Integrasi SMOTE dan *Information Gain* pada *Naive Bayes* untuk Prediksi Cacat *Software*

Sukmawati Anggraini Putri
Program Studi Ilmu Komputer, STMIK Nusa Mandiri Jakarta
sukma.chan@gmail.com

Romi Satria Wahono
Fakultas Ilmu Komputer, Dian Nuswantoro University
Email: romi@brainmatics.com

Abstrak: Perangkat lunak memainkan peran penting banyak. Oleh karena itu, kewajiban untuk memastikan kualitas, seperti pengujian perangkat lunak dapat dianggap mendasar dan penting. Tapi di sisi lain, pengujian perangkat lunak adalah pekerjaan yang sangat mahal, baik dalam biaya dan waktu penggunaan. Oleh karena itu penting untuk sebuah perusahaan pengembangan perangkat lunak untuk melakukan pengujian kualitas perangkat lunak dengan biaya minimum. Naive Bayes pada prediksi cacat perangkat lunak telah menunjukkan kinerja yang baik dan menghsilkan probabilitas rata-rata 71 persen. Selain itu juga merupakan *classifier* yang sederhana dan waktu yang dibutuhkan dalam proses belajar mengajar lebih cepat dari algoritma pembelajaran mesin lainnya. NASA adalah dataset yang sangat populer digunakan dalam pengembangan model prediksi cacat *software*, umum dan dapat digunakan secara bebas oleh para peneliti. Dari penelitian yang dilakukan sebelumnya ada dua isu utama pada prediksi cacat perangkat lunak yaitu *noise attribute* dan *imbalance class*. Penerapan teknik SMOTE (*Minority Synthetic Over-Sampling Technique*) menghasilkan hasil yang baik dan efektif untuk menangani ketidakseimbangan kelas pada teknik oversampling untuk memproses kelas minoritas (positif). Dan *Information Gain* digunakan dalam pemilihan atribut untuk menangani kemungkinan *noise attribute*. Setelah dilakukan percobaan bahwa penerapan model SMOTE dan *Information Gain* terbukti menangani *imbalance class* dan *noise attribute* untuk prediksi cacat software.

Kata Kunci: prediksi cacat software, *Information Gain*, *Naive Bayes*, SMOTE

1 PENDAHULUAN

Perangkat lunak ini memainkan peran penting dalam semua bidang dan aktivitas manusia. Terbukti dengan ukuran nilai pasar pada perusahaan pengembangan perangkat lunak di dunia yang bernilai 407 juta dolar pada 2013 (Rivera & Meulen, 2014). Tugas untuk memastikan kualitas seperti pengujian perangkat lunak dapat dianggap dasar dan penting. Tapi di sisi lain, pengujian perangkat lunak adalah pekerjaan yang sangat mahal, baik dalam biaya dan penggunaan waktu (de Carvalho, Pozo, & Vergilio, 2010).

Sekitar 30 tahun, prediksi cacat *software* telah menjadi topik penelitian penting di bidang rekayasa perangkat lunak. Prediksi yang akurat dari modul software cacat rawan dapat membantu mengurangi biaya (Catal, 2011). Penelitian prediksi cacat software berfokus pada 1) perkiraan jumlah cacat yang tersisa dalam sistem perangkat lunak, 2) menemukan hubungan cacat perangkat lunak, 3) klasifikasi rawan cacat dalam komponen software, yang terdiri dari dua kelas, yaitu rawan cacat dan bukan rawan cacat (Song, Jia, Shepperd, Ying, & Liu, 2011).

Klasifikasi adalah pendekatan populer untuk memprediksi cacat software (Lessmann, Member, Baesens, Mues, and Pietsch, 2008). Pengklasifikasi yang telah digunakan seperti: C4.5, Naif Bayes, Logictic Regresi, Regresi Linear dan SVM (Hall, Beecham, Bowes, Gray, & Counsell, 2010). Jadi dalam penelitian ini mengadopsi dan lebih fokus pada pendekatan klasifikasi. Ini mengkategorikan kode software atribut ke cacat atau tidak cacat, yang dikumpulkan dari penelitian sebelumnya. Algoritma klasifikasi mampu memprediksi komponen lebih cenderung mendukung cacat rawan lebih tepat sasaran pada sumber pengujian, sehingga meningkatkan efisiensi. Jika kesalahan dilaporkan selama tes sistem atau dari percobaan, modul data kesalahan ditandai sebagai 1, jika tidak 0. Untuk pemodelan prediksi, metrik perangkat lunak yang digunakan sebagai variabel independen dan data kesalahan digunakan sebagai variabel dependen (Catal, 2011). Berbagai jenis algoritma klasifikasi telah diterapkan untuk memprediksi cacat perangkat lunak, termasuk logistic regression (Lessmann, Member, Baesens, Mues, & Pietsch, 2008), J48 (Riquelme, Ruiz, & Moreno, 2008), OneR (Song et al., 2011), Neural Network (Wahono & Suryana, 2013) dan naive bayes (Menzies, Greenwald, & Frank, 2007). Dari penelitian tersebut Naive Bayes dan Logistic Regression, menunjukkan akurasi yang baik, dibandingkan dengan algoritma klasifikasi lainnya (Hall et al., 2010).

