TESIS – IF 185401

WEIGHTED KNN MENGGUNAKAN *GREY RELATIONAL ANALYSIS* UNTUK SOLUSI NILAI YANG ABSEN PADA PREDIKSI KESALAHAN PERANGKAT LUNAK LINTAS RANAH

DESEPTA ISNA ULUMI
NRP. 5116201018

DOSEN PEMBIMBING
Daniel Oranova Siahaan, S.Kom., M.Sc., PD.Eng.
NIP. 197411232006041001

PROGRAM MAGISTER
BIDANG KEAHLIAN REKAYASA PERANGKAT LUNAK
DEPARTEMEN INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN KOMUNIKASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2019

[Halaman ini sengaja dikosongkan]

TESIS – IF 185401

WEIGHTED KNN USING GREY RELATIONAL ANALYSIS TO SOLVE MISSING VALUE FOR CROSS-PROJECT DEFECT

DESEPTA ISNA ULUMI
NRP. 5116201018

SUPERVISOR
Daniel Oranova Siahaan, S.Kom., M.Sc., PD.Eng.
NIP. 197411232006041001

MASTER PROGRAM
THE EXPERT OF SOFTWARE ENGINEERING
DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2019

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

Tesis ini disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Komputer (M. Kom)
di

Institut Teknologi Sepuluh Nopember Surabaya

oleh:
DESEPTA ISNA ULUMI
Nrp. 5116201018

Dengan judul:
Weighted KNN Menggunakan *Grey Relational Analysis* Untuk Solusi Nilai Yang Absen Pada Prediksi Kesalahan Perangkat Lunak Lintas Ranah

Tanggal Ujian : 19 Desember 2018
Periode Wisuda : 2019 Ganjil

Disetujui oleh:

Daniel Oranova Siahaan, S.Kom, M.Sc, PD.Eng (Pembimbing)
NIP. 197411232006041001

Prof. Ir. Drs. Ec. Riyanarto Sarno, M.Sc., Ph.D. (Penguji 1)
NIP. 195908031986011001

Dr. Ir. Raden Venantinus Hari Ginardi, M.Sc. (Penguji 2)
NIP. 196505181992031003

Dr. Eng. Radityo Anggoro, S.Kom., M.Sc. (Penguji 3)
NIP. 198410162008121002

Dekan Fakultas Teknologi Informasi dan Komunikasi,

Dr. Agus Zainal Arifin, S.Kom, M.Kom
NIP. 19720809 199512 1 001

[Halaman ini sengaja dikosongkan]

PERNYATAAN KEASLIAN

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tesis saya dengan judul:

***WEIGHTED* KNN MENGGUNAKAN *GREY RELATIONAL ANALYSIS* UNTUK SOLUSI NILAI YANG ABSEN PADA PREDIKSI KESALAHAN PERANGKAT LUNAK LINTAS**

adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pusaka.

Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai dengan peraturan yang berlaku.

Surabaya, 15 Januari 2019

Desepta Isna Ulumi
NRP: 5116201018

[Halaman ini sengaja dikosongkan]

WEIGHTED KNN MENGGUNAKAN *GREY RELATIONAL ANALYSIS* UNTUK SOLUSI NILAI YANG ABSEN PADA PREDIKSI KESALAHAN PERANGKAT LUNAK LINTAS RANAH

Nama	:	Desepta Isna Ulumi
NRP	:	5116201018
Pembimbing	:	Daniel Oranova Siahaan, S.Kom, M.Sc, PD.Eng.

ABSTRAK

Prediksi kesalahan perangkat lunak memiliki peranan penting dalam mendeteksi komponen yang paling rentan terjadi kesalahan perangkat lunak. Beberapa penelitian telah berupaya meningkatkan akurasi prediksi kesalahan perangkat lunak agar dapat mengelola sumber daya (manusia, biaya dan waktu) lebih baik. Namun penelitian sebelumnya masih membuat model prediksi kesalahan perangkat lunak untuk ranah tertentu saja. Belum terdapat penanganan dataset yang lintas ranah.

Penelitian ini memperbaiki model prediksi kesalahan perangkat lunak agar dapat menangani dataset yang digabung (lintas ranah) dengan jumlah fitur yang berbeda-beda. Agar jumlah fitur tiap dataset seimbang maka pengisian nilai yang absen akibat penggabungan dataset lintas ranah dilakukan. Penelitian ini mengembangkan metode *weighted* KNN untuk mengisi nilai yang absen tersebut. Dataset yang diperlengkapi tesebut selanjutnya diklasifikasi menggunakan naive bayes dan *random forest*. Penelitian ini juga mencari kumpulan fitur apa yang relevan dalam mendeteksi *defect* dengan cara melakukan analisis perbandingan metode seleksi fitur.

Untuk pengujian, penelitian ini menggunakan tujuh dataset NASA *public* MDP (*Modular toolkit for Data Preprocessing)*. Hasil pengujian menunjukkan bahwa data tidak imbang (*imbalance*) menghasilkan nilai *balance* terbaik jika metode naive bayes dikombinasi dengan metode seleksi fitur *information gain* (IG) atau *symmetric uncertainty* (SU), yaitu 0.4975. Hasil pengujian juga menunjukkan bahwa data imbang (*balance*) menghasilkan nilai *balance* terbaik jika metode *random forest* dikombinasi dengan metode seleksi fitur *gain ratio* (GR), yaitu 0.7795. Secara umum, hasil klasifikasi dengan masing-masing tujuh dataset NASA *public* MDP relative tidak jauh berbeda dari hasil klasifikasi data lintas ranah dimana hasil klasifikasi lintah ranah adalah 0.4975. Bahkan hasil ini masih diatas dari hasil klasifikasi atas dataset PC2, yaitu 0.4033.

Kata kunci*:* Kesalahan perangkat lunak, NASA *public* MDP, *weighted* KNN, *naive bayes, random forest*

[Halaman ini sengaja dikosongkan]

WEIGHTED KNN USING GREY RELATIONAL ANALYSIS TO SOLVE MISSING VALUE FOR CROSS-PROJECT DEFECT PREDICTION

Name	: Desepta Isna Ulumi
Student Identity Number	: 5116201018
Supervisor	: Daniel Oranova Siahaan, S.Kom, M.Sc,PD.Eng.

ABSTRACT

Defect prediction plays important roles in detecting vulnerable component within a software. Some researches have tried to improve the accuration of software defect prediction so that it helps developer to manage resources (human, cost, and time) better. Those researches focus on building the software defect prediction model only for a specific domain. Research on cross-project domain has not been carried out before.

This research developed a software defect prediction model for cross-project domain. Thus, the domain contains datasets with different number of features. To extend shorted features in a dataset, the method calculates the missing values. This research developed a method, called weighted KNN, to fill in the missing value. The refill datasets are then classified using naive bayes and random forest. This research also conducted a feature selection process to select relevant featues for detecting defects by means of a comparative analysis of methods of selection of features.

For the experimentation, this research used seven NASA public dataset MDPs. The results show that for imbalance data, naïve bayes combined with information gain (IG) or symmetric uncertainty (SU) feature selection produced the best balance, i.e. 0.4975. It also shows that for balance data, random forest combined with gain ratio (GR) produced the best balance, i.e. 0.7795. In general, the developed method performed relatively alike the previous method, which classifiy only specific domain, i.e. 0.4975. It even outperformed previous method for dataset PC2, i.e. 0.4033.

***Keywords**: Software defect*, NASA *public* MDP, *weighted* KNN, *Naive bayes, Random Forest*

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Segala puji bagi Allah SWT, yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis bias menyelesaikan Tesis yang berjudul "Weighted KNN Menggunakan *Grey Relational Analysis* untuk Solusi Nilai yang Hilang pada Prediksi Kesalahan Perangkat Lunak Lintas Ranah. Pengerjaan Tesis ini adalah suatu kesempatan yang sangat berharga bagi penulis untuk belajar memperdalam ilmu pengetahuan. Terselesaikannya buku Tesis ini tidak terlepas dari bantuan dan dukungan semua pihak. Oleh karena itu, penulis ingin menyampaikan terimakasih sebesar-besarnya kepada:

1. Allah SWT atas limpahan rahmat-Nya sehingga penulis dapat menyelesaikan buku Tesis ini dengan baik.
2. Kedua orang tua penulis, kedua orang tua (mertua), suami, anak yang selalu mendoakan agar selalu diberikan kelancaran dan kemudahan dalam menyelesaikan Tesis ini.
3. Bapak Daniel Oranova Siahaan, S.Kom., M.Sc, PD.Eng selaku dosen pembimbing yang telah memberikan kepercayaan, motivasi, bimbingan, nasehat, perhatian serta semua bantuan yang telah diberikan kepada penulis dalam menyelesaikan buku tesis ini.
4. Bapak Prof. Drs. Ec. Ir. Riyanarto S.M.Sc, Dr. Ir. Raden Venantius Hari Ginardi, M.Kom dan Dr. Eng. Radityo Anggoro, S.Kom., M.Sc. selaku dosen penguji yang telah memberikan bimbingan, saran, arahan, dan koreksi dalam pengerjaan buku tesis ini.
5. Bapak Waskitho Wibisono, S.Kom., M.Eng., PhD selaku ketua program pascasarjana Teknik Informatika ITS, dan Bapak Daniel Oranova Siahaan, S.Kom., M.Sc, PD.Eng selaku dosen wali penulis dan segenap dosen Teknik Informatika yang telah memberikan ilmunya.
6. Segenap staf Tata Usaha yang telah memberikan segala bantuan dan kemudahan kepada penulis selama menjalani kuliah di Teknik Informatika ITS.

7. Rekan-rekan mahasiswa Pasca Sarjana Teknik Informatika ITS tahun 2016 yang telah menemani dan memberika bantuan serta motivasi untuk segera menyelesaikan buku tesis ini.
8. Juga semua pihak yang belum sempat disebutkan satu per satu yang telah membantu penyelesaian buku tesis ini.

Sebagai manusia biasa, penulis menyadari bahwa buku tesis ini masih jauh dari kesempurnaan dan terdapat banyak kekurangan. Sehingga dengan segala kerendahan hati, penulis mengharapkan saran dan kritik yang membangun dari pembaca.

Surabaya, 19 Desember 2018

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

BAB 1
PENDAHULUAN

Pada bab ini akan dijelaskan mengenai beberapa hal dasar dalam pembuatan proposal penelitian yang meliputi latar belakang, perumusan masalah, tujuan, manfaat, kontribusi penelitian, dan batasan masalah.

1.1. Latar Belakang

Dalam rekayasa perangkat lunak, prediksi kesalahan perangkat lunak dapat secara tepat memperkirakan komponen perangkat lunak yang paling rentan mengalami kesalahan perangkat lunak [1]. Sebuah kesalahan perangkat lunak dapat diamati saat proses pengujian berlangsung [2]. Setiap kesalahan saat pengujian ditandai sebagai kesalahan perangkat lunak. Metrik perangkat lunak yang berisi pengukuran digunakan untuk mengidentifikasi modul yang berpotensi terjadinya kesalahan. Sehingga hal tersebut diharapkan dapat meminimalkan sumber daya (manusia, biaya dan waktu) yang dihabiskan dalam sebuah proyek karena proses pengujian difokuskan pada modul-modul yang paling rentan terjadinya kesalahan.

Terdapat banyak model prediksi kesalahan perangkat lunak yang telah berhasil dikembangkan. Salah satu penelitian yang memodelkan kesalahan perangkat lunak dengan menggunakan metode seleksi fitur *Greedy Forward Selection* (GFS) dan teknik klasifikasi *ensemble learning*, *Average Probability Ensemble* (APE) *Learning* [3]. Pada penelitian ini menggunakan 6 dataset NASA *public* MDP (*Modular toolkit for Data Preprocessing*) yaitu Ant-1.7, Camel-1.6, KC3, MC1, PC2, PC4 yang berisi matrik perangkat lunak yang umum digunakan termasuk loc (*line of code*), kompleksitas *mccabe, halsted's length, halsteds volume, halsted difficulty, total number of operand* (TNO) dan *branch count* (BC). Selain itu metode *ensemble learning* dapat mengatasi data yang tidak seimbang dan fitur yang redundan. Namun penelitian ini memiliki kekurangan spesifik terhadap dataset tertentu untuk model prediksi kesalahan perangkat lunak dan tidak semua matrik digunakan. Pada penelitian yang lain [4], menggunakan metode klasifikasi DPRAR (*DefectPrediction using Relational Association Rule*). Data yang digunakan NASA *public* MDP yaitu CM1, KC1, KC3, PC1, JM1, MC2, MW1, PC2, PC3 dan PC4. Model prediksi kesalahan perangkat lunak pada penelitian ini,

tahap awal *pre-processing* yaitu menentukan ketergantungan antara fitur dan target keluaran, menggunakan koefisien korelasi peringkat Spearman, yang tidak berpengaruh dihapus. Kemudian tahap pelatihan, menggunakan algoritma DPRAR yang berfokus untuk mengidentifikasi hubungan antara dua metrik perangkat lunak (fitur). Jika hasilnya relevan, maka fitur akan digunakan pada tahap selanjutnya. Pada tahap pengujian model yang telah dibangun diuji per dataset. Hasil yang diperoleh selanjutnya dikomparasi menggunakan beberapa alat ukur akurasi, *probability of detection* (pd), *specificity*, *precision*, dan *area under the ROC curve* (ROC). Namun model prediksi kesalahan perangkat lunak yang dibangun masih spesifik terhadap dataset tertentu. Jadi, untuk perangkat lunak lain belum tentu dapat digunakan. Terdapat penelitian yang memaparkan metode rejoELM (ELM *with reject option*) dan IrejoELM (*Imbalance* ELM *with reject option*) untuk proses klasifikasi [5]. Akan tetapi nilai pembobotan masih bersifat acak dan IrejoELM hanya bisa digunakan pada klasifikasi berbasis biner serta model hanya bisa digunakan untuk dataset tertentu. Nilai f-*measure* dari setiap data berbeda, Lucene 84.88%, Equinox 72,92% Ecl.JDT 64.88%, Ecl. PDE 89,67%, dan Mylyn 84,56%.

Adapun cara lain untuk meningkatkan akurasi prediksi kesalahan perangkat lunak adalah dengan menambahkan metode seleksi fitur pada setiap dataset. Pada penelitian ini menggunakan tujuh dataset NASA *public* MDP yaitu CM1, KC3, MW1, PC1, PC2, PC3 dan PC4. Setiap data memiliki ranah atau domain masing-masing. Ranah atau domain adalah ruang lingkup dari sekumpulan projek perangkat lunak. Setiap domain memiliki jumlah fitur yang berbeda dan memiliki keserupaan fitur. Fitur adalah sebuah properti atau karakteristik dalam sebuah data yang bervariasi, baik dari satu objek ke yang lain atau dari satu waktu ke waktu lain. Sedangkan seleksi fitur adalah pemilihan fitur untuk memperoleh metrik perangkat lunak yang baik. Lima metode seleksi fitur yang digunakan yaitu *gain ratio (GR), information gain (IG), one-r (OR), relief-f (RFF), dan symmetric uncertainty (SU)* [2]. Seleksi fitur dapat memberi informasi kombinasi fitur yang menghasilkan performa terbaik. Metode yang diusulkan mampu meningkatkan akurasi prediksi kesalahan perangkat lunak yang mengungguli metode sebelumnya.

Selain metode seleksi fitur, pemilihan metode klasifikasi juga mempengaruhi hasil dari model kesalahan perangkat lunak. Menurut penelitian [6], naïve bayes memiliki performa terbaik pada data *imbalance*, yang menggunakan data Haberman's survival yang diambil dari UCI *Machine Learning Repository*. Data Haberman's survival berasal dari penelitian yang dilakukan di Rumah Sakit Billings Universitas Chicago yang mengumpulkan data pasien yang bertahan hidup yang telah menjalani perawatan operasi kanker payudara. Terdiri dari 306 data dengan tiga fitur, dan dua kelas yaitu pasien selamat lima tahun atau lebih dan pasien meninggal dalam kurun waktu lima tahun. Pada Haberman's perbandingan kelas mayoritas dan minoritas adalah 81/225. Menurut penelitian [7], untuk data NASA *public* MDP, dengan melakukan seleksi fitur dan penyemibangan jumlah data terlebih dahulu sebelum diklasifikasi dapat meningkatkan akurasi dari pada menggunakan data primer. Hasil klasifikasi menunjukkan *random forest* lebih unggul dibandingkan metode klasifikasi yang lain, naïve bayes, KNN, J48, SVM, MLP, DSt. *Random forest* dibangun berdasarkan klasifikasi *decision tree*.

Hasil yang didapatkan dari kombinasi klasifikasi *cluster based classification* (CBC) dan seleksi fitur IG memberikan akurasi yang tinggi dibandingkan metode seleksi fitur yang lain. Akan tetapi metode ini kurang efisien, karena pemrosesan setiap dataset dilakukan satu-satu dengan model prediksi kesalahan perangkat lunak yang berbeda-beda sesuai dengan jumlah fitur setiap dataset. Oleh karena itu, diperlukan pemodelan prediksi kesalahan perangkat lunak untuk data lintas ranah dengan jumlah fitur yang berbeda.

Penelitian ini mengajukan pembangunan model prediksi kesalahan perangkat lunak menggunakan beberapa dataset NASA (lintas ranah) dengan jumlah fitur yang berbeda. Untuk mengatasi perbedaan jumlah fitur, penelitian ini mengusulkan pengisian nilai yang absen pada dataset NASA *public* MDP menggunakan metode *weighted* KNN. Selanjutnya proses klasifikasi akan dilakukan dengan menggunakan naïve bayes dan *random forest*. Pembangunan model prediksi kesalahan perangkat lunak diharapkan dapat mendekati akurasi prediksi kesalahan perangkat lunak yang spesifik. Sehingga pengembang perangkat lunak tidak membutuhkan sumber daya (manusia, biaya dan waktu) yang banyak untuk melakukan prediksi kesalahan perangkat lunak.

1.2. Perumusan Masalah

Rumusan masalah dalam penelitian ini adalah sebagai berikut:

1. Bagaimana mengisi fitur yang kosong (nilai yang absen) pada lintas dataset?
2. Bagaimana cara memilih fitur mana saja yang paling baik untuk memodelkan prediksi kesalahan perangkat lunak?
3. Algoritma seleksi fitur mana yang paling tepat digunakan dalam kasus ini?

1.3. Tujuan Penelitian

Tujuan dari penelitian ini adalah memodelkan prediksi kesalahan perangkat lunak dari gabungan beberapa dataset dengan jumlah fitur yang berbeda-beda.

1.4. Manfaat Penelitian

Manfaat penelitian ini adalah mempermudah pengembang perangkat lunak dalam melakukan prediksi kesalahan perangkat lunak menggunakan model prediksi yang dibangun dari gabungan beberapa dataset agar dapat lebih efisien dengan cara mengisi nilai yang absen dengan metode *weighted* kNN. Usulan metode tersebut diharapkan dapat meningkatkan akurasi performa, yang dilakukan secara bersamaan dengan jumlah fitur yang berbeda-beda.

1.5. Kontribusi Penelitian

Kontribusi yang diharapkan dari penelitian ini adalah membuat model prediksi kesalahan perangkat lunak berdasarkan gabungan dataset yang memiliki jumlah fitur yang berbeda-beda dengan cara mengisi nilai yang absen dengan metode *weighted* kNN dan pemilihan fitur terbaik.

1.6. Batasan Masalah

Batasan masalah dalam penelitian ini adalah:

1. Penelitian berfokus pada prediksi kesalahan perangkat lunak
2. Dataset yang digunakan 7 dataset dari NASA *public* MDP yaitu CM1, KC3, MW1, PC1, PC2 PC3, dan PC4.

BAB 2
KAJIAN PUSTAKA

Pada bab ini akan dijelaskan tentang pustaka yang terkait dengan landasan penelitian. Pustaka yang terkait adalah seputar kesalahan perangkat lunak, model prediksi kesalahan perangkat lunak, pengisian nilai yang absen, pemilihan fitur, dataset NASA *public* MDP, dan *weighted* kNN.

2.1. Kesalahan Perangkat Lunak

Kesalahan perangkat lunak adalah memprediksi modul yang rawan terjadinya kesalahan perangkat lunak berdasarkan metrik perangkat lunak versi sebelumnya dan label kesalahan/ label bukan kesalahan [8]. Prediksi kesalahan perangkat lunak sangat penting sehingga memotivasi beberapa peneliti untuk membangun model prediksi kesalahan perangkat lunak [1]. Terdapat beberapa penelitian prediksi kesalahan perangkat lunak seperti yang terdapat pada Tabel 2.1.

Tabel 2.1 Komparasi Penelitian Prediksi Kesalahan Perangkat Lunak

No	Judul	Masalah	Metode	Hasil
1	*Software defect prediction using ensemble learning on selected features* [3]	1. Menunjukkan efek positif dari penggunaan kombinasi *feature selection* dan *ensemble learning* dalam meningkatkan akurasi klasifikasi prediksi kesalahan perangkat lunak 2. Mengusulkan algoritma *ensemble learning* yang tahan terhadap	Seleksi fitur menggunakan Greedy Forward Selection (GFS) dan klasifikasi menggunakan Ensemble Learning yaitu Average Probability Ensemble (APE)	G means dari klasifikasi dataset ant-1.7 : 84%, camel-1.6 : 79%, KC3 : 83%, MC1 : 90%, PC2 : 95%, PC4 : 95%, camel-1.6 : 79%

No	Judul	Masalah	Metode	Hasil
		data yang tidak seimbang dan fitur yang redundan		
2	*Software defect prediction using relational association rule mining* [4]	Mengusulkan DPRAR untuk mengidentifikasi modul terjadi kesalahan perangkat lunak	*Preprocessing* menggunakan Spearman dan klasifikasi menggunakan DPRAR	Pd dari klasifikasi dataset CM1 0.92, KC1 0,81, KC3 0,88, PC1 0,88, JM1 0,84, MC2 0,77, MW1 0,88, PC2 0,93, PC3 0,85, PC4 0,81.
3.	*Classification with reject option for software defectprediction* [5]	Mengusulkan penggabungan *reject option* dalam proses klasifikasi	Klasifikasi menggunakan rejoELM (ELM *with reject option*) dan IrejoELM (*Imbalance* ELM *with reject option*)	Lucene 84.88%, Equinox 72,92% Ecl.JDT 64.88%, Ecl. PDE 89,67%, dan Mylyn 84,56%.
4.	Perbaikan prediksi kesalahan perangkat lunak menggunakan seleksi fitur dan *cluster-based classification* [2]	Mengatasi fitur redundan dan fitur tidak relevan pada kesalahan perangkat lunak	5 seleksi fitur *gain ratio* (GR), *information gain* (IG), *one-r* (OR), *relief-f* (RFF), dan *symmetric uncertainty* (SU) dan metode klasifikasi menggunakan *Cluster Based Classification* (CBC)	Kombinasi *Cluster Based Classification* (CBC) dengan IG menghasilkan nilai balance terbaik 63,91%.

Tabel 2.1 menampilkan beberapa metode yang digunakan dalam melakukan seleksi fitur dan klasifikasi prediksi kesalahan perangkat lunak. Penelitian ini akan berfokus pada pengembangan model prediksi kesalahan perangkat lunak yang menggunakan acuan penelitian [2]. Di penelitian sebelumnya dapat meningkatkan nilai rata-rata *balance* CBC sebesar 9.12%, yang awalnya tanpa menggunakan seleksi fitur 54.79%, setelah menggunakan seleksi fitur IG menjadi 63.91%. Namun pengujian dilakukan tiap dataset. Sehingga pada penelitian ini mengembangkan model prediksi kesalahan perangkat lunak yang dilakukan dengan gabungan beberapa dataset dengan jumlah fitur yang berbeda. Untuk mengatasi jumlah fitur yang berbeda dengan mengisi nilai yang absen menggunakan metode *wighted* kNN seperti yang telah dilakukan oleh paper [9]. Penggunaan lima metode seleksi fitur seperti yang telah dilakukan pada paper [2]. Pada penelitian ini, untuk mengetahui metode seleksi fitur mana yang bagus untuk kasus generik. Metode klasifikasi yang digunakan seperti pada paper [1] menggunakan metode Naïve Bayes karena cocok pada kasus prediksi kesalahan perangkat lunak dengan predictor yang simpel.

2.2. Model Prediksi Kesalahan Perangkat Lunak

Pada perbandingan penelitian ini, penulis akan membandingkan model-model pada kasus prediksi kesalahan perangkat lunak. Pada penelitian yang berjudul "*Software Defect Prediction Using Ensemble Learning on Selected Features*" mengusulkan model prediksi kesalalahan perangkat lunak dengan melakukan seleksi fitur terlebih dahulu menggunakan metode GFS, *Pearson's Correlation, Fisher's Criterion* kemudian melakukan klasifikasi menggunakan *ensemble learning* yaitu kombinasi dari 7 algoritma klasifikasi (*random forests, gradient boosting, stochastic gradient descent, logistic regression, multinomial naive bayes, bernoulli naive bayes, weighted support vector machine*).Hasil dari setiap algoritma kemudian dirata-rata sebagai hasil akhir dari klasifikasi tiap dataset[3].

Pada penelitian yang berjudul "*Software defect prediction using relational association rule mining"* mengusulkan model prediksi kesalalahan perangkat lunak dengan

preprocessing dengan metode statistik spearman. Selanjutnya proses klasifikasi menggunakan DPRAR untuk tiap dataset [21].

Penelitian yang berjudul *"Classification with reject option for software defect prediction"* mengusulkan model prediksi kesalahan perangkat lunak tanpa melakukan seleksi fitur dan proses klasifikasi dengan menggabungkan *reject option* untuk tiap dataset[5].

Pada penelitian lain yang berjudul "*Perbaikan Prediksi Kesalahan Perangkat Lunak Menggunakan Seleksi Fitur dan Cluster-Based Classification"* model prediksi yang diusulkan adalah dengan melakukan reduksi data numerik *redundant* dengan kelas yang sama, diskritisasi data numerik menggunakan metode EBD, reduksi data biner *redundant* dengan kelas yang sama sebelum melakukan seleksi fitur (IG, GR, OR, RFF, SU). Kemudian proses klasifikasi menggunakan metode CBC [2]. Namun pada penelitian ini proses klasifikasi masih dilakukan per dataset.

2.3. *Weighted* KNN

Nearest Neighbor (NN) adalah metode yang digunakan untuk mengidentifikasi data poin yang belum diketahui kelasnya [10]. K-*Nearest Neighbor* (kNN) adalah metode yang membagi data *training* dan data *testing*. Jarak dievaluasi dari semua data *training* ke data *testing* dan poin yang memiliki nilai jarak paling rendah disebut *nearest neighbor*. k-*Nearest neighbor* memiliki beberapa keuntungan komputasi data *training* yang cepat, mudah untuk dipelajari, tahan terhadap data *training* yang *noisy*, efektif jika data *training* berukuran besar [10]. Namun metode kNN masih memiliki kekurangan keterbatasan memori, komputasi yang komplek, proses *running* yang perlahan dan mudah tertipu dengan fitur yang tidak relevan. Teknik ini mudah dimplementasikan tetapi nilai k mempengaruhi hasilnya. Sehingga T. Bailey dan A. K. Jain memodifikasi kNN dengan pembobotan dan memberi nama *weighted* kNN (wkNN). wkNN adalah metode yang mengevaluasi jarak berdasarkan nilai k dan bobot setiap menghitung nilai. Keuntungan yang didapat dari wkNN adalah mengatasi keterbatasan kNN dengan menambah bobot di setiap k, menggunakan semua sampel *training* tidak hanya nilai k, dan cocok diimplementasikan di semua dataset

2.4. Seleksi Fitur

Berdasarkan penelitian sebelumnya, penggunaan semua fitur pada data set sebagai masukan model prediksi belum tentu dapat meningkatkan nilai prediksi [11]. Salah satu faktor penyebabnya adalah kualitas data dari setiap fitur [12]. Pendekatan metode seleksi fitur yang digunakan pada penelitian ini adalah filter, yaitu pemilihan fitur berdasarkan peringkat fitur.

2.4.1. *Information Gain* (IG)

Information gain adalah salah satu teknik seleksi fitur yang mampu menilai pentingnya fitur dengan mengukur informasi kelas terkait [13]. Umumnya *information gain* dapat mengubah nilai ketidakpastian sebuah informasi (*entropy*) menjadi ukuran nilai informasi yang akan didapat. Adapun rumus seleksi fitur menggunakan information gain seperti berikut:

$$IG(Class|Attribute = H(Class) - H(Class|Attribute \quad (2.1)$$

dimana H adalah *entropy*. Diasumsikan bahwa A adalah semua fitur dan kelas bergantung dari semua *training*. Nilai (*a,y*) dengan $y \in Class$ mendefinisikan nilai dari contoh spesifik untuk fitur $a \in A$, V adalah nilai satu set fitur yaitu $V = \{value(a, y)|a \in A \cap y \in Class\}$. Rumus IG pada setiap fitur $a \in A$ didefinisikan sebagai berikut:

$$IG(Class, a) = H(class) - \sum_{v \in V} \frac{\{y \in Class \vee value(a,y)=v\}}{|Class|} xH(y\{Class|value(a, y) = v\}) \quad (2.2)$$

2.4.2. *Gain Ratio* (GR)

Gain ratio memodifikasi teknik *information gain* dengan memperhatikan jumlah hasil yang diperoleh dari pengujian fitur [14]. Adapun rumus fitur sebagai berikut:

$$GR(Class, a) = \frac{IG(Class,a)}{H(a)} \quad (2.3)$$

dimana perhitungan $H(a)$ sebagai berikut :

$$H(a) = -\sum_{v \in V} \frac{|\{y \in Class \vee value(a,y)=v\}|}{|Class|} \; x$$

$$log_2 \frac{|\{y \in Class \vee value(a,y)=v\}|}{|Class|} \qquad (2.4)$$

Notasi rumus GR sama dengan notasi rumus IG.

2.4.3. *One-R* (OR)

One R dibangun berdasarkan satu fitur yang disebut sebagai satu aturan untuk setiap fitur di dataset [9]. Adapun algoritma One-R sebagai berikut [12]:

For each fitur *f*,

For each nilai *v* dari domain *f*

Pilih set *instance* dengan fitur *f* mempunyai nilai *v*

Diasumsikan *c* adalah kelas yang memiliki frekuensi paling tinggi

Terapkan "*if* fitur *f* memiliki nilai *v then* kelas *is c*" pada fitur *f*

Output aturan dengan akurasi klasifikasi tertinggi

2.4.4. *Relief-F* (RFF)

Relief-F adalah teknik seleksi fitur yang melakukan evaluasi berulang kali dan memberikan bobot setiap fitur berdasarkan kemampuan fitur membedakan antar kelas dan memilih fitur yang bobotnya melebihi ambang batas yang ditetapkan pengguna sebagai fitur yang relevan [15]. Perhitungan bobot didasarkan pada probabiltas tetangga terdekat dengan kelas berbeda untuk nilai fitur yang berbeda dan kelas sama untuk nilai fitur yang sama. Semakin tinggi nilai perbedaan antara keduanya maka semakin relevan fitur tersebut. Adapun algoritma *relief* adalah sebagai berikut [16]:

Input : sebuah *training* set *D*, jumlah iterasi *m*, jumlah *nearest neighbors* k, jumlah fitur *n*, *predefine* bobot fitur batas ambang δ

Output : sebuah fitur *S* yang didasari oleh fitur yang bobotnya lebih besar dari batas ambang δ

Langkah 1.

Diasumsikan $S = \emptyset$, set semua bobot fitur $W(ft) = 0, t = 1,2, \ldots n$

Langkah 2.

1. Pilih sampel *R* dari *D* secara acak.
2. Cari *k nearest neighbors* $H_i = (i = 1,2, \dots, k)$dari kelas yang sama dan *k nearest neighbors* $M_i(C) = (i = 1,2, \dots, k)$dari kelas *C* yang berbeda.
3. For *t = 1* to *n* do

$$W(F_t) = W(F_t) - \sum_{i=1}^{k} \frac{diff(F_t,R,H_i)}{mk} + \sum_{C \notin ClassR} \frac{\frac{P(C)}{1-P(Class(R))} \sum_{i=1}^{k} diff(F_t,R,M_i(C))}{(mk)} \quad (2.5)$$

Langkah 3.

For *t = 1* to *n* do

If $W(F_t) > \delta$ then add fitur (F_t) to *S*

$P(C)$ adalah Jarakribusi probabilitas kelas *C*, $Class(R)$ adalah kelas yang masuk kategori kelas R, $M_i(C)$ adalah menunjukkan *i near miss* dari *R* dalam kelas *C*, $diff(F_t, R_1, R_2)$adalah menunjukkan perbedaan R_1 dan R_2 pada F_t

If (F_t) adalah diskrit:

$$diff(F_t, R_1, R_2) = \begin{cases} 0; R_1[F_t] = R_2[F_t] \\ 1; R_1[F_t] \neq R_2[F_t] \end{cases}$$

If (F_t) adalah kontinyu:

$$diff(F_t, R_1, R_2) = \frac{|R_1[F_t] - R_2[F_t]|}{max[F_t] - min[F_t]}$$

dimana R_1 dan R_2 adalah dua sampel, $R_1[F_t]$ dan $R_2[F_t]$adalah nilai fitur dari R_1dan R_2.

2.4.5. *Symmetric Uncertainty* (SU)

Symmetric Uncertainty (SU) juga mengkompensasi bias IG terhadap fitur dengan nilai yang lebih berbeda dan menormalkan kembali nilainya ke kisaran 0 sampai 1 [14]. Nilai 1 menunjukkan bahwa fitur *a* benar-benar memprediksi nilai kelas dan nilai 0 menunjukkan bahwa kelas dan fitur *a* adalah independen. Adapun SU didefinisikan sebagai berikut:

$$SU(Class, a) = 2x\frac{IG(Class,a)}{H(Class)+H(a)} \quad (2.6)$$

Karena persamaan memiliki keserupaan anatara IG, GR dan SU maka akan memiliki hasil yang serupa [14].

2.5. *Grey Relational Analysis* (GRA)

Teori *grey* diajukan oleh Profesor Julong Deng pada tahun 1982, yang meneliti dengan beberapa orang mengenai informasi yang diketahui maupun tidak diketahui [17]. Metode ini cocok digunakan untuk sistem dengan sampel berskala kecil [18]. *Grey Relational Analysis* secara umum diterapkan dalam mengevaluasi atau menilai sebuah proyek yang kompleks dengan sedikit informasi [19]. *Grey Relational Analysis* adalah teknik pengukuran yang diusulkan untuk mengetahui hubungan *instance* yang terdapat *missing value* dan *instance* yang tidak terdapat *missing value* dengan menghitung *Grey Relational Grade* (GRG) [18]. Sebelum tahap GRG terlebih dahulu menghitung jarak antara *referential instance* (*instance* yang terdapat *missing value*) dan *compared instances* (*instance* yang tidak terdapat *missing value*). Jarak setiap data yang terdapat *missing value* dan data yang tidak terdapat *missing value* didefinisikan sebagai berikut

$$jarak(x_i, x_j) = \sqrt{\sum_{l=1}^{n}(x_{il} - x_{jl})^2}. \quad (2.7)$$

Dimana x_i adalah *instance* yang terdapat *missing value* dan x_j *instance* yang tidak terdapat *missing value*. Nilai jarak untuk memperoleh jarak terdekat. Untuk setiap *instance* yang terdapat *missing value* diisi dengan nilai pada data *complete* (tidak terdapat *missing value*).

Menurut penelitian [20], sebelum masuk ketahap selanjutnya data perlu dilakukan normalisasi. Normalisasi didefinisikan dengan persamaan 2.8,

$$x_i(j) = \frac{x_i(j) - min_{i=1}^{n}[x_i(j)]}{max_{i=1}^{n}[x_i(j)] - min_{i=1}^{n}[x_i(j)]}. \quad (2.8)$$

Dimana $x_i(j)$ adalah urutan data ke i yang sesuai dengan fitur ke j, $min_{i=1}^{n}[x_i(j)]$ adalah nilai minimum setiap fitur dan $max_{i=1}^{n}[x_i(j)]$ adalah nilai maksimum setiap fitur.

