Penerapan Teknik Ensemble untuk Menangani Ketidakseimbangan Kelas pada Prediksi Cacat Software

Aries Saifudin
Fakultas Teknik, Universitas Pamulang
aries.saifudin@yahoo.co.id

Romi Satria Wahono
Fakultas Ilmu Komputer, Universitas Dian Nuswantoro
romi@romisatriawahono.net

Abstract: Software berkualitas tinggi adalah software yang tidak ditemukan cacat selama pemeriksaan dan pengujian. Memperbaiki software yang cacat setelah pengiriman membutuhkan biaya jauh lebih mahal dari pada selama pengembangan. Pengujian merupakan proses paling mahal dan menghabiskan waktu sampai 50% dari jadwal pengembangan software. Tetapi belum ada model prediksi cacat software yang berlaku umum. Naïve Bayes merupakan model paling efektif dan efisien, tetapi belum dapat mengklasifikasikan dataset berbasis metrik dengan kinerja terbaik secara umum dan selalu konsisten dalam semua penelitian. Dataset software metrics secara umum bersifat tidak seimbang, hal ini dapat menurunkan kinerja model prediksi cacat software karena cenderung menghasilkan prediksi kelas mayoritas. Secara umum ketidakseimbangan kelas dapat ditangani dengan dua pendekatan, yaitu level data dan level algoritma. Pendekatan level data ditujukan untuk memperbaiki keseimbangan kelas. Sedangkan pendekatan level algoritma dilakukan dengan memperbaiki algoritma atau menggabungkan (*ensemble*) pengklasifikasi tunggal agar menjadi lebih baik. Algoritma *ensemble* yang populer adalah boosting dan bagging. AdaBoost merupakan salah satu algoritma boosting yang telah menunjukkan dapat memperbaiki kinerja pengklasifikasi. Maka pada penelitian ini diusulkan penerapan teknik *ensemble* menggunakan algoritma AdaBoost dan Bagging. Pengklasifikasi yang digunakan adalah Naïve Bayes. Hasil penelitian menunjukkan bahwa teknik *ensemble* dengan algoritma Bagging dapat meningkatkan sensitivitas dan G-Mean secara signifikan, sedangkan AdaBoost tidak dapat meningkatkan secara signifikan. Sehingga disimpulkan bahwa Bagging lebih baik daripada AdaBoost ketika digunakan untuk meningkatkan kinerja Naïve Bayes pada prediksi cacat software.

Keywords: teknik ensemble, ketidakseimbangan kelas, prediksi cacat software

1 PENDAHULUAN

Kualitas software biasanya diukur dari jumlah cacat yang ada pada produk yang dihasilkan (Turhan & Bener, 2007, p. 244). Dengan mengurangi jumlah cacat pada software yang dihasilkan dapat meningkatkan kualitas software. Software berkualitas tinggi adalah software yang tidak ditemukan cacat selama pemeriksaan dan pengujian (McDonald, Musson, & Smith, 2008, p. 6). Pemeriksaan dan pengujian dilakukan terhadap alur dan keluaran dari software. Jumlah cacat yang ditemukan dalam pemeriksaan dan pengujian tidak dapat menjadi satu-satunya ukuran dari kualitas software. Pada banyak kasus, kualitas software lebih banyak dipengaruhi penggunanya, sehingga perlu diukur secara subyektif berdasarkan pada persepsi dan harapan *customer*.

Pengujian merupakan proses pengembangan perangkat lunak yang paling mahal dan banyak memakan waktu, karena sekitar 50% dari jadwal proyek digunakan untuk pengujian (Fakhrahmad & Sami, 2009, p. 206). Software yang cacat menyebabkan biaya pengembangan, perawatan dan estimasi menjadi tinggi, serta menurunkan kualitas software (Gayatri, Nickolas, Reddy, & Chitra, 2009, p. 393). Biaya untuk memperbaiki cacat akibat salah persyaratan (*requirement*) setelah fase penyebaran (*deployment*) dapat mencapai 100 kali, biaya untuk memperbaiki cacat pada tahap desain setelah pengiriman produk mencapai 60 kali, sedangkan biaya untuk memperbaiki cacat pada tahap desain yang ditemukan oleh pelanggan adalah 20 kali (Strangio, 2009, p. 389). Hal ini karena cacat software dapat mengakibatkan *business process* tidak didukung oleh software yang dikembangkan, atau software yang telah selesai dikembangkan harus dilakukan perbaikan atau dikembangkan ulang jika terlalu banyak cacat.

Aktifitas untuk mendukung pengembangan software dan proses menajemen *project* adalah wilayah penelitian yang penting (Lessmann, Baesens, Mues, & Pietsch, 2008, p. 485). Karena pentingnya kesempurnaan software yang dikembangkan, maka diperlukan prosedur pengembangan yang sempurna juga untuk menghindari cacat software. Prosedur yang diperlukan adalah strategi pengujian yang efektif dengan menggunakan sumber daya yang efisien untuk mengurangi biaya pengembangan software.

Prosedur untuk meningkatan kualitas software dapat dilakukan dengan berbagai cara, tetapi pendekatan terbaik adalah pencegahan cacat, karena manfaat dari upaya pencegahannya dapat diterapkan kembali pada masa depan (McDonald, Musson, & Smith, 2008, p. 4). Untuk dapat melakukan pencegahan cacat, maka harus dapat memprediksi kemungkinan terjadinya cacat.

Saat ini prediksi cacat software berfokus pada: 1) memperkirakan jumlah cacat dalam software, 2) menemukan hubungan cacat, dan 3) mengklasifikasikan kerawanan cacat dari komponen software, biasanya ke dalam kelompok rawan dan tidak rawan (Song, Jia, Shepperd, Ying, & Liu, 2011, p. 356). Klasifikasi adalah pendekatan yang populer untuk memprediksi cacat software (Lessmann, Baesens, Mues, & Pietsch, 2008, p. 485) atau untuk mengidentifikasi kegagalan software (Gayatri, Nickolas, Reddy, & Chitra, 2009, p. 393). Untuk melakukan klasifikasi diperlukan data.

Software metrics merupakan data yang dapat digunakan untuk mendeteksi modul software apakah memiliki cacat atau tidak (Chiş, 2008, p. 273). Salah satu metode yang efektif untuk mengidentifikasi modul software dari potensi rawan kegagalan adalah dengan menggunakan teknik *data mining*

Copyright © 2015 IlmuKomputer.Com

yang diterapkan pada *software metrics* yang dikumpulkan selama proses pengembangan software (Khoshgoftaar, Gao, & Seliya, 2010, p. 137). *Software metrics* yang dikumpulkan selama pengembangan disimpan dalam bentuk dataset.

Dataset NASA yang telah tersedia untuk umum merupakan data metrik perangkat lunak yang sangat populer dalam pengembangan model prediksi cacat software, karena 62 penelitian dari 208 penelitian telah menggunakan dataset NASA (Hall, Beecham, Bowes, Gray, & Counsell, 2011, p. 18). Dataset NASA yang tersedia untuk umum telah banyak digunakan sebagai bagian dari penelitian cacat software (Shepperd, Song, Sun, & Mair, 2013, p. 1208). Dataset NASA tersedia dari dua sumber, yaitu NASA MDP (*Metrics Data Program*) *repository* dan PROMISE (*Predictor Models in Software Engineering*) *Repository* (Gray, Bowes, Davey, Sun, & Christianson, 2011, p. 98). Semua dataset dari NASA tersedia secara online, sehingga analisa empiris dapat dengan mudah diulang, diperbaiki, disangkal, dan dievaluasi oleh peneliti lain (Singh & Verma, 2014, pp. 35-36). Menggunakan dataset NASA merupakan pilihan yang terbaik, karena mudah diperoleh dan kinerja dari model yang digunakan menjadi mudah untuk dibandingkan dengan penelitian sebelumnya.

Metode prediksi cacat menggunakan probabilitas dapat menemukan sampai 71% (Menzies, Greenwald, & Frank, 2007, p. 2), lebih baik dari metode yang digunakan oleh industri. Jika dilakukan dengan menganalisa secara manual dapat menemukan sampai 60% (Shull, et al., 2002, p. 254). Hasil tersebut menunjukkan bahwa menggunakan probalitas merupakan metode terbaik untuk menemukan cacat software.

Berdasarkan hasil penelitian yang ada, tidak ditemukan satu metode terbaik yang berguna untuk mengklasifikasikan berbasis metrik secara umum dan selalu konsisten dalam semua penelitian yang berbeda (Lessmann, Baesens, Mues, & Pietsch, 2008, p. 485). Tetapi model Naïve Bayes merupakan salah satu algoritma klasifikasi paling efektif (Tao & Wei-hua, 2010, p. 1) dan efisien (Zhang, Jiang, & Su, 2005, p. 1020), secara umum memiliki kinerja yang baik (Hall, Beecham, Bowes, Gray, & Counsell, 2011, p. 13), serta cukup menarik karena kesederhanaan, keluwesan, ketangguhan dan efektifitasnya (Gayatri, Nickolas, Reddy, & Chitra, 2009, p. 395). Maka dibutuhkan pengembangan prosedur penelitian yang lebih dapat diandalkan sebelum memiliki keyakinan dalam menyimpulkan perbandingan penelitian dari model prediksi cacat software (Myrtveit, Stensrud, & Shepperd, 2005, p. 380). Pengembangan prosedur penelitian dapat dilakukan dengan memperbaiki kualitas data yang digunakan atau dengan memperbaiki model yang digunakan.

