

**PENENTUAN FITUR YANG RELEVAN TERHADAP PREDIKSI
CACAT PERANGKAT LUNAK BERDASARKAN
SELEKSI FITUR GAIN RATIO**

SKRIPSI

Oleh :
DIKO ANDRI VIDIAN
NIM. 14650072



**JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2018**

**PENENTUAN FITUR YANG RELEVAN TERHADAP PREDIKSI
CACAT PERANGKAT LUNAK BERDASARKAN
SELEKSI FITUR GAIN RATIO**

SKRIPSI

**Diajukan Kepada:
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang
Untuk Memenuhi Salah Satu Persyaratan Dalam
Memperoleh Gelar Sarjana Komputer (S.Kom)**

**Oleh :
DIKO ANDRI VIDIAN
NIM. 14650072**

**JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK BRAHIM
MALANG
2018**

**PENENTUAN FITUR YANG RELEVAN TERHADAP PREDIKSI
CACAT PERANGKAT LUNAK BERDASARKAN
SELEKSI FITUR GAIN RATIO**

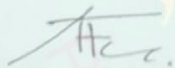
SKRIPSI

Oleh :
DIKO ANDRI VIDIAN
NIM. 14650072

Telah Diperiksa dan Disetujui untuk diuji:
Tanggal, Juni 2018

Dosen Pembimbing I

Dosen Pembimbing II



Fatchurrochman, M.Kom
NIP. 19700731 200501 1 002



Ajib Hanani, M.T
NIDT. 19840731 20160801 1 076

Mengetahui,

Ketua Jurusan Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang





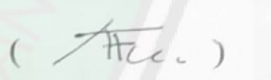

Dr. Cahyo Crysdian
NIP. 19740424 200901 1 008

**PENENTUAN FITUR YANG RELEVAN TERHADAP PREDIKSI
CACAT PERANGKAT LUNAK BERDASARKAN
SELEKSI FITUR GAIN RATIO**

SKRIPSI


Oleh:
DIKO ANDRI VIDIAN
NIM. 14650072

Telah Dipertahankan di Depan Dewan Penguji Skripsi dan
Dinyatakan Diterima Sebagai Salah Satu Pernyataan Untuk
Memperoleh Gelar Sarjana Komputer (S.Kom)
Tanggal: Juni 2018

Susunan Dewan Penguji	Tanda Tangan
Penguji Utama : <u>Irwan Budi Santoso, M.Kom</u> NIP. 19770103 201101 1 004	()
Ketua Penguji : <u>Supriyono, M.Kom</u> NIDT. 19841010 20160801 1 078	()
Sekretaris Penguji : <u>Fatchurrochman, M.Kom</u> NIP. 19700731 200501 1 002	()
Anggota Penguji : <u>Ajib Hanani, M.T</u> NIDT. 19840731 20160801 1 076	()

Mengetahui dan Mengesahkan,
Ketua Jurusan Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang




Cahyo Crysdian
NIP. 19740424 200901 1 008

HALAMAN PERNYATAAN KEASLIAN TULISAN

Saya yang bertanda tangan dibawah ini,

Nama : Diko Andri Vidian

NIM : 14650072

Jurusan : Teknik Informatika

Fakultas : Sains dan Teknologi

Menyatakan dengan sebenarnya bahwa skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri, bukan merupakan pengambilan tulisan atau pikiran orang lain yang saya akui sebagai hasil tulisan atau pikiran saya sendiri, kecuali dengan mencantumkan sumber cuplikan pada daftar pustaka.

Apabila dikemudian hari terbukti atau dapat dibuktikan skripsi ini hasil jiplakan, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Malang, 7 Juni 2018

Yang membuat pernyataan,



Diko Andri Vidian

NIM. 14650072

MOTTO

“Bekerja keras diam-diam dan biarkan kesuksesan yang membuat keramaian”

- *Diko Andri Vidian*



HALAMAN PERSEMBAHAN

Bismillahirrahmanirrahim

Segala puji bagi Allah atas ridho-Nya dan juga dukungan serta doa dari orang-orang tercinta akhirnya skripsi ini dapat terselesaikan dengan baik. Oleh karena itu kupersembahkan karya sederhana ini kepada:

Ayah dan ibu tercinta yang selalu memberikan motivasi, pengorbanan dan doa yang tiada henti untuk kesuksesan saya.

Kakakku, Saiful Arifin yang selalu mengingatkan dan membimbingku untuk kesuksesan dan kebaikan dimasa depan.

Bapak Fatchurrochman, M.kom dan Bapak Ajib Hanani, MT selaku dosen pembimbing yang selama ini tulus, sabar serta ikhlas meluangkan waktunya untuk memberikan bimbingan, selain itu juga untuk seluruh dosen Teknik Informatika yang selama ini telah mendidik dan menyalurkan segala ilmu informatika dan juga pengalaman yang sangat berarti untuk saya. Bapak dan Ibu dosenku jasmu akan selalu terpatri dihati.

Teman-teman Biner (TI 2014) terima kasih atas semangat, dukungan dan bantuan serta kekompakan selama ini, terima kasih juga atas canda tawa dan kenangan manis yang telah mengukir selama ini. Semoga suatu hari ada moment bertemu berkumpul bersama lagi.

Seluruh teman-temanku yang senantiasa selalu memotivasi dan menjadi tempat untuk saling belajar.

Serta seluruh pihak yang tidak dapat disebutkan satu persatu, terima kasih.

KATA PENGANTAR

Assalamualaikum Wr. Wb.

Segala puji bagi Allah SWT tuhan semesta alam, karena atas segala rahmat dan karunia-Nya sehingga penulis mampu menyelesaikan skripsi ini dengan baik dan lancar. Shalawat serta salam selalu tercurahkan kepada tauladan terbaik Nabi Muhammad SAW yang telah membimbing umatnya dari zaman kebodohan menuju Islam yang rahmatan lil alamin.

Selanjutnya penulis haturkan ucapan terima kasih karena dalam penyelesaian skripsi ini tidak lepas dari bantuan, bimbingan serta dukungan dari beberapa pihak. Ucapan terima kasih ini penulis sampaikan kepada:

1. Prof. DR. H. Abd. Haris, M.Ag, selaku rektor UIN Maulana Malik Ibrahim Malang beserta seluruh staf. Bakti Bapak dan Ibu sekalian terhadap UIN Maliki Malang yang menaungi segala kegiatan di kampus UIN Maliki Malang.
2. Dr. Sri Harini, M.Si selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang beserta seluruh staf. Bapak dan ibu sekalian sangat berjasa memupuk dan menumbuhkan semangat untuk maju kepada penulis.
3. Bapak Dr. Cahyo Crys dian, selaku Ketua Jurusan Teknik Informatika Universitas Islam Negeri Maulana Malik Ibrahim Malang, yang sudah memberi banyak menginspirasi dan memotivasi untuk terus berkembang.

4. Bapak Fatchurrochman, M.Kom selaku dosen pembimbing I yang telah bersedia meluangkan waktu untuk membimbing, memotivasi dan memberi arahan kepada penulis dalam pengerjaan skripsi ini hingga akhir.
5. Bapak Ajib Hanani, M.T selaku dosen pembimbing II yang juga senantiasa memberi masukan dan nasihat serta petunjuk dalam penyusunan skripsi ini.
6. Ibu Hani Nurhayati, M.T selaku dosen wali yang telah memberi arahan dan bimbingan selama ini.
7. Ayah, Ibu dan Kakak serta keluarga besar tercinta yang selalu memberi dorongan dan doa yang senantiasa mengiringi setiap langkah penulis.
8. Seluruh Dosen Teknik Informatika yang telah memberikan keilmuan serta pengalaman yang berarti kepada penulis selama ini.
9. Teman-teman yang telah memotivasi dan membantu banyak hal selama ini.
10. Seluruh teman-teman Teknik Informatika UIN Maulana Malik Ibrahim Malang yang telah banyak berbagi ilmu, pengalaman dan menjadi inspirasi untuk terus semangat belajar.
11. Teman-teman seperjuangan Teknik Informatika 2014 yang telah berjuang bersama dan saling mendukung selama ini.
12. Para peneliti yang telah melakukan penelitian tentang prediksi cacat perangkat lunak dan seleksi fitur yang menjadi acuan penulis dalam pembuatan skripsi ini. Serta semua pihak yang telah membantu yang tidak bisa disebutkan satu persatu. Terimakasih banyak.

Penulis menyadari bahwa skripsi ini masih jauh dari sempurna dikarenakan terbatasnya pengalaman dan pengetahuan yang dimiliki penulis. Oleh karena itu, penulis mengharapkan segala bentuk saran dan masukan serta kritik yang membangun dari berbagai pihak. Terlepas dari berbagai kekurangan tersebut, semoga skripsi ini dapat memberikan manfaat bagi pembaca dan mendorong peneliti selanjutnya dalam menyempurnakannya. Amin.

Wassalamualaikum Wr. Wb.

Malang, 7 Juni 2018

Penulis



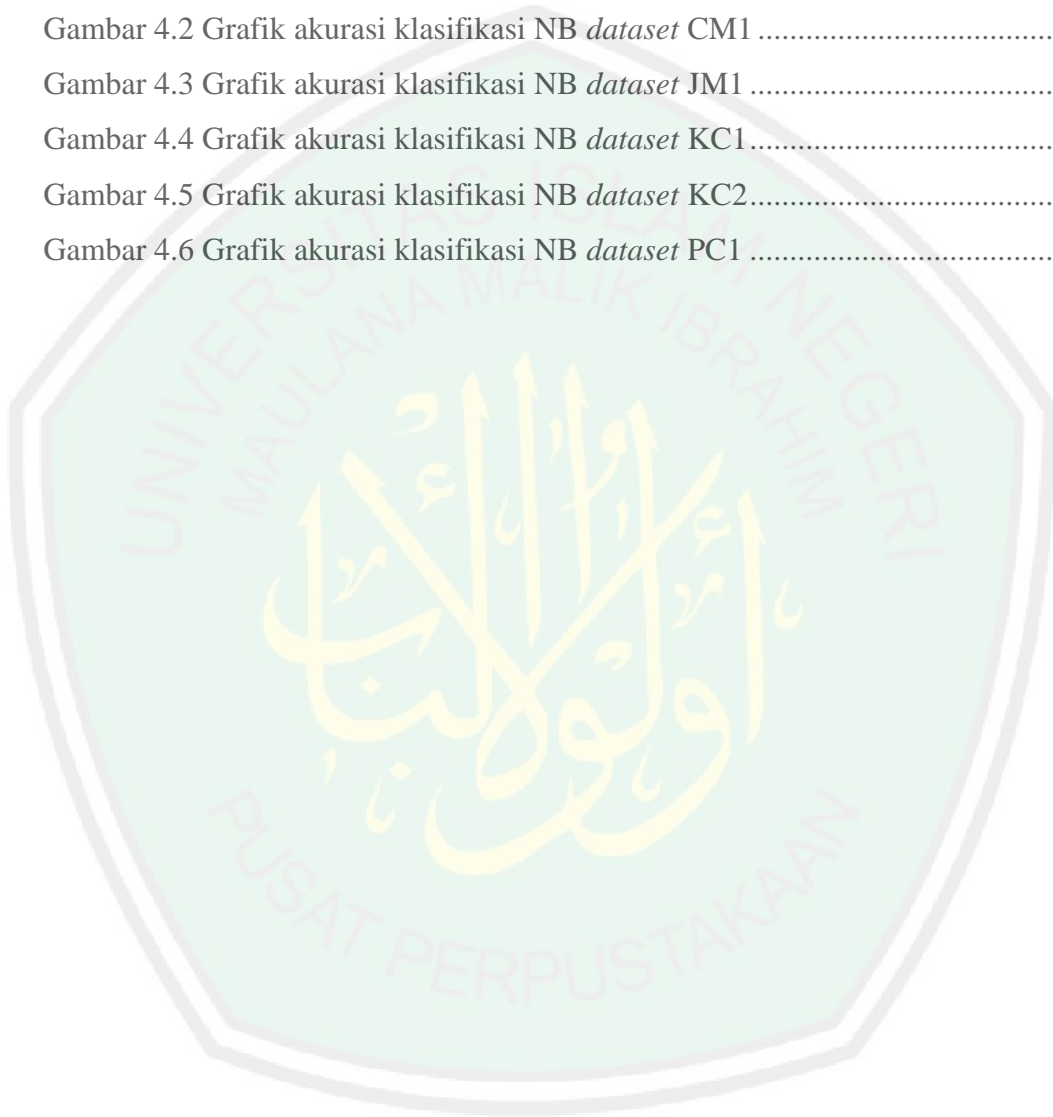
DAFTAR ISI

HALAMAN JUDUL.....	i
HALAMAN PERSETUJUAN.....	ii
HALAMAN PENGESAHAN.....	iii
HALAMAN PERNYATAAN KEASLIAN TULISAN.....	iv
MOTTO.....	v
HALAMAN PERSEMBAHAN.....	vi
KATA PENGANTAR.....	vii
DAFTAR ISI.....	x
DAFTAR GAMBAR.....	xii
DAFTAR TABEL.....	xiii
ABSTRAK.....	xiv
ABSTRACT.....	xv
ملخص.....	xvi
BAB I PENDAHULUAN.....	1
1.1. Latar Belakang.....	1
1.2. Identifikasi Masalah.....	5
1.3. Tujuan Penelitian.....	6
1.4. Batasan Masalah.....	6
1.5. Manfaat Penelitian.....	6
1.6. Sistematika Penulisan.....	7
BAB II STUDI PUSTAKA.....	8
2.1 Cacat Perangkat Lunak.....	8
2.2 Prediksi Cacat Perangkat Lunak.....	9
2.3 Diskritisasi Data.....	15
2.4 Diskritisasi Data dengan <i>Equal Width Binning</i>	16
2.5 Seleksi Fitur.....	16
2.6 Seleksi Fitur dengan <i>Gain Ratio</i>	17
2.7 <i>K-fold Cross Validation</i>	19
2.8 Klasifikasi Naïve Bayes.....	20
2.9 Evaluasi Klasifikasi.....	23
2.10 Penelitian Terkait.....	24

BAB III ANALISIS DAN PERANCANGAN	27
3.1 Analisis Sistem.....	27
3.2 Perancangan Sistem	29
3.2.1 Proses Cleaning.....	30
3.2.2 Proses Diskritisasi dengan <i>Equal Width Binning</i>	31
3.2.3 Proses seleksi fitur dengan <i>Gain Ratio</i>	34
3.2.4 Klasifikasi Naïve Bayes	40
3.2.5 Perhitungan Akurasi.....	41
BAB IV UJI COBA DAN PEMBAHASAN	43
4.1 Uji Coba.....	43
4.1.1 Lingkungan Uji Coba.....	43
4.1.2 Data Uji coba.....	44
4.1.3 Tampilan Sistem	47
4.1.4 Hasil Pengujian	48
4.2 Pembahasan.....	60
4.2.1 Pembahasan hasil pengujian <i>dataset</i> CM1.....	60
4.2.2 Pembahasan hasil pengujian <i>dataset</i> JM1.....	61
4.2.3 Pembahasan hasil pengujian <i>dataset</i> KC1	61
4.2.4 Pembahasan hasil pengujian <i>dataset</i> KC2	62
4.2.5 Pembahasan hasil pengujian <i>dataset</i> PC1	63
4.3 Integrasi dengan Islam	66
BAB V PENUTUP.....	69
5.1 Kesimpulan	69
5.2 Saran	70
DAFTAR PUSTAKA	71
LAMPIRAN-LAMPIRAN	

DAFTAR GAMBAR

Gambar 2.1 Teknik M x N <i>Cross-Validation</i>	20
Gambar 3.1 Desain Sistem yang diusulkan.....	30
Gambar 4.1 Tampilan Sistem.....	47
Gambar 4.2 Grafik akurasi klasifikasi NB <i>dataset</i> CM1	60
Gambar 4.3 Grafik akurasi klasifikasi NB <i>dataset</i> JM1	61
Gambar 4.4 Grafik akurasi klasifikasi NB <i>dataset</i> KC1.....	62
Gambar 4.5 Grafik akurasi klasifikasi NB <i>dataset</i> KC2.....	63
Gambar 4.6 Grafik akurasi klasifikasi NB <i>dataset</i> PC1	64



DAFTAR TABEL

Tabel 2.1 Informasi fitur prediksi cacat perangkat lunak	10
Tabel 2.2 <i>Confusion Matrix</i>	23
Tabel 3.1 Isi <i>Dataset</i> CM1	28
Tabel 3.2 <i>Dataset</i> CM1	33
Tabel 3.3 Hasil diskritisasi fitur ke-1	34
Tabel 3.4 Hasil diskritisasi semua fitur <i>dataset</i> CM1	34
Tabel 3.5 Hasil perankingan sesuai nilai <i>Gain Ratio</i> pada <i>dataset</i> CM1.....	39
Tabel 3.6 Hasil seleksi 5 fitur pada <i>dataset</i> CM1	40
Tabel 3.7 <i>Confusion Matrix</i> klasifikasi <i>dataset</i> CM1 dengan 5 fitur hasil seleksi fitur GR.....	42
Tabel 4.1 <i>Dataset</i> CM1	44
Tabel 4.2 <i>Dataset</i> JM1	44
Tabel 4.3 <i>Dataset</i> KC1.....	44
Tabel 4.4 <i>Dataset</i> KC2.....	45
Tabel 4.5 <i>Dataset</i> PC1	45
Tabel 4.6 <i>Dataset</i> Penelitian	45
Tabel 4.7 Fitur setiap <i>dataset</i>	46
Tabel 4.8 Perankingan Fitur <i>dataset</i> CM1 Berdasarkan nilai <i>Gain Ratio</i>	49
Tabel 4.9 Perankingan Fitur <i>dataset</i> JM1 Berdasarkan nilai <i>Gain Ratio</i>	50
Tabel 4.10 Perankingan Fitur <i>dataset</i> KC1 Berdasarkan nilai <i>Gain Ratio</i>	51
Tabel 4.11 Perankingan Fitur <i>dataset</i> KC2 Berdasarkan nilai <i>Gain Ratio</i>	52
Tabel 4.12 Perankingan Fitur <i>dataset</i> PC1 Berdasarkan nilai <i>Gain Ratio</i>	53
Tabel 4.13 Hasil Uji Coba Klasifikasi <i>Dataset</i> CM1.....	55
Tabel 4.14 Hasil Uji Coba Klasifikasi <i>Dataset</i> JM1.....	56
Tabel 4.15 Hasil Uji Coba Klasifikasi <i>Dataset</i> KC1	57
Tabel 4.16 Hasil Uji Coba Klasifikasi <i>Dataset</i> KC2	58
Tabel 4.17 Hasil Uji Coba Klasifikasi <i>Dataset</i> PC1	59
Tabel 4.18 Fitur yang relevan berdasarkan seleksi fitur <i>Gain Ratio</i>	65

ABSTRAK

Vidian, Diko Andri. 2018. **Penentuan Fitur yang Relevan Terhadap Prediksi Cacat Perangkat Lunak Berdasarkan Seleksi Fitur Gain Ratio**. Skripsi. Jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang. Pembimbing: (I) Fatchurrochman, M.Kom. (II) Ajib Hanani, M.T.

Kata kunci: prediksi cacat perangkat lunak, seleksi fitur *Gain Ratio*.

Software development life cycle (SDLC) merupakan rangkaian tahap untuk menghasilkan produk perangkat lunak. Tahap paling penting dalam SDLC ada pada tahap pengujian, tahap ini melakukan pengujian perangkat lunak dan pemeriksaan validasi, setelah produk dinyatakan tidak cacat maka persetujuan diberikan dan perangkat lunak bisa digunakan pada kebutuhan. Salah satu cara yang bisa dilakukan untuk pengujian adalah teknik prediksi cacat perangkat lunak, teknik tersebut melakukan prediksi menggunakan *dataset Metrics Data Program* (MDP). Perlu diketahui bahwa dalam *dataset* tidak semua fitur yang ada memiliki pengaruh besar terhadap prediksi cacat perangkat lunak karena *dataset* yang digunakan dibuat tidak khusus untuk prediksi cacat perangkat lunak. Oleh karena itu, pemilihan fitur diperlukan untuk mendapatkan fitur yang berpengaruh terhadap prediksi cacat perangkat lunak.

Penelitian ini melakukan pemilihan fitur menggunakan seleksi fitur *Gain Ratio* dengan jumlah pengambilan fitur yang berbeda-beda dengan tujuan mendapatkan fitur yang paling berpengaruh atau relevan. Hasil dari penelitian ini menyimpulkan bahwa fitur yang relevan terhadap prediksi cacat perangkat lunak berdasarkan seleksi fitur *Gain Ratio* pada *dataset* CM1 adalah fitur *Time to write program* (t). *Dataset* JM1 adalah fitur *Error estimate* (b), *Count of Statement Lines* (IOCode), *Line of code* (loc), *Unique operands* (uniq_Opnd), *Unique operator* (uniq_Op), *Count of Code and Comments Lines* (IOCodeAndComment), *Cyclomatic complexity* (v(g)), *Branch count* (branchCount), *Time to write program* (t) dan *Effort to write program* (e). *Dataset* KC1 adalah fitur *Count of lines of comments* (IOComment), *Error estimate* (b) dan *Count of blank lines* (IOBlank). *Dataset* KC2 adalah fitur *Count of Code and Comments Lines* (IOCodeAndComment), *Count of blank lines* (IOBlank), *Unique operands* (uniq_Opnd) dan *Time to write program* (t). *Dataset* PC1 adalah fitur *Unique operands* (uniq_Opnd). Fitur-fitur tersebut dikatakan sebagai fitur yang relevan terhadap prediksi cacat perangkat lunak karena berdasarkan uji coba klasifikasi yang menghasilkan akurasi terbaik dari uji coba yang telah dilakukan.

ABSTRACT

Vidian, Diko Andri. 2018. **Determination of Relevant Features Against Software Defect Prediction Based on Gain Ratio Feature Selection.** Undergraduate Thesis. Department of Informatics Engineering Faculty of Science and Technology Islamic State University of Maulana Malik Ibrahim of Malang. Supervisor: (I) Fatchurrochman, M.Kom. (II) Ajib Hanani, M.T.

Keywords: software defect prediction, selection feature Gain Ratio.

Software development life cycle (SDLC) is a series of stages to produce software products. The most important stage in the SDLC is in the testing phase, this stage performs software testing and validation checks, after the product is declared not defective then approval is given and the software can be used on demand. One of the way that testing can be done is a software defect prediction technique, the technique predicts using the Metrics Data Program (MDP) dataset. Note that in the dataset not all features exist have a major effect on the prediction of software defects because the dataset used is not made specifically for software defect deficits. Therefore, feature selection is required to get features that affect the software defect prediction.

This study selects features using Gain Ratio feature selection with varying number of feature captures with the goal of getting the most influential or relevant features. The results of this study conclude that the relevant feature of software defect prediction based on Gain Ratio feature selection on CM1 dataset is Time to write program (t) feature. The JM1 dataset is a feature of Error Estimate (b), Count of Statement Lines (IOCode), Line of code (loc), Unique operands (uniq_Opnd), Unique operators (uniq_Op), Count of Code and Comments Lines (IOCodeAndComment), Cyclomatic complexity (v(g)), Branch count (branchCount), Time to write program (t) and Effort to write program (e). The KC1 dataset is a feature of Count of lines of comments (IOComment), Error estimate (b) and Count of blank lines (IOBlank). The KC2 dataset is a feature of Count of Code and Comments Lines (IOCodeAndComment), Count of blank lines (IOBlank), Unique operands (uniq_Opnd) and Time to write program (t). The PC1 dataset is a feature of Unique operands (uniq_Opnd). These features are said to be features relevant to the prediction of software defects because they are based on a classification test that produces the best accuracy of the experimental tests.

ملخص

فيديان، ديقو أندري. 2018. **تعيين الميزات ذات الصلة بتنبؤ علة البرنامج وفقا لاختيار الميزة Gain Ratio**. البحث العلمي. قسم المعلوماتية كلية العلوم والتكنولوجيا، جامعة مولانا مالك إبراهيم الإسلامية الحكومية مالانج. المشرف: (1) فتح الرحمن الماجستير. (2) عجيب حناني الماجستير.

الكلمات الرئيسية: تنبؤ علة البرنامج، اختيار الميزة Gain Ratio.

يكون *Software development life cycle (SDLC)* سلسلة المراحل لإنتاج منتجات البرنامج. وأهم المراحل في SDLC هي مرحلة الاختبار، تقوم هذه المرحلة باختبار البرنامج وتحقق صحته، بعد أن كان البرنامج خاليا عن العيوب فتمنح الإتفاقية ويمكن استخدام البرنامج في القضاء بالحاجة. من إحدى الطرق التي يمكن استخدامه للاختبار هي طريقة تنبؤ علة البرنامج، وهي أن يدور في عملية التنبؤ باستخدام *dataset Metrics Data Program (MDP)*. ويعد مهما أن يلاحظ أنه ليس لكل ملاحظ في مجموعة البيانات أثرا كبيرا تنبؤ علة البرنامج إذ هي لا تبنى للتخصص تنبؤ علة البرنامج. لذا، احتيج إلى اختيار الميزة للحصول على الميزة المؤثرة على تنبؤ علة البرنامج.

يقوم هذا البحث باختيار الميزة باستخدام اختيار الميزة *Gain Ratio* بأخذ عدد الميزات المختلفة من أجل الحصول على الميزات الأكثر تأثيرا أو ذات الصلة. واستنتجت نتائج هذا البحث أن الميزات ذات الصلة بتنبؤ علة البرنامج وفقا لاختيار الميزة *Gain Ratio* في مجموعة البيانات CM1 هي الميزة *Time to write program (t)*. ومجموعة البيانات JM1 هي الميزة *Error estimate (b)*، و *Count of Statement Lines (IOCode)*، و *Line of code (loc)*، و *Unique operands (uniq_Opnd)*، و *Count of Code and Comments Lines (IOCodeAndComment)*، و *Cyclomatic complexity ((g)v)*، و *Branch count (branchCount)*، و *Time to write program (t)*، و *Effort to write program (e)*. ومجموعة البيانات KC1 هي الميزة *Count of blank lines (IOBlank)*، و *lines of comments (IOComment)*، و *Error estimate (b)*، و *Count of Code and Comments Lines (IOBlank)*، و *Count of blank lines (IOCodeAndComment)*، و *Unique operands (uniq_Opnd)*، و *Time to write program (t)*. ومجموعة البيانات PC1 هي الميزة *Unique operands (uniq_Opnd)*. ويقال إن هذه الميزات ذات الصلة بتنبؤ علة البرنامج على أساس اختبار التصنيف التي تحصل على أفضل دقة من التجارب الذي تم القيام به.

BAB I

PENDAHULUAN

Pada bab pendahuluan ini akan dijelaskan tentang latar belakang penelitian, identifikasi masalah penelitian, tujuan penelitian, batasan masalah, manfaat penelitian dan sistematika penulisan.

1.1. Latar Belakang

Software atau biasa disebut dengan perangkat lunak adalah istilah khusus untuk data yang diformat dan disimpan secara digital yang didalamnya termasuk program komputer, dokumentasinya dan berbagai informasi yang dapat dibaca dan ditulis oleh komputer. Perangkat lunak juga dapat dikatakan sebagai bagian sistem komputer yang tidak berwujud, istilah ini digunakan untuk menunjukkan perbedaannya dengan *hardware* atau perangkat keras komputer.

Software development life cycle (SDLC) merupakan rangkaian tahap untuk menghasilkan produk perangkat lunak. Tahap paling penting dalam SDLC ada pada tahap pengujian, tahap ini melakukan pengujian perangkat lunak dan pemeriksaan validasi, setelah produk dinyatakan tidak cacat maka persetujuan diberikan dan perangkat lunak bisa digunakan pada kebutuhan. Biaya untuk memperbaiki cacat yang terdeteksi ketika masih pada tahap pengembangan jauh lebih sedikit dibandingkan ketika perangkat lunak sudah digunakan pada kebutuhan (Pelayo dan Dick, 2007). Salah satu cara yang bisa dilakukan untuk pengujian adalah teknik prediksi cacat perangkat lunak, teknik tersebut dapat mendeteksi hingga modul terkecil perangkat lunak yang memiliki kecenderungan cacat.

Beberapa penelitian telah *dilakukan* terkait topik prediksi cacat perangkat lunak dengan menerapkan algoritma *machine learning* yang berbeda-beda. Proses prediksi cacat perangkat lunak dengan *machine learning* tersebut dilakukan dengan menggunakan sebuah *dataset* dari NASA MDP (*metrics data program*) yang berisi sekumpulan fitur, namun perlu diketahui bahwa dari keseluruhan fitur yang ada tidak semua fitur tersebut memiliki pengaruh besar terhadap proses prediksi cacat perangkat lunak karena *dataset* yang digunakan dibuat tidak khusus untuk prediksi cacat perangkat lunak (Arar & Ayan, 2017). Oleh karena itu pemilihan fitur perlu dilakukan untuk mendapatkan fitur yang memiliki pengaruh besar atau relevan terhadap proses prediksi cacat perangkat lunak dan mengabaikan atau membuang fitur yang tidak memiliki pengaruh besar terhadap proses prediksi cacat perangkat lunak.

Beberapa penelitian terkait yang menerapkan seleksi fitur, Wahono (2014) dalam penelitiannya yang berjudul "*Genetic Feature Selection for Software Defect Prediction*" menerapkan seleksi fitur *Genetic Algorithm*, hasil dari penelitian tersebut menunjukkan performa klasifikasi yang meningkat dibandingkan dengan tanpa seleksi fitur. Penelitian lain oleh Wang *et.al* (2012) dalam penelitiannya yang berjudul "*Software measurement data reduction using ensemble techniques*", pada penelitiannya Wang menggunakan 17 fitur seleksi berbasis teknik *ensemble* yang diterapkan pada 16 *dataset* yang berbeda dan setelah fitur yang memiliki pengaruh besar berhasil didapatkan langkah selanjutnya melakukan pengujian dari fitur yang didapat memang benar memiliki pengaruh besar terhadap prediksi cacat perangkat lunak dengan cara melakukan klasifikasi terhadap beberapa algoritma klasifikasi dan membandingkan performa

antara tanpa seleksi fitur dan dengan seleksi fitur. Penelitian lain dilakukan oleh Karabulut *et.al* (2012) yang berjudul “*A comparative study on the effect of feature selection on classification accuracy*”, pada penelitiannya Karabulut *et.al* melakukan perbandingan beberapa seleksi fitur yang diterapkan pada beberapa algoritma klasifikasi dan hasil akhir pada penelitiannya menyimpulkan bahwa dari 6 seleksi fitur yang diteliti berikut adalah seleksi fitur yang menghasilkan performa klasifikasi paling baik; klasifikasi Naïve Bayes dengan seleksi fitur *Gain Ratio*, klasifikasi *Multilayer Perceptron* dengan seleksi fitur *Chi-square* dan klasifikasi J48 dengan seleksi fitur *Information Gain*.

Fokus penelitian ini adalah untuk membantu menentukan fitur yang memiliki pengaruh besar terhadap proses prediksi cacat perangkat lunak, fitur yang memiliki pengaruh besar memungkinkan proses prediksi cacat perangkat lunak akan menjadi lebih akurat. Saling membantu atau tolong-menolong adalah suatu sikap yang diperintahkan dalam islam, penejelasan tersebut ada pada potongan Ayat Alqur'an surat al-ma'idah ayat 2 berikut (Al-Sheikh, 2011:9);

يَا أَيُّهَا الَّذِينَ آمَنُوا لَا تُلْجُوا شَعَائِرَ اللَّهِ وَلَا الشُّهُرَ الْحَرَامَ وَلَا الْهَدْيَ وَلَا الْقَلَائِدَ وَلَا آمِينَ الْبَيْتِ الْحَرَامِ
يَبْتَغُونَ فَضْلًا مِنْ رَبِّهِمْ وَرِضْوَانًا وَإِذَا حَلَلْتُمْ فَاصْطَادُوا وَلَا يَجْرِمَنَّكُمْ شَنَاٰنُ قَوْمٍ أَنْ صَدُّوكُمْ عَنِ الْمَسْجِدِ
الْحَرَامِ أَنْ تَعْتَدُوا وَتَعَاوَنُوا عَلَى الْبِرِّ وَالتَّقْوَىٰ وَلَا تَعَاوَنُوا عَلَى الْإِثْمِ وَالْعُدْوَانِ وَاتَّقُوا اللَّهَ إِنَّ اللَّهَ شَدِيدُ
الْعِقَابِ

Artinya : “*Hai orang-orang yang beriman, janganlah kamu melanggar syi'ar-syi'ar Allah, dan jangan melanggar kehormatan bulan-bulan haram, jangan (mengganggu) binatang-binatang had-ya, dan binatang-binatang qalaa-id, dan jangan (pula) mengganggu orang-orang yang mengunjungi Baitullah sedang mereka mencari kurnia dan keridhaan dari Tuhannya dan apabila kamu telah menyelesaikan ibadah haji, maka bolehlah berburu. Dan janganlah sekali-kali*

kebencian(mu) kepada sesuatu kaum karena mereka menghalang-halangi kamu dari Masjidilharam, mendorongmu berbuat aniaya (kepada mereka). Dan tolong-menolonglah kamu dalam (mengerjakan) kebajikan dan takwa, dan jangan tolong-menolong dalam berbuat dosa dan pelanggaran. Dan bertakwalah kamu kepada Allah, sesungguhnya Allah amat berat siksa-Nya.” (al-ma’idah:2).

Pada potongan ayat tersebut Allah SWT memerintahkan untuk saling tolong-menolong dalam aktivitas kebaikan dan dilarang untuk tolong-menolong dalam perbuatan yang salah dan mengakibatkan dosa, potongan ayat tersebut dalam Tafsir Ibnu Katsir dijelaskan Allah SWT memerintahkan kepada hamba-hamba-Nya yang beriman untuk saling menolong dalam berbuat kebaikan yaitu kebajikan dan meninggalkan hal-hal yang mungkar; hal ini dinamakan ketakwaan. Allah SWT melarang mereka bantu- membantu dalam kebatilan serta tolong-menolong dalam perbuatan dosa dan hal-hal yang diharamkan. Ibnu Jarir mengatakan bahwa dosa itu ialah meninggalkan apa yang diperintahkan oleh Allah untuk dikerjakan. Pelanggaran itu artinya melampaui apa yang digariskan oleh Allah dalam agama kalian, serta melupakan apa yang difardukan oleh Allah atas diri kalian dan atas diri orang lain. Penjelasan tersebut kemudian diperkuat dengan hadits yang diriwayatkan oleh Imam Bukhari secara munfarid melalui hadis Hasyim dengan sanad yang sama dan lafaz yang semisal. Keduanya mengetengahkan hadis ini melalui jalur Sabit, dari Anas yang menceritakan bahwa Rasulullah Saw. telah bersabda:

أَنْصُرْ أَخَاكَ ظَالِمًا أَوْ مَظْلُومًا". قِيلَ: يَا رَسُولَ اللَّهِ، هَذَا نَصْرُهُ مَظْلُومًا، فَكَيْفَ أَنْصُرُهُ ظَالِمًا؟ قَالَ: "تَمَنَعُهُ
مِنَ الظُّلْمِ، فَذَاكَ نَصْرُكَ إِيَّاهُ

Artinya : *"Tolonglah saudaramu, baik dia berbuat aniaya ataupun dianiaya." Ditanyakan, "Wahai Rasulullah, orang ini dapat aku tolong bila dalam*

keadaan teraniaya, tetapi bagaimana menolongnya jika dia berbuat aniaya?" Rasulullah Saw. menjawab, "Kamu cegah dia dari perbuatan aniaya, itulah cara kamu menolongnya."

Berdasarkan penjelasan masalah yang ada dan juga ayat Alqur'an serta hadits, penulis melakukan penelitian ini untuk membantu menentukan fitur-fitur yang memiliki pengaruh besar atau relevan terhadap prediksi cacat perangkat lunak, proses pemilihan fitur yang relevan dilakukan dengan teknik seleksi fitur. Pada penelitian ini penulis mengusulkan teknik seleksi fitur menggunakan metode seleksi fitur *Gain ratio* karena berdasarkan penelitian-penelitian terdahulu yang menunjukkan bahwa seleksi fitur *Gain Ratio* dapat meningkatkan akurasi prediksi/klasifikasi, hal ini menunjukkan bahwa fitur yang dihasilkan dari seleksi fitur *Gain Ratio* memiliki pengaruh besar terhadap proses prediksi/klasifikasi. Oleh karena itu, pada penelitian ini penulis ingin mengangkat judul "Penentuan Fitur yang Relevan Terhadap Prediksi Cacat Perangkat Lunak Berdasarkan Seleksi Fitur *Gain Ratio*".

1.2. Identifikasi Masalah

Berdasarkan uraian pada latar belakang masalah, maka dapat diidentifikasi bahwa masalahnya adalah tidak semua fitur yang ada dalam *dataset* NASA MDP memiliki pengaruh besar atau relevan terhadap prediksi cacat perangkat lunak sehingga perlu dilakukan seleksi fitur untuk mendapatkan fitur yang relevan terhadap prediksi cacat perangkat lunak, penentuan fitur yang relevan terhadap prediksi cacat perangkat lunak pada penelitian ini menggunakan seleksi fitur *Gain Ratio*.

Masalah yang ada dapat dirumuskan menjadi pertanyaan penelitian yaitu fitur apa saja yang relevan terhadap prediksi cacat perangkat lunak berdasarkan seleksi fitur *Gain Ratio*?

1.3. Tujuan Penelitian

Berdasarkan identifikasi masalah, maka tujuan dari penelitian ini untuk menentukan fitur yang relevan terhadap prediksi cacat perangkat lunak berdasarkan seleksi fitur *Gain Ratio*.

1.4. Batasan Masalah

Untuk menjaga fokus dari penelitian ini, maka ada beberapa batasan masalah sebagai berikut:

- a. Data yang digunakan berupa *dataset metrics data program* (MDP) yang diunduh dari PROMISE (*Predictor Models in Software Engineering repository*) melalui <http://promise.site.uottawa.ca/SERepository> pada hari kamis, 8 Maret 2018 yang terdiri dari 5 *dataset* yaitu *dataset* CM1, JM1, KC1, KC2 dan PC1.
- b. Proses *cleaning* dan klasifikasi Naïve Bayes menggunakan fungsi dari *library Weka* versi 3.6.7.

1.5. Manfaat Penelitian

Adapun manfaat yang diharapkan dari penelitian ini adalah dapat meningkatkan kualitas perangkat lunak melalui pengujian perangkat lunak dengan cara prediksi cacat perangkat lunak yang akurat berdasarkan fitur yang relevan terhadap prediksi cacat perangkat lunak.

1.6. Sistematika Penulisan

Sistematika penulisan ini memberikan gambaran dan kerangka yang jelas mengenai pokok bahasan di setiap bab dalam penelitian ini. Berikut penjelasan sistematika pembahasan pada masing-masing bab:

BAB I : PENDAHULUAN

Pada bab pendahuluan berisi latar belakang masalah, identifikasi masalah, tujuan penelitian, manfaat penelitian, batasan masalah dan sistematika penulisan.

BAB II: STUDI PUSTAKA

Bab dua yaitu studi pustaka menjelaskan tentang penelitian terkait dan teori yang berhubungan dengan permasalahan penelitian.

BAB III: ANALISIS DAN PERANCANGAN

Pada bab ini akan dibahas tentang analisis dan perancangan sistem atau program terhadap metode yang digunakan.

BAB IV: UJI COBA DAN PEMBAHASAN

Pada bab ini menjelaskan tentang proses uji coba yang meliputi lingkungan uji coba, data uji coba, tampilan sistem dan hasil uji coba serta pembahasan terhadap hasil uji coba.

BAB V: PENUTUP

Berisi kesimpulan dan saran berdasarkan hasil yang telah dicapai sehingga dapat digunakan sebagai bahan pertimbangan bagi pihak-pihak yang berkepentingan serta kemungkinan pengembangannya.

BAB II

STUDI PUSTAKA

Pada bab studi pustaka ini menjelaskan teori-teori yang digunakan dalam penelitian ini yang meliputi cacat perangkat lunak, prediksi cacat perangkat lunak, diskritisasi data, diskritisasi data dengan *Equal Width Binning*, seleksi fitur, seleksi fitur dengan *Gain Ratio*, *K-fold Cross validation*, klasifikasi Naïve Bayes, Evaluasi klasifikasi dan terakhir tentang penelitian terkait yang pernah dilakukan untuk mendapatkan gambaran secara lengkap terkait penelitian prediksi cacat perangkat lunak.

2.1 Cacat Perangkat Lunak

Cacat perangkat lunak atau biasa dikenal dengan *Software Defect* didefinisikan sebagai cacat pada perangkat lunak seperti cacat pada dokumentasi, pada kode program, pada desain dan hal-hal lain yang menyebabkan kegagalan perangkat lunak (Akbar, 2017). Penjelasan lain dari Runeson *et.al* (2006) yang menjelaskan bahwa cacat perangkat lunak terdiri dari defect kebutuhan (*requirement defect*), defect desain (*design defect*), dan kode cacat (*code defect*).

Defect atau cacat perangkat lunak merupakan faktor penting yang mempengaruhi kualitas perangkat lunak. Kualitas perangkat lunak dapat ditingkatkan dengan meminimalkan adanya cacat pada perangkat lunak melalui perbaikan pada bagian yang mungkin menghasilkan cacat perangkat lunak ketika proses pengembangan perangkat lunak.

Sebagai upaya untuk menghasilkan perangkat lunak yang kualitas maka perlu adanya pengujian yang baik seperti melakukan deteksi kecacatan perangkat

lunak. Tahap pengujian merupakan tahap yang paling penting dalam *Software Development Life Cycle (SDLC)*, tahap pengujian melakukan pengujian perangkat lunak dan pemeriksaan validasi, setelah produk dinyatakan tidak cacat maka persetujuan diberikan dan perangkat lunak bisa digunakan pada kebutuhan. Biaya untuk memperbaiki cacat yang terdeteksi ketika masih pada tahap pengembangan jauh lebih sedikit dibandingkan ketika perangkat lunak sudah digunakan pada kebutuhan (Pelayo & Dick, 2007). Salah satu cara yang bisa dilakukan untuk pengujian adalah teknik prediksi cacat perangkat lunak, teknik tersebut dapat mendeteksi hingga modul terkecil perangkat lunak yang memiliki kecenderungan cacat.

2.2 Prediksi Cacat Perangkat Lunak

Prediksi cacat perangkat lunak atau *Software Defect Prediction (SDP)* merupakan salah satu cara yang dapat membantu pada tahap pengujian perangkat lunak, sebagaimana telah dijelaskan bahwa dengan prediksi cacat perangkat lunak dapat mengetahui hingga modul terkecil perangkat lunak yang memiliki kecenderungan cacat.

Proses prediksi cacat perangkat lunak yang dilakukan oleh NASA MDP berdasarkan pada modul atau kode program dengan menggunakan fitur-fitur sebagai berikut;

Tabel 2.1 Informasi fitur prediksi cacat perangkat lunak

Simbol	Keterangan
loc	McCabe's "line count of code"
v(g)	McCabe "Cyclomatic complexity"
ev(g)	McCabe "Essential complexity"
iv(g)	McCabe "Design complexity"
uniq_Op	unique operators
uniq_Opnd	unique operands
total_Op	total operator
total_Opnd	total operand
n	Halstead "Total operator + operand"
v	Halstead "Volume"
l	Halstead "Program length"
d	Halstead "Difficulty"
i	Halstead "Intelligence"
e	Halstead "Effort to write program "
b	Halstead "Error estimate"
t	Halstead "Time to write program"
IOCode	Count of Statement Lines
IOComment	Count of lines of comments
IOBlank	Count of blank lines
IOCodeAndComment	Count of Code and Comments Lines
branchCount	Branch count
defect/problem	Hasil prediksi {false,true/yes,no}

Berdasarkan tabel 2.1 terlihat bahwa fitur atau metrik yang digunakan berjumlah 21 fitur, berikut adalah penjelasan dari setiap fitur yang digunakan (Menzies, 2005; Menzies, 2007; Saifudin, 2014; Akbar, 2017);

a. *Line of code (loc)*

Metrik loc ini merupakan metrik yang umum digunakan untuk mengukur kompleksitas sebuah kode program dengan cara menghitung jumlah baris kode program suatu perangkat lunak.

b. *Cyclomatic complexity (v(g))*

Metrik $v(g)$ adalah nilai metrik kompleksitas yang digunakan untuk menilai struktur logika keputusan sebuah kode program. Logika keputusan dapat berupa statemen *if* atau *looping* yang diilustrasikan sebagai sebuah *graph* yang terdiri dari *arc(e)* dan *node(n)*. Rumus perhitungannya adalah $v(g) = e - n + 2$.

c. *Essential complexity (ev(g))*

Metrik $ev(g)$ adalah nilai metrik kompleksitas yang dihitung dengan mengurangi nilai *cyclomatic complexity* $v(g)$ dengan jumlah *subflowgraph* yang terdiri dari *single entry* dan *single exit*. Rumus perhitungan *Essential complexity* adalah $ev(g) = v(g) - m$, m adalah jumlah *subflowgraph* yang merupakan *single entry* dan *single exit*.

d. *Design complexity (iv(g))*

Design complexity adalah *cyclomatic complexity* dari suatu modul yang mengurangi *flowgraph*. *Flowgraph* "g" dari sebuah modul dikurangi untuk menghilangkan kompleksitas yang tidak mempengaruhi keterkaitan antara modul desain. *Design complexity* dihitung dengan mengurangi modul pada *flowgraph* dengan cara menghilangkan *node decision* yang tidak memiliki dampak pada kontrol sebuah program.

e. *Unique operators (uniq_Op)*

Metrik uniq_Op merupakan nilai metrik yang didapat dengan cara menghitung jumlah operator yang berbeda dalam sebuah kode program.

Sebagai contoh perhitungan perhatikan kode program berikut;

```
main()
{
    int a, b, c, avg;
    scanf("%d %d %d", &a, &b, &c);
    avg = (a + b + c) / 3;
    printf("avg = %d", avg);
}
```

uniq_Op pada program tersebut adalah main, (), {}, int, scanf, &, =, +, /, printf,',', ; sehingga nilai uniq_Op = 12.

f. *Unique operands (uniq_Opnd)*

Metrik uniq_Opnd merupakan nilai metrik yang didapat dengan cara menghitung jumlah operand yang berbeda dalam sebuah kode program.

Pada kode program contoh sebelumnya maka uniq_Opnd-nya adalah a, b, c, avg, "%d %d %d", 3, "avg = %d" sehingga nilai uniq_Opnd = 7.

g. *Total operator (total_Op)*

Metrik total_Op merupakan nilai metrik yang didapat dengan cara menghitung jumlah operator dalam sebuah kode program. Berdasarkan contoh kode program sebelumnya maka nilai total_Op-nya adalah 27.

h. *Total operand (total_Opnd)*

Metrik total_Opnd merupakan nilai metrik yang didapat dengan cara menghitung jumlah operand dalam sebuah kode program. Berdasarkan contoh kode program sebelumnya maka nilai total_Opnd-nya adalah 15.

i. Total operator + operand (n)

Metrik “n” merupakan nilai metrik yang didapat dengan cara menghitung jumlah total_Op dengan total_Opnd dalam sebuah kode program. Rumus untuk menghitung “n” adalah $n = \text{total_Op} + \text{total_Opnd}$. Berdasarkan contoh sebelumnya maka nilai $n = 27 + 15 = 42$.

j. Volume (v)

Metrik “v” merupakan nilai metrik yang didapat dengan cara menghitung sesuai rumus $v = n * \log_2(\text{uniq_Op} + \text{uniq_Opnd})$. Berdasarkan contoh sebelumnya maka nilai $v = 42 * \log_2(12 + 7) = 178.4$.

k. Program length (l)

Metrik “l” merupakan nilai metrik yang didapat dengan cara menghitung sesuai rumus $l = V^* / n$. V^* adalah volume pada implementasi minimal yang didapat dengan perhitungan $V^* = (\text{uniq_Op}' + \text{uniq_Opnd}') * \log_2(\text{uniq_Op}' + \text{uniq_Opnd}')$, $\text{uniq_Op}'$ adalah jumlah operator potensial (hanya nama fungsi dan operator "return") sehingga nilai $\text{uniq_Op}' = 2$ dan $\text{uniq_Opnd}'$ adalah jumlah operan potensial (jumlah argumen pada modul). Berdasarkan penjelasan tersebut sehingga rumus v^* menjadi $V^* = (2 + \text{uniq_Opnd}') * \log_2(2 + \text{uniq_Opnd}')$.

l. Difficulty (d)

Metrik “d” merupakan nilai matrik hasil perhitungan dari $d = (\text{uniq_Op} / 2) * (\text{total_Opnd} / \text{uniq_Opnd})$. Berdasarkan contoh sebelumnya maka nilai $d = (12/2) * (15/7) = 12.85$.

m. Intelligence (i)

Metrik “i” digunakan untuk mengukur kompleksitas algoritma yang diterapkan pada kode program dan tidak bergantung pada bahasa pemrograman yang dipakai. Rumus dari metrik $i = v / d$. Berdasarkan contoh sebelumnya maka nilai $i = 178.4 / 12.85 = 13.8833$.

n. Effort to write program (e)

Metrik e adalah metrik yang digunakan untuk mengukur *effort* atau usaha untuk menulis sebuah kode program. Rumus untuk menghitung nilai metrik ini dengan persamaan $e = d * v$. Berdasarkan contoh sebelumnya maka nilai $e = 12.85 * 178.4 = 2292.44$.

o. Error estimate (b)

Metrik b merupakan metrik yang menunjukkan jumlah bug validasi, perhitungan b dilakukan dengan rumus $b = v / 3000$. Berdasarkan contoh sebelumnya maka nilai $b = 178.4 / 3000 = 0.0595$.

p. Time to write program (t)

Metrik “t” merupakan metrik yang digunakan untuk mengukur perkiraan waktu yang proportional dari setiap effort. Rumus untuk menghitung nilai t dengan $t = e / 18(\text{seconds})$. Berdasarkan contoh sebelumnya maka nilai $t = 2292.44 / 18 = 127.357$.

q. Count of Statement Lines (IOCode)

Metrik IOCode merupakan metrik yang menghitung baris statement (if, while, for) dalam baris kode program.

r. *Count of lines of comments (IOComment)*

Metrik IOComment merupakan metrik yang menghitung jumlah baris komen dalam kode program.

s. *Count of blank lines (IOBlank)*

Metrik IOBlank merupakan metrik yang menghitung jumlah baris yang kosong dalam kode program.

t. *Count of Code and Comments Lines (IOCodeAndComment)*

Metrik IOCodeAndComment merupakan metrik yang menghitung jumlah baris kode yang terdapat komen.

u. *Branch count (branchCount)*

Metrik branchCount adalah metrik yang menghitung jumlah percabangan dalam kode program. Jumlah percabangan oleh statement if dan case dalam kode program.

2.3 Diskritisasi Data

Diskritisasi adalah prosedur pengolahan data yang mentransformasikan data kuantitatif menjadi data kualitatif (Yang, *et.al*, 2010). Diskritisasi adalah salah satu tahap pada praproses data yang paling penting. Sebagian besar algoritma *machine learning* yang ada mampu mengekstrak pengetahuan dari database yang menyimpan fitur diskrit. Jika fitur kontinyu, algoritma dapat diintegrasikan dengan algoritma diskritisasi yang mengubahnya menjadi fitur diskrit. Metode diskritisasi digunakan untuk mengurangi jumlah nilai fitur kontinyu yang diberikan dengan membagi rentang fitur menjadi interval. Label interval kemudian dapat digunakan untuk menggantikan nilai data aktual. Diskritisasi data

dalam proses prediksi cacat perangkat lunak dapat meningkatkan akurasi, metode *machine learning* yang digunakan Naïve Bayes dan j48 (Singh & Verma, 2009).

2.4 Diskritisasi Data dengan *Equal Width Binning*

Equal Width Binning adalah salah satu metode diskritisasi *unsupervised*, metode ini banyak digunakan karena *simple* sehingga mudah diterapkan. Yang, *et.al* (2010) menjelaskan alur proses diskritisasi dengan *Equal Width Binning* sebagai berikut;

- a. Mendefinisikan nilai K, K adalah jumlah interval. K akan membagi garis bilangan antara Vmin dan Vmax menjadi interval K dengan lebar yang sama. Vmin adalah nilai terkecil dari fitur dan Vmax adalah nilai terbesar dari fitur. Daugherty, *et.al.* (1995) mendefinisikan nilai $K = \max\{1, 2 \log L\}$, nilai L adalah jumlah nilai pengamatan yang berbeda untuk setiap fitur. Pemilihan tersebut berdasarkan penelitian oleh Spector pada tahun 1994, heuristik dipilih berdasarkan algoritma sistogram histogram S-plus.
- b. Menghitung lebar setiap interval dengan rumus $W = \frac{V_{max} - V_{min}}{K}$
- c. Menentukan titik potong yang didapat dengan menghitung, $V_{min} + W, V_{min} + 2W, \dots, V_{min} + (K-1)W$.

2.5 Seleksi Fitur

Esra *et.al* (2012) menjelaskan bahwa seleksi fitur merupakan salah satu teknik dari reduksi data, konsep dasar dari seleksi fitur adalah menghilangkan data yang tidak begitu berpengaruh dalam suatu *dataset*. Seleksi Fitur merupakan

proses menghapus data yang redundan atau data yang kiranya tidak begitu berpengaruh dari suatu *dataset*, dari proses seleksi fitur dapat meningkatkan akurasi dari klasifikasi karena fitur-fitur yang sebelumnya tidak begitu berpengaruh atau redundan akan terseleksi terlebih dahulu termasuk data-data yang mengandung *noisy*.

Manfaat dari adanya seleksi fitur adalah dapat meningkatkan akurasi klasifikasi dan juga dapat meningkatkan proses *training* (Bratu *et.al*, 2008). Pada penelitian ini seleksi fitur menggunakan metode *Gain Ratio* hal ini berdasarkan penelitian yang pernah dilakukan dan seleksi fitur dengan *Gain Ratio* menunjukkan peningkatan akurasi prediksi.

2.6 Seleksi Fitur dengan *Gain Ratio*

Tahap awal dari prosedur ini adalah merubah fitur numerik menjadi fitur kategoris karena pendekatan ini bekerja dengan data kategoris (Trabelsi, *et.al*. 2017). Proses penghitungan *Gain Ratio* (Suyanto, 2017:134-138), sebagai berikut;

- a. Penghitungan nilai *Entropy*

$$Entropy(S) = \sum_i^c -p_i \log_2 p_i \quad (2,1)$$

C = jumlah nilai yang ada pada fitur target (jumlah kelas)

Pi = rasio antara jumlah sampel pada kelas i terhadap semua sampel pada himpunan data

Dari persamaan (2,1) dapat dicermati bahwa apabila hanya terdapat 2 kelas dan dari kedua kelas tersebut memiliki komposisi jumlah sampel yang sama, maka entropynya = 0.

b. Hitung nilai *Information Gain*

Setelah mendapatkan nilai entropy, maka langkah selanjutnya adalah melakukan perhitungan nilai *Information Gain*. *Information Gain* didefinisikan sebagai ukuran efektivitas suatu fitur dalam mengklasifikasikan data. Perhitungan nilai *Information Gain* dilakukan setelah perhitungan nilai *Entropy* karena pada perhitungan nilai *Information Gain* memerlukan inputan berupa nilai *Entropy* dan nilai *Information Gain* hasil perhitungan ini nantinya akan menjadi input perhitungan *Gain Ratio*. Berdasarkan perhitungan matematis *Information Gain* dari suatu fitur A dapat dilihat pada persamaan (2,2).

$$Gain(S,A) = Entropy(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} entropy(S_v) \quad (2,2)$$

A : fitur

V : menyatakan suatu nilai yang mungkin untuk fitur A

Values (A) : himpunan nilai-nilai yang mungkin untuk fitur A

$|S_v|$: jumlah sampel untuk nilai v

$|S|$: jumlah seluruh sampel data

$Entropy(S_v)$: *entropy* untuk sampel-sampel yang memiliki nilai v

c. Hitung nilai *Gain Ratio*

Untuk menghitung *Gain Ratio* diperlukan *split information*. *Split information* dihitung dengan persamaan (2,3).

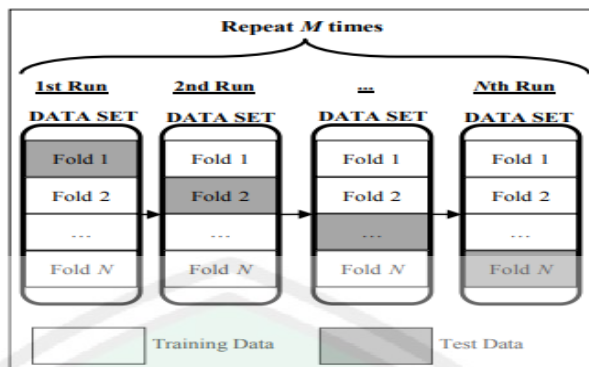
$$SplitInfo(S,A) = - \sum_{j=1}^c \frac{|S_j|}{|S|} \log_2 \frac{|S_j|}{|S|} \quad (2,3)$$

Dimana S_1 sampai S_c adalah c subset yang dihasilkan dari pemecahan S dengan menggunakan fitur A yang mempunyai banyak C nilai. Selanjutnya *Gain Ratio* dihitung dengan persamaan (2,4).

$$GainRatio(S,A) = \frac{Gain(S,A)}{SplitInfo(S,A)} \quad (2,4)$$

2.7 K-fold Cross Validation

K-fold Cross-Validation merupakan teknik pembagian data untuk proses testing suatu klasifikasi, *k-fold Cross-Validation* membagi data menjadi data *training* dan data *testing* dalam K bagian. Metode *k-fold Cross-Validation* bekerja dengan cara mempartisi himpunan data D secara acak menjadi k *fold* (subhimpunan) yang saling bebas : f_1, f_2, \dots, f_k , sehingga masing-masing *fold* berisi $1/k$ bagian data (Suyanto, 2017, p.243-244). Penjelasan lain oleh Arrar & Ayan (2017), teknik *cross-validation* ini akan melakukan pembagian dari keseluruhan data dibagi menjadi *subset* N ; satu *subset* digunakan sebagai data uji, *subset* tersisa $(N-1)$ digunakan sebagai data pelatihan. Proses ini diulang sebanyak N kali, untuk memungkinkan setiap *subset* digunakan sebagai data uji. Gambar 2.1 menunjukkan teknik dari *cross-validation*.



Gambar 2.1 Teknik M x N Cross-Validation

Ilustrasi dari proses *k-fold cross-validation* adalah sebagai berikut; misal data D akan dilakukan 5 *fold cross validation* maka data D akan menjadi data D_1 , D_2 , D_3 , D_4 dan D_5 . Data D_1 berisi empat fold: f_2 , f_3 , f_4 , dan f_5 untuk data latih serta satu *fold* f_1 untuk data uji. Himpunan data D_2 berisi *fold* f_1 , f_3 , f_4 , dan f_5 untuk data latih *fold* f_2 untuk data uji. Demikian seterusnya untuk himpunan data D_3 , D_4 , dan D_5 , sehingga setiap *fold* pernah menjadi data uji sebanyak satu kali (Suyanto, 2017, p.243-244).

2.8 Klasifikasi Naïve Bayes

Metode klasifikasi ini didasarkan pada teorema Bayes yang ditemukan oleh Thomas Bayes pada abad ke-18 dengan mengasumsikan independensi atau ketidaktergantungan yang kuat (naif) pada fitur yang ada artinya sebuah fitur pada sebuah data tidak saling berkaitan atau tidak memiliki hubungan apapun antara fitur yang satu dengan fitur lainnya, berikut adalah persamaan umum teorema Bayes (Prasetyo, 2012:59-62; Suyanto, 2017:126-129);

$$P(Y|X) = \frac{P(Y)P(X|Y)}{P(X)} \quad (2,5)$$

$P(Y|X)$: probabilitas akhir bersyarat, suatu hipotesis Y terjadi jika diberikan

bukti X terjadi

$P(X|Y)$: probabilitas sebuah bukti X terjadi akan mempengaruhi hipotesis Y

$P(Y)$: probabilitas awal hipotesis Y terjadi tanpa memandang bukti apapun

$P(X)$: probabilitas awal bukti X terjadi tanpa memandang hipotesis/bukti yang lain

Dasar dari aturan bayes adalah bahwa hasil dari hipotesis atau peristiwa (Y) dapat diperkirakan berdasarkan pada beberapa bukti (X) yang diamati, misalnya ada beberapa bukti sebagai berikut X_1, X_2, \dots, X_n , sehingga probabilitas akhir dituliskan sebagai;

$$P(Y|X_1, X_2, \dots, X_n) = \frac{P(X_1, X_2, \dots, X_n|Y) P(Y)}{P(X_1, X_2, \dots, X_n)} \quad (2,6)$$

Karena Naïve Bayes mengasumsikan bahwa setiap fitur independen maka persamaan (2,6) dapat diubah menjadi;

$$P(Y|X_1, X_2, \dots, X_n) = \frac{P(X_1|Y) \times P(X_2|Y) \times \dots \times P(X_n|Y) \times P(Y)}{P(X_1) \times P(X_2) \times \dots \times P(X_n)} \quad (2,7)$$

Berdasarkan persamaan (2,7) dengan k kelas maka persamaan Naïve Bayes untuk klasifikasi adalah sebagai berikut;

$$P(Y_k|X) = \frac{P(Y_k) \prod_{i=1}^n P(X_i|Y_k)}{P(X)} \quad (2,8)$$

$P(Y_k|X)$ merupakan probabilitas data dengan vektor X pada kelas Y_k . $P(Y_k)$ probabilitas awal kelas Y_k . $\prod_{i=1}^n P(X_i|Y_k)$ probabilitas independen kelas Y_k dari semua fitur (n) dalam vector X. Karena nilai $P(X)$ selalu tetap (artinya, tuple X memiliki probabilitas yang sama untuk masuk kedalam kelas manapun) sehingga

dalam perhitungan prediksi akan memaksimalkan $P(Y_k) \prod_{i=1}^n P(X_i|Y_k)$. Dalam perhitungan $P(X_i|Y_k)$ umumnya menggunakan fitur bertipe kategoris namun untuk fitur bertipe numerik (kontinu) ada 2 perlakuan yaitu;

- Melakukan diskritisasi terhadap fitur kontinu, mengganti nilai fitur kontinu dengan nilai interval diskrit.
- Membiarkan fitur tetap bertipe kontinu dan melakukan perhitungan sesuai dengan distribusi Gaussian yang didefinisikan sebagai;

$$P(X_i|Y_k) = \frac{1}{\sigma_{Y_k} \sqrt{2\pi}} e^{-\frac{(x-\mu_{Y_k})^2}{2\sigma_{Y_k}^2}} \quad (2,9)$$

Yang diketahui bahwa σ_{Y_k} dan μ_{Y_k} adalah standart deviasi dan rata-rata dari nilai-nilai pada fitur kontinu X_i untuk kelas Y_k .

Penentuan kelas hasil prediksi dengan memilih nilai terbesar dari probabilitas kelas hasil perhitungan $P(Y_k) \prod_{i=1}^n P(X_i|Y_k)$, jika ditulis dalam persamaan tampil sebagai berikut;

$$\arg \max P(Y_k) \prod_{i=1}^n P(X_i|Y_k) \quad (2,10)$$

Perlu diketahui bahwa algoritma ini dalam menentukan hasil klasifikasi berdasarkan perhitungan dari data latih, sehingga untuk dapat menggunakan algoritma ini harus memiliki data yang bisa digunakan untuk data latih dan pengujian dilakukan dengan menggunakan data *testing* atau data baru.

2.9 Evaluasi Klasifikasi

Hasil evaluasi ini akan menunjukkan informasi mengenai seberapa besar akurasi yang dicapai. Teknik evaluasi yang akan digunakan dikenal sebagai *confusion matrix*. *Confusion matrix* ini akan merepresentasikan kebenaran dari sebuah prediksi.

Confusion matrix didapat dengan menghitung hasil dari proses klasifikasi terhadap data yang ada, teknik yang bisa digunakan untuk menguji akurasi klasifikasi adalah dengan teknik klasifikasi menggunakan *cross-validation*, dengan teknik tersebut bisa mempermudah dalam mengukur akurasi algoritma klasifikasi dengan jumlah data yang besar. Penggunaan teknik *cross validation* cukup mudah karena user hanya perlu melakukan input seluruh data klasifikasi kemudian sistem akan membagi data inputan menjadi data *training* dan *testing* sesuai dengan teknik *cross validation*.

Dari hasil proses klasifikasi dengan teknik *cross validation* akan didapat sebuah *confusion matrix*. Arar & Ayan (2017) menjelaskan *confusion matrix* sebagai berikut;

Tabel 2.2 *Confusion Matrix*

		Kenyataan	
		+	-
Hasil Prediksi	+	<i>True Positive</i>	<i>False Positive</i>
	-	<i>False Negative</i>	<i>True Negative</i>

- **True Positive (TP)** adalah jumlah tuple positif yang dilabeli dengan benar oleh klasifikasi.
- **False Positive (FP)** jumlah tuple negative yang dilabeli salah oleh klasifikasi
- **False Negative (FN)** jumlah tuple positif yang dilabeli salah oleh klasifikasi.
- **True Negative (TN)** adalah jumlah tuple negatif yang dilabeli dengan benar oleh klasifikasi.

Berdasarkan Nilai dari *confusion matrix* tersebut bisa dihitung nilai akurasi suatu prediksi. Untuk perhitungannya didefinisikan dengan Persamaan (2,11).

$$Akurasi = \frac{TP + TN}{TP + TN + FP + FN} \quad (2,11)$$

2.10 Penelitian Terkait

Terdapat beberapa penelitian terdahulu yang mendukung antara lain sebagai berikut;

Penelitian yang dilakukan oleh Wahono (2014) yang berjudul “*Genetic Feature Selection for Software Defect Prediction*”, pada penelitian tersebut Wahono menerapkan seleksi fitur *Genetic Algorithm* (GA) untuk mendapatkan fitur yang relevan terhadap prediksi cacat perangkat lunak dan teknik *Bagging* untuk menangani *class imbalance*. Eksperimen untuk membuktikan pengaruh seleksi fitur GA dan *Bagging* dilakukan dengan melakukan klasifikasi terhadap 10 algoritma klasifikasi sebagai berikut; *Logistic Regression* (LR), *Linear Discriminant Analysis* (LDA), *Naïve Bayes* (NB), *k-nearest neighbor* (k-NN), K^* ,

Back Propagation (BP), *Support Vector Machine* (SVM), *C4.5, Classification and Regression Tree* (CART) dan *Random Forest* (RF). Berdasarkan hasil eksperimen yang telah dilakukan disimpulkan bahwa seleksi fitur GA dan *Bagging* menunjukkan peningkatan performa klasifikasi dibandingkan dengan tanpa menerapkan seleksi fitur GA dan *Bagging*. Jika ditinjau dari persamaan dan perbedaan antara penelitian yang dilakukan oleh Wahono dengan penelitian ini maka diketahui yaitu sama-sama untuk prediksi cacat perangkat lunak dan uji coba menggunakan klasifikasi NB, sedangkan perbedaannya adalah pada penelitian Wahono metode seleksi fitur yang digunakan GA sedangkan pada penelitian ini menggunakan seleksi fitur GR dan perbedaan yang lain terletak pada data yang digunakan juga berbeda.

Penelitian lain dilakukan oleh Wang *et.al* (2012) dengan judul “*Software measurement data reduction using ensemble techniques*”, pada penelitian tersebut menggunakan 17 teknik seleksi fitur berbasis *ensemble* yang diterapkan pada 16 *dataset* berbeda yang kemudian dilakukan klasifikasi untuk mengetahui performa dari hasil seleksi fitur dan membandingkan dengan performa tanpa seleksi fitur menggunakan algoritma klasifikasi sebagai berikut; *Logistic Regression* (LR), *Naïve Bayes* (NB), *k-nearest neighbor* (k-NN) dan *Multilayer Perceptron* (MLP). Berdasarkan eksperimen yang telah dilakukan disimpulkan bahwa tidak ada metode *ensemble* yang mendominasi, namun secara umum bisa disimpulkan bahwa *ensemble* dengan sedikit *rangker* sama atau lebih baik dari pada banyak *rangker* atau bahkan seluruh *rangker*. Jika ditinjau dari persamaan dan perbedaan antara penelitian yang dilakukan oleh Wang *et.al* dengan penelitian ini maka diketahui yaitu sama-sama untuk prediksi cacat perangkat lunak dan uji coba

sama-sama menggunakan klasifikasi NB, sedangkan perbedaannya adalah pada penelitian Wang *et.al* metode seleksi fitur dengan teknik *ensemble* yang berjumlah 17 seleksi fitur *ensemble* sedangkan pada penelitian ini menggunakan seleksi fitur GR dan perbedaan yang lain terletak pada data yang digunakan juga berbeda.

Penelitian lain juga dilakukan oleh Karabulut *et.al* (2012) dengan judul “*A comparative study on the effect of feature selection on classification accuracy*”, pada penelitian tersebut Karabulut *et.al* melakukan perbandingan beberapa seleksi fitur antara lain *Information Gain* (IG), *Gain Ratio* (GR), *Symmetrical Uncertainty* (SU), *Relief-F* (RF), *One-R*, *Chi-square* (CS) yang diterapkan untuk mendapatkan fitur yang relevan pada 15 *dataset* dan untuk melakukan pengujian dilakukan pengukuran performa terhadap beberapa algoritma klasifikasi antara lain; Naïve Bayes (NB), *Multilayer Perceptron* (MLP) dan *decision tree* J48. Berdasarkan eksperimen yang telah dilakukan disimpulkan bahwa dari 6 seleksi fitur yang diteliti, berikut adalah seleksi fitur yang menghasilkan performa klasifikasi paling baik; klasifikasi Naïve Bayes dengan seleksi fitur *Gain Ratio*, klasifikasi *Multilayer Perceptron* dengan seleksi fitur *Chi-square* dan klasifikasi J48 dengan seleksi fitur *Information Gain*. Jika ditinjau dari persamaan dan perbedaan antara penelitian yang dilakukan oleh Karabulut *et.al* dengan penelitian ini maka diketahui yaitu sama-sama menggunakan seleksi fitur *Gain Ratio* dan juga sama-sama melakukan uji coba dengan klasifikasi Naïve Bayes, sedangkan perbedaannya terletak pada data yang digunakan.

BAB III

ANALISIS DAN PERANCANGAN

Pada bab ini akan dijelaskan terkait analisis dan perancangan sistem yang diusulkan pada penelitian ini.

3.1 Analisis Sistem

Berdasarkan penjelasan pada bab I, penelitian ini berupaya untuk menentukan atau memilih fitur yang relevan terhadap prediksi cacat perangkat lunak dari keseluruhan fitur yang ada pada *dataset* NASA MDP karena diketahui bahwa *dataset* yang digunakan dibuat tidak khusus untuk prediksi cacat perangkat lunak.

Dataset NASA MDP yang digunakan pada penelitian ini diunduh dari website PROMISE (*Predictor Models in Software Engineering*) repository melalui alamat <http://promise.site.uottawa.ca/SERepository>. *Dataset* yang didapat berupa data yang menyimpan *log* cacat perangkat lunak yang setiap baris menunjukkan modul terkecil dari sebuah program yaitu berupa *method* atau *function* dan setiap kolom menunjukkan nilai metrik. *Dataset* yang akan digunakan pada penelitian ini berjumlah 5 *dataset* yaitu *dataset* CM1, JM1, KC1, KC2 dan PC1. Sebagai gambaran terkait isi dari *dataset* yang akan digunakan pada penelitian ini penulis menunjukkan salah satu isi *dataset* dari dataset CM1 yang ditunjukkan pada tabel 3.1.

Tabel 3.1 Isi *Dataset CM1*

No	loc	v(g)	ev(g)	iv(g)	defect
1	1.1	1.4	1.4	1.4	...	False
2	1	1	1	1	...	True
3	24	5	1	3	...	False
...
498	28	6	5	5	...	True

Setelah data yang dibutuhkan berhasil didapatkan langkah selanjutnya penulis melakukan beberapa pengkajian berbagai referensi yang berkaitan dengan klasifikasi cacat perangkat lunak dengan berbagai metode yang digunakan dari masing-masing penelitian sebelumnya. Studi pustaka ini diharapkan dapat memberikan gambaran secara lengkap terkait penelitian dan dapat memberikan dasar kontribusi untuk prediksi cacat perangkat lunak yang akan dilakukan pada penelitian ini. Berikut ini merupakan beberapa kajian referensi studi pustaka yang dilakukan:

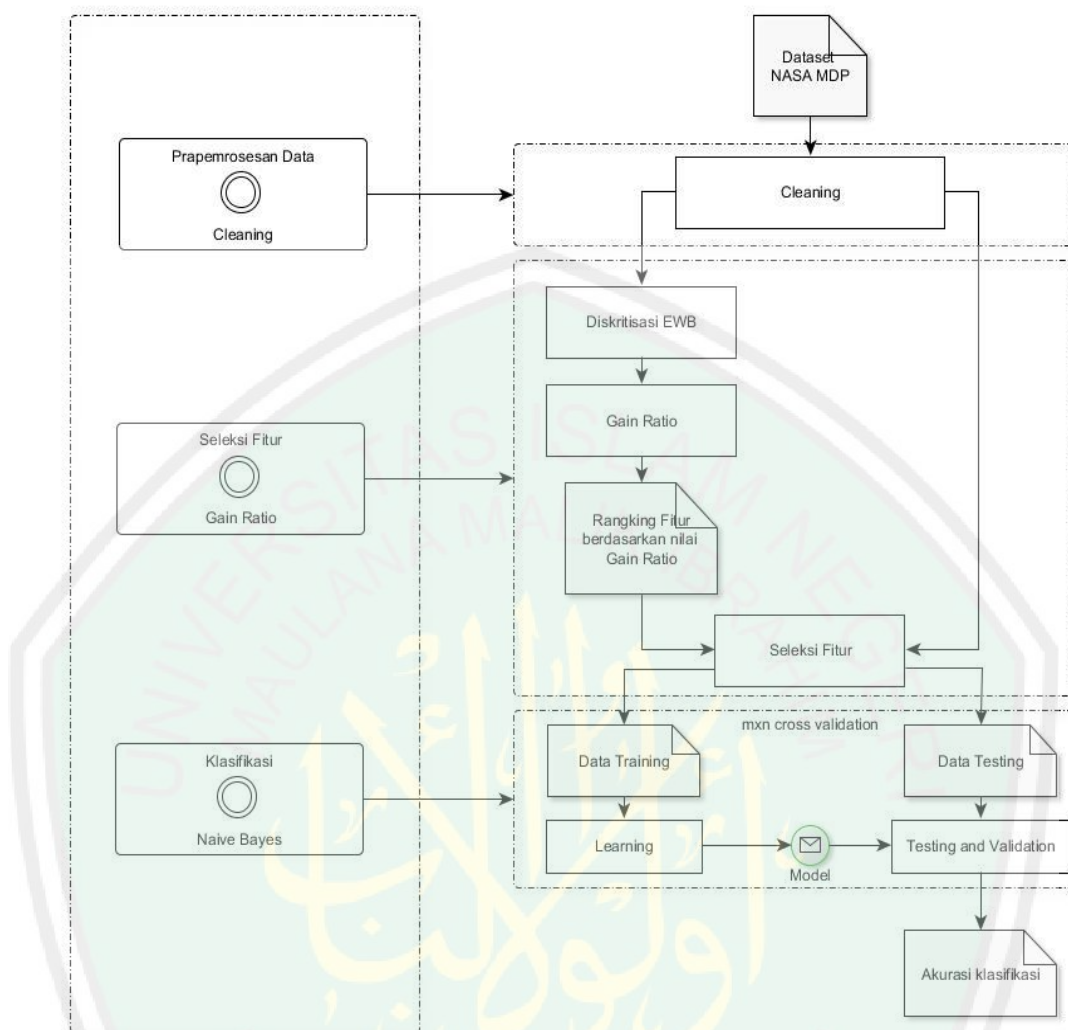
- a. Penelitian terdahulu yang telah melakukan prediksi cacat perangkat lunak berdasarkan *dataset* kode program dan menerapkan seleksi fitur terhadap *dataset* yang digunakan.
- b. Metode diskritisasi dengan *Equal Width Binning*.
- c. Metode seleksi fitur dengan *Gain Ratio*.
- d. Klasifikasi Naïve Bayes.
- e. Evaluasi klasifikasi

Setelah melakukan studi literatur penulis mendapat gambaran tentang proses prediksi cacat perangkat lunak dan pada penelitian ini penulis mengusulkan alur proses yang ditunjukkan pada gambar 3.1.

Urutan proses yang penulis usulkan adalah pertama *Dataset* yang telah didapat nantinya menjadi input pada penelitian ini yang kemudian dilakukan *cleaning* untuk menghilangkan *missing value*, selanjutnya dilakukan diskritisasi dan perhitungan nilai *Gain Ratio* dengan hasil berupa urutan fitur yang relevan berdasarkan nilai *Gain Ratio*. Setelah fitur yang relevan sudah terurut maka *dataset* yang asli diurutkan sesuai dengan urutan fitur yang relevan berdasarkan nilai *Gain Ratio* dan dilakukan seleksi atau pengambilan fitur dengan jumlah tertentu yang pada penelitian ini penulis melakukan pengambilan jumlah fitur sebanyak 1 fitur, 2 fitur, 3 fitur, dan seterusnya sampai keseluruhan fitur. Tahap setelah pengambilan fitur adalah proses klasifikasi dengan teknik *10 fold cross validation* terhadap algoritma klasifikasi Naïve Bayes, hasil dari proses ini berupa akurasi klasifikasi.

3.2 Perancangan Sistem

Pada bagian ini akan dibahas tentang alur sistem yang diusulkan, sesuai dengan analisis yang telah dilakukan maka alur sistem yang diusulkan ditunjukkan pada gambar 3.1.



Gambar 3.1 Desain Sistem yang diusulkan

3.2.1 Proses Cleaning

Dataset yang didapat memiliki kemungkinan terdapat *missing value*, oleh karena itu tahap ini dilakukan untuk memperbaiki *dataset* yang terdapat *missing value* dengan cara mengisi data *missing value* dengan nilai rata-rata dari fitur tersebut. Pada penelitian ini untuk proses *cleaning* memanfaatkan fungsi dari *library Weka* yaitu `weka.filters.unsupervised.attribute.ReplaceMissingValues`. Hasil dari proses ini berupa *dataset* yang sudah tidak lagi terdapat *missing value*.

3.2.2 Proses Diskritisasi dengan *Equal Width Binning*

Proses ini dilakukan untuk merubah data dari setiap fitur yang bersifat kontinyu menjadi data diskrit karena data diskrit dibutuhkan untuk proses selanjutnya yaitu seleksi fitur dengan *Gain Ratio*, selain itu menurut Singh & Verma (2009) bahwa klasifikasi Naïve Bayes memiliki akurasi yang lebih baik jika menggunakan data diskrit.

Seperti yang telah dijelaskan pada bab 2 tentang diskritisasi dengan *Equal Width Binning*, adapun tahapan melakukan diskritisasi data meliputi;

- Tentukan nilai $K = \max\{1, 2 \log L\}$ jika K bernilai 1 maka diskritisasi data menjadi all.
- Hitung $W = (V_{\max} - V_{\min}) / K$.
- Tentukan titik potong dengan perhitungan;
 $V_{\min} + W, V_{\min} + 2W, \dots, V_{\min} + (K-1)W$
- Langkah terakhir adalah menentukan nilai dari *dataset* termasuk dalam rentang yang sesuai.

Berdasarkan alur tahap perhitungan yang telah dijelaskan jika dibentuk dalam sebuah pseudocode tampil seperti berikut;

```

Algoritma equal_width_binning(dataset[][])
//input : Array dataset[][]
//Asumsi : Fungsi yang sudah ada sorttingasc(x),
//countDistinct(x[]) dan max(x) membulatkan x keatas
//Output : Data hasil diskritisasi

K ← 0, L ← 0, W ← 0, dataBeda[] //inisial
cutPoints[][] //Float
data [][] ← dataset //Float
dataDiskrit[dataset.length][dataset[0].length]//String
For i ← 0 : data.length do
    sorttingasc(data[i])
    Vmin ← data[i][0]
  
```



```

Vmax ← data[i][data[i].length - 1]
L ← countDistinct(data[i].length)
K ←  $\max(1, 2 \log L)$ 
IF K = 1 Then
    cutPoints ← All //String
Else
    W ← (Vmax - Vmin)/K
    For m ← 1 : K+1 do
        cutPoints ← Vmin + (m x W)
    EndFor
EndIf
dataBeda ← null
L ← 0
K ← 0
W ← 0
EndFor

For i ← 0 : dataset.length do
    For j ← 0 : dataset[i].length do
        For m ← 0 : cutPoints[i].length do
            IF m = (cutPoints[i].length - 1) Then
                Do nothing
            Else
                a ← cutPoints[i][m]
                b ← cutPoints[i][m+1]
                IF data[i][j] ≤ a && dataDiskrit[i][j] ==
                    null Then
                    range ← "(≤"+a+)"
                    dataDiskrit[i][j] ← range
                EndIf
                IF data[i][j] > a && data[i][j] ≤ b Then
                    range ← a + "< & ≤"+ b
                    dataDiskrit[i][j] ← range
                EndIf
                IF data[i][j] > a && dataDiskrit[i][j] ==
                    null Then
                    range ← ("+a"<)"
                    dataDiskrit[i][j] ← range
                EndIf
            EndIf
        EndFor
    EndFor
EndFor
return dataDiskrit

```

Berikut adalah contoh perhitungan secara manual dari proses diskritisasi dengan *Equal Width Binning*. Contoh perhitungan manual ini penulis menggunakan *dataset* CM1.

Tabel 3.2 Dataset CM1

No	loc	v(g)	ev(g)	iv(g)	defect
1	1.1	1.4	1.4	1.4	...	False
2	1	1	1	1	...	True
3	24	5	1	3	...	False
...
498	28	6	5	5	...	True

Diketahui:

L (Jumlah Nilai berbeda) untuk fitur ke-1=102, ke-2=34, ke-3=18, ..., ke-n

Vmin fitur ke-1 = 1, ke-2 = 1, ke-3 = 1, ..., ke-N

Vmax fitur ke-1 = 423, ke-2 = 96, ke-3 = 30, ..., ke-N

Perhitungan untuk fitur ke-1;

Langkah 1, hitung nilai $K = \max\{1, 2 \log L\}$

$$K = \max(1, 2 \log 102) \rightarrow K = \max(2.41) \rightarrow K = 3$$

Langkah 2, hitung nilai $W = (V_{\max} - V_{\min}) / K$

$$W = (423 - 1) / 3 \rightarrow W = 140.66666666666666$$

Langkah 3, tentukan titik potong $V_{\min} + W, V_{\min} + 2W, \dots, V_{\min} + (K-1)W$

$$V_{\min} + W \rightarrow 1 + 140.66666666666666 = 141.66667$$

$$V_{\min} + 2W \rightarrow 1 + (2 \times 140.66666666666666) = 282.33334$$

Karena $K - 1 = 2$ titik potong sudah terhitung semua, hasilnya;

$$\geq 141.66667, 141.66667 < \& \leq 282.33334 \text{ dan } 282.33334 <$$

Langkah 4, menentukan nilai *dataset* termasuk dalam interval yang sesuai

Tabel 3.3 Hasil diskritisasi fitur ke-1

No	loc	v(g)	ev(g)	iv(g)	defect
1	≥ 141.66667	1.4	1.4	1.4	...	False
2	≥ 141.66667	1	1	1	...	True
3	≥ 141.66667	5	1	3	...	False
...
498	≥ 141.66667	6	5	5	...	True

Ulangi langkah 1 sampai 4 untuk semua fitur sehingga hasil akhir dari proses diskritisasi bisa dilihat pada tabel 3.4.

Tabel 3.4 Hasil diskritisasi semua fitur dataset CM1

No	loc	v(g)	ev(g)	iv(g)	defect
1	≥ 141.66667	≥ 48.5	≥ 15.5	≥ 32.0	...	False
2	≥ 141.66667	≥ 48.5	≥ 15.5	≥ 32.0	...	True
3	≥ 141.66667	≥ 48.5	≥ 15.5	≥ 32.0	...	False
...
498	≥ 141.66667	≥ 48.5	≥ 15.5	≥ 32.0	...	True

3.2.3 Proses seleksi fitur dengan *Gain Ratio*

Setiap *dataset* terdiri dari beberapa fitur, namun dari seluruh fitur yang ada terdapat beberapa fitur yang memiliki pengaruh yang besar namun juga ada yang tidak begitu berpengaruh pada hasil prediksi. Tahap ini dilakukan untuk memilih sebagian dari fitur yang memiliki pengaruh besar terhadap proses prediksi, seleksi fitur ini berdasarkan perankingan nilai *Gain Ratio*. Semakin besar nilai *Gain Ratio* maka fitur tersebut memiliki pengaruh yang besar terhadap proses prediksi.

Dari keseluruhan fitur akan dipilih beberapa fitur saja yang memiliki pengaruh terhadap proses prediksi berdasarkan perankingan nilai *Gain Ratio*.

Sesuai dengan penjelasan pada bab 2 adapun proses perhitungan nilai *Gain Ratio* adalah sebagai berikut;

Tahap 1, menghitung nilai *Entropy* setiap fitur dengan persamaan (2,1).

Tahap 2, menghitung nilai *Information Gain* setiap fitur dengan persamaan (2,2).

Tahap 3, menghitung nilai *Split Information* setiap fitur dengan persamaan (2,3).

Tahap 4, menghitung nilai *Gain Ratio* setiap fitur dengan persamaan (2,4).

Tahap terakhir adalah melakukan perankingan berdasarkan nilai *Gain Ratio* dan mengambil beberapa fitur yang dianggap memiliki pengaruh terhadap proses prediksi, jumlah fitur yang dipilih sesuai dengan keinginan user.

Berdasarkan alur tahapan yang telah dijelaskan jika dibentuk dalam sebuah pseudocode tampil seperti berikut;

```

Algoritma Gain_Ratio(dataset[][])
//input : Array dataset yang sudah didiskritisasi
//Asumsi : Fungsi yang sudah ada getFitur(x[]),
//getKeputusan(x[]), //HitungEntropy(x[]),
//HitungGain(x,y[]), HitungSplitInfo(x[]), //HitungGR(x,y),
reset(), sortDesc(x,y)
//Output : Nilai Gain Ratio setiap fitur

GR[], Ent[], Gain[], SplitInfo = 0//inisial
fitur[][] ← getFitur(dataset) //inisial
Keputusan[] ← getKeputusan(dataset)
EntTotal ← HitungEntropy(keputusan)
For i ← 0 : fitur.length do
    Ent ← HitungEntropy(fitur[i])
    Gain ← HitungGain(EntTotal, Ent)
    SplitInfo ← HitungSplitInfo(dataset)
    GR[i] ← HitungGR(Gain, SplitInfo)
    Reset() //mensest ulang Ent[], Gain[], SplitInfo
EndFor
GR ← sortDesc(GR, 5)

return GR

```

Contoh perhitungan nilai *Gain Ratio* berdasarkan dari data yang telah didiskritisasi pada proses sebelumnya.

Langkah 1, menghitung nilai *Entropy*

Entropy(Defect)

$$= \left(\left(\frac{-449}{498} \right) \log_2 \frac{449}{498} \right) + \left(\left(\frac{-49}{498} \right) \log_2 \frac{49}{498} \right) = 0.46388254$$

Entropy Fitur loc

Entropy((-inf - 141.66667))

$$= \left(\left(\frac{-440}{485} \right) \log_2 \frac{440}{485} \right) + \left(\left(\frac{-45}{485} \right) \log_2 \frac{45}{485} \right) = 0.4456932$$

Entropy((141.66667 - 282.33334))

$$= \left(\left(\frac{-7}{10} \right) \log_2 \frac{7}{10} \right) + \left(\left(\frac{-3}{10} \right) \log_2 \frac{3}{10} \right) = 0.8812909$$

Entropy((282.33334 - inf))

$$= \left(\left(\frac{-2}{3} \right) \log_2 \frac{2}{3} \right) + \left(\left(\frac{-1}{3} \right) \log_2 \frac{1}{3} \right) = 0.91829586$$

Entropy Fitur v(g)

Entropy((-inf-48.5))

$$= \left(\left(\frac{-447}{495} \right) \log_2 \frac{447}{495} \right) + \left(\left(\frac{-48}{495} \right) \log_2 \frac{48}{495} \right) = 0.45931554$$

Entropy((48.5-inf))

$$= \left(\left(\frac{-2}{3} \right) \log_2 \frac{2}{3} \right) + \left(\left(\frac{-1}{3} \right) \log_2 \frac{1}{3} \right) = 0.91829586$$

Ulangi langkah 1 untuk fitur selanjutnya hingga mendapat nilai entropy seluruh fitur!

Langkah 2, menghitung nilai *Information Gain*

***Information Gain* Fitur loc**

Gain(Defect, loc)

$$= 0.46388254$$

$$- \left(\left(\frac{485}{498} \right) 0.4456932 + \left(\frac{10}{498} \right) 0.8812909 + \left(\frac{3}{498} \right) 0.91829586 \right)$$

$$= 0.006595403$$

***Information Gain* Fitur v(g)**

Gain(Defect, v(g))

$$= 0.46388254 - \left(\left(\frac{495}{498} \right) 0.45931554 + \left(\frac{3}{498} \right) 0.91829586 \right)$$

$$= 0.001802057$$

Ulangi langkah 2 untuk fitur selanjutnya hingga mendapat nilai *Information*

Gain seluruh fitur!

Langkah 3, menghitung nilai *Split Information*

***SplitInfo*(loc)**

$$= - \left(\frac{485}{498} \right) \log_2 \frac{485}{498} - \left(\frac{10}{498} \right) \log_2 \frac{10}{498} - \left(\frac{3}{498} \right) \log_2 \frac{3}{498}$$

$$= 0.19480705$$

SplitInfo(v(g))

$$= -\left(\frac{495}{498}\right)\log_2\frac{495}{498} - \left(\frac{3}{498}\right)\log_2\frac{3}{498} = 0.05309269$$

Ulangi langkah 3 untuk fitur selanjutnya hingga mendapat nilai *Split Information* seluruh fitur!

Langkah 4, menghitung nilai *Gain Ratio*

GainRatio(loc)

$$= \frac{0.006595403}{0.19480705} = 0.03385608$$

GainRatio(v(g))

$$= \frac{0.001802057}{0.05309269} = 0.033941716$$

Ulangi langkah 4 untuk fitur selanjutnya hingga mendapat nilai *Gain Ratio* seluruh fitur!

Langkah terakhir, melakukan perankingan sesuai dengan nilai *Gain Ratio* setiap fitur. Perankingan ini dilakukan bertujuan untuk mengetahui urutan dari fitur yang memiliki pengaruh terhadap proses prediksi cacat perangkat lunak. Sudah dijelaskan bahwa semakin besar nilai *Gain Ratio* maka fitur tersebut memiliki pengaruh yang besar terhadap proses prediksi. Hasil dari proses ini dapat dilihat pada Tabel 3.5 yang menunjukkan hasil perankingan fitur sesuai dengan nilai *Gain Ratio* pada *dataset* CM1.

Tabel 3.5 Hasil perankingan sesuai nilai *Gain Ratio* pada dataset CM1

GR	Fitur
0.17764027	<i>Time to write program (t)</i>
0.17764027	<i>Effort to write program (e)</i>
0.08940102	<i>Count of lines of comments (IOComment)</i>
0.07676358	<i>Unique operators (UniqOp)</i>
0.066612095	<i>Unique operands (UniqOpnd)</i>
0.06614874	<i>Error estimate (b)</i>
0.04657121	Volume (v)
0.03866517	<i>Essential complexity (ev(g))</i>
0.034703486	<i>Intelligence (i)</i>
0.033941716	<i>Design complexity (iv(g))</i>
0.033941716	<i>Cyclomatic complexity (v(g))</i>
0.033941716	<i>Branch count (branchCount)</i>
0.03385608	Total operand (TotalOpnd)
0.03385608	<i>Line of code (loc)</i>
0.032714628	Total operator + operand (n)
0.028663296	Total operator (TotalOp)
0.022667497	<i>Difficulty (d)</i>
0.017010132	<i>Count of blank lines (IOBlank)</i>
0.001916128	<i>Count of Statement Lines (IOCode)</i>
0.000617	<i>Program length (L)</i>
0.0	<i>Count of Code and Comments Lines (IOCodeAndComment)</i>

Jumlah seleksi fitur tergantung yang ditinginkan oleh user, pada penelitian ini penulis menggunakan asumsi pengambilan fitur dengan jumlah pengambilan fitur sebanyak 1 fitur, 2 fitur, 3 fitur, dan seterusnya sampai keseluruhan fitur, hal ini dilakukan untuk melakukan percobaan dengan jumlah yang berbeda untuk menentukan jumlah yang terbaik. Sebagai contoh pada perhitungan manual ini penulis mengambil jumlah fitur sebanyak 5 fitur, yang hasilnya ditunjukkan pada tabel 3.6.

Tabel 3.6 Hasil seleksi 5 fitur pada dataset CM1

Nilai GR	Fitur
0.17764027	<i>Time to write program (t)</i>
0.17764027	<i>Effort to write program (e)</i>
0.08940102	<i>Count of lines of comments (IOComment)</i>
0.07676358	<i>Unique operators (uniq_Op)</i>
0.066612095	<i>Unique operands (uniq_Opnd)</i>

3.2.4 Klasifikasi Naïve Bayes

Klasifikasi ini merupakan tahap untuk melakukan prediksi berdasarkan data yang ada dengan teknik klasifikasi yang dilakukan dengan menggunakan algoritma klasifikasi Naïve Bayes. Sebelum proses klasifikasi Naïve Bayes dilakukan data akan dibagi menjadi data *training* dan juga data *testing* dengan teknik *cross validation* yang telah dijelaskan pada bab 2. Proses klasifikasi dilakukan untuk *dataset* dengan seleksi fitur *Gain Ratio* yang mengambil jumlah fitur berbeda-beda.

Pada penelitian ini proses klasifikasi Naïve Bayes menggunakan fungsi yang disediakan oleh *WEKA*, alasan penelitian ini menggunakan klasifikasi yang sudah ada karena fokus dari penelitian ini terdapat pada proses seleksi fitur. Fungsi *WEKA* dan parameter yang digunakan adalah sebagai berikut;

- `weka.classifiers.bayes.NaiveBayes`
- semua parameter; default

Proses klasifikasi dengan Naïve Bayes pada penelitian ini dilakukan dengan 10 *cross-validation*, sehingga dari data yang ada akan dibagi menjadi 10 set bagian dan dari 10 set akan diambil 1 set sebagai *testing* dan 9 set digunakan sebagai *training*. Proses ini diulang sebanyak 10 kali klasifikasi dengan masing-masing data set *training* dan *testing* yang berubah-ubah setiap klasifikasi sehingga keseluruhan data memungkinkan untuk dijadikan sebagai *testing* dan juga *training*. Berdasarkan percobaan dengan *dataset* CM1 yang telah melalui proses seleksi fitur *Gain Ratio* hasil klasifikasi Naïve Bayes dengan 10 *cross validation* menghasilkan *confusion matrix* pada tabel 3.7.

3.2.5 Perhitungan Akurasi

Seperti yang telah dijelaskan pada bab 2 bahwa proses evaluasi klasifikasi dilakukan dengan melakukan perhitungan berdasarkan *confusion matrix*. Pada tahap ini melakukan perhitungan sesuai dengan *confusion matrix* untuk mendapatkan nilai akurasi dari proses klasifikasi pada *dataset* dengan seleksi fitur *Gain Ratio* dan tanpa seleksi fitur *Gain Ratio*. Berikut salah satu contoh perhitungan akurasi dari proses prediksi cacat perangkat lunak pada *dataset* CM1 dengan 5 fitur hasil seleksi fitur *Gain Ratio*.

Tabel 3.7 *Confusion Matrix* klasifikasi *dataset* CM1 dengan 5 fitur hasil seleksi fitur GR

		Kenyataan	
		TRUE	FALSE
Hasil Prediksi	TRUE	14	27
	FALSE	35	442

Berdasarkan tabel 3.7 terlihat bahwa hasil pencocokan antara hasil klasifikasi oleh Naïve Bayes dengan kenyataan menunjukkan bahwa hasil prediksi “TRUE” dan kenyataan “TRUE” berjumlah 14, hasil prediksi “TRUE” dan kenyataan “FALSE” berjumlah 27, hasil prediksi “FALSE” dan kenyataan “TRUE” berjumlah 35 dan hasil prediksi “FALSE” dan kenyataan “FALSE” berjumlah 442. Sesuai dengan rumus pada persamaan (2,11) maka perhitungan akurasi dari *confusion matrix* tabel 3.7 sebagai berikut;

$$\text{Akurasi} = \frac{14 + 442}{14 + 442 + 27 + 35} = \frac{456}{498} = 0.8755 \times 100\% = 87.55\%$$

BAB IV

UJI COBA DAN PEMBAHASAN

Pada bab ini akan dibahas terkait uji coba dari rancangan atau desain sistem yang telah dibuat pada bab sebelumnya. Penjelasan pertama pada bab ini adalah tentang uji coba penelitian yang berisi tentang lingkungan uji coba, data yang digunakan, tampilan sistem yang berhasil dibuat, hasil uji coba seleksi fitur dan hasil uji coba klasifikasi. Penjelasan selanjutnya yaitu tentang pembahasan hasil dari uji coba yang telah dilakukan.

4.1 Uji Coba

Sebelum menjelaskan proses uji coba terlebih dahulu akan dijelaskan hal-hal yang berkaitan terhadap proses uji coba yaitu lingkungan uji coba dan juga data yang digunakan untuk uji coba.

4.1.1 Lingkungan Uji Coba

Lingkungan uji coba menjelaskan tentang spesifikasi perangkat keras dan perangkat lunak yang digunakan dalam penelitian ini, adapun spesifikasi perangkat keras yang digunakan adalah sebagai berikut;

- a. *Processor AMD FX-7600P Radeon R7, 12 Compute Cores 4C+8G 2.70 GHz*
- b. *Memory 4.00 GB*
- c. *Hardisk 1 TB*

Sedangkan spesifikasi perangkat lunak yang digunakan adalah sebagai berikut;

- a. *Sistem operasi windows 8.1 Pro 64bit*
- b. *Weka versi 3.7.2*
- c. *Netbeans 8.2 dan Ms.excel 2010*

4.1.2 Data Uji coba

Data yang digunakan pada penelitian ini berupa *dataset* yang berjumlah 5 *dataset* yaitu CM1, JM1, KC1, KC2 dan PC1. Berikut adalah 5 *dataset* yang digunakan pada penelitian ini;

Tabel 4.1 *Dataset* CM1

No	loc	v(g)	ev(g)	iv(g)	defect
1	1.1	1.4	1.4	1.4	...	False
2	1	1	1	1	...	True
3	24	5	1	3	...	False
...
498	28	6	5	5	...	True

Tabel 4.2 *Dataset* JM1

No	loc	v(g)	ev(g)	iv(g)	defect
1	1.1	1.4	1.4	1.4	...	False
2	1	1	1	1	...	True
3	72	7	1	6	...	True
...
10885	19	3	1	1	...	False

Tabel 4.3 *Dataset* KC1

No	loc	v(g)	ev(g)	iv(g)	defect
1	1.1	1.4	1.4	1.4	...	False
2	1	1	1	1	...	True
3	83	11	1	11	...	True
...
2109	11	2	1	2	...	False

Tabel 4.4 *Dataset KC2*

No	loc	v(g)	ev(g)	iv(g)	defect
1	1.1	1.4	1.4	1.4	...	no
2	1	1	1	1	...	yes
3	415	59	50	51	...	yes
...
522	3	1	1	1	...	yes

Tabel 4.5 *Dataset PC1*

No	loc	v(g)	ev(g)	iv(g)	defect
1	1.1	1.4	1.4	1.4	...	False
2	1	1	1	1	...	True
3	91	9	3	2	...	True
...
1109	26	18	13	6	...	False

Berdasarkan data setiap *dataset* maka Tabel 4.6 dan tabel 4.7 menunjukkan informasi 5 *dataset* yang digunakan pada penelitian ini. Informasi yang ditunjukkan antara lain jumlah total modul setiap *dataset*, jumlah modul yang cacat dan jumlah modul yang dinyatakan tidak cacat serta informasi tentang fitur-fitur yang ada pada setiap *dataset*.

Tabel 4.6 *Dataset Penelitian*

<i>Dataset</i>	Jumlah total modul	Jumlah modul cacat	Jumlah modul tidak cacat	Jumlah fitur
CM1	498	449	49	21
JM1	10885	8779	2106	21
KC1	2109	1783	326	21
KC2	522	415	107	21
PC1	1109	1032	77	21

Tabel 4.7 Fitur setiap *dataset*

No	Fitur	Dataset				
		CM1	JM1	KC1	KC2	PC1
1	loc	√	√	√	√	√
2	v(g)	√	√	√	√	√
3	ev(g)	√	√	√	√	√
4	iv(g)	√	√	√	√	√
5	uniq_Op	√	√	√	√	√
6	uniq_Opnd	√	√	√	√	√
7	total_Op	√	√	√	√	√
8	total_Opnd	√	√	√	√	√
9	n	√	√	√	√	√
10	v	√	√	√	√	√
11	l	√	√	√	√	√
12	d	√	√	√	√	√
13	i	√	√	√	√	√
14	e	√	√	√	√	√
15	b	√	√	√	√	√
16	t	√	√	√	√	√
17	IOCode	√	√	√	√	√
18	IOComment	√	√	√	√	√
19	IOBlank	√	√	√	√	√
20	locCodeAndComment	√	√	√	√	√
21	branchCount	√	√	√	√	√

Berdasarkan tabel 4.7 dapat diketahui bahwa semua *dataset* yang digunakan menggunakan fitur yang sama dalam melakukan prediksi cacat perangkat lunak dengan jumlah 21 fitur yaitu fitur loc, v(g), ev(g), iv(g), uniq_Op, uniq_Opnd, total_Op, total_Opnd, n, v, l, d, I, e, b, t, IOCode, IOComment, IOBlank, IOCodeAndComment dan branchCount.

4.1.3 Tampilan Sistem

Berdasarkan dari rencana yang telah dijelaskan pada desain sistem yang meliputi proses *cleaning*, proses diskritisasi *Equal Width Binning*, proses seleksi fitur *Gain Ratio*, proses klasifikasi dengan Naïve Bayes dan terakhir proses perhitungan akurasi. Proses-proses tersebut diimplementasikan dalam bahasa pemrograman java yang dibuat dengan tampilan ditunjukkan pada gambar 4.1.

The screenshot shows a Java application window titled "Aplikasi" with the following content:

**PENENTUAN FITUR YANG RELEVAN TERHADAP PREDIKSI
CACAT PERANGKAT LUNAK BERDASARKAN
SELEKSI FITUR GAIN RATIO**

Seleksi Fitur

Buttons and status: "Open File" (ELURUH FITURjm1.csv), "Cleaning" (Proses selesai), "Diskritisasi" (Proses selesai), "Gain Ratio" (Proses selesai), "Naive Bayes" (Proses selesai).

Hasil Perankingan Gain Ratio

Gain Ratio	Atribut
0.19059384	b
0.1876871	IOCode
0.18379816	loc
0.1820183	unig_Opnd
0.17562628	unig_Op
0.15953024	locCodeAndComment
0.14623563	v(g)
0.14276169	branchCount
0.13281156	t
0.13281153	e
0.12094725	iv(g)
0.0979709	total_Opnd
0.09328442	v
0.09065093	total_Op
0.08561899	n
0.07875244	IOComment
0.067217074	ev(g)
0.060341917	i
0.05631962	IOBlank
0.031671084	d
0.019286584	l

Hasil Klasifikasi

Jml Fitur	Akurasi
1	80.54%
2	80.28%
3	80.59%
4	80.45%
5	80.59%
6	80.51%
7	80.52%
8	80.55%
9	80.54%
10	80.63%
11	80.59%
12	80.61%
13	80.57%
14	80.53%
15	80.51%
16	80.46%
17	80.43%
18	80.40%
19	80.47%
20	80.45%
21	80.42%

Gambar 4.1 Tampilan Sistem

Berdasarkan gambar 4.1 dapat dilihat bahwa terdapat beberapa *button* yang berguna untuk menjalankan sebuah proses sesuai dengan algoritma yang telah dijelaskan pada bab sebelumnya. Alur untuk menjalankan sistem tersebut dimulai dari klik *button* "open file" untuk mengambil file yang akan digunakan

selanjutnya klik *button* “cleaning” tunggu sampai tulisan proses selesai muncul, kemudian klik *button* “diskritisasi” untuk menjalankan proses diskritisasi tunggu hingga muncul proses selesai. Selanjutnya klik *button* “Gain Ratio” untuk menjalankan proses perangkingan sesuai nilai Gain Ratio, tunggu hingga muncul proses selesai. Langkah selanjutnya adalah proses klasifikasi dengan cara klik *button* “Naïve Bayes”, hasil dari proses klasifikasi berupa akurasi klasifikasi.

4.1.4 Hasil Pengujian

Uji coba pada penelitian ini dilakukan terhadap lima *dataset* yaitu *dataset* CM1, JM1, KC1, KC2 dan PC1. Tahap pertama dalam pengujian ini adalah menentukan urutan fitur yang relevan dari setiap *dataset* berdasarkan seleksi fitur *Gain Ratio* dan tahap kedua adalah melakukan pengujian akurasi klasifikasi Naïve Bayes pada setiap *dataset* dengan seleksi fitur *Gain Ratio*, teknik klasifikasi yang digunakan adalah *10x10 cross-validation* atau biasa disebut dengan *10-fold cross-validation*.

4.1.2.1 Seleksi Fitur dengan *Gain Ratio*

Fitur yang relevan didapat dengan melakukan seleksi fitur menggunakan metode seleksi fitur *Gain Ratio*. Hasil dari seleksi fitur ini berupa urutan atau perangkingan fitur sesuai dengan nilai *Gain Ratio*. Tabel 4.8 hingga tabel 4.12 menunjukkan urutan atau perankingan fitur berdasarkan nilai *Gain Ratio* dari kelima *dataset* yang digunakan.

Tabel 4.8 Perankingan Fitur *dataset* CM1 Berdasarkan nilai *Gain Ratio*

GR	Fitur
0.17764027	t
0.17764027	e
0.08940102	IOComment
0.07676358	uniq_Op
0.066612095	uniq_Opnd
0.06614874	b
0.04657121	v
0.03866517	ev(g)
0.034703486	i
0.033941716	branchCount
0.033941716	iv(g)
0.033941716	v(g)
0.03385608	total_Opnd
0.03385608	loc
0.032714628	n
0.028663296	total_Op
0.022667497	d
0.017010132	IOBlank
0.001916128	IOCode
6.1650225E-4	l
0.0	locCodeAndComment

Berdasarkan pada tabel 4.8 dapat diketahui bahwa urutan fitur berdasarkan seleksi fitur *Gain Ratio* dari keseluruhan fitur pada *dataset* CM1 adalah fitur *Time to write program* (t), *Effort to write program* (e), *Count of lines of comments* (IOComment), *Unique operators* (uniq_Op), *Unique operands* (uniq_Opnd) dan seterusnya hingga urutan paling terakhir adalah fitur *Count of Code and Comments Lines* (locCodeAndComment).

Tabel 4.9 Perankingan Fitur *dataset JM1* Berdasarkan nilai *Gain Ratio*

GR	Fitur
0.19059384	b
0.1876871	IOCode
0.18379815	loc
0.1820183	uniq_Opnd
0.17562628	uniq_Op
0.15953024	locCodeAndComment
0.14623563	v(g)
0.14276169	branchCount
0.13281153	t
0.13281153	e
0.12094725	iv(g)
0.0979709	total_Opnd
0.09328443	v
0.09065093	total_Op
0.08561899	n
0.07875244	IOComment
0.06721482	ev(g)
0.060341917	i
0.05631962	IOBlank
0.031671084	d
0.019286584	l

Tabel 4.9 menunjukkan urutan fitur berdasarkan nilai *Gain Ratio* hasil proses seleksi fitur *Gain Ratio* pada *dataset JM1* adalah fitur *Error estimate* (b), *Count of Statement Lines* (IOCode), *Line of code* (loc), *Unique operands* (uniq_Opnd), *Unique operators* (uniq_Op), *Count of Code and Comments Lines* (locCodeAndComment), dan seterusnya hingga urutan paling akhir adalah fitur *Program length* (l).

Tabel 4.10 Perankingan Fitur *dataset* KC1 Berdasarkan nilai *Gain Ratio*

GR	Fitur
0.29109028	IOComment
0.1402622	b
0.121450596	IOBlank
0.119554184	uniq_Op
0.10195719	total_Op
0.09851719	total_Opnd
0.09668424	uniq_Opnd
0.09301534	v
0.090276174	loc
0.089849725	n
0.08840168	d
0.081891894	t
0.081891894	e
0.07088878	IOCode
0.0708482	i
0.06509653	ev(g)
0.05176313	l
0.045877665	branchCount
0.045877665	v(g)
0.031937465	iv(g)
8.2550956E-5	locCodeAndComment

Berdasarkan tabel 4.10 dapat diketahui bahwa urutan fitur berdasarkan seleksi fitur *Gain Ratio* pada *dataset* KC1 adalah fitur *Count of lines of comments* (IOComment), *Error estimate* (b), *Count of blank lines* (IOBlank), *Unique operators* (uniq_Op) dan seterusnya hingga urutan paling akhir adalah fitur *Total operator* (total_Op).

Tabel 4.11 Perankingan Fitur *dataset* KC2 Berdasarkan nilai *Gain Ratio*

GR	Fitur
0.29443324	IOCodeAndComment
0.24263638	IOBlank
0.23497549	uniq_Opnd
0.23497549	t
0.23497549	loc
0.23497547	total_Opnd
0.23497547	IOCode
0.23497547	e
0.23497547	n
0.21945715	total_Op
0.21945715	b
0.21945715	b
0.21890852	branchCount
0.21890852	iv(g)
0.21890852	ev(g)
0.21890852	v(g)
0.20772645	d
0.19890262	i
0.1527788	uniq_Op
0.13352568	IOComment
0.039263908	l

Urutan fitur yang relevan pada *dataset* KC2 berdasarkan seleksi fitur *Gain Ratio* ditunjukkan pada tabel 4.11 yaitu fitur *Count of Code and Comments Lines* (IOCodeAndComment), *Count of blank lines* (IOBlank), *Unique operands* (Uniq_Opnd), *Time to write program* (t), *Line of code* (loc) dan seterusnya hingga urutan paling akhir adalah fitur *Program length* (l).

Tabel 4.12 Perankingan Fitur *dataset* PC1 Berdasarkan nilai *Gain Ratio*

GR	Fitur
0.36620554	uniq_Opnd
0.23978049	IOCode
0.23978047	b
0.23978047	loc
0.22594728	t
0.22594725	e
0.17856956	IOComment
0.16976582	uniq_Op
0.16976582	iv(G)
0.12490867	IOBlank
0.09281469	ev(g)
0.08957554	branchCount
0.08957554	v(g)
0.080116116	v
0.069097996	i
0.06795057	total_Op
0.048390288	total_Opnd
0.046210535	n
0.024065675	locCodeAndComment
0.009840329	l
0.009806872	d

Pada *dataset* PC1 urutan fitur yang relevan berdasarkan seleksi fitur *Gain Ratio* ditunjukkan pada tabel 4.12, urutan fitur tersebut adalah fitur *Unique operands* (uniq_Opnd), *Count of Statement Lines* (IOCode), *Error estimate* (b), *Line of code* (loc), *Time to write program* (t) dan seterusnya hingga urutan terakhir adalah fitur *Difficulty* (d).

4.1.2.2 Uji coba Klasifikasi

Pada bagian ini dilakukan pengujian klasifikasi prediksi cacat perangkat lunak dengan fitur hasil seleksi fitur *Gain Ratio* pada setiap *dataset*. Hasil dari proses pengujian ini berupa akurasi hasil klasifikasi dengan algoritma klasifikasi Naïve Bayes dengan teknik *10-fold cross-validation*. Proses uji coba klasifikasi akan dilakukan dengan beberapa jumlah fitur yang berbeda untuk mendapatkan jumlah fitur yang paling relevan, pada penelitian ini penulis melakukan uji coba dengan jumlah fitur sebanyak 1 fitur, 2 fitur, 3 fitur, dan seterusnya sampai keseluruhan fitur.

Hasil dari proses uji coba klasifikasi adalah akurasi klasifikasi yang diperoleh dari perhitungan *confusion matrix*, sebagai gambaran perhitungan akurasi klasifikasi penulis melampirkan perhitungan akurasi klasifikasi *dataset* CM1 ditunjukkan pada bagian lampiran I. Perlu diketahui bahwa nilai yang ditunjukkan pada table *confusion matrix* merupakan nilai dari hasil perhitungan kecocokan antara hasil prediksi terhadap kenyataan atau data.

Hasil dari proses uji coba klasifikasi ditunjukkan mulai tabel 4.13 hingga tabel 4.17, tabel-tabel tersebut menunjukkan hasil dari proses uji coba kelima dataset yaitu dataset CM1, JM1, KC1, KC2 dan PC1.

Tabel 4.13 Hasil Uji Coba Klasifikasi *Dataset* CM1

Jumlah fitur	Akurasi klasifikasi NB(%)
1	89.96%
2	89.56%
3	89.16%
4	87.95%
5	87.55%
6	87.15%
7	86.95%
8	86.55%
9	86.35%
10	85.74%
11	85.94%
12	86.14%
13	85.74%
14	85.74%
15	85.74%
16	85.74%
17	85.54%
18	85.74%
19	85.54%
20	85.34%
21	85.34%

Berdasarkan pada tabel 4.13 dapat diketahui bahwa hasil kasifikasi prediksi cacat perangkat lunak pada *dataset* CM1 dengan fitur hasil seleksi fitur *Gain Ratio* didapatkan akurasi paling baik dengan jumlah fitur sebanyak 1 dengan akurasi klasifikasi Nave Bayes sebesar 89.96%.

Tabel 4.14 Hasil Uji Coba Klasifikasi *Dataset JM1*

Jumlah fitur	Akurasi klasifikasi NB(%)
1	80.54%
2	80.28%
3	80.59%
4	80.45%
5	80.59%
6	80.51%
7	80.52%
8	80.55%
9	80.54%
10	80.63%
11	80.59%
12	80.61%
13	80.57%
14	80.53%
15	80.51%
16	80.46%
17	80.43%
18	80.40%
19	80.47%
20	80.45%
21	80.42%

Berdasarkan pada tabel 4.14 dapat diketahui bahwa hasil kasifikasi prediksi cacat perangkat lunak pada *dataset JM1* dengan fitur hasil seleksi fitur *Gain Ratio* didapatkan akurasi klasifikasi Naïve Bayes paling baik sebesar 80.61% dengan jumlah fitur sebanyak 12.

Tabel 4.15 Hasil Uji Coba Klasifikasi *Dataset* KC1

Jumlah fitur	Akurasi klasifikasi NB(%)
1	84.02%
2	84.35%
3	84.59%
4	83.97%
5	83.50%
6	83.45%
7	83.21%
8	83.21%
9	82.98%
10	83.03%
11	82.93%
12	83.03%
13	83.03%
14	83.07%
15	82.74%
16	82.60%
17	82.36%
18	82.36%
19	82.36%
20	82.31%
21	82.36%

Berdasarkan pada tabel 4.15 dapat diketahui bahwa hasil kasifikasi prediksi cacat perangkat lunak pada *dataset* KC1 dengan fitur hasil seleksi fitur *Gain Ratio* didapatkan akurasi klasifikasi Naïve Bayes paling baik sebesar 84.59% dengan jumlah fitur sebanyak 3.

Tabel 4.16 Hasil Uji Coba Klasifikasi *Dataset* KC2

Jumlah fitur	Akurasi klasifikasi NB(%)
1	80.65%
2	82.18%
3	83.52%
4	83.91%
5	83.52%
6	83.91%
7	83.72%
8	83.72%
9	83.72%
10	83.91%
11	83.72%
12	83.72%
13	83.52%
14	83.33%
15	83.72%
16	83.91%
17	83.72%
18	83.52%
19	83.52%
20	83.52%
21	83.52%

Tabel 4.16 menunjukkan hasil kasifikasi prediksi cacat perangkat lunak *dataset* KC2 dengan fitur hasil seleksi fitur *Gain Ratio*, akurasi klasifikasi Naïve Bayes paling baik sebesar 83.91% dengan jumlah fitur sebanyak 4, 6, 10, dan 16, namun pada percobaan ini terdapat beberapa fitur dengan hasil akurasi yang sama nantinya akan dipilih jumlah fitur yang paling kecil yaitu jumlah fitur 4 karena dengan jumlah fitur yang sedikit bisa menghasilkan akurasi yang sama besarnya.

Tabel 4.17 Hasil Uji Coba Klasifikasi *Dataset* PC1

Jumlah fitur	Akurasi klasifikasi NB(%)
1	91.97%
2	91.25%
3	91.07%
4	90.26%
5	90.26%
6	90.98%
7	90.44%
8	90.26%
9	89.54%
10	89.27%
11	89.45%
12	89.27%
13	89.36%
14	89.45%
15	89.36%
16	89.00%
17	89.18%
18	89.27%
19	89.18%
20	89.27%
21	89.18%

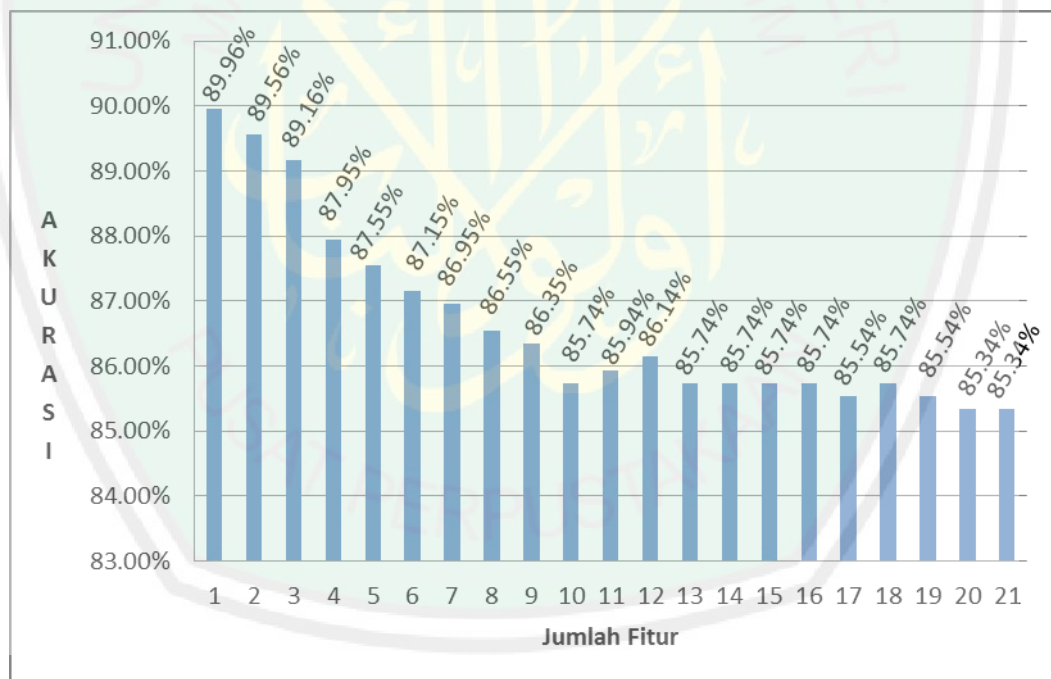
Berdasarkan pada tabel 4.17 dapat diketahui bahwa hasil kasifikasi prediksi cacat perangkat lunak pada *dataset* PC1 dengan fitur hasil seleksi fitur *Gain Ratio* didapatkan akurasi klasifikasi Naïve Bayes paling baik sebesar 91.97% dengan jumlah fitur sebanyak 1.

4.2 Pembahasan

Pada bagian ini akan dibahas mengenai hasil dari pengujian yang telah dilakukan, pembahasan dilakukan terhadap semua hasil pengujian yang meliputi pengujian pada *dataset* CM1, JM1, KC1, KC2 dan PC1.

4.2.1 Pembahasan hasil pengujian *dataset* CM1

Sesuai hasil pengujian yang telah dilakukan pada *dataset* CM1 yang ditunjukkan pada tabel 4.13 maka gambar 4.2 menunjukkan grafik nilai akurasi hasil dari pengujian *dataset* CM1 dengan jumlah fitur sebanyak 1 fitur, 2 fitur, 3 fitur, dan seterusnya sampai keseluruhan fitur.

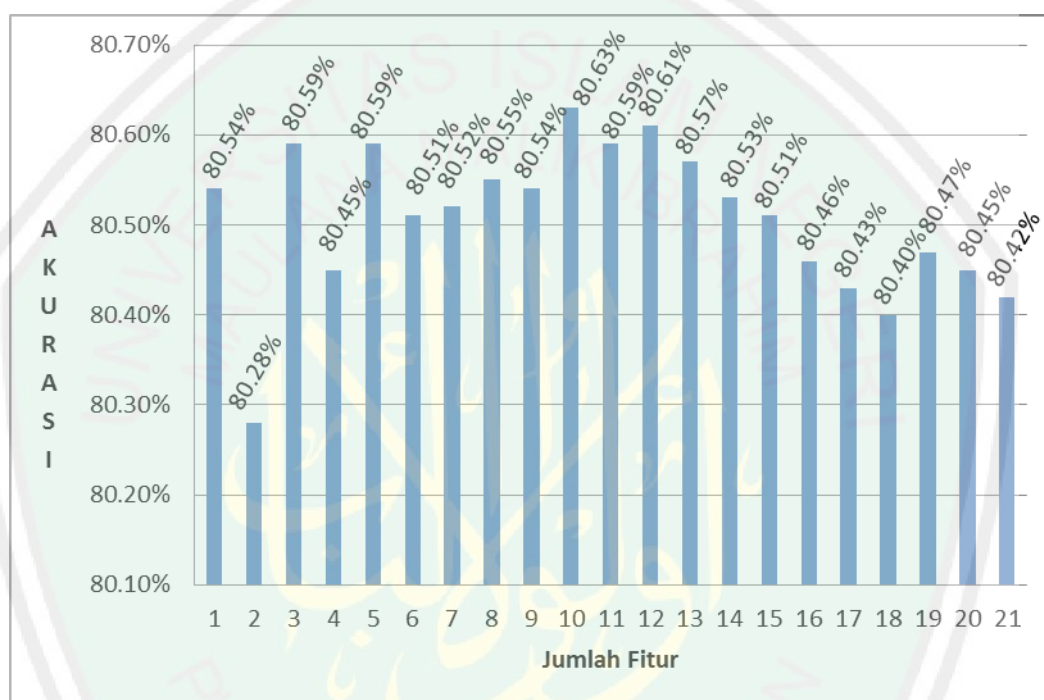


Gambar 4.2 Grafik akurasi klasifikasi NB *dataset* CM1

Pada grafik 4.2 terlihat bahwa akurasi paling baik ditunjukkan pada jumlah fitur sebanyak 1 fitur yaitu sebesar 89.96%, sehingga fitur yang memiliki pengaruh besar terhadap prediksi cacat perangkat lunak untuk *dataset* CM1 berdasarkan seleksi fitur *Gain Ratio* berjumlah 1 fitur yaitu fitur t.

4.2.2 Pembahasan hasil pengujian *dataset* JM1

Hasil pengujian yang telah dilakukan pada *dataset* JM1 yang ditunjukkan pada tabel 4.14 maka gambar 4.3 menunjukkan grafik nilai akurasi hasil dari pengujian *dataset* JM1 dengan jumlah fitur sebanyak 1 fitur, 2 fitur, 3 fitur, dan seterusnya sampai keseluruhan fitur.



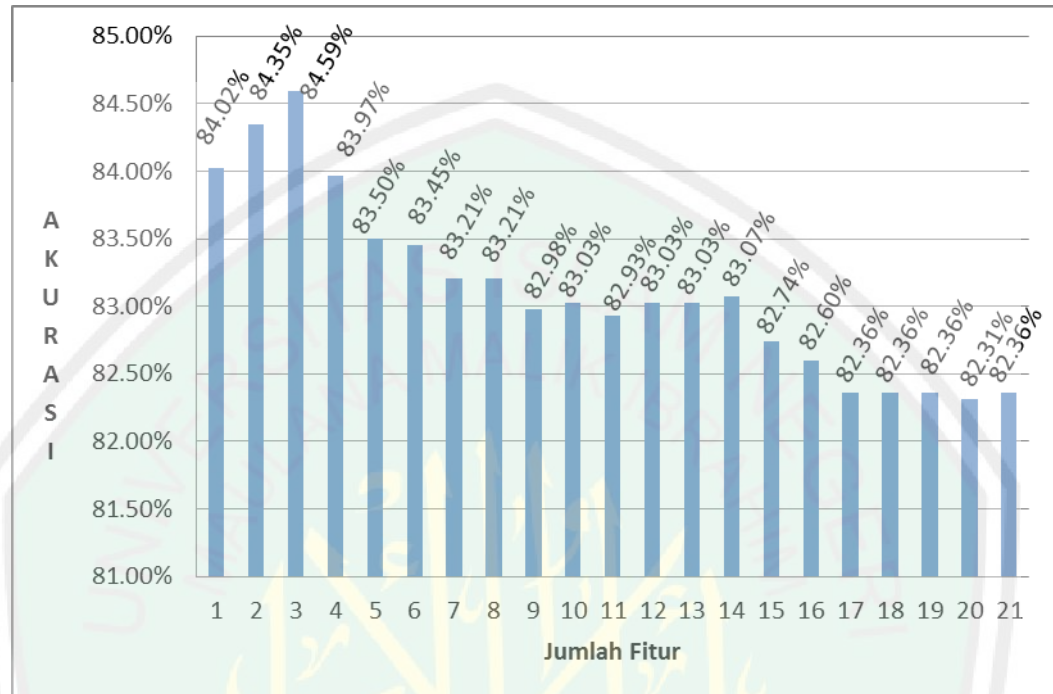
Gambar 4.3 Grafik akurasi klasifikasi NB *dataset* JM1

Pada grafik 4.3 terlihat bahwa akurasi paling baik ditunjukkan pada jumlah fitur sebanyak 10 fitur yaitu sebesar 80.63%, sehingga fitur yang memiliki pengaruh besar terhadap prediksi cacat perangkat lunak untuk *dataset* JM1 berdasarkan seleksi fitur *Gain Ratio* berjumlah 10 fitur yaitu fitur b, lOCode, loc, uniq_Opnd, uniq_Op, locCodeAndComment, v(g), branchCount, t dan e.

4.2.3 Pembahasan hasil pengujian *dataset* KC1

Sesuai hasil pengujian yang telah dilakukan pada *dataset* KC1 yang ditunjukkan pada tabel 4.15 maka gambar 4.4 menunjukkan grafik nilai akurasi

hasil dari pengujian *dataset* KC1 dengan jumlah fitur sebanyak 1 fitur, 2 fitur, 3 fitur, dan seterusnya sampai keseluruhan fitur.

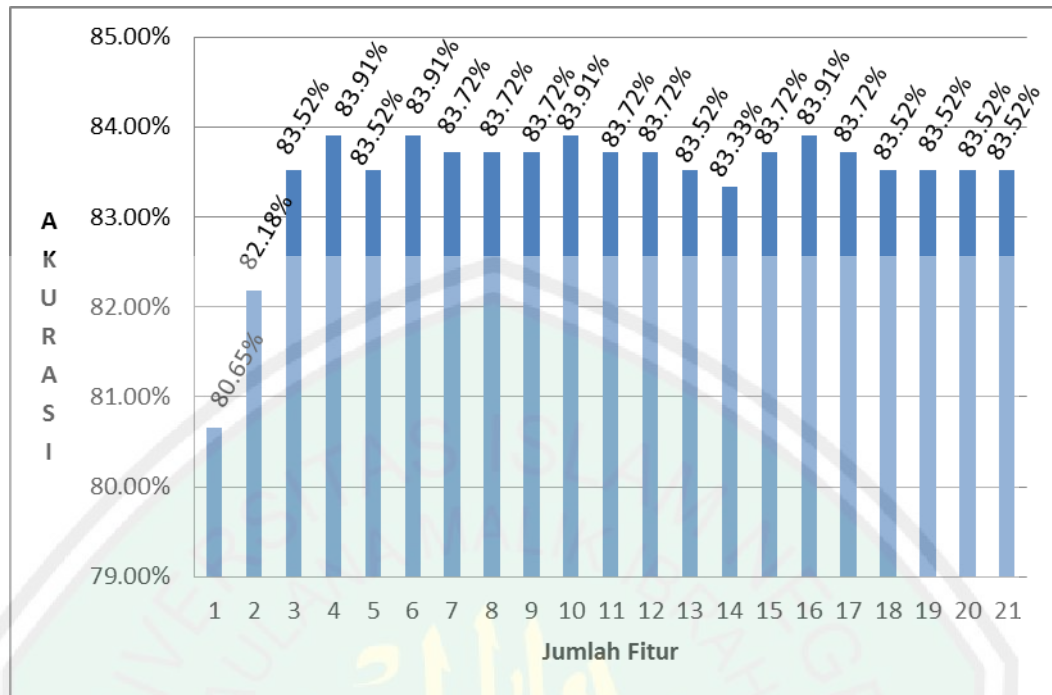


Gambar 4.4 Grafik akurasi klasifikasi NB *dataset* KC1

Pada grafik 4.4 terlihat bahwa akurasi paling baik ditunjukkan pada jumlah fitur sebanyak 3 fitur yaitu sebesar 84.59%, sehingga fitur yang memiliki pengaruh besar terhadap prediksi cacat perangkat lunak untuk *dataset* KC1 berdasarkan seleksi fitur *Gain Ratio* berjumlah 3 fitur yaitu fitur *IOComment*, *b* dan *IOBlank*.

4.2.4 Pembahasan hasil pengujian *dataset* KC2

Sesuai hasil pengujian yang telah dilakukan pada *dataset* KC2 yang ditunjukkan pada tabel 4.16 maka gambar 4.5 menunjukkan grafik nilai akurasi hasil dari pengujian *dataset* KC2 dengan jumlah fitur sebanyak 1 fitur, 2 fitur, 3 fitur, dan seterusnya sampai keseluruhan fitur.

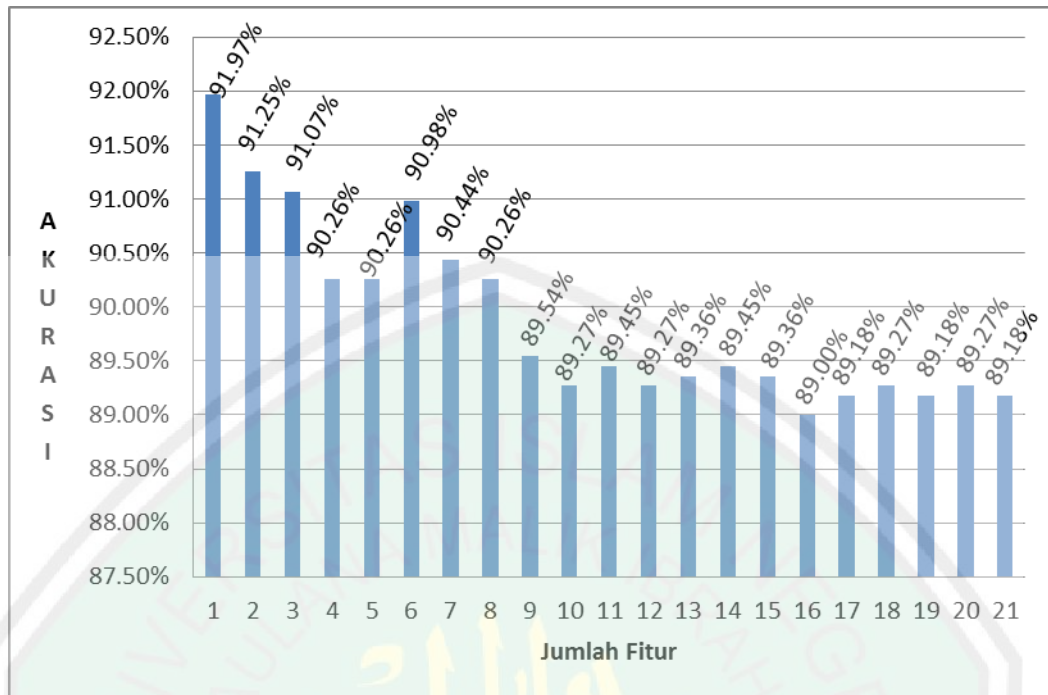


Gambar 4.5 Grafik akurasi klasifikasi NB *dataset* KC2

Pada grafik 4.5 terlihat bahwa akurasi paling baik ditunjukkan pada jumlah fitur sebanyak 4 fitur yaitu sebesar 83.91%, sehingga fitur yang memiliki pengaruh besar terhadap prediksi cacat perangkat lunak untuk *dataset* KC2 berdasarkan seleksi fitur *Gain Ratio* berjumlah 4 fitur yaitu fitur *IOCodeAndComment*, *IOBlank*, *uniq_Opnd* dan *t*.

4.2.5 Pembahasan hasil pengujian *dataset* PC1

Sesuai hasil pengujian yang telah dilakukan pada *dataset* PC1 yang ditunjukkan pada tabel 4.17 maka gambar 4.6 menunjukkan grafik nilai akurasi hasil dari pengujian *dataset* PC1 dengan jumlah fitur sebanyak 1 fitur, 2 fitur, 3 fitur, dan seterusnya sampai keseluruhan fitur.



Gambar 4.6 Grafik akurasi klasifikasi NB *dataset* PC1

Pada grafik 4.6 terlihat bahwa akurasi paling baik ditunjukkan pada jumlah fitur sebanyak 1 fitur yaitu sebesar 91.97%, sehingga fitur yang memiliki pengaruh besar terhadap prediksi cacat perangkat lunak untuk *dataset* PC1 berdasarkan seleksi fitur *Gain Ratio* berjumlah 1 fitur yaitu fitur *uniq_Opnd*.

Rangkuman dari keseluruhan pembahasan hasil pengujian yang telah dilakukan terhadap lima *dataset* yaitu *dataset* CM1, JM1, KC1, KC2 dan PC1 dengan jumlah fitur yang diambil sebanyak 1 fitur, 2 fitur, 3 fitur, dan seterusnya sampai keseluruhan fitur maka tabel 4.18 menunjukkan fitur yang relevan terhadap prediksi cacat perangkat lunak berdasarkan seleksi fitur *Gain Ratio* yang dibuktikan dengan hasil dari pengujian klasifikasi menunjukkan tingkat akurasi paling besar.

Tabel 4.18 Fitur yang relevan berdasarkan seleksi fitur *Gain Ratio*

<i>Dataset</i>	Fitur	Akurasi NB
CM1	<ul style="list-style-type: none"> • t 	89.96%
JM1	<ul style="list-style-type: none"> • b • IOCode • Loc • uniq_Opnd • uniq_Op • locCodeAndComment • v(g) • branchCount • t • e 	80.63%
KC1	<ul style="list-style-type: none"> • IOComment • b • IOBlank 	84.59%
KC2	<ul style="list-style-type: none"> • IOCodeAndComment • IOBlank • uniq_Opnd • t 	83.91%
PC1	<ul style="list-style-type: none"> • uniq_Opnd 	91.97%

Berdasarkan tabel 4.18 maka fitur yang relevan terhadap prediksi cacat perangkat lunak untuk *dataset* CM1 adalah fitur *Time to write program* (t) dengan akurasi sebesar 89.96%. *Dataset* JM1 adalah fitur *Error estimate* (b), *Count of Statement Lines* (IOCode), *Line of code* (loc), *Unique operands* (uniq_Opnd), *Unique operator* (uniq_Op), *Count of Code and Comments Lines* (IOCodeAndComment), *Cyclomatic complexity* (v(g)), *Branch count* (branchCount), *Time to write program* (t) dan *Effort to write program* (e) dengan akurasi sebesar 80.63%. *Dataset* KC1 adalah fitur *Count of lines of comments* (IOComment), *Error estimate* (b) dan *Count of blank lines* (IOBlank) dengan akurasi sebesar 84.59%. *Dataset* KC2 adalah fitur *Count of Code and Comments*

Lines (IOCodeAndComment), *Count of blank lines* (IOBlank), *Unique operands* (uniq_Opnd) dan *Time to write program* (t) dengan akurasi sebesar 83.91%. *Dataset* PC1 adalah fitur *Unique operands* (uniq_Opnd) dengan akurasi sebesar 91.97%.

4.3 Integrasi dengan Islam

Tolong-menolong adalah sebuah sikap yang harus dimiliki oleh seorang manusia karena sudah kodratnya bahwa manusia adalah makhluk sosial yang hidupnya membutuhkan manusia lain. Sebagaimana dalam islam diperintahkan untuk saling tolong-menolong dalam hal kebaikan, penjelasan tersebut ada pada potongan Ayat Alqur'an surat al-ma'idah ayat 2 berikut (Al-Sheikh, 2011:9);

يَا أَيُّهَا الَّذِينَ آمَنُوا لَا تَحْلُوا شَعَائِرَ اللَّهِ وَلَا الشُّهُرَ الْحَرَامَ وَلَا الْهَدْيَ وَلَا الْقَلَائِدَ وَلَا آمِينَ الْبَيْتِ الْحَرَامِ
يَبْتَغُونَ فَضْلًا مِنْ رَبِّهِمْ وَرِضْوَانًا وَإِذَا حَلَلْتُمْ فَاصْطَادُوا وَلَا يَجْرِمَنَّكُمْ شَنَانُ قَوْمٍ أَنْ صَدُّوكُمْ عَنِ الْمَسْجِدِ
الْحَرَامِ أَنْ تَعْتَدُوا وَتَعَاوَنُوا عَلَى الْبِرِّ وَالتَّقْوَىٰ وَلَا تَعَاوَنُوا عَلَى الْإِثْمِ وَالْعُدْوَانِ وَاتَّقُوا اللَّهَ إِنَّ اللَّهَ شَدِيدُ
الْعِقَابِ

Artinya : “*Hai orang-orang yang beriman, janganlah kamu melanggar syi'ar-syi'ar Allah, dan jangan melanggar kehormatan bulan-bulan haram, jangan (mengganggu) binatang-binatang had-ya, dan binatang-binatang qalaa-id, dan jangan (pula) mengganggu orang-orang yang mengunjungi Baitullah sedang mereka mencari kurnia dan keridhaan dari Tuhannya dan apabila kamu telah menyelesaikan ibadah haji, maka bolehlah berburu. Dan janganlah sekali-kali kebencian(mu) kepada sesuatu kaum karena mereka menghalang-halangi kamu dari Masjidilharam, mendorongmu berbuat aniaya (kepada mereka). Dan tolong-menolonglah kamu dalam (mengerjakan) kebajikan dan takwa, dan jangan tolong-menolong dalam berbuat dosa dan pelanggaran. Dan bertakwalah kamu kepada Allah, sesungguhnya Allah amat berat siksa-Nya.*” (al-ma'idah:2).

Pada potongan ayat tersebut Allah SWT memerintahkan untuk saling tolong-menolong dalam aktivitas kebaikan dan dilarang untuk tolong-menolong dalam perbuatan yang salah dan mengakibatkan dosa, potongan ayat tersebut dalam Tafsir Ibnu Katsir dijelaskan Allah SWT memerintahkan kepada hamba-hamba-Nya yang beriman untuk saling menolong dalam berbuat kebaikan yaitu kebajikan dan meninggalkan hal-hal yang mungkar; hal ini dinamakan ketakwaan. Allah SWT melarang mereka bantu- membantu dalam kebatilan serta tolong-menolong dalam perbuatan dosa dan hal-hal yang diharamkan. Ibnu Jarir mengatakan bahwa dosa itu ialah meninggalkan apa yang diperintahkan oleh Allah untuk dikerjakan. Pelanggaran itu artinya melampaui apa yang digariskan oleh Allah dalam agama kalian, serta melupakan apa yang difardukan oleh Allah atas diri kalian dan atas diri orang lain. Penjelasan tersebut kemudian diperkuat dengan hadits yang diriwayatkan oleh Imam Bukhari secara munfarid melalui hadis Hasyim dengan sanad yang sama dan lafaz yang semisal. Keduanya menyetengahkan hadis ini melalui jalur Sabit, dari Anas yang menceritakan bahwa Rasulullah Saw. telah bersabda:

انصُرْ أَخَاكَ ظَالِمًا أَوْ مَظْلُومًا". قِيلَ: يَا رَسُولَ اللَّهِ، هَذَا نَصْرُهُ مَظْلُومًا، فَكَيْفَ أَنْصُرُهُ ظَالِمًا؟ قَالَ:
"تَمْنَعُهُ مِنَ الظُّلْمِ، فَذَلِكَ نَصْرُكَ إِيَّاهُ"

Artinya : *"Tolonglah saudaramu, baik dia berbuat aniaya ataupun dianiaya." Ditanyakan, "Wahai Rasulullah, orang ini dapat aku tolong bila dalam keadaan teraniaya, tetapi bagaimana menolongnya jika dia berbuat aniaya?" Rasulullah Saw. menjawab, "Kamu cegah dia dari perbuatan aniaya, itulah cara kamu menolongnya."*

Menurut kajian yang penulis lakukan pada Alqur'an potongan surat al-ma'idah:2 yang artinya *"Dan tolong-menolonglah kamu dalam (mengerjakan)*

kebajikan dan takwa, dan jangan tolong-menolong dalam berbuat dosa dan pelanggaran” serta telah dijelaskan di atas dengan rujukan tafsir Ibnu Katsir yang menjelaskan bahwasannya Allah SWT memerintahkan kepada hamba-hamba-Nya yang beriman untuk saling menolong dalam berbuat kebaikan yaitu kebajikan dan meninggalkan hal-hal yang mungkar, hal ini dinamakan ketakwaan.

Berdasarkan ayat Alqur'an surah al-ma'idah ayat 2 dan hadist yang memperkuatnya memerintahkan untuk berbuat saling tolong-menolong atau saling membantu sesama manusia dalam hal kebaikan, maka pada penelitian ini penulis bertujuan untuk membantu menentukan fitur yang relevan terhadap prediksi cacat perangkat lunak karena diketahui bahwa fitur-fitur yang terdapat pada *dataset* NASA MDP tidak semuanya relevan atau memiliki pengaruh besar terhadap prediksi cacat perangkat lunak. Prediksi cacat perangkat lunak dengan fitur yang relevan akan menghasilkan sebuah prediksi cacat perangkat lunak yang lebih akurat yang dapat meningkatkan kualitas perangkat lunak.

BAB V

PENUTUP

Pada bab penutup ini menjelaskan tentang kesimpulan dari penelitian ini serta memberi saran bagi pembaca untuk pengembangan penelitian.

5.1 Kesimpulan

Berdasarkan dari hasil penelitian yang telah dilakukan, maka dapat disimpulkan bahwa fitur yang relevan berdasarkan seleksi fitur *Gain Ratio* untuk setiap *dataset* yaitu *dataset* CM1 adalah fitur *Time to write program* (t) dengan akurasi sebesar 89.96%. *Dataset* JM1 adalah fitur *Error estimate* (b), *Count of Statement Lines* (lOCode), *Line of code* (loc), *Unique operands* (uniq_Opnd), *Unique operator* (uniq_Op), *Count of Code and Comments Lines* (lOCodeAndComment), *Cyclomatic complexity* (v(g)), *Branch count* (branchCount), *Time to write program* (t) dan *Effort to write program* (e) dengan akurasi sebesar 80.63%. *Dataset* KC1 adalah fitur *Count of lines of comments* (lOComment), *Error estimate* (b) dan *Count of blank lines* (lOBlank) dengan akurasi sebesar 84.59%. *Dataset* KC2 adalah fitur *Count of Code and Comments Lines* (lOCodeAndComment), *Count of blank lines* (lOBlank), *Unique operands* (uniq_Opnd) dan *Time to write program* (t) dengan akurasi sebesar 83.91%. *Dataset* PC1 adalah fitur *Unique operands* (uniq_Opnd) dengan akurasi sebesar 91.97%. Fitur-fitur tersebut dikatakan sebagai fitur yang relevan untuk prediksi cacat perangkat lunak karena berdasarkan uji coba klasifikasi yang menghasilkan akurasi terbaik dari uji coba yang telah dilakukan.

5.2 Saran

Peneliti menyadari bahwa dalam penelitian ini masih belum sempurna, dengan demikian perlu adanya pengembangan untuk mendapatkan hasil yang lebih baik, beberapa saran dari peneliti antara lain:

- A. Melakukan seleksi fitur dengan metode seleksi fitur yang lain seperti Correlation-based Feature Selection (CFS), Symmetrical Uncertainty, Chi-square dan Relief-F dengan tujuan mendapatkan fitur yang mungkin lebih relevan terhadap prediksi cacat perangkat lunak.
- B. Melakukan uji coba dengan *dataset* yang berbeda.
- C. Melakukan pembuktian dengan algoritma klasifikasi yang lain seperti C4.5, Random Forest, Support Vector Machine (SVM) atau algoritma klasifikasi yang lain dengan tujuan untuk menguji fitur yang didapat pada seleksi fitur *Gain Ratio* ini dan melakukan analisa terhadap hasil akurasi klasifikasi.

DAFTAR PUSTAKA

- Akbar, M. S. 2017. *PREDIKSI CACAT PERANGKAT LUNAK DENGAN OPTIMASI NAIVE BAYES MENGGUNAKAN GAIN RATIO*. Surabaya : ITS.
- Al-Sheikh, Abdullah. 2011. Luubaabut Tafsir Min Ibni Katsiir, Kairo:Mu-assasah Daar al-Hilaal. 1994M, Terj. M. Abdul Ghoftar. Tafsir Ibnu Katsir jilid 3. Bogor:Pustaka Imam Asy-Syafi'i.
- Arar, Ö.F., Ayan, K. 2017. *A Feature Dependent Naive Bayes Approach and Its Application to the Software Defect Prediction Problem*. Applied Soft Computing Journal.
- Bratu, C.V., Muresan, T., Potolea, R. 2008. *Improving Classification Accuracy through Feature Selection*. In : Proceeding of the 4th IEEE Internaional Conference on Intelligent Computer Communication and Processing, pp. 25-32. IEEE Press, New York.
- Dougherty, J., Kohavi, R., Sahami, M. 1995. *Supervised and unsupervised discretization of continuous features*. In Proceedings of the 12th International Conference on Machine Learning, pages 194–202.
- Essra , A., Rahmadani , & Safriadi . (2016, Desember). ANALISIS INFORMATION GAIN ATTRIBUTE EVALUATION UNTUK KLASIFIKASI SERANGAN INTRUSI. *Jurnal ISD (Information System Development)*.
- Karabulut, Esra M., *et.al.* 2012. *A comparative study on the effect of feature selection on classification accuracy*. Procedia Technology 1, 323-327.
- Menzies, T., Sayyad Shirabad, J. 2005. "PROMISE Software Engineering Repository". <http://promise.site.uottawa.ca/SERpository/>. Diakses pada 8 Maret 2018.
- Menzies, T., Greenwald, J., & Frank, A. 2007. Data Mining Static Code Attributes to Learn Defect Predictors. IEEE Transactions on Software Engineering.
- Pelayo, L., Dick, S., 2007. Applying Novel Resampling Strategies To Software Defect Prediction. NAFIPS 2007 - 2007 Annual Meeting of the North American Fuzzy Information Processing Society 69–72.
- Prasetyo, Eko. 2012. DATA MINING – Konsep dan Aplikasi Menggunakan MATLAB. Yogyakarta : ANDI.
- Runeson , P., Andersson , C., Thelin , T., Andrews , A., & Berling , T. (2006). What Do We Know about Defect Detection Methods? *IEEE SOFTWARE*.

- Saifudin, A. 2014. *Pendekatan Level Data dan Algoritma untuk Penanganan Ketidakseimbangan Kelas pada Prediksi Cacat Software Berbasis Naive Bayes*. M.Kom Thesis, Program Pascasarjana Magister Teknik Informatika STMIK Eresha.
- Singh, P., Verma, S. 2009. *An Investigation of the Effect of Discretization on Defect Prediction Using Static Measures*. in: 2009 International Conference on Advances in Computing, Control, and Telecommunication Technologies. IEEE, pp. 837–839.
- Suyanto. 2017. *DATA MINING UNTUK KLASIFIKASI DAN KLASTERISASI DATA*. Bandung : Penerbit Informatika.
- Trabelsi, M. 2017. *A New Feature Selection Method for Nominal Classifier based on Formal Concept Analysis*. Procedia Computer Science, 186–194.
- Wahono, Romi S., S. Nanna S. 2014. *Genetic Feature Selection for Software Defect Prediction*. American Scientific Publiser, 239-244.
- Wang, Huajing, Taghi M. K., Amri Napolitano. 2012. *Software measurement data reduction using ensemble techniques*. Neurocomputing, 124-132.
- Yang Y., Webb G.I., Wu X. 2005. *Discretization Methods*. In: Maimon O., Rokach L. (eds) *Data Mining and Knowledge Discovery Handbook*. Springer, Boston, MA.



LAMPIRAN-LAMPIRAN

LAMPIRAN I

Perhitungan *confusion matrix* dan akurasi klasifikasi

Tabel 1 Hasil Prediksi dataset CM1 dengan jumlah fitur sebanyak 5 fitur

No	Hasil Prediksi	Kenyataan			
1	FALSE	FALSE	37	FALSE	FALSE
2	FALSE	FALSE	38	FALSE	FALSE
3	TRUE	FALSE	39	FALSE	FALSE
4	FALSE	FALSE	40	FALSE	FALSE
5	FALSE	FALSE	41	FALSE	FALSE
6	TRUE	FALSE	42	FALSE	FALSE
7	FALSE	FALSE	43	FALSE	FALSE
8	FALSE	FALSE	44	FALSE	FALSE
9	FALSE	FALSE	45	FALSE	FALSE
10	FALSE	FALSE	46	FALSE	TRUE
11	TRUE	FALSE	47	FALSE	TRUE
12	FALSE	FALSE	48	FALSE	TRUE
13	FALSE	FALSE	49	FALSE	TRUE
14	FALSE	FALSE	50	FALSE	TRUE
15	FALSE	FALSE	51	FALSE	FALSE
16	FALSE	FALSE	52	FALSE	FALSE
17	FALSE	FALSE	53	FALSE	FALSE
18	FALSE	FALSE	54	FALSE	FALSE
19	FALSE	FALSE	55	FALSE	FALSE
20	FALSE	FALSE	56	FALSE	FALSE
21	FALSE	FALSE	57	FALSE	FALSE
22	FALSE	FALSE	58	FALSE	FALSE
23	FALSE	FALSE	59	FALSE	FALSE
24	TRUE	FALSE	60	FALSE	FALSE
25	FALSE	FALSE	61	FALSE	FALSE
26	FALSE	FALSE	62	FALSE	FALSE
27	FALSE	FALSE	63	FALSE	FALSE
28	FALSE	FALSE	64	FALSE	FALSE
29	FALSE	FALSE	65	FALSE	FALSE
30	FALSE	FALSE	66	FALSE	FALSE
31	FALSE	FALSE	67	FALSE	FALSE
32	FALSE	FALSE	68	FALSE	FALSE
33	TRUE	FALSE	69	FALSE	FALSE
34	FALSE	FALSE	70	FALSE	FALSE
35	FALSE	FALSE	71	FALSE	FALSE
36	TRUE	FALSE	72	FALSE	FALSE
			73	FALSE	FALSE
			74	FALSE	FALSE

75	FALSE	FALSE
76	FALSE	FALSE
77	FALSE	FALSE
78	FALSE	FALSE
79	FALSE	FALSE
80	TRUE	FALSE
81	FALSE	FALSE
82	FALSE	FALSE
83	FALSE	FALSE
84	FALSE	FALSE
85	FALSE	FALSE
86	FALSE	FALSE
87	FALSE	FALSE
88	FALSE	FALSE
89	FALSE	FALSE
90	FALSE	FALSE
91	FALSE	FALSE
92	FALSE	FALSE
93	FALSE	FALSE
94	FALSE	FALSE
95	FALSE	FALSE
96	FALSE	TRUE
97	TRUE	TRUE
98	FALSE	TRUE
99	FALSE	TRUE
100	FALSE	TRUE
101	FALSE	FALSE
102	FALSE	FALSE
103	FALSE	FALSE
104	FALSE	FALSE
105	FALSE	FALSE
106	FALSE	FALSE
107	FALSE	FALSE
108	FALSE	FALSE
109	FALSE	FALSE
110	FALSE	FALSE
111	FALSE	FALSE
112	FALSE	FALSE
113	FALSE	FALSE
114	FALSE	FALSE
115	FALSE	FALSE
116	FALSE	FALSE
117	FALSE	FALSE

118	FALSE	FALSE
119	FALSE	FALSE
120	FALSE	FALSE
121	FALSE	FALSE
122	FALSE	FALSE
123	FALSE	FALSE
124	FALSE	FALSE
125	FALSE	FALSE
126	FALSE	FALSE
127	FALSE	FALSE
128	FALSE	FALSE
129	FALSE	FALSE
130	FALSE	FALSE
131	FALSE	FALSE
132	FALSE	FALSE
133	FALSE	FALSE
134	FALSE	FALSE
135	FALSE	FALSE
136	FALSE	FALSE
137	TRUE	FALSE
138	FALSE	FALSE
139	FALSE	FALSE
140	FALSE	FALSE
141	FALSE	FALSE
142	FALSE	FALSE
143	FALSE	FALSE
144	FALSE	FALSE
145	FALSE	FALSE
146	FALSE	TRUE
147	TRUE	TRUE
148	TRUE	TRUE
149	TRUE	TRUE
150	TRUE	TRUE
151	FALSE	FALSE
152	FALSE	FALSE
153	FALSE	FALSE
154	FALSE	FALSE
155	FALSE	FALSE
156	FALSE	FALSE
157	FALSE	FALSE
158	FALSE	FALSE
159	FALSE	FALSE
160	FALSE	FALSE

161	FALSE	FALSE
162	FALSE	FALSE
163	FALSE	FALSE
164	FALSE	FALSE
165	FALSE	FALSE
166	FALSE	FALSE
167	FALSE	FALSE
168	FALSE	FALSE
169	FALSE	FALSE
170	FALSE	FALSE
171	FALSE	FALSE
172	FALSE	FALSE
173	FALSE	FALSE
174	FALSE	FALSE
175	FALSE	FALSE
176	TRUE	FALSE
177	FALSE	FALSE
178	FALSE	FALSE
179	TRUE	FALSE
180	FALSE	FALSE
181	FALSE	FALSE
182	FALSE	FALSE
183	FALSE	FALSE
184	FALSE	FALSE
185	FALSE	FALSE
186	FALSE	FALSE
187	FALSE	FALSE
188	FALSE	FALSE
189	FALSE	FALSE
190	FALSE	FALSE
191	FALSE	FALSE
192	FALSE	FALSE
193	FALSE	FALSE
194	FALSE	FALSE
195	TRUE	FALSE
196	FALSE	TRUE
197	FALSE	TRUE
198	FALSE	TRUE
199	FALSE	TRUE
200	FALSE	TRUE
201	FALSE	FALSE
202	FALSE	FALSE
203	FALSE	FALSE

204	FALSE	FALSE
205	FALSE	FALSE
206	FALSE	FALSE
207	FALSE	FALSE
208	FALSE	FALSE
209	FALSE	FALSE
210	FALSE	FALSE
211	FALSE	FALSE
212	TRUE	FALSE
213	FALSE	FALSE
214	FALSE	FALSE
215	FALSE	FALSE
216	FALSE	FALSE
217	FALSE	FALSE
218	FALSE	FALSE
219	FALSE	FALSE
220	FALSE	FALSE
221	FALSE	FALSE
222	FALSE	FALSE
223	FALSE	FALSE
224	FALSE	FALSE
225	FALSE	FALSE
226	FALSE	FALSE
227	FALSE	FALSE
228	FALSE	FALSE
229	FALSE	FALSE
230	FALSE	FALSE
231	TRUE	FALSE
232	FALSE	FALSE
233	FALSE	FALSE
234	FALSE	FALSE
235	FALSE	FALSE
236	FALSE	FALSE
237	FALSE	FALSE
238	FALSE	FALSE
239	FALSE	FALSE
240	FALSE	FALSE
241	FALSE	FALSE
242	FALSE	FALSE
243	TRUE	FALSE
244	FALSE	FALSE
245	FALSE	FALSE
246	TRUE	TRUE

247	FALSE	TRUE
248	TRUE	TRUE
249	TRUE	TRUE
250	FALSE	TRUE
251	FALSE	FALSE
252	FALSE	FALSE
253	FALSE	FALSE
254	FALSE	FALSE
255	FALSE	FALSE
256	FALSE	FALSE
257	FALSE	FALSE
258	FALSE	FALSE
259	FALSE	FALSE
260	FALSE	FALSE
261	TRUE	FALSE
262	FALSE	FALSE
263	FALSE	FALSE
264	FALSE	FALSE
265	FALSE	FALSE
266	FALSE	FALSE
267	FALSE	FALSE
268	FALSE	FALSE
269	FALSE	FALSE
270	FALSE	FALSE
271	TRUE	FALSE
272	FALSE	FALSE
273	FALSE	FALSE
274	FALSE	FALSE
275	FALSE	FALSE
276	FALSE	FALSE
277	FALSE	FALSE
278	FALSE	FALSE
279	TRUE	FALSE
280	TRUE	FALSE
281	FALSE	FALSE
282	FALSE	FALSE
283	FALSE	FALSE
284	FALSE	FALSE
285	FALSE	FALSE
286	FALSE	FALSE
287	FALSE	FALSE
288	FALSE	FALSE
289	FALSE	FALSE

290	FALSE	FALSE
291	FALSE	FALSE
292	FALSE	FALSE
293	FALSE	FALSE
294	FALSE	FALSE
295	FALSE	FALSE
296	TRUE	TRUE
297	TRUE	TRUE
298	FALSE	TRUE
299	TRUE	TRUE
300	FALSE	TRUE
301	FALSE	FALSE
302	FALSE	FALSE
303	FALSE	FALSE
304	FALSE	FALSE
305	FALSE	FALSE
306	FALSE	FALSE
307	FALSE	FALSE
308	FALSE	FALSE
309	FALSE	FALSE
310	FALSE	FALSE
311	FALSE	FALSE
312	FALSE	FALSE
313	FALSE	FALSE
314	FALSE	FALSE
315	FALSE	FALSE
316	FALSE	FALSE
317	FALSE	FALSE
318	FALSE	FALSE
319	FALSE	FALSE
320	FALSE	FALSE
321	FALSE	FALSE
322	FALSE	FALSE
323	FALSE	FALSE
324	FALSE	FALSE
325	FALSE	FALSE
326	FALSE	FALSE
327	FALSE	FALSE
328	FALSE	FALSE
329	FALSE	FALSE
330	FALSE	FALSE
331	FALSE	FALSE
332	FALSE	FALSE

333	FALSE	FALSE
334	FALSE	FALSE
335	FALSE	FALSE
336	FALSE	FALSE
337	FALSE	FALSE
338	FALSE	FALSE
339	FALSE	FALSE
340	FALSE	FALSE
341	FALSE	FALSE
342	FALSE	FALSE
343	TRUE	FALSE
344	FALSE	FALSE
345	FALSE	FALSE
346	TRUE	TRUE
347	FALSE	TRUE
348	FALSE	TRUE
349	TRUE	TRUE
350	FALSE	TRUE
351	FALSE	FALSE
352	FALSE	FALSE
353	FALSE	FALSE
354	FALSE	FALSE
355	FALSE	FALSE
356	TRUE	FALSE
357	FALSE	FALSE
358	FALSE	FALSE
359	FALSE	FALSE
360	FALSE	FALSE
361	FALSE	FALSE
362	FALSE	FALSE
363	FALSE	FALSE
364	FALSE	FALSE
365	FALSE	FALSE
366	FALSE	FALSE
367	FALSE	FALSE
368	FALSE	FALSE
369	FALSE	FALSE
370	FALSE	FALSE
371	FALSE	FALSE
372	FALSE	FALSE
373	FALSE	FALSE
374	FALSE	FALSE
375	FALSE	FALSE

376	FALSE	FALSE
377	FALSE	FALSE
378	FALSE	FALSE
379	FALSE	FALSE
380	FALSE	FALSE
381	FALSE	FALSE
382	FALSE	FALSE
383	FALSE	FALSE
384	FALSE	FALSE
385	FALSE	FALSE
386	FALSE	FALSE
387	FALSE	FALSE
388	FALSE	FALSE
389	FALSE	FALSE
390	FALSE	FALSE
391	FALSE	FALSE
392	FALSE	FALSE
393	FALSE	FALSE
394	FALSE	FALSE
395	FALSE	FALSE
396	FALSE	TRUE
397	TRUE	TRUE
398	FALSE	TRUE
399	FALSE	TRUE
400	FALSE	TRUE
401	FALSE	FALSE
402	FALSE	FALSE
403	FALSE	FALSE
404	FALSE	FALSE
405	FALSE	FALSE
406	FALSE	FALSE
407	FALSE	FALSE
408	FALSE	FALSE
409	FALSE	FALSE
410	FALSE	FALSE
411	FALSE	FALSE
412	FALSE	FALSE
413	TRUE	FALSE
414	FALSE	FALSE
415	FALSE	FALSE
416	FALSE	FALSE
417	FALSE	FALSE
418	FALSE	FALSE

419	FALSE	FALSE
420	FALSE	FALSE
421	FALSE	FALSE
422	FALSE	FALSE
423	FALSE	FALSE
424	FALSE	FALSE
425	FALSE	FALSE
426	FALSE	FALSE
427	FALSE	FALSE
428	FALSE	FALSE
429	FALSE	FALSE
430	FALSE	FALSE
431	FALSE	FALSE
432	FALSE	FALSE
433	FALSE	FALSE
434	FALSE	FALSE
435	TRUE	FALSE
436	FALSE	FALSE
437	FALSE	FALSE
438	FALSE	FALSE
439	TRUE	FALSE
440	TRUE	FALSE
441	FALSE	FALSE
442	FALSE	FALSE
443	FALSE	FALSE
444	FALSE	FALSE
445	FALSE	FALSE
446	FALSE	TRUE
447	FALSE	TRUE
448	FALSE	TRUE
449	FALSE	TRUE
450	FALSE	FALSE
451	FALSE	FALSE
452	FALSE	FALSE
453	FALSE	FALSE
454	FALSE	FALSE
455	FALSE	FALSE
456	FALSE	FALSE
457	FALSE	FALSE
458	FALSE	FALSE
459	FALSE	FALSE
460	FALSE	FALSE
461	FALSE	FALSE

462	FALSE	FALSE
463	FALSE	FALSE
464	FALSE	FALSE
465	FALSE	FALSE
466	FALSE	FALSE
467	FALSE	FALSE
468	FALSE	FALSE
469	FALSE	FALSE
470	FALSE	FALSE
471	TRUE	FALSE
472	FALSE	FALSE
473	FALSE	FALSE
474	FALSE	FALSE
475	TRUE	FALSE
476	FALSE	FALSE
477	FALSE	FALSE
478	FALSE	FALSE
479	FALSE	FALSE
480	FALSE	FALSE
481	FALSE	FALSE
482	FALSE	FALSE
483	FALSE	FALSE
484	FALSE	FALSE
485	FALSE	FALSE
486	FALSE	FALSE
487	TRUE	FALSE
488	FALSE	FALSE
489	FALSE	FALSE
490	FALSE	FALSE
491	FALSE	FALSE
492	FALSE	FALSE
493	FALSE	FALSE
494	FALSE	TRUE
495	FALSE	TRUE
496	FALSE	TRUE
497	FALSE	TRUE
498	FALSE	TRUE

Berdasarkan tabel 1 yang menunjukkan hasil prediksi dengan kenyataan jika dibentuk dalam *confusion matrix* maka akan tampil seperti tabel 2, setelah *confusion matrix* berhasil dibuat maka nilai akurasi dapat dihitung sesuai persamaan (2,11).

Tabel 1 *Confusion Matrix* hasil klasifikasi dataset CM1 dengan 1 fitur hasil seleksi fitur GR

		Kenyataan	
		TRUE	FALSE
Hasil Prediksi	TRUE	14	27
	FALSE	35	442

$$\text{Akurasi} = \frac{14 + 442}{14 + 442 + 27 + 35} = \frac{456}{498} = 0.8755 \times 100\% = 87.55\%$$