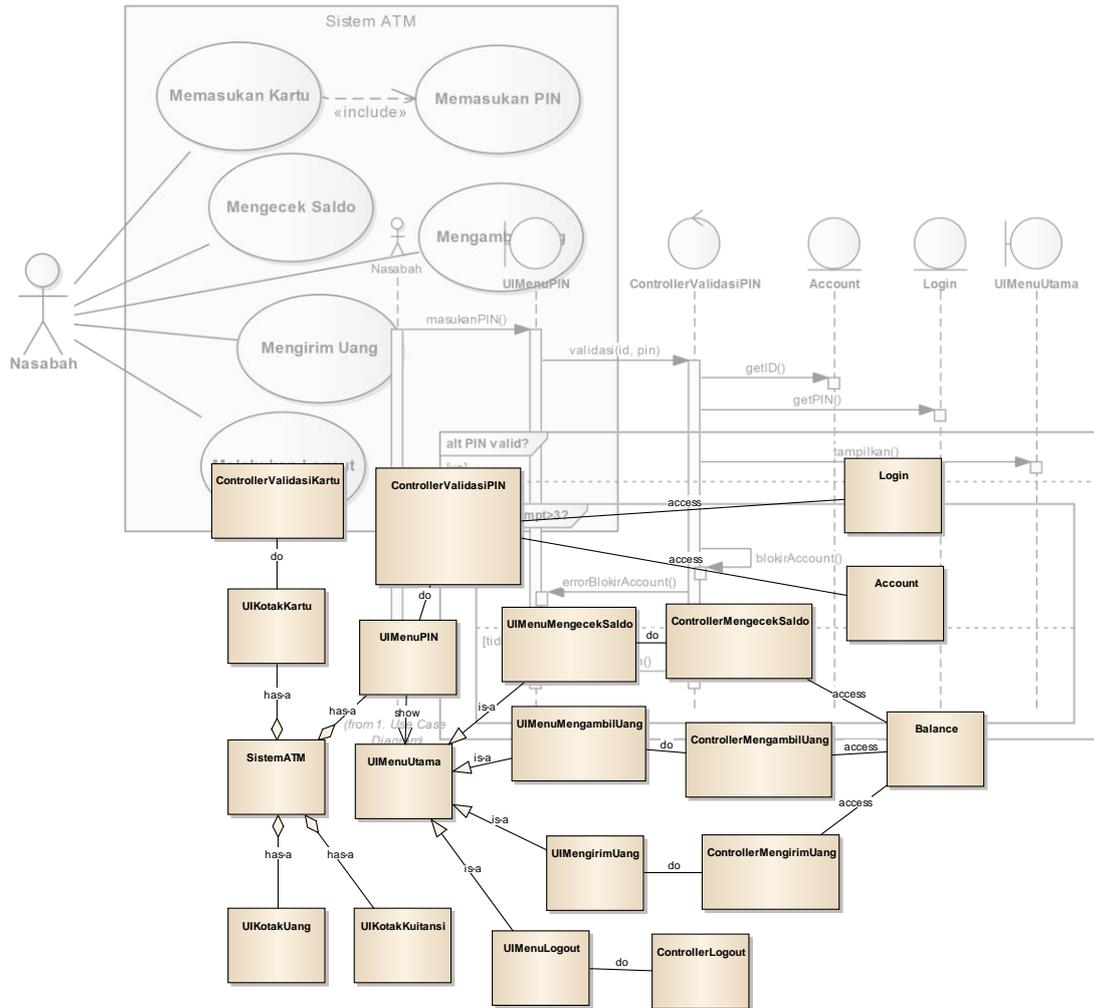


Journal of Software Engineering



Editorial Board

Editor-in-Chief: Romi Satria Wahono, M.Eng, Ph.D

Editor:

Mansyur, S.Kom

Mulyana, S.Kom

Reviewer:

Prof. Dr. Nanna Suryana Herman (Universiti Teknikal Malaysia)

Affandy, Ph.D (Universitas Dian Nuswantoro)

Hendro Subagyo, M.Eng (Lembaga Ilmu Pengetahuan Indonesia)

Yudho Giri Sucahyo, Ph.D (Universitas Indonesia)

