Pendekatan Level Data untuk Menangani Ketidakseimbangan Kelas pada Prediksi Cacat Software

Aries Saifudin[1] dan Romi Satria Wahono[2]

[1]*Fakultas Teknik, Universitas Pamulang*
aries.saifudin@yahoo.co.id

[2]*Fakultas Ilmu Komputer, Universitas Dian Nuswantoro*
romi@romisatriawahono.net

Abstrak: 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 ditujukan untuk memperbaiki algoritma atau menggabungkan (*ensemble*) pengklasifikasi agar lebih konduktif terhadap kelas minoritas. Pada penelitian ini diusulkan pendekatan level data dengan *resampling*, yaitu *random oversampling* (ROS), dan *random undersampling* (RUS), dan mensintesis menggunakan algoritma FSMOTE. Pengklasifikasi yang digunakan adalah Naïve Bayes. Hasil penelitian menunjukkan bahwa model FSMOTE+NB merupakan model pendekatan level data terbaik pada prediksi cacat software karena nilai sensitivitas dan G-Mean model FSMOTE+NB meningkat secara signifikan, sedangkan model ROS+NB dan RUS+NB tidak meningkat secara signifikan.

Kata Kunci: Pendekatan Level Data, 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). Cacat adalah kontributor utama untuk limbah teknologi informasi dan menyebabkan pengerjaan ulang proyek secara signifikan, keterlambatan, dan biaya lebih berjalan (Anantula & Chamarthi, 2011). Potensi cacat tertinggi terjadi pada tahap pengkodean (Jones, 2013, p. 3). Sejumlah cacat yang ditemukan di akhir proyek secara sistematis menyebabkan penyelesaian proyek melebihi jadwal (Lehtinen, Mäntylä, Vanhanen, Itkonen, & Lassenius, 2014, p. 626). Secara umum, biaya masa depan untuk koreksi cacat yang tidak terdeteksi pada rilis produk mengkonsumsi sebagian besar dari total biaya pemeliharaan (In, Baik, Kim, Yang, & Boehm, 2006, p. 86), sehingga perbaikan cacat harus dilakukan sebelum rilis.

Untuk mencari cacat software biasanya dilakukan dengan *debugging*, yaitu pencarian dengan melibatkan semua *source code*, berjalannya, keadaannya, dan riwayatnya (Weiss, Premraj, Zimmermann, & Zeller, 2007, p. 1). Hal ini membutuhkan sumber daya yang banyak, dan tidak efisien karena menggunakan asumsi dasar bahwa penulis kode program tidak dapat dipercaya.

Dengan mengurangi jumlah cacat pada software yang dihasilkan dapat meningkatkan kualitas software. Secara tradisional, software berkualitas tinggi adalah software yang tidak ditemukan cacat selama pemeriksaan dan pengujian, serta dapat memberikan nilai kepada pengguna dan memenuhi harapan mereka (McDonald, Musson, & Smith, 2008, pp. 4-6). 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 manajemen proyek 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 mencegah terjadinya 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 memperkirakan jumlah cacat dalam software, menemukan hubungan cacat, dan 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).

Klasifikasi untuk prediksi cacat software dapat dilakukan dengan menggunakan data yang dikumpulkan dari pengembangan software sebelumnya.

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

Dataset NASA (*National Aeronautics and Space Administration*) 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). Proyek NASA tentunya dikerjakan oleh banyak developer dan organisasi, sehingga memiliki banyak masalah asumsi, tetapi secara umum mereka diasumsikan memiliki kemampuan yang sama dan cenderung membuat kesalahan yang serupa. 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). Menggunakan dataset NASA merupakan pilihan yang terbaik, karena mudah diperoleh dan kinerja dari metode 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). Klasifikasi data dengan pembagian kelas yang tidak seimbang dapat menurunkan kinerja algoritma belajar (*learning algorithm*) pengklasifikasi standar secara signifikan, karena mengasumsikan distribusi kelas 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). 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*).

Untuk mencari solusi terbaik terhadap masalah ketidakseimbangan kelas (*class imbalance*), maka pada penelitian ini akan dilakukan pengukuran kinerja pendekatan level data yang dilakukan dengan *resampling*, yaitu *random oversampling* (ROS), *random undersampling* (RUS), dan mensintesis menggunakan algoritma FSMOTE. Algoritma pengklasifikasi yang digunakan adalah Naïve Bayes. Diharapkan didapat pendekatan level data terbaik untuk menyelesaikan permasalahan ketidakseimbangan kelas pada prediksi cacat software.

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*) (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.

