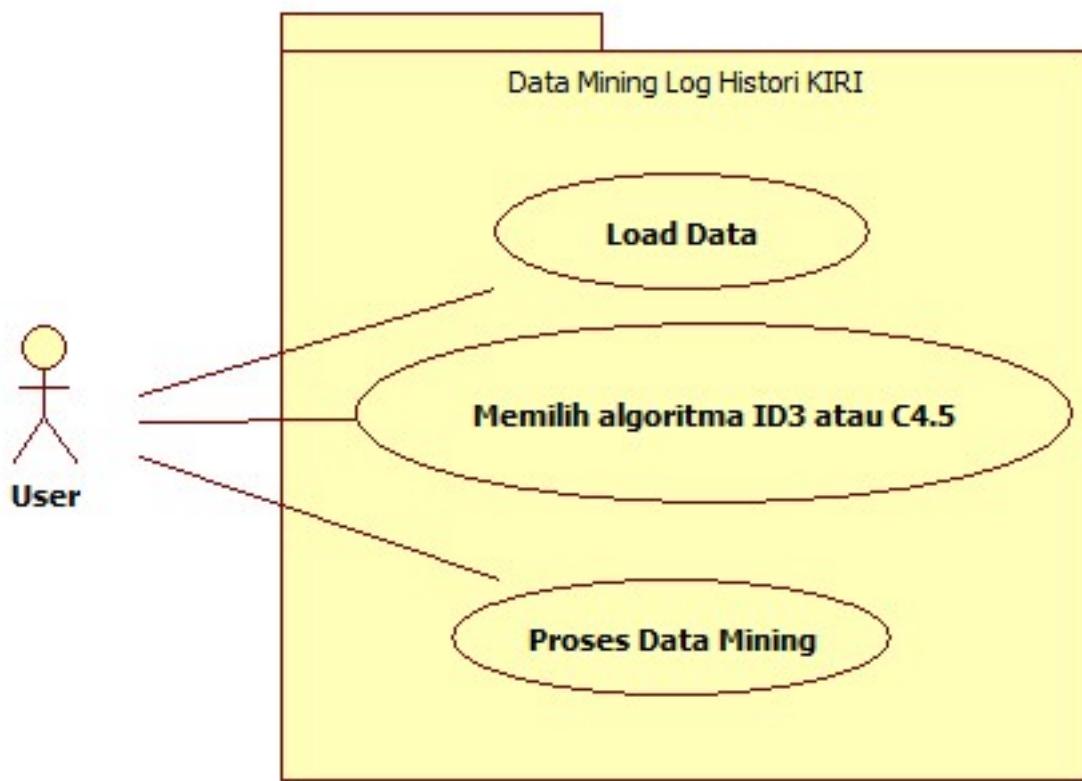
1. Memuat data dan memproses data
2. Mengeksekusi algoritma untuk membuat *decision tree*
3. Membuat grafik dari hasil algoritma *decision tree*
4. Menampilkan grafik *decision tree*

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

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

- Melakukan *load* data yang digunakan sebagai input data dengan cara memasukan alamat data di program
- Memilih algoritma yang digunakan, terdapat dua algoritma, yaitu ID3 dan C4.5
- Melakukan proses *data mining* dengan input data dari alamat data yang sudah dimasukan. Setelah proses berhasil dilaksanakan, program akan menampilkan hasil yang dipilih

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



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

Tab 1 3.5: Sk nario M lakukan *load Data*

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

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

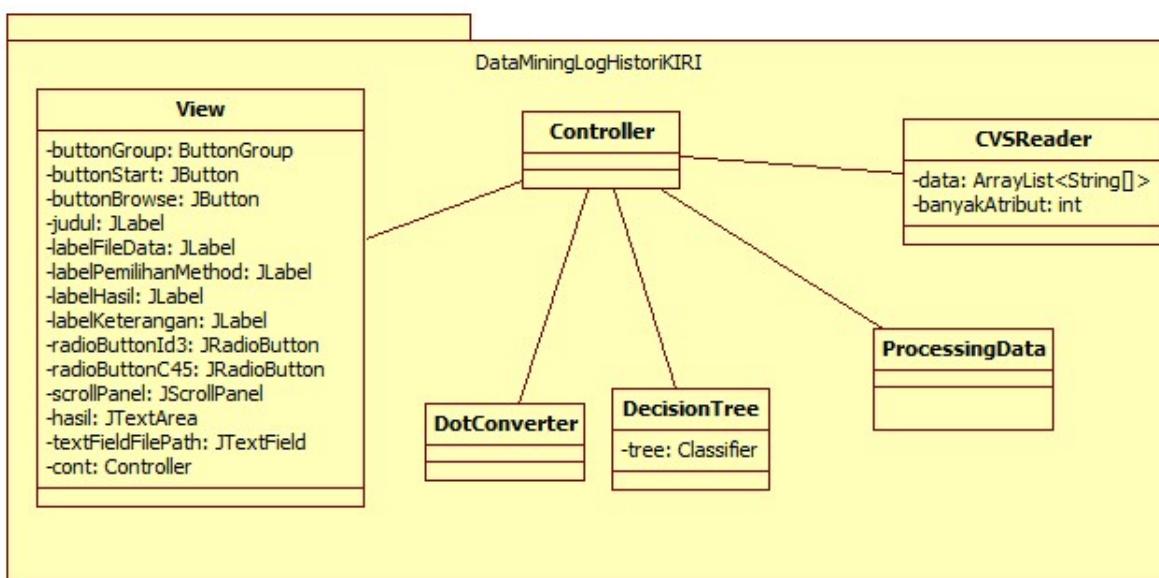
| | |
|----------------|---|
| Nama | Proses Data Mining |
| Aktor | User |
| D skripsi | Mengkan tombol proses pada interface |
| Kondisi awal | Textbox belum terisi |
| Kondisi akhir | Textbox sudah terisi dengan hasil data mining |
| Sk nario utama | User mengkan tombol proses |
| Eks spi | Data tidak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

Tab 1 3.7: Sk nario M milih Algoritma yang Dugunakan

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

3.3.2 Diagram kelas Perangkat Lunak Data Mining Log Histori KIRI

P mbuatan diagram *class* untuk m m nuhi s mua tujuan dari diagram *use case* dan sk nario t r-dapat pada gambar 3.3.

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

B rikut d skripsi k las diagram *class*:

- Vi w, m rupakan k las untuk m ngatur d sain antar muka.
- Controll r, m rupakan k las untuk m ngatur vi w dan modul k tika program dijalankan.
- CSVR ad r, m rupakan k las yang m miliki m thod untuk m mbaca fil d ngan format CSV.
- Proc ssingData, m rupakan k las yang m miliki m thod untuk m lakukan *preprocessing data*.
- D cissionTr , m rupakan k las yang m miliki m thod untuk m mbuat *decision tree* dan m ngitung *confident* dari pohon yang sudah dihasilkan.
- DotConv rt r, m rupakan k las yang m miliki m thod untuk m ngubah *string* yang m ru-pakan hasil dari k las D cissionTr (yaitu, *decision tree* dalam b ntuk *string*) m njadi bahasa DOT yang siap dijadikan *input* untuk graphviz.

BAB 4

PERANCANGAN PERANGKAT LUNAK

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

4.1 Perancangan Perangkat Lunak

4.1.1 Perancangan Kelas

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

- Kelas Controll r, merupakan kelas untuk mengatur viw dan modul ketika program dijalankan.
 - Method
 - * public controll r(), merupakan konstruktor dari kelas controll r.
 - * public void startMining(String inputFil Path, String miningAlgo, JLab1 lab1, JT xtAr a t xtAr a), merupakan method untuk menjalankan modul-modul yang melakukan *data mining* dan membuat *decision tree* dari data yang menjadi masukan program.
 - * public static void main(String[] args), merupakan method main untuk menjalankan program.
- Kelas Viw, merupakan kelas untuk mengatur tampilan antar muka.
 - Atribut
 - * ButtonGroup buttonGroup, digunakan untuk mengelompokkan jRadioButton.
 - * JButton buttonStart, merupakan sebuah tombol yang dapat menggil method buttonStartActionPerfomed() bila diklik.
 - * JButton buttonBrows , merupakan sebuah tombol yang dapat menggil method buttonBrows ActionPerformed() bila diklik.
 - * JLab1 judul, merupakan sebuah lab1 yang berisi judul dari aplikasi ini.
 - * JLab1 lab1Fil Data, merupakan lab1 untuk menunjukkan bagian pilihan file data path.
 - * JLab1 lab1P milihanMethod, merupakan lab1 untuk menunjukkan bagian pilihan method.

- * JLab 1 lab lHasil, m rupakan lab 1 untuk m nunjukkan bagian hasil program.
- * JLab 1 lab lK t rangan, m rupakan lab 1 untuk m nunjukkan k t rangan dari program.
- * JRadioButton radioButtonId3, m rupakan *radio button* yang m nunjukkan bahwa us r m milih m 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 l scrollPan l, m rupakan variab l yang digunakan untuk m ngaktifkan fungsi scroll pada JT xtAr a hasil.
- * JT xtAr a hasil, m rupakan s buah JT xtAr a yang digunakan untuk m nunjukkan hasil *data mining* dari program.
- * JT xtFi ld t xtFi ldFil Path, digunakan untuk m lakukan *input path file* baik dilakukan s cara manual atau m lalui tombol *browse*.
- * Controll r cont, digunakan untuk m panggil *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 panggil m thod startMining d ngan masukan k dua string t rs but, lab l dan t xtAr a.
- K las CSVR ad r, m rupakan k las yang m miliki m thod untuk m mbaca fil d ngan format CSV.
 - Atribut
 - * ArrayList<String[]> data, digunakan untuk m nyimpan isi dari fil CSV yang sudah dibaca.
 - * int banyakAtribut, digunakan untuk m nyimpan banyak atribut yang 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 mendapatkan nilai variabel banyakAtribut.
- * public void setBanyakAtribut(int banyakAtribut), digunakan untuk mengganti nilai variabel banyakAtribut sesuai dengan parameter t_r.
- Kelas Proc ssingData, merupakan kelas yang memiliki method untuk melakukan *preprocessing data*.
 - Method
 - * public Proc ssingData(), merupakan konstruktor dari kelas Proc ssingData.
 - * public void processSorting(ArrayList array, ArrayList data, String action), digunakan untuk memilih arraylist s hingga arraylist ters but hanya berhasil action yang diinginkan saja (pada penitikan ini, action yang diharapkan adalah FINDROUTE). Hasil pilah disimpan pada variabel array dari parameter t_r method s hingga tidak diperlukan return value.
 - * public ArrayList processData(ArrayList<String[]> data), Digunakan untuk melakukan tahap *preprocessing data* seperti yang sudah dilakukan pada pembelajaran data di bab 3. Tujuan dari fungsi ini adalah mendapatkan nilai waktu yang sudah diubah menjadi GMT+7 dan sudah dikompokkan menjadi jam, hari, bulan, dan tahun serta mengetahui klasifikasi kelas dari untuk setiap record dengan menghitung jarak dari titik ke berbagai titik pusat Bandung dan titik tujuan terhadap titik pusat Bandung.
 - * public int KlasifikasiKelas(double jarakKategori, double jarakTujuan), Digunakan untuk menentukan kelas dari hasil jarak titik ke berbagai titik pusat Bandung dan titik tujuan dengan titik pusat Bandung.
- Kelas DecisionTree, merupakan kelas yang memiliki method untuk membuat decision tree dan menghitung akurasi dari pohon yang sudah dihasilkan.
 - Atribut
 - * Classifier tree, digunakan untuk menyimpan decision tree yang sudah dihasilkan.
 - Method
 - * public DecisionTree(), merupakan konstruktor untuk kelas DecisionTree.
 - * public double calculatePrecision(Instances data), digunakan untuk mendapatkan nilai akurasi dari decision tree yang dihasilkan.
 - * public String id3(Instances data), digunakan untuk membuat decision tree dengan menggunakan method ID3 dari API Weka.
 - * public String j48(Instances data), digunakan untuk membuat decision tree dengan menggunakan method C4.5 dari API Weka.
- Kelas DotConverter, merupakan kelas yang memiliki method untuk mengubah string yang merupakan hasil dari kelas DecisionTree (yaitu, decision tree dalam bentuk string) menjadi bahasa dot yang siap dijadikan masukan untuk graphviz.