Dana Indra Sensuse, Ph.D (Universitas Indonesia)

Dr. Eng. Iko Pramudiono (Mitsui Indonesia)

Dr. Eng. Suprapedi (Lembaga Ilmu Pengetahuan Indonesia)

Romi Satria Wahono, M.Eng, Ph.D (Universitas Dian Nuswantoro)

Contents

REGULAR PAPERS

- Pendekatan Level Data untuk Menangani Ketidakseimbangan Kelas pada Prediksi Cacat Software
Aries Saifudin and Romi Satria Wahono 76-85
- Integrasi SMOTE dan Information Gain pada Naive Bayes untuk Prediksi Cacat Software
Sukmawati Anggraini Putri and Romi Satria Wahono 86-91
- Penggunaan Random Under Sampling untuk Penanganan Ketidakseimbangan Kelas pada Prediksi Cacat Software Berbasis Neural Network
Erna Irawan and Romi Satria Wahono 92-100
- Integrasi Bagging dan Greedy Forward Selection pada Prediksi Cacat Software dengan Menggunakan Naive Bayes
Fitriyani and Romi Satria Wahono 101-108
- Komparasi Metode Machine Learning dan Metode Non Machine Learning untuk Estimasi Usaha Perangkat Lunak
Ega Kartika Adhitya and Romi Satria Wahono 109-113
- Integrasi Pareto Fitness, Multiple-Population dan Temporary Population pada Algoritma Genetika untuk Pembangkitan Data Tes pada Pengujian Perangkat Lunak
Mohammad Reza Maulana, Romi Satria Wahono and Catur Supriyanto 114-120

Pendekatan Level Data untuk Menangani Ketidakseimbangan Kelas pada Prediksi Cacat Software

Aries Saifudin¹ dan Romi Satria Wahono²

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

²Fakultas 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 & Chamathi, 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 penggunaannya, 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 (Fakhrhmad & 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 meningkatkan 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 probabilitas 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 memasang (*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 skenario 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

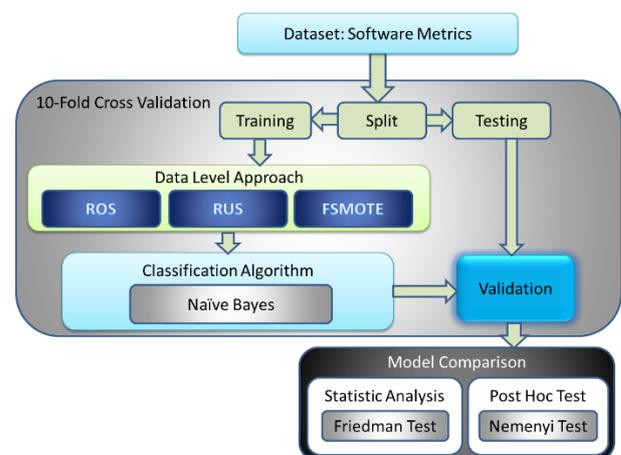
software. AUC rata-rata meningkat 25,99% untuk GA dan 20,41% untuk PSO.

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.



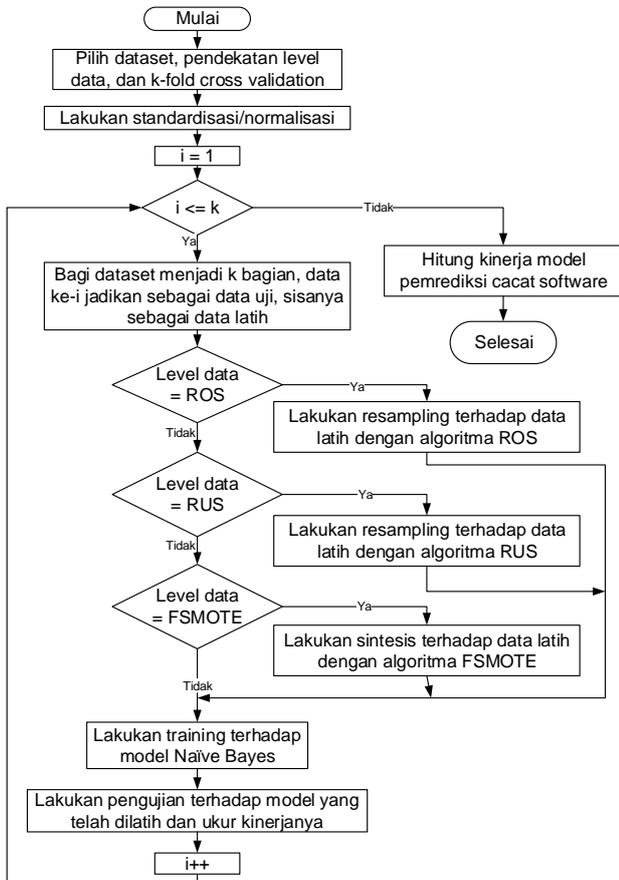
Gambar 1 Kerangka Kerja Model yang Diusulkan