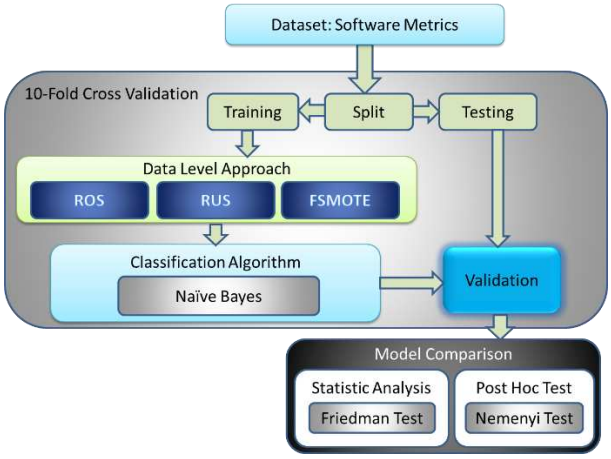
Penelitian yang dilakukan oleh Wahono, Suryana, dan Ahmad (Wahono, Suryana, & Ahmad, 2014, pp. 1324-1333) menyatakan bahwa kinerja model prediksi cacat software berkurang secara signifikan karena dataset yang digunakan mengandung *noise* (kegaduhan) dan ketidakseimbangan kelas. Pada penelitian ini, diusulkan 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 diterapkan pendekatan level data untuk mengurangi pengaruh ketidakseimbangan kelas dan meningkatkan kemampuan memprediksi kelas minoritas. Pendekatan level data akan dilakukan dengan *resampling*, yaitu *random oversampling* (ROS), *random undersampling* (RUS), dan mensintesis menggunakan algoritma FSMOTE.

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 pendekatan level data yang dilakukan dengan *resampling*, dan mensintesis data latih. Algoritma *resampling* yang digunakan adalah *random oversampling* (ROS), dan *random undersampling* (RUS), sedangkan algoritma sintesis yang digunakan adalah FSMOTE. Algoritma pengklasifikasi yang digunakan adalah Naïve Bayes. Validasi pada pengukuran kinerja digunakan *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.

