INTEGRASI TEKNIK SMOTE BAGGING DENGAN INFORMATION GAIN PADA NAIVE BAYES UNTUK PREDIKSI CACAT SOFTWARE

Sukmawati Anggraeni Putri
Program Studi Sistem Informasi
STMIK Nusa Mandiri Jakarta
sukmawati@nusamandiri.ac.id

Abstract— *The prediction accuracy of defects in code, can help direct the test effort, reduce costs and improve software quality. Until now, many researchers have applied various types of algorithm based on machine learning and statistical methods to build predictive performance software defects. One of them uses machine learning approach to the classification, which is a popular approach to predict software defects. While Naive Bayes one simple classification to have good performance that produces an average probability of 71 percent. As well as the time required in the process of learning faster than on any other machine learning. Additionally it has a good reputation on the accuracy of the prediction. While NASA MDP is a very popular data used by previous researchers in the development of predictive models of software defects. Because it is common and freely used by researchers. However, these data have deficiencies, including the occurrence of imbalance class and attribute noise. Therefore by using SMOTE (Synthetic Minority Over-Sampling Technique) for sampling techniques and Bagging on the ensemble method, is used to deal with the class imbalance. As for dealing with noise attribute, in this research using information gain in the process of selecting the relevant attributes. So after the trial that the application of the model SMOTE Bagging and Information Gain proven to obtain good results to handled imbalance class and attribute noise at prediction software defects, and can increase the accuracy of the prediction results software defects.*

Key word: Prediction Software Defect, Information Gain, Naïve Bayes, SMOTE, Bagging

Intisari—Akurasi prediksi cacat pada kode, dapat membantu upaya tes langsung, mengurangi biaya dan meningkatkan kualitas perangkat lunak. Sampai saat ini, banyak peneliti telah menerapkan berbagai jenis algoritma berdasarkan *learning machine* dan metode statistik untuk membangun kinerja prediksi cacat software. Salah satunya menggunakan pendekatan klasifikasi pada learning machine, yang merupakan pendekatan populer untuk memprediksi cacat software. Sementara *Naive Bayes* salah satu klasifikasi sederhana dengan memiliki kinerja baik yang menghasilkan probabilitas rata-rata 71 persen. Serta waktu yang dibutuhkan dalam proses pembelajar lebih cepat dari pada pembelajaran mesin lain. Selain itu memiliki reputasi yang baik pada keakuratan prediksi. Sedangkan NASA MDP merupakan data yang sangat popular digunakan oleh para peneliti sebelumnya dalam pengembangan model prediksi cacat pada software. Karena sifatnya yang umum dan bebas digunakan oleh para peneliti. Namun data tersebut memiliki kekurangan, diantaranya terjadinya imbalance class dan noise attribute. Oleh karenanya dengan menggunakan SMOTE (Minority Synthetic Over-Sampling Technique) untuk teknik sampling, sedangan Bagging pada metode ensemble, berfungsi untuk menangani ketidaseimbangan kelas tersebut. Sedangkan untuk menangani noise attribute, pada penilitian ini menggunakan metode information gain pada proses pemilihan atribut yang relevan. Sehingga setelah dilakukan percobaan bahwa penerapan model SMOTE Bagging dan Information Gain terbukti memperoleh hasil yang baik untuk menangani *imbalance class* dan *noise attribute* untuk prediksi cacat *software*, serta dapat meningkatkan hasil akurasi dari prediksi cacat pada software.

Kata Kunci: Prediksi Cacat *Software*, *Information Gain*, *Naive Bayes*, SMOTE, Bagging

PENDAHULUAN

Akurasi prediksi dimana kesalahan mungkin terjadi pada kode dapat membantu upaya tes langsung, mengurangi biaya dan meningkatkan kualitas perangkat lunak. Sampai saat ini, banyak peneliti telah menerapkan berbagai jenis algoritma berdasarkan machine learning dan metode statistik untuk membangung kinerja prediksi cacat software.

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).

Sedangkan pendekatan klasifikasi pada machine learning merupakan pendekatan populer untuk memprediksi cacat software (Lessmann, Member, Baesens, Mues, & Pietsch, 2008). Seperti, *Logistic Regression* (Lessmann et al., 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, Beecham, Bowes, Gray, & Counsell, 2010). Sehingga pada penelitian ini fokus pada penggunaan algoritma Naive Bayes.