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.

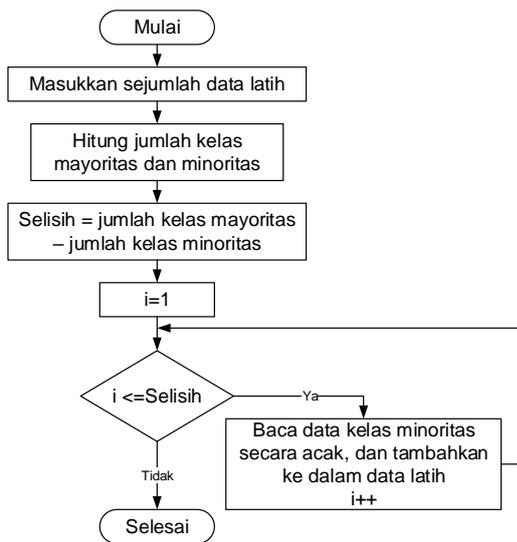
Pada algoritma ROS, data kelas minoritas dipilih secara acak, kemudian ditambahkan ke dalam data latih. Proses pemilihan dan penambahan ini diulang-ulang sampai jumlah data kelas minoritas sama dengan jumlah kelas mayoritas.

Algoritma 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.

mayoritas dihapus secara acak, sehingga jumlah kelas mayoritas sama dengan jumlah kelas minoritas.

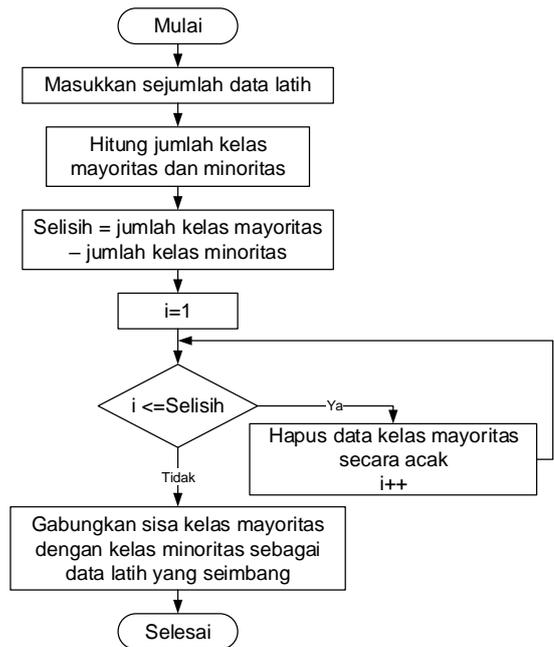


Gambar 2 Flowchart Model yang Diusulkan



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



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 narray
- 4: Sierpinski (N, i, narray, times); panggil fungsi (6) untuk mensintesis data
- 5: end for

/*Fungsi untuk mensintesis data*/

- 6: Sierpinski (N, i, narray, 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 indeks1 to times do
- 11: for attr1 to numattr 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 TP-rate/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):

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

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

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

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

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

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

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

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*.