Jika dilihat dari *software metrics* yang digunakan, secara umum dataset kualitas software bersifat tidak seimbang (*imbalanced*), karena umumnya cacat dari software ditemukan dalam persentase yang kecil dari modul software (Seiffert, Khoshgoftaar, Hulse, & Folleco, 2011, p. 1). Jumlah dari dataset yang rawan cacat (*fault-prone*) jauh lebih kecil dari pada dataset yang tidak rawan cacat (*nonfault-prone*). Klasifikasi data dengan pembagian kelas yang tidak seimbang dapat menimbulkan penurunan kinerja yang signifikan yang dicapai oleh algoritma belajar (*learning algorithm*) pengklasifikasi standar, yang mengasumsikan distribusi kelas yang relatif seimbang dan biaya kesalahan klasifikasi yang sama (Sun, Mohamed, Wong, & Wang, 2007, p. 3358). Ketepatan parameter tidak dapat digunakan untuk mengevaluasi kinerja dataset yang tidak seimbang (Catal, 2012, p. 195). Membangun model kualitas perangkat lunak tanpa melakukan pengolahan awal terhadap data tidak akan menghasilkan model prediksi cacat software yang efektif, karena jika data yang digunakan tidak seimbang maka hasil prediksi cenderung menghasilkan kelas mayoritas (Khoshgoftaar, Gao, & Seliya, 2010, p. 138). Karena cacat software merupakan kelas minoritas, maka banyak cacat yang tidak dapat ditemukan.

Ada tiga pendekatan untuk menangani dataset tidak seimbang (*unbalanced*), yaitu pendekatan pada level data, level algoritmik, dan menggabungkan atau memasangkan (*ensemble*) metode (Yap, et al., 2014, p. 14). Pendekatan pada level data mencakup berbagai teknik *resampling* dan sintesis data untuk memperbaiki kecondongan distribusi kelas data latih. Pada tingkat algoritmik, metode utamanya adalah menyesuaikan operasi algoritma yang ada untuk membuat pengklasifikasi (*classifier*) agar lebih konduktif terhadap klasifikasi kelas minoritas (Zhang, Liu, Gong, & Jin, 2011, p. 2205). Sedangkan pada pendekatan menggabungkan atau memasangkan (*ensemble*) metode, ada dua algoritma *ensemble-learning* paling populer, yaitu *boosting* dan *bagging* (Yap, et al., 2014, p. 14). Algoritma *boosting* telah dilaporkan sebagai meta-teknik untuk mengatasi masalah ketidakseimbangan kelas (*class imbalance*) (Sun, Mohamed, Wong, & Wang, 2007, p. 3360). Pada pendekatan algoritma dan *ensemble* memiliki tujuan yang sama, yaitu memperbaiki algoritma pengklasifikasi tanpa mengubah data, sehingga dapat dianggap ada 2 pendekatan saja, yaitu pendekatan level data dan pendekatan level algoritma (Peng & Yao, 2010, p. 111). Dengan membagi menjadi 2 pendekatan dapat mempermudah fokus objek perbaikan, pendekatan level data difokuskan pada pengolahan awal data, sedangkan pendekatan level algoritma difokuskan pada perbaikan algoritma atau menggabungkan (*ensemble*).

Bagging dan Boosting telah berhasil meningkatkan akurasi pengklasifikasi tertentu untuk dataset buatan dan yang sebenarnya. Bagging adalah metode *ensemble* yang sederhana namun efektif dan telah diterapkan untuk banyak aplikasi di dunia nyata (Liang & Zhang, 2011, p. 31). Bagging merupakan metode *ensemble* yang banyak diterapkan pada algoritma klasifikasi, dengan tujuan untuk meningkatkan akurasi pengklasifikasi dengan menggabungkan pengklasifikasi tunggal, dan hasilnya lebih baik daripada *random sampling* (Alfaro, Gamez, & Garcia, 2013, p. 1). Secara umum algoritma Boosting lebih baik dari pada Bagging, tetapi tidak merata baik. *Boosting* telah menunjukkan dapat meningkatkan kinerja pengklasifikasi dalam banyak situasi, termasuk ketika data tidak seimbang (Seiffert, Khoshgoftaar, Hulse, & Napolitano, 2008, p. 445). AdaBoost adalah kependekan dari *Adaptive Boosting* (Afza, Farid, & Rahman, 2011, p. 105) (Harrington, 2012, p. 132), merupakan algoritma *machine learning* yang dirumuskan oleh Yoav Freund and Robert Schapire. AdaBoost secara teoritis dapat secara signifikan digunakan untuk mengurangi kesalahan dari beberapa algoritma pembelajaran yang secara konsisten menghasilkan kinerja pengklasifikasi yang lebih baik. AdaBoost diterapkan pada Naïve Bayes dapat meningkatkan kinerja sebesar 33,33% dan memberikan hasil yang akurat dengan mengurangi nilai kesalahan klasifikasi dengan meningkatkan iterasi (Korada, Kumar, & Deekshitulu, 2012, p. 73). Beberapa penelitian tersebut telah menunjukkan bahwa metode *ensemble* (AdaBoost dan Bagging) dapat memperbaiki kinerja pengklasifikasi.

Untuk mencari solusi terbaik terhadap masalah ketidakseimbangan kelas (*class imbalance*) pada prediksi cacat software, maka pada penelitian ini akan diterapkan pendekatan level algoritma, yaitu teknik *ensemble* dengan algoritma AdaBoost dan Bagging. Diharapkan dapat diperoleh model terbaik untuk menyelesaikan masalah ketidakseimbangan

Copyright © 2015 IlmuKomputer.Com

kelas pada prediksi cacat software. Sedangkan algoritma pengklasifikasi yang digunakan adalah Naïve Bayes.

2 PENELITIAN TERKAIT

Penelitian tentang prediksi cacat software telah lama dilakukan, dan sudah banyak hasil penelitian yang dipublikasikan. Sebelum memulai penelitian, perlu dilakukan kajian terhadap penelitian sebelumnya, agar dapat mengetahui metode, data, maupun model yang sudah pernah digunakan. Kajian penelitian sebelumnya ditujukan untuk mengetahui *state of the art* tentang penelitian prediksi cacat software yang membahas tentang ketidakseimbangan (*imbalanced*) kelas.

Penelitian yang dilakukan oleh Riquelme, Ruiz, Rodriguez, dan Moreno (Riquelme, Ruiz, Rodriguez, & Moreno, 2008, pp. 67-74) menyatakan bahwa kebanyakan dataset untuk rekayasa perangkat lunak sangat tidak seimbang (*unbalanced*), sehingga algoritma *data mining* tidak dapat menghasilkan model pengklasifikasi yang optimal untuk memprediksi modul yang cacat. Pada penelitian ini dilakukan analisa terhadap dua teknik penyeimbangan (*balancing*) (yaitu: WEKA *randomly resampling* dan SMOTE) dengan dua algoritma pengklasifikasi umum (J48 dan Naïve Bayes), dan menggunakan lima dataset dari PROMISE *repository*. Hasil penelitian menunjukkan bahwa teknik penyeimbangan (*balancing*) mungkin tidak meningkatkan persentase kasus yang diklasifikasikan dengan benar, tetapi dapat meningkatkan nilai AUC, khususnya menggunakan SMOTE, AUC rata-rata meningkat 11,6%. Hal ini menunjukkan bahwa teknik penyeimbangan (*balancing*) dapat mengklasifikasikan kelas minoritas dengan lebih baik.