- M thod

- * public String conv rt(String data, String miningAlgo, String nod Nam), Digunakan untuk m ngubah nilai string yang sudah diprolah dari klas DecisionTree menjadi bahasa DOT untuk membuat visualisasi dengan menggunakan graphviz.

Pada kelas ProcesssingData, nilai data waktu perlu diganti menjadi GMT+7 dan perlu menghitung jarak antar dua titik. Maka dari itu, dibuat dua kelas tambahan untuk melakukan k dua hal tersebut, yaitu Timezone Conv rt dan DistanceHaversine .

- Kelas Timezone Conv rt, merupakan kelas yang memiliki method untuk mengubah waktu dari UTC menjadi GMT+7

- M thod

- * public static String conv rtToGMT7(String data), digunakan untuk mengubah waktu dari UTC menjadi GMT+7.

- Kelas DistanceHaversine , kelas yang memiliki method untuk menghitung jarak dua titik di bumi.

- Atribut

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

- M thod

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

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

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

- M thod

- * public ArffIO, merupakan konstruktor dari kelas ArffIO.

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

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

Kita mulai dengan rancang *method* `conv_rt` yang berada di kelas `DotConv rt`, lebih mudah jika dirancang menjadi `rt` kursif. Karena data yang diolah pada *method* tersebut cukup banyak dan dipilihkan nama yang berbeda pada setiap node yang ditulis pada DOT, maka perlu ditambahkan kelas yang berfungsi untuk struktur data pada kelas tersebut, yaitu `SDForConv rtTr`.

- Kelas `SDForConv rtTr`, kelas yang berfungsi untuk menyimpan data yang dibutuhkan untuk mengubah String hasil dari kelas `DecisionTr` menjadi bahasa DOT.

– Atribut

- * `String[] data`, digunakan untuk menyimpan nama-nama atribut yang diubah dalam bahasa DOT.
- * `int[] count`, digunakan untuk menghitung penggunaan nama setiap atribut sehingga dapat menghasilkan nama node yang berbeda untuk setiap atribut.

– Method

- * `public SDForConv rtTr (String[] data)`, merupakan konstruktor untuk kelas ini dan melakukan inisialisasi data pada atribut dengan nilai data pada parameter tersebut. Perlu dilakukan inisialisasi nilai variabel `count` dengan 0.
- * `public void setData(String data, int x)`, digunakan untuk mengubah nilai data pada index `x` tersebut.
- * `public String[] getData()`, digunakan untuk mendapatkan nilai atribut data.
- * `public String getData(int index)`, digunakan untuk mendapatkan nilai data pada index `x` tersebut.
- * `public void setCount(int count, int index)`, digunakan untuk mengubah nilai `count` pada index `x` tersebut.
- * `public int getCount(int index)`, digunakan untuk mendapatkan nilai `count` pada index `x` tersebut.
- * `public boolean hasNext()`, digunakan untuk mengetahui apakah ada isi dari variabel data masih ada atau tidak.
- * `public void buangArrayPertama()`, digunakan untuk membuang nilai array yang pertama (index ke -0).
- * `public String getaDataNumber(String atribut)`, digunakan untuk mendapatkan angka pada nama atribut tersebut untuk membuat nama node pada kelas `DotConv rt` agar semua nama node berbeda.

Setelah melakukan *convert string* hasil dari *method* pembuatan *decision tree API* ke dalam bahasa Dot, maka dipilihkan panggilan fungsi dot yang terdapat pada graphviz. Cara memanggil fungsi tersebut yaitu dengan menggunakan *command prompt*. Oleh karena itu, dipilihkan kelas yang memiliki *method* untuk memanggil *command prompt* dan menjalankan fungsi dot tersebut, yaitu kelas CMD.

- Kelas CMD, merupakan kelas yang digunakan untuk memanggil *command prompt*.

– Method

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

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

Dari p rancangan k las dan *method* yang sudah dilakukan, maka dip rol h diagram k las s p rti pada 4.1

4.1.2 Sequence Diagram

Pada subbab ini, dij laskan alur program d ngan m nggunakan *sequence diagram* pada 4.2.

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

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

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

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

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

Tahap s lanjutnya adalah m ngubah nilai String yang dip rol h dari *method* id3 atau j48 m njadi bahasa DOT d ngan cara m manggil *method* conv rt pada k las DotConv rt r dan m ng mbalikan nilai String(tahap 24-25). S t lah dip rol h hasil dari *method* conv rt, maka dip rlukan

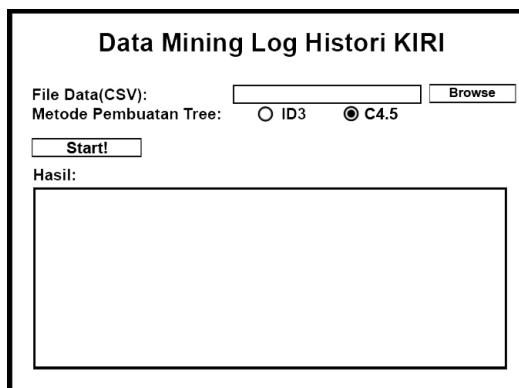
command prompt untuk menghasilkan gambar grafik visualisasi dari *decision tree* yang sudah dihasilkan(tahap 26-27).

Setelah menghasilkan gambar visualisasi *decision tree*, method startMining membuat JFram yang baru untuk menampilkan hasil gambar *decision tree* yang sudah dihasilkan serta mengambil nilai String *decision tree* ke dalam *String* yang ditampilkan di JT xtAr a(tahap 28-29).

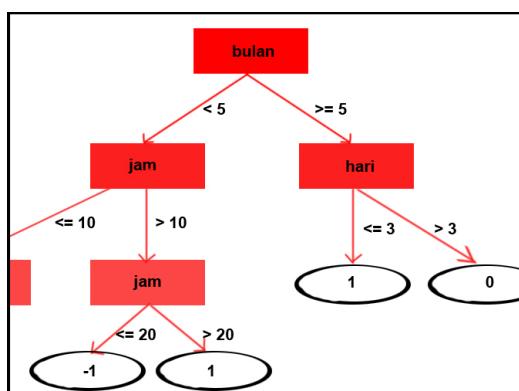
4.1.3 Perancangan Desain Antar Muka

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

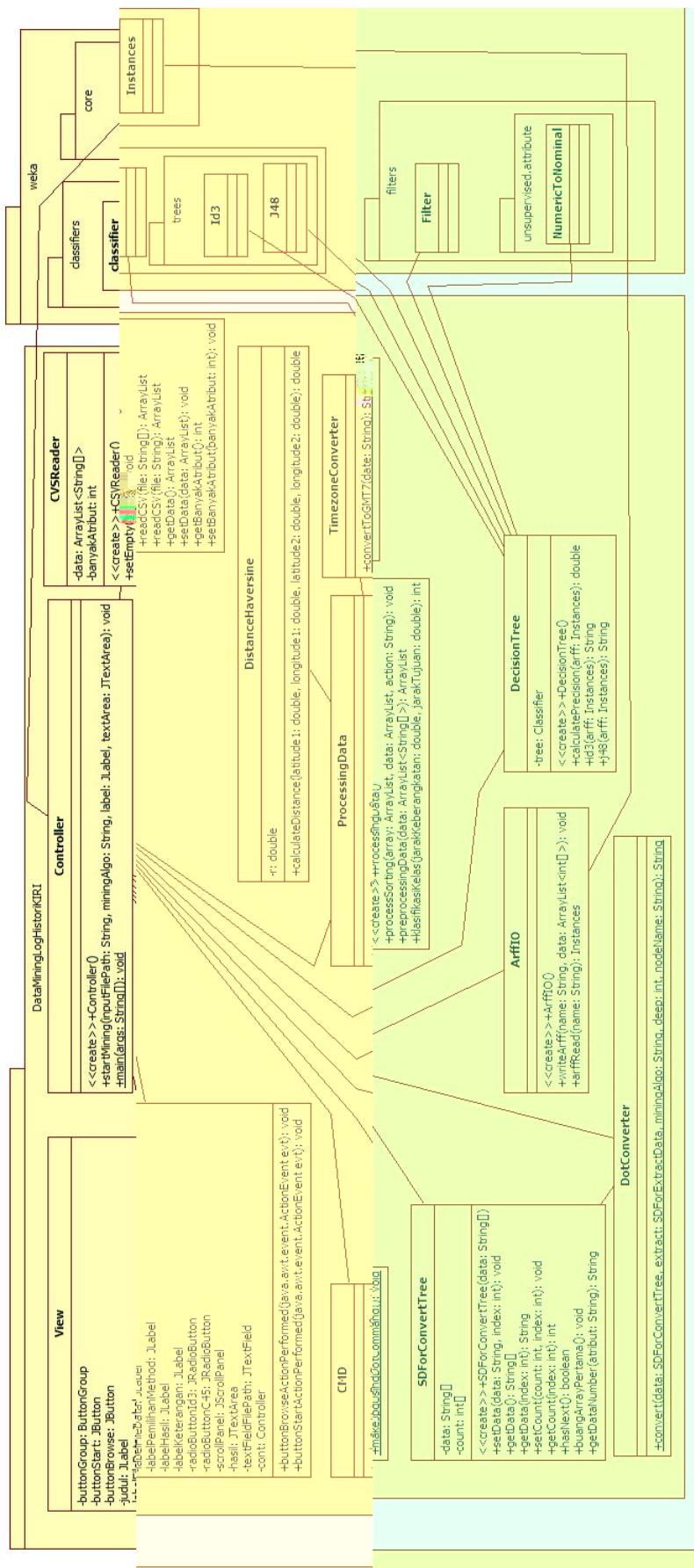
Aplikasi ini memiliki dua form untuk melakukan *data mining* dan membuat *decision tree*. Pada form pertama(dapat dilihat di 4.3) disediakan JTextField dan JButton yang digunakan untuk memilih file, JRadioButton yang digunakan untuk memilih metode pembuatan *decision tree*, JT xtAr a yang digunakan untuk menampilkan 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).