Naive Bayes merupakan klasifikasi sederhana (Domingos, 1997) dengan memiliki kinerja baik yang menghasilkan probabilitas rata-rata 71 persen. Serta 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 penelitian sebelumnya hampir semua menggunakan dataset NASA MDP dan PROMISE yang bersifat publik (Hall et al., 2010). Dan ditemukan dua masalah utama yang dihadapi dari penggunaan dataset tersebut, diantarnya *high dimensionality* dan *class imbalance* (Wahono & Suryana, 2013).

Imbalance class dapat menyebabkan kesulitan pembelajaran pada machine learning, karena kebanyakan kasus akan diprediksi sebagai bukan rawan cacat (Khoshgoftaar & Gao, 2009). Sedangkan untuk menangani *imbalance class* terdapat dua pendekatan yang populer digunakan oleh penelitian sebelumnya, yaitu pendeketan level (*sampling technique*) dan metode ensemble (*ensemble learning*) (Yap et al., 2014).

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). Sedangkan 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).

Sedangkan pada *ensemble learning* terdapat dua algoritma yang populer yaitu *boosting* dan *bagging*. *Bagging* merupakan singkatkan dari *bootstrap aggregating*, dimana sampel *bootstrap* diambil secara acak dengan penggantian. Sementara itu, *boosting* mencoba untuk meningkatkan akurasi *classifier* dengan bobot ulang sampel yang salah diklasifikasikan. Sehingga pada penelitian sebelumnya (Wahono & Suryana, 2013), menyatakan bahwa *bagging* lebih baik dibandingkan dengan *boosting*.

High dimensionality akan menyebabkan noise attribute oleh karenanya pemilihan atribut baik untuk menangani high dimensionality maupun noise atrribute (Gao, Khoshgoftaar, Wang, & Seliya, 2011). 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 integrasi antara algoritma *Information Gain* (IG), teknik SMOTE, dan *bagging* untuk meningkatkan prediksi cacat *software* pada machine learning Naive Bayes. Penerapan Information Gain untuk menangani *noise attribute*, sedangakan teknik SMOTE dan *bagging* 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). Sedangkan bagging pada *ensamble learning* memperlihatkan hasil dengan baik untuk menangani class imbalance (Wahono & Suryana, 2013).

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.

BAHAN DAN METODE

1. 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 menggunakaan *Information Gain*. Hasil ini secara signifikan mengungguli *Rule Induction (J.48 dan OneR).*

Penelitian yang dilakukan oleh Lessmaan pada tahun 2008 (Lessmann et al., 2008), dimana mereka meneliti sebanyak 22 pengklasifikasi pada prediksi cacat perangkat lunak, yang terdiri dari LDA, QDA, LogReg, NB, Bayes Net, LARS, RVM, k-NN, K*, MLP-1, MLP-2, RBF net, SVM, L-SWM, LS-SVM, LP, VP, C4.5, CART, ADT, RndFor, LMT, dengan menggunakan

dataset NASA MDP. AUC direkomendasikan sebagai indikator akurasi utama untuk studi banding pada prediksi cacat perangkat lunak. Keseluruhan tingkat akurasi dari prediksi semua klasifikasi dinyatakan layak untuk mengidentifikasi modul *fault prone*.

Penelitian yang dilakukan oleh Riquelme sebelumnya pada tahun 2008 (Riquelme et al., 2008), dimana mereka meneliti *Resample* dan SMOTE (*Synthetic Minority Oversampling Technique*) dengan dua algoritma klasifikasi yaitu *Naïve Bayes* dan C4.5 dalam menangani class imbalance. Hasil dari penelitian tersebut menyatakan bahwa teknik SMOTE dapat meningkatkan hasil AUC. Selain itu juga menyatakan bahwa *Naïve Bayes* lebih baik dari J48, dengan nilai AUC tertinggi sebesar 87 persen.

Dataset NASA MDP yang telah digunakan oleh para peneliti sebelumnya pada bidang prediksi cacat software merupakan dataset berskala besar dan berdimensi tinggi (H. Wang, Khoshgoftaar, Gao, & Seliya, 2009). Sehingga menimbulkan masalah *imbalance class* dan *noise attribute* (Wahono & Suryana, 2013).

Penelitian yang dilakukan oleh Wahono (Wahono & Suryana, 2013) menggunakan metode *optimization metaheuristics* (*genetic algorithm* dan *Particle Swarm Optimization* (PSO)) pada pemilihan atribut dan teknik *Bagging* untuk menangani *class imbalance*, sehingga dapat meningkatkan kinerja prediksi cacat perangkat lunak.