Penelitian yang dilakukan oleh Khoshgoftaar, Gao dan Seliya (Khoshgoftaar, Gao, & Seliya, 2010, pp. 137-144) menyatakan bahwa komunitas *data mining* dan *machine learning* sering dihadapkan pada dua masalah utama, yaitu bekerja dengan data tidak seimbang dan memilih fitur terbaik. Penelitian ini menerapkan teknik seleksi fitur untuk memilih atribut penting dan teknik pengambilan data untuk mengatasi ketidakseimbangan kelas. Beberapa sekenario diusulkan menggunakan fitur seleksi dan *resampling* data bersama-sama, yaitu: (1) seleksi fitur berdasarkan data asli, dan pemodelan (prediksi cacat) berdasarkan data asli, (2) seleksi fitur berdasarkan data asli, dan pemodelan berdasarkan data sampel, (3) seleksi fitur berbasis pada data sampel, dan pemodelan berdasarkan data asli, dan (4) seleksi fitur berdasarkan data sampel, dan pemodelan berdasarkan data sampel. Tujuan penelitian ini adalah untuk membandingkan kinerja model prediksi cacat software berdasarkan empat skenario. Pada penelitian ini menggunakan sembilan dataset pengukuran perangkat lunak yang diperoleh dari repositori proyek software PROMISE (*Java-based Eclipse project*). Seleksi fitur menggunakan teknik peringkat fitur (*feature ranking techniques*), *Chi-Square* (CS), *Information Gain* (IG), *Gain Ratio* (GR), dua tipe ReliefF (RF and RFW), dan *Symmetrical Uncertainty* (SU). Metode *resampling* yang digunakan adalah *Random Under-Sampling* (RUS). Sedangkan metode pengklasifikasi yang digunakan adalah *k-Nearest Neighbors* (kNN) dan *Support Vector Machine* (SVM). Hasil empiris menunjukkan bahwa seleksi fitur berdasarkan data sampel menghasilkan lebih baik secara signifikan daripada seleksi fitur berdasarkan data asli, dan bahwa model prediksi cacat melakukan hal yang sama terlepas dari apakah data pelatihan dibentuk menggunakan data sampel atau asli. AUC rata-rata meningkat 1,44% untuk KNN dan 1,04% untuk SVM.

Penelitian yang dilakukan oleh Wahono, Suryana, dan Ahmad (Wahono, Suryana, & Ahmad, 2014, pp. 1324-1333) menyatakan bahwa kinerja model prediksi cacat software berkurang secara signifikan karena dataset yang digunakan mengandung *noise* (kegaduhan) dan ketidakseimbangan kelas. Pada penelitian ini, diusulkan kombinasi metode optimasi metaheuristik dan teknik Bagging untuk meningkatkan kinerja prediksi cacat software. Metode optimasi metaheuristik (algoritma genetik dan *particle swarm optimization*) diterapkan untuk menangani pemilihan fitur, dan teknik Bagging digunakan untuk menangani masalah ketidakseimbangan kelas. Penelitian ini menggunakan 9 NASA MDP dataset dan 10 algoritma pengklasifikasi yang dikelompokkan dalam 5 tipe, yaitu pengklasifikasi statistik tradisional (*Logistic Regression* (LR), *Linear Discriminant Analysis* (LDA), dan Naïve Bayes (NB)), *Nearest Neighbors* (*k-Nearest Neighbor* (k-NN) dan K*), *Neural Network* (*Back Propagation* (BP)), *Support Vector Machine* (SVM), dan *Decision Tree* (C4.5, *Classification and Regression Tree* (CART), dan *Random Forest* (RF)). Hasilnya menunjukkan bahwa metode yang diusulkan dapat memberikan peningkatan yang mengesankan pada kinerja model prediksi untuk sebagian besar pengklasifikasi. Hasil penelitian menunjukkan bahwa tidak ada perbedaan yang signifikan antara optimasi PSO dan algoritma genetik ketika digunakan sebagai seleksi fitur untuk sebagian besar pengklasifikasi pada model prediksi cacat software. AUC rata-rata meningkat 25,99% untuk GA dan 20,41% untuk PSO.

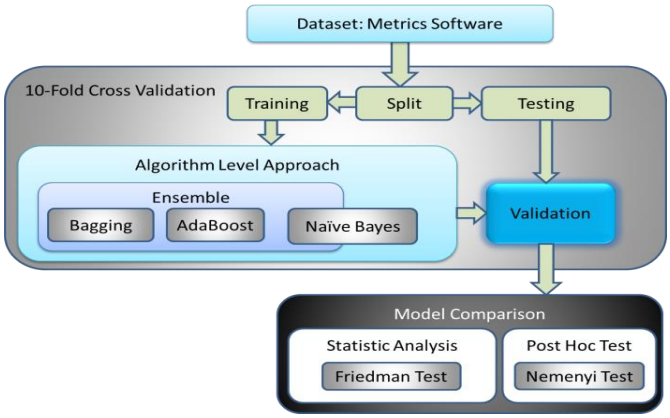
Pada penelitian ini akan menerapkan teknik *ensemble* untuk mengurangi pengaruh ketidakseimbangan kelas dan meningkatkan kemampuan memprediksi kelas minoritas. Teknik *ensemble* yang akan digunakan adalah AdaBoost dan Bagging.

3 METODE YANG DIUSULKAN

Penelitian ini dilakukan dengan mengusulkan model, mengembangkan aplikasi untuk mengimplementasikan model yang diusulkan, menerapkan pada NASA dataset yang diperoleh dari NASA MDP (*Metrics Data Program*) *repository* dan PROMISE (*Predictor Models in Software Engineering*) *Repository*, dan mengukur kinerjanya. Perangkat lunak yang digunakan untuk mengembangkan aplikasi adalah IDE (*Integrated Development Environment*) NetBeans dengan bahasa Java.

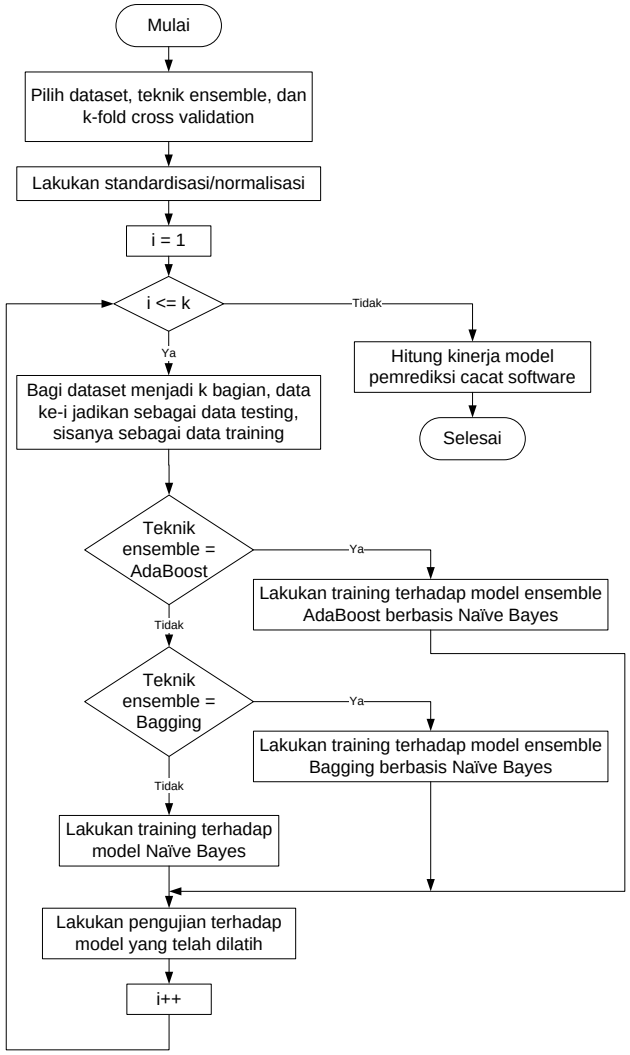
Untuk menangani masalah ketidakseimbangan kelas pada dataset software metrics, diusulkan model menggunakan teknik *ensemble* (AdaBoost, dan Bagging), dan algoritma pengklasifikasi Naïve Bayes. Validasi menggunakan *10-fold cross validation*. Hasil pengukuran dianalisa menggunakan uji Friedman, Nemenyi *post hoc*, dan dibuatkan diagram Demsar. Kerangka kerja model yang diusulkan ditunjukkan pada Gambar 1.

Copyright © 2015 IlmuKomputer.Com



Gambar 1 Kerangka Kerja Model yang Diusulkan

Pada model yang diusulkan, dataset software metrics akan dibagi menjadi 10 bagian. Secara bergantian, setiap bagian secara berurutan dijadikan sebagai data uji, sedangkan bagian lain sebagai data latih. Jika teknik ensemble yang dipilih adalah AdaBoost, maka data latih digunakan untuk melatih model AdaBoost berbasis Naïve Bayes. Jika teknik ensemble yang dipilih adalah Bagging, maka data latih digunakan untuk melatih model Bagging berbasis Naïve Bayes. Model yang telah dilatih diuji dengan data uji, kemudian diukur kinerjanya. Proses ini ditunjukkan dengan flowchart pada Gambar 2.



Gambar 2 Flowchart Model yang Diusulkan

Teknik *ensemble* AdaBoost menggunakan persamaan berikut ini:

$$F(x) = \sum_{t=1}^{T} \propto_t h_t(x)$$

Di mana:
$h_t(x)$: Pengklasifikasi dasar atau lemah
$\propto_t$: Tingkat pembelajaran (*learning rate*)
F(x) : Hasil, berupa pengklasifikasi kuat atau akhir

Algoritma AdaBoost (Zhou & Yu, AdaBoost, 2009, p. 130):

Masukan:
Dataset D = {(x_1,y_1), (x_2,y_2), ..., (x_m,y_m)};
Algoritma pembelajaran lemah (*Weak Learner*) L;
Sebuah *integer* T menyatakan banyaknya iterasi.

Proses:
Inisialisasi berat distribusi:
$D_1(i) = \frac{1}{m}$ untuk semua $i = 1, \dots, m$
for t=1, ... , T:
Melatih pembelajar dasar/lemah h_t dari D menggunakan distribusi D_t
$h_t = L(D,D_t)$
Mengkalkulasi kesalahan dari h_t: $\varepsilon_t = Pr_{x\sim D_{t,y}} I[h_t(\underline{x_i}) \neq y_i]$
if $\varepsilon_t > 0.5$ then break
Menetapkan berat dari h_t: $\propto_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \text{x} \begin{cases} exp(-\propto_t) \; if \; h_t(x_i) = y_i \\ exp(\propto_t) \; if \; h_t(i) \neq y_i \end{cases}$$

Meng-*update* distribusi, di mana Zt adalah faktor normalisasi yang mengaktifkan D_{t+1} menjadi distribusi:

$$\frac{D_t(i) exp\left(-\propto_t y_i h_t(x_i)\right)}{Z_t}$$

end

Keluaran:
Pengklasifikasi akhir/kuat:

$$H(x) = sign(\sum_{t=1}^{T} \propto_t h_t(x))$$

Kesalahan diukur dengan memperhatikan distribusi D_t di mana algoritma pembelajar lemah dilatih. Dalam prakteknya, algoritma pembelajar lemah mungkin merupakan suatu algoritma yang dapat menggunakan bobot D_t pada sampel pelatihan. Atau, bila hal ini tidak memungkinan, bagian dari sampel pelatihan dapat di-*resampling* menurut D_t, dan hasil dari *resampling* yang tidak berbobot (*unweighted*) dapat digunakan untuk melatih algoritma pembelajar yang lemah.
Sedangkan algoritma Bagging adalah:

Masukan: B adalah jumlah *bag*, T adalah data *training* yang berukuran N, x adalah data yang diuji.
Keluaran: Hasil klasifikasi.

Copyright © 2015 IlmuKomputer.Com

Ulangi untuk b = 1, 2, ..., B
a. Buat bootstrap BS_b dengan mengambil sampel dari T sejumlah N dengan penggantian.
b. Latih pengklasifikasi tunggal C_b dengan *boostrap* BS_b

akhir perulangan

Gabungkan hasil pengklasifikasi tunggal C_b (x_i); b = 1, 2, ..., B , hasil klasifikasi yang paling banyak dijadikan keputusan final klasifikasi, mengikuti rumus:

$$C_f(x_i) = \arg max_{j \in Y} \sum_{b=1}^{B} I(C_b(x_i) = j)$$

Bagging adalah metode yang mengkombinasikan *bootstrapping* dan *aggregating* (Alfaro, Gamez, & Garcia, 2013, p. 4). Sampel *bootstrap* ini diperoleh dengan melakukan *resampling* dengan penggantian (*with replacemants*) dari dataset asli sehingga menghasilkan jumlah elemen yang sama dari dataset asli.

Bagging dapat benar-benar berguna untuk membangun pengklasifikasi menjadi lebih baik bila pada pengamatan kumpulan data latih yang terdapat *noise* (kegaduhan) (Alfaro, Gamez, & Garcia, 2013, p. 5). Berdasarkan penelitian, perbaikan yang besar didapat ketika menggunakan 10 *bootstrap*, jika menggunakan lebih dari 25 *bootstrap*, banyak tenaga yang dihabiskan.

Untuk mengukur kinerja digunakan *confusion matrix*. *Confusion matrix* memberikan keputusan yang diperoleh dalam pelatihan dan pengujian (Bramer, 2007, p. 89). *Confusion matrix* memberikan penilaian kinerja klasifikasi berdasarkan objek dengan benar atau salah (Gorunescu, 2011, p. 319). *Confusion matrix* merupakan matrik 2 dimensi yang menggambarkan perbandingan antara hasil prediksi dengan kenyataan.

Untuk data tidak seimbang, akurasi lebih didominasi oleh ketepatan pada data kelas minoritas, maka metrik yang tepat adalah AUC (*Area Under the ROC Curve*), F-Measure, G-Mean, akurasi keseluruhan, dan akurasi untuk kelas minoritas (Zhang & Wang, 2011, p. 85). Akurasi kelas minoritas dapat menggunakan metrik TPrate/recall (sensitivitas). G-Mean dan AUC merupakan evaluasi prediktor yang lebih komprehensif dalam konteks ketidakseimbangan (Wang & Yao, 2013, p. 438).

Rumus-rumus yang digunakan untuk melakukan perhitungannya adalah (Gorunescu, 2011, pp. 320-322):

$$Akurasi = \frac{TP+TN}{TP + TN + FP + FN}$$

$$Sensitivitas = recall = TP_{rate} = \frac{TP}{TP + FN}$$

$$Specificity = TN_{rate} = \frac{TN}{TN + FP}$$

$$FP_{rate} = \frac{FP}{FP + TN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$F - Measure = \frac{(1 + \beta^2)\ x\ recall\ x\ precision}{(\beta\ x\ recall + precision)}$$

$$G - Mean = \sqrt{Sensitivitas\ x\ Specificity}$$

F-Measure adalah metrik evaluasi yang populer untuk masalah ketidakseimbangan. *F-Measure* mengkombinasikan *recall*/sensitivitas dan *precision* sehingga menghasilkan metrik yang efektif untuk pencarian kembali informasi dalam himpunan yang mengandung masalah ketidakseimbangan. *F-Measure* juga bergantung pada faktor β, yaitu parameter yang bernilai dari 0 sampai tak terhingga, dan digunakan untuk mengontrol pengaruh dari *recall* dan *precision* secara terpisah. Ini dapat menunjukkan bahwa ketika β bernilai 0, maka pengaruh *precision* terhadap *F-Measure* berkurang, dan sebaliknya, ketika β bernilai tak terhingga, maka pengaruh *recall* berkurang.

Ketika β bernilai 1, maka *F-Measure* terlihat sebagai integrasi kedua ukuran secara seimbang. Secara prinsip, *F-Measure* merepresentasikan rata-rata harmonis antara *recall* dan *precision*.

$$F - Measure = \frac{2\ x\ recall\ x\ precision}{(\ recall\ +\ precision)}$$

Rata-rata harmonis dari dua angka cenderung lebih dekat dengan yang lebih kecil dari keduanya. Oleh karena itu nilai *F-Measure* yang tinggi menjamin bahwa keduanya dari *recall* dan *precision* bernilai cukup tinggi.

Area Under the ROC (Receiver Operating Characteristic) Curve (AUROC atau AUC) adalah ukuran numerik untuk membedakan kinerja model, dan menunjukkan seberapa sukses dan benar peringkat model dengan memisahkan pengamatan positif dan negatif (Attenberg & Ertekin, 2013, p. 114). AUC menyediakan ukuran tunggal dari kinerja pengklasifikasi untuk menilai model mana yang lebih baik secara rata-rata (López, Fernández, & Herrera, 2014, p. 4). AUC merangkum informasi kinerja pengklasifikasi ke dalam satu angka yang mempermudah perbandingan model ketika tidak ada kurva ROC yang mendominasi (Weiss, 2013, p. 27). AUC adalah cara yang baik untuk mendapatkan nilai kinerja pengklasifikasi secara umum dan untuk membandingkannya dengan pengklasifikasi yang lain (Japkowicz, 2013, p. 202). AUC adalah ukuran kinerja yang populer dalam ketidakseimbangan kelas, nilai AUC yang tinggi menunjukkan kinerja yang lebih baik (Liu & Zhou, 2013, p. 75). Sehingga untuk memilih model mana yang terbaik, dapat dilakukan dengan menganalisa nilai AUC.

AUC dihitung berdasarkan rata-rata perkiraan bidang berbentuk trapesium untuk kurva yang dibuat oleh TP_{rate} dan FP_{rate} (Dubey, Zhou, Wang, Thompson, & Ye, 2014, p. 225). AUC memberikan ukuran kualitas antara nilai positif dan negatif dengan nilai tunggal (Rodriguez, Herraiz, Harrison, Dolado, & Riquelme, 2014, p. 375). Ukuran AUC dihitung sebagai daerah kurva ROC (López, Fernández, & Herrera, 2014, p. 4) (Galar, Fernández, Barrenechea, & Herrera, 2013, p. 3462) menggunakan persamaan:

$$AUC = \frac{1 + TP_{rate} - FP_{rate}}{2}$$

Berdasarkan kerangka kerja model yang diusulkan pada Gambar 1, hasil validasi dan pengukuran kinerja, selanjutnya dilakukan analisa statistik dan uji post hoc. Hanya sedikit penelitian prediksi cacat software yang dilaporkan menggunakan kesimpulan statistik dalam menentukan signifikansi hasil penelitian (Lessmann, Baesens, Mues, & Pietsch, 2008, p. 487). Biasanya hanya dilakukan berdasarkan

Copyright © 2015 IlmuKomputer.Com

pengalaman dan percobaan tanpa menerapkan pengujian statistik secara formal. Hal ini dapat menyesatkan dan bertentangan dengan penelitian lain.

Dalam literatur ada dua jenis uji statistik, yaitu uji parametrik dan uji nonparametrik (García, Fernández, Luengo, & Herrera, 2010, p. 2045). Uji parametrik dapat menggunakan uji T (*T-test*) dan ANOVA (*Analysis of Variance*). Uji nonparametrik dapat menggunakan uji tanda (*sign test*), uji peringkat bertanda Wilcoxon (*Wilcoxon signed-rank test*), uji Friedman (*Friedman test*), dan uji konkordansi (keselarasan) Kendall (*Kendall's coefficient of concordance* atau Kendall's W). Untuk melakukan uji statistik tergantung pada data yang digunakan. Untuk uji parametrik menggunakan data dengan distribusi normal, varians bersifat homogen, data bersifat ratio atau interval, nilai tengah berupa rata-rata. Jika tidak memenuhi syarat untuk uji parametrik, sebaiknya menggunakan uji nonparametrik. Aturan secara umum, lebih baik menggunakan uji parametrik daripada nonparametrik, karena uji parametrik lebih deskriminatif dan kuat (Carver & Nash, 2012, p. 249). Tetapi jika tidak tidak memenuhi syarat, maka disarankan menggunakan uji nonparametrik.

Uji t dan ANOVA tidak cocok digunakan pada penelitian *machine learning* (Demšar, 2006, pp. 6, 10), karena perbedaan kinerja harus terdistribusi normal, untuk memastikan bentuk distribusinya minimal 30 data, hasil pengukuran kinerja *machine learning* sering tidak mencukupi. Berdasarkan sifat data yang dihasilkan *machine learning*, uji nonparametrik sebaiknya lebih diutamakan daripada parametrik. Secara keseluruhan, uji nonparametrik yang diberi nama uji peringkat bertanda Wilcoxon dan uji Friedman cocok untuk menguji model *machine learning* (Demšar, 2006, p. 27). Tetapi beberapa uji perbandingan harus digunakan jika ingin membentuk perbandingan statistik dari hasil eksperimen di antara berbagai algoritma (García, Fernández, Luengo, & Herrera, 2010, p. 2060).

Pada uji statistik mungkin telah dinyatakan bahwa ada perbedaan signifikan pada model yang diuji, tetapi tidak menunjukkan model mana yang memiliki perbedaan signifikan. Untuk mengidentifikasi model mana yang berbeda secara signifikan, maka dapat digunakan uji *post hoc* (Corder & Foreman, 2009, p. 80). Uji *post hoc* dapat membantu peneliti dalam upaya memahami pola yang benar dari rata-rata populasi (Huck, 2012, p. 258). Uji post hoc dilakukan dengan membuat tabel perbandingan, atau visualisasi grafis.

Jika hipotesis nol (H_0) ditolak, maka dapat dilanjutkan dengan melakukan uji *post hoc* (Demšar, 2006, p. 11). Untuk menunjukkan perbedaan secara signifikan, maka digunakan Nemenyi *post hoc*. Nemenyi *post hoc* digunakan untuk menyatakan bahwa dua atau lebih pengklasifikasi memiliki kinerja yang berbeda secara signifikan jika rata-rata peringkatnya memiliki perbedaan setidaknya sebesar *Critical Different* (CD), sesuai persamaan:

$$CD = q_{\propto,\infty,K}\sqrt{\frac{K(K+1)}{12D}}$$

Pada di atas nilai $q_{\propto,\infty,K}$ didasarkan pada *studentized range statistic* untuk derajat kebebasan (*degree of freedom*) bernilai tak terhingga (*infinity*). Sedangkan K adalah jumlah pengklasifikasi, dan D adalah jumlah dataset.

Hasil dari uji Friedman dan Nemenyi *post hoc* selanjutnya disajikan dalam bentuk grafis menggunakan diagram Demsar (Demšar, 2006, p. 16) yang telah dimodifikasi (Lessmann, Baesens, Mues, & Pietsch, 2008, p. 491).

4 HASIL EKSPERIMEN

Eksperimen dilakukan menggunakan sebuah laptop DELL Inspiron 1440 dengan prosesor Pentium® Dual-Core CPU T4500 @ 2.30 GHz, memori (RAM) 4,00 GB, dan sistem operasi Windows 7 Ultimate Service Pack 1 32-bit. Sedangkan perangkat lunak untuk mengembangkan aplikasi adalah IDE (*Integrated Development Environment*) NetBeans menggunakan bahasa Java. Untuk menganalisis hasil pengukuran kinerja digunakan aplikasi IBM SPSS statistic 21 dan XLSTAT versi percobaan (*trial*).

Hasil pengukuran kinerja model ketika diterapkan 20 NASA dataset (10 dari NASA MDP *repository* dan 10 dari PROMISE *repository*) ditunjukkan pada Tabel 1 sampai Tabel 10.

Tabel 1 Akurasi NASA MDP repository

Model	NASA MDP Repository									
	CM1	JM1	KC1	KC3	MC2	PC1	PC2	PC3	PC4	PC5
NB	81,65%	78,87%	74,10%	78,87%	72,58%	88,81%	88,09%	36,75%	85,59%	75,03%
AB+NB	81,65%	78,87%	74,10%	78,87%	73,39%	90,87%	88,37%	84,81%	85,98%	75,03%
BG+NB	81,96%	78,76%	73,84%	78,35%	71,77%	88,22%	76,18%	39,51%	85,04%	74,85%

Tabel 2 Akurasi PROMISE repository

Model	PROMISE Repository									
	CM1	JM1	KC1	KC3	MC2	PC1	PC2	PC3	PC4	PC5
NB	82,61%	78,81%	74,01%	82,10%	72,90%	89,66%	92,51%	47,34%	85,43%	74,73%
AB+NB	83,30%	78,81%	74,01%	84,26%	70,97%	89,66%	92,51%	84,74%	85,04%	74,73%
BG+NB	81,92%	78,86%	73,67%	82,10%	72,26%	88,58%	90,38%	63,66%	85,12%	74,91%

Tabel 3 Sensitifitas NASA MDP repository

Model	NASA MDP Repository									
	CM1	JM1	KC1	KC3	MC2	PC1	PC2	PC3	PC4	PC5
NB	30,95%	19,85%	33,67%	36,11%	38,64%	40,00%	12,50%	90,77%	28,98%	22,27%
AB+NB	30,95%	19,85%	33,67%	36,11%	40,91%	34,55%	12,50%	9,23%	48,30%	22,27%
BG+NB	33,33%	20,60%	34,35%	36,11%	40,91%	41,82%	43,75%	91,54%	35,23%	23,58%

Tabel 4 Sensitifitas PROMISE repository

Model	PROMISE Repository									
	CM1	JM1	KC1	KC3	MC2	PC1	PC2	PC3	PC4	PC5
NB	32,61%	19,73%	33,67%	42,86%	35,29%	35,00%	23,81%	85,14%	28,98%	21,62%
AB+NB	32,61%	19,73%	33,67%	40,48%	35,29%	35,00%	23,81%	14,19%	50,00%	21,62%
BG+NB	30,44%	20,10%	33,67%	42,86%	35,29%	36,67%	33,33%	79,05%	31,82%	23,14%

Tabel 5 F-Measure NASA MDP repository

Model	NASA MDP Repository									
	CM1	JM1	KC1	KC3	MC2	PC1	PC2	PC3	PC4	PC5
NB	0,302	0,282	0,397	0,388	0,500	0,367	0,044	0,262	0,358	0,325
AB+NB	0,302	0,282	0,397	0,388	0,522	0,380	0,045	0,130	0,489	0,325
BG+NB	0,322	0,288	0,399	0,382	0,507	0,365	0,075	0,272	0,395	0,336

Tabel 6 F-Measure PROMISE repository

Model	PROMISE Repository									
	CM1	JM1	KC1	KC3	MC2	PC1	PC2	PC3	PC4	PC5
NB	0,283	0,280	0,396	0,383	0,462	0,307	0,089	0,254	0,355	0,316
AB+NB	0,291	0,280	0,396	0,400	0,444	0,307	0,089	0,163	0,481	0,316
BG+NB	0,262	0,284	0,393	0,383	0,456	0,295	0,097	0,314	0,372	0,333

Tabel 7 G-Mean NASA MDP repository

Model	NASA MDP Repository									
	CM1	JM1	KC1	KC3	MC2	PC1	PC2	PC3	PC4	PC5
NB	0,525	0,433	0,544	0,566	0,594	0,610	0,335	0,514	0,524	0,459
AB+NB	0,525	0,433	0,544	0,566	0,611	0,575	0,336	0,297	0,667	0,459
BG+NB	0,545	0,440	0,547	0,564	0,603	0,621	0,580	0,543	0,573	0,470

Tabel 8 G-Mean PROMISE repository

Model	PROMISE Repository									
	CM1	JM1	KC1	KC3	MC2	PC1	PC2	PC3	PC4	PC5
NB	0,537	0,432	0,543	0,614	0,568	0,572	0,472	0,604	0,523	0,452
AB+NB	0,539	0,432	0,543	0,606	0,559	0,572	0,472	0,363	0,673	0,452
BG+NB	0,517	0,436	0,542	0,614	0,565	0,581	0,552	0,699	0,546	0,467

Copyright © 2015 IlmuKomputer.Com

Tabel 9 AUC NASA MDP repository

Model	NASA MDP Repository									
	CM1	JM1	KC1	KC3	MC2	PC1	PC2	PC3	PC4	PC5
NB	0,600	0,572	0,607	0,624	0,649	0,666	0,512	0,600	0,618	0,584
AB+NB	0,600	0,572	0,607	0,624	0,661	0,652	0,513	0,523	0,702	0,584
BG+NB	0,612	0,574	0,608	0,620	0,648	0,671	0,603	0,619	0,641	0,587

Tabel 10 AUC PROMISE repository

Model	PROMISE Repository									
	CM1	JM1	KC1	KC3	MC2	PC1	PC2	PC3	PC4	PC5
NB	0,605	0,571	0,607	0,654	0,633	0,642	0,587	0,640	0,617	0,580
AB+NB	0,609	0,571	0,607	0,656	0,619	0,642	0,587	0,536	0,703	0,580
BG+NB	0,592	0,572	0,604	0,654	0,628	0,644	0,623	0,705	0,628	0,586

Untuk mengetahui algoritma *ensemble* manakah antara AdaBoost dan Bagging yang dapat meningkatkan kinerja Naïve Bayes menjadi lebih baik pada prediksi cacat software, maka dilakukan uji Friedman, Nemenyi *post hoc*, dan disajikan dalam diagram Demsar yang dimodifikasi. Karena nilai CD (*Critical Difference*) pada Nemenyi *post hoc* dipengaruhi oleh nilai *studentized range statistic* (3,314), jumlah model (K=3), dan jumlah dataset (D=20), maka nilainya dihitung sebagai berikut:

$$CD = q_{\propto,\infty,K}\sqrt{\frac{K(K+1)}{12D}} = 3{,}314\sqrt{\frac{3(3+1)}{12 \cdot 20}}$$

$$CD = 0{,}741032928$$

Untuk mengetahui signifikansi perbedaan di antara keempat model, dilakukan uji Friedman menggunakan aplikasi SPSS. Nilai p-value dari uji Friedman ditunjukkan pada Tabel 11.

Tabel 11 Rekap P-Value pada Uji Friedman

Pengukuran	P-Value	Hasil (α = 0,05)
Akurasi	0,011	Sig., Ada perbedaan nilai di antara model
Sensitivitas	0,002	Sig., Ada perbedaan nilai di antara model
F-Measure	0,225	Not Sig., Tidak ada perbedaan nilai di antara model
G-Mean	0,019	Sig., Ada perbedaan nilai di antara model
AUC	0,073	Not Sig., Tidak ada perbedaan nilai di antara model

Berdasarkan signifikansi dari uji Friedman tersebut, maka hanya pengukuran yang memiliki perbedaan secara signifikan saja yang dibuatkan diagram Demsar yang telah dimodifikasi.

Berikut ini adalah tahapan pembuatan diagram Demsar sesuai hasil pengukuran yang diuji:

A. Akurasi

Data yang diperoleh dari pengukuran akurasi model ditunjukkan pada Tabel 12.

Tabel 12 Hasil Pengukuran Nilai Akurasi

Data Set		Model		
		NB	AB+NB	BG+NB
MDP Repository	CM1	81,65%	81,65%	81,96%
	JM1	78,87%	78,87%	78,76%
	KC1	74,10%	74,10%	73,84%
	KC3	78,87%	78,87%	78,35%
	MC2	72,58%	73,39%	71,77%
	PC1	88,81%	90,87%	88,22%
	PC2	88,09%	88,37%	76,18%
	PC3	36,75%	84,81%	39,51%
	PC4	85,59%	85,98%	85,04%
	PC5	75,03%	75,03%	74,85%
PROMISE Repository	CM1	82,61%	83,30%	81,92%
	JM1	78,81%	78,81%	78,86%
	KC1	74,01%	74,01%	73,67%
	KC3	82,10%	84,26%	82,10%
	MC2	72,90%	70,97%	72,26%
	PC1	89,66%	89,66%	88,58%
	PC2	92,51%	92,51%	90,38%
	PC3	47,34%	84,74%	63,66%
	PC4	85,43%	85,04%	85,12%
	PC5	74,73%	74,73%	74,91%

Uji Friedman dilakukan dengan memberikan peringkat setiap nilai pengukuran. Peringkat diberikan pada model untuk setiap dataset, nilai terbesar diberi peringkat 1, terbesar kedua diberi peringkat 2, dan seterusnya. Hasilnya ditunjukkan pada Tabel 13.

Tabel 13 Peringkat Akurasi Model pada Uji Friedman

Data Set		Model		
		NB	AB+NB	BG+NB
MDP Repository	CM1	2,5	2,5	1
	JM1	1,5	1,5	3
	KC1	1,5	1,5	3
	KC3	1,5	1,5	3
	MC2	2	1	3
	PC1	2	1	3
	PC2	2	1	3
	PC3	3	1	2
	PC4	2	1	3
	PC5	1,5	1,5	3
PROMISE Repository	CM1	2	1	3
	JM1	2,5	2,5	1
	KC1	1,5	1,5	3
	KC3	2,5	1	2,5
	MC2	1	3	2
	PC1	1,5	1,5	3
	PC2	1,5	1,5	3
	PC3	3	1	2
	PC4	1	3	2
	PC5	2,5	2,5	1
Jumlah		38,5	32	49,5
Rata-Rata		1,925	1,6	2,475

Nemenyi *post hoc* dilakukan dengan membuat tabel perbandingan berpasangan, tabel p-value, dan tabel signifikansi perbedaan untuk menunjukkan pasangan model mana yang memiliki perbedaan secara signifikan.

Tabel 14 Perbandingan Berpasangan pada Nemenyi *Post Hoc*

	NB	AB+NB	BG+NB
NB	0	0,325	-0,550
AB+NB	-0,325	0	-0,875
BG+NB	0,550	0,875	0

Copyright © 2015 IlmuKomputer.Com

Untuk menghitung nilai P-value Nemenyi *post hoc* digunakan software XLSTAT. P-value yang bernilai lebih kecil dari 0,05 (nilai α) dicetak lebih tebal, ini menunjukkan bahwa ada perbedaan yang signifikan di antara model pada baris dan kolom yang sesuai.

Tabel 15 P-value pada Nemenyi *Post Hoc*

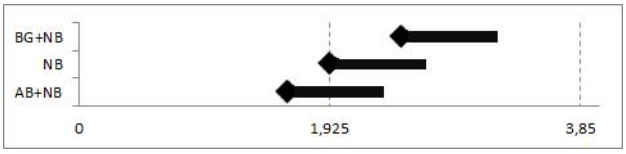
	NB	AB+NB	BG+NB
NB	1	0,559306	0,190612
AB+NB	0,559306	1	**0,015615**
BG+NB	0,190612	**0,015615**	1

Model pada kolom dan baris yang memiliki perbedaan signifikan, diberi nilai Sig. Sedangkan yang perbedaannya tidak signifikan diberi nilai Not.

Tabel 16 Perbedaan Signifikan pada Nemenyi *Post Hoc*

	NB	AB+NB	BG+NB
NB		Not	Not
AB+NB	Not		**Sig**
BG+NB	Not	**Sig**	

Setelah dilakukan uji Friedman dan Nemenyi *post hoc*, selanjutnya ditampilkan pada Gambar 3 menggunakan diagram Demsar yang telah dimodifikasi.



Gambar 3 Perbandingan Akurasi Model Menggunakan Diagram Demsar

Berdasarkan Nemenyi *post hoc* dan diagram Demsar yang dimodifikasi di atas menunjukkan bahwa akurasi model AB+NB dan BG+NB tidak meningkat atau menurun secara signifikan terhadap model NB. Menurut uji Friedman ada perbedaan secara signifikan, perbedaan yang ada adalah antara model AB+NB dengan BG+NB.

B. Sensitivitas

Uji Friedman untuk pengukuran sensitivitas dilakukan dengan cara yang sama pada pengukuran akurasi, dan diperoleh rata-rata peringkat 2,275 untuk NB, 2,275 untuk AB+NB, dan 1,45 untuk BG+NB. Selanjutnya Nemenyi *post hoc* dilakukan dengan membuat tabel perbandingan berpasangan, tabel p-value, dan tabel signifikansi perbedaan untuk menunjukkan pasangan model mana yang memiliki perbedaan secara signifikan.

Tabel 17 Perbandingan Berpasangan pada Nemenyi *Post Hoc*

	NB	AB+NB	BG+NB
NB	0	0	0,825
AB+NB	0	0	0,825
BG+NB	-0,825	-0,825	0

Untuk menghitung nilai P-value Nemenyi *post hoc* digunakan software XLSTAT. P-value yang bernilai lebih kecil dari 0,05 (nilai α) dicetak lebih tebal, ini menunjukkan bahwa ada perbedaan yang signifikan di antara model pada baris dan kolom yang sesuai.

Tabel 18 P-value pada Nemenyi *Post Hoc*

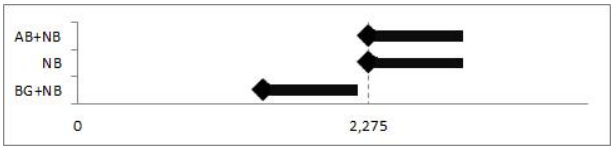
	NB	AB+NB	BG+NB
NB	1	1	**0,024644**
AB+NB	1	1	**0,024644**
BG+NB	**0,024644**	**0,024644**	1

Model pada kolom dan baris yang memiliki perbedaan signifikan, diberi nilai Sig. Sedangkan yang perbedaannya tidak signifikan diberi nilai Not.

Tabel 19 Perbedaan Signifikan pada Nemenyi *Post Hoc*

	NB	AB+NB	BG+NB
NB		Not	**Sig**
AB+NB	Not		**Sig**
BG+NB	**Sig**	**Sig**	

Setelah dilakukan uji Friedman dan Nemenyi *post hoc*, selanjutnya ditampilkan pada Gambar 4 menggunakan diagram Demsar yang telah dimodifikasi.



Gambar 4 Perbandingan Sensitivitas Model Menggunakan Diagram Demsar

Berdasarkan Nemenyi *post hoc* dan diagram Demsar yang dimodifikasi di atas menunjukkan bahwa sensitivitas model BG+NB meningkat secara signifikan dibanding model NB maupun model AB+NB, sedangkan model AB+NB tidak meningkat secara signifikan dibanding model NB.

C. G-Mean

Uji Friedman untuk pengukuran sensitivitas dilakukan dengan cara yang sama pada pengukuran akurasi, dan diperoleh rata-rata peringkat 2,275 untuk NB, 2,2 untuk AB+NB, dan 1,525 untuk BG+NB. Selanjutnya Nemenyi *post hoc* dilakukan dengan membuat tabel perbandingan berpasangan, tabel p-value, dan tabel signifikansi perbedaan untuk menunjukkan pasangan model mana yang memiliki perbedaan secara signifikan.

Tabel 20 Perbandingan Berpasangan pada Nemenyi *Post Hoc*

	NB	AB+NB	BG+NB
NB	0	0,075	0,750
AB+NB	-0,075	0	0,675
BG+NB	-0,750	-0,675	0

Untuk menghitung nilai P-value Nemenyi *post hoc* digunakan software XLSTAT. P-value yang bernilai lebih kecil dari 0,05 (nilai α) dicetak lebih tebal, ini menunjukkan bahwa ada perbedaan yang signifikan di antara model pada baris dan kolom yang sesuai.

Tabel 21 P-value pada Nemenyi *Post Hoc*

	NB	AB+NB	BG+NB
NB	1	0,969467	**0,046560**
AB+NB	0,969467	1	0,082975
BG+NB	**0,046560**	0,082975	1

Model pada kolom dan baris yang memiliki perbedaan signifikan, diberi nilai Sig. Sedangkan yang perbedaannya tidak signifikan diberi nilai Not.

Copyright © 2015 IlmuKomputer.Com

Tabel 22 Perbedaan Signifikan pada Nemenyi *Post Hoc*

	NB	AB+NB	BG+NB
NB		Not	**Sig**
AB+NB	Not		Not
BG+NB	**Sig**	Not	

Setelah dilakukan uji Friedman dan Nemenyi *post hoc*, selanjutnya ditampilkan pada Gambar 5 menggunakan diagram Demsar yang telah dimodifikasi.

Gambar 5 Perbandingan G-Mean Model Menggunakan Diagram Demsar

Berdasarkan Nemenyi *post hoc* dan diagram Demsar yang dimodifikasi di atas menunjukkan bahwa model BG+NB memiliki nilai yang meningkat secara signifikan dibanding NB, sedangkan model AB+NB tidak mengalami peningkatan secara signifikan.

Berdasarkan uji Friedman, Nemenyi *post hoc*, dan diagram Demsar yang dimodifikasi model BG+NB (Bagging berbasis Naïve Bayes) meningkat secara signifikan terhadap model NB untuk pengukuran sensitivitas dan G-Mean, sedangkan model AB+NB (AdaBoost berbasis Naïve Bayes) tidak memberikan peningkatan secara signifikan. Sehingga disimpulkan bahwa model *ensemble* terbaik adalah model BG+NB (Bagging berbasis Naïve Bayes). Hasil penelitian ini bertentangan dengan penelitian yang menyatakan bahwa Naïve Bayes merupakan pembelajar stabil yang tidak dapat ditingkatkan dengan Bagging (Liang, Zhu, & Zhang, 2011, p. 1803), dan AdaBoost dapat meningkatkan kinerja Naïve Bayes (Korada, Kumar, & Deekshitulu, 2012, p. 73). Tetapi menegaskan hasil penelitian yang menyatakan bahwa Boosting tidak bekerja dengan baik pada pengklasifikasi Naïve Bayes, karena merupakan pengklasifikasi yang stabil dengan bias yang kuat (Ting & Zheng , 2003, p. 199).

5 KESIMPULAN

Dua metode *ensemble*, yaitu AdaBoost dan Bagging, telah diusulkan untuk meningkatkan kinerja pengklasifikasi Naïve Bayes dan dilakukan pengukuran kinerjanya. Selanjutnya dilakukan uji statistik terhadap hasil pengukuran menggunakan uji Friedman untuk mengetahui signifikansi perbedaan antarmodel. Untuk pengukuran yang memiliki perbedaan signifikan pada uji Friedman dilakukan Nemenyi *post hoc*, meliputi perbandingan berpasangan, menghitung p-value, dan membuat tabel signifikansi, serta dibuatkan diagram Demsar yang dimodifikasi. Berdasarkan hasil eksperimen yang telah dilakukan menunjukkan bahwa model yang mengintegrasikan Bagging dengan pengklasifikasi Naïve Bayes (BG+NB) memberikan hasil terbaik. Walaupun akurasi secara umum tidak meningkat, tetapi kemampuan memprediksi kelas minoritas meningkat secara signifikan berdasarkan pengukuran sensitivitas dan G-Mean. Sehingga disimpulkan bahwa teknik *ensemble* Bagging lebih baik daripada AdaBoost ketika diintegrasikan dengan pengklasifikasi Naïve Bayes pada prediksi cacat software.

REFERENSI

Afza, A. J., Farid, D. M., & Rahman, C. M. (2011). A Hybrid Classifier using Boosting, Clustering, and Naïve Bayesian Classifier. World of Computer Science and Information Technology Journal (WCSIT), *105-109.*

Alfaro, E., Gamez, M., & Garcia, N. (2013). adabag: An R Package for Classification with Boosting and Bagging. Journal of Statistical Software, *1-35.*

Attenberg, J., & Ertekin, S. (2013). Class Imbalance and Active Learning. In H. He, & Y. Ma, Imbalanced Learning: Foundations, Algorithms, and Applications *(pp. 101-149). New Jersey: John Wiley & Sons.*

Bramer, M. (2007). Principles of Data Mining. *London: Springer.*

Carver, R. H., & Nash, J. G. (2012). Doing Data Analysis with SPSS® Version 18. *Boston: Cengage Learning.*

Catal, C. (2012). Performance Evaluation Metrics for Software Fault Prediction Studies. Acta Polytechnica Hungarica, *9(4), 193-206.*

Chiş, M. (2008). Evolutionary Decision Trees and Software Metrics for Module Defects Identification. World Academy of Science, Engineering and Technology, *273-277.*

Corder, G. W., & Foreman, D. I. (2009). Nonparametric Statistics for Non-statisticians: A Step-by-step Approach. *New Jersey: John Wiley & Sons.*

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

Dubey, R., Zhou, J., Wang, Y., Thompson, P. M., & Ye, J. (2014). Analysis of Sampling Techniques for Imbalanced Data: An n = 648 ADNI Study. NeuroImage, *220–241.*

Fakhrahmad, S. M., & Sami, A. (2009). Effective Estimation of Modules' Metrics in Software Defect Prediction. Proceedings of the World Congress on Engineering *(pp. 206-211). London: Newswood Limited.*

Galar, M., Fernández, A., Barrenechea, E., & Herrera, F. (2013). EUSBoost: Enhancing Ensembles for Highly Imbalanced Data-sets by Evolutionary Under Sampling. Pattern Recognition, *3460–3471.*

García, S., Fernández, A., Luengo, J., & Herrera, F. (2010). Advanced Nonparametric Tests for Multiple Comparisons in the Design of Experiments in Computational Intelligence and Data Mining: Experimental Analysis of Power. Information Sciences, *2044–2064.*

Gayatri, N., Nickolas, S., Reddy, A., & Chitra, R. (2009). Performance Analysis Of Data Mining Algorithms for Software Quality Prediction. International Conference on Advances in Recent Technologies in Communication and Computing *(pp. 393-395). Kottayam: IEEE Computer Society.*

Gorunescu, F. (2011). Data Mining: Concepts, Models and Techniques. *Berlin: Springer-Verlag.*

Gray, D., Bowes, D., Davey, N., Sun, Y., & Christianson, B. (2011). The Misuse of the NASA Metrics Data Program Data Sets for Automated Software Defect Prediction. Evaluation & Assessment in Software Engineering (EASE 2011), 15th Annual Conference on, *(pp. 96-103). Durham.*

Hall, T., Beecham, S., Bowes, D., Gray, D., & Counsell, S. (2011). A Systematic Literature Review on Fault Prediction Performance in Software Engineering. IEEE Transactions on Software Engineering, Accepted for publication - available online, *1-31.*

Harrington, P. (2012). Machine Learning in Action. *New York: Manning Publications Co.*

Huck, S. W. (2012). Reading Statistics and Research. *Boston: Pearson Education.*

Japkowicz, N. (2013). Assessment Metrics for Imbalanced Learning. In H. He, & Y. Ma, Imbalanced Learning: Foundations, Algorithms, and Applications *(pp. 187-206). New Jersey: John Wiley & Sons.*

Khoshgoftaar, T. M., Gao, K., & Seliya, N. (2010). Attribute Selection and Imbalanced Data: Problems in Software Defect Prediction. International Conference on Tools with Artificial Intelligence *(pp. 137-144). IEEE Computer Society.*

Korada, N. K., Kumar, N. P., & Deekshitulu, Y. (2012). Implementation of Naïve Bayesian Classifier and Ada-Boost

Copyright © 2015 IlmuKomputer.Com

Algorithm Using Maize Expert System. International Journal of Information Sciences and Techniques (IJIST) Vol.2, No.3, *63-75.*

Lessmann, 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, *485-496.*

Liang, G., & Zhang, C. (2011). Empirical Study of Bagging Predictors on Medical Data. Proceedings of the 9-th Australasian Data Mining Conference (AusDM'11) *(pp. 31-40). Ballarat, Australia: Australian Computer Society, Inc.*

Liang, G., Zhu, X., & Zhang, C. (2011). An Empirical Study of Bagging Predictors for Different Learning Algorithms. Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence *(pp. 1802-1803). California: AAAI Press.*

Liu, X.-Y., & Zhou, Z.-H. (2013). Ensemble Methods for Class Imbalance Learning. In H. He, & Y. Ma, Imbalanced Learning: Foundations, Algorithms, and Applications *(pp. 61-82). New Jersey: John Wiley & Sons.*

López, V., Fernández, A., & Herrera, F. (2014). On the Importance of the Validation Technique for Classification with Imbalanced Datasets: Addressing Covariate Shift when Data is Skewed. Information Sciences, *1–13. doi:10.1016/j.ins.2013.09.038*

McDonald, M., Musson, R., & Smith, R. (2008). The Practical Guide to Defect Prevention. *Washington: Microsoft Press.*

Menzies, T., Greenwald, J., & Frank, A. (2007). Data Mining Static Code Attributes to Learn Defect Predictors. IEEE Transactions on Software Engineering, *1-12.*

Myrtveit, I., Stensrud, E., & Shepperd, M. (2005). Reliability and Validity in Comparative Studies of Software Prediction Models. IEEE Transactions on Software Engineering, *380-391.*

Peng, Y., & Yao, J. (2010). AdaOUBoost: Adaptive Over-sampling and Under-sampling to Boost the Concept Learning in Large Scale Imbalanced Data Sets. Proceedings of the international conference on Multimedia information retrieval *(pp. 111-118). Philadelphia, Pennsylvania, USA: ACM.*

Riquelme, J. C., Ruiz, R., Rodriguez, D., & Moreno, J. (2008). Finding Defective Modules From Highly Unbalanced Datasets. Actas de los Talleres de las Jornadas de Ingeniería del Software y Bases de Datos *(pp. 67-74). Gijon, España: SISTEDES.*

Rodriguez, D., Herraiz, I., Harrison, R., Dolado, J., & Riquelme, J. C. (2014). Preliminary Comparison of Techniques for Dealing with Imbalance in Software Defect Prediction. 18th International Conference on Evaluation and Assessment in Software Engineering (EASE 2014) *(pp. 371-380). New York: ACM. doi:10.1145/2601248.2601294*

Seiffert, C., Khoshgoftaar, T. M., Hulse, J. V., & Folleco, A. (2011). An Empirical Study of the Classification Performance of Learners on Imbalanced and Noisy Software Quality Data. Information Sciences, *1-25.*

Seiffert, C., Khoshgoftaar, T. M., Hulse, J. V., & Napolitano, A. (2008). Resampling or Reweighting: A Comparison of Boosting Implementations. 20th IEEE International Conference on Tools with Artificial Intelligence, *445-451.*

Shepperd, M., Song, Q., Sun, Z., & Mair, C. (2013). Data Quality: Some Comments on the NASA Software Defect Data Sets. IEEE Transactions on Software Engineering, *1208-1215. doi:10.1109/TSE.2013.11*

Shull, F., Basili, V., Boehm, B., Brown, A. W., Costa, P., Lindvall, M., . . . Zelkowitz, M. (2002). What We Have Learned About Fighting Defects. METRICS '02 Proceedings of the 8th International Symposium on Software Metrics *(pp. 249-258). Washington: IEEE Computer Society.*

Singh, P., & Verma, S. (2014). An Efficient Software Fault Prediction Model using Cluster Based Classification. International Journal of Applied Information Systems (IJAIS), *7(3), 35-41.*

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

Strangio, M. A. (2009). Recent Advances in Technologies. *Vukovar: In-Teh.*

Sun, Y., Mohamed, K. S., Wong, A. K., & Wang, Y. (2007). Cost-sensitive Boosting for Classification of Imbalanced Data. Pattern Recognition Society, *3358-3378.*

Tao, W., & Wei-hua, L. (2010). Naïve Bayes Software Defect Prediction Model. Computational Intelligence and Software Engineering (CiSE) *(pp. 1-4). Wuhan: IEEE Computer Society. doi:10.1109/CISE.2010.5677057*

Ting, K. M., & Zheng , Z. (2003). A Study of AdaBoost with Naive Bayesian Classifiers: Weakness and Improvement. Computational Intelligence, 19*(2), 186-200. doi:10.1111/1467-8640.00219*

Turhan, B., & Bener, A. (2007). Software Defect Prediction: Heuristics for Weighted Naive Bayes. Proceedings of the 2nd International Conference on Software and Data Technologies (ICSOFT'07), *(pp. 244-249).*

Wahono, R. S., Suryana, N., & Ahmad, S. (2014, May). Metaheuristic Optimization based Feature Selection for Software Defect Prediction. Journal of Software, 9*(5), 1324-1333. doi:10.4304/jsw.9.5.1324-1333*

Wang, S., & Yao, X. (2013). Using Class Imbalance Learning for Software Defect Prediction. IEEE Transactions on Reliability, *434-443.*

Weiss, G. M. (2013). Foundations of Imbalanced Learning. In H. He, & Y. Ma, Imbalanced Learning: Foundations, Algorithms, and Applications *(pp. 13-41). New Jersey: John Wiley & Sons.*

Yap, B. W., Rani, K. A., Rahman, H. 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*, pp. 13-22. Singapore: Springer. doi:10.1007/978-981-4585-18-7_2*

Zhang, D., Liu, W., Gong, X., & Jin, H. (2011). A Novel Improved SMOTE Resampling Algorithm Based on Fractal. Computational Information Systems, *2204-2211.*

Zhang, H., & Wang, Z. (2011). A Normal Distribution-Based Over-Sampling Approach to Imbalanced Data Classification. Advanced Data Mining and Applications - 7th International Conference *(pp. 83-96). Beijing: Springer.*

Zhang, H., Jiang, L., & Su, J. (2005). Augmenting Naïve Bayes for Ranking. ICML '05 Proceedings of the 22nd international conference on Machine learning *(pp. 1020 - 1027). New York: ACM Press. doi:http://dx.doi.org/10.1145/1102351.1102480*

Zhou, Z.-H., & Yu, Y. (2009). The Top Ten Algorithms in Data Mining. *(X. Wu, & V. Kumar, Eds.) Florida: Chapman & Hall/CRC.*

BIOGRAFI PENULIS

Aries Saifudin. Memperoleh gelar A.Md. di bidang Teknik Elektronika dari Politeknik Universitas Brawijaya, Malang, gelar S.T. di bidang Teknik Informatika dari Universitas Mercu Buana, Jakarta, dan gelar M.Kom di bidang Rekayasa Perangkat Lunak dari STMIK ERESHA, Jakarta. Dia sebagai dosen tetap di Universitas Pamulang. 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.

Copyright © 2015 IlmuKomputer.Com