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

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:

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

Pada persamaan di atas, nilai $q_{\alpha, \infty, K}$ 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 disajikan 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:

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

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 ($\alpha = 0,05$)
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

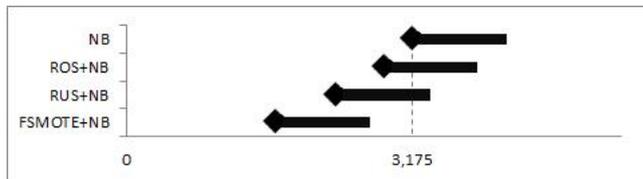
	NB	ROS+NB	RUS+NB	FSMOTE+NB
NB	1	0,85625123	0,158924874	0,001074849
ROS+NB	0,85625123	1	0,571872541	0,017325155
RUS+NB	0,158924874	0,571872541	1	0,348681121
FSMOTE+NB	0,001074849	0,017325155	0,348681121	1

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, Khoshgoftar, 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 antar model. 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 Datasets 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.
- Khoshgoftar, 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). Gijón, 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 (ICSOF'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.

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

Sukmawati Anggraini Putri

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

Romi Satria Wahono

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

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

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

1 PENDAHULUAN

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

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

kelas, yaitu rawan cacat dan bukan rawan cacat (Song, Jia, Shepperd, Ying, & Liu, 2011).

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

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

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

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

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

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

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

2 PENELITIAN TERKAIT

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

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

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

Sementara penelitian yang dilakukan oleh Gao et al (Gao & Khoshgoftaar, 2011) menyatakan bahwa algoritma seleksi atribut dibagi menjadi dua teknik, yaitu teknik *filter* dan

wrapper. Teknik *filter* seperti *chi-square*, *information gain* dan *relief*. Dan hasil pada penelitian ini menemukan bahwa *information gain* menunjukkan kinerja yang lebih baik dibandingkan dengan dua algoritma lainnya.

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

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

3 METODE USULAN

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

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

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

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

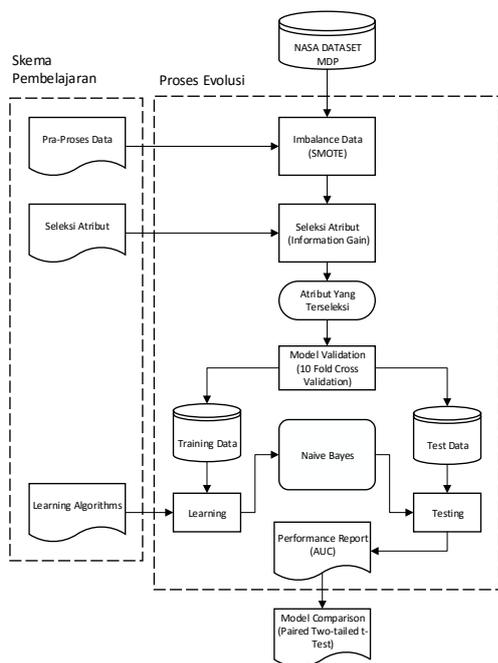
Atribut F dengan nilai $\{f_1, f_2, f_3, \dots, f_v\}$ dapat membagi data sebagai data training yang ditetapkan dalam subset $\{S_1, S_2, S_3, \dots, S_v\}$, dimana S_j merupakan subset yang memiliki nilai f_j untuk F . Selanjutnya atribut S_j berisi sampel S_{ij} kelas i . Entropy attribute F diberikan pada:

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

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

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

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



Gambar 1. Diagram Aktifitas dari Metode NB SMOTE+IG

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

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

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

- Jika A_k merupakan kategori, kemudian $P(X_k|C_j)$ merupakan jumlah tupel kelas C_j pada D mempunyai nilai X_k untuk atribut A_k , dibagi dari $|C_{1,D}|$, jumlah tupel kelas C_j pada D .
- Jika A_k merupakan nilai kontinu, maka biasanya berasumsi bahwa nilai-nilai memiliki distribusi *Gaussian* dengan *mean* (μ) dan *standard deviation* (σ), dapat didefinisikan sebagai berikut:

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

Sehingga

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

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

4 HASIL PENELITIAN

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

Table 1. NASA MDP Datasets dan Atribut

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

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

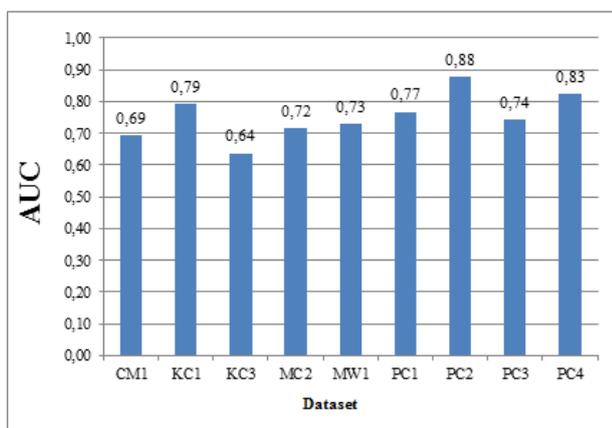
State-of-the-art menggunakan 10-fold cross-validation (Kohavi & Edu, 1995) untuk data pembelajaran dan pengujian. Ini berarti bahwa kami membagi data pelatihan menjadi 10 bagian yang sama dan kemudian dilakukan proses belajar 10 kali. Kami menerapkan 10-fold cross validation, karena metode ini telah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

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

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

Tabel 2. AUC Model NB Model Pada 9 Datasets

Klasifikasi	CM 1	KC 1	KC 3	MC 2	MW 1	PC 1	PC 2	PC 3	PC 4
Naive Bayes	0,69	0,79	0,64	0,72	0,73	0,77	0,88	0,74	0,83

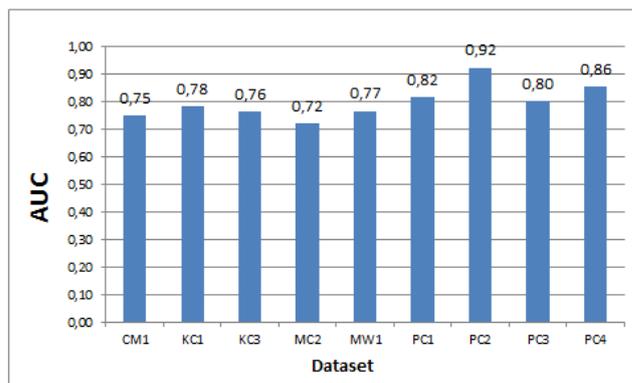


Gambar 2. AUC Model NB Model Pada 9 Datasets

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

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

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

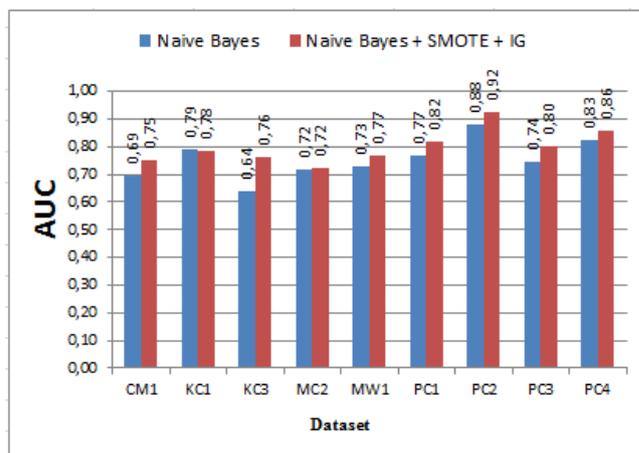


Gambar 3. AUC Model NB SMOTE+IG Pada 9 Datasets

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

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

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



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

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

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

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

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

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

Test Statistics ^b	
	NB IG+SMOTE - NB
Z	-2.490 ^b
Asymp. Sig. (2-tailed)	.013

5 KESIMPULAN

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

REFERENSI

- Catal, C. (2011). Software fault prediction: A literature review and current trends. *Expert Systems with Applications*, 38(4), 4626–4636. doi:10.1016/j.eswa.2010.10.024
- Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique, 16, 321–357.
- De Carvalho, A. B., Pozo, A., & Vergilio, S. R. (2010). A symbolic fault-prediction model based on multiobjective particle swarm optimization. *Journal of Systems and Software*, 83(5), 868–882.
- Demsar, J. (2006). Statistical Comparisons of Classifiers over Multiple Data Sets. *The Journal of Machine Learning Research*, 7, 1–30.
- Domingos, P. (1997). On the Optimality of the Simple Bayesian Classifier under Zero-One Loss. *Machine Learning*, 29(2-3), 103–130.
- Gao, K., & Khoshgoftaar, T. M. (2011). Software Defect Prediction for High-Dimensional and Class-Imbalanced Data.