Naive Bayes pada (Menzies et al., 2007) telah menunjukkan kinerja yang baik dan menghasilkan probabilitas rata-rata 71 persen. Naive Bayes merupakan klasifikasi sederhana (Domingos, 1997) dan waktu yang dibutuhkan dalam proses pembelajar lebih cepat dari pada pembelajaran mesin lain (Menzies et al., 2007). Selain itu memiliki reputasi yang baik pada keakuratan prediksi (Turhan & Bener, 2009).

Berdasarkan permasalahan yang dihadapi oleh para peneliti prediksi cacat software yang dilakukan sebelumnya, ada dua masalah utama meliputi *noise attribute* (Kabir & Murase, 2012) dan *class imbalance* (Wahono & Suryana, 2013). *Imbalance* dapat menyebabkan model yang tidak praktis dalam perangkat lunak prediksi cacat, karena kebanyakan kasus akan diprediksi sebagai non-cacat rawan (Khoshgoftaar & Gao, 2009). Pembelajaran dari dataset yang tidak seimbang sangat sulit. Pada pendekatan level melibatkan teknik pengambilan sampel untuk mengurangi ketidakseimbangan kelas. Pendekatan pertama menggunakan undersampling untuk menangani ketidakseimbangan dalam kelas dari *not fault prone* (nfp) modul kelas mayoritas (negative) dan untuk menangani ketidakseimbangan dalam kelas dari *fault prone* (fp modul kelas minoritas (positive) (Yap et al., 2014). Pada laporan menyatakan oversampling dapat menyebabkan overfitting

untuk membuat duplikat jumlah yang sama dengan sampel minoritas, sementara undersampling dapat membuang sebagian besar potensi sampel berguna (Yap et al., 2014). Penggunaan teknik SMOTE (*Synthetic Minority Over-Sampling Technique*) menghasilkan hasil yang baik dan cara yang efektif untuk menangani ketidakseimbangan kelas yang mengalami *overfitting* pada teknik *oversampling* untuk memproses kelas minoritas (positive)(Chawla, Bowyer, Hall, & Kegelmeyer, 2002).

Pemilihan atribut digunakan untuk menangani *noise* atribut (Wahono & Suryana, 2013). Pada riset yang dilakukan oleh (Kabir & Murase, 2012) menjelaskan tujuan utama dari pemilihan atribut adalah untuk menyederhanakan dan meningkatkan kualitas dataset dengan memilih atribut atribut yang relevan. Information Gain menunjukkan hasil yang baik dalam bobot atribut untuk pemilihan atribut yang relevan (Khoshgoftaar & Gao, 2009).

Pada penelitian ini, kami mengusulkan kombinasi antara algoritma Information Gain (IG) dan teknik SMOTE untuk meningkatkan prediksi cacat software. Penerapan Information Gain untuk menangani noise atribut dan teknik SMOTE untuk menangani masalah *class imbalance*. Information gain dipilih karena mampu untuk menemukan dan memilih atribut yang relevan (Khoshgoftaar & Gao, 2009). Teknik SMOTE dipilih karena efektivitas dalam penanganan masalah ketidakseimbangan kelas dalam dataset cacat perangkat lunak (Riquelme et al., 2008).

Penyusunan pada makalah ini sebagai berikut. Penelitian-penelitan terkait dijelaskan pada bagian 2. Pada bagian 3 akan dijelaskan metode yang diusulkan. Hasil ekperimen membandingkan metode yang diusulkan dengan hasil penelitian lainnya akan dijelaskan pada bagian 4. Dan yang terkahir adalah meringkas hasil makalah ini dalam bagian terakhir.

2 PENELITIAN TERKAIT

Menzies, Greenwald dan Frank (Menzies et al., 2007) melakukan riset pada tahun 2007, dimana mereka membandingkan kinerja algoritma pembelajaran mesin yang terdiri dari *Rule Induction* dan *Naive Bayes* untuk memprediksi komponen software yang cacat. Mereka menggunakan 10 dataset dari NASA *Metric Data Program* (MDP) repository yang merupakan dataset bersifat publik. Pada penelitian tersebut mereka menemukan Naive Bayes classification memiliki probabilitas yang baik yaitu 71%, yang sebelumnya dilakukan preprocessing menggunakan *log filtering* dan seleksi atribut menggunkaan Information Gain. Hasil ini secara signifikan mengungguli *Rule Induction (J.48 dan OneR).*

Namun dataset NASA MDP merupakan dataset berskala besar dan berdimensi tinggi (Wang, Khoshgoftaar, Gao, & Seliya, 2009). Sehingga menimbulkan masalah *imbalance class* dan *noise* atribut (Wahono & Suryana, 2013).