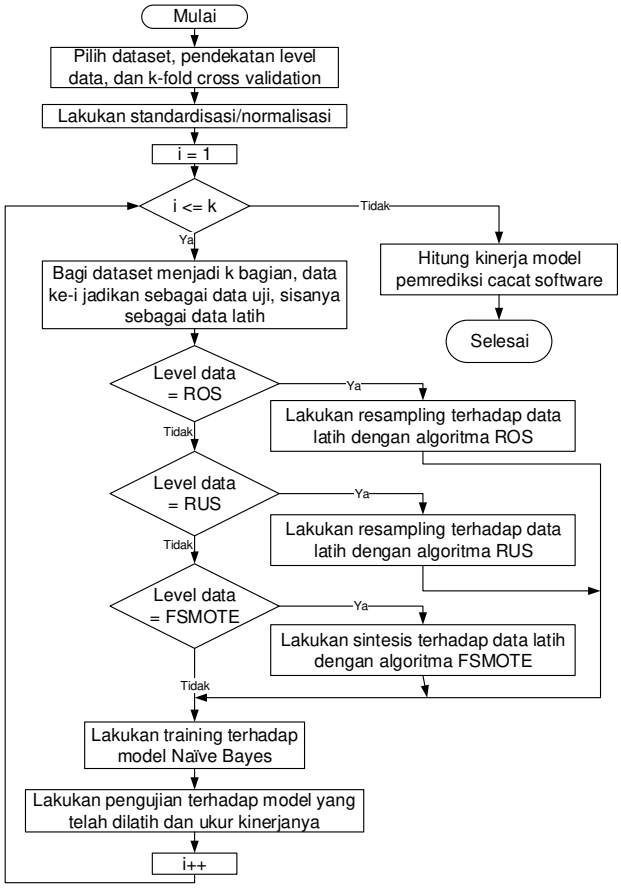


Gambar 1 Kerangka Kerja Model yang Diusulkan

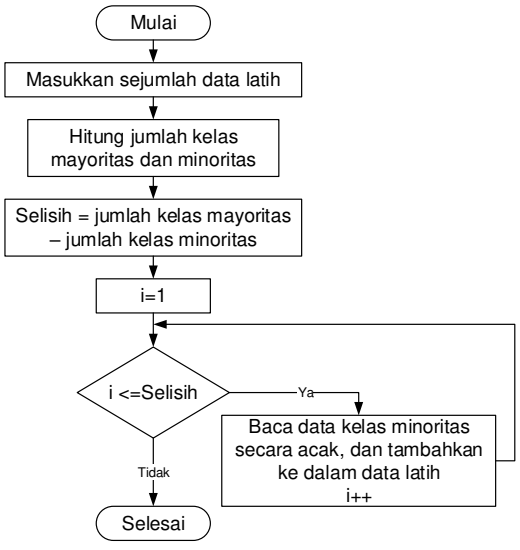
Pada model yang diusulkan, dataset *software metrics* akan dibagi menjadi 10 bagian. Secara berurutan setiap bagian dijadikan sebagai data uji, sedangkan bagian lain sebagai data latih. Data latih akan diseimbangkan menggunakan algoritma *resampling* dan sintesis. Algoritma *resampling* yang digunakan adalah *random oversampling* (ROS), dan *random undersampling* (RUS). Sedangkan algoritma sintesis yang digunakan adalah FSMOTE. Data latih yang sudah diseimbangkan digunakan untuk melatih algoritma pengklasifikasi Naïve Bayes, dan diuji dengan data uji, kemudian diukur kinerjanya. Proses ini ditunjukkan dengan flowchart pada Gambar 2.

Pada algoritma ROS, data kelas minoritas dipilih secara acak, kemudian ditambahkan ke dalam data latih. Proses pemilihan dan penambahan ini diulang-ulang sampai jumlah data kelas minoritas sama dengan jumlah kelas mayoritas.

Algorima ROS digambarkan menggunakan *flowchart* pada Gambar 3. Pertama dihitung selisih antara kelas mayoritas dengan kelas minoritas. Kemudian dilakukan perulangan sebanyak hasil penghitungan selisih sambil membaca data kelas minoritas secara acak, dan ditambahkan ke dalam data latih.

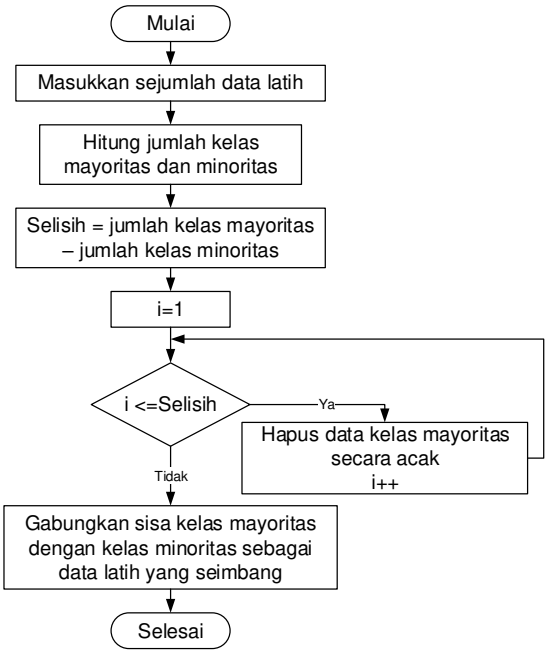


Gambar 2 Flowchart Model yang Diusulkan



Gambar 3 *Flowchart* Algoritma ROS (*Random Oversampling*)

Algoritma RUS digambarkan dengan *flowchart* pada Gambar 4. Hampir sama dengan ROS, pertama dihitung selisih antara kelas mayoritas dengan kelas minoritas. Kemudian dilakukan perulangan sebanyak hasil penghitungan selisih kelas mayoritas dan minoritas. Selama perulangan data kelas mayoritas dihapus secara acak, sehingga jumlah kelas mayoritas sama dengan jumlah kelas minoritas.



Gambar 4 *Flowchart* Algoritma RUS (*Random Undersampling*)

FSMOTE adalah algoritma sintesis yang ditujukan untuk memperbaiki SMOTE dengan mensintesis data latih mengikuti teori interpolasi fraktal, sehingga data yang dihasilkan lebih representatif, dan menghasilkan kinerja lebih baik dari pada SMOTE (Zhang, Liu, Gong, & Jin, 2011, p. 2210). Algoritma FSMOTE terdiri dari dua bagian utama, bagian pertama berisi perulangan untuk mencari k tetangga terdekat, dan bagian kedua untuk membuat sintesis data kelas minoritas berdasarkan interpolasi fraktal. Algoritma FSMOTE ditulis dalam bentuk *pseudocode* sebagai berikut:

Masukan: Algorithm FSMOTE(T, N, k)
m: Jumlah sampel kelas minoritas T; persentase jumlah hasil sintesis FSMOTE N%; jumlah *nearest neighbors* k
Keluaran: (N/100)*T jumlah data hasil sintesis