- Conference: Proceedings of the 23rd International Conference on Software Engineering & Knowledge Engineering*, (2).
- Guyon, I. (2003). An Introduction to Variable and Feature Selection 1 Introduction. *Journal of Machine Learning Research*, 3, 1157–1182.
- Hall, T., Beecham, S., Bowes, D., Gray, D., & Counsell, S. (2010). A Systematic Literature Review on Fault Prediction Performance in Software Engineering. *IEEE Transactions on Knowledge and Data Engineering*, 38(6), 1276 – 1304.
- Jain, M., & Richariya, V. (2012). An Improved Techniques Based on Naive Bayesian for Attack Detection. *International Journal of Emerging Technology and Advanced Engineering*, 2(1), 324–331.
- Kabir, M., & Murase, K. (2012). Expert Systems with Applications A new hybrid ant colony optimization algorithm for feature selection. *Expert Systems With Applications*, 39(3), 3747–3763.
- Khoshgoftaar, T. M., & Gao, K. (2009). Feature Selection with Imbalanced Data for Software Defect Prediction. *2009 International Conference on Machine Learning and Applications*, 235–240.
- Kohavi, R., & Elu, S. (1995). A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI)*, 1137–1143.
- Lessmann, S., Member, S., Baesens, B., Mues, C., & Pietsch, S. (2008). Benchmarking Classification Models for Software Defect Prediction: A Proposed Framework and Novel Findings. *IEEE Transactions on Software Engineering*, 34(4), 485–496.
- Ling, C. X. (2003). Using AUC and Accuracy in Evaluating Learning Algorithms, 1–31.
- Ling, C. X., & Zhang, H. (2003). AUC: a statistically consistent and more discriminating measure than accuracy. *Proceedings of the 18th International Joint Conference on Artificial Intelligence*.
- Mccabe, T. J. (1976). A Complexity Measure. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, SE-2(4), 308–320.
- Menzies, T., Greenwald, J., & Frank, A. (2007). Data Mining Static Code Attributes to Learn Defect Predictors. *IEEE Transactions on Software Engineering*, 33(1), 2–13.
- Riquelme, J. C., Ruiz, R., & Moreno, J. (2008). Finding Defective Modules from Highly Unbalanced Datasets. *Engineering*, 2(1), 67–74.
- Rivera, J., & Meulen, R. van der. (2014). Gartner Says Worldwide IT Spending on Pace to Reach \$3.8 Trillion in 2014. Retrieved August 01, 2015, from <http://www.gartner.com/newsroom/id/2643919>
- Shepperd, M., Song, Q., Sun, Z., & Mair, C. (2013). Data Quality : Some Comments on the NASA Software Defect Data Sets. *Software Engineering, IEEE Transactions*, 39(9), 1–13.
- Song, Q., Jia, Z., Shepperd, M., Ying, S., & Liu, J. (2011). A General Software Defect-Proneness Prediction Framework. *IEEE Transactions on Software Engineering*, 37(3), 356–370.
- Turhan, B., & Bener, A. (2009). Analysis of Naive Bayes' assumptions on software fault data: An empirical study. *Data & Knowledge Engineering*, 68(2), 278–290.
- Wahono, R. S., & Suryana, N. (2013). Combining Particle Swarm Optimization based Feature Selection and Bagging Technique for Software Defect Prediction. *International Journal of Software Engineering and Its Applications*, 7(5), 153–166.
- Wang, H., Khoshgoftaar, T. M., Gao, K., & Seliya, N. (2009). High-Dimensional Software Engineering Data and Feature Selection. *Proceedings of 21st IEEE International Conference on Tools with Artificial Intelligence*, Nov. 2-5, 83–90.
- Wilcoxon, F. (1945). Individual Comparisons by Ranking Methods. *International Biometric Society Stable*, 1(6), 80–83.
- Yap, B. W., Rani, K. A., Aryani, H., Rahman, A., Fong, S., Khairudin, Z., & Abdullah, N. N. (2014). An Application of Oversampling, Undersampling, Bagging and Boosting in Handling Imbalanced Datasets. *Proceedings of the First International Conference on Advanced Data and Information Engineering (DaEng-2013)*, 285, 13–23.