Penelitian yang dilakukan oleh Wahono et al attributesb (Wahono & Suryana, 2013) mengklaim bahwa umumnya seleksi atribut digunakan dalam pembelajaran mesin ketika melibatkan dataset berdimensi tinggi yang memungkinkan mengalami *noise* atribut. Metode yang digunakan pada penelitian ini adalah *optimization metaheuristics* (*genetic algorithm* dan *Particle Swarm Optimization* (PSO)) dan teknik Bagging untuk menangani *class imbalance*, sehingga dapat meningkatkan kinerja prediksi cacat perangkat lunak.

Sementara penelitian yang dilakukan oleh Gao et al (Gao & Khoshgoftaar, 2011) menyatakan bahwa algoritma seleksi atribut dibagi menjadi dua teknik, yaitu teknik *filter* dan *wrapper*. Teknik *filter* seperti *chi-square*, *information gain* dan *relief*. Dan hasil pada penelitian ini menemukan bahwa information gain menunjukkan kinerja yang lebih baik dibandingkan dengan dua algoritma lainnya.

Masalah *imbalance class* diamati di berbagai domain, termasuk perangkat lunak prediksi cacat. Beberapa metode telah diusulkan dalam literatur untuk menangani *imbalance class*: pendekatan level data (Data sampling) dan pendekatan algoritma (algoritma pembelajaran meta). Data sampel merupakan pendekatan utama untuk menangani imbalance class, proses ini melibatkan penyeimbangan distribusi kelas dari dataset. Ada dua jenis data sampel: *undersampling* dan *oversampling* (Yap et al., 2014). SMOTE merupakan teknik oversampling yang baik dan efektif untuk menangani *overfitting* pada proses *oversampling* untuk menangani ketidakseimbangan di kelas modul yang cacat pada kelas minoritas (positif)(Chawla et al., 2002)(Riquelme et al., 2008).

Pada penelitian ini, melakukan kombinasi antara teknik SMOTE untuk menangani masalah *imblance class* dan algoritma *Information Gain* untuk seleksi atribut, dalam konteks prediksi cacat *software*. Sementara pada penelitian sebelumnya *Naive Bayes* menangani noise atibut dan *imbalance class* dengan baik, namun dilakukan secara terpisah. Sehingga pada penelitian ini menggunakan dua teknik tersebut secara bersama-sama pada prediksi cac*t software*.

3 METODE USULAN

Pada penilitan ini mengusulkan metode yaitu NB SMOTE+IG, yang merupakan kombinasi SMOTE+IG berbasis NB, untuk mencapai kinerja yang lebih baik dari prediksi software prediksi cacat. Gambar 1 menunjukkan diagram activity dari metode yang diusulkan yaitu NB SMOTE+IG.

Tujuan utama dari teknik SMOTE adalah untuk menangani *imbalance class*. Penggunaan teknik SMOTE (*Synthetic Minority Over-Sampling Technique*) (Chawla et al., 2002) menghasilkan hasil yang baik dan efektif untuk menangani imbalance class yang mengalami *over-fitting* pada proses teknik *over-sampling* untuk kelas minoritas (positif) (Riquelme et al., 2008). SMOTE menciptakan sebuah contoh dari kelas minoritas sintetis yang beroperasi di ruang fitur daripada ruang data. Dengan menduplikasi contoh kelas minoritas, teknik SMOTE menghasilkan contoh sintetis baru dengan melakukan ekstrapolasi sampel minoritas yang ada dengan sampel acak yang diperoleh dari nilai k tetangga terdekat. Dengan hasil sintetis pada contoh yang lebih dari kelompok minoritas, sehingga mampu memperluas area keputusan mereka untuk minoritas (Chawla et al., 2002).

Seleksi atribut (pilihan fitur) adalah bagian dari atribut dari proses seleksi yang relevan untuk membangun model pembelajaran yang baik (Guyon, 2003). Pada (Jain & Richariya, 2012) dijelaskan perhitungan dari *information gain*. Misalkan terdapat kelas *m* dan *training set* berisi sampel *SI*, yng diberikan nama kelas *I* and *S* adalah jumlah sampel dalam *training set* adalah informasi yang diharapkan diperlukan untuk mengklasifikasikan sampel yang diberikan harus dihitung.

$$I(S1,S2,\ldots,S) = \sum_{i=1}^{m} \frac{Si}{S} \log_2 \frac{Si}{S}$$

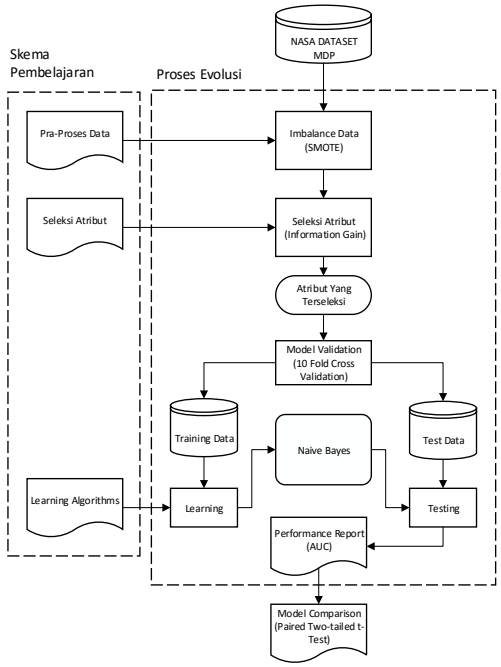
Atribut F dengan nilai $\{f1, f2, f3, \ldots, f_v\}$ dapat membagi data sebagai data training yang ditetapkan dalam subset {S1, S2, S3, ..., S v}, dima Sj merupakan subset yang memiliki nilai fj untuk F. Selanjutnya atribut Sj beisi sampel Sij kelas i. Entropy attribute F diberikan pada:

$$E(F) = -\sum_{i=1}^{m} \frac{S_{1j}+S_{2j}+S_{3j}+\cdots S_{ij}}{S} I\,(S_{1j}, S_{2j}, S_{3j}, \ldots, S_{1j})$$

Information gain untuk atribut F dapat dihitung dengan menggunakan rumus, sebagai berikut:

$$Gain\ (F) = I(S_1, S_2, \ldots S_m) - E(F)$$

Seperti yang ditunjukkan pada Gambar 1, dataset masukan termasuk dataset pelatihan dan dataset pengujian. Dimulai dengan menerapkan teknik SMOTE (Chawla et al., 2002) untuk menangani *imbalance class* pada dataset dan pembobotan menggunakan algoritma *Information Gain* (Gao & Khoshgoftaar, 2011) untuk memilih atribut yang relevan. Maka distribusi dari dataset dengan 10 metode cross validasi, dibagi menjadi data latih dan data pengujian, selanjutnya proses klasifikasi data dilakukan dengan menggunakan algoritma *Naïve Bayes*, sehingga model evaluasi menggunakan *Area Under the* ROC (*Receiver Operating Characteristics*) (AUC).



Gambar 1. Diagram Aktifitas dari Metode NB SMOTE+IG

Pengklasifikasian *Naïve Bayes* berasumsi bahwa dampak dari nilai atribut pada kelas tertentu merupakan independen dari nilai-nilai atribut lainnya. Asumsi ini disebut kelas independen bersyarat. Hal ini dilakukan menyederhanakan penghitungan yang terlibat, dan dalam pengertian ini adalah mempertimbangkan *Naïv*e. Pengklasifikasi *Naïve Bayes* memungkinkan representasi ketergantungan antar himpunan bagian dari atribut (Jain & Richariya, 2012). *Mathematically means* that:

$$(X + C_i) \quad = \sum_{k=1}^{n} P(X_k|C_j)$$

Probabilitas $P(X_1|C_1)$, $P(X_2|C_j)$, …, $P(X_n|C_i)$ dapat dengan mudah diperkirakan dari training set. Mengingat bahwa X_k mengacu pada nilai atribut untuk sampel X.

a). Jika A_k merupakan kategori, kemudian $P(X_k|C_j)$ merupakan jumlah tupel kelas C_j pada D mempunyai nilai X_k untuk atribut A_k, dibagi dari $|C_{1,D}|$, jumlah tupel kelas C_j pada D.

b). Jika A_k merupakan nilai kontinu, maka biasanya berasumsi bahwa nilai-nilai memiliki distribusi *Gaussian* dengan *mean* (μ) dan *standard deviation* (σ), dapat didefinisikan sebagai berikut:

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

Sehingga

$$P(X_k|C_j) = g(X_k, \mu_{ci}, \sigma_{ci})$$

Kita perlu menghitung μ_{ci} dan σ_{ci}, dimana mean dan standard deviation dari nilai atribut A_k untuk sampel pelatihan dari kelas C_j.

4 HASIL PENELITIAN

Percobaan dilakukan menggunakan platform komputasi berdasarkan Intel Core i5 1.6 GHz CPU, 4 GB RAM, dan Microsoft Windows 8 Professional 64-bit dengan SP1 operating system. Pengembangan *environment* menggunakan Netbeans 7 IDE, Java *programming language*, dan Weka 3.7 *library*.

Table 1. NASA MDP Datasets dan Atribut

Code Attributes		NASA MDP Dataset								
		CM1	KC1	KC3	MC2	MW1	PC1	PC2	PC3	PC4
LOC counts	LOC_total	√	√	√	√	√	√	√	√	√
	LOC_blank	√	√	√	√	√	√		√	√
	LOC_code_and_comment	√	√	√	√	√	√	√	√	√
	LOC_comments	√	√	√	√	√	√	√	√	√
	LOC_executable	√	√	√	√	√	√	√	√	√
	number_of_lines	√		√	√	√	√	√	√	√
Halstead	content	√	√	√	√	√	√	√	√	√
	difficulty	√	√	√	√	√	√	√	√	√
	effort	√	√	√	√	√	√	√	√	√
	error_est	√	√	√	√	√	√	√	√	√
	length	√	√	√	√	√	√	√	√	√
	level	√	√	√	√	√	√	√	√	√
	prog_time	√	√	√	√	√	√	√	√	√
	volume	√	√	√	√	√	√	√	√	√
	num_operands	√	√	√	√	√	√	√	√	√
	num_operators	√	√	√	√	√	√	√	√	√
	num_unique_operands	√	√	√	√	√	√	√	√	√
	num_unique_operators	√	√	√	√	√	√	√	√	√
McCabe	cyclomatic_complexity	√	√	√	√	√	√	√	√	√
	cyclomatic_density	√		√	√	√	√	√	√	√
	design_complexity	√	√	√	√	√	√	√	√	√
	essential_complexity	√	√	√	√	√	√	√	√	√
Misc.	branch_count	√	√	√	√	√	√	√	√	√
	call_pairs	√		√	√	√	√	√	√	√
	condition_count	√		√	√	√	√	√	√	√
	decision_count	√		√	√	√	√	√	√	√
	decision_density	√		√	√	√	√	√	√	√
	edge_count	√		√	√	√	√	√	√	√
	essential_density	√		√	√	√	√	√	√	√
	parameter_count	√		√	√	√	√	√	√	√
	maintenance_severity	√		√	√	√	√	√	√	√
	modified_condition_count	√		√	√	√	√	√	√	√
	multiple_condition_count	√		√	√	√	√	√	√	√
	global_data_complexity			√	√					
	global_data_density			√	√					
	normalized_cyclo_complx	√		√	√	√	√	√	√	√
	percent_comments	√		√	√	√	√	√	√	√
	node_count	√		√	√	√	√	√	√	√
Programming Language		*C*	*C++*	*Java*	*C*	*C*	*C*	*C*	*C*	*C*
Number of Code Attributes		*37*	*21*	*39*	*39*	*37*	*37*	*36*	*37*	*37*
Number of Modules		*344*	*2096*	*200*	*127*	*264*	*759*	*1585*	*1125*	*1399*
Number of fp Modules		*42*	*325*	*36*	*44*	*27*	*61*	*16*	*140*	*178*
Percentage of fp Modules		*12.21*	*15.51*	*18*	*34.65*	*10.23*	*8.04*	*1.01*	*12.44*	*12.72*

Pada penelitian ini menggunakan 9 dataset software defect dari NASA MDP (Wahono & Suryana, 2013). Atribut individu per dataset, bersama-sama dengan beberapa statistik umum dan deskripsi, yang jelaskan pada Tabel 1. Dataset ini memiliki berbagai skala (Shepperd, Song, Sun, & Mair, 2013) *Line of Code* (LOC), berbagai modul software dikodekan oleh beberapa bahasa pemrograman yang berbeda, termasuk C, C++ dan Java, dan berbagai jenis kode metrik, termasuk ukuran kode, Halstead's complexity dan McCabe's cyclomatic complexity (Mccabe, 1976).

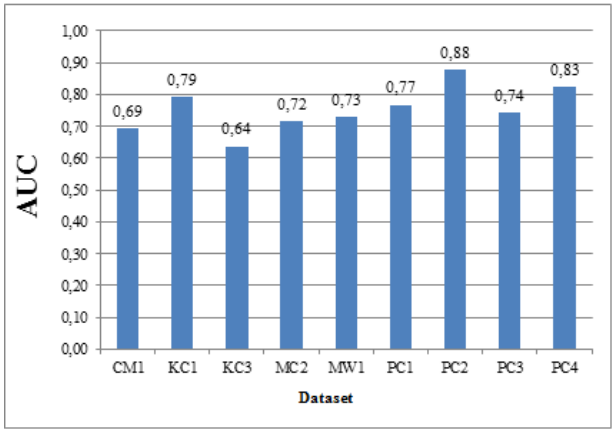
State-of-the-art menggunakan 10-fold cross-validation (Kohavi & Edu, 1995) untuk data pembelajaran dan pengujian. Ini berarti bahwa kami membagi data pelatihan menjadi 10 bagian yang sama dan kemudian dilakukan proses belajar 10 kali. Kami menerapkan 10-*fold cross validation*, karena metode ini telah menjadi metode standar dalam hal praktis.

Kurva ROC (*Receiver Operating Characteristics*) telah diperkenalkan untuk mengevaluasi kinerja algoritma classifier. *Area Under the* ROC (AUC) memberikan ringkasan untuk kinerja algoritma *classifier*. Lessmann et al. (Lessmann et al., 2008) menganjurkan penggunaan AUC untuk meningkatkan cross-studi komparatif. AUC (Ling, 2003) adalah pengukuran nilai tunggal yang berasal dari deteksi sinyal. Nilai AUC berkisar dari 0 sampai 1. Kurva ROC digunakan untuk mengkarakterisasi *trade-off* antara *true positive rate* (TPR) and *false positive rate* (FPR). Sebuah classifier yang menyediakan area yang luas di bawah kurva lebih dari classifier dengan area yang lebih kecil di bawah kurva (Khoshgoftaar & Gao, 2009). Penelitian yang dilakukan oleh Ling dan Zhang membuktikan bahwa AUC secara statistik lebih konsisten dan akurasi diskriminatif. AUC yang merupakan pengukuran yang lebih baik dari akurasi dalam mengevaluasi dan membandingkan kinerja algoritma *classifier* (Ling & Zhang, 2003).

Pertama-tama, kami melakukan percobaan pada 9 NASA MDP dataset oleh NB classifier. Hasil eksperimen dilaporkan dalam Tabel 2 dan Gambar 2. Model NB tampil prima pada dataset PC2, baik pada dataset PC4, cukup di KC1, MC2, MW1, PC1, dataset PC3, tapi sayangnya buruk pada dataset CM1 dan KC3.

Tabel 2. AUC Model NB Model Pada 9 Datasets

Klasifikasi	CM1	KC1	KC3	MC2	MW1	PC1	PC2	PC3	PC4
Naive Bayes	0,69	0,79	0,64	0,72	0,73	0,77	0,88	0,74	0,83

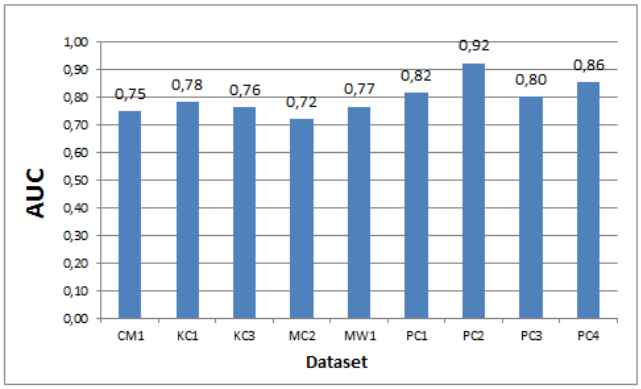


Gambar 2. AUC Model NB Model Pada 9 Datasets

Pada percobaan berikutnya, kami menerapkan model NB SMOTE+IG pada 9 dataset NASA MDP. Hasil percobaan dapat dilihat pada Tabel 3 dan Gambar 3. Model yang memiliki hasil lebih baik dicetak tebal. Model NB SMOTE + IG tampil prima pada dataset PC2, baik pada PC1, PC3 dan PC4 dataset, dan cukup pada dataset lainnya. Hasil penelitian menunjukkan bahwa tidak ada hasil yang buruk ketika model NB SMOTE + IG diterapkan.

Tabel 3. AUC Model NB SMOTE+IG Pada 9 Datasets

Klasifikasi	CM1	KC1	KC3	MC2	MW1	PC1	PC2	PC3	PC4
NB SMOTE + IG	0,75	0,78	0,76	0,72	0,77	0,82	0,92	0,80	0,86

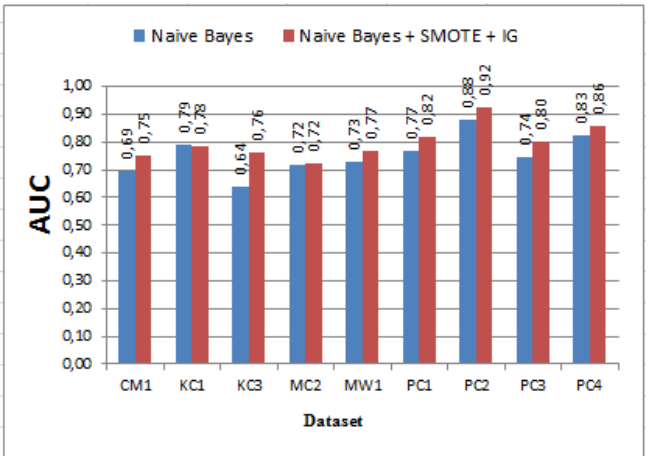


Gambar 3. AUC Model NB SMOTE+IG Pada 9 Datasets

Tabel 4 dan Gambar 4 menunjukkan perbandingan AUC model NB diuji pada 9 NASA MDP dataset. Seperti yang ditunjukkan pada Tabel 4 dan Gambar 4, meskipun model NB SMOTE pada dataset KC1, MC2 tidak memiliki peningkatan akurasi, namun pada dataset CM1, KC3, MW1, PC1, PC2, PC3, dan PC4 mengungguli metode NB asli. Hal ini menunjukkan bahwa integrasi teknik SMOTE dan algoritma *Information Gain* efektif untuk meningkatkan secara signifikan kinerja dari klasifikasi NB.