1: /*Hitung k *nearest neighbors* untuk setiap sampel kelas minoritas.*/
2: for i = 1 to T do
3: Hitung k *nearest neighbors* data minoritas ke-i, dan simpan ke dalam nnarray
4: Sierpinski (N, i, nnarray, times); panggil fungsi (6) untuk mensistesis data
5: end for

/*Fungsi untuk mensintesis data*/
6: Sierpinski (N, i, nnarray, times)
7: Lakukan perulangan selama N !=0:
8: Pilih 2 data yang berbeda secara acak antara 1 sampai k, sebut sebagai nn2 dan nn3. Langkah ini memilih dua data di antara k data terdekat dari data minoritas ke-i.
9: /*Lakukan operasi interpolasi fraktal, pilih satu dari sampel yang disintesis sebagai sampel kelas minoritas tambahan*/
10: for indeks1to times do
11: for attr1 to numattrs do
12: Hitung jarak:

13: dif1=Sampel[nnarray[nn3]][attr]-
Sampel[nnarray[nn2]][attr]
14: dif2=Sampel[nnarray[nn3]][attr]- Sampel[i][attr]
15: dif3=Sampel[nnarray[nn2]][attr]- Sampel[i][attr]
16: Hitung data sintesis sementara:
17: SintesisSementara[0][attr]=
Sampel[nnarray[nn2]][attr]+dif1/2
18: SintesisSementara [1][attr]=Sampel[i][attr]+dif2/2
19: SintesisSementara [2][attr]=Sampel[i][attr]+dif3/2
20: Akhir for indeks1
21: Tambahkan hasil sintesis SintesisSementara[0],
SintesisSementara[1], SintesisSementara[2] ke dalam List.
22: Akhir for attr1
23: Ind = random(List.length)
/*Lakukan pemilihan secara acak untuk dijadikan hasil sintesis*/
24: Sintesis[indeksBaru]=List[Ind]
25: indeksBaru ++
26: N = N - 1
27: Akhir perulangan, kembali ke (7).
28: return /*akhir fungsi mensintesis data Sierpinski.*/
Akhir dari Pseudo-Code.

Untuk mengukur kinerja model digunakan *confusion matrix*, karena *confusion matrix* merupakan alat yang berguna untuk menganalisa seberapa baik pengklasifikasi dapat mengenali tupel/fitur dari kelas yang berbeda (Han, Kamber, & Pei, 2011, p. 365). *Confusion matrix* dapat membantu menunjukkan rincian kinerja pengklasifikasi dengan memberikan informasi jumlah fitur suatu kelas yang diklasifikasikan dengan tepat dan tidak tepat (Bramer, 2007, p. 89). *Confusion matrix* memberikan penilaian kinerja model klasifikasi berdasarkan jumlah objek yang diprediksi dengan benar dan 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 penghitungannya 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 merupakan 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, dilakukan validasi dan pengukuran kinerja model, kemudian 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

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 memenuhi syarat, maka disarankan menggunakan uji nonparametrik.

Uji t dan ANOVA tidak cocok digunakan pada penelitian *machine learning*, karena hasil pengukuran kinerja model harus terdistribusi normal, untuk memastikan bentuk distribusinya minimal 30 data, tetapi hasil pengukuran kinerja *machine learning* sering tidak mencukupi (Demšar, 2006, pp. 6-10). Jika datanya mencukupi, uji parametrik harus lebih diutamakan dari pada uji nonparametrik karena memiliki kesalahan yang lebih rendah dan memiliki kekuatan yang lebih tinggi (García, Fernández, Luengo, & Herrera, 2010, p. 2045). Tetapi jika datanya tidak mencukupi, 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 persamaan 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). Sehingga perbedaan antarmodel dapat sajikan dalam bentuk visualisasi grafis.

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 dataset dari NASA MDP *repository* dan 10 dataset dari PROMISE *repository*) ditunjukkan pada Tabel 1 sampai Tabel 5.

Tabel 1 Akurasi Model

Data Set		Model			
		NB	ROS+NB	RUS+NB	FSMOTE+NB
MDP Repository	CM1	81,65%	81,65%	79,82%	72,78%
	JM1	78,87%	78,81%	78,80%	78,80%
	KC1	74,10%	74,10%	73,41%	69,79%
	KC3	78,87%	78,87%	77,84%	76,29%
	MC2	72,58%	71,77%	73,39%	72,58%
	PC1	88,81%	88,81%	87,19%	90,72%
	PC2	88,09%	88,50%	68,28%	92,94%
	PC3	36,75%	36,75%	59,54%	56,22%
	PC4	85,59%	85,91%	84,41%	84,02%
	PC5	75,03%	75,03%	74,97%	75,27%
PROMISE Repository	CM1	82,61%	82,84%	76,66%	68,19%
	JM1	78,81%	78,86%	78,76%	78,74%
	KC1	74,01%	73,67%	73,84%	71,77%
	KC3	82,10%	82,41%	67,90%	79,32%
	MC2	72,90%	72,90%	74,19%	74,19%
	PC1	89,66%	89,34%	87,27%	85,09%
	PC2	92,51%	92,44%	68,87%	95,08%
	PC3	47,34%	44,64%	57,56%	65,08%
	PC4	85,43%	85,43%	82,99%	83,15%
	PC5	74,73%	74,85%	74,62%	75,15%

Tabel 2 Sensitifitas Model

Data Set		Model			
		NB	ROS+NB	RUS+NB	FSMOTE+NB
MDP Repository	CM1	30,95%	33,33%	35,71%	42,86%
	JM1	19,85%	19,60%	19,98%	21,46%
	KC1	33,67%	34,01%	32,65%	45,58%
	KC3	36,11%	36,11%	36,11%	22,22%
	MC2	38,64%	36,36%	43,18%	38,64%
	PC1	40,00%	40,00%	43,64%	30,91%
	PC2	12,50%	12,50%	62,50%	25,00%
	PC3	90,77%	89,23%	70,77%	74,62%
	PC4	28,98%	28,98%	31,82%	47,73%
	PC5	22,27%	22,05%	24,02%	25,33%
PROMISE Repository	CM1	32,61%	32,61%	32,61%	47,83%
	JM1	19,73%	20,10%	20,16%	23,70%
	KC1	33,67%	34,01%	33,33%	47,28%
	KC3	42,86%	42,86%	52,38%	45,24%
	MC2	35,29%	35,29%	39,22%	39,22%
	PC1	35,00%	36,67%	40,00%	41,67%
	PC2	23,81%	23,81%	47,62%	19,05%
	PC3	85,14%	85,14%	69,60%	65,54%
	PC4	28,98%	30,11%	28,98%	46,59%
	PC5	21,62%	22,05%	22,05%	31,66%

Tabel 3 F-Measure Model

Data Set		Model			
		NB	ROS+NB	RUS+NB	FSMOTE+NB
MDP Repository	CM1	0,302	0,318	0,313	0,288
	JM1	0,282	0,279	0,282	0,297
	KC1	0,397	0,399	0,383	0,433
	KC3	0,388	0,388	0,377	0,258
	MC2	0,500	0,478	0,535	0,500
	PC1	0,367	0,367	0,356	0,351
	PC2	0,044	0,046	0,080	0,136
	PC3	0,262	0,258	0,302	0,296
	PC4	0,358	0,363	0,361	0,453
	PC5	0,325	0,323	0,342	0,356
PROMISE Repository	CM1	0,283	0,286	0,227	0,240
	JM1	0,280	0,284	0,284	0,318
	KC1	0,396	0,395	0,392	0,459
	KC3	0,383	0,387	0,297	0,362
	MC2	0,462	0,462	0,500	0,500
	PC1	0,307	0,310	0,291	0,267
	PC2	0,089	0,088	0,045	0,107
	PC3	0,254	0,244	0,256	0,283
	PC4	0,355	0,364	0,321	0,434
	PC5	0,316	0,322	0,320	0,408

Tabel 4 G-Mean Model

Data Set		Model			
		NB	ROS+NB	RUS+NB	FSMOTE+NB
MDP Repository	CM1	0,525	0,544	0,555	0,575
	JM1	0,433	0,430	0,434	0,449
	KC1	0,544	0,546	0,534	0,596
	KC3	0,566	0,566	0,562	0,444
	MC2	0,594	0,576	0,623	0,594
	PC1	0,610	0,610	0,630	0,545
	PC2	0,335	0,336	0,654	0,486
	PC3	0,514	0,512	0,640	0,633
	PC4	0,524	0,525	0,544	0,655
	PC5	0,459	0,457	0,475	0,487
PROMISE Repository	CM1	0,537	0,538	0,517	0,581
	JM1	0,432	0,436	0,436	0,470
	KC1	0,543	0,544	0,540	0,615
	KC3	0,614	0,615	0,606	0,618
	MC2	0,568	0,568	0,599	0,599
	PC1	0,572	0,584	0,602	0,606
	PC2	0,472	0,472	0,574	0,428
	PC3	0,604	0,583	0,625	0,653
	PC4	0,523	0,533	0,515	0,644
	PC5	0,452	0,456	0,456	0,538

Tabel 5 AUC Model

Data Set		Model			
		NB	ROS+NB	RUS+NB	FSMOTE+NB
MDP Repository	CM1	0,600	0,611	0,610	0,600
	JM1	0,572	0,570	0,571	0,577
	KC1	0,607	0,608	0,599	0,618
	KC3	0,624	0,624	0,617	0,554
	MC2	0,649	0,638	0,666	0,649
	PC1	0,666	0,666	0,673	0,635
	PC2	0,512	0,514	0,655	0,597
	PC3	0,600	0,593	0,644	0,641
	PC4	0,618	0,620	0,623	0,688
	PC5	0,584	0,584	0,589	0,595
PROMISE Repository	CM1	0,605	0,607	0,572	0,592
	JM1	0,571	0,572	0,572	0,585
	KC1	0,607	0,606	0,604	0,637
	KC3	0,654	0,656	0,613	0,648
	MC2	0,633	0,633	0,653	0,653
	PC1	0,642	0,648	0,653	0,649
	PC2	0,587	0,587	0,584	0,577
	PC3	0,640	0,625	0,629	0,653
	PC4	0,617	0,622	0,603	0,678
	PC5	0,580	0,582	0,581	0,615

Untuk mengetahui pendekatan level data manakah antara ROS, RUS, dan FSMOTE yang dapat mengurangi pengaruh ketidakseimbangan kelas, dan 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* (3,633), jumlah model (K=4), dan jumlah dataset (D=20), maka nilainya dihitung sebagai berikut:

$$CD = q_{\propto,\infty,K}\sqrt{\frac{K(K+1)}{12D}} = 3{,}633\sqrt{\frac{4(4+1)}{12\cdot 20}} = 1{,}048757$$

Untuk mengetahui signifikansi perbedaan di antara keempat model, dilakukan uji Friedman menggunakan aplikasi SPSS. Nilai p-value dari uji Friedman ditunjukkan pada Tabel 6.

Tabel 6 Rekap P-Value pada Uji Friedman untuk Model yang Mengintegrasikan Pendekatan Level Data

Pengukuran	P-Value	Hasil (α = 0,05)
Akurasi	0,08998	Not Sig., Tidak ada perbedaan nilai di antara model
Sensitivitas	0,01143	Sig., Ada perbedaan nilai di antara model
F-Measure	0,13036	Not Sig., Tidak ada perbedaan nilai di antara model
G-Mean	0,00084	Sig., Ada perbedaan nilai di antara model
AUC	0,19285	Not Sig., Tidak ada perbedaan nilai di antara model

Hasil pada Tabel 6 menunjukkan bahwa nilai sensitivitas, dan G-Mean memiliki perbedaan secara signifikan. 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. Sensitivitas

Uji Friedman dilakukan dengan memberikan peringkat setiap nilai pengukuran sensitivitas model yang telah ditunjukkan pada Tabel 2. Peringkat diberikan pada model untuk setiap dataset, nilai terbesar diberi peringkat 1, terbesar kedua diberi peringkat 2, dan seterusnya. Jika ada beberapa nilai yang sama, maka diberi peringkat rata-rata. Hasilnya ditunjukkan pada Tabel 7.

Tabel 7 Peringkat Sensitivitas Model pada Uji Friedman

Data Set		Model			
		NB	ROS+NB	RUS+NB	FSMOTE+NB
MDP Repository	CM1	4	3	2	1
	JM1	3	4	2	1
	KC1	3	2	4	1
	KC3	2	2	2	4
	MC2	2,5	4	1	2,5
	PC1	2,5	2,5	1	4
	PC2	3,5	3,5	1	2
	PC3	1	2	4	3
	PC4	3,5	3,5	2	1
	PC5	3	4	2	1
PROMISE Repository	CM1	3	3	3	1
	JM1	4	3	2	1
	KC1	3	2	4	1
	KC3	3,5	3,5	1	2
	MC2	3,5	3,5	1,5	1,5
	PC1	4	3	2	1
	PC2	2,5	2,5	1	4
	PC3	1,5	1,5	3	4
	PC4	3,5	2	3,5	1
	PC5	4	2,5	2,5	1
Jumlah		60,5	57	44,5	38
Rata-rata		3,025	2,85	2,225	1,9

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 8 Perbandingan Berpasangan pada Nemenyi Post Hoc

	NB	ROS+NB	RUS+NB	FSMOTE+NB
NB	0	0,175	0,8	1,125
ROS+NB	-0,175	0	0,625	0,95
RUS+NB	-0,8	-0,625	0	0,325
FSMOTE+NB	-1,125	-0,95	-0,325	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 9 P-value pada Nemenyi Post Hoc

	NB	ROS+NB	RUS+NB	FSMOTE+NB
NB	1	0,973579434	0,203469636	**0,029872996**
ROS+NB	0,973579434	1	0,418872526	0,091935992
RUS+NB	0,203469636	0,418872526	1	0,85625123
FSMOTE+NB	**0,029872996**	0,091935992	0,85625123	1

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

Tabel 10 Perbedaan Signifikan pada Nemenyi Post Hoc

	NB	ROS+NB	RUS+NB	FSMOTE+NB
NB		Not	Not	**Sig**
ROS+NB	Not		Not	Not
RUS+NB	Not	Not		Not
FSMOTE+NB	**Sig**	Not	Not	

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

Gambar 5 Perbandingan Sensitivitas Model Menggunakan Diagram Demsar

Berdasarkan Nemenyi post hoc dan diagram Demsar yang dimodifikasi di atas menunjukkan bahwa sensitivitas model FSMOTE+NB meningkat secara signifikan dibanding model NB. Sedangkan model ROS+NB dan RUS+NB menunjukkan peningkatan, tetapi tidak signifikan.

B. G-Mean

Uji Friedman dilakukan dengan memberikan peringkat setiap nilai pengukuran G-Mean model yang telah ditunjukkan pada Tabel 4. Peringkat diberikan pada model untuk setiap dataset, nilai terbesar diberi peringkat 1, terbesar kedua diberi peringkat 2, dan seterusnya. Jika ada beberapa nilai yang sama, maka diberi peringkat rata-rata. Hasilnya ditunjukkan pada Tabel 11.

Tabel 11 Peringkat G-Mean Model pada Uji Friedman

Data Set		Model			
		NB	ROS+NB	RUS+NB	FSMOTE+NB
MDP Repository	CM1	4	3	2	1
	JM1	3	4	2	1
	KC1	3	2	4	1
	KC3	1,5	1,5	3	4
	MC2	2,5	4	1	2,5
	PC1	2,5	2,5	1	4
	PC2	4	3	1	2
	PC3	3	4	1	2
	PC4	4	3	2	1
	PC5	3	4	2	1
PROMISE Repository	CM1	3	2	4	1
	JM1	4	2,5	2,5	1
	KC1	3	2	4	1
	KC3	3	2	4	1
	MC2	3,5	3,5	1,5	1,5
	PC1	4	3	2	1
	PC2	2,5	2,5	1	4
	PC3	3	4	2	1
	PC4	3	2	4	1
	PC5	4	2,5	2,5	1
Jumlah		**63,5**	**57**	**46,5**	**33**
Rata-rata		**3,175**	**2,85**	**2,325**	**1,65**

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 12 Perbandingan Berpasangan pada Nemenyi Post Hoc

	NB	ROS+NB	RUS+NB	FSMOTE+NB
NB	0	0,325	0,85	1,525
ROS+NB	-0,325	0	0,525	1,2
RUS+NB	-0,85	-0,525	0	0,675
FSMOTE+NB	-1,525	-1,2	-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 13 P-value pada Nemenyi Post Hoc

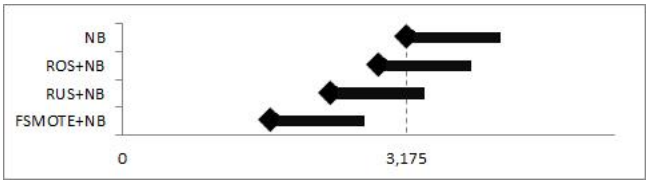
	NB	ROS+NB	RUS+NB	FSMOTE+NB
NB	1	0,85625123	0,158924874	**0,001074849**
ROS+NB	0,85625123	1	0,571872541	**0,017325155**
RUS+NB	0,158924874	0,571872541	1	0,348681121
FSMOTE+NB	**0,001074849**	**0,017325155**	0,348681121	1

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

Tabel 14 Perbedaan Signifikan pada Nemenyi Post Hoc

	NB	ROS+NB	RUS+NB	FSMOTE+NB
NB		Not	Not	**Sig**
ROS+NB	Not		Not	**Sig**
RUS+NB	Not	Not		Not
FSMOTE+NB	**Sig**	**Sig**	Not	

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



Gambar 6 Perbandingan G-Mean Model Menggunakan Diagram Demsar

Berdasarkan Nemenyi post hoc dan diagram Demsar yang dimodifikasi di atas menunjukkan bahwa nilai G-Mean model FSMOTE+NB meningkat secara signifikan terhadap model NB. Sedangkan model ROS+NB dan RUS+NB menunjukkan peningkatan, tetapi tidak signifikan.

Berdasarkan uji Friedman, Nemenyi post hoc, dan diagram Demsar yang dimodifikasi menunjukkan bahwa nilai sensitivitas dan G-Mean model FSMOTE+NB meningkat secara signifikan, sedangkan model ROS+NB dan RUS+NB tidak meningkat secara signifikan. Hal ini bertentangan dengan penelitian yang menyatakan bahwa RUS menghasilkan kinerja terbaik dari pada pendekatan level data yang lain (Seiffert, Khoshgoftaar, Hulse, & Napolitano, 2008, p. 309). Dari diagram Demsar yang telah dimodifikasi menunjukkan bahwa RUS+NB lebih baik dari ROS+NB walaupun tidak signifikan, hal ini bertentangan dengan penelitian yang menyatakan bahwa ROS lebih baik daripada RUS (Batuwita & Palade, 2010, p. 8). Tetapi mendukung penelitian yang menyatakan SMOTE lebih baik daripada *resampling* acak (Dubey, Zhou, Wang, Thompson, & Ye, 2014, p. 234), dan ditingkatkan menggunakan algoritma FSMOTE dengan mengikuti interpolasi fraktal (Zhang, Liu, Gong, & Jin, 2011, p. 2210).

5 KESIMPULAN

Pendekatan level data untuk mengurangi pengaruh ketidakseimbangan kelas menggunakan dua algoritma *resampling*, yaitu *random oversampling* (ROS) dan *random undersampling* (RUS), dan satu algoritma sintesis, yaitu FSMOTE telah diimplementasikan. Dataset baru yang dihasilkan dari masing-masing algoritma tersebut digunakan untuk melatih pengklasifikasi Naïve Bayes. Kinerja yang dihasilkan diukur dan dilakukan uji statistik. Uji statistik dilakukan dengan uji Friedman untuk mengetahui signifikansi perbedaan antarmodel. Pada pengukuran yang memiliki perbedaan signifikan dilakukan Nemenyi post hoc, meliputi perbandingan berpasangan, menghitung p-value, dan membuat tabel signifikansi, serta dibuatkan diagram Demsar yang dimodifikasi. Hasil penelitian menunjukkan bahwa model FSMOTE+NB merupakan model pendekatan level data terbaik pada prediksi cacat software.

REFERENSI

Anantula, P. R., & Chamarthi, R. (2011). Defect Prediction and Analysis Using ODC Approach in a Web Application. (IJCSIT) International Journal of Computer Science and Information Technologies, 2*(5), 2242-2245.*

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.*

Batuwita, R., & Palade, V. (2010). Efficient Resampling Methods for Training Support Vector Machines with Imbalanced Datasets. Proceedings of the International Joint Conference on Neural Networks (IJCNN) *(pp. 1-8). Barcelona: IEEE Computer Society. doi:10.1109/IJCNN.2010.5596787*

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.*

Han, J., Kamber, M., & Pei, J. (2011). Data Mining: Concepts and Techniques *(3rd ed.). San Francisco: Morgan Kaufmann Publishers Inc.*

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

In, H. P., Baik, J., Kim, S., Yang, Y., & Boehm, B. (2006, December). A Quality-Based Cost Estimation Model for the Product Line Life Cycle. Communications of the ACM, 49*(12), 85-88. doi:10.1145/1183236.1183273*

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.*

Jones, C. (2013). Software Defect Origins and Removal Methods. *Namcook Analytics.*

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.*

Lehtinen, T. O., Mäntylä, M. V., Vanhanen, J., Itkonen, J., & Lassenius, C. (2014). Perceived Causes of Software Project Failures - An Analysis of Their Relationships. Information and Software Technology*, 623–643.*

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.*

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). Building Useful Models from Imbalanced Data with Sampling and Boosting. Proceedings of the Twenty-First International Florida Artificial Intelligence Research Society Conference *(pp. 306-311). California: AAAI Press.*

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.*

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*

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, C., Premraj, R., Zimmermann, T., & Zeller, A. (2007). How Long will it Take to Fix This Bug? Fourth International Workshop on Mining Software Repositories (MSR'07) *(pp. 1-8). Washington: IEEE Computer Society.*

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*

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. di bidang Ilmu Komputer dari Saitama University, Japan, dan gelar Ph.D. di bidang Software Engineering dari Universiti Teknikal Malaysia Melaka. Dia saat ini sebagai dosen program Pascasarjana Ilmu Komputer di Universitas Dian Nuswantoro, Indonesia. Dia juga pendiri dan CEO PT Brainmatics Cipta Informatika, sebuah perusahaan pengembangan perangkat lunak di Indonesia. Minat penelitiannya saat ini meliputi rekayasa perangkat lunak dan machine learning. Anggota Profesional ACM dan IEEE Computer Society.