BIOGRAFI PENULIS



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



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

Penggunaan *Random Under Sampling* untuk Penanganan Ketidakseimbangan Kelas pada Prediksi Cacat Software Berbasis *Neural Network*

Erna Irawan

Sekolah Tinggi Manajemen Informatika Komputer Nusa Mandiri
stnaira@gmail.com

Romi Satria Wahono

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

Abstrak: Penurunan kualitas *software* dan biaya perbaikan yang tinggi dapat diakibatkan kesalahan atau cacat pada *software*. Prediksi cacat *software* sangat penting di dalam *software engineering*, terutama dalam mengatasi masalah efektifitas dan efisiensi sehingga dapat meningkatkan kualitas *software*. *Neural Network* (NN) merupakan algoritma klasifikasi yang telah terbukti mampu mengatasi masalah data nonlinear dan memiliki sensitifitas yang tinggi terhadap suatu data serta mampu menganalisa data yang besar. Dataset NASA MDP merupakan data metric yang nonlinear perangkat lunak yang biasa digunakan untuk penelitian *software defect prediction* (prediksi cacat *software*). Terdapat 62 penelitian dari 208 penelitian menggunakan dataset NASA. NASA MDP memiliki kelemahan yaitu kelas yang tidak seimbang sehingga dapat menurunkan kinerja dari model prediksi cacat *software*. Untuk menangani ketidakseimbangan kelas dalam dataset NASA MDP adalah dengan menggunakan metode level data yaitu *Random Under Sampling* (RUS). RUS ditujukan untuk memperbaiki ketidakseimbangan kelas. Metode yang diusulkan untuk menangani ketidakseimbangan kelas pada *Neural Network* (NN) adalah penerapan RUS. Eksperimen yang diusulkan untuk membandingkan hasil kinerja *Neural Network* sebelum dan sesudah diterapkan metode RUS, serta dibandingkan dengan model yang lainnya. Hasil Eksperimen rata-rata AUC pada NN (0.80) dan NN+RUS (0.82). Hasil uji *Wilcoxon* dan *Friedman* menunjukkan bahwa bahwa AUC NN+RUS memiliki perbedaan yang signifikan dengan NN dengan *p-value wilcoxon* = 0.002 dan *p-value friedman* = 0.003 ($p < 0.05$). Menurut uji *friedman* terdapat perbedaan AUC yang signifikan antara NN+RUS dengan NN, NN+SMOTE, NB, dan C45 karena nilai *p-value* < 0.0001. Maka dapat disimpulkan bahwa penerapan model RUS terbukti dapat menangani masalah ketidakseimbangan kelas pada prediksi cacat *software* berbasis *neural network*.

Kata Kunci: Ketidakseimbangan Kelas, Neural Network, Random Under Sampling

1 PENDAHULUAN

Pembelian *software* di seluruh dunia pada tahun 2013 \$3,7 triliun, dengan 23% dari biaya ini untuk jaminan kualitas dan pengujian (Lovelock, 2014). Penurunan kualitas *software* dan biaya perbaikan yang tinggi dapat diakibatkan kesalahan atau cacat pada *software*. Kualitas *software* yang rendah dapat mengakibatkan kemungkinan pembatalan proyek, tuntutan pidana jika *software* menyebabkan kematian, kerugian bagi

pelanggan, dan biaya pemeliharaan yang tinggi. Pencarian dan perbaikan cacat *software* membutuhkan biaya yang paling besar pada saat pengembangan *software* dibandingkan dengan biaya lainnya (Jones & Bonsignour, 2012). Biaya untuk perbaikan cacat *software* sangat besar, yaitu biaya untuk memperbaiki cacat akibat salah permintaan setelah fase pengembangan 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 (Jones & Bonsignour, 2012). Hal ini menunjukkan sangat dibutuhkannya prediksi cacat *software*.