Table 4. AUC Perbandingan antara Model NB dan Model NB SMOTE+IG

Klasifikasi	CM1	KC1	KC3	MC2	MW1	PC1	PC2	PC3	PC4
NB	0,69	0,79	0,64	0,72	0,73	0,77	0,88	0,74	0,83
NB SMOTE + IG	**0,75**	0,78	**0,76**	0,72	**0,77**	**0,82**	**0,92**	**0,80**	**0,86**



Gambar 4. AUC Perbandingan antara Model NB dan Model NB SMOTE+IG

Akhirnya, untuk melakukan verifikasi apakah terdapat perbedaan yang signifikan antara NB dan metode yang diusulkan NB SMOTE+IG, maka hasil dari kedua metode dibandingkan. Kami melakukan statistik Wilcoxon-Test (2-tailed) (Wilcoxon, 1945)(Demsar, 2006) untuk pasangan antara model NB dan NB SMOTE+ IG pada setiap dataset. Pada signikasi statistik pengujian nila p merupakan peluang mendapatkan uji statistik yang salah satunya lebih ektrim apabila benar-benar diamati, dengan asumsi bahwa hipotesis nol benar. Salah satu sering menolak hipotesis nol, ketika nilai p kurang dari tingkat signifikan yang telah ditentukan (α), menunjukkan bahwa hasil diamati akan sangat tidak mungkin di bawah hipotesis nol. Dalam hal ini, kita mengatur tingkat signifikansi statistik (α) menjadi 0,05. Ini berarti bahwa tidak

ada perbedaan yang signifikan secara statistik jika nilai p > 0,05.

Hasil yang ditunjukkan pada Tabel 5. Nilai $p = 0{,}013$ ($p < 0{,}05$), berarti ada perbedaan yang signifikan secara statistik antara model NB dan model NB SMOTE+IG. Dapat disimpulkan bahwa integrasi teknik SMOTE dan algoritma *Information Gain* berdasarkan NB mencapai kinerja yang lebih baik pada prediksi cacat software.

Tabel 5. *Wilcoxon Test* Model NB dan Model NB SMOTE+IG

Ranks

		N	*Mean Rank*	*Sum of Ranks*
NB SMOTE+IG	*Negative Ranks*	1[a]	1.50	1.50
	Positive Ranks	8[b]	5.44	43.50
	Ties	0[c]		
	Total	9		

Test Statistics[b]

	NB IG+SMOTE - NB
Z	-2.490[a]
Asymp. Sig. (2-tailed)	.013

5 KESIMPULAN

Kombinasi teknik SMOTE dan *algoritma Information Gain* diusulkan untuk meningkatkan kinerja pada prediksi cacat *software*. SMOTE diterapkan untuk menangani *imbalance class*. Sedangkan algoritma *Information Gain* digunakan untuk proses pemilihan atribut yang relevan untuk menangani noise atribut. Model yang diusulkan diterapkan pada 9 dataset NASA MDP pada prediksi cacat software. Hasil penelitian menunjukkan bahwa model yang diusulkan mencapai akurasi klasifikasi yang lebih tinggi. Dimana nilai rata-rata AUC pada model NB SMOTE+IG adalah 0.798, dimana mengungguli model NB yang memiliki nilai rata-rata AUC adalah 0.753. Dan hasil uji *friedman* menunjukkan bahwa NB SMOTE+IG memiliki perbedaan yang signifikan dengan NB yaitu memiliki nilai *p* adalah 0.02, bahwa nilai p < 0.05. Oleh karena itu, dapat disimpulkan bahwa model yang diusulkan yaitu NB SMOTE+IG meningkatkan kinerja dari *Naive Bayes* pada prediksi cacat *software*.

REFERENSI

Catal, C. (2011). Software fault prediction: A literature review and current trends. *Expert Systems with Applications*, *38*(4), 4626–4636. doi:10.1016/j.eswa.2010.10.024

Chawla, N. V, Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE : Synthetic Minority Over-sampling Technique, *16*, 321–357.

De Carvalho, A. B., Pozo, A., & Vergilio, S. R. (2010). A symbolic fault-prediction model based on multiobjective particle swarm optimization. *Journal of Systems and Software*, *83*(5), 868–882.

Demsar, J. (2006). Statistical Comparisons of Classifiers over Multiple Data Sets. *The Journal of Machine Learning Research*, *7*, 1–30.

Domingos, P. (1997). On the Optimality of the Simple Bayesian Classifier under Zero-One Loss. *Machine Learning*, *29*(2-3), 103–130.

Gao, K., & Khoshgoftaar, T. M. (2011). Software Defect Prediction for High-Dimensional and Class-Imbalanced Data. *Conference: Proceedings of the 23rd International Conference on Software Engineering & Knowledge Engineering*, (2).

Guyon, I. (2003). An Introduction to Variable and Feature Selection 1 Introduction. *Journal of Machine Learning Research*, *3*, 1157–1182.

Hall, T., Beecham, S., Bowes, D., Gray, D., & Counsell, S. (2010). A Systematic Literature Review on Fault Prediction Performance in Software Engineering. *IEEE Transactions on Knowledge and Data Engineering*, *38*(6), 1276 – 1304.

Jain, M., & Richariya, V. (2012). An Improved Techniques Based on Naive Bayesian for Attack Detection. *International Journal of Emerging Technology and Advanced Engineering*, *2*(1), 324–331.

Kabir, M., & Murase, K. (2012). Expert Systems with Applications A new hybrid ant colony optimization algorithm for feature selection. *Expert Systems With Applications*, *39*(3), 3747–3763.

Khoshgoftaar, T. M., & Gao, K. (2009). Feature Selection with Imbalanced Data for Software Defect Prediction. *2009 International Conference on Machine Learning and Applications*, 235–240.

Kohavi, R., & Edu, S. (1995). A Study of Cross-Validation and Bootstrap for Accuracy Estimation and M o d e l Selection. *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI)*, 1137–1143.

Lessmann, S., Member, S., Baesens, B., Mues, C., & Pietsch, S. (2008). Benchmarking Classification Models for Software Defect Prediction : A Proposed Framework and Novel Findings. *IEEE Transactions on Software Engineering*, *34*(4), 485–496.

Ling, C. X. (2003). Using AUC and Accuracy in Evaluating Learning Algorithms, 1–31.

Ling, C. X., & Zhang, H. (2003). AUC: a statistically consistent and more discriminating measure than accuracy. *Proceedings of the 18th International Joint Conference on Artificial Intelligence*.

Mccabe, T. J. (1976). A Complexity Measure. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING,*, *SE-2*(4), 308–320.

Menzies, T., Greenwald, J., & Frank, A. (2007). Data Mining Static Code Attributes to Learn Defect Predictors. *IEEE Transactions on Software Engineering*, *33*(1), 2–13.

Riquelme, J. C., Ruiz, R., & Moreno, J. (2008). Finding Defective Modules from Highly Unbalanced Datasets. *Engineering*, *2*(1), 67–74.

Rivera, J., & Meulen, R. van der. (2014). Gartner Says Worldwide IT Spending on Pace to Reach $3.8 Trillion in 2014. Retrieved August 01, 2015, from http://www.gartner.com/newsroom/id/2643919

Shepperd, M., Song, Q., Sun, Z., & Mair, C. (2013). Data Quality : Some Comments on the NASA Software Defect Data Sets. *Software Engineering, IEEE Transactions*, *39*(9), 1–13.

Song, Q., Jia, Z., Shepperd, M., Ying, S., & Liu, J. (2011). A General Software Defect-Proneness Prediction Framework. *IEEE Transactions on Software Engineering*, *37*(3), 356–370.

Turhan, B., & Bener, A. (2009). Analysis of Naive Bayes' assumptions on software fault data: An empirical study. *Data & Knowledge Engineering*, *68*(2), 278–290.

Wahono, R. S., & Suryana, N. (2013). Combining Particle Swarm Optimization based Feature Selection and Bagging Technique for Software Defect Prediction. *International Journal of Software Engineering and Its Applications*, *7*(5), 153–166.

Wang, H., Khoshgoftaar, T. M., Gao, K., & Seliya, N. (2009). High-Dimensional Software Engineering Data and Feature Selection. *Proceedings of 21st IEEE International Conference on Tools with Artificial Intelligence, Nov. 2-5*, 83–90.

Wilcoxon, F. (1945). Individual Comparisons by Ranking Methods. *International Biometric Society Stable*, *1*(6), 80–83.

Yap, B. W., Rani, K. A., Aryani, H., Rahman, A., Fong, S., Khairudin, Z., & Abdullah, N. N. (2014). An Application of Oversampling, Undersampling, Bagging and Boosting in Handling Imbalanced Datasets. *Proceedings of the First International Conference on Advanced Data and Information Engineering (DaEng-2013)*, *285*, 13–23.

BIOGRAFI PENULIS

Sukmawati Anggraini Putri. Memperoleh gelar S.Kom pada bidang Sistem Informasi dari Pascasarjana STMIK Nusa Mandiri, Jakarta dan gelar M.Kom dari STMIK Nusa Mandiri Jakarta. Dia sebagai dosen tetap di STMIK Nusa Mandiri. Minat penelitiannya saat ini meliputi rekayasa perangkat lunak dan *machine learning*.

Romi Satria Wahono. Memperoleh gelar B.Eng dan M.Eng pada bidang ilmu komputer di Saitama University Japan, dan Ph.D pada bidang software engineering di Universiti Teknikal Malaysia Melaka. Pengajar dan peneliti di Fakultas Ilmu Komputer, Universitas Dian Nuswantoro. Pendiri dan CEO PT Brainmatics, perusahaan yang bergerak di bidang pengembangan software. Minat penelitian pada bidang software engineering dan machine learning. Profesional member dari asosiai ilmiah ACM, PMI dan IEEE Computer Society.