Pada 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.

Sementara penelitian yang dilakukan oleh (Putri, 2015), mengintegrasikan teknik SMOTE dan algoritma *Information Gain* menyatakan bahwa dapat menangani *imbalance class* dan *noise*, sehingga dapat meningkatkan kinerja prediksi cacat perangkat lunak.

Masalah *class imbalance* diamati di berbagai domain, termasuk perangkat lunak prediksi cacat. Beberapa metode telah diusulkan dalam literatur untuk menangani *class imbalance*: 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). Teknik SMOTE juga dapat diintegrasikan menggunakan teknik pembelajaran *ensemble* seperti Bagging (Wahono & Suryana, 2013).

Pada penelitian ini, melakukan integrasi antara teknik SMOTE dan Bagging untuk menangani masalah *class imblance*. Sedangkan pada seleksi atribut untuk menangani noise atribut akan menggunakan algoritma *Information Gain* (IG. Sementara machine learning yang telah digunakan pada penelitian prediksi cacat software sebelumnya *Naive Bayes* (NB) terbukti dapat menangani *noise attibute* dan *class imbalance* dengan baik. Sehingga pada penelitian ini menggunakan ketiga teknik tersebut secara bersama-sama pada *learning machine* Naive Bayes untuk memprediksi cacat *software*.

2. Metode

Penelitian ini akan mengusulkan penggunaan learning machine Naive Bayes (NB) dengan mengintegrasikan antara teknik SMOTE dan Bagging serta algoritma Information Gain (IG), untuk menghasilkan kinerja prediksi cacat software yang lebih baik dari para peneliti sebelumnya. Seperti Gambar 1 yang menunjukkan *activity diagram* dari metode yang diusulkan yaitu NB SMOTEBagging + IG.

A. Teknik Sampling

Tugas teknik SMOTE untuk menangani *class imbalance*. Penggunaan teknik SMOTE (*Synthetic Minority Over-Sampling Technique*) (Chawla et al., 2002) menghasilkan hasil yang baik dan efektif untuk menangani *class imbalance* 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).

Di bawah ini akan dijelaskan algoritma SMOTE dalam bentuk *pseudocode* (Chawla et al., 2002). Algoritma SMOTE terdiri dari dua bagian utama, bagian pertama berisi perulangan untuk mencari *k* tetangga terdekat dan bagian kedua untuk membuat sintesis data kelas minoritas.

Masukan: Jumlah kelas minoritas pada sampel; jumlah SMOTE N%; jumlah *nearest neighbors k*.

Keluaran: (*N*/100) * *T* sampel kelas sintetis minoritas

1. /* Jika *N* adalah kurang dari 100%, sampel random kelas minoritas hanya persentase random merupakan SMOTE. */

2. ***If*** N < 100
3. ***then*** random dari sampel T kelas minoritas
4. $T = (N/100) * T$
5. $N = 100$
6. ***endif***
7. *N = (int)(N/100)* /* jumlah SMOTE diasumsikan kelipatan integral dari 100 */
8. *k* = jumlah *nearest neighbors*
9. *numattrs* = jumlah atribut
10. *Sample [ ][ ]: array* dari sampel asli kelas minoritas
11. *newindex*: membuat hitungan jumlah sampel sintetik yang dihasilkan, diinisialisasi menjadi 0
12. *Synthetic* [ ] [ ]: *array* untuk sampel sintetik /* menghitung *k nearest neighbors* untuk masing-masing hanya sampel kelas minoritas. */
13. ***for*** *i* ← 1 to *T*
14. Menghitung *k nearest neighbors* untuk *i*, dan menyimpan *index* di *nnarray*
15. Populasi (*N*, *i*, *nnarray*) /* fungsi untuk menghasilkan sampel sintetik. */
16. ***endfor***
17. ***while*** N ≠ 0
18. Memilih jumlah random antara 1 dan k, yang disebut nn. Pada tahap ini memilih 1 *k nearest neighbors* dari *i*
19. ***for*** *attr* ← 1 to *numattrs*
20. Menghitung : *dif* = Sampel[*nnarray[nn][attr]* – sampel *[i][attr]*
21. Menghitung : *gap* = jumlah random antara 0 dan 1
22. *Synthetic[newindex[attr] = sample[i][attr] + gap * dif*
23. ***endfor***
24. *newindex++*
25. *N = N – 1*
26. ***endwhile***
27. ***return*** (* akhir populasi.*)
28. Akhir dari *Pseudo-Code*

ngkan, dalam pendekatan bagging, semua contoh di dataset pelatihan memiliki peluang yang sama untuk dipilih. Semua sampel ulangan berdasarkan pendekatan bootstrap. Ulangan adalah sampel yang diambil dengan penggantian dan dengan ukuran yang sama dengan sampel pelatihan. Untuk setiap set bootstrap, satu model yang dilengkapi. Prediksi akhir dari kasus dihasilkan menggunakan pendekatan voting (Wahono & Suryana, 2013).

B. Ensamble Learning

Langkah-langkah yang terlibat dalam proses *bagging* (Yap et al., 2014) yaitu sebagai berikut:

1. Untuk iterasi t=1,2, ..., T: #dengan menggunakan T=10
2. Memilih acak dataset N sampel dari pelatingan asli dengan penggantian
3. Memperoleh pempelajar, ƒ(x) (model prediktif atau classifier) dari dataset ulang sampel.
4. Menggunakan model, ƒ (x), memprediksi kasus
5. Mengkombinasikan semua model prediksi $f^t(x)$ menjadi model agregat $f^A(x)$.
6. Menggunakan pendekatan voting, mengembalikan kelas yang telah diprediksi paling sering.

C. Seleksi Atribut

Sedangkan untuk menangani noise atribute proses seleksi atribut (pilihan fitur) pilihan yang baik. Serta dari algoritma yang telah digunakan oleh penenilitian sebelumnya, *information gain* menunjukkan hasil yang lebih baik (Gao et al., 2011). 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:

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

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

$$Gain\,(F) = I(S_1, S_2, \ldots S_m) - E(F) \quad \ldots\ldots (2)$$

D. Metode Klasifikasi

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. *Naïve Bayes* memungkinkan representasi ketergantungan antar himpunan bagian dari atribut (Jain & Richariya, 2012). *Mathematically means* that:

$$(X + C_i) \quad = \sum_{k=1}^{n} P(X_k|C_j) \quad \ldots\ldots (3)$$

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:

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad \ldots\ldots (4)$$

Sehingga

$$P(X_k|C_j) = g(X_k, \mu_{ci}, \sigma_{ci}) \quad \ldots\ldots (5)$$

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.

E. Teknik Evaluasi dan Validasi

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. 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) menyarankan menggunakan AUC untuk meningkatkan cross-studi komparatif.

AUC (Ling, 2003) merupakan 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).

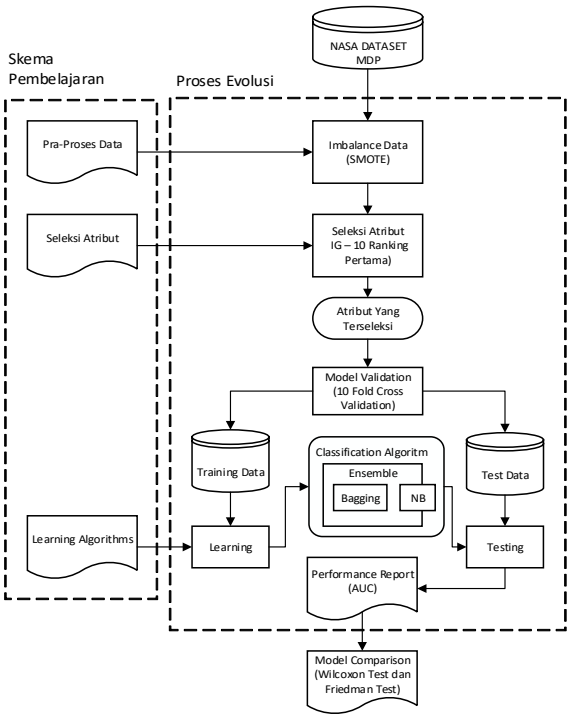
Penelitian yang dilakukan oleh Ling dan Zhang membuktikan bahwa AUC secara statistik lebih konsisten dan akurasi diskriminatif. AUC yang merupakan pengukuran yang lebih baik dari akurasi dalam mengevaluasi dan membandingkan kinerja algoritma *classifier* (Ling & Zhang, 2003).

HASIL DAN PEMBAHASAN

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*.

A. Metode Usulan

Seperti yang dijelaskan pada Gambar 1, dataset masukan terbagi menjadi dua bagian diantaranya dataset pelatihan dan dataset pengujian. Proses dimulai dengan menerapkan teknik SMOTE dan bagging untuk menangani *class imbalance* pada dataset. Selanjutnya proses pembobotan menggunakan algoritma *Information Gain* untuk memilih atribut yang relevan. Kemudian dataset yang telah siap, didistribusikan dengan menggunakan 10 metode cross validasi yang akan dibagi menjadi data latih dan data uji. Kemudian proses klasifikasi data dilakukan dengan menggunakan algoritma *learning machine Naïve Bayes*, yang dieksekusi dengan menggunakan model *Area Under the* ROC (*Receiver Operating Characteristics*) (AUC) sebagai model evaluasi akhir.



Sumber: Hasil Analisa (2016)

Gambar 1. Diagram Aktifitas dari Metode NB SMOTEBagging + IG

B. Data Set

Pada penelitian ini menggunakan 9 dataset *software defect* dari NASA *Metrics Data Program* (MDP) repository yang bersifat umum dan dapat digunakan bebas oleh para peneliti (Wahono & Suryana, 2013). Dataset NASA MDP yang dapat diakses bebas oleh umum dan dapat diperoleh melalui halaman *website* wikispaces (http://nasa-softwaredefectdatasets.wikispaces.com/space/content).

Dataset NASA MDP yang terdiri dari CM1 merupakan instrumen pesawat ruang angkasa, KC1 dan KC3 merupakan manajemen penyimpanan data tanah, MC2 merupakan sistem panduan video, MW1 merupakan sistem pencitraan gambar, PC1 merupakan *software* penerbangan untuk satelit yang mengorbit bumi, PC2 merupakan simulator dimanis untuk sistem kontrol perilaku, PC3 dan PC4 merupakan *software* penerbangan untuk satelit yang mengorbit bumi (T. Wang, Li, Shi, & Liu, 2011).

Deskripsi setiap datasetnya dijelaskan pada pada Tabel 1. Dataset tersebut 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).

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*

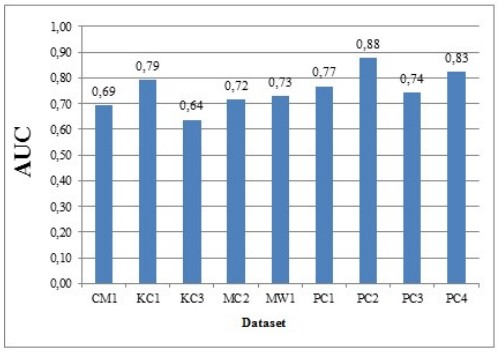
Sumber: Hasil Analisa (2016)

C. Pelaksanaan dan Hasil Eksperimen

Pertama-tama, kami melakukan percobaan pada 9 NASA MDP dataset oleh *Naive Bayes* (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 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



Sumber: Hasil Analisa (2016)

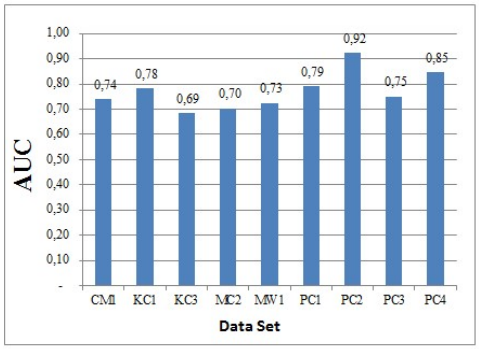
Gambar 2. AUC Model NB Model Pada 9 Datasets

Pada percobaan berikutnya, kami menerapkan model NB *Information Gain* (IG) pada 9 dataset NASA MDP. Hasil percobaan dapat dilihat pada Tabel 3 dan Gambar 3. Model NB IG tampil prima pada dataset PC2, hasil yang baik pada dataset PC4, hasil yang cukup pada dataset CM1, KC1, MC2, MW1, PC1, dan PC3. Namun pada dataset KC3 menghasilkan hasil yang buruk.

Tabel 3. AUC Model NB IG Pada 9 Datasets

Klasifikasi	CM1	KC1	KC3	MC2	MW1	PC1	PC2	PC3	PC4
NB IG	0,74	0,78	0,69	0,70	0,73	0,79	**0,92**	0,75	0,85

Sumber: Hasil Analisa (2016)



Sumber: Hasil Analisa (2016)

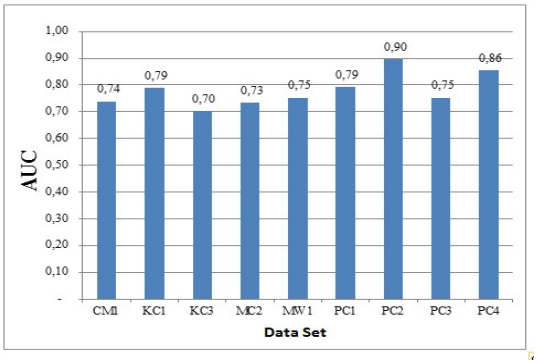
Gambar 3. AUC Model NB IG Pada 9 Datasets

Pada percobaan berikutnya, kami menerapkan model NB SMOTE pada 9 dataset NASA MDP. Hasil percobaan dapat dilihat pada Tabel 4 dan Gambar 4. Model NB SMOTE tampil prima pada dataset PC2, hasil yang baik pada dataset PC4, dan pada dataset yang lainnya menghasilkan nilai yang cukup baik. Sehingga pada model NB SMOTE tidak ada dataset yang menghasilkan nilai yang buruk.

Tabel 4. AUC Model NB SMOTE Pada 9 Datasets

Klasifikasi	CM1	KC1	KC3	MC2	MW1	PC1	PC2	PC3	PC4
NB SMOTE	0,74	0,79	0,7	0,73	0,75	0,79	0,9	0,75	0,86

Sumber: Hasil Analisa (2016)



Sumber: Hasil Analisa (2016)

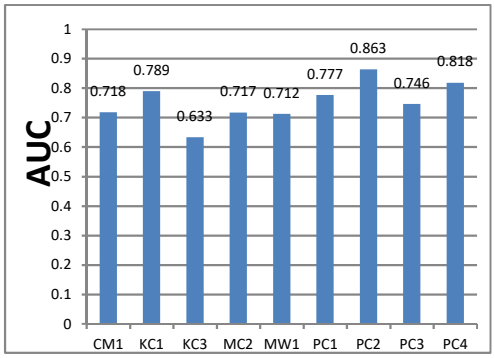
Gambar 4. AUC Model NB SMOTE Pada 9 Datasets

Pada percobaan berikutnya, kami menerapkan model NB SMOTE Bagging pada 9 dataset NASA MDP. Hasil percobaan dapat dilihat pada Tabel 5 dan Gambar 5. Pada model NB SMOTE Bagging tampil prima pada dataset PC2, baik pada KC1, PC1, dan PC4 dataset, dan cukup pada dataset lainnya. Hasil penelitian menunjukkan bahwa tidak ada hasil yang buruk ketika model NB SMOTE + IG diterapkan.

Tabel 5. AUC Model NB Bagging Pada 9 Datasets

Klasifikasi	CM1	KC1	KC3	MC2	MW1	PC1	PC2	PC3	PC4
NB + Bagging	0,72	0,79	0,63	0,72	0,71	0,78	**0,86**	0,75	0,82

Sumber: Hasil Analisa (2016)



Sumber: Hasil Analisa (2016)

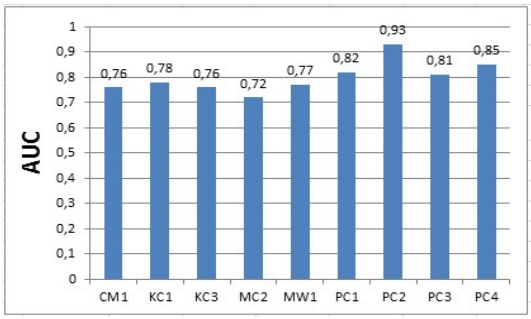
Gambar 5. AUC Model NB Bagging Pada 9 Datasets

Pada percobaan berikutnya, kami menerapkan model NB SMOTEBagging + IG pada 9 dataset NASA MDP. Hasil percobaan dapat dilihat pada Tabel 6 dan Gambar 6. Model NB SMOTEBagging + 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 SMOTEBagging + IG diterapkan.

Tabel 6. AUC Model NB SMOTEBagging +IG Pada 9 Datasets

Klasifikasi	CM1	KC1	KC3	MC2	MW1	PC1	PC2	PC3	PC4
NB SMOTEBagging + IG	0,76	0,78	0,76	0,72	0,77	0,82	0,93	0,81	0,85

Sumber: Hasil Analisa (2016)



Sumber: Hasil Analisa (2016)

Gambar 6. AUC Model NB SMOTEBagging + IG Pada 9 Datasets

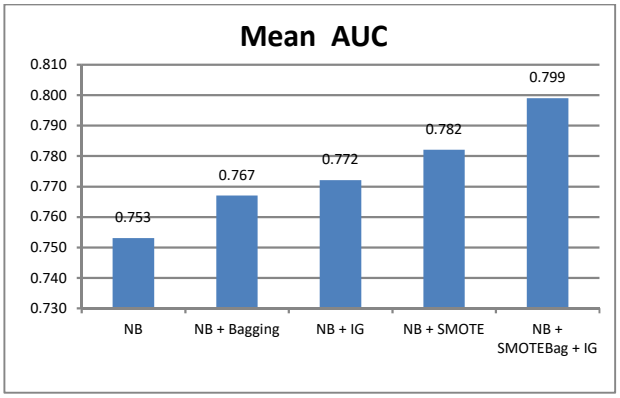
Ringkasan kinerja AUC dari masing-masing model dapat dilihat pada Tabel 7.

Table 7. AUC Perbandingan antara Model NB dan Model NB SMOTEBagging + 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 Bagging	0,72	0,79	0,63	0,72	0,71	0,78	0,86	0,75	0,82
NB IG	0,74	0,78	0,69	0,70	0,73	0,79	0,92	0,75	0,85
NB SMOTE	0,74	0,79	0,7	0,73	0,75	0,79	0,9	0,75	0,86
NB SMOTEBagging + IG	**0,76**	0,78	0,76	0,72	0,77	0,82	**0,93**	**0,81**	0,85

Sumber: Hasil Analisa (2016)

Dapat dilihat pada Gambar 8, integrasi antara SMOTE dan Bagging dengan *Information Gain* pada klasifikasi NB menghasilkan nilai rata-rata AUC yang tertinggi. Sehingga NB SMOTE Bagging + IG mengungguli model lainnya, dimana model tersebut merupakan model yang diusulkan oleh para peneliti yang melakukan penelitian sebelumnya pada bidang prediksi cacat perangkat lunak.



Sumber: Hasil Analisa (2016)

Gambar 8 Grafik Nilai Rata-rata AUC Perbandingan Metode Usulan dengan Metode Lain

Selanjutnya memverifikasi apakah terdapat perbedaan yang signifikan antara model yang telah diusulkan NB SMOTE Bagging + IG dengan model yang diusulkan oleh para peneliti sebelumnya. Kami melakukan statistik *Wilcoxon-Test* (2-*tailed*) (Wilcoxon, 1945)(Demsar, 2006) untuk uji berpasangan diantara model pada setiap dataset yang dijelaskan pada Tabel 8. Pada signifikasi statistik pengujian nilai *p* merupakan peluang mendapatkan uji statistik dengan asumsi bahwa hipotesis nol benar. Jika menolak hipotesis nol, ketika nilai *p* kurang dari tingkat signifikan yang telah ditentukan (α). 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.

Tabel 8. Nilai P dari AUC Perbandingan Metode Usulan dengan Metode Lain Menggunakan Uji Peringkat Bertanda *Wilcoxon*

MODEL	NB	NB + IG	NB + SMOTE	NB + Bagging	NB + SMOTE + IG + B
NB	1	0,075 (Not Sig)	0,011 (Sig)	0,722 (Not Sig)	0,015 (Sig)
NB + IG	0,075 (Not Sig)	1	0,066 (Not Sig)	0.044 (Sig)	0,008 (Sig)
NB + SMOTE	0,011 (Sig)	0,066 (Not Sig)	1	0.012 (Sig)	0,066 (Not Sig)
NB + Bagging	0,722 (Not Sig)	0,044 (Sig)	0,012 (Sig)	1	0.021 (Sig)
NB + SMOTE + IG + B	0,015 (Sig)	0,008 (Sig)	0,066 (Not Sig)	0.021 (Sig)	1

Seperti yang ditampilkan pada Tabel 8, memperlihatkan bahwa NB SMOTE, NB SMOTE + IG, NB SMOTE Bagging + IG memiliki perbedaan yang signifikan dengan NB. Nilai *p* dari NB SMOTE dengan NB uji peringkat bertanda *Wilcoxon* nya adalah 0,015 kurang dari *alpha* (0,05). Sedangkan nilai *p* dari NB SMOTE + IG dengan NB uji peringkat bertanda *Wilcoxon* nya adalah 0,008 kurang dari *alpha* (0,05). Dan nilai p dari NB SMOTE Bagging + IG dengan NB uji peringkat bertanda Wilcoxon bernilai 0.015. Sehingga dapat disimpulkan bahwa NB SMOTE, NB SMOTE + IG, NB SMOTE Bagging + IG dapat meningkatkan kinerja NB pada prediksi cacat perangkat lunak.

Sedangkan untuk melakukan uji lebih dari satu model kami menggunakan uji friedman diantara model pada setiap dataset yang dijelaskan pada Tabel 9.

Tabel 9. Nilai P dari AUC Perbandingan Metode Usulan dengan Metode Lain Menggunakan Uji *Friedman*

MODEL	NB	NB + IG	NB + SMOTE	NB + Bagging	NB + SMOTE + IG + B
NB	1	0,317 (Not Sig)	0,020 (Sig)	0,739 (Not Sig)	0,020 (Sig)
NB + IG	0,317 (Not Sig)	1	0,096 (Not Sig)	0,096 (Not Sig)	0,003 (Sig)
NB + SMOTE	0,020 (Sig)	0,096 (Not Sig)	1	0,005 (Sig)	0,317 (Not Sig)
NB + Bagging	0,739 (Not Sig)	0,096 (Not Sig)	0,005 (Sig)	1	0,096 (Not Sig)
NB + SMOTE + IG + B	0,020 (Sig)	0,003 (Sig)	0,317 (Not Sig)	0,096 (Not Sig)	1

Sumber: Hasil Analisa (2016)

Seperti yang ditampilkan pada Tabel 9 memperlihatkan bahwa NB SMOTE, NB SMOTE + IG, NB PSO Bagging dan NB SMOTE Bagging + IG memiliki perbedaan yang signifikan dengan NB. Nilai *p* dari NB SMOTE dengan NB uji *friedman* adalah 0,020 kurang

dari *alpha* (0,05). Nilai *p* dari NB SMOTE + IG dengan NB uji *friedman* nya adalah 0,020 kurang dari alpha (0,05). Sedangkan nilai *p* dari NB PSO Bagging dengan NB uji *friedman* nya adalah 0,020 kurang dari alpha (0,05). Dan Nilai *p* dari NB SMOTE Bagging + IG dengan NB uji *friedman* nya adalah 0,020 kurang dari alpha (0,05). Sehingga dapat disimpulkan bahwa NB SMOTE, NB SMOTE + IG, NB PSO Bagging dan NB pada prediksi cacat perangkat lunak.

KESIMPULAN

Integrasi antara teknik SMOTE dengan Bagging dan *algoritma Information Gain* diusulkan untuk meningkatkan kinerja *classifier* Naive Bayes pada prediksi cacat *software*. SMOTE dengan Bagging 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 Bagging + 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 Bagging + 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 Bagging + IG meningkatkan kinerja dari *Naive Bayes* pada prediksi cacat *software*.

REFERENSI

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

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).

Gao, K., Khoshgoftaar, T. M., Wang, H., & Seliya, N. (2011). Choosing software metrics for defect prediction : an investigation on feature selection techniques. *Software: Practice and Experience*, *41*(5), 579–606. doi:10.1002/spe

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. doi:10.1016/j.eswa.2011.09.073

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. doi:10.1109/ICMLA.2009.18

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. doi:10.1109/TSE.2007.256941

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

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. doi:10.1109/TSE.2010.90

Turhan, B., & Bener, A. (2009). Analysis of Naive Bayes' assumptions on software fault data: An empirical study. *Data & Knowledge Engineering*, *68*(2), 278–290. doi:10.1016/j.datak.2008.10.005

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. doi:10.1109/ICTAI.2009.20

Wang, T., Li, W., Shi, H., & Liu, Z. (2011). Software Defect Prediction Based on Classifiers Ensemble. *Journal of Information & Computational Science 8*, *16*(December), 4241–4254.

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. doi:10.1007/978-981-4585-18-7

BIODATA PENULIS

Sukmawati Anggraeni Putri. Memperoleh gelar S. Kom pada bidang Sistem Informasi dari STMIK Nusa Mandiri, Jakarta dan gelar M. Kom dari Pascasarjana STMIK Nusa Mandiri Jakarta, pada bidang Ilmu Komputer. Sat ini ia sebagai dosen tetap di STMIK Nusa Mandiri. Minat penelitiannya saat ini meliputi *software engineering*, *information system* dan *machine learning*