Prediksi cacat *software* sangat penting di dalam *software engineering*, terutama dalam mengatasi masalah efektifitas dan efisiensi di luar pengujian perangkat lunak dan review, sehingga ketepatan prediksi cacat *software* dapat mempermudah pengujian, mengurangi biaya, dan meningkatkan kualitas *software* (Wahono, Suryana, & Ahmad, 2014). Selain itu prediksi cacat *software* memungkinkan pengembang *software* mengalokasikan sumber daya yang terbatas sesuai anggaran dan dengan waktu yang efisien (Zheng, 2010). Saat ini penelitian prediksi cacat *software* fokus kepada tiga topik yaitu, estimasi jumlah cacat *software*, asosiasi cacat *software*, dan mengklasifikasikan kerawanan cacat dari komponen *software* (Song, Jia, Shepperd, & Liu, 2011). Para pengembang dapat menghindari resiko yang mungkin terjadi akibat cacat *software* dengan memakai prediksi cacat *software*.

Dataset NASA MDP merupakan data metric perangkat lunak yang biasa digunakan untuk penelitian *software defect prediction* (prediksi cacat *software*). Terdapat 62 penelitian dari 208 penelitian menggunakan dataset NASA (Hall, Beecham, Bowes, Gray, & Counsell, 2011). Dataset NASA merupakan data nonlinear yang mudah diperoleh dan kinerja dari metode yang digunakan menjadi lebih mudah dibandingkan dengan penelitian sebelumnya (Arisholm, Briand, & Johannessen, 2010). Pada dataset cacat *software* ditemukan dataset yang tidak seimbang (*imbalanced class*) karena dataset yang tidak cacat (*nonfault-prone*) lebih banyak daripada cacat (*fault-prone*) Karena dataset cacat *software* merupakan kelas minoritas maka banyak cacat yang tidak dapat ditemukan (Saifudin & Wahono, 2015).

Terdapat beberapa algoritma klasifikasi untuk prediksi cacat *software*. Metode Naïve Bayes adalah salah satu algoritma yang sering digunakan karena mudah digunakan untuk data yang besar (Wu & Kumar, 2010) dan merupakan

salah satu algoritma klasifikasi yang simple untuk prediksi cacat *software* (Hall, Beecham, Bowes, Gray, & Counsell, 2012). Namun Naïve Bayes berasumsi bahwa atribut dataset sama penting dan tidak terkait satu sama lain (Yap et al., 2014). Sedangkan Logistic Regression memiliki kelebihan dalam kemudahan dalam pengaplikasiannya namun memiliki kelemahan dalam *underfitting* dan akurasi yang rendah dan tidak mampu menangani hubungan kompleks dalam dataset yang besar (Harrington, 2012) (Setiyorini et al., 2014). Selain itu terdapat *Decision Tree* merupakan metode klasifikasi dan prediksi yang kuat namun memiliki kelemahan yaitu terjadi overlap terutama ketika kelas-kelas dan atribut yang digunakan jumlahnya sangat banyak. Kemudian *Neural Network*, *Neural Network* telah terbukti mampu mengatasi masalah data nonlinear dan memiliki sensitifitas yang tinggi terhadap suatu data serta mampu menganalisa data yang besar dan kompleks (Zheng, 2010) (Zamani & Amaliah, 2012). *Neural Network* memiliki keunggulan dalam memprediksi hasil keputusan diagnostik dan hubungan yang kompleks bersifat nonlinear antara faktor dan hasil prediksi (Chen, Zhang, Xu, Chen, & Zhang, 2012).

Secara umum terdapat dua pendekatan untuk menangani dataset yang tidak seimbang (*imbalance class*), yaitu pendekatan level algoritma dan level data (Z.-Z. Zhang et al., 2008). Pada level algoritma, metode utamanya adalah menyesuaikan operasi algoritma yang ada untuk membuat pengklasifikasian agar lebih kondusif terhadap klasifikasi kelas minoritas (D. Zhang, Liu, Gong, & Jin, 2011). Kelemahan level algoritma jika diaplikasikan dalam algoritma klasifikasi kuat seperti *Neural Network* adalah waktu yang lebih lama karena adanya penyesuaian bobot dan iterasi sampai mendapat nilai yang sesuai (Korada, Kumar, & Deekshitulu, 2012). Pada level data terdapat berbagai teknik resampling yang digunakan untuk memperbaiki ketidakseimbangan kelas. Tiga teknik yang biasa digunakan adalah Random Over Sampling (ROS) dan Random Under Sampling (RUS) dan SMOTