Perbaikan Prediksi Kesalahan Perangkat Lunak Menggunakan Seleksi Fitur dan *Cluster-Based Classification*

Fachrul Pralienka Bani Muhamad[1], Daniel Oranova Siahaan[2], Chastine Fatichah[3]

***Abstract*— High balance value of software fault prediction can help in conducting test effort, saving test costs, saving test resources, and improving software quality. Balance values in software fault prediction need to be considered, as in most cases, the class distribution of true and false in the software fault data set tends to be unbalanced. The balance value is obtained from trade-off between probability detection (pd) and probability false alarm (pf). Previous researchers had proposed Cluster-Based Classification (CBC) method which was integrated with Entropy-Based Discretization (EBD). However, predictive models with irrelevant and redundant features in data sets can decrease balance value. This study proposes improvement of software fault prediction outcomes on CBC by integrating feature selection methods. Some feature selection methods are integrated with CBC, i.e. Information Gain (IG), Gain Ration (GR), One-R (OR), Relief-F (RFF), and Symmetric Uncertainty (SU). The result shows that combination of CBC with IG gives best average balance value, compared to other feature selection methods used in this research. Using five NASA public MDP data sets, the combination of IG and CBC generates 63.91% average of balance, while CBC method without feature selection produce 54.79% average of balance. It shows that IG can increase CBC balance average by 9.12%.**

***Intisari*— Tingginya nilai *balance* pada prediksi kesalahan perangkat lunak dapat membantu usaha pengujian, menghemat biaya pengujian, menghemat sumber daya pengujian, dan meningkatkan kualitas perangkat lunak. Nilai *balance* pada hasil prediksi kesalahan perangkat lunak perlu menjadi perhatian, karena pada umumnya, persebaran kelas *true* dan kelas *false* pada *data set* kesalahan perangkat lunak cenderung tidak seimbang. Nilai *balance* diperoleh dari hasil tarik-ulur *(trade-off)* antara nilai *probability detection* (*pd*) dan *probability false alarm* (*pf*). Peneliti sebelumnya mengusulkan metode *Cluster-Based Classification* (CBC) yang diintegrasikan dengan *Entropy-Based Discretization* (EBD). Namun, model prediksi dengan fitur yang redundan dan tidak relevan pada *data set* dapat menurunkan nilai *balance*. Dalam makalah ini diusulkan peningkatan hasil prediksi kesalahan perangkat lunak pada metode CBC dengan mengintegrasikan metode seleksi fitur. Adapun metode seleksi fitur akan diintegrasikan dengan CBC yaitu Perolehan Informasi (IG), Rasio Perolehan (GR), *One-R* (OR), *Relief-F* (RFF), dan *Symmetric Uncertainty* (SU). Hasil menunjukkan bahwa kombinasi CBC dengan IG menghasilkan nilai rata-rata *balance* terbaik, dibandingkan dengan keempat metode seleksi fitur lainnya. Kombinasi metode IG dan CBC menghasilkan 63,91% nilai rata-rata *balance*, sedangkan metode CBC tanpa seleksi fitur menghasilkan 54,79% nilai rata-rata *balance*. Jadi, IG dapat meningkatkan nilai rata-rata *balance CBC* sebesar 9,12%.**

***Kata Kunci*— *Cluster-based Classification*, *Entropy-based Discretization*, kesalahan perangkat lunak, seleksi fitur.**

I. PENDAHULUAN

Prediksi kesalahan perangkat lunak sangat penting untuk dilakukan, karena alokasi biaya dan sumber daya pengujian terbatas. Dengan prediksi kesalahan perangkat lunak, proses pengujian dapat difokuskan pada modul-modul yang rawan terhadap kesalahan, sehingga sumber daya pengujian dapat dihemat untuk modul lain yang rentan salah. Kegiatan prediksi kesalahan perangkat lunak adalah kegiatan penjamin kualitas yang paling efisien, karena dapat menghemat sumber daya pengujian. Berdasarkan penelitian sebelumnya, perangkat lunak yang baik adalah perangkat lunak yang mempunyai jumlah kemunculan kesalahan yang minimal [1].

Para peneliti sebelumnya telah mengusulkan beberapa metode *data mining* untuk melakukan prediksi kesalahan perangkat lunak antara lain *Genetic Programming*, *Decision Tree*, *Neural Network*, *Density-based Clustering*, *Case-based Reasoning*, *Fuzzy Logic*, *Logistic Regression*, dan *Naive Bayes* (NB) [2]. Berdasarkan hasil penelitian, secara umum metode yang menghasilkan nilai prediksi yang tinggi cenderung menggunakan teknik pemodelan yang sederhana, yaitu [3]. Metode NB dapat memprediksi kesalahan perangkat lunak pada *data set NASA public MDP* dengan 71% *probability detection* (*pd*) dan 25% *probability false alarm* (*pf*) [4]. Namun, menurut penelitian sebelumnya, jika dibandingkan dengan metode NB di *data set* yang sama, metode *cluster-based classification* (CBC) dapat menghasilkan nilai prediksi yang lebih baik, yaitu 83% *pd* dan 40% *pf* [2].

Salah satu faktor penyebab rendahnya kualitas perangkat lunak adalah nilai *balance* prediksi kesalahan. Beberapa peneliti telah memaparkan bahwa model prediksi dengan fitur yang tidak relevan serta *redundant* pada *data set* dapat menurunkan nilai *balance* prediksi. Nilai *balance* adalah nilai tarik ulur *(trade-off)* antara nilai *pd* dan *pf*. Semakin nilai *pd* mendekati 100% dan nilai *pf* mendekati 0%, semakin tinggi pula nilai *balance* hasil prediksi. Pada makalah ini, proses seleksi fitur dilakukan untuk menghilangkan fitur yang berlebihan dan tidak relevan agar nilai *balance* prediksi meningkat.

Pada penelitian sebelumnya, metode seleksi fitur dimanfaatkan untuk meningkatkan nilai prediksi kesalahan

[1]*Mahasiswa, Jurusan Teknik Informatika Institut Teknologi Sepuluh Nopember, Kampus ITS Sukolilo Surabaya 60111 INDONESIA (tlp: 031-5939214; fax: 031-5939363; e-mail: fachrul15@mhs.if.its.ac.id)*

[2]*Dosen, Jurusan Teknik Informatika Institut Teknologi Sepuluh Nopember, Kampus ITS Sukolilo Surabaya 60111 INDONESIA (tlp: 031-5939214; fax: 031-5939363; e-mail: daniel@if.its.ac.id)*

[3]*Dosen, Jurusan Teknik Informatika Institut Teknologi Sepuluh Nopember, Kampus ITS Sukolilo Surabaya 60111 INDONESIA (tlp: 031-5939214; fax: 031-5939363; e-mail: chastine@cs.its.ac.id)*

perangkat lunak di *data set Turkish white-goods manufacturer* [5]. Metode seleksi fitur yang digunakan adalah Rasio Perolehan (*Gain Ration*/GR), Perolehan Informasi (*Information Gain*/IG), *One*-R (OR), *Relief-F* (RFF), dan *Symmetric Uncertainty* (SU). Hasil menunjukkan bahwa kombinasi NB dan OR dapat menghasilkan nilai prediksi paling tinggi dibandingkan keempat metode seleksi fitur lainnya.

Berdasarkan beberapa penelitian tersebut, maka pada makalah ini diusulkan perbaikan nilai prediksi CBC dengan menggunakan lima metode seleksi fitur, yaitu GR, IG, OR, RFF, dan SU. Dari kelima metode seleksi fitur tersebut, dilakukan analisis untuk menemukan kombinasi metode CBC dan seleksi fitur yang dapat menghasilkan nilai *balance* tertinggi dan jumlah fitur terbaik yang perlu dijadikan sebagai masukan (*input*) prediksi kesalahan perangkat lunak menggunakan CBC. Uji coba dilakukan pada lima *data set* kesalahan perangkat lunak *NASA public MDP* yaitu CM1, KC3, MW1, PC1, dan PC4. Hasil kombinasi CBC dengan seleksi fitur diharapkan dapat meningkatkan nilai *balance* prediksi kesalahan perangkat lunak.

Adapun sistematika penulisan pada makalah ini yaitu bagian II berisi pembahasan tentang landasan teori yang terdiri atas usulan metode seleksi fitur GR, IG, OR, RFF, dan SU, bagian III menjelaskan tentang metodologi pada penelitian ini, sedangkan pada bagian IV dipaparkan tentang uji coba dan analisis hasil. Terakhir, pada bagian V dijelaskan kesimpulan dan saran dari penelitian yang telah dilakukan.

II. SELEKSI FITUR

Berdasarkan penelitian sebelumnya, penggunaan semua fitur pada *data set* sebagai masukan model prediksi belum tentu dapat meningkatkan nilai prediksi [4], [5]. Salah satu faktor penyebabnya adalah kualitas data dari setiap fitur. Hanya fitur-fitur relevan dan tidak *redundant* saja yang akan dijadikan masukan model prediksi [6]. Pendekatan metode seleksi fitur yang digunakan pada makalah ini adalah filter, yaitu pemilihan fitur berdasarkan peringkat fitur.

A. Perolehan Informasi (IG)

Dikarenakan IG mampu menilai pentingnya fitur dengan mengukur nilai informasi yang diperoleh (relevan dengan label *class* atau tidak), maka IG diharapkan dapat mengubah nilai ketidakpastian sebuah rata-rata kandungan informasi *(entropy)* menjadi ukuran nilai informasi yang akan didapat, sebelum akhirnya informasi tersebut diambil dari beberapa fitur. Adapun perhitungan seleksi fitur menggunakan IG adalah sebagai berikut.

$$IG(Class, a) = H(Class) - H(Class|a) \quad (1)$$

dengan *H* adalah nilai *entropy*. Diasumsikan *A* adalah himpunan semua fitur dan *class* menjadi fitur dependen dari semua *training*, nilai (*a*,*y*) dengan $y \in Class$ mendefinisikan nilai dari contoh spesifik untuk fitur $a \in A$, nilai *v* merepresentasikan satu set nilai-nilai dari fitur *a*, yaitu $v = \{value(a,y) \mid a \in A \cap y \in Class\}$, dan |*s*| adalah jumlah elemen pada set *s*. Nilai *v* dapat juga diartikan sebagai jumlah kemunculan data tertentu di fitur *a*. Rumus IG pada setiap fitur $a \in A$ didefinisikan sebagai berikut.

$$IG(Class, a) = H(Class) - \sum_{v \in V} \frac{|\{y \in Class | value(a,y) = v\}|}{|Class|} \times H(\{y \in Class | value(a,y) = v\}) \quad (2)$$

Pada umumnya, IG dapat memberikan ukuran yang baik dalam menentukan relevansi fitur, tetapi terdapat keterbatasan yaitu *bias* yang mendukung fitur dengan banyak pilihan data [7].

B. Rasio Perolehan (GR)

GR memodifikasi IG dengan memperhatikan jumlah hasil yang diperoleh dari kondisi pengujian fitur [7]. Adapun rumus GR adalah sebagai berikut.

$$GR(Class, a) = \frac{IG(Class, a)}{H(a)} \quad (3)$$

dengan perhitungan *H(a)* sebagai berikut.

$$H(a) = -\sum_{v \in V} \frac{|\{y \in Class | value(a,y) = v\}|}{|Class|} \times log_2 \left(\frac{|\{y \in Class | value(a,y) = v\}|}{|Class|} \right). \quad (4)$$

Semua notasi rumus pada GR sama seperti rumus pada IG. *H(a)* pada GR disebut juga sebagai *split info* [8].

C. One-R (OR)

Seleksi fitur *OR* dapat membangun aturan berdasarkan fitur tunggal untuk setiap fitur dalam kumpulan data. Adapun algoritme *OR* adalah sebagai berikut [9].

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.

D. Relief-F (RFF)

Evaluasi fitur ini memberikan bobot untuk masing-masing fitur berdasarkan kemampuan fitur membedakan antar kelas, dan kemudian memilih fitur-fitur yang nilainya melebihi batas ambang yang ditetapkan sebelumnya sebagai fitur yang relevan [10]. Adapun algoritma *RFF* adalah sebagai berikut [11].

Input: sebuah *training* set *D*, jumlah iterasi *m*, jumlah *nearest neighbors k*, jumlah fitur *n*, *predefine* bobot fitur batas ambang δ.
Output: subset fitur *S* yang didasari oleh fitur yang bobotnya lebih besar dari batas ambang δ.
Langkah 1.
  Diasumsikan $S = \emptyset$, set semua bobot fitur $W(F_t) = 0$, $t = 1, 2, \ldots, n$.
Langkah 2.
- Pilih *sample R* dari *D* secara acak.

- Cari *k nearest neighbors* H_i (i = 1, 2, …, k) dari kelas yang sama dan *k nearest neighbors* M_i (C)(i = 1, 2, …, k) dari setiap kelas C yang berbeda.
- For t = 1 to n do

$$W(F_t) = W(F_t) - \sum_{i=1}^{k} \frac{diff(F_t, R, H_i)}{(mk)} + \sum_{C \neq class(R)} \frac{\left(\frac{P(C)}{1-P(Class(R))} \sum_{i=1}^{k} diff(F_t, R, M_i(C))\right)}{(mk)} \quad (5)$$

Langkah 3.

For t = 1 to n do

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

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:

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

dengan R_1 dan R_2 adalah dua sampel, $R_1[F_t]$, dan $R_2[F_t]$ adalah nilai fitur dari masing-masing R_1 dan R_2. Sedangkan pada rumus, *P(C)* adalah distribusi probabilitas *class C*. *Class(R)* adalah kelas yang masuk ke dalam kategori. *R*, M_i (C) menunjukkan *i near miss* dari *R* dalam *class C*, sedangkan *diff*(F_t, R_1, R_2) menunjukkan perbedaan R_1, R_2, F_t.

E. Symmetric Uncertainty (SU)

SU mengkompensasi *bias* IG terhadap fitur dengan nilai lebih tersendiri dan menormalkan nilai-nilai ke kisaran 0 hingga 1. Pengukuran SU dapat dihitung dengan (6).

$$SU(Class, a) = \frac{2 \times IG(Class, a)}{H(Class) + H(a)}. \quad (6)$$

Dikarenakan kemiripan rumus SU terhadap IG dan GR, maka hasil dapat menunjukkan nilai yang tidak jauh berbeda [5]. Namun pada penelitian yang lain, hasil perhitungan SU berbeda dengan hasil perhitungan IG dan GR [7].

III. METODOLOGI

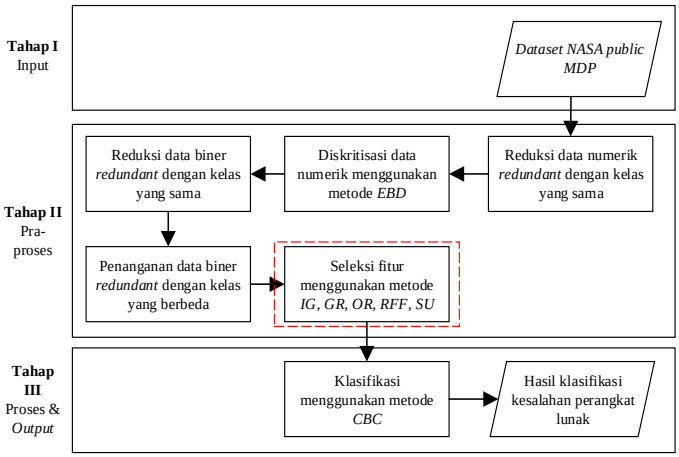
Prediksi kesalahan perangkat lunak pada makalah ini terdiri atas tiga tahapan utama, yaitu masukan, pra-proses, serta proses dan keluaran (*output*). Adapun desain sistem yang diusulkan dan kontribusi penelitian ini ditampilkan pada Gbr. 1.

A. Masukan

Dikarenakan tidak adanya akses untuk menggunakan *data set* privat, maka dimanfaatkanlah *data set* publik, yaitu *NASA public MDP*. Sebanyak lima *data set* digunakan pada penelitian ini, yaitu CM1, KC3, MW1, PC1, dan PC4. Jumlah dan pilihan *data set* disesuaikan dengan penelitian sebelumnya [2]. Adapun rincian setiap *data set* disajikan pada Tabel I [12], [13].

B. Pra-proses

1) Reduksi Data Numerik Redundant dengan Kelas yang Sama: Dalam setiap *data set NASA public MDP* terdapat data yang *redundant*. *Redundant* yang dimaksudkan adalah baris data yang sama persis, termasuk label kelasnya. Pada kasus ini, dalam *data set NASA public MDP* tidak terdapat data numerik *redundant* dengan kelas yang berbeda. Urutan langkah pada tahap ini ditunjukkan pada Gbr. 2. Sedangkan detail jumlah data numerik *redundant* pada *data set NASA public MDP* disajikan pada Tabel II.



Gbr. 1 Tahapan utama penelitian prediksi kesalahan perangkat lunak.

TABEL I
RINCIAN TUJUH *DATA SET NASA PUBLIC MDP*

Nama	Data	*Fault*	*% Fault*	Deskripsi
CM1	498	49	9,84	Instrumen pesawat ruang angkasa NASA
KC3	458	43	9,38	Manajemen menyimpanan untuk data ground
MW1	403	31	7,69	Percobaan gravitasi nol terhadap pembakaran
PC1	1109	77	6,94	Perangkat lunak penerbangan untuk satelit yang mengorbit di bumi
PC4	1458	178	12,21	Perangkat lunak penerbangan untuk satelit yang mengorbit di bumi

2) Diskritisasi Data Numerik: Perubahan data numerik ke data diskrit dilakukan untuk dapat meningkatkan nilai prediksi kesalahan perangkat lunak dan meningkatkan efisiensi pada proses komputasi [2], [14]. Metode yang digunakan untuk mengubah data numerik menjadi data diskrit adalah *Entropy-Based Discretization (EBD).*

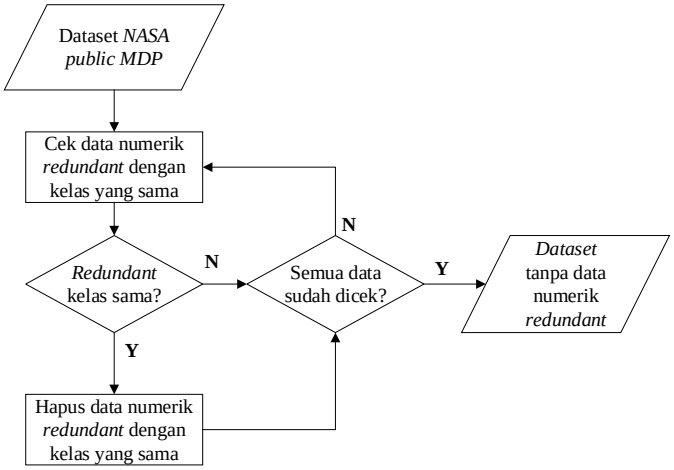
Algoritma EBD menggunakan nilai *entropy* informasi kelas dari kandidat pembagi untuk memilih batas diskrit [15]. Diasumsikan satu set *instance* yang telah diurutkan secara *ascending* (dari nilai terendah ke nilai tertinggi) adalah *S*. Satu set *instance S* tersebut kemudian dipartisi menjadi subset *S1* dan *S2*. Adapun *entropy* kelas subset *S* didefinisikan sebagai berikut [16].

$$Ent(S) = -\sum_{i=1}^{z} p(C_i, S) \log_2(p(C_i, S)) \quad (7)$$

dengan $p(C_i, S)$ adalah proporsi *instance* dengan kelas C_i, sedangkan z adalah jumlah kelas. *Entropy* kelas dihitung

berdasarkan bobot rata-rata *entropy* individual, yaitu S_1 dan S_2, yaitu hasil partisi *S*. *Entropy* informasi kelas dari partisi yang diinduksi oleh titik potong *T*, untuk sebuah fitur *F*, dihitung menggunakan (8) sebagai berikut.

$$E(F,T;S) = \frac{|S_1|}{S} Ent(S_1) + \frac{|S_2|}{S} Ent(S_2) \quad (8)$$



Gbr. 2 Urutan langkah reduksi data numerik *redundant* dengan kelas yang sama.

TABEL II
DAFTAR JUMLAH DATA *REDUNDANT* PADA *NASA PUBLIC MDP*

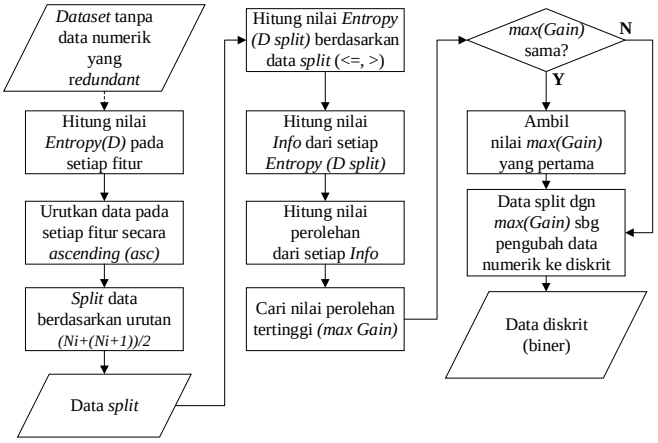
No	Nama	Fitur	Data	Jumlah *Redundant*
1	CM1	22	498	56
2	KC3	40	458	132
3	MW1	38	403	21
4	PC1	22	1109	155
5	PC4	38	1458	114

Nilai *T* adalah titik potong yang diambil dari titik tengah nilai fitur yang sudah diurutkan. Adapun titik potong *E(F,T;S)* minimum di antara semua kandidat titik potong diambil sebagai titik potong terbaik T_F dan sebagai penentu diskritisasi biner dari sebuah fitur. Titik potong minimum tersebut dipilih berdasarkan perolehan yang terbesar. Semakin kecil nilai titik potong *E(F,T;S*maka semakin besar pula nilai perolehan yang diperoleh. Untuk mendapatkan nilai perolehan, digunakan rumus berikut.

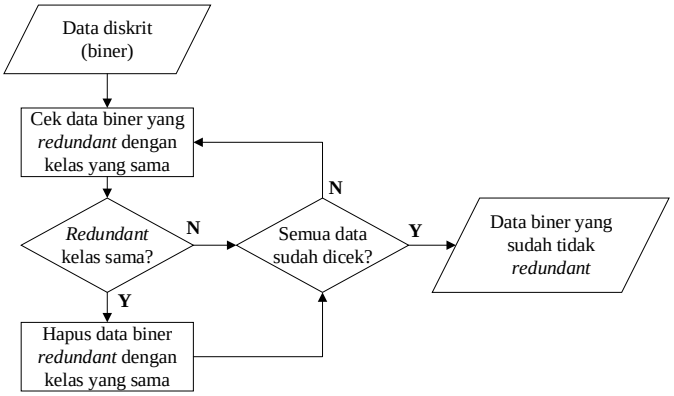
$$Gain(F,T;S) = Ent(S) - E(F,T;S) \quad (9)$$

Semua kemungkinan *split* dievaluasi secara rekursif, kemudian dipilih *split* terbaik, yaitu yang menghasilkan nilai perolehan tertinggi *(maximum Gain)*. Sebuah titik *split* ditentukan oleh diskritisasi yang paling mampu meminimalkan fungsi *E*. Adapun langkah-langkah yang dilakukan pada tahap ini disusun pada Gbr. 3.

Berdasarkan Gbr. 3, perhitungan nilai *entropy* dilakukan tiga kali, yaitu nilai *entropy* pada setiap fitur, serta nilai *entropy* pada pembagi data dengan kriteria kurang dari sama dengan (<=) dan lebih dari (>). Data *split* yang memiliki nilai perolehan tertinggi dijadikan kandidat utama untuk mengubah data numerik ke data diskrit.



Gbr. 3 Diskritisasi data numerik menggunakan EBD.

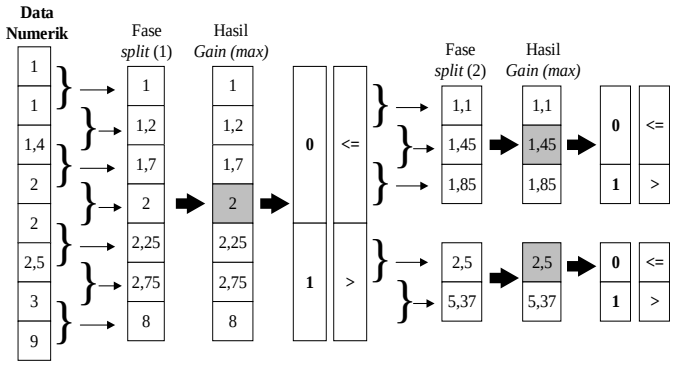


Gbr. 4 Proses reduksi data biner *redundant* dengan kelas yang sama.

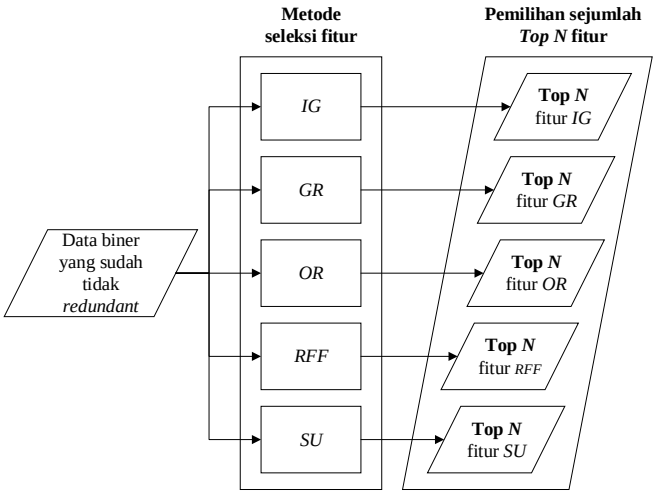
3) Reduksi Data Biner Redundant Dengan Kelas yang Sama: Data *redundant* dapat memengaruhi kinerja dan hasil prediksi kesalahan perangkat lunak [17]. Pada tahap ini, data biner hasil diskritisasi yang *redundant* dihilangkan. Tahapan pada proses ini ditunjukkan pada Gbr. 4.

4) Penanganan Data Biner Redundant dengan Kelas yang Berbeda: Algoritma EBD mengubah data numerik menjadi data diskrit. Jika interval numerik pada data itu tinggi, maka kemungkinan kemunculan data diskrit yang *redundant* dengan kelas berbeda akan tinggi pula. Data yang *redundant* seperti itu dapat membuat kinerja sistem menjadi tidak konsisten. Proses ini dilakukan dengan cara melakukan ulang proses *splitting* sehingga data biner hasil diskritisasi menjadi dua digit. Proses ini disebut juga sebagai proses kuantisasi. Ilustrasi *split* EBD dua fase ditunjukkan pada Gbr. 5.

5) Seleksi Fitur: Setelah *data set* mengalami diskritisasi, fitur pada *data set* selanjutnya diseleksi berdasarkan percobaan kelima metode seleksi fitur, yaitu IG, GR, OR, RFF, dan SU. Keluaran dari seleksi fitur adalah urutan fitur berdasarkan nilai peringkat fitur dari masing-masing metode. Sejumlah *Top N* fitur diambil dari nilai peringkat fitur tertinggi untuk dijadikan sebagai masukan pada tahap selanjutnya. Penentuan nilai *N* fitur dilakukan secara iteratif. Jumlah *Top N* fitur yang dapat menghasilkan nilai prediksi tertinggi pada proses klasifikasi adalah yang akan disajikan sebagai hasil. Adapun urutan langkah seleksi fitur pada tahap ini ditunjukkan pada Gbr. 6.



Gbr. 5 Ilustrasi *split* EBD dua fase untuk menangani data biner *redundant* dengan kelas yang berbeda.

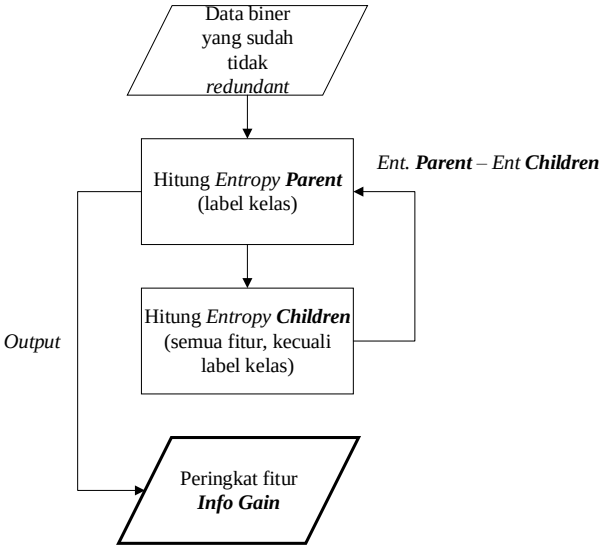


Gbr. 6 Urutan proses seleksi fitur.

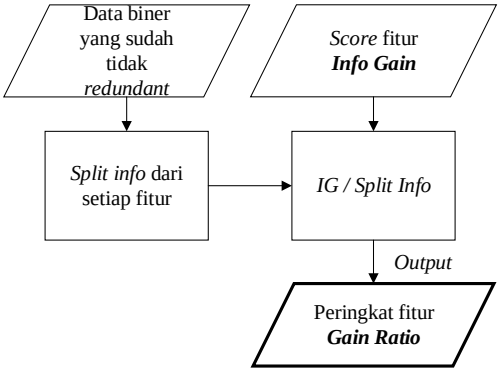
IG memanfaatkan hasil perhitungan *entropy* pada *data set* dan *entropy* setiap fiturnya dari data diskrit hasil proses EBD. *Entropy* pada satu *data set* disebut *entropy parent*, sedangkan *entropy* setiap fitur pada satu *data set* disebut *entropy children*. Berdasarkan Gbr. 7, peringkat fitur diperoleh dari *score* fitur. Adapun *score* fitur diperoleh dari hasil pengurangan nilai *entropy parent* dengan *entropy children*. Semakin besar *score* fitur, semakin tinggi pula peringkat fitur tersebut.

GR tidak hanya menggunakan keluaran proses EBD, tetapi juga *score* dari peringkat fitur IG. Langkah pertama yang dilakukan adalah menghitung nilai *split info* dari hasil proses EBD. Setelah nilai *split info* diketahui, *score IG* dari setiap fitur dibagi dengan masing-masing *split info* dari tiap fitur. Hasil pembagian tersebut adalah *score* GR. Sama halnya dengan IG, semakin tinggi *score* yang diperoleh, semakin tinggi pula peringkat fitur tersebut. Urutan langkah untuk mendapatkan peringkat fitur dengan GR ditunjukkan pada Gbr. 8.

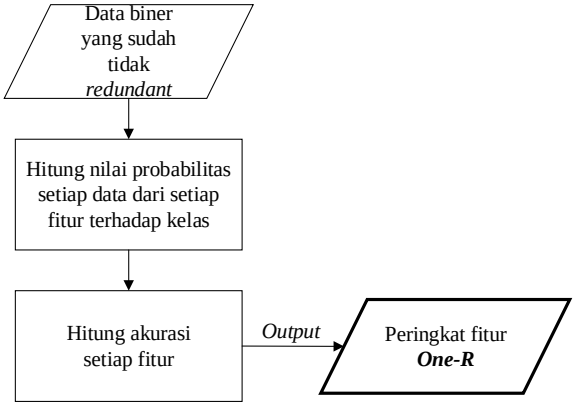
OR memanfaatkan nilai probabilitas setiap fitur untuk memberikan *score* pada setiap fitur. Sebelum *score* ditentukan, perlu dihitung nilai akurasi setiap fitur terlebih dahulu. Nilai akurasi fitur tersebut dijadikan sebagai *score* fitur. Nilai *score* berbanding lurus dengan peringkat fitur. Adapun urutan proses OR ditunjukkan pada Gbr. 9.



Gbr. 7 Urutan proses pemeringkatan fitur dengan IG.

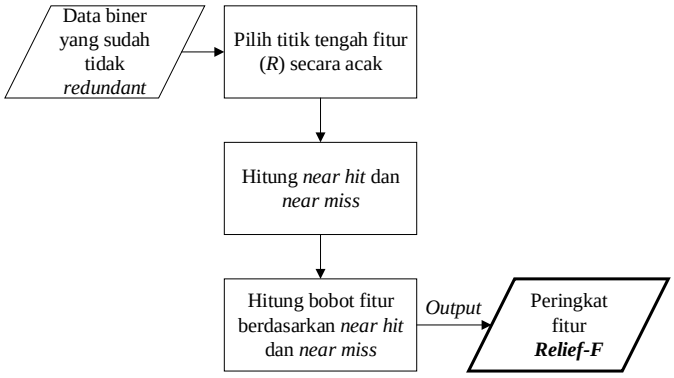


Gbr. 8 Urutan proses pemeringkatan fitur dengan GR.

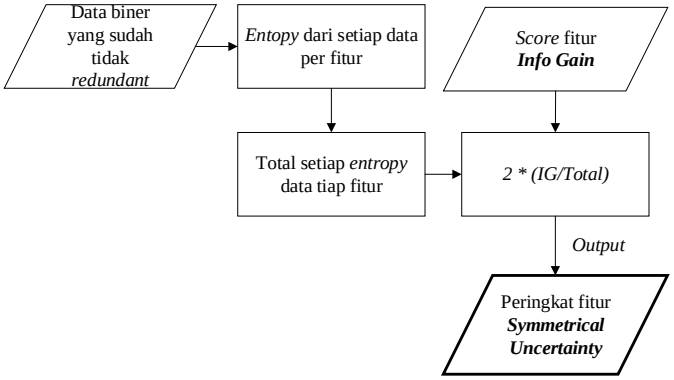


Gbr. 9 Urutan proses pemeringkatan fitur dengan OR.

Pada proses RFF, untuk mendapatkan *score* dari setiap fitur, RFF memanfaatkan nilai *near miss* dan *near hit*. Sebelum mencari nilai *near miss* dan *near hit*, perlu ditentukan terlebih dahulu nilai tengahnya (*R*). Penentuan *R* dilakukan secara acak. Setelah nilai *R* ditentukan, nilai *near miss* dan nilai *near hit* diketahui, sehingga *score* fitur dapat dihitung. Keluaran pada proses ini adalah urutan *peringkat* fitur berdasarkan perhitungan *score*. Urutan proses RFF ditunjukkan pada Gbr. 10.



Gbr. 10 Urutan proses pemeringkatan fitur dengan RFF.



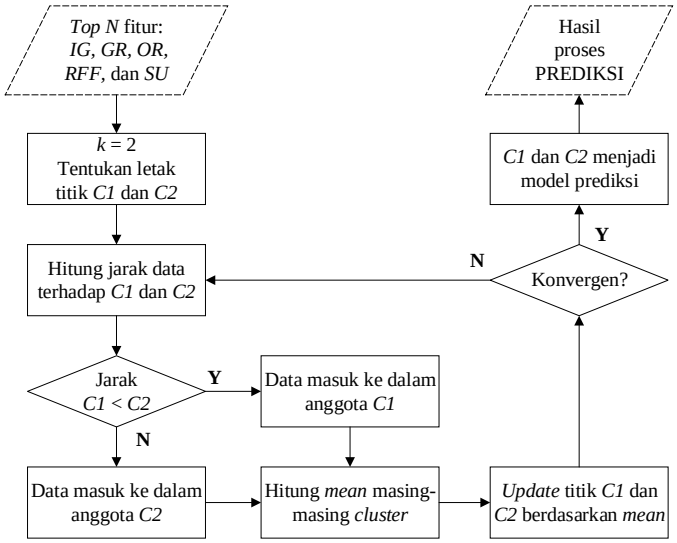
Gbr. 11 Urutan proses pemeringkatan fitur dengan *SU*.

Pada proses SU terdapat kesamaan dengan GR, karena SU juga memanfaatkan keluaran *score* dari IG. Namun, sebelum menggunakan *score* fitur dari IG, SU perlu mengetahui nilai total *entropy* dari setiap pilihan data per fitur. Setelah itu *score* IG dibagi dengan nilai total tersebut dan dikalikan dua. Hasil perhitungan tersebut adalah *score* fitur dari SU. Adapun urutan proses SU diperlihatkan pada Gbr. 11.

C. Proses & Keluaran

1) Prediksi Kesalahan Perangkat Lunak: Prediksi kesalahan perangkat lunak dilakukan dengan menggunakan CBC. Penggunaan metode CBC didasarkan pada penelitian sebelumnya, yaitu CBC dapat menghasilkan nilai prediksi yang lebih baik dibandingkan pemodelan prediksi kesalahan perangkat lunak menggunakan NB [2]. Pada dasarnya, proses *clustering* mengelompokkan data-data berdasarkan kesamaan atau perbedaannya pada setiap titik *cluster* [18]. Adapun pada kasus ini, jumlah titik *cluster (k)* ada dua. Penentuan nilai *k* = 2 didasarkan pada jumlah pilihan *class* di *NASA public MDP*, yaitu *faulty (true)* dan *non-faulty (false)*. Artinya, data akan dibagi berdasarkan dua kelompok tersebut. Adapun urutan proses prediksi kesalahan perangkat lunak menggunakan CBC ditunjukkan pada Gbr. 12.

Pada Gbr. 12 dijelaskan tentang langkah-langkah pada proses CBC. *C1* dan *C2* ditentukan berdasarkan kelas *true* dan *false*. Jika *C1* mewakili kelas *true*, maka *C2* harus mewakili kelas *false*, dan begitu juga sebaliknya. Dikarenakan data hasil diskritisasi adalah biner, maka perhitungan jarak CBC yang cocok pada penelitian adalah *hamming distance* [19]. Ketika sebuah data cenderung lebih dekat jaraknya terhadap *C1*, maka data tersebut masuk ke dalam anggota *C1*, dan sebaliknya. *Cluster-center* pada setiap *cluster* diperbaharui berdasarkan *mean* dari masing-masing *cluster*. Apabila anggota *cluster* sudah konvergen, maka titik *cluster-center* tersebut akan dijadikan acuan (model) untuk menentukan keanggotaan data pada data *testing*. Perbedaan metode CBC pada peneltian ini dengan penelitian sebelumnya adalah penggunaan perhitungan jarak, yaitu pada penelitian sebelumnya digunakan *Euclidean distance* [2].



Gbr. 12 Proses prediksi kesalahan perangkat lunak menggunakan CBC.

2) Hasil Prediksi Kesalahan Perangkat Lunak: Hasil klasifikasi metode *CBC* digunakan sebagai acuan untuk menentukan sebuah modul perangkat lunak rentan kesalahan *(true)* atau tidak *(false)*. Namun, sebelum menggunakan hasil prediksi sebagai acuan, nilai prediksi perlu diukur terlebih dahulu untuk mengetahui sebarapa besar keandalannya. Pada penelitian sebelumnya, pengukuran prediksi kesalahan perangkat lunak dilakukan dengan menggunakan *confusion matrix* [2]. *Confusion matrix* ditunjukkan pada Tabel III.

TABEL III
CONFUSION MATRIX

	AKTUAL	
PREDIKSI	*Faulty*	*Non-faulty*
Faulty	TP	FP
Non-faulty	FN	TN

Penelitian ini menggunakan ukuran kinerja prediksi *pd* dan *pf* untuk mengevaluasi dan membandingkan hasil prediksi. Nilai *pd* adalah nilai keberhasilan sistem dalam memprediksi kesalahan perangkat lunak. Sedangkan nilai *pf* adalah nilai misklasifikasi sistem dalam menentukan modul yang secara aktual tidak terdapat kesalahan, tetapi diklasifikasikan sebagai modul yang salah.

Idealnya, sistem prediksi kesalahan perangkat lunak dapat menghasilkan nilai *pd* sebesar 100% dan nilai *pf* sebesar 0%. Artinya, sistem dapat memprediksi secara tepat kesalahan perangkat lunak tanpa melakukan misklasifikasi terhadap modul yang tidak terdapat kesalahan perangkat lunak. Nilai

ideal sulit untuk dicapai, karena persebaran jumlah kelas *false* dan kelas *true* pada *data set* tidak seimbang. Oleh karena itu, diperlukan nilai *balance* sebagai *trade-off* antara nilai *pd* dan nilai *pf*. Semakin nilai *pd* mendekati 100% dan nilai *pf* mendekati 0%, semakin tinggi pula nilai *balance* sistem. Semakin tinggi nilai *balance*, maka sistem prediksi semakin dapat diandalkan. Perhitungan nilai *pd*, *pf*, dan *balance* ditunjukkan pada (9) sampai dengan (11).

$$pd = \frac{TP}{(TP + FN)} \quad (9)$$

$$pf = \frac{FP}{(FP + TN)} \quad (10)$$

$$balance = 1 - \frac{\sqrt{(0 - pf)^2 + (1 - pd)^2}}{\sqrt{2}} \quad (11)$$

IV. UJI COBA DAN ANALISIS HASIL

Setelah mengimplementasikan metode yang diusulkan, tahapan selanjutnya adalah uji coba. Uji coba dilakukan untuk mengetahui pengaruh usulan kombinasi metode terhadap nilai *balance*. Sejumlah lima *fold* digunakan untuk melakukan *cross validation* di tahap pengujian penelitian ini.

Data masukan pada uji coba ini adalah lima *data set* kesalahan perangkat lunak yang diperoleh dari *NASA public MDP*, yaitu CM1, KC3, MW1, PC1, dan PC4. Adapun skenario uji coba pada penelitian ini juga terbagi menjadi lima. Langkah-langkah yang diterapkan pada setiap skenario adalah sama, yang membedakannya adalah penggunaan *data set*.

Di setiap hasil skenario dilakukan perbandingan hasil antar usulan metode yaitu kombinasi seleksi fitur dengan CBC dan CBC tanpa seleksi fitur.

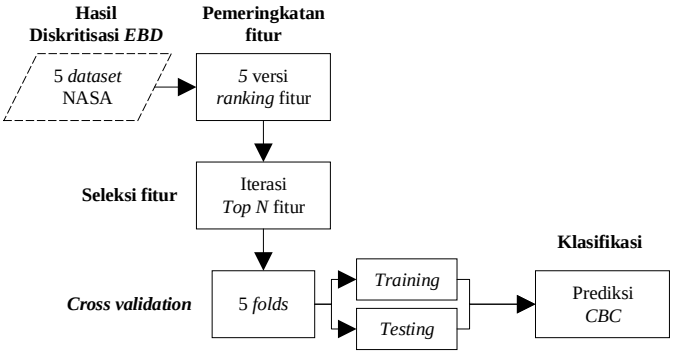
A. Uji Coba

Untuk melakukan proses uji coba, setiap *data set* didiskritkan terlebih dahulu menggunakan metode *EBD* dua fase. Selanjutnya, setiap fitur di setiap *data set* diurutkan berdasarkan lima metode seleksi fitur. Pada tahap ini, ada lima versi peringkat fitur dari tiap *data set*. Setelah itu, dilakukan pemilihan sejumlah *Top N* fitur secara iteratif dari lima versi peringkat fitur. Setiap iterasi *Top N* fitur dijadikan sebagai masukan model klasifikasi *CBC* dengan lima *fold cross validation*. Ilustrasi skenario pengujian ini diperlihatkan pada Gbr. 13.

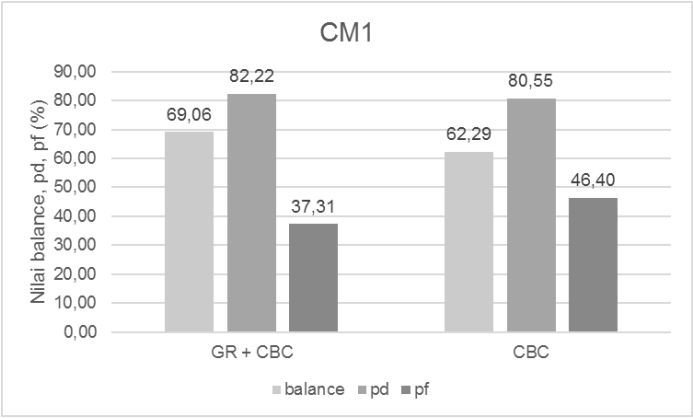
B. Analisis Hasil Uji Coba

1) Evaluasi Uji Coba di CM1: Dihasilkan nilai *balance* tertinggi dari usulan metode kombinasi GR dan CBC, yaitu **69,06%** dengan nilai *pd* 82,22% dan nilai *pf* 37,31%. Jika dibandingkan dengan CBC tanpa seleksi fitur, maka *balance* yang dihasilkan usulan metode 6,77% lebih tinggi. Perbandingan nilai *balance*, *pd*, dan *pf* pada CM1 ditunjukkan pada Gbr. 14.

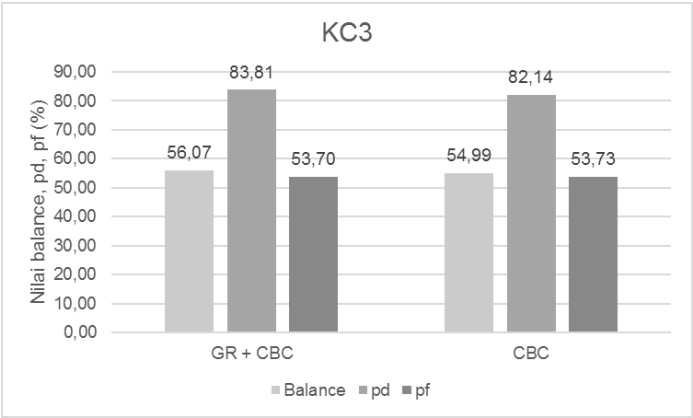
2) Evaluasi Uji Coba di KC3: Dihasilkan nilai *balance* tertinggi dari usulan metode kombinasi GR dan CBC, yaitu **56,07%** dengan nilai *pd* 83,81% dan nilai *pf* 53,70%. Jika dibandingkan dengan CBC tanpa seleksi fitur, maka *balance* yang dihasilkan usulan metode 1,08% lebih tinggi. Perbandingan nilai *balance*, *pd*, dan *pf* pada KC3 ditunjukkan pada Gbr. 15.



Gbr. 13 Ilustrasi skenario pengujian.

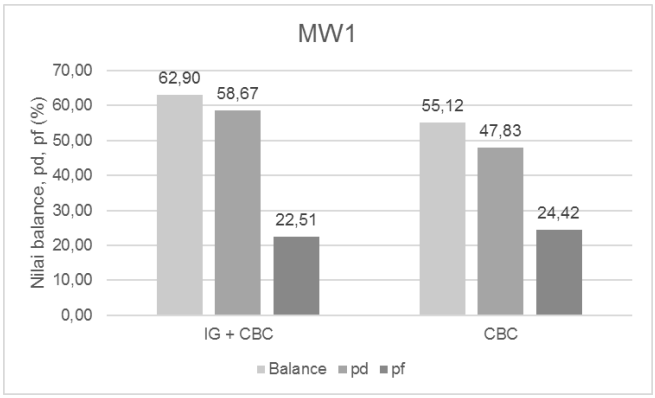


Gbr. 14 Evaluasi uji coba di CM1.

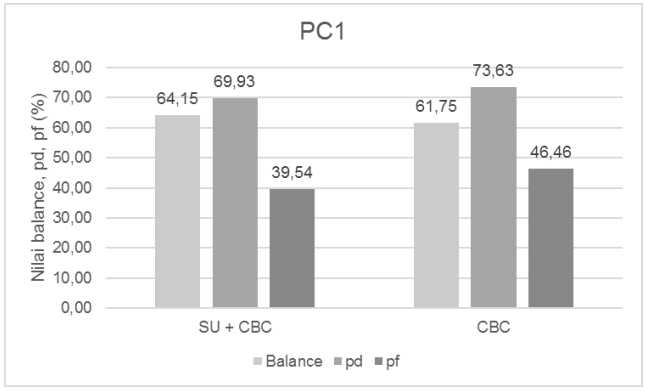


Gbr. 15 Evaluasi uji coba di KC3.

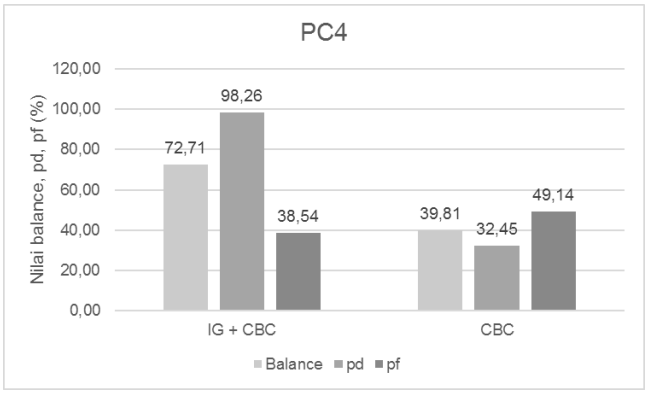
3) Evaluasi Uji Coba di MW1: Dihasilkan nilai *balance* tertinggi dari usulan metode kombinasi IG dan CBC, yaitu **62,90%** dengan nilai *pd* 58,67% dan nilai *pf* 22,51%. Jika dibandingkan dengan CBC tanpa seleksi fitur, maka *balance* yang dihasilkan usulan metode 7,78% lebih tinggi. Perbandingan nilai *balance*, *pd*, dan *pf* pada MW1 diperlihatkan pada Gbr. 16.



Gbr. 16 Evaluasi uji coba di MW1.



Gbr. 17 Evaluasi uji coba di PC1.



Gbr. 18 Evaluasi uji coba di PC4.

4) Evaluasi Uji Coba di PC1: Dihasilkan nilai *balance* tertinggi dari usulan metode kombinasi SU dan CBC, yaitu **64,15%** dengan nilai *pd* 69,93% dan nilai *pf* 39,54%. Jika dibandingkan dengan CBC tanpa seleksi fitur, maka *balance* yang dihasilkan usulan metode 2,40% lebih tinggi. Perbandingan nilai *balance*, *pd*, dan *pf* pada PC1 ditunjukkan pada Gbr. 17.

5) Evaluasi Uji Coba di PC4: Dihasilkan nilai *balance* tertinggi dari usulan metode kombinasi IG dan CBC, yaitu **72,71%** dengan nilai *pd* 98,26% dan nilai *pf* 38,54%. Jika dibandingkan dengan CBC tanpa seleksi fitur, maka *balance* yang dihasilkan usulan metode 32,90% lebih tinggi. Perbandingan nilai *balance*, *pd*, dan *pf* pada PC4 ditunjukkan pada Gbr. 18.

TABEL IV
PERBANDINGAN NILAI *BALANCE* (%) ANTARA METODE CBC DENGAN DAN TANPA SELEKSI FITUR

No	*Data set*	CBC					
		IG	**GR**	**OR**	**RFF**	**SU**	
1	CM1	68,70	**69,06**	63,20	65,70	65,05	62,29
		11 fitur	9 fitur	9 fitur	11 fitur	11 fitur	22 fitur
2	KC3	51,45	**56,07**	50,18	54,07	**56,07**	54,99
		38 fitur	16 fitur	19 fitur	17 fitur	38 fitur	40 fitur
3	MW1	**62,90**	53,04	42,22	59,14	53,62	55,12
		31 fitur	26 fitur	19 fitur	6 fitur	19 fitur	38 fitur
4	PC1	63,78	62,50	62,35	62,36	**64,15**	61,75
		15 fitur	5 fitur	15 fitur	20 fitur	3 fitur	22 fitur
5	PC4	**72,71**	**72,71**	55,44	**72,71**	**72,71**	39,81
		2 fitur	2 fitur	1 fitur	2 fitur	2 fitur	38 fitur
Rata-rata		**63,91**	62,67	54,68	62,80	62,32	54,79

6) Evaluasi Uji Coba di Lima Data Set: Adapun perbandingan nilai *balance* dari setiap kombinasi metode disajikan pada Tabel IV. Sebagai keterangan tambahan, kolom yang berwana hitam adalah metode CBC tanpa seleksi fitur. Kolom yang berwarna abu-abu adalah hasil dari metode CBC tanpa seleksi fitur. Angka dengan cetak tebal menandakan bahwa nilai *balance* tersebut adalah yang tertinggi.

V. KESIMPULAN DAN SARAN

Berdasarkan skenario uji coba dan analisis hasil pengujian, kombinasi IG dengan CBC menghasilkan nilai rata-rata *balance* tertinggi, yaitu 63,91%. Sedangkan metode CBC tanpa seleksi fitur hanya memperoleh nilai rata-rata *balance* sebesar 54,79%. Jadi, dapat disimpulkan bahwa metode seleksi fitur IG dapat meningkatkan nilai rata-rata *balance* metode CBC tertinggi di lima *data set NASA public MDP*, yaitu dengan peningkatan nilai rata-rata *balance* sebesar 9,12%. Berdasarkan nilai rata-rata *balance* tersebut, dapat diambil kesimpulan pula bahwa metode seleksi fitur IG mampu meningkatkan nilai rata-rata *pd* sebesar 13,72% dan menurunkan nilai rata-rata *pf* sebesar 6,48% dari metode CBC di lima *data set NASA public MDP*.

Berdasarkan hasil pengujian, perlu dicari nilai *dumping factor* sebagai *threshold* jumlah maksimal toleransi *redundance* data biner dengan kelas yang berbeda pada proses EBD. Selain itu, perlu diteliti lebih lanjut tentang penggunaan metode yang dapat mengatasi persebaran kelas yang tidak merata. Metode seleksi fitur dengan pendekatan *wrapper* juga dapat diteliti lebih lanjut untuk mengoptimalkan hasil kombinasi fitur yang akan digunakan CBC.

REFERENSI

[1] B. Pudjoatmodjo and M. Hendayun, "Kehandalan Software Berdasarkan Data Sekunder Menggunakan Distribusi Poisson dan Kualifikasi Cronbach ' s Alpha," *J. Nas. Tek. Elektro dan Teknol. Inf.*, vol. 5, no. 2, 2016.

[2] P. Singh and S. Verma, "An Efficient Software Fault Prediction Model using Cluster based Classification," *Int. J. Appl. Inf. Syst.*, vol. 7, no. 3, pp. 35–41, 2014.

[3] T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell, "A Systematic Review of Fault Prediction Performance in Software Engineering," *IEEE Trans. Softw. Eng.*, vol. 38, no. 6, pp. 1276–1304, 2012.

[4] T. Menzies, J. Greenwald, and A. Frank, "Data Mining Static Code Attributes to Learn Defect Predictors," *IEEE Trans. Softw. Eng.*, vol. 33, no. 1, pp. 2–14, 2007.

[5] D. A. A. G. Singh, A. E. Fernando, and E. J. Leavline, "Experimental study on feature selection methods for software fault detection," *Int. Conf. Circuit, Power Comput. Technol.*, pp. 1–6, 2016.

[6] R. Abraham, J. B. Simha, and S. S. Iyengar, "Effective Discretization and Hybrid feature selection using Naïve Bayesian classifier for Medical datamining," *Int. J. Comput. Intell. Res.*, vol. 5, no. 2, pp. 116–129, 2009.

[7] K. Gao, T. M. Khoshgoftaar, H. Wang, and N. Seliya, "Choosing software metrics for defect prediction: an investigation on feature selection techniques," *Softw. - Pract. Exp.*, vol. 39, no. 7, pp. 701–736, 2011.

[8] J. R. Quinlan, "Induction of Decision Trees," *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, 1986.

[9] G. Holmes and C. G. Nevill-Manning, "Feature Selection Via The Discovery Of Simple Classification Rules," *Work. Pap. 95/10*, vol. ISSN 1170-, pp. 1–5, 1995.

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

[11] 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.*, pp. 246–250, 2011.

[12] P. Singh and S. Verma, "Cross Project Software Fault Prediction at Design Phase," *Int. J. Comput. Electr. Autom. Control Inf. Eng.*, vol. 9, no. 3, pp. 800–805, 2015.

[13] R. Sathyaraj and S. Prabu, "An Approach for Software Fault Prediction to Measure the Quality of Different Prediction Methodologies using Software Metrics," *Indian J. Sci. Technol.*, vol. 8, no. December, 2015.

[14] P. Singh and O. P. Vyas, "Software Fault Prediction Model for Embedded Software : A Novel finding," *Int. J. Comput. Sci. Inf. Technol.*, vol. 5, no. 2, pp. 2348–2354, 2014.

[15] J. Dougherty, R. Kohavi, and M. Sahami, "Supervised and unsupervised discretization of continuous features," *Proc. 12th Int. Conf. Mach. Learn.*, pp. 194–202, 1995.

[16] U. M. Fayyad and K. B. Irani, "Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning." *IJCAI*, pp. 1022-1029, 1993.

[17] C. Akalya Devi, K. E. Kannammal, and B. Surendiran, "A Hybrid Feature Selection Model For Software Fault Prediction," *Int. J. Comput. Sci. Appl.*, vol. 2, no. 2, pp. 25–35, 2012.

[18] D. T. Hidayat, C. Fatichah, and R. V. H. Ginardi, "Pengelompokan Data Menggunakan Pattern Reduction Enhanced Ant Colony Optimization dan Kernel Clustering," *J. Nas. Tek. Elektro dan Teknol. Inf.*, vol. 5, no. 3, pp. 1–6, 2016.

[19] D. H. Murti, N. Suciati, and D. J. Nanjaya, "Clustering data non-numerik dengan pendekatan algoritma k-means dan hamming distance studi kasus biro jodoh," *J. Ilm. Teknol. Inf.*, vol. 4, pp. 46–53, 2005.