Kemudian mencari jarak yang telah dinormalisasi, diukur dengan cara absolut dan dalam bentuk matriks, yang didefinisikan dengan persamaan 2.9,

$$\triangle_{oi}(j) = x'_o(j) - x'_i(j). \quad (2.9)$$

Dimana $\triangle_{oi}(j)$ jarak antara nilai yang absen dengan nilai yang tidak terdapat nilai yang absen setelah dinormalisasidan, $x'_0(j)$ adalah nilai fitur ke *j* yang memiliki *missing value*, $x'_i(j)$ adalah nilai fitur ke *j* yang tidak memiliki *missing value* setelah dinormalisasi. Nilai *grey relational coefficient* dihitung untuk mengetahui hubungan hasil eksperimen yang ideal dan aktual. Adapun rumus *grey relational coefficient* sebagai berikut:

$$GRC\ \gamma_{oi}(j) = \frac{\triangle_{min} + \rho\triangle_{max}}{\triangle_{oi}(j) + \rho\triangle_{max}}. \quad (2.10)$$

$GRC\ \gamma_{oi}(j)$ adalah *grey relational coefficient*, ρ (ρ €[0,1]) adalah koefisien pembeda yang pada umumnya ditetapkan ρ = 0.5 [18][19][20], $\triangle_{min}$ adalah nilai minimum pada $\triangle_{oi}(j)$dan $\triangle_{max}$ adalah nilai maksimum pada $\triangle_{oi}(j)$.

Setelah mendapatkan hasil *grey relational coefficient*, nilai rata-rata dari *grey relational coefficient* diambil sebagai nilai *grey relational grade* yang didefinisikan sebagai berikut:

$$GRG(Y, X_i) = \frac{1}{m}\sum_{k=1}^{m} GRC\ \gamma_{oi}(j), \quad (2.11)$$

dimana *m* adalah jumah atribut. Semakin tinggi nilai $GRG(Y, X_i)$ korelasi antara Y dan X_i semakin kuat. Jadi, jika nilai $GRG(Y, X_1)$ lebih besar dibandingkan nilai $GRG(Y, X_2)$ maka perbedaan antara Y dan X_1lebih kecil dibandingkan perbedaan antara Y dan X_2. Semakin kuat korelasi, semakin besar bobot yang diperoleh. Dalam kebanyakan kasus, bobot setiap tetangga terdekat didefinisikan sebagai berikut:

$$w_j = \frac{1}{d_j}, \quad (2.12)$$

dimana d_j adalah jarak antara tetangga j dan target *i*. Adapun rumus untuk mengisi *missing value* sebagai berikut:

$$x_{ip} = \frac{\sum_{j=1}^{k} w_j x_{jp}}{\sum_{j}^{k} w_j}, \quad (2.13)$$

dimana x_{ip} adalah *missing value* dari *instance*X_i.

2.6. Naïve Bayes

Naïve Bayes adalah salah satu algoritma klasifikasi yang paling banyak digunakan karena sederhana, efektif dan tahan [21]. Berdasarkan [22], *machine learning* yang paling banyak digunakan pada kesalahan perangkat lunak adah naïve bayes sebanyak 47.7%. Algoritma naïve bayes adalah algoritma yang didasarkan pada asumsi ketidak bergantungan (*independent*) fitur [21]. Adapun rumus naïve bayes untuk data kontinyu sebagai berikut [23]:

$$P(X = x|C = c) = g(x, \mu_c, \sigma_c) \quad (2.14)$$

dimana $P(X = x|C = c)$ adalah model dengan nilai kontinyu antara 0 sampai dengan 1 yang mempresentasikan probabilitas bahwa fitur *x* akan diambil pada kelas *c*. $g(x, \mu_c, \sigma_c)$ didefinisikan sebagai berikut :

$$g(x, \mu_c, \sigma_c) = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \quad (2.15)$$

2.7. Random Forest

Random forest adalah *ensemble* dari klasifier dengan struktur *tree* [24]. Setiap *tree* dari *forest* tersebut memberikan *vote* (pilihan), menentukan setiap *input* ke label kelas yang paling memungkinkan. Menurut penelitian [24] kelebihan *random forest* adalah metode tergolong cepat, tahan terhadap *noise,* mudah menangani data numerik maupun kategori. Berdasarkan percobaan pada penelitian [7], *random forest* memberikan performa terbaik pada *balance* data. Adapun alur kerja *random forest* sebagai berikut [25]:

1. Membuat pohon keputusan *k* secara acak (D1, D2, D3,…, Dk) dari data D. Tahapan ini dinamakan *boostrap*
2. Memilih *m* secara acak dari masing-masing indeks N pada masing masing simpul dari pohon klasifikasi. Fitur yang paling baik dipilih dari *m* index kandidat dan node diklasifikasi. Tahapan ini dinamakan *random fitur selection*.
3. Mengulangi langkah 2 untuk membentuk *k decision trees,* sehingga membentuk *random forest*
4. Hasil final klasifikasi berdasarkan mayoritas hasil *voting* kombinasi klasifikasi *random forest.* Adapun rumus klasifikasi *random forest* sebagai berikut:

$$f(x_t) = majority\ vote\ \{h_i(x)\}\ (i = 1,2, \dots k) \quad (2.16)$$

dimana *majority vote* adalah hasil dari *majority vote.*

2.8. Nilai *Balance*

Nilai *balance* adalah jarak *euclidean* ke titik ideal pd dan pf (pd = 1 dan pf = 0 [21][8]. Probabilitas prediksi (pd) adalah nilai keberhasilan dalam memprediksi kesalahan perangkat lunak. Probabilitas *false alarm* (pf) adalah kelas yang tidak terdapat kesalahan, tetapi diprediksi sebagai kelas yang salah. Nilai pd dan pf ideal adalah pd=1 dan pf=0 [8]. Semakin tinggi nilai pd dan semakin rendah nilai pf maka nilai *balance* juga semakin tinggi.

Untuk mendapatkan nilai pengukuran maka rumus yang digunakan dapat dilihat pada persamaan (2.17) sampai dengan (2.19) di bawah ini:

$$\text{pd} = \frac{TP}{(TP+FN)} \quad (2.17)$$

$$\text{pf} = \frac{FP}{(FP+TN)} \quad (2.18)$$

$$\text{nilai } balance = 1 - \frac{\sqrt{(0-pf)^2 + (1-pd)^2}}{\sqrt{2}} \quad (2.19)$$

dimana *True Positive* (TP) adalah jumlah kasus kelas kesalahan perangkat lunak, diprediksi benar sebagai kelas kesalahan perangkat lunak. *True Negative* (TN) adalah jumlah kasus kelas bukan kesalahan perangkat lunak, diprediksi benar sebagai kelas bukan kesalahan perangkat lunak. *False Positive* (FP) adalah jumlah kelas bukan kesalahan perangkat lunak diprediksi sebagai kelas kesalahan perangkat lunak. *False Negative* (FN) adalah jumlah kelas kesalahan perangkat lunak diprediksi tidak merugikan. Untuk memperoleh nilai TP, TN, FP dan FN didapatkan dari metrik *confussion*.

2.9. K *Fold Cross Validation*

Metode pengujian yang paling umum digunakan untuk tahap pengujian yaitu k *fold cross validation*. K *fold cross validation* membagi data dalam satu putaran menjadi dua kategori yaitu data *training* dan data *testing.* Jika terdapat 6000 data dan *fold* terbagi menjadi 10-*fold*, maka 9-*fold* dijadikan sebagai data *training*

dan 1-*fold* sebagai data *testing*. Proses ini diulang k kali dan masing-masing *fold* satu kali digunakan sebagai data *testing* [26].

BAB 3
METODE PENELITIAN

Dalam melakukan penelitian ini, terdapat beberapa tahapan yang akan dilakukan yaitu studi literatur, rancangan metode, pengujian dan evaluasi, penyusunan laporan dan jadwal kegiatan penelitian seperti yang ditunjukkan pada Gambar 3.1

Gambar 3.1 Diagram Alur Penelitian

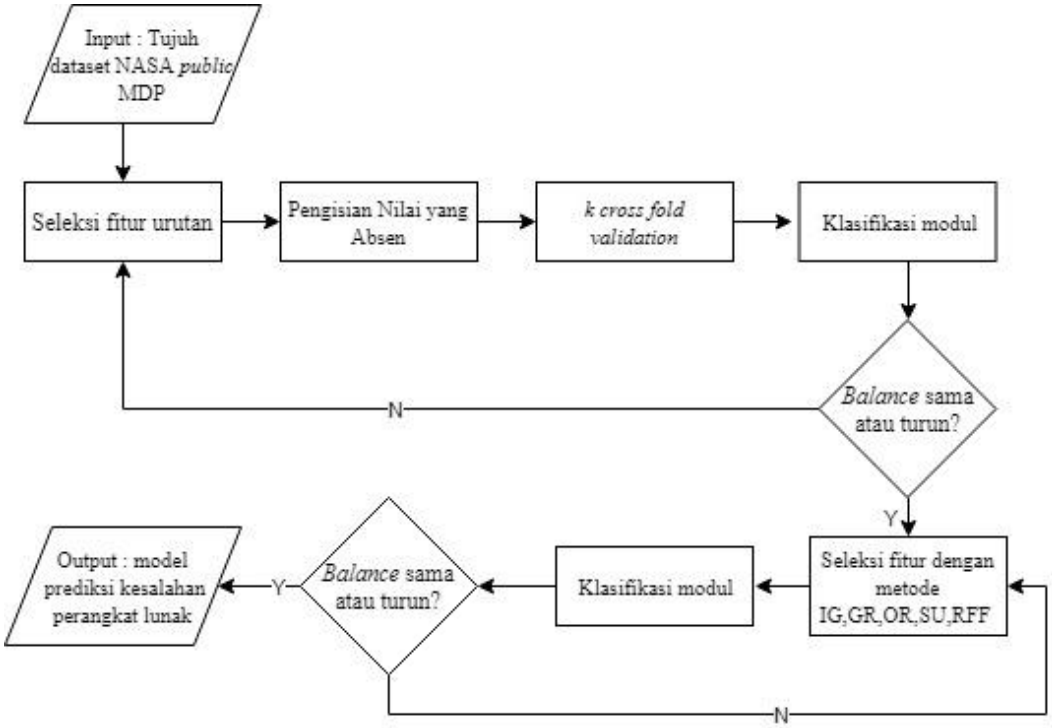
3.1. Studi Literatur

Studi literatur dilakukan dengan mempelajari metode yang diusulkan dari berbagai literatur seperti jurnal, maupun sumber lainnya yang sesuai dengan penelitian ini. Beberapa konsep dasar yang berkaitan dengan penelitian ini yaitu prediksi kesalahan perangkat lunak, metode pengisian nilai yang absen menggunakan weighted KNN, seleksi fitur Information Gain (IG), Gain Ratio (GR), One-R (OR), Relief-F (RFF), Symmetric Uncertainty (SU) dan metode klasifikasi menggunakan Naïve Bayes.

3.2. Rancangan Metode

Rancangan metode dalam penelitian ini terdiri dari metode seleksi fitur urutan, pengisian nilai yang absen menggunakan *weighted* KNN, klasifikasi

menggunakan Naïve Bayes, seleksi fitur *Information Gain* (IG), *Gain Ratio* (GR), *One-R* (OR), *Relief-F* (RFF), *Symmetric Uncertainty* (SU). Rancangan metode dalam penelitian ini secara umum seperti terlihat pada Gambar 3.2



Gambar 3.2 Rancangan Metode

3.2.1. Seleksi Fitur Urutan

Pada tahapan awal yaitu melakukan seleksi fitur urutan dari dataset yang masih memiliki *missing value*. Sebelum melakukan seleksi fitur data yang redundan direduksi [2]. Data redundan adalah data yang memiliki nilai fitur dan kelas yang sama seperti contoh pada Tabel 3.1.

Tabel 3.1 Contoh Data Redundan pada PC1

No	loc	v(g)	ev(g)	iv(G)	N	*defect*
1	6	1	1	1	11	T
2	23	4	1	4	101	T
3	5	1	1	1	11	T
4	5	1	1	1	11	T
5	43	9	7	7	186	T

Setelah data tidak terdapat redundan, maka dilakukan seleksi fitur dengan mengolah fitur yang terdapat di semua dataset terlebih dahulu yang disajikan pada Tabel 3.2. Kemudian jumlah fitur terbanyak selanjutnya yang ada pada Tabel 3.3. Jika terdapat dataset *missing value* maka dilakukan pembangunan data sintetis menggunakan *weighted* KNN dan klasifikasi menggunakan naïve bayes. Dilakukan perulangan hingga nilai *balance* sama atau turun.

Tabel 3.2 Fitur yang Terdapat di Semua Dataset

A	Kategori	Fitur	Representasi	Dataset NASA public MDP							B
				CM1	KC3	MW1	PC1	PC2	PC3	PC4	
1	line of code	LOC total	LOC	1	1	1	1	1	1	1	7
3		LOC e1ecutable	SLOC	1	1	1	1	1	1	1	7
4		LOC comments	CLOC	1	1	1	1	1	1	1	7
5		LOC code and comment	C&SLOC	1	1	1	1	1	1	1	7
8	halstead	Number of operators	N1	1	1	1	1	1	1	1	7
9		Number of operands	N2	1	1	1	1	1	1	1	7
11		Number of unique operators	n1	1	1	1	1	1	1	1	7
12		Number of unique operands	n2	1	1	1	1	1	1	1	7
13		Length	L	1	1	1	1	1	1	1	7
14		Difficulty	D	1	1	1	1	1	1	1	7
16		Volume	V	1	1	1	1	1	1	1	7
17		Programming effort	E	1	1	1	1	1	1	1	7
18		Programming time	T	1	1	1	1	1	1	1	7
19		Fault estimate	BLOC	1	1	1	1	1	1	1	7
20		Content (intellegence) vocabulary	I	1	1	1	1	1	1	1	7
21	mcCabe	Cyclomatic comple1ity	v(G)	1	1	1	1	1	1	1	7
24		Decision comple1ity	iv(G)	1	1	1	1	1	1	1	7
26		Essential comple1ity	ev(G)	1	1	1	1	1	1	1	7
32	miscellaneous	Branch count	Branch_C	1	1	1	1	1	1	1	7
41	classification	Defectmodule	defect(true/ false)	1	1	1	1	1	1	1	7

Ket : A = Nomor fitur, B = Jumlah fitur

Tabel 3.3 Jumlah Fitur Terbanyak Selanjutnya

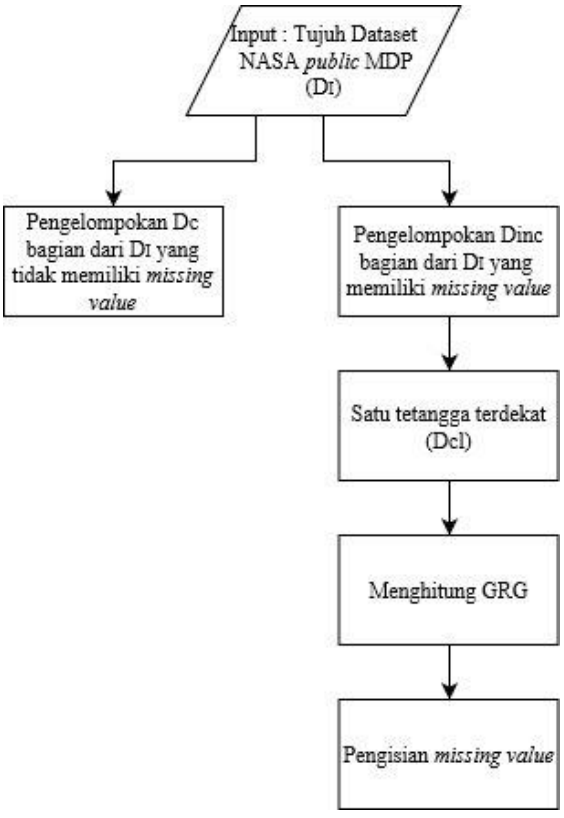
A	Kategori	Fitur	Representasi	Dataset NASA public MDP							B
				CM1	KC3	MW1	PC1	PC2	PC3	PC4	
1	line of code	LOC total	LOC	1	1	1	1	1	1	1	7
3		LOC e1ecutable	SLOC	1	1	1	1	1	1	1	7
4		LOC comments	CLOC	1	1	1	1	1	1	1	7
5		LOC code and comment	C&SLOC	1	1	1	1	1	1	1	7
8	halstead	Number of operators	N1	1	1	1	1	1	1	1	7
9		Number of operands	N2	1	1	1	1	1	1	1	7
11		Number of unique operators	n1	1	1	1	1	1	1	1	7
12		Number of unique operands	n2	1	1	1	1	1	1	1	7

A	Kategori	Fitur	Representasi	Dataset NASA public MDP							B
				CM1	KC3	MW1	PC1	PC2	PC3	PC4	
13		Length	L	1	1	1	1	1	1	1	7
14		Difficulty	D	1	1	1	1	1	1	1	7
16		Volume	V	1	1	1	1	1	1	1	7
17		Programming effort	E	1	1	1	1	1	1	1	7
18		Programming time	T	1	1	1	1	1	1	1	7
19		Fault estimate	BLOC	1	1	1	1	1	1	1	7
20		Content (intellegence) vocabulary	I	1	1	1	1	1	1	1	7
21	mcCabe	Cyclomatic comple1ity	v(G)	1	1	1	1	1	1	1	7
24		Decision comple1ity	iv(G)	1	1	1	1	1	1	1	7
26		Essential comple1ity	ev(G)	1	1	1	1	1	1	1	7
32	miscellaneous	Branch count	Branch_C	1	1	1	1	1	1	1	7
41	classification	Defectmodule	defect(true/false)	1	1	1	1	1	1	1	7
2	line of code	LOC bank	BLOC	1	1	1	1	0	1	1	6

Ket : A = Nomor fitur, B = Jumlah fitur

3.2.2. Pengisian Nilai yang Absen Menggunakan WkNN

Tahapan pada *weighted* KNN terdapat empat proses yaitu pengelompokan data, menghitung satu tetangga terdekat, menghitung GRG, dan pengisian *missing value* seperti yang terdapat pada Gambar 3.3.



Gambar 3.3 Tahapan Weighted KNN

Berdasarkan Gambar 3.3, tujuh dataset NASA *public* MDP dikelompokkan menjadi dua yaitu kelompok data berdasarkan label kelas *non-defect* dan *defect* terlebih dahulu sebelum masuk ke proses pengelompokan data *complete* dan *incomplete*. Untuk proses hingga pengisian *missing value* memiliki proses yang sama. Jika *missing value* dari kelas kelas *non-defect* dan *defect* sudah terisi maka digabungkan menjadi satu dan siap untuk diproses pada tahap selanjutnya yaitu klasifikasi. Adapun penjelasan pengisian nilai yang absen kelas *non-defect*:

1. Pengelompokan Data *Non-Defect* dan *Defect*

Setelah data yang redundan direduksi kemudian data dibagi menjadi dua yaitu *non-defect* pada Tabel 3.5 dan *defect* pada Tabel 3.6. Adapun data awal tujuh dataset NASA public MDP terdapat pada Tabel 3.4.

Tabel 3.4 Tujuh Dataset NASA *public* MDP (D_1)

Sumber	Rec ke-	Fitur						
		1	2	3	4	5	6	kelas
CM1	1	1.1	1.4	1.4	1.3	1.3	1.4	0
KC3	2	5	6	1	0	0.18	6	1
MW1	3	9	17	0	4	0.16	9	1
PC1	4	3	1	2	5	2	2	0
	5	3	1	1	10	3	1	1
PC2	6	1	4	7	24	0.13		0
	7	1	1	11	3	0.08		0
	8	7	7	21	6	0.17		1
	9	9	3	27	13	0.17		1
PC3	10	18	4	2	6	0.17	15	0
	11	10	1	1	19	0.17	1	0
	12	12	3	1	5	0.16	4	0
	13	6	2	5	4	0.23	37	0
PC4	14	17	5	2	8	0.25	11	0
	15	2	1	1	1	0.17	5	0
	16	36	1	18	43	0.16	21	1

Ket : kelas 0 = *non-defect*, kelas 1 = *defect*

Tabel 3.5 Data Kelas *Non-Defect*

Sumber	Rec ke-	Fitur						Kelas
		1	2	3	4	5	6	
CM1	1	1.1	1.4	1.4	1.3	1.3	1.4	0

Sumber	Rec ke-	Fitur						Kelas
		1	2	3	4	5	6	
PC1	4	3	1	2	5	2	2	0
PC2	6	1	4	7	24	0.13		0
	7	1	1	11	3	0.08		0
PC3	10	18	4	2	6	0.17	15	0
	11	10	1	1	19	0.17	1	0
	12	12	3	1	5	0.16	4	0
	13	6	2	5	4	0.23	37	0
PC4	14	17	5	2	8	0.25	11	0
	15	2	1	1	1	0.17	5	0

Tabel 3.6 Data Kelas *Defect*

Sumber	Rec ke-	Fitur						Kelas
		1	2	3	4	5	6	
KC3	2	5	6	1	0	0.18	6	1
MW1	3	9	17	0	4	0.16	9	1
PC1	5	3	1	1	10	3	1	1
PC2	8	7	7	21	6	0.17		1
	9	9	3	27	13	0.17		1
PC4	16	36	1	18	43	0.16	21	1

Pada penjelasan selanjutnya berfokus pada data kelas *non-defect.* Karena penjelasan kelas *defect* memiliki proses yang sama.

2. Pengelompokan Data *Complete* (D_c) dan Data *Incomplete* (D_{inc})

Setelah data dikelompokkan kelas *non-defect*, data dibagi menjadi dua lagi yaitu data *complete* adalah data yang tidak memiliki *missing value* (D_c) dan data *incomplete* adalah data yang memiliki *missing value* (D_{inc}) seperti contoh pada Tabel 3.7 sampai Tabel 3.8.

Tabel 3.7 Data *Complete* (D_c)

Sumber	Rec ke-	Fitur					
		1	2	3	4	5	6
CM1	1	1.1	1.4	1.4	1.3	1.3	1.4
PC1	4	3	1	2	5	2	2
PC3	10	18	4	2	6	0.17	15
	11	10	1	1	19	0.17	1
	12	12	3	1	5	0.16	4

Sumber	Rec ke-	Fitur					
		1	2	3	4	5	6
	13	6	2	5	4	0.23	37
PC4	14	17	5	2	8	0.25	11
	15	2	1	1	1	0.17	5

Tabel 3.8. Contoh Data *Incomplete* (D_{inc})

Sumber	Rec ke	Fitur					
		1	2	3	4	5	6
PC2	6	1	4	7	24	0.13	
	7	1	1	11	3	0.08	

3. Menghitung Satu Tetangga Terdekat

Tahap selanjutnya adalah menghitung satu tetangga terdekat menggunakan persamaan (2.7) seperti pada Tabel 3.9 dan Tabel 3.10 yang telah dijelaskan pada bab sebelumnya.

Tabel 3.9 Jarak Satu Tetangga Terdekat

(xi,xj)	Nilai Jarak	Nilai pada D_c	(xi,xj)	Nilai Jarak	Nilai pada D_c
(6,1)	23.55396		(7,1)	9.834043	
(6,4)	20.06233		(7,4)	9.627378	
(6,10)	25.25869		(7,10)	19.69792	
(6,11)	12.28827	1	(7,11)	20.90474	
(6,12)	22.78159		(7,12)	15.13296	
(6,13)	20.80889		(7,13)	7.9387	37
(6,14)	23.19514		(7,14)	19.44297	
(6,15)	23.97919		(7,15)	10.24735	

Pada *missing value record* ke 6, memiliki jarak terdekat dengan *record* ke 11. Kemudian melihat data *complete record* ke 11 pada fitur ke 6, nilainya adalah 1. Begitu juga untuk *missing value record* ke 7 memiliki jarak terdekat dengan *record* ke 13 dan memiliki nilai 37. Kemudian data awal D_I yang terdapat *missing value* diisi dengan nilai tersebut yaitu 1 dan 37 yang kemudian disebut D_{cl} seperti pada Tabel 3.10.

Tabel 3.10 Hasil Satu Tetangga Terdekat

Sumber	Rec ke-	Fitur					
		1	2	3	4	5	6
CM1	1	1.1	1.4	1.4	1.3	1.3	1.4
PC1	4	3	1	2	5	2	2
PC2	6	1	4	7	24	0.13	1
	7	1	1	11	3	0.08	37
PC3	10	18	4	2	6	0.17	15
	11	10	1	1	19	0.17	1
	12	12	3	1	5	0.16	4
	13	6	2	5	4	0.23	37
PC4	14	17	5	2	8	0.25	11
	15	2	1	1	1	0.17	5
min		1	1	1	1	0.08	1
max		18	5	11	24	2	37

Masing-masing fitur dicari nilai *minimum* dan maksimum. Nilai tersebut akan digunakan untuk menghitung normalisasi menggunakan persamaan (2.8). Hasil normalisasi seperti pada Tabel 3.11.

Tabel 3.11 Contoh Hasil Normalisasi

Sumber	Rec ke-	Fitur					
		1	2	3	4	5	6
CM1	1	0.00588	0.1	0.04	0.01304	0.63542	0.01111
PC1	4	0.11765	0	0.1	0.17391	1	0.02778
PC2	6	0	0.75	0.6	1	0.02604	0
	7	0	0	1	0.08696	0	1
PC3	10	1	0.75	0.1	0.21739	0.04688	0.38889
	11	0.52941	0	0	0.78261	0.04688	0
	12	0.64706	0.5	0	0.17391	0.04167	0.08333
	13	0.29412	0.25	0.4	0.13043	0.07813	1
PC4	14	0.94118	1	0.1	0.30435	0.08854	0.27778
	15	0.05882	0	0	0	0.04688	0.11111

Setelah mendapatkan nilai $x'_0(j)$ yaitu menghitung jarak antara *instance* yang terdapat *missing value* yaitu data PC2 *record* ke 6 dan 7 dengan *instance* yang tidak terdapat missing value data CM1, PC1, PC3, PC4 yang merupakan hasil

setelah normalisasi yang dinamakan $\triangle_{oi}(j)$. Untuk menghitung $\triangle_{oi}(j)$ terdapat pada persamaan (2.9). Adapun hasil$\triangle_{oi}(j)$ dapat dilihat pada Tabel 3.12.

Tabel 3.12 Contoh Jarak Hasil Normalisasi

Sumber	Rec ke	Fitur					
		1	2	3	4	5	6
CM1	1	0.0767	0.31623	0.9798	0.27187	0.79713	0.99443
PC1	4	0.343	0	0.94868	0.29488	1	0.98601
PC3	10	1	0.86603	0.94868	0.36116	0.21651	0.78174
	11	0.72761	0	1	0.83406	0.21651	1
	12	0.8044	0.70711	1	0.29488	0.20412	0.95743
	13	0.54233	0.5	0.7746	0.20851	0.27951	0
PC4	14	0.97014	1	0.94868	0.46625	0.29756	0.84984
	15	0.24254	0	1	0.29488	0.21651	0.94281

4. Menghitung GRG

Setelah menghitung jarak hasil normalisasi tahap selanjutnya adalah menghitung nilai *Grey Relational Coefficient* (GRC) terlebih dahulu dengan menggunakan persamaan (2.10). Pada tabel 3.10 menghitung GRC pada dataset PC2.

Tabel 3.13 Contoh Hasil GRC PC2

Sumber	Rec ke-	Fitur					
		1	2	3	4	5	6
CM1	1	1	0.61257	0.86133	0.90803	0.54283	0.33458
PC1	4	0.6841	1	0.87983	0.87868	0.46942	0.33647
PC3	10	0.38446	0.36603	0.87983	0.80385	0.98272	0.3901
	11	0.46977	1	0.84973	0.5	0.98272	0.33333
	12	0.44212	0.41421	0.84973	0.87868	1	0.34307
	13	0.55328	0.5	1	1	0.90329	1
PC4	14	0.39227	0.33333	0.87983	0.7082	0.88285	0.37042
	15	0.77666	1	0.84973	0.87868	0.98272	0.34655

Setelah menghitug GRC, maka nilai GRG dapat dihitung menggunakan persamaan (2.11). Adapun hasil GRG PC2 seperti pada Tabel 3.14.

Tabel 3.14 Contoh Hasil GRG PC2

Sumber	Rec ke-	GRG	Ranking
CM1	1	0.709892	3
PC1	4	0.708083	4
PC3	10	0.634497	7
	11	0.689259	5
	12	0.654635	6
	13	0.826095	1
PC4	14	0.594484	8
	15	0.805722	2

Kemudian mengurutkan GRG dari yang terbesar sampai yang terkecil. Semakin besar nilai GRG menunjukkan korelasi antara *referential instance* dan *compared instance* semakin kuat [18]. Pengurutan nilai GRG seperti pada Tabel 3.14.

5. Pengisian *Missing Value*

Mengisi *missing value* menggunakan persamaan (2.12) dan (2.13). Untuk setiap *missing value* memiliki bobot yang berbeda-beda dan didapatkan hasil seperti Tabel 3.15.

Tabel 3.15 Contoh Hasil Pengisian *Missing Value* PC2

Sumber	Rec ke	Fitur					
		1	2	3	4	5	6
CM1	1	1.1	1.4	1.4	1.3	1.3	1.4
PC1	4	3	1	2	5	2	2
PC2	6	1	4	7	24	0.13	5.58188
	7	1	1	11	3	0.08	23.0312
PC3	10	18	4	2	6	0.17	15
	11	10	1	1	19	0.17	1
	12	12	3	1	5	0.16	4
	13	6	2	5	4	0.23	37
PC4	14	17	5	2	8	0.25	11
	15	2	1	1	1	0.17	5

Setelah selesai melakukan pengisian *missing value* pada kelas *non-defect*, kemudian melakukan pengisian pada kelas *defect* yang memiliki proses sama

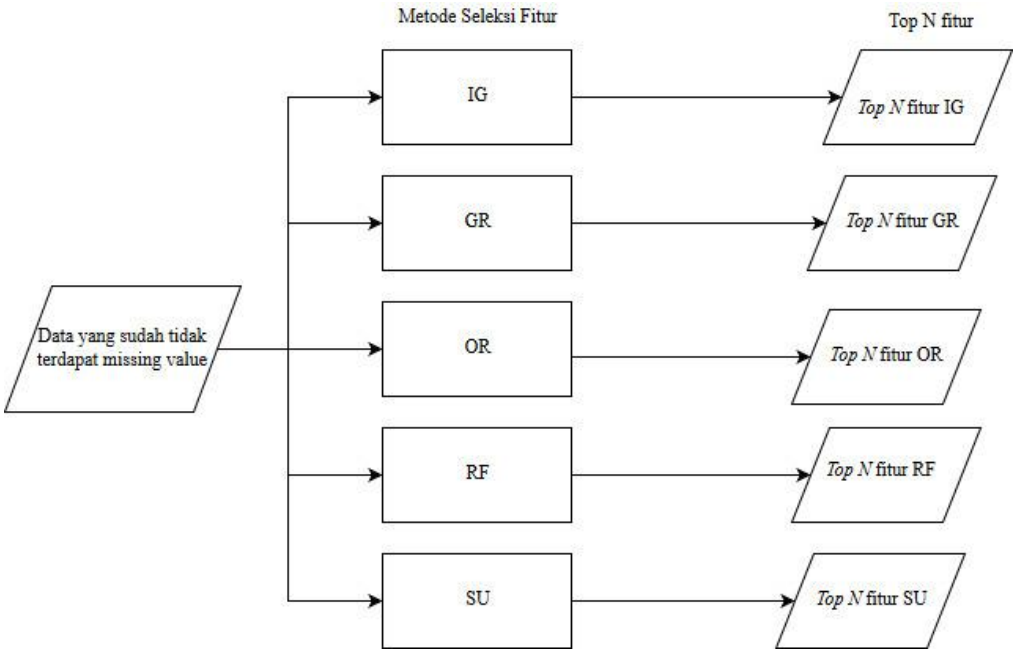
dengan kelas *non-defect*. Kemudian data *non-defect* dan *defect* digabung kembali serta dapat dilakukan proses selanjutnya yaitu tahap klasifikasi.

3.2.3. Klasifikasi Naïve Bayes dan Random Forest

Klasifikasi dalam penelitian ini menggunakan metode naïve bayes seperti persamaan (2.14) dan (2.15) dan *random forest* yang memiliki alur proses seperti yang telah dijelaskan pada bab sebelumnya.

3.2.4. Seleksi Fitur Menggunakan Metode IG, GR, OR, RFF dan SU

Proses seleksi fitur untuk masing-masing algoritma menghasilkan ranking fitur, maka fitur tersebut dipilih. Pemilihan fitur dilakukan untuk menunjukkan fitur mana saja yang relevan. Semakin tinggi ranking fitur menunjukkan semakin relevan. Sejumlah *Top N* fitur diambil dari nilai peringkat fitur tertinggi yang digunakan untuk masukan proses selanjutnya. Penentuan nilai *N* fitur berdasarkan iterasi [12]. Jumlah *Top N* fitur yang dapat menghasilkan nilai prediksi tertinggi pada proses klasifikasi yang akan digunakan sebagai hasil. Adapun alur seleksi fitur pada tahap ini ditunjukkan pada Gambar 3.4



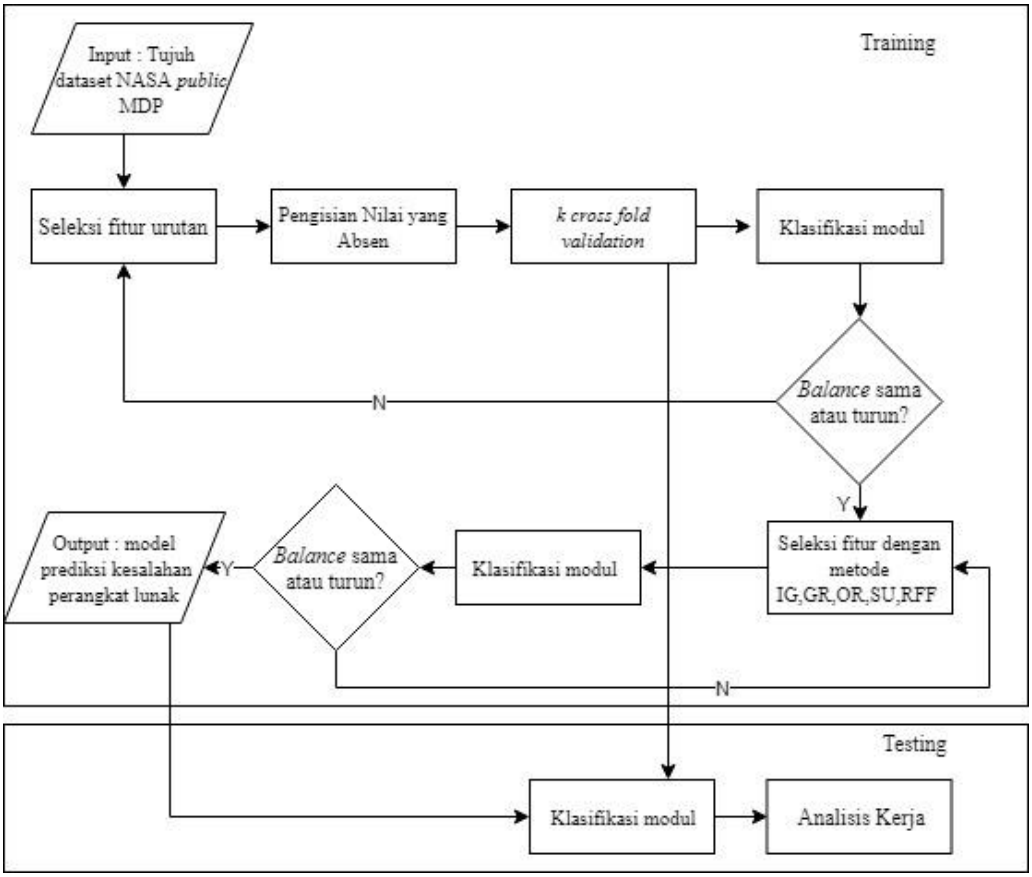
Gambar 3.4 Alur Seleksi Fitur IG, GR, OR, RFF, SU

Berdasarkan Gambar 3.4, terdapat lima metode seleksi fitur yaitu IG, GR, OR, RFF dan SU. Masing-masing metode seleksi fitur diambil *Top N* fitur.

Kemudian urutan fitur teratas (Top 1) dicari nilai *balance* dan menghasilkan nilai *balance* pertama (R1). Urutan fitur teratas berikutnya (Top 2) diambil ditambah dengan Top 1 fitur kemudian dicari nilai *balance* kedua (R2). Jika nilai *balance* kedua (R2) lebih tinggi dari pada nilai *balance* pertama (R1) iterasi dilanjutkan sampai nilai *balance* Rn lebih kecil dari nilai *balance* Rn-1.

3.3. Pengujian dan Evaluasi

Pengujian dilakukan dengan pengisian nilai yang absen terlebih dahulu. Kemudian membagi data *training* dan data *testing* menggunakan *k-fold cross validation*. Jika terdapat 6000 data dan *fold* terbagi menjadi 10 *fold*, maka 9 *fold* dijadikan sebagai data *training* dan 1 *fold* data *testing*. Proses selanjutnya adalah klasifikasi menggunakan naïve bayes dan random forest. Jika nilai *balance* sama atau turun maka iterasi berhenti, dilanjutkan pemilihan fitur yang paling optimal menggunakan beberapa algoritma seleksi fitur yaitu IG, GR, OR, RFF, SU yang masing-masing menghasilkan ranking fitur. Fitur yang relevan kemudian diklasifikasi sampai nilai *balance* sama atau turun dan *output* menghasilkan model prediksi kesalahan lunak. Sedangkan pada tahap *testing* data yang sudah memiliki model kesalahan perangkat lunak akan digunakan untuk memproses data *testing* NASA public MDP. Adapun alur pengujian dapat dilihat pada Gambar 3.5



Gambar 3.5 Alur Pengujian

Terdapat dua step skenario pengujian dalam penelitian ini, yaitu: step 1 dan step 2.

Step 1: fokus pada proses pengisian nilai yang absen, terdiri dari tiga skenario pengujian, yaitu:

1. Rata-rata tetangga terdekat k=2 sampai k=10
2. Rata-rata tetangga terdekat k=2 sampai k=10 dan menggunakan pemeringkatan fitur (*information gain)*
3. Satu tetangga terdekat dan menggunakan pemeringkatan fitur (*information gain)*

Step 2: hasil terbaik dari step 1, terdiri dari dua skenario pengujian, yaitu:

1. Data tidak imbang: menggunakan semua dataset (6293 data), *defect : non-defect* = 545 : 5748
2. Data imbang: menggunakan semua dataset (1090 data), *defect : non-defect* = 545 : 545

Alat ukur yang biasa digunakan untuk mengevaluasi hasil prediksi adalah metric *confussion* [8]. Aturan metrik *confussion* berisi *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), *False Negative* (FN). *True Positive* (TP) adalah jumlah kasus kelas kesalahan perangkat lunak, diprediksi benar sebagai kelas kesalahan perangkat lunak. *True Negative* (TN) adalah jumlah kasus kelas bukan kesalahan perangkat lunak, diprediksi benar sebagai kelas bukan kesalahan perangkat lunak. *False Positive* (FP) adalah jumlah kelas bukan kesalahan perangkat lunak diprediksi sebagai kelas kesalahan perangkat lunak. *False Negative* (FN) adalah jumlah kelas kesalahan perangkat lunak diprediksi tidak merugikan. Adapun metrik *confussion* dapat dilihat pada Tabel 3.16.

Pada penelitian ini, akan diukur menggunakan probabilitas prediksi (pd), probabilitas *false alarm* (pf), dan *balance*. Probabilitas prediksi (pd) adalah nilai keberhasilan dalam memprediksi kesalahan perangkat lunak. Probabilitas *false alarm* (pf) adalah kelas yang tidak terdapat kesalahan, tetapi diprediksi sebagai kelas yang salah. Nilai pd dan pf ideal adalah pd=1 dan pf=0 [8]. Semakin tinggi nilai pd dan semakin rendah nilai pf maka nilai *balance* juga semakin tinggi.

Tabel 3.16 Metrik *Confussion*

	Aktual	
Prediksi	*Defect*	*Non-Defect*
Faulty	TP	FP
Non-Defect	FN	TN

Untuk mendapatkan nilai pengukuran maka rumus yang digunakan dapat dilihat di bawah ini:

$$\text{pd} = \frac{TP}{(TP+FN)} \tag{3.1}$$

$$\text{pf} = \frac{FP}{(FP+TN)} \tag{3.2}$$

$$\text{nilai } balance = 1 - \frac{\sqrt{(0-pf)^2+(1-pd)^2}}{\sqrt{2}} \tag{3.3}$$

Data yang digunakan adalah tujuh dataset NASA *public* MDP yaitu CM1, KC3, MW1, PC1, PC2, PC3, PC4 yang berisi sejumlah data numerik berdasarkan penelitian sebelumnya [2]. Adapun rincian setiap data ada di Tabel 3.17.

Tabel 3.17 Rincian Tujuh Dataset NASA *public* MDP

No	Nama	Modul	Fault	%	Bahasa	Deskripsi
1	CM1	498	49	9,84	C/C++	Simulasi untuk uji coba pembakaran
2	KC3	458	43	9,38	Java	Manajemen penyimpanan untuk data ground
3	MW1	403	31	7,69	C	Percobaan gravitasi nol terhadap pembakaran
4	PC1	1109	77	6,94	C	Perangkat lunak penerbangan untuk satelit yang mengorbit di bumi
5	PC2	5589	23	0,41	C	Simulator dinamis pada sistem kontrol
6	PC3	1563	160	10,24	C	Perangkat lunak penerbangan untuk satelit yang mengorbit di bumi
7	PC4	1458	178	12,21	C	Perangkat lunak penerbangan untuk satelit yang mengorbit di bumi

3.4. Penyusunan Laporan dan Jadwal Kegiatan Penelitian

Pada tahap penyusunan laporan ini akan dilakukan penulisan laporan hasil penelitian dari setiap tahapan yang dilakukan. Tujuan dari tahapan ini adalah menghasilkan dokumentasi tertulis dari penelitian yang dilakukan. Jadwal penelitian yang dilakukan dapat dilihat pada Tabel 3.18.

Tabel 3.18 Jadwal Kegiatan Penelitian

No	Kegiatan	Bulan																							
		I				II				III				IV				V				VI			
1	Studi Literatur																								
2	Pengerjaan Paper																								
3	Perancangan																								
4	Analisa																								
5	Penyusunan Laporan																								
6	Publikasi																								

[Halaman ini sengaja dikosongkan]

BAB 4
HASIL PENELITIAN DAN PEMBAHASAN

Bab ini dijelaskan tentang uji coba dan evaluasi pada usulan metode yang telah dilakukan. Terdapat tiga sub pokok bahasan, yaitu pengumpulan data, skenario pengujian, dan analisis hasil.

4.1. Pengumpulan Dataset

Dataset yang digunakan pada penelitian ini adalah tujuh dataset NASA *public* MDP, yaitu KC3, MW1, CM1, PC1, PC2, PC3 dan PC4 yang bertipe numerik. Dataset tersebut dapat diakses secara umum melalui repositori dataset perangkat lunak *promise.* Ketujuh dataset dilakukan reduksi redudansi data numerik terlebih dahulu sebelum digunakan. Pada semua dataset tidak ada redudansi data numerik dengan kelas yang berbeda. Dataset PC2 memiliki jumlah redudansi data numerik terbanyak dapat dilihat di Tabel 4.1. Data yang digunakan merupakan data setelah dilakukan reduksi sebanyak 6293 data.

Tabel 4.1 Rincian Redudansi Data Numerik

No	Dataset	Jumlah fitur	Jumlah data	Jumlah redudansi	Jumlah data setelah reduksi
1	KC3	40	458	132	326
2	MW1	38	403	21	382
3	CM1	22	498	56	442
4	PC1	22	1109	155	954
5	PC2	37	5589	4183	1406
6	PC3	38	1563	124	1439
7	PC4	38	1458	114	1344

4.1.1. Seleksi Fitur Urutan

Tahap awal memilih fitur yang dimiliki oleh ketujuh dataset NASA *public* MDP yang dapat dilihat di Tabel 4.2.

Tabel 4.2 Rincian 19 Fitur

No	No Fitur	Nama Fitur	Representasi
1	1	*LOC total*	LOC
2	3	*LOC executable*	SLOC

3	4	*LOC comments*	CLOC
4	5	*LOC code and comment*	C&SLOC
5	8	*Number of operators*	N1
6	9	*Number of operands*	N2
7	11	*Number of unique operators*	n1
8	12	*Number of unique operands*	n2
9	13	*Length*	L
10	14	*Difficulty*	D
11	16	*Volume*	V
12	17	*Programming effort*	E
13	18	*Programming time*	T
14	19	*Error estimate*	BLOC
15	20	*Content (intellegence) vocabulary*	I
16	21	*Cyclomatic comple1ity*	v(G)
17	24	*Decision comple1ity*	iv(G)
18	26	*Essential comple1ity*	ev(G)
19	32	*Branch count*	Branch_C

Sembilan belas fitur diklasifikasi menggunakan naïve bayes dan didapatkan nilai *balance* pertama. Kemudian menambah satu fitur yang memiliki *missing value* paling sedikit yaitu pada nomor fitur kedua, LOC *bank*, yang dapat dilihat pada Tabel 4.3.

Tabel 4.3 Rincian 20 Fitur

No	No Fitur	Nama Fitur	Representasi
1	1	*LOC total*	LOC
2	3	*LOC executable*	SLOC
3	4	*LOC comments*	CLOC
4	5	*LOC code and comment*	C&SLOC
5	8	*Number of operators*	N1
6	9	*Number of operands*	N2
7	11	*Number of unique operators*	n1
8	12	*Number of unique operands*	n2
9	13	*Length*	L
10	14	*Difficulty*	D
11	16	*Volume*	V
12	17	*Programming effort*	E
13	18	*Programming time*	T
14	19	*Error estimate*	BLOC

No	No Fitur	Nama Fitur	Representasi
15	20	*Content (intellegence) vocabulary*	I
16	21	*Cyclomatic comple1ity*	v(G)
17	24	*Decision comple1ity*	iv(G)
18	26	*Essential comple1ity*	ev(G)
19	32	*Branch count*	Branch_C
20	2	*LOC bank*	BLOC

Fitur ke 20 yang terdapat *missing value* diisi terlebih dahulu menggunakan *weighted* KNN. Kemudian 20 fitur diklasifikasi menggunakan naïve bayes dan mendapatkan nilai *balance* yang kedua. Nilai *balance* kedua lebih baik dibandingkan nilai *balance* pertama. Sebelum menambah satu fitur lagi, diranking dahulu menggunakan *information gain* karena terdapat 17 fitur yang memiliki *missing value* di dua dataset, CM1 dan PC1, yang dapat dilihat pada Tabel 4.4. Sehingga metode seleksi fitur IG dipakai untuk pemeringkatan 17 fitur.

Tabel 4.4 Hasil Ranking 17 Fitur Menggunakan *Information Gain*

Ranking	Nomor fitur	Nama fitur	Representasi	*Score*
1	38	*Parameter count*	Parameter_C	0.07384
2	40	*Modified condition count*	Mod_Cond_C	0.06757
3	23	*Decision density*	dd(G)	0.05554
4	35	*Decission count*	Dec_C	0.05279
5	22	*Cyclomatic density*	vd(G)	0.05274
6	36	*Edge count*	Edge_C	0.05125
7	33	*Call pairs*	Call_C	0.05051
8	6	*Number of lines*	nl	0.04716
9	34	*Condition count*	Cond_C	0.0468
10	25	*Design density*	id(G)	0.04349
11	30	*Norm cyclomatic compl*	Nomv(G)	0.03489
12	31	*Maintenance severity*	Mainsev	0.03432
13	39	*Multiple condition count*	Mul_Cond_C	0.03306
14	37	*Node count*	Node_C	0.02832
15	7	*Percent comment*	%comment	0.02608
16	27	*Essential density*	ed(G)	0.02238
17	15	*Level*	I/D	0.00744

Dari tabel di atas, fitur ke 38 yang diambil untuk tahap selanjutnya karena memiliki *score* ranking tertinggi. Fitur ke 38 digabungkan dengan 20 fitur sebelumnya, yang dapat dilihat di Tabel 4.5. Kemudian *missing value* diisi menggunakan *weighted* KNN dan diklasifikasi menggunakan naïve bayes. Nilai *balance* ketiga lebih baik dari pada nilai *balance* kedua.

Tabel 4.5 Rincian 21 Fitur

No	Nomor Fitur	Nama Fitur	Representasi
1	1	*LOC total*	LOC
2	3	*LOC executable*	SLOC
3	4	*LOC comments*	CLOC
4	5	*LOC code and comment*	C&SLOC
5	8	*Number of operators*	N1
6	9	*Number of operands*	N2
7	11	*Number of unique operators*	n1
8	12	*Number of unique operands*	n2
9	13	*Length*	L
10	14	*Difficulty*	D
11	16	*Volume*	V
12	17	*Programming effort*	E
13	18	*Programming time*	T
14	19	*Error estimate*	BLOC
15	20	*Content (intellegence) vocabulary*	I
16	21	*Cyclomatic comple1ity*	v(G)
17	24	*Decision comple1ity*	iv(G)
18	26	*Essential comple1ity*	ev(G)
19	32	*Branch count*	Branch_C
20	2	*LOC bank*	BLOC
21	38	*Parameter count*	Parameter_C

Fitur yang ke 22 diambil dari nomor fitur 40 sesuai dengan pemeringkatan 17 fitur menggunakan *information gain*. *Missing value* diisi menggunakan *weighted* KNN dan diklasifikasi. Nilai *balance* keempat lebih bagus dari pada nilai *balance* ketiga.

Fitur yang ke 23 diambil dari nomor fitur 23, *decision density*. *Missing value* diisi menggunakan *weighted* KNN dan diklasifikasi. Nilai *balance* kelima

lebih rendah dibandingkan nilai *balance* keempat. Sehingga yang diolah ke tahap selanjutnya sebanyak 22 fitur yang dapat dilihat pada Tabel 4.6.

Tabel 4.6 Rincian 22 Fitur

No	Nomor Fitur	Nama Fitur	Representasi
1	1	*LOC total*	LOC
2	3	*LOC executable*	SLOC
3	4	*LOC comments*	CLOC
4	5	*LOC code and comment*	C&SLOC
5	8	*Number of operators*	N1
6	9	*Number of operands*	N2
7	11	*Number of unique operators*	n1
8	12	*Number of unique operands*	n2
9	13	*Length*	L
10	14	*Difficulty*	D
11	16	*Volume*	V
12	17	*Programming effort*	E
13	18	*Programming time*	T
14	19	*Error estimate*	BLOC
15	20	*Content (intellegence) vocabulary*	I
16	21	*Cyclomatic comple1ity*	v(G)
17	24	*Decision comple1ity*	iv(G)
18	26	*Essential comple1ity*	ev(G)
19	32	*Branch count*	Branch_C
20	2	*LOC bank*	BLOC
21	38	*Parameter count*	Parameter_C
22	40	*Modified condition count*	Mod_Cond_C

4.1.2. Pengisian Nilai yang Absen Menggunakan WkNN

Pengisian nilai yang absen pada fitur ke 20 sampai 22 dilakukan secara bertahap. Proses pengisian nilai yang absen seperti yang telah dijelaskan pada Sub-bab 3.2.2. Rekapitulasi jumlah *missing value* fitur ke 20 sampai 22 disajikan pada Tabel 4.7.

Tabel 4.7 Rekapitulasi Jumlah *Missing Value*

No	Fitur Ke-	Jumlah *Missing Value*
1	20	1406 data
2	21	2802 data
3	22	4198 data

4.1.3. Seleksi Fitur Menggunakan Metode IG, GR, OR, SU dan RFF

Setelah pengisian nilai yang absen, maka tahap selanjutnya yang perlu dilakukan sebelum memasuki tahap uji coba adalah seleksi fitur menggunakan metode IG, GR, OR, RFF, SU. Dua puluh dua fitur dari ketujuh dataset yang terpilih selanjutnya dilakukan seleksi fitur, yang dapat dilihat pada Tabel 4.8. Untuk memperoleh nilai gain pada metode IG menggunakan persamaan (2.1)-(2.2), nilai pada metode GR menggunakan persamaan (2.3)-(2.4), nilai pada metode OR menggunakan aturan yang dijelaskan pada sub bab sebelumnya, nilai pada metode SU menggunakan persamaan (2.6) dan nilai pada RFF menggunakan algoritma yang telah dijelaskan sub bab sebelumnya.

Tabel 4.8 Ranking Fitur

Rank	IG		GR		OR		SU		RFF	
	Nilai	B	Nilai	B	Nilai	B	Nilai	B	Nilai	B
1	0.05867	22	0.05547	11	91.3368	6	0.04217	22	0.0048066	15
2	0.04705	12	0.02489	22	91.3368	2	0.03723	21	0.0041224	21
3	0.04427	21	0.02267	21	91.3368	19	0.03561	12	0.0032466	13
4	0.04024	1	0.02122	12	91.3368	20	0.03332	1	0.0029864	2
5	0.03344	18	0.02065	4	91.3209	3	0.03086	18	0.0024944	12
6	0.03203	5	0.02022	1	91.3209	17	0.02951	4	0.0021414	3
7	0.03107	7	0.0192	18	91.305	13	0.02788	5	0.002079	9
8	0.02897	16	0.01799	10	91.2891	11	0.02773	7	0.0020743	22
9	0.02861	6	0.01711	7	91.2732	4	0.02547	14	0.0020704	1
10	0.02594	14	0.01711	5	91.2732	21	0.02483	10	0.0018632	7
11	0.02488	19	0.01609	14	91.2732	1	0.02179	15	0.0015936	19
12	0.02405	15	0.01349	15	91.2732	18	0.02177	6	0.0014143	20
13	0.02351	20	0.01325	17	91.2415	8	0.0213	16	0.0013097	14
14	0.02197	4	0.01311	20	91.2415	10	0.0212	20	0.0010083	4
15	0.01918	17	0.01299	6	91.2256	7	0.0207	19	0.0009905	11
16	0.01787	3	0.01262	16	91.2256	5	0.02048	17	0.0008051	6
17	0.01704	10	0.01257	19	91.1938	22	0.01801	3	0.0006976	17

Rank	IG		GR		OR		SU		RFF	
	Nilai	B	Nilai	B	Nilai	B	Nilai	B	Nilai	B
18	0.01467	9	0.01146	3	91.1461	15	0.01597	9	0.0004714	5
19	0.01347	2	0.01039	9	91.1302	16	0.01325	2	0.0003704	18
20	0.00885	8	0.00929	8	91.1143	12	0.01285	8	0.0001073	10
21	0.00722	13	0.00837	2	91.1143	9	0.01185	11	0.0001006	8
22	0.00282	11	0.00655	13	91.0984	14	0.00945	13	0.0000977	16

Ket: B = Nomor Fitur

Tahap selanjutnya, untuk setiap metode seleksi fitur, dataset diklasifikasi menggunakan naïve bayes. Jika nilai *balance* pada Rn lebih dari Rn-1 maka masih berlanjut penambahan fitur sampai mendapatkan Rn kurang dari sama dengan Rn-1 maka iterasi berhenti, seperti disajikan pada Tabel 4.9 sampai Tabel 4.13. Untuk memperoleh nilai *balance* menggunakan persamaan (2.17) sampai dengan (2.19).

Tabel 4.9 Fitur Relevan pada IG

	IG					
	R1	R2	R3	R4	R5	R6
	22	R1 +12	R2+21	R3+1	R4+18	R5+5
Nilai *balance*	0.3584	0.418	0.4691	0.4918	**0.4986**	0.4768

R1 adalah satu fitur teratas pada metode IG yaitu fitur ke 22, seperti ditunjukkan pada Tabel 4.8. R1 dihitung nilai *balance*. R2 adalah dua fitur teratas pada metode IG (fitur ke 12) yang digabung dengan R1. Berdasarkan Tabel 4.9 hasil nilai *balance* pada R2 lebih besar dibandingkan R1, maka proses penambahan fitur dilanjutkan sesuai ranking metode IG yang ada pada Tabel 4.8. Pada metode IG, R6 mengalami penurunan nilai *balance.* Maka proses penambahan fitur berhenti sampai R6. Nilai *balance* yang berwana abu-abu artinya nilai *balance* tertinggi di metode IG yaitu R5.

Tabel 4.10 Fitur Relevan pada GR

	GR				
	R1	R2	R3	R4	R5
	11	R1+22	R2+21	R3+12	R4+4
Nilai *balance*	0.3514	0.3891	0.4233	**0.4662**	0.4626

Pada GR, fitur yang memiliki ranking teratas yaitu fitur ke 11, seperti ditunjukkan pada Tabel 4.8. Berdasarkan Tabel 4.10 hasil nilai *balance* pada R2 lebih besar dibandingkan R1, maka proses penambahan fitur dilanjutkan sesuai ranking metode GR yang ada pada Tabel 4.8. Pada metode GR, R5 mengalami penurunan nilai *balance.* Maka proses penambahan fitur berhenti sampai R5. Nilai *balance* yang berwana abu-abu artinya nilai *balance* tertinggi di metode GR yaitu R4.

Tabel 4.11 Fitur Relevan pada OR

	OR				
	R1	R2	R3	R4	R5
	6	R1+2	R2+19	R3+20	R4+3
Nilai *balance*	0.2929	0.3836	0.4012	**0.4042**	0.3976

Pada OR, fitur yang memiliki ranking teratas yaitu fitur ke 6, seperti ditunjukkan pada Tabel 4.8. Berdasarkan Tabel 4.11 hasil nilai *balance* pada R2 lebih besar dibandingkan R1, maka proses penambahan fitur dilanjutkan sesuai ranking metode OR yang ada pada Tabel 4.8. Pada metode OR, R5 mengalami penurunan nilai *balance.* Maka proses penambahan fitur berhenti sampai R5. Nilai *balance* yang berwana abu-abu artinya nilai *balance* tertinggi di metode OR yaitu R4.

Tabel 4.12 Fitur Relevan pada SU

	SU					
	R1	R2	R3	R4	R5	R6
	22	R1+21	R2+12	R3+1	R4+18	R5+4
Nilai *balance*	0.3584	0.4137	0.4691	0.4918	**0.4986**	0.4768

Pada SU, fitur yang memiliki ranking teratas yaitu fitur ke 22, seperti ditunjukkan pada Tabel 4.8. Berdasarkan Tabel 4.12 hasil nilai *balance* pada R2 lebih besar dibandingkan R1, maka proses penambahan fitur dilanjutkan sesuai ranking metode SU yang ada pada Tabel 4.8. Pada metode SU, R6 mengalami penurunan nilai *balance.* Maka proses penambahan fitur berhenti sampai R5. Nilai *balance* yang berwana abu-abu artinya nilai *balance* tertinggi di metode SU yaitu R5.

Tabel 4.13 Fitur Relevan pada RFF

	RFF					
	R1	R2	R3	R4	R5	R6
	15	R1+21	R2+13	R3+2	R4+12	R5+3
Nilai *balance*	0.3184	0.3744	0.3889	0.4471	**0.4924**	0.4811

Pada RFF, fitur yang memiliki ranking teratas yaitu fitur ke 15, seperti ditunjukkan pada Tabel 4.8. Berdasarkan Tabel 4.13 hasil nilai *balance* pada R2 lebih besar dibandingkan R1, maka proses penambahan fitur dilanjutkan sesuai ranking metode RFF yang ada pada Tabel 4.8. Pada metode RFF, R6 mengalami penurunan nilai *balance.* Maka proses penambahan fitur berhenti sampai R5. Nilai *balance* yang berwana abu-abu artinya nilai *balance* tertinggi di metode RFF yaitu R5.

Dari kelima versi pemeringkatan fitur, maka metode IG dan SU memperoleh nilai *balance* yang sama-sama tinggi, 0.4986 dan terdapat keserupaan anggota fitur. Penelitian ini sesuai dengan [27], bahwa seleksi fitur IG memberikan performa terbaik dan dapat meningkatkan nilai *balance*. Menurut [14], persamaan IG, GR, SU memiliki keserupaan, sehingga menghasilkan nilai yang serupa. Kelima ranking (R1, R2, R3, R4 dan R5) pada metode seleksi fitur IG dan SU yang diproses ke tahap pengujian.

4.2. Skenario Pengujian

Setelah terpilih fitur yang relevan berdasarkan pemeringkatan kelima metode seleksi fitur dan nilai *balance*, selanjutnya ditentukan skenario pengujian untuk mengetahui pengaruh usulan metode terhadap *nilai balance*. Terdapat dua step skenario pengujian:

Step 1: fokus pada proses pengisian nilai yang absen, terdiri dari tiga skenario pengujian, yaitu:

1. Rata-rata tetangga terdekat k=2 sampai k=10
2. Rata-rata tetangga terdekat k=2 sampai k=10 dan menggunakan pemeringkatan fitur (*information gain)*

3. Satu tetangga terdekat dan menggunakan pemeringkatan fitur (*information gain)*

Step 2: hasil terbaik dari step 1, terdiri dari dua skenario pengujian, yaitu:

1. Data tidak imbang: menggunakan semua dataset (6293 data), *defect : non-defect* = 545 : 5748
2. Data imbang: menggunakan semua dataset (1090 data), *defect : non-defect* = 545 : 545

Adapun jumlah *fold* untuk melakukan *cross validation* pada pengujian ini adalah 10 *folds*. Skenario pengujian menggunakan 7 dataset NASA *public* MDP yaitu KC3, MW1, CM1, PC1, PC2, PC3 dan PC4. Pengujian dilakukan dengan pengisian nilai yang absen terlebih dahulu.

Alat ukur yang digunakan untuk mengevaluasi hasil prediksi adalah metrik *confussion.* Pada penelitian ini, diukur menggunakan probabilitas prediksi (pd), probabilitas *false alarm* (pf), dan *balance*.

4.2.1. Skenario Step 1

Skenario step satu terdiri dari tiga skenario pengujian seperti yang telah dijelaskan sebelumnya. Untuk masing-masing hasil skenario pengujian dijelaskan pada sub bab di bawah ini.

4.2.1.1. Hasil Skenario 1

Hasil pengujian prediksi kesalahan perangkat lunak untuk skenario satu, menggunakan rata-rata tetangga terdekat untuk mengisi nilai yang absen, nilai k=2 sampai dengan k=10, fitur ke 21 sampai 25 tidak diranking menggunakan IG terlebih dahulu, dan klasifikasi menggunakan Naïve Bayes dapat dilihat pada Tabel 4.14.

Tabel 4.14 Hasil Pengisian Nilai yang Absen pada Skenario 1

A	B	Data Tidak Imbang								
		k=2	k=3	k=4	k=5	k=6	k=7	k=8	k=9	k=10
19		0.4429	0.4429	0.4429	0.4429	0.4429	0.4429	0.4429	0.4429	0.4429
20		0.4460	0.4437	0.4437	0.4459	0.4459	0.4459	0.4459	0.4459	0.4459
21	6	0.4425	0.4424	0.4424	0.4443	0.4446	0.4446	0.4446	0.4526	0.4445
22	7								0.4658	0.4428

A	B	Data Tidak Imbang								
		k=2	k=3	k=4	k=5	k=6	k=7	k=8	k=9	k=10
23	15								0.4659	
24	22								**0.4806**	
25	23								0.4664	

Ket: A= Jumlah fitur, B= Nomor Fitur

Nilai *balance* yang berwana abu-abu artinya nilai *balance* tertinggi di antara k = 2 sampai dengan k = 9. Berdasarkan tabel 4.14, pada skenario satu, dua puluh empat fitur (k=9) diolah ke tahap seleksi fitur yang menggunakan lima metode seleksi fitur (IG, GR, OR, SU dan RFF). Adapun hasil dari kelima metode seleksi fitur seperti yang terdapat di Tabel 4.15 sampai Tabel 4.19.

Tabel 4.15 Fitur Relevan pada IG

	IG					
	R1	R2	R3	R4	R5	R6
	12	R1+22	R2+1	R3+21	R4+24	R5+18
Nilai *balance*	0.3613	0.4144	0.4544	0.4876	**0.4906**	0.4905

Pada IG, fitur yang memiliki ranking teratas yaitu fitur ke 12, seperti ditunjukkan pada Tabel 4.15. Berdasarkan Tabel 4.15 hasil nilai *balance* pada R2 lebih besar dibandingkan R1, maka proses penambahan fitur dilanjutkan. Karena nilai *balance* R6 lebih kecil dari pada nilai *balance* R5 maka iterasi berhenti. Nilai *balance* yang berwana abu-abu artinya nilai *balance* tertinggi di metode IG yaitu R5.

Tabel 4.16 Fitur Relevan pada GR

	GR							
	R1	R2	R3	R4	R5	R6	R7	R8
	fit 11	R1+22	R2+24	R3+21	R4+12	R5+4	R6+ 1	R7+5
Nilai *balance*	0.3514	0.386	0.4315	0.4182	0.4451	0.4643	**0.4736**	0.464

Pada GR, fitur yang memiliki ranking teratas yaitu fitur ke 11, seperti ditunjukkan pada Tabel 4.16. Berdasarkan Tabel 4.16 hasil nilai *balance* pada R2 lebih besar dibandingkan R1, maka proses penambahan fitur dilanjutkan. Karena nilai *balance* R8 lebih kecil dari pada nilai *balance* R7 maka iterasi

berhenti. Nilai *balance* yang berwana abu-abu artinya nilai *balance* tertinggi di metode GR yaitu R7.

Tabel 4.17 Fitur Relevan pada OR

	OR		
	R1	R2	R3
	fit 17	R1+19	R2+23
Nilai *balance*	0.3678	**0.3855**	0.3838

Pada OR, fitur yang memiliki ranking teratas yaitu fitur ke 17, seperti ditunjukkan pada Tabel 4.17. Berdasarkan Tabel 4.17 hasil nilai *balance* pada R2 lebih besar dibandingkan R1, maka proses penambahan fitur dilanjutkan. Karena nilai *balance* R3 lebih kecil dari pada nilai *balance* R2 maka iterasi berhenti. Nilai *balance* yang berwana abu-abu artinya nilai *balance* tertinggi di metode OR yaitu R2.

Tabel 4.18 Fitur Relevan pada SU

	SU					
	R1	R2	R3	R4	R5	R6
	22	R1+12	R2+24	R3+21	R4+1	R5+18
Nilai *balance*	0.3585	0.4144	0.4282	0.4637	**0.4906**	0.4905

Pada SU, fitur yang memiliki ranking teratas yaitu fitur ke 22, seperti ditunjukkan pada Tabel 4.18. Berdasarkan Tabel 4.18 hasil nilai *balance* pada R2 lebih besar dibandingkan R1, maka proses penambahan fitur dilanjutkan. Karena nilai *balance* R6 lebih kecil dari pada nilai *balance* R5 maka iterasi berhenti. Nilai *balance* yang berwana abu-abu artinya nilai *balance* tertinggi di metode SU yaitu R5.

Tabel 4.19 Fitur Relevan pada RFF

	RFF				
	R1	R2	R3	R4	R5
	fit 15	R1+21	R2+13	R3+2	R4+12
Nilai *balance*	0.3184	0.3398	0.3408	**0.3729**	0.3613

Pada RFF, fitur yang memiliki ranking teratas yaitu fitur ke 15, seperti ditunjukkan pada Tabel 4.19. Berdasarkan Tabel 4.19 hasil nilai *balance* pada

R2 lebih besar dibandingkan R1, maka proses penambahan fitur dilanjutkan. Karena nilai *balance* R5 lebih kecil dari pada nilai *balance* R4 maka iterasi berhenti. Nilai *balance* yang berwana abu-abu artinya nilai *balance* tertinggi di metode RFF yaitu R4.

Berdasarkan lima metode seleksi fitur pada skenario 1 menghasilkan performa terbaik pada metode IG dan SU yaitu R5, 0.4906 yang disajikan pada Tabel 4.15 dan Tabel 4.18. Sehingga lima fitur pada metode IG (R1, R2, R3, R4 dan R5) diproses ke tahap pengujian seperti yang disajikan pada Tabel 4.15. Lima fitur pada metode SU (R1, R2, R3, R4 dan R5) juga diproses ke tahap pengujian seperti yang disajikan pada Tabel 4.16. Pada tahap pengujian, kombinasi Naïve Bayes dan IG atau SU memiliki nilai *balance* yang sama yaitu 0.4959.

4.2.1.2. Hasil Skenario 2

Skenario dua, menggunakan rata-rata tetangga terdekat untuk mengisi nilai yang absen, nilai k=2 sampai dengan k=10, fitur ke 21 sampai 24 diranking menggunakan IG terlebih dahulu, dan klasifikasi menggunakan Naïve Bayes dapat dilihat pada Tabel 4.20.

Tabel 4.20 Hasil Pengisian Nilai yang Absen pada Skenario 2

A	B	Data Tidak Imbang								
		k=2	k=3	k=4	k=5	k=6	k=7	k=8	k=9	k=10
19		0.4429	0.4429	0.4429	0.4429	0.4429	0.4429	0.4429	0.4429	0.4429
20		0.446	0.4437	0.4437	0.4459	0.4459	0.4459	0.4459	0.4459	0.4459
21	38	0.4527	0.4527	0.4526	0.4525	0.4526	0.4526	0.4526	0.4526	0.4525
22	40	0.4656	0.4658	0.4659	0.4658	0.4659	0.4658	0.4658	0.4658	0.4658
23	23	0.4713	0.4693	0.4649	0.467	0.4669	0.4713	0.4713	0.4713	0.4713
24	35	**0.4751**	0.4731		0.4731	0.473	0.473	0.473	0.473	0.473
25	22	0.4714	0.4715		0.4714	0.4692	0.471	0.4712	0.4692	0.469

Ket: A= Jumlah fitur, B= Nomor Fitur

Nilai *balance* yang berwana abu-abu artinya nilai *balance* tertinggi di antara k = 2 sampai dengan k = 9. Berdasarkan tabel 4.20, pada skenario dua, dua puluh empat fitur (k=2) diolah ke tahap seleksi fitur yang menggunakan lima

metode seleksi fitur (IG, GR, OR, SU dan RFF). Adapun hasil dari kelima metode seleksi fitur seperti yang terdapat di Tabel 4.21 sampai Tabel 4.25.

Tabel 4.21 Fitur Relevan pada IG

	IG					
	R1	R2	R3	R4	R5	R6
	12	R1+22	R2+24	R3+21	R4+1	R5+23
Nilai *balance*	0.3613	0.414	0.4328	0.4876	**0.5045**	0.4821

Pada IG, fitur yang memiliki ranking teratas yaitu fitur ke 12, seperti ditunjukkan pada Tabel 4.21. Berdasarkan Tabel 4.21 hasil nilai *balance* pada R2 lebih besar dibandingkan R1, maka proses penambahan fitur dilanjutkan. Karena nilai *balance* R6 lebih kecil dari pada nilai *balance* R5 maka iterasi berhenti. Nilai *balance* yang berwana abu-abu artinya nilai *balance* tertinggi di metode IG yaitu R5.

Tabel 4.22 Fitur Relevan pada GR

	GR						
	R1	R2	R3	R4	R5	R6	R7
	fit 11	R1,22	R2,23	R3,24	R4,21	R5,12	R6, 4
Nilai *balance*	0.3514	0.3838	0.4112	0.4182	0.4363	**0.4612**	0.4604

Pada GR, fitur yang memiliki ranking teratas yaitu fitur ke 11, seperti ditunjukkan pada Tabel 4.22. Berdasarkan Tabel 4.22 hasil nilai *balance* pada R2 lebih besar dibandingkan R1, maka proses penambahan fitur dilanjutkan. Karena nilai *balance* R7 lebih kecil dari pada nilai *balance* R6 maka iterasi berhenti. Nilai *balance* yang berwana abu-abu artinya nilai *balance* tertinggi di metode GR yaitu R5.

Tabel 4.23 Fitur Relevan pada OR

	OR	
	R1	R2
	fit 17	R1,6
Nilai *balance*	**0.3678**	0.3678

Pada OR, fitur yang memiliki ranking teratas yaitu fitur ke 17, seperti ditunjukkan pada Tabel 4.23. Berdasarkan Tabel 4.23 hasil nilai *balance* pada

R2 lebih besar dibandingkan R1, maka proses penambahan fitur dilanjutkan. Karena nilai *balance* R2 sama dengan nilai *balance* R6 maka iterasi berhenti. Nilai *balance* yang berwana abu-abu artinya nilai *balance* tertinggi di metode OR yaitu R1.

Tabel 4.24 Fitur Relevan pada SU

	SU							
	R1	R2	R3	R4	R5	R6	R7	R8
	fit 22	R1,23	R2,24	R3,21	R4,12	R5,1	R6, 18	R7, 4
Nilai *balance*	0.3584	0.3713	0.3848	0.4187	0.4433	0.4821	**0.486**	0.4808

Pada SU, fitur yang memiliki ranking teratas yaitu fitur ke 22, seperti ditunjukkan pada Tabel 4.24. Berdasarkan Tabel 4.24 hasil nilai *balance* pada R2 lebih besar dibandingkan R1, maka proses penambahan fitur dilanjutkan. Karena nilai *balance* R8 kurang dari nilai *balance* R7 maka iterasi berhenti. Nilai *balance* yang berwana abu-abu artinya nilai *balance* tertinggi di metode SU yaitu R7.

Tabel 4.25 Fitur Relevan pada RFF

	RFF					
	R1	R2	R3	R4	R5	R6
	fit 15	R1,21	R2,13	R3,2	R4,12	R5,9
Nilai *balance*	0.3184	0.3909	0.3911	0.4496	**0.4923**	0.4877

Pada RFF, fitur yang memiliki ranking teratas yaitu fitur ke 15, seperti ditunjukkan pada Tabel 4.25. Berdasarkan Tabel 4.25 hasil nilai *balance* pada R2 lebih besar dibandingkan R1, maka proses penambahan fitur dilanjutkan. Karena nilai *balance* R6 kurang dari nilai *balance* R5 maka iterasi berhenti. Nilai *balance* yang berwana abu-abu artinya nilai *balance* tertinggi di metode RFF yaitu R5.

Berdasarkan lima metode seleksi fitur pada skenario dua menghasilkan performa terbaik pada metode IG yaitu R5, 0.5045 yang disajikan pada Tabel 4.21. Sehingga lima fitur pada metode IG (R1, R2, R3, R4 dan R5) diproses ke tahap pengujian seperti yang disajikan pada Tabel 4.21. Pada tahap pengujian, kombinasi IG dengan Naïve Bayes memberikan performa terbaik yaitu 0.4762.

4.2.1.3. Hasil Skenario 3

Pada skenario 3, menggunakan satu tetangga terdekat, yaitu satu nilai yang memiliki jarak paling rendah untuk mengisi *missing value*. Fitur ke 21 dan seterusnya diranking menggunakan IG terlebih dahulu. Klasifikasi menggunakan Naïve Bayes yang disajikan pada Tabel 4.26. Pada skenario tiga ini menggunakan satu tetangga terdekat saat pengisian nilai yang absen, karena berdasarkan percobaan yang telah dilakukan memberikan performa yang lebih baik dibandingkan menggunakan rata-rata tetangga terdekat.

Tabel 4.26 Hasil Pengisian Nilai yang Absen Skenario 3

	Data Tidak Imbang
Jumlah Fitur	Satu Tetangga Terdekat
19	0.4429
20	0.4437
21	0.4526
22	**0.4658**
23	0.4654

Berdasarkan Tabel 4.26 dua puluh dua fitur diproses ke tahap selanjutnya yaitu seleksi fitur. Nilai *balance* yang berwarna abu-abu merupakan nilai *balance* tertinggi. Proses seleksi fitur menggunakan metode IG, GR, OR, SU dan RFF seperti pada Tabel 4.9 sampai Tabel 4.13. Berdasarkan kelima metode seleksi fitur, IG dan SU memberikan performa nilai *balance* yang baik dan memiliki keserupaan anggota fitur antara metode IG dan SU, seperti yang terlihat pada Tabel 4.27 dan Tabel 4.28.

Tabel 4.27 Fitur Relevan pada IG

	IG					
	R1	R2	R3	R4	R5	R6
	22	R1+12	R2+21	R3+1	R4+18	R5+5
Nilai *balance*	0.3584	0.418	0.4691	0.4918	**0.4986**	0.4768

Pada IG, fitur yang memiliki ranking teratas yaitu fitur ke 22, seperti ditunjukkan pada Tabel 4.27. Berdasarkan Tabel 4.27 hasil nilai *balance* pada R2 lebih besar dibandingkan R1, maka proses penambahan fitur dilanjutkan. Karena nilai *balance* R6 kurang dari nilai *balance* R5 maka iterasi berhenti. Nilai

balance yang berwana abu-abu artinya nilai *balance* tertinggi di metode IG yaitu R5.

Tabel 4.28 Fitur Relevan pada SU

	SU					
	R1	R2	R3	R4	R5	R6
	22	R1+21	R2+12	R3+1	R4+18	R5+4
Nilai *balance*	0.3584	0.4137	0.4691	0.4918	**0.4986**	0.4768

Pada SU, fitur yang memiliki ranking teratas yaitu fitur ke 22, seperti ditunjukkan pada Tabel 4.28. Berdasarkan Tabel 4.28 hasil nilai *balance* pada R2 lebih besar dibandingkan R1, maka proses penambahan fitur dilanjutkan. Karena nilai *balance* R6 kurang dari nilai *balance* R5 maka iterasi berhenti. Nilai *balance* yang berwana abu-abu artinya nilai *balance* tertinggi di metode IG yaitu R5.

Berdasarkan lima metode seleksi fitur pada skenario tiga menghasilkan performa terbaik pada metode IG dan SU yaitu R5, 0.4986 yang disajikan pada Tabel 4.27 dan Tabel 4.28. Sehingga lima fitur pada metode IG (R1, R2, R3, R4 dan R5) diproses ke tahap pengujian seperti yang disajikan pada Tabel 4.27. Lima fitur pada metode SU (R1, R2, R3, R4 dan R5) juga diproses ke tahap pengujian seperti yang disajikan pada Tabel 4.28. Pada tahap pengujian, kombinasi Naïve Bayes dan IG atau SU memiliki nilai *balance* yang sama yaitu 0.4975.

Tahap pengujian menggunakan *tool* menghasilkan nilai *balance*, 0.4975. Tahap pengujian secara manual, dengan membagi 6293 data menjadi 10 *fold*, menghasilkan nilai *balance* 0.4893, seperti yang terdapat pada Tabel 4.29.

Tabel 4.29 Hasil Tahap Pengujian Membagi Data Manual

Fold	Nilai *balance*
1	0.2413
2	0.5483
3	0.6481
4	0.6093
5	0.4714
6	0.4751
7	0.5047

Fold	Nilai *balance*
8	0.5389
9	0.4864
10	0.3693

Berdasarkan Tabel 4.29, hasil rata-rata nilai *balance* 10 *fold* adalah 0.4893. Terdapat perbedaaan hasil nilai *balance* antara menggunakan *tool* dengan manual yaitu 0.4975 dengan 0.4893. Hal ini terjadi karena jika menggunakan *tool* pemilihan data secara random.

4.2.2. Skenario Step 2

Untuk skenario pengujian pada step dua menggunakan hasil terbaik dari step satu yaitu satu tetangga terdekat serta fitur ke 21 dan seterusnya diranking menggunakan IG terlebih dahulu. Kemudian membandingkan performa menggunakan random forest dan naïve bayes.

4.2.2.1. Hasil Skenario 1

Hasil skenario satu pada step dua menggunakan data tidak imbang yaitu kombinasi IG dan naïve bayes atau SU dan naïve bayes memberikan hasil terbaik dengan nilai *balance* 0.4975. Data tidak imbang, naïve bayes memperoleh performa lebih unggul dibandingkan dengan *random forest.*

4.2.2.2. Hasil Skenario 2

Pada skenario dua yaitu melakukan penyeimbangan data dengan mengurangi kelas mayoritas yaitu kelas *non-defect* hingga jumlah data sama dengan jumlah data kelas minoritas yaitu kelas *defect.* Namun untuk data imbang yang terdiri dari kelas *defect* 545 data dan *non-defect* 545 data, metode klasifikasi *random forest* lebih unggul dibandingkan naïve bayes, seperti yang terlihat pada Tabel 4.30. Menurut [7] untuk data imbang, *random forest* memiliki performa yang lebih baik dibandingkan teknik klasifikasi yang lain. Dalam penelitian ini, penulis melakukan hal yang serupa dengan menyeimbangkan data dengan mengurangi kelas mayoritas, *non-defect*.

Tabel 4.30 Hasil Nilai *Balance* Pengisian *Missing Value* Data Imbang

A	Data Imbang	
	Naïve Bayes	Random Forest
19	0.5325	**0.77075**
20	0.526	0.76903

Ket: A= Jumlah fitur

Pada data imbang 19 fitur terpilih untuk ke proses seleksi fitur menggunakan lima metode seperti yang disajikan pada Tabel 4.31 sampai Tabel 4.35. Nilai *balance* yang berwarna abu-abu merupakan nilai *balance* tertinggi pada teknik klasifikasi Random Forest.

Tabel 4.31 Fitur Relevan pada IG

	IG										
	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11
	fit 3	R1+1	R2+16	R3+18	R4+6	R5+12	R6+19	R7+5	R8+7	R9+15	R10+4
Nilai balance	0.6845	0.6975	0.7306	0.7431	0.7441	0.7481	0.7509	0.7511	0.7626	**0.7701**	0.7698

Pada IG, fitur yang memiliki ranking teratas yaitu fitur ke 3, seperti ditunjukkan pada Tabel 4.31. Berdasarkan Tabel 4.31 hasil nilai *balance* pada R2 lebih besar dibandingkan R1, maka proses penambahan fitur dilanjutkan. Karena nilai *balance* R11 kurang dari nilai *balance* R10 maka iterasi berhenti. Nilai *balance* yang berwana abu-abu artinya nilai *balance* tertinggi di metode IG yaitu R10.

Tabel 4.32 Fitur Relevan pada GR

	GR							
	R1	R2	R3	R4	R5	R6	R7	R8
	7	R1+10	R2+3	R3+1	R4+12	R5+15	R6+6	R7+14
Nilai *balance*	0.6263	0.7019	0.7164	0.7449	0.7581	0.7807	**0.7795**	0.7700

Pada GR, fitur yang memiliki ranking teratas yaitu fitur ke 7, seperti ditunjukkan pada Tabel 4.32. Berdasarkan Tabel 4.32 hasil nilai *balance* pada R2 lebih besar dibandingkan R1, maka proses penambahan fitur dilanjutkan. Karena nilai *balance* R8 kurang dari nilai *balance* R7 maka iterasi berhenti. Nilai *balance* yang berwana abu-abu artinya nilai *balance* tertinggi di metode GR yaitu R7.

Tabel 4.33 Fitur Relevan pada OR

	OR					
	R1	R2	R3	R4	R5	R6
	fit 6	R1+3	R2+7	R3+1	R4+16	R5+12
Nilai *balance*	0.68629	0.72409	0.74368	0.76059	**0.76217**	0.7538

Pada OR, fitur yang memiliki ranking teratas yaitu fitur ke 6, seperti ditunjukkan pada Tabel 4.33. Berdasarkan Tabel 4.33 hasil nilai *balance* pada R2 lebih besar dibandingkan R1, maka proses penambahan fitur dilanjutkan. Karena nilai *balance* R6 kurang dari nilai *balance* R5 maka iterasi berhenti. Nilai *balance* yang berwana abu-abu artinya nilai *balance* tertinggi di metode OR yaitu R5.

Tabel 4.34 Fitur Relevan pada SU

	SU			
	R1	R2	R3	R4
	fit 3	R1+1	R2+7	R3+12
Nilai *balance*	0.68452	0.69754	**0.76257**	0.7502

Pada SU, fitur yang memiliki ranking teratas yaitu fitur ke 3, seperti ditunjukkan pada Tabel 4.34. Berdasarkan Tabel 4.34 hasil nilai *balance* pada R2 lebih besar dibandingkan R1, maka proses penambahan fitur dilanjutkan. Karena nilai *balance* R4 kurang dari nilai *balance* R3 maka iterasi berhenti. Nilai *balance* yang berwana abu-abu artinya nilai *balance* tertinggi di metode SU yaitu R3.

Tabel 4.35 Fitur Relevan pada RFF

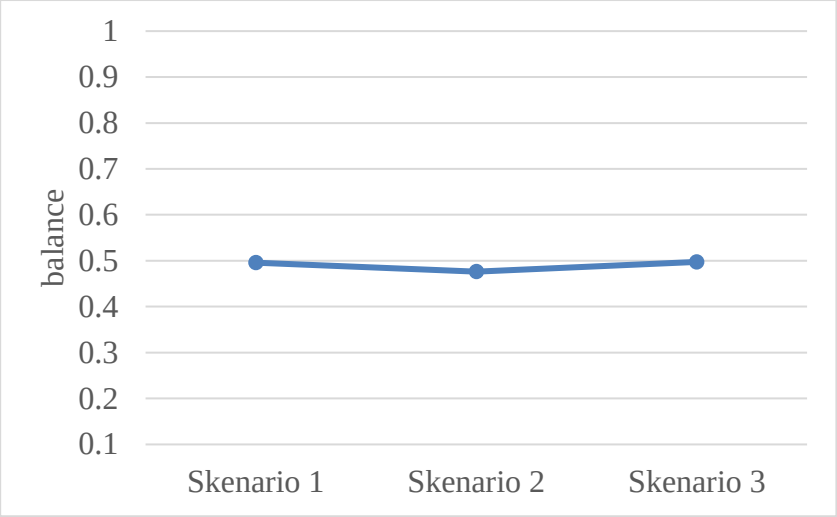
	RFF	
	R1	R2
	fit 15	R1+19
Nilai *balance*	**0.64421**	0.60791

Pada RFF, fitur yang memiliki ranking teratas yaitu fitur ke 15, seperti ditunjukkan pada Tabel 4.35. Berdasarkan Tabel 4.35 hasil nilai *balance* pada R2 lebih kecil dibandingkan R1, maka iterasi berhenti. Nilai *balance* yang berwana abu-abu artinya nilai *balance* tertinggi di metode RFF yaitu R1.

Berdasarkan lima metode seleksi fitur pada skenario dua menghasilkan performa terbaik pada metode GR yaitu R5, 0.7795 yang disajikan pada Tabel 4.33. Sehingga tujuh fitur pada metode IG (R1, R2, R3, R4, R5, R6 dan R7) diproses ke tahap pengujian Pada tahap pengujian, kombinasi Random Forest dan GR memperoleh nilai *balance* terbaik yaitu 0.7795.

4.3. Analisis Hasil

Berdasarkan hasil ketiga skenario pada step satu, yang berfokus untuk pengisian nilai yang absen, nilai *balance* terbaik yaitu skenario tiga, seperti pada Gambar 4.1.

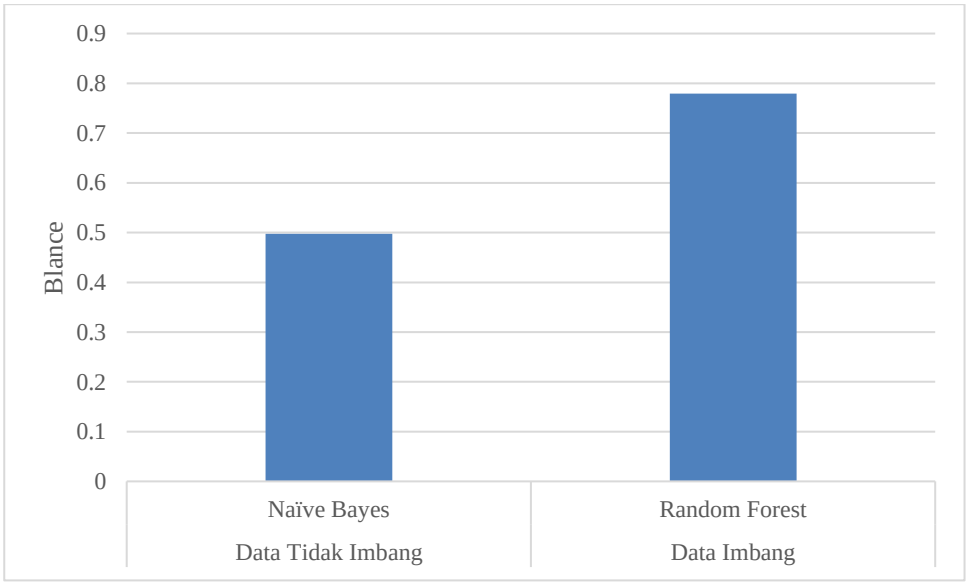


Gambar 4.1 Hasil Nilai *Balance* Tiga Skenario pada Step Satu

Berdasarkan Gambar 4.1, skenario satu memperoleh nilai *balance* 0.4959, skenario 2 memperoleh nilai *balance* 0.4762 dan skenario 3, 0.4975. Berdasarkan hasil uji coba, peningkatan nilai *balance* pada metode naïve bayes dapat dicapai dengan melakukan perhitungan satu tetangga terdekat untuk mengisi nilai *missing value* dan jika terdapat beberapa fitur yang memiliki *jumlah missing value* sama maka diranking terlebih dahulu menggunakan *information gain*, untuk menentukan fitur yang *missing value* mana yang akan diolah terlebih dahulu menggunakan *weighted* KNN. Hal ini terjadi karena jika menggunakan satu tetangga terdekat dapat mewakili kedekatan hubungan *instance* yang terdapat nilai yang absen dengan *instance* yang tidak terdapat nilai

yang absen. Namun jika menggunakan rata-tata tetangga terdekat mengakibatkan interval nilai yang terlalu jauh sehingga tidak mewakili kedekatan hubungan *instance* yang terdapat nilai yang absen dengan *instance* yang tidak terdapat nilai yang absen.

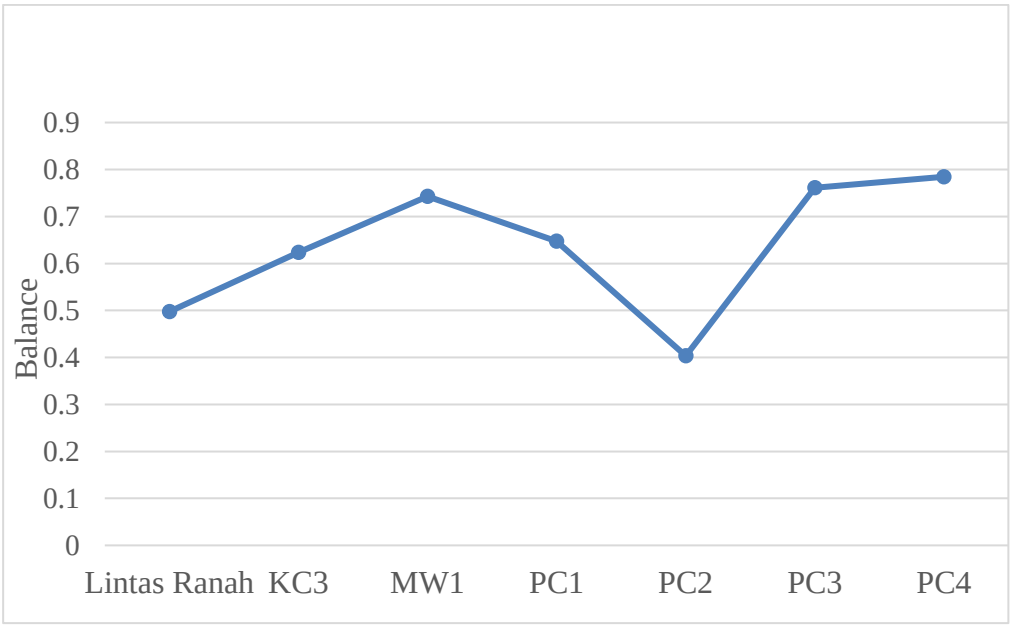
Berdasarkan hasil skenario step dua, yaitu hasil terbaik dari step satu diperoleh hasil seperti pada Gambar 4.2.



Gambar 4.2 Nilai *Balance* Data Tidak Imbang dan Data Imbang

Berdasarkan Gambar 4.2 pada data tidak imbang nilai *balance* tertinggi menggunakan metode klasifikasi naïve bayes yaitu 0.4975 dan data imbang nilai *balance* tertinggi menggunakan metode klasifikasi random forest yaitu 0.7795. Hal ini sesuai dengan [6], teknik klasifikasi naïve bayes di data tidak imbang memberikan nilai *balance* terbaik dibandingkan teknik klasifikasi lain dan pada penelitian [7], teknik klasifikasi random forest yang menggunakan data NASA *public* MDP yang sudah seimbang (*balance*) memiliki performa yang lebih baik dari pada data primer (data yang belum seimbang).

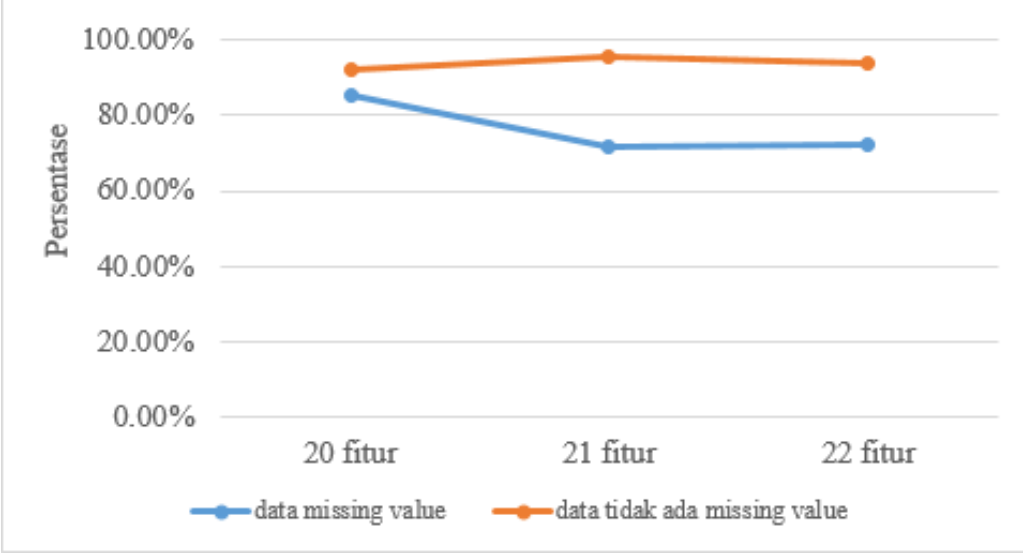
Untuk memberikan penjelasan lebih rinci tentang kinerja metode yang diusulkan, kami melakukan analisis lebih lanjut, seperti pada Gambar 4.3 sampai Gambar 4.5. Gambar 4.3 menunjukkan analisis data dengan membandingkan kinerja metode pada domain lintas ranah dan domain spesifik.



Gambar 4.3 Nilai *Balance* Dataset

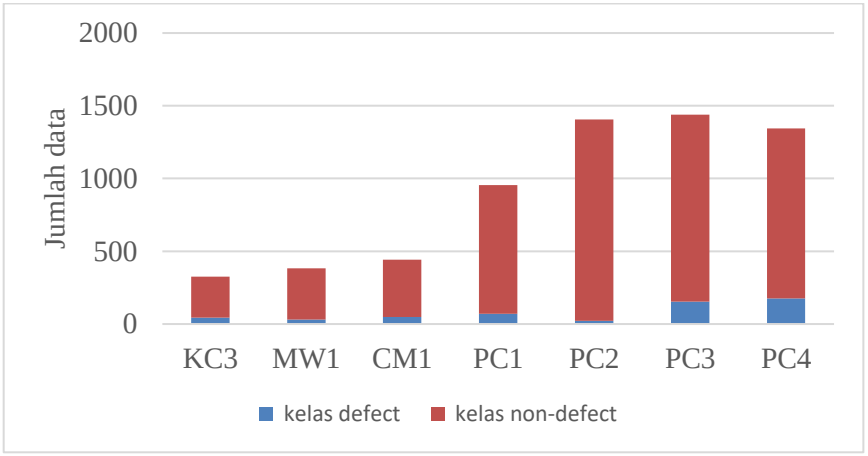
Berdasarkan Gambar 4.3. data lintas ranah yaitu data gabungan dari 7 dataset NASA *public* MDP lebih unggul dibandingkan data spesifik, PC2. Data lintas ranah memperoleh nilai *balance* 0.4975 dan pada PC2 0.4033. Hal ini terjadi karena pada data PC2 memiliki jumlah *missing value* yang banyak yaitu 1406 data. Cara pengisian *missing value* dapat meningkatkan performa klasifikasi. Untuk pengolahan data spesifik disini setiap dataset langsung dilakukan pemeringkatan fitur menggunakan lima metode seleksi fitur, kemudian dilakukan uji coba. KC3 memperoleh nilai *balance* 0.6233, MW1 0.7430, CM1 0.6075, PC1 0.6474, PC2 0.4033, PC3 0.7611, dan PC4 0.784.

Pada poin misklasifikasi, misklasifikasi pada *missing value* lebih rendah dibandingkan misklasifikasi pada data yang tidak memiliki *missing value*, seperti pada Gambar 4.4.



Gambar 4.4 Perbandingan Misklasifikasi

Berdasarkan Gambar 4.4, misklasifikasi antara data *missing value* dengan data yang tidak memiliki *missing value* pada setiap fiturnya, data *missing value* memiliki nilai misklasifikasi lebih rendah dari pada data yang tidak memiliki *missing value*. Pada 20 fitur data *missing value* 85,36% dan data tidak ada *missing value* 91,96%. Pada 21 fitur data *missing value* 71,82% dan data tidak ada *missing value* 95.51%. Pada 22 fitur data *missing value* 72,35% dan data tidak ada *missing value* 94,10%.



Gambar 4.5 Perbandingan Kelas *Defect* dan *Non-Defect* Tiap Dataset

Data lintas ranah cenderung lebih rendah dibandingkan dengan data spesifik, karena setiap dataset memiliki ketidakseimbangan kelas.

Ketidakseimbangan kelas terbesar terdapat pada dataset PC2 yang memiliki rasio data defect dan non-defect sebesar 1:60 seperti pada Gambar 4.5. Ketika menggabungkan tujuh dataset, ketidakseimbangan kelas mengalami peningkatan sehingga berpengaruh terhadap performa model klasifikasi perangkat lunak.

Metode pengisian *missing value*, berdasarkan penelitian [18], menggunakan nilai k=5 sampai k=10 memberikan nilai *Normalized Root Mean Suare Error* (NRMSE) yang relatif lebih rendah. Hal ini sesuai dengan penelitian ini pada tahap pengujian skenario satu pada step satu yaitu pada k=9 memberikan nilai *balance* tertinggi. Pada saat k=7 menghasilkan nilai NRMSE paling rendah pada dataset *Trash Pickup Logistics Management System* (TPLMS) yang merupakan performa terbaik. Namun pada penelitian ini, metode pengisian *missing value* menggunakan satu tetangga terdekat memberikan performa terbaik.

[Halaman ini sengaja dikosongkan]

BAB 5
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Dari hasil penelitian yang telah dilakukan, maka dapat diambil kesimpulan sebagai berikut:

1. Metode *weighted* KNN dapat menangani masalah untuk mengisi fitur yang kosong (nilai yang absen) pada lintas ranah. Penurunan prosentase misklasifikasi terbaik antara data *missing value* dengan data tidak *missing value* terdapat pada fitur ke 21 sebesar 23.69%. Dengan menggunakan satu tetangga terdekat pada proses pengisiannya memberikan performa yang lebih baik dari pada menggunakan rata-rata tetangga terdekat, nilai k=2 sampai dengan k=10. Data lintas ranah memperoleh nilai *balance* lebih baik dibandingkan data spesifik PC2 yaitu 0,4975 dibanding 0,4033.
2. Hasil pengujian menunjukkan bahwa data tidak imbang (*imbalance*) menghasilkan nilai *balance* terbaik jika metode naive bayes dikombinasi dengan metode seleksi fitur *information gain* (IG) atau *symmetric uncertainty* (SU), yaitu 0.4975. Hasil pengujian juga menunjukkan bahwa data imbang (*balance*) menghasilkan nilai *balance* terbaik jika metode *random forest* dikombinasi dengan metode seleksi fitur *gain ratio* (GR), yaitu 0.7795. Seleksi fitur menggunakan pendekatan filter memungkinkan pemeringkatan fitur berdasarkan tingkat relevansinya terhadap label kelas.

5.2. Saran

Berdasarkan hasil pengujian, untuk proses pengisian *missing value* perlu dicari metode lain yang memberikan performa lebih baik dibandingkan semua data spesifik tidak hanya PC2. Metode yang memberikan performa misklasifikasi dalam jumlah sedikit. Metode seleksi fitur dengan pendekatan wrapper juga perlu diteliti lebih lanjut untuk mengoptimalkan hasil kombinasi fitur yang akan digunakan.

[Halaman ini sengaja dikosongkan]

DAFTAR PUSTAKA

[1] P. He, B. Li, X. Liu, J. Chen, and Y. Ma, "An Empirical Study on Software Defect Prediction with a Simplified Metric Set," vol. 59, pp. 170–190, 2014.

[2] D. P. Ii, P. Magister, B. Keahlian, R. Perangkat, J. T. Informatika, and F. T. Informasi, *Perbaikan Prediksi Kesalahan Perangkat Lunak Menggunakan Seleksi Fitur Dan Cluster-Based*. 2017.

[3] I. H. Laradji, M. Alshayeb, and L. Ghouti, "Software defect prediction using ensemble learning on selected features," *Inf. Softw. Technol.*, vol. 58, pp. 388–402, 2015.

[4] G. Czibula, Z. Marian, and I. G. Czibula, "Software defect prediction using relational association rule mining," *Inf. Sci. (Ny).*, vol. 264, pp. 260–278, 2014.

[5] D. P. P. Mesquita, L. S. Rocha, J. Paulo, P. Gomes, and A. R. Rocha, "Classification with reject option for software defect prediction," vol. 49, pp. 1085–1093, 2016.

[6] N. Maisarah, M. Sobran, A. Ahmad, and Z. Ibrahim, "CLASSIFICATION OF IMBALANCED DATASET USING CONVENTIONAL NAÏVE BAYES CLASSIFIER," no. November, pp. 25–26, 2013.

[7] K. Bashir, T. Li, and C. Wondaferaw, "Enhancing Software Defect Prediction Using Supervised-Learning Based Framework," 2017.

[8] D. Ryu and J. Baik, "Effective multi-objective na??ve Bayes learning for cross-project defect prediction," *Appl. Soft Comput. J.*, vol. 49, pp. 1062–1077, 2016.

[9] S. Sen, M. N. Das, and R. Chatterjee, "A Weighted kNN approach to estimate missing values," *3rd Int. Conf. Signal Process. Integr. Networks, SPIN 2016*, pp. 210–214, 2016.

[10] N. Bhatia and C. Author, "Survey of Nearest Neighbor Techniques," *IJCSIS) Int. J. Comput. Sci. Inf. Secur.*, vol. 8, no. 2, pp. 302–305, 2010.

[11] H. Zhang and X. Zhang, "Comments on 'Data mining static code attributes to learn defect predictors,'" *IEEE Trans. Softw. Eng.*, vol. 33, no. 9, pp. 635–636, 2007.

[12] F. Pralienka, B. Muhamad, D. O. Siahaan, and C. Fatichah, "Perbaikan Prediksi Kesalahan Perangkat Lunak Menggunakan Seleksi Fitur dan Cluster-Based Classification," *J. Nas. Tek. Elektro dan Teknol. Inf.*, vol. 6, no. 3, pp. 275–283, 2017.

[13] S. A. Putri and Frieyadie, "Combining integreted sampling technique with feature selection for software defect prediction," *2017 5th Int. Conf. Cyber IT Serv. Manag. CITSM 2017*, pp. 1–6, 2017.

[14] Y. H. Wang and I. C. Wu, "Achieving high and consistent rendering performance of java AWT/Swing on multiple platforms," *Softw. - Pract. Exp.*, vol. 39, no. 7, pp. 701–736, 2009.

[15] J. Novakovic, "The Impact of Feature Selection on the Accuracy of Naive Bayes Classifier," *18th Telecommun. forum TELFOR*, vol. 2, pp. 1113–1116, 2010.

[16] F. Yang, W. Cheng, R. Dou, and N. Zhou, "An improved feature selection approach based on ReliefF and Mutual Information," *Int. Conf. Inf. Sci. Technol.*, vol. 8, pp. 246–250, 2011.

[17] B. Han, S. Xiao, L. Liu, and Z. Wu, "A new method for filling missing values by gray relational analysis," *2011 2nd Int. Conf. Artif. Intell. Manag. Sci. Electron. Commer. AIMSEC 2011 - Proc.*, pp. 2721–2724, 2011.

[18] M. Zhu and X. Cheng, "Iterative KNN imputation based on GRA for missing values in TPLMS," *Proc. 2015 4th Int. Conf. Comput. Sci. Netw. Technol. ICCSNT 2015*, no. Iccsnt, pp. 94–99, 2016.

[19] N. Senthilkumar, T. Tamizharasan, and V. Anandakrishnan, "Experimental investigation and performance analysis of cemented carbide inserts of different geometries using Taguchi based grey relational analysis," *Meas. J. Int. Meas. Confed.*, vol. 58, pp. 520–536, 2014.

[20] K. Vatansever and Y. Akgűl, "Performance evaluation of websites using entropy and grey relational analysis methods: The case of airline

companies," *Decis. Sci. Lett.*, vol. 7, pp. 119–130, 2018.

[21] Ö. F. Arar and K. Ayan, "A feature dependent Naive Bayes approach and its application to the software defect prediction problem," *Appl. Soft Comput. J.*, vol. 59, pp. 197–209, 2017.

[22] R. Malhotra, "A systematic review of machine learning techniques for software fault prediction," *Appl. Soft Comput. J.*, vol. 27, pp. 504–518, 2015.

[23] I. H. Pq, "Uˆ½†˜w²dxrv*f*„uxw$u § w"de*f*„p"rvª„*f*·“”gU*f*„ v«"□¨dedxsvp¯rU¶ x u „ *f*."

[24] A. Chaudhary, S. Kolhe, and R. Kamal, "An improved random forest classifier for multi-class classification," *Inf. Process. Agric.*, vol. 3, no. 4, pp. 215–222, 2016.

[25] I. Engineering, "Water Bloom Warning Model Based on Random," pp. 45–48, 2017.

[26] A. Rohani, M. Taki, and M. Abdollahpour, "A novel soft computing model (Gaussian process regression with K-fold cross validation) for daily and monthly solar radiation forecasting (Part : I)," *Renew. Energy*, vol. 115, pp. 411–422, 2018.

[27] F. P. B. Muhamad, D. O. Siahaan, and C. Fatichah, "Software Fault Prediction Using Filtering Feature Selection in Cluster-Based Classification," *IPTEK J. Proc. Ser.*, vol. 4, no. 1, p. 59, 2018.