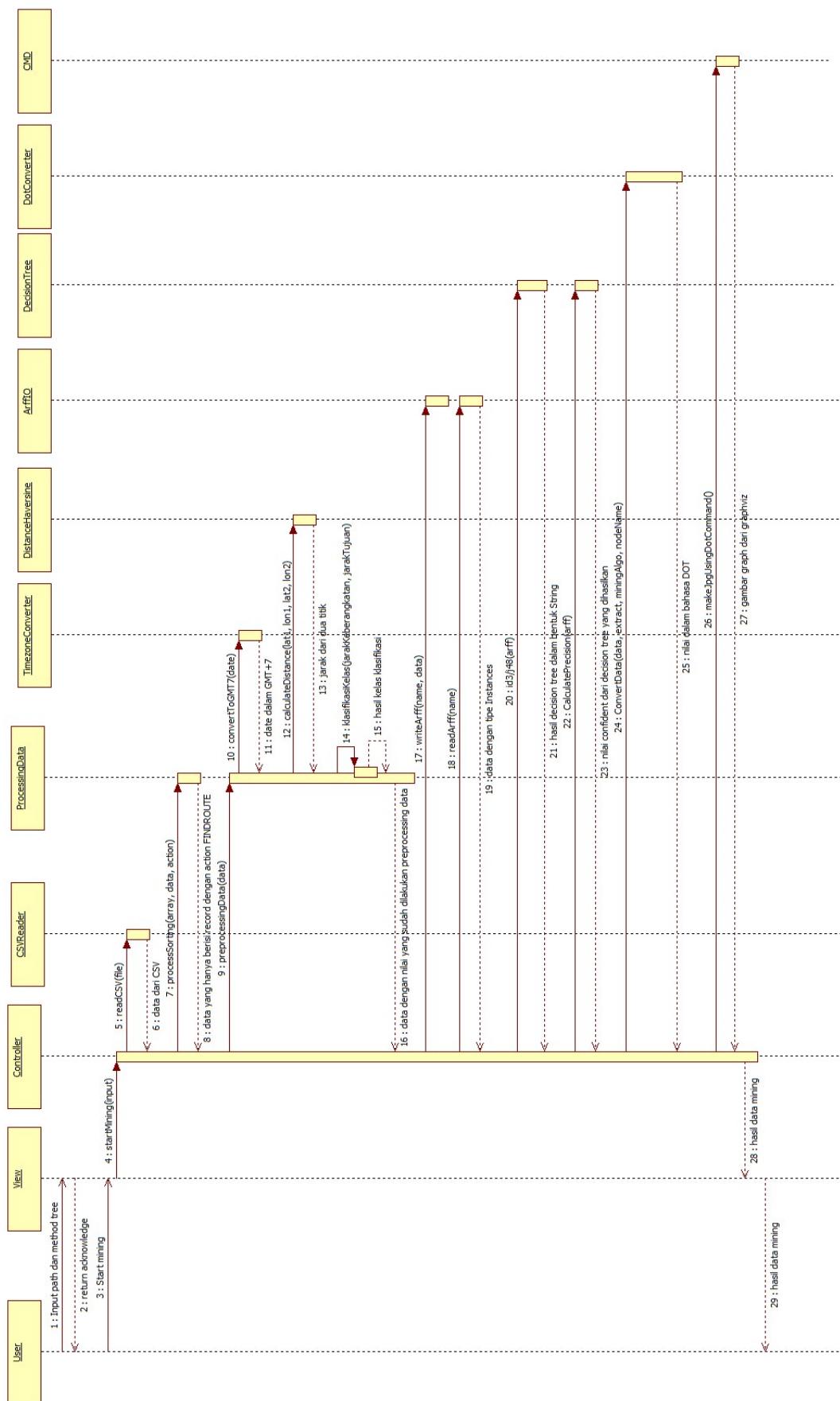
Gambar 4.3: Mock Up Form Pertama



Gambar 4.4: Mock Up Form Kedua



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



Gambar 4.2: Sequence diagram Perangkat Lunak Data Mining Log Histori KIRI

BAB 5

IMPLEMENTASI PROGRAM DAN PENGUJIAN

Bab ini menjelaskan tentang lingkungan pembangunan, implementasi rancangan antarmuka, serta pengujian secara fungsional dan spesifikasi pada aplikasi *data mining*.

5.1 Lingkungan Pembangunan

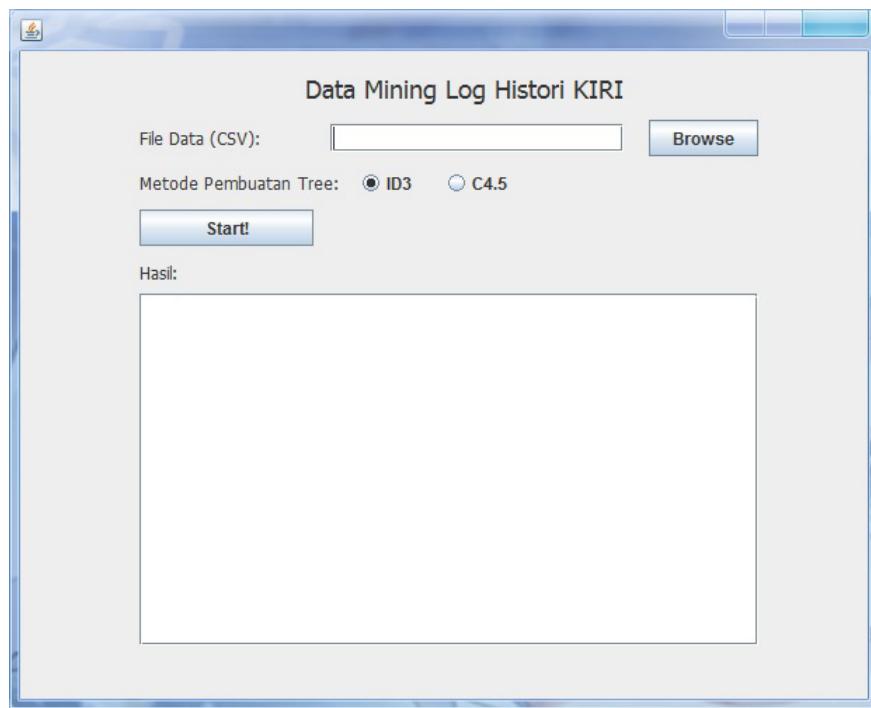
Lingkungan perangkat lunak dan perangkat keras yang digunakan untuk membangun dan menguji aplikasi *data mining* ini adalah:

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

5.2 Hasil Tampilan Antarmuka

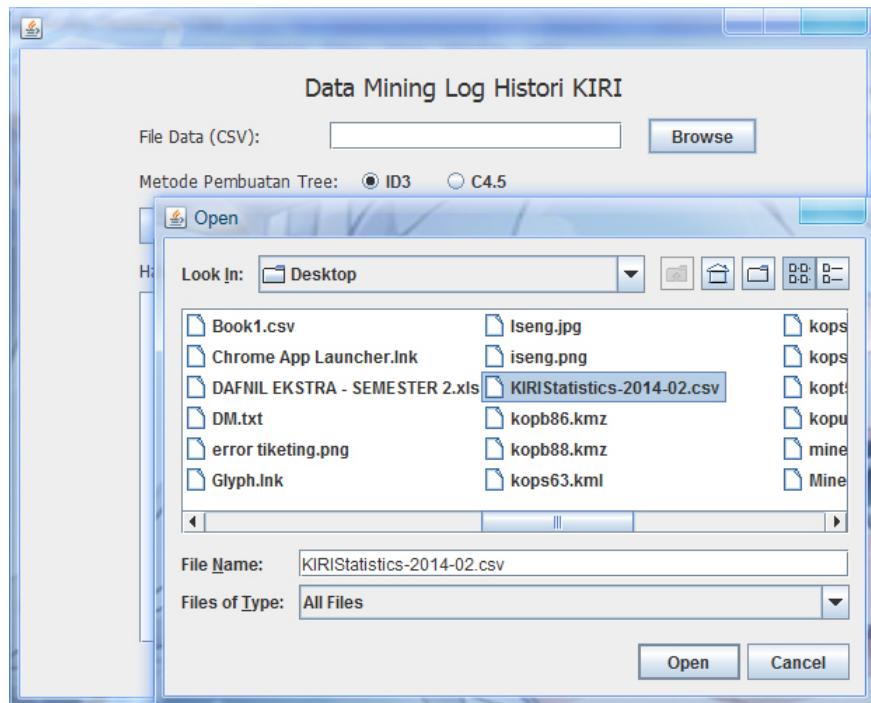
Pada aplikasi *data mining* ini, terdapat dua form. Form pertama berfungsi untuk memilih file CSV yang dilakukan *data mining*, memilih metode pembuatan *decision tree*, serta menunjukkan hasil *decision tree* yang dihasilkan dalam bentuk String. Sedangkan untuk form kedua, berfungsi untuk mewujudkan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

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

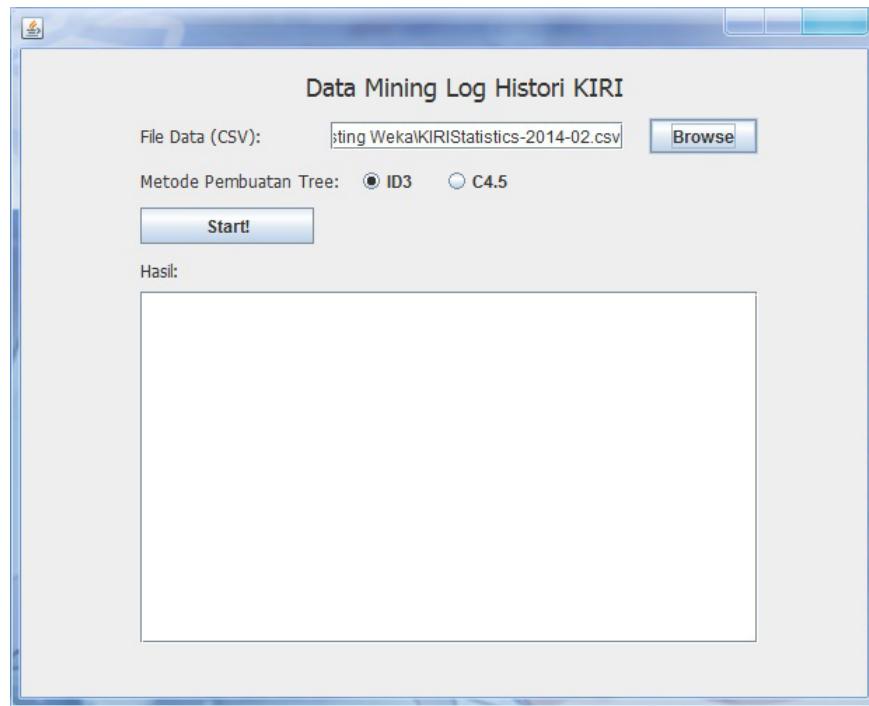


Gambar 5.1: Tampilan Form Awal Aplikasi *Data Mining*

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

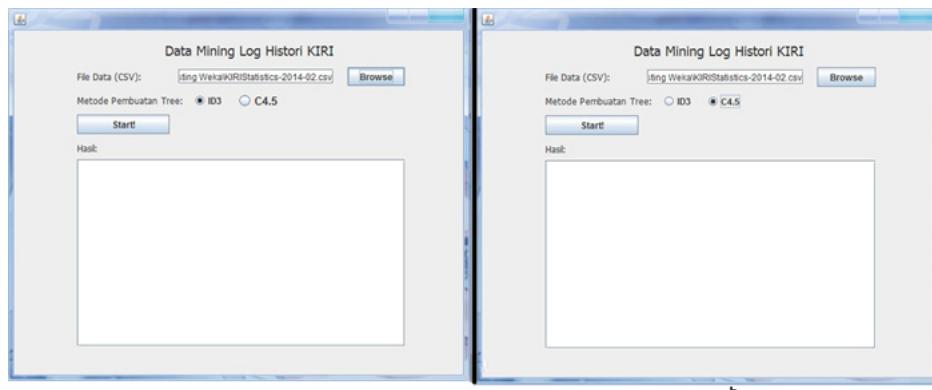


Gambar 5.2: Tampilan File Selector untuk Memilih File CSV



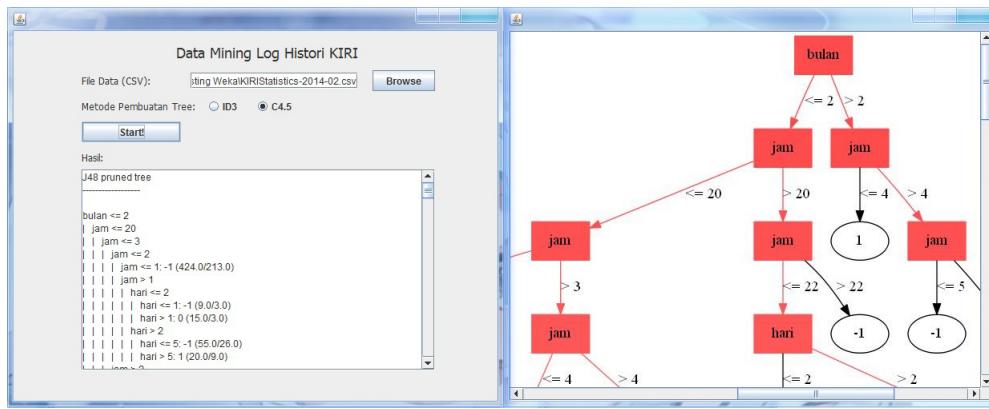
Gambar 5.3: Tampilan Form saat Membuat File CSV

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



Gambar 5.4: Tampilan Form Pada Membuat Decision Tree. Gambar a merupakan kondisi tampilan ketika metode ID3 dipilih sedangkan gambar b merupakan kondisi tampilan ketika metode C4.5 dipilih.

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



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

5.3 Pengujian Aplikasi *Data Mining*

5.3.1 Pengujian Fungsional

Rancangan Pengujian

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

- Pengujian *method* utama untuk mencapai tujuan dari aplikasi *data mining* ini. Terdapat beberapa *method* yang perlu diuji, yaitu
 - Membaca file CSV
 - M Melakukan preprocessing data
 - M Membuat decision tree
 - M Mengubah decision tree dalam bentuk String menjadi bahasa DOT
- Karena rangka lunak hanya dilihat pada hasil kluaran program atau kondisi masukan yang dibutuhkan tanpa melihat bagaimana proses untuk mendapatkan kluaran tersebut.
- Dari kluaran yang dihasilkan, kemampuan program untuk memenuhi kebutuhan permakai dapat diukur dan dapat diketahui jika ada.

Hasil Pengujian

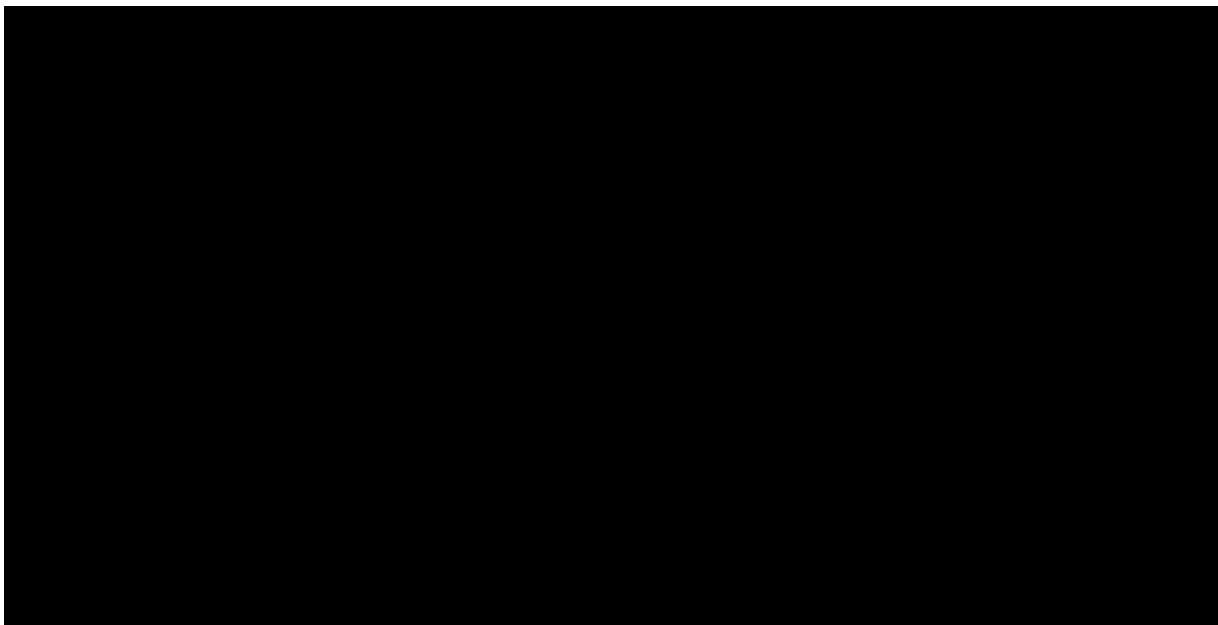
Dibuat sebanyak 20 data log histori KIRI dengan data pada tabel 5.1

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

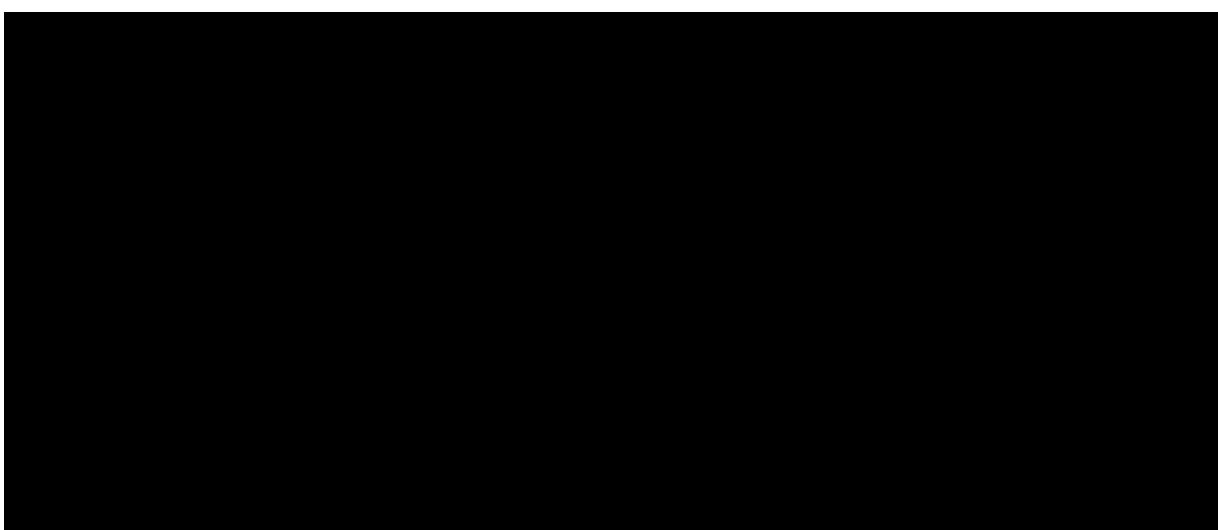
Tab 15.1: Data untuk Test Case Aplikasi Data Mining

Berikut hasil pengujian yang dilakukan dengan menggunakan data tersebut:

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



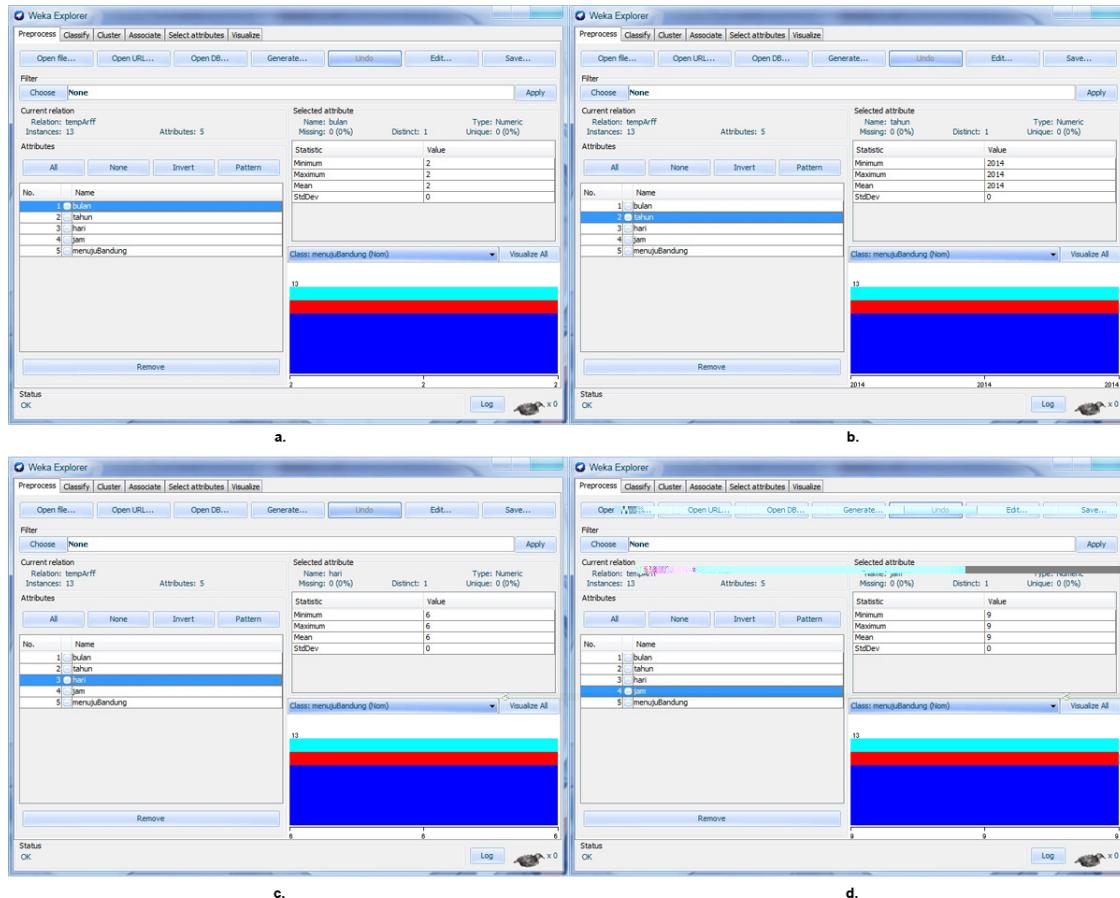
Gambar 5.6: Pengujian Mengambil Data CSV



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

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

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

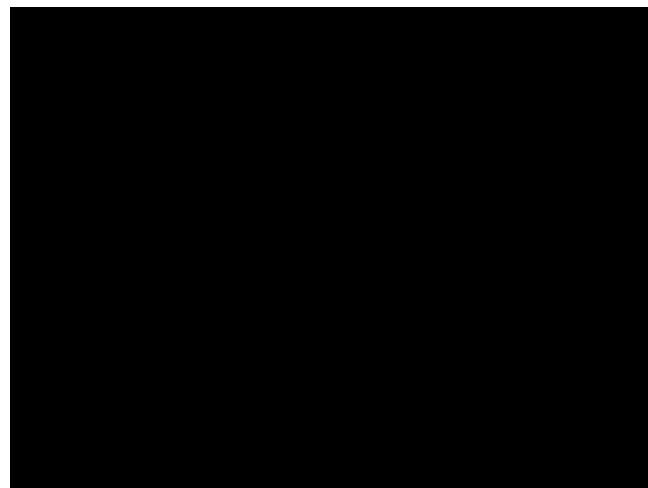


Gambar 5.8: Pengujian *Preprocessing Data*. Gambar a menunjukkan nilai pada atribut bulan. Gambar b menunjukkan nilai pada atribut tahun. Gambar c menunjukkan nilai pada atribut hari. Gambar d menunjukkan nilai pada atribut jam.

Terdapat dua tahap pertama pada *preprocessing data*, yaitu pengubahan waktu dari UTC menjadi GMT+7 dan klasifikasi arah. Pada tahap pengubahan waktu dari UTC menjadi GMT+7, pada gambar [5.8](#) d, atribut jam yang dimiliki menjadi ber nilai sembilan karena pada stcas, nilai atribut jam adalah dua. Pada tahap klasifikasi arah, dapat dilihat pada [5.2](#).

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

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

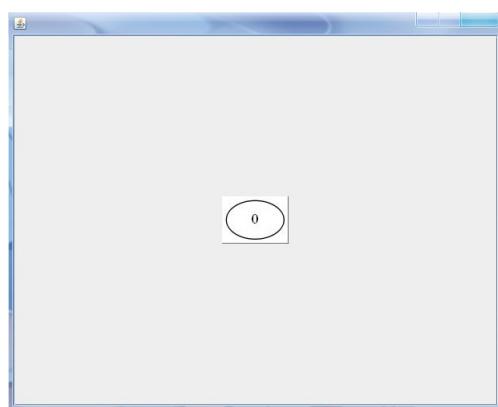


Gambar 5.9: Pengujian *Preprocessing Data* untuk Klasifikasi Data

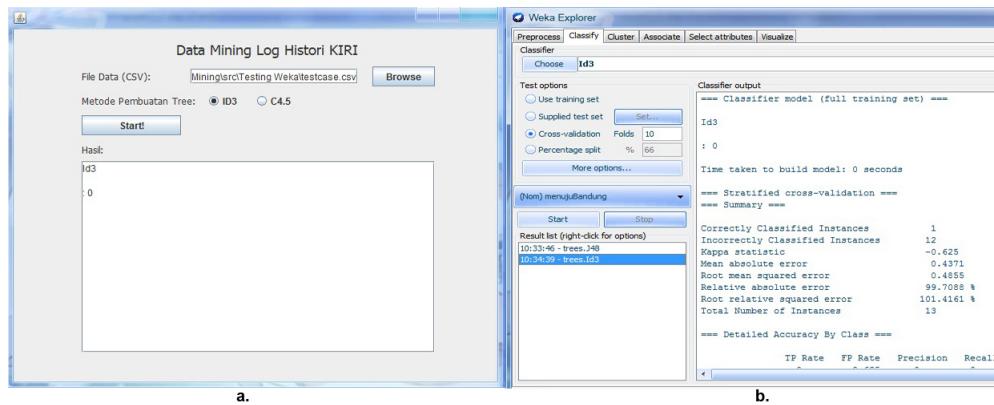
| Region Keberangkatan | Region Tujuan | Hasil Klasifikasi |
|----------------------|---------------|-------------------|
| 3 | 3 | 0 |
| 3 | 3 | 0 |
| 3 | 3 | 0 |
| 3 | 4 | 1 |
| 4 | 4 | 0 |
| 11 | 10 | -1 |
| 5 | 10 | 1 |
| 11 | 1 | -1 |
| 11 | 3 | -1 |
| 11 | 3 | -1 |
| 1 | 1 | 0 |
| 2 | 0 | -1 |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

Tab 15.2: Hasil Pengujian dan Klasifikasi

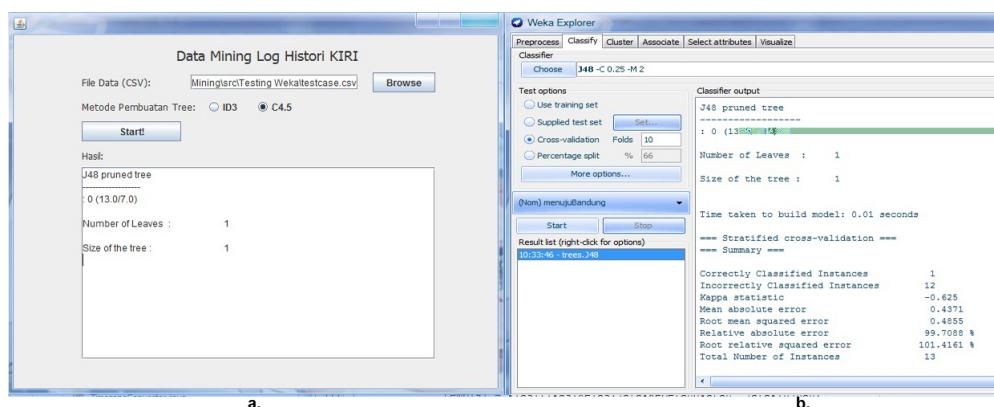
Untuk melakukan visualisasi *decision tree* dengan menggunakan graph dalam bahasa DOT, dapat dilihat pada gambar 5.12. Dari gambar tersebut, dapat disimpulkan bahwa pengubahan *decision tree* menjadi bahasa DOT telah berhasil.



Gambar 5.12: Pengujian Hasil Visualisasi dengan Menggunakan Bahasa DOT.



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



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

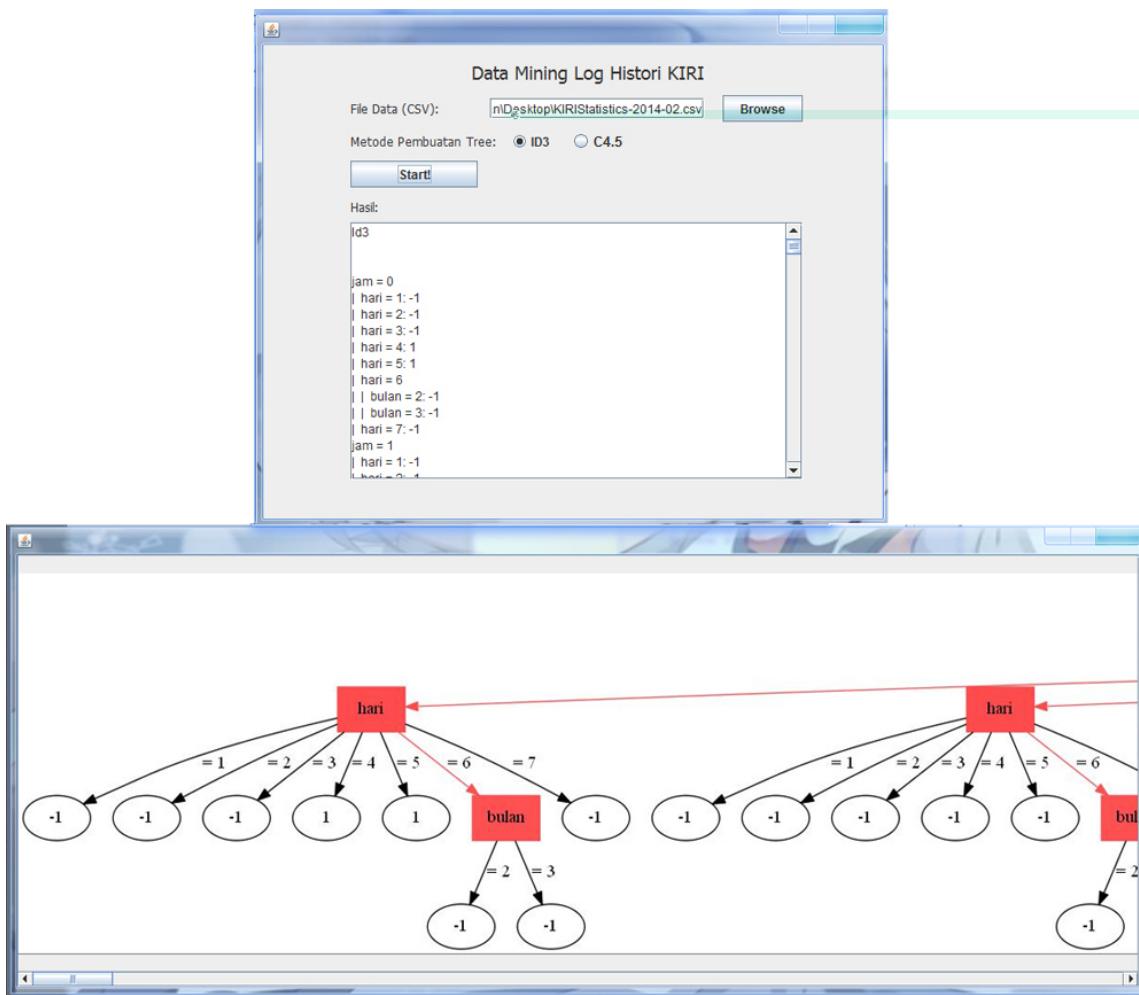
5.3.2 Pengujian Eksperimental

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

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

Gambar 5.13: Pengcobaan Data Mining untuk Melakukan Data Cleaning.

Setelah melakukannya data cleaning, program dapat berjalan dengan baik. Berikut hasil dari pengcobaan menggunakan ID3 pada gambar 5.14 dan menggunakan C4.5 pada gambar 5.15.



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

Gambar 5.15: Pengujian Data Mining dengan menggunakan Model C4.5 pada Log Histori KIRI pada Bulan 2 Tahun 2014.

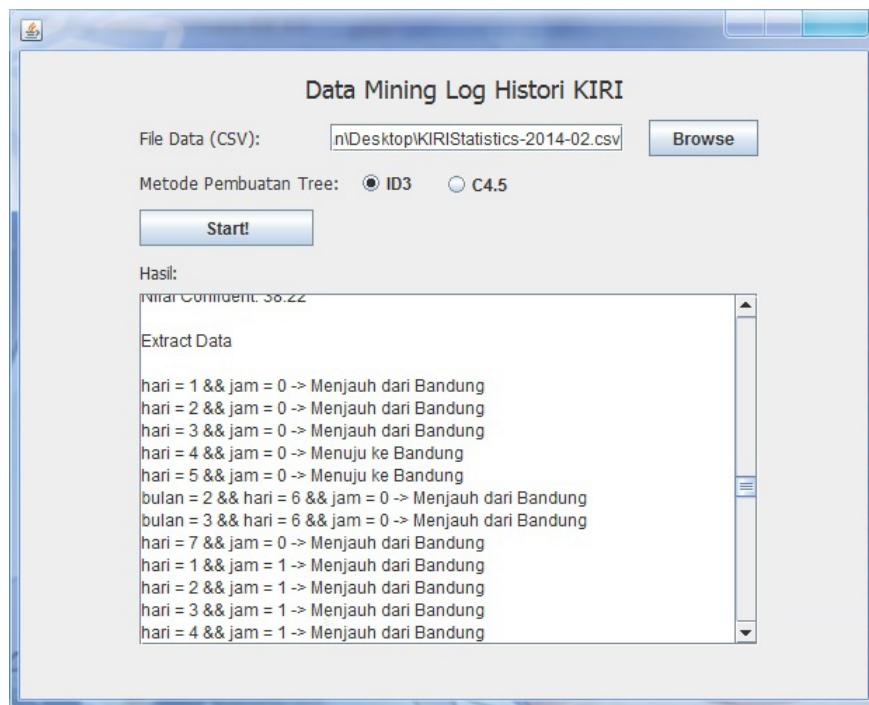
Pada kedua percobaan, terdapat perbedaan bulan, hal ini dikarenakan perubahan waktu dari UTC menjadi GMT+7 sejak 19 September 2014 hingga terdapat perubahan bulan atau tahun. Karena data pada bulan September lanjutnya hanya tujuh jam, maka hasil pada bulan September 1 lebih baik tidak dianggap karena data tersebut tidak cukup untuk merepresentasikan waktu satu bulan.

Hasil pengujian pada percobaan pembuatan *decision tree* pada model ID3 mengalami overfitting karena menghasilkan klasifikasi pada hampir setiap klasifikasi mungkin. Oleh karena itu, dapat disimpulkan bahwa *decision tree* dengan model ID3 menghasilkan *tree* yang jauh lebih besar namun tidak overfitting dan banyak user yang mengunjungi Bandung. Tetapi, *tree* yang dihasilkan cukup kecil. Dapat disimpulkan hasil *decision tree* pada percobaan di bulan ini cukup jauh.

Nilai akurasi dari pengujian dengan model ID3 adalah 38.22% sedangkan untuk percobaan dengan model C4.5 adalah 50.37%, dari sini dapat dinyatakan bahwa hasil *decision tree* dari C4.5 lebih tepat daripada ID3 dan model ID3 menghasilkan nilai akurasi yang belum memenuhi karena masih dibawah 50%.

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

Berikut hasil dari fungsi yang dibuat agar *decision tree* lebih mudah dibaca, dapat dilihat pada gambar 5.16. Dengan fungsi tersebut, diharapkan user dapat lebih mudah membaca *decision tree* yang dihasilkan.



Gambar 5.16: Hasil dari SDForExtractData

5.3.3 Analisis Hasil Uji

Berdasarkan pengujian di atas, dapat disimpulkan bahwa

1. Maka ID3 menghasilkan *decision tree* yang bersifat overfitting pada data log histori KIRI dengan *preprocessing data* dan klasifikasi yang sudah dilakukan pada bab 3.
2. Maka C4.5 menghasilkan *decision tree* yang lebih baik dan tetap daripada ID3, khususnya pada data log histori KIRI dengan *preprocessing data* dan klasifikasi yang sudah dilakukan pada bab 3 (C4.5 menghasilkan 50.37% sedangkan ID3 menghasilkan 38.22%).
3. Maka ID3 belum menghasilkan nilai akurasi yang memuaskan, karena masih dibawah 50%.
4. Dari data log histori KIRI pada bulan Februari 2014, *decision tree* dengan ID3 mengatakan bahwa user lebih sering menjauhi Bandung daripada mendekati Bandung atau berangkat menuju daerah yang sama.
5. *Decision tree* yang dihasilkan di bulan ini tidak membentuk rikan nilai lebih yang signifikan dibandingkan statistik biasa karena *decision tree* yang dihasilkan hanya memiliki satu leaf.

BAB 6

KESIMPULAN DAN SARAN

6.1 Kesimpulan

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

1. Salah satu cara untuk memperoleh pola yang menarik dan bermakna, dipergunakan pengolahan data dengan cara membuat klasifikasi dari data yang dihasilkan sesuai dengan tujuan yang ingin dicari. Pada penelitian ini dilakukan klasifikasi tujuan dari user apakah mereka ingin menuju Bandung atau ke luar Bandung atau masih berada di area yang sama seingga diperoleh pengaruh rakan user KIRI di Bandung.
2. Pembuatan perangkat lunak untuk melakukan *data mining* pada *log* histori KIRI dapat dilakukan.
3. Pola yang diperoleh dari data *log* histori KIRI adalah user sering pergi menuju Bandung pada bulan Februari 2014. Namun *decision tree* yang dihasilkan tidak memberikan nilai yang lebih signifikan dibanding statistik biasa.

6.2 Saran

Untuk pengembangan aplikasi *data mining log* histori KIRI lebih lanjut, dapat dilakukan dengan cara menggunakan klasifikasi yang lebih baik dan detail. Pada daannya dengan menggunakan klasifikasi yang lebih detail adalah hasil dari *decision tree* mungkin lebih besar (jika dibandingkan dengan *decision tree* yang dihasilkan dengan metode C4.5) namun diharapkan dapat menghasilkan nilai akurasi mencapai 75 persen.

DAFTAR REFERENSI

- [1] J. Han, M. Kamb r, and J. P i, *Data Mining : concepts and techniques, Third Edition*, p. 744. Els vi r, 2011.
- [2] C. V n ss, “Calculat distanc , b aring and mor b tw n latitud /longitud points.” <http://www.movable-type.co.uk/scripts/latlong.html>, 2002.
- [3] T. U. of Waikato, “W ka api.” <http://weka.sourceforge.net/doc.stable/>, 2009.
- [4] E. R. Gansn r, E. Koutsofios, and S. North, “Drawing graphs with dot.” <http://www.gphviz.org/pdf/dotguide.pdf>, January 2015.

LAMPIRAN A

| | | | | |
|--------|------------------|---------------|-------------|--|
| 113925 | A44EB361A179A49E | 2/1/2014 0:18 | SEARCHPLACE | kantor+po/10 |
| 113926 | A44EB361A179A49E | 2/1/2014 0:18 | SEARCHPLACE | kantor+pos/10 |
| 113927 | A44EB361A179A49E | 2/1/2014 0:18 | SEARCHPLACE | kantor+pos+ci/10 |
| 113928 | A44EB361A179A49E | 2/1/2014 0:18 | SEARCHPLACE | kantor+pos+cimahi/10 |
| 113929 | A44EB361A179A49E | 2/1/2014 0:18 | FINDROUTE | -6.7185828,107.0150728/- 6.918881548242062,107.60667476803064/1 |
| 113930 | A44EB361A179A49E | 2/1/2014 0:18 | FINDROUTE | -6.9015366,107.5414474/-6.88574,107.53816/1 |
| 113931 | E5D9904F0A8B4F99 | 2/1/2014 0:22 | PAGELOAD | /5.10.83.49/ |
| 113932 | E5D9904F0A8B4F99 | 2/1/2014 0:22 | PAGELOAD | /180.253.140.219/ |
| 113933 | E5D9904F0A8B4F99 | 2/1/2014 0:24 | PAGELOAD | /180.253.140.219/ |
| 113934 | E5D9904F0A8B4F99 | 2/1/2014 0:25 | PAGELOAD | /180.253.140.219/ |
| 113935 | E5D9904F0A8B4F99 | 2/1/2014 0:25 | FINDROUTE | -6.90608,107.61530/-6.89140,107.61060/2 |
| 113936 | E5D9904F0A8B4F99 | 2/1/2014 0:26 | PAGELOAD | /118.137.96.28/ |
| 113937 | E5D9904F0A8B4F99 | 2/1/2014 0:26 | FINDROUTE | -6.89459,107.58818/-6.89876,107.60886/2 |
| 113938 | E5D9904F0A8B4F99 | 2/1/2014 0:27 | FINDROUTE | -6.90608,107.61530/-6.89140,107.61060/2 |
| 113939 | E5D9904F0A8B4F99 | 2/1/2014 0:28 | FINDROUTE | -6.89977,107.62706/-6.89140,107.61060/2 |
| 113940 | E5D9904F0A8B4F99 | 2/1/2014 0:28 | FINDROUTE | -6.89459,107.58818/-6.86031,107.61287/2 |
| 113941 | D0AB08D956A351E4 | 2/1/2014 0:28 | FINDROUTE | -6.90598,107.59714/-6.90855,107.61082/1 |
| 113942 | A44EB361A179A49E | 2/1/2014 0:29 | FINDROUTE | -6.9172304,107.6042556/-6.92663,107.63644/1 |
| 113943 | A44EB361A179A49E | 2/1/2014 0:29 | FINDROUTE | -6.9172448,107.6042255/-6.92663,107.63644/1 |
| 113944 | D0AB08D956A351E4 | 2/1/2014 0:30 | FINDROUTE | -6.90598,107.59714/-6.90855,107.61082/1 |
| 113945 | D0AB08D956A351E4 | 2/1/2014 0:32 | FINDROUTE | -6.90598,107.59714/-6.90855,107.61082/1 |
| 113946 | D0AB08D956A351E4 | 2/1/2014 0:33 | FINDROUTE | -6.90598,107.59714/-6.90855,107.61082/1 |
| 113947 | A44EB361A179A49E | 2/1/2014 0:35 | SEARCHPLACE | jalan+asia+af/8 |
| 113948 | A44EB361A179A49E | 2/1/2014 0:35 | FINDROUTE | -6.9172448,107.6042255/-6.92163,107.61046/1 |
| 113949 | A44EB361A179A49E | 2/1/2014 0:35 | SEARCHPLACE | taman+fotog/10 |

| | | | | |
|--------|------------------|---------------|-----------|---|
| 113950 | A44EB361A179A49E | 2/1/2014 0:36 | FINDROUTE | -6.917321,107.6043132/- 6.921568846707516,107.61015225201845/1 |
| 113951 | E5D9904F0A8B4F99 | 2/1/2014 0:38 | PAGELOAD | /5.10.83.68/ |
| 113952 | E5D9904F0A8B4F99 | 2/1/2014 0:38 | PAGELOAD | /5.10.83.28/ |
| 113953 | E5D9904F0A8B4F99 | 2/1/2014 0:40 | | |

| | | | | |
|--------|------------------|---------------|-------------|---|
| 113976 | E5D9904F0A8B4F99 | 2/1/2014 1:25 | PAGELOAD | /5.10.83.24/ |
| 113977 | E5D9904F0A8B4F99 | 2/1/2014 1:25 | FINDROUTE | -6.91485,107.59123/-6.91593,107.65588/1 |
| 113978 | E5D9904F0A8B4F99 | 2/1/2014 1:26 | PAGELOAD | /5.10.83.82/ |
| 113979 | E5D9904F0A8B4F99 | 2/1/2014 1:28 | FINDROUTE | -6.91593,107.65588/-6.91485,107.59123/1 |
| 113980 | A44EB361A179A49E | 2/1/2014 1:29 | FINDROUTE | -6.9250709,107.6204635/-6.91728,107.60417/1 |
| 113981 | A44EB361A179A49E | 2/1/2014 1:35 | FINDROUTE | -6.9252132,107.6200288/-6.91728,107.60417/1 |
| 113982 | A44EB361A179A49E | 2/1/2014 1:36 | FINDROUTE | -6.922427886995373,107.61768691241741/-6.91728,107.60417/1 |
| 113983 | E5D9904F0A8B4F99 | 2/1/2014 1:36 | FINDROUTE | -6.91431,107.63921/-6.94024,107.71550/1 |
| 113984 | E5D9904F0A8B4F99 | 2/1/2014 1:37 | PAGELOAD | /5.10.83.98/ |
| 113985 | A44EB361A179A49E | 2/1/2014 1:37 | FINDROUTE | -6.921635413232821,107.61909071356058/-6.91728,107.60417/1 |
| 113986 | E5D9904F0A8B4F99 | 2/1/2014 1:38 | FINDROUTE | -6.88936,107.57533/-6.92600,107.63628/1 |
| 113987 | E5D9904F0A8B4F99 | 2/1/2014 1:39 | PAGELOAD | http://www.kiri.trav1/m/r/?qs=trans+studi... |
| 113988 | E5D9904F0A8B4F99 | 2/1/2014 1:39 | FINDROUTE | -6.92600,107.63628/-6.88936,107.57533/1 |
| 113989 | A44EB361A179A49E | 2/1/2014 1:41 | SEARCHPLACE | t riminal+ta/10 |
| 113990 | A44EB361A179A49E | 2/1/2014 1:41 | FINDROUTE | -6.9158359,107.6101751/-6.90658,107.61623/1 |
| 113991 | A44EB361A179A49E | 2/1/2014 1:42 | FINDROUTE | -6.9158359,107.6101751/-6.90658,107.61623/1 |
| 113992 | D0AB08D956A351E4 | 2/1/2014 1:50 | FINDROUTE | -6.38355,106.919975/-7.08933734335005,107.562576737255/1 |
| 113993 | A44EB361A179A49E | 2/1/2014 1:51 | SEARCHPLACE | taman+ci/10 |
| 113994 | A44EB361A179A49E | 2/1/2014 1:51 | SEARCHPLACE | taman+cilaki/10 |
| 113995 | E5D9904F0A8B4F99 | 2/1/2014 1:52 | PAGELOAD | /206.53.152.33/m |
| 113996 | E5D9904F0A8B4F99 | 2/1/2014 1:52 | FINDROUTE | -6.90598,107.59714/-6.91728,107.60417/1 |
| 113997 | A44EB361A179A49E | 2/1/2014 1:54 | FINDROUTE | -6.901306,107.6214169/-6.90336,107.62235/1 |
| 113998 | A44EB361A179A49E | 2/1/2014 1:54 | FINDROUTE | -6.901306,107.6214169/-6.90336,107.62235/1 |
| 113999 | E5D9904F0A8B4F99 | 2/1/2014 | PAGELOAD | /5.10.83.27/ |

| | | | | |
|--------|------------------|---------------|-------------|---|
| 114000 | 308201BB30820124 | 2/1/2014 1:15 | SEARCHPLACE | riau+jucnction/10 |
| 114001 | 308201BB30820124 | 2/1/2014 1:56 | FINDROUTE | -6.90687,107.61239/-6.89032,107.57961/2 |
| 114002 | E5D9904F0A8B4F99 | 2/1/2014 1:57 | PAGELOAD | /118.99.112.66/ |
| 114003 | 308201BB30820124 | 2/1/2014 1:57 | FINDROUTE | -6.90687,107.61239/-6.90159,107.60442/1 |
| 114004 | 308201BB30820124 | 2/1/2014 1:57 | FINDROUTE | -6.90687,107.61239/-6.89032,107.57961/2 |
| 114005 | E5D9904F0A8B4F99 | 2/1/2014 1:58 | FINDROUTE | -6.88211,107.60378/-6.90774,107.60908/1 |
| 114006 | A44EB361A179A49E | 2/1/2014 1:59 | FINDROUTE | -6.9212516,107.6196466/-6.91728,107.60417/1 |
| 114007 | 308201BB30820124 | 2/1/2014 1:59 | FINDROUTE | -6.90687,107.61239/-6.91486,107.60824/1 |
| 114008 | 687C44EB2424285D | 2/1/2014 1:59 | WIDGETLOAD | http://www.cndkialadrshipschool.sc... |
| 114009 | E5D9904F0A8B4F99 | 2/1/2014 2:00 | FINDROUTE | -6.88166,107.61561/-6.90774,107.60908/1 |

LAMPIRAN B

THE SOURCE CODE

Listing B.1: Controll r.java

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

```

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

Listing B.2: Vi w.java

```

1 package DataMiningLogHistoriKIRI;
2
3 import java.io.File;
4 import java.io.IOException;
5 import java.util.logging.Level;
6 import java.util.logging.Logger;
7 import javax.swing.JFileChooser;
8
9 /**
10 *
11 * @author Jovan Gunawan
12 */
13 public class View extends javax.swing.JPanel {
14
15     /**
16      * Creates new form View
17      */
18     public View(Controller cont) {
19         this.cont = cont;
20         initComponents();
21         buttonGroup1.add(radioButtonId3);
22         buttonGroup1.add(radioButtonC45);
23     }
24
25     /**
26      * This method is called from within the constructor to initialize the form.
27     */
28 }
```



```

119         .addComponent(labelHasil, javax.swing.GroupLayout.PREFERRED_SIZE, 89,
120                     javax.swing.GroupLayout.PREFERRED_SIZE)
121         .addComponent(scrollPanel, javax.swing.GroupLayout.DEFAULT_SIZE, 441,
122                     Short.MAX_VALUE)
123         .addComponent(labelKeterangan, javax.swing.GroupLayout.DEFAULT_SIZE,
124                     javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
125     .addGap(0, 0, Short.MAX_VALUE)))
126 );
127 layout.setVerticalGroup(
128   layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
129   .addGroup(layout.createSequentialGroup()
130     .addComponent(judul, javax.swing.GroupLayout.PREFERRED_SIZE, 31, javax.swing.GroupLayout.PREFERRED_SIZE)
131     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
132     .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
133       .addComponent(labelFileData, javax.swing.GroupLayout.PREFERRED_SIZE, 26, javax.swing.GroupLayout.PREFERRED_SIZE)
134       .addComponent(textFieldFilePath, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
135       .addComponent(buttonBrowse)))
136     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
137     .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
138       .addComponent(labelPemilihanMethod, javax.swing.GroupLayout.PREFERRED_SIZE, 26, javax.swing.GroupLayout.PREFERRED_SIZE)
139       .addComponent(radioButtonC45))
140     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
141     .addComponent(buttonStart)
142     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
143     .addComponent(labelHasil, javax.swing.GroupLayout.PREFERRED_SIZE, 26, javax.swing.GroupLayout.PREFERRED_SIZE)
144     .addGap(2, 2, 2)
145     .addComponent(scrollPanel, javax.swing.GroupLayout.PREFERRED_SIZE, 251, javax.swing.GroupLayout.PREFERRED_SIZE)
146     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 10, Short.MAX_VALUE)
147     .addComponent(labelKeterangan, javax.swing.GroupLayout.PREFERRED_SIZE, 26, javax.swing.GroupLayout.PREFERRED_SIZE)
148   )
149 );
150 // </editor-fold>
151
152 private void buttonBrowseActionPerformed(java.awt.event.ActionEvent evt) {
153   final JFileChooser fc = new JFileChooser();
154   int returnVal = fc.showOpenDialog(buttonStart);
155
156   if (returnVal == JFileChooser.APPROVE_OPTION) {
157     File file = fc.getSelectedFile();
158     textFieldFilePath.setText(file.getPath());
159   } else {
160     System.out.println("Error in capturing the path");
161     System.exit(1);
162   }
163
164 private void buttonStartActionPerformed(java.awt.event.ActionEvent evt) {
165   String inputPath = textFieldFilePath.getText();
166   String miningAlgo = "";
167   if (radioButtonId3.isSelected())
168   {
169     miningAlgo = "id3";
170   }
171   else
172   {
173     miningAlgo = "c45";
174   }
175
176   try {
177     cont.startMining(inputPath, miningAlgo, labelKeterangan, hasil);
178   } catch (IOException e)
179   {
180     System.out.println("Error start mining");
181     System.exit(1);
182   }
183 }
184
185 private void radioButtonC45ActionPerformed(java.awt.event.ActionEvent evt) {
186   // TODO add your handling code here:
187 }
188
189 // Variables declaration - do not modify
190 private javax.swing.JButton buttonBrowse;
191 private javax.swing.ButtonGroup buttonGroup1;
192 private javax.swing.JButton buttonStart;
193 private javax.swing.JTextArea hasil;
194 private javax.swing.JLabel judul;
195 private javax.swing.JLabel labelFileData;
196 private javax.swing.JLabel labelHasil;
197 private javax.swing.JLabel labelKeterangan;
198 private javax.swing.JLabel labelPemilihanMethod;
199 private javax.swing.JRadioButton radioButtonId3;
200 private javax.swing.JRadioButton radioButtonC45;
201 private javax.swing.JScrollPane scrollPanel;
202 private javax.swing.JTextField textFieldFilePath;
203 // End of variables declaration
204 private Controller cont;
205 }

```

Listing B.3: CSVR ad r.java

```

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

```

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

Listing B.4: ProcessingData.java

```

1 package DataMiningLogHistoriKIRI;
2
3 import java.util.ArrayList;
4
5 /**
6 * 
7 * @author Jovan Gunawan
8 */
9 public class ProcessingData
10 {
11     ProcessingData()
12     {
13     }
14     public void processSorting(ArrayList array, ArrayList data, String action)
15     {
16         for(int i = 0; i < data.size(); i++)
17         {
18             String [] tempData = (String [])data.get(i);
19             if(tempData[3].equals(action))
20             {
21                 array.add(tempData);
22             }
23         }
24     }
25     public ArrayList preprocessingData(ArrayList<String[]> data, int[] maxMin)
26     {
27         // 2 int[] untuk mendapatkan nilai max dan min dari variabel bulan dan tahun yang digunakan untuk
28         // inisialisasi max min pada kelas SDForExtractData.
29         int []max = new int [2];
30         int []min = new int [2];
31         max[0] = 0;
32         max[1] = 0;
33         min[0] = Integer.MAX_VALUE;
34         min[1] = Integer.MAX_VALUE;
35
36         ArrayList<int []> result = new ArrayList<int []>();
37
38         // tahap pertama: ubah waktu dari UTC ke GMT+7
39         for(int i = 0; i < data.size(); i++)
40         {
41             //System.out.println(data.get(i)[3]);
42             data.get(i)[2] = TimezoneConverter.convertToGMT7(data.get(i)[2]);
43         }
44
45         // tahap kedua sampai kesembilan
46         DistanceHaversine haversine = new DistanceHaversine();
47         for(int i = 0; i < data.size(); i++)
48         {
49             //cek apakah format sudah benar atau belum
50             if(data.get(i)[4].split(",").length == 3)
51             {
52                 // tahap kedua: pecah string atribut tanggal
53                 int [] temp = new int [5];
54                 String [] splitted = data.get(i)[2].split(",");
55                 temp[2] = Integer.parseInt(splitted[0]);
56             }
57         }
58     }
59 }
```

```
56 // tahap ketiga: pecah nilai string yang tanggal
57 String[] splitted2 = splitted[1].split("/");
58 temp[0] = Integer.parseInt(splitted2[0]);
59 temp[1] = Integer.parseInt(splitted2[1]);
60 // tahap keempat: pecah nilai string yang jam
61 splitted2 = splitted[2].split(":");
62 temp[3] = Integer.parseInt(splitted2[0]);
63 // tahap kelima: pecah string atribut additional data
64 splitted = data.get(i)[4].split("/");
65 // tahap keenam: pecah lokasi keberangkatan dan lokasi tujuan untuk mendapatkan lat n lon
66 // dan (ini tahap ketujuh) menghitung jarak terhadap titik pusat
67 splitted2 = splitted[0].split(",");
68 double jarakKeberangkatan = haversine.calculateDistance(Double.parseDouble(splitted2[0]),
69 Double.parseDouble(splitted2[1]), -6.916667, 107.6) * 1000;
70 splitted2 = splitted[1].split(",");
71 double jarakTujuan = haversine.calculateDistance(Double.parseDouble(splitted2[0]), Double.
72 parseDouble(splitted2[1]), -6.916667, 107.6) * 1000;
73 // tahap kedelapan, set semua data ke array
74 temp[4] = klasifikasiKelas(jarakKeberangkatan, jarakTujuan);
75
76 if(temp[4] != -2)
77 {
78     result.add(temp);
79     // proses untuk mencatat nilai max dan min
80     if(max[0] < temp[0])
81     {
82         max[0] = temp[0];
83     }
84     if(min[0] > temp[0])
85     {
86         min[0] = temp[0];
87     }
88     if(max[1] < temp[1])
89     {
90         max[1] = temp[1];
91     }
92     if(min[1] > temp[1])
93     {
94         min[1] = temp[1];
95     }
96 }
```

```

3| import java.text.ParseException;
4| import java.text.SimpleDateFormat;
5| import java.util.Date;
6| import java.util.TimeZone;
7| import java.util.logging.Level;
8| import java.util.logging.Logger;
9|
10| /**
11|  * 
12|  * @author Jovan Gunawan
13|  */
14| public class TimezoneConverter
15| {
16|     // Mengubah input waktu menjadi waktu GMT+7
17|     // Jika ingin mengubah dari UTC ke GMT+7 maka komputer harus set timezone ke UTC
18|     // input parameter string harus dalam format dd-MM-yyyy HH:mm:ss --> contoh 1/1/2014 3:51:15 AM
19|     // output akan mengubah waktu menjadi GMT+7 dalam String dengan format EEE MM/dd/yyyy HH:mm:ss -->
20|     // contoh Wed 01/01/2014 03:51:15
21|     public static String convertToGMT7(String date)
22|     {
23|         Date d = null;
24|         long milliseconds = 0;
25|
26|         try {
27|             SimpleDateFormat f = new SimpleDateFormat("MM/dd/yyyy HH:mm");
28|             d = (Date)f.parse(date);
29|         } catch (ParseException ex) {
30|             SimpleDateFormat f = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
31|             try {
32|                 d = (Date)f.parse(date);
33|             } catch (ParseException ex1) {
34|                 Logger.getLogger(TimezoneConverter.class.getName()).log(Level.SEVERE, null, ex);
35|                 System.exit(1);
36|             }
37|         }
38|         milliseconds = d.getTime();
39|
40|         Date currentTime = new Date(milliseconds);
41|         // untuk hari --> 1 = senin, ..., 7 = minggu
42|         SimpleDateFormat sdf = new SimpleDateFormat("u MM/dd/yyyy HH:mm:ss");
43|
44|         sdf.setTimeZone(TimeZone.getTimeZone("GMT+7"));
45|         return sdf.format(currentTime);
46|     }
47| }

```

Listing B.6: Distanc Hav rsin .java

```

1| package DataMiningLogHistoriKIRI;
2|
3| /**
4|  * 
5|  * @author Jovan Gunawan
6|  */
7| public class DistanceHaversine
8| {
9|     private double r;
10|     public DistanceHaversine()
11|     {
12|         r = 6.371;
13|     }
14|
15|     public double calculateDistance(double latitude1, double longitude1, double latitude2, double
16|                                     longitude2)
17|     {
18|         latitude1 = Math.toRadians(latitude1);
19|         longitude1 = Math.toRadians(longitude1);
20|         latitude2 = Math.toRadians(latitude2);
21|         longitude2 = Math.toRadians(longitude2);
22|
23|         double dlon = longitude2 - longitude1;
24|         double dlat = latitude2 - latitude1;
25|
26|         double a = Math.pow((Math.sin(dlat/2)),2) + Math.cos(latitude1) * Math.cos(latitude2) * Math.pow(
27|             Math.sin(dlon/2),2);
28|
29|         double c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
30|         return r * c;
31|     }
32| }

```

Listing B.7: ArffIO.java

```

1| package DataMiningLogHistoriKIRI;
2|
3| import java.io.BufferedReader;
4| import java.io.BufferedWriter;
5| import java.io.FileReader;
6| import java.io.FileWriter;
7| import java.io.IOException;
8| import java.io.PrintWriter;
9| import java.util.ArrayList;
10| import java.util.logging.Level;
11| import java.util.logging.Logger;
12| import weka.core.Instances;
13|
14| /**

```

```

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

```

Listing B.8: DecisionTree.java

```

1 package DataMiningLogHistoriKIRI;
2
3 import java.util.logging.Level;
4 import java.util.logging.Logger;
5 import weka.classifiers.Classifier;
6 import weka.classifiers.trees.Id3;
7 import weka.classifiers.trees.J48;
8 import weka.core.Instances;
9 import weka.filters.Filter;
10 import weka.filters.unsupervised.attribute.NumericToNominal;
11
12 /**
13  *
14  * @author Jovan Gunawan
15  */
16 public class DecisionTree
17 {
18     private Classifier tree;
19
20     public double calculatePrecision(Instances arff)
21     {
22         int nilaiBenar = 0, resultInt;
23         float result = 0;
24         for (int i = 0; i < arff.numInstances(); i++)
25         {
26             try {
27                 result = (float) tree.classifyInstance(arff.instance(i));
28                 resultInt = Math.round(result)-1;
29                 if(resultInt == Integer.parseInt(arff.instance(i).stringValue(4)))
30                 {
31                     nilaiBenar++;
32                 }
33             } catch (Exception ex) {
34             }
35         }
36         double confident = Math.round(nilaiBenar * 1.0 / arff.numInstances() * 10000) / 100.0;
37         return confident;
38     }
39
40     public String id3(Instances arff)
41     {
42         tree = new Id3();
43         try {
44             NumericToNominal convert= new NumericToNominal();
45             String [] options= new String [2];
46             options[0] = "-R";
47             options[1] = "1-4";
48
49             convert.setOptions(options);
50             convert.setInputFormat(arff);
51
52             Instances newData=Filter.useFilter(arff, convert);
53

```

```

54         tree.buildClassifier(newData);
55     } catch (Exception ex) {
56         Logger.getLogger(Controller.class.getName()).log(Level.SEVERE, null, ex);
57     }
58
59     return tree.toString();
60 }
61
62 public String j48(Instances arff)
63 {
64     tree = new J48();
65     try {
66         tree.buildClassifier(arff);
67     } catch (Exception ex) {
68         Logger.getLogger(Controller.class.getName()).log(Level.SEVERE, null, ex);
69     }
70
71     return tree.toString();
72 }
73 }
```

Listing B.9: DotConv rt r.java

```

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

```

73}
74}
75tempExtract.extract();
76extract.addStringResult(tempExtract.getList());
77// menghasilkan daun
78
79nodeName2 = data.getDataNumber(temp[3]);
80result += nodeName2 + "\n["label="" + temp[1] + "\n" + temp[2].substring(0, temp[2].
81length() - 1) + "\n"]\n";
82if(iniName1 && loop == 0)
83{
84    result += nodeName1 + "\n["label="" + temp[0] + "\n", shape=box, style=filled,
85    color="\n1.0\n" + color + "\n1.0\n"]\n";
86    result += nodeName2 + "\n["label="" + temp[3] + "\n"]\n";
87    data.buangArrayPertama();
88}
89else
90{
91    // masukin data bwt extract
92    tempExtract.setRules(temp[0], temp[1], Integer.parseInt(temp[2]));
93    // menghasilkan node
94    String [] temp2;
95    temp2 = data.getData(1).split("\n");
96    temp2 = temp2[deep+1].split("\n");
97    nodeName2 = data.getDataNumber(temp2[0]);
98    result += nodeName2 + "\n["label="" + temp[1] + "\n" + temp[2] + "\n", shape=box,
99    style=filled, color="\n1.0\n" + color + "\n1.0\n"]\n";
100   data.buangArrayPertama();
101
102   SDForExtractData newExtract = new SDForExtractData(tempExtract);
103   result += DotConverter.convert(data, newExtract, miningAlgo, deep+1, nodeName2);
104
105   if(iniName1 && loop == 0)
106   {
107       result += nodeName1 + "\n["label="" + temp[0] + "\n", shape=box, style=filled,
108       color="\n1.0\n" + color + "\n1.0\n"]\n";
109   }
110   result += nodeName2 + "\n["label="" + temp2[0] + "\n", shape=box, style=filled, color
111   ="\n1.0\n" + color + "\n1.0\n"]\n";
112   extract.addStringResult(newExtract.getList());

```

```

165         result += nodeName1 + "[label=\"" + temp[0] + "\",shape=box,style=filled ,
166             color=\"1.0\" + color + "1.0\"]\n";
167     }
168     result += nodeName2 + "[label=\"" + temp[3] + "\"]\n";
169     data.buangArrayPertama();
170 }
171 else
172 {
173     // masukin data bwt extract
174     tempExtract.setRules(temp[0], temp[1], Integer.parseInt(temp[2]));
175     // menghasilkan node
176     String [] temp2;
177     temp2 = data.getData(1).split("~~~");
178     temp2 = temp2[deep+1].split("~~");
179     nodeName2 = data.getDataNumber(temp2[0]);
180     result += nodeName2 + "[label=\"" + temp[1] + "~~" + temp[2] + "\",shape=box,
181             style=filled,color=\"1.0\" + color + "1.0"]\n";
182     data.buangArrayPertama();
183
184     SDForExtractData newExtract = new SDForExtractData(tempExtract);
185     result += DotConverter.convert(data, newExtract, miningAlgo, deep+1, nodeName2);
186
187     if(iniName1 && i == 0)
188     {
189         result += nodeName1 + "[label=\"" + temp[0] + "\",shape=box,style=filled ,
190             color=\"1.0\" + color + "1.0\"]\n";
191     }
192     result += nodeName2 + "[label=\"" + temp2[0] + "\"]\n",shape=box,style=filled ,color
193     ="\\"1.0\" + color + "1.0"]\n";
194     extract.addStringResult(newExtract.getList());
195 }
196 }
197 }
198
199 }
```

Listing B.10: SDForConv rtTr .java

```

1 package DataMiningLogHistoriKIRI;
2
3 /**
4 *
5 * @author Jovan Gunawan
6 * Class ini dibuat untuk struktur data yang digunakan untuk mengubah hasil output dari weka menjadi
7 * gambar dengan bahasa DOT
8 *
9 * count akan digunakan untuk inisialisasi nomor data --> contoh tanggal1, tanggal 2, .... etc, angka
10 * tersebut yang akan ditaruh pada array tersebut
11 * array count akan digunakan sebagai berikut --> [0] = tanggal, [1] = bulan, [2] = tahun, [3] = hari, [4]
12 * = jam, [5] = menit, [6] = nilai 0, [7] = nilai 1, [8] = nilai 2
13 */
14 public class SDForConvertTree
15 {
16     private String [] data;
17     private int [] count;
18
19     public SDForConvertTree(String [] data)
20     {
21         this.data = data;
22         count = new int[9];
23         for(int i = 0; i < 9; i++)
24         {
25             count[i] = 0;
26         }
27
28         public void setData(String data, int index)
29         {
30             this.data[index] = data;
31         }
32         public String [] getData()
33         {
34             return data;
35         }
36         public String getData(int index)
37         {
38             return this.data[index];
39         }
40         public void setCount(int count, int index)
41         {
42             this.count[index] = count;
43         }
44         public int getCount(int index)
45         {
46             int temp = this.count[index];
47             this.count[index]++;
48             return temp;
49         }
50         public boolean hasNext()
51         {
52             if(this.data.length > 0)
53             {
54                 return true;
55             }
56         }
57     }
58 }
```

```
56|         {  
57|             }      return false;  
58|         }  
59|     }
```

```

34         this.maxBatasAtas[i] = max[i];
35         this.batasAtas[i] = max[i];
36         this.minBatasBawah[i] = min[i];
37         this.batasBawah[i] = min[i];
38         check[i] = false;
39     }
40 }
41
42 public SDForExtractData(SDForExtractData data)
43 {
44     String[] atribut = data.getAtribut();
45     int[] max = data.getMaxBatasAtas();
46     int[] min = data.getMinBatasBawah();
47     int[] bmax = data.getBatasAtas();
48     int[] bmin = data.getBatasBawah();
49     boolean[] check = data.getCheck();
50     int length = data.getAtribut().length;
51     list = data.getList();
52
53     this.atribut = new String[length];
54     this.maxBatasAtas = new int[length];
55     this.minBatasBawah = new int[length];
56     this.batasAtas = new int[length];
57     this.batasBawah = new int[length];
58     this.check = new boolean[length];
59
60     for(int i = 0; i < length; i++)
61     {
62         this.atribut[i] = atribut[i];
63         this.maxBatasAtas[i] = max[i];
64         this.batasAtas[i] = bmax[i];
65         this.minBatasBawah[i] = min[i];
66         this.batasBawah[i] = bmin[i];
67         this.check[i] = check[i];
68     }
69 }
70
71 public void setKelas(int kelas)
72 {
73     this.kelas = kelas;
74 }
75
76 public void setRules(String atribut, String rulesOperation, int value)
77 {
78     int index = 0;
79     for(int i = 0; i < this.atribut.length; i++)
80     {
81         if(this.atribut[i].equals(atribut))
82         {
83             index = i;
84             break;
85         }
86     }
87     check[index] = true;
88
89     if(rulesOperation.equals("<="))
90     {
91         batasAtas[index] = value;
92     }
93     else if(rulesOperation.equals("<"))
94     {
95         batasAtas[index] = value - 1;
96     }
97     else if(rulesOperation.equals(">="))
98     {
99         batasBawah[index] = value;
100    }
101    else if(rulesOperation.equals(">"))
102    {
103        batasBawah[index] = value + 1;
104    }
105    else if(rulesOperation.equals("!="))
106    {
107        batasAtas[index] = value;
108        batasBawah[index] = value;
109    }
110 }
111
112 public void extract()
113 {
114     String[] hari = new String[]{"Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu", "Minggu"};
115     boolean in = false;
116     String result = "";
117     for(int i = 0; i < atribut.length; i++)
118     {
119         if(check[i])
120         {
121             if(in)
122             {
123                 result += "&&";
124             }
125             if(batasBawah[i] == batasAtas[i])
126             {
127                 result += atribut[i] + "==" + batasAtas[i];
128             }
129             else
130             {
131                 result += atribut[i] + ":" + batasBawah[i] + "-=" + batasAtas[i];
132             }
133         }
134     }
135 }

```

```

133         in = true;
134     }
135 }
136
137 if(kelas == 1)
138 {
139     result += "=>_Menuju_ke_Bandung";
140 }
141 else if(kelas == 0)
142 {
143     result += "=>_Menuju_daerah_yang_sama";
144 }
145 else
146 {
147     result += "=>_Menjauh_dari_Bandung";
148 }
149 list.add(result);
150 }
151
152 public void addStringResult(ArrayList<String> result)
153 {
154     list = result;
155 }
156
157 public String[] getAtribut() {
158     return atribut;
159 }
160
161 public boolean[] getCheck() {
162     return check;
163 }
164
165 public int[] getBatasAtas() {
166     return batasAtas;
167 }
168
169 public int[] getBatasBawah() {
170     return batasBawah;
171 }
172
173 public int[] getMaxBatasAtas() {
174     return maxBatasAtas;
175 }
176
177 public int[] getMinBatasBawah() {
178     return minBatasBawah;
179 }
180
181 public int getKelas() {
182     return kelas;
183 }
184
185 public ArrayList<String> getList() {
186     return list;
187 }
188 }
189 }
```

Listing B.12: CMD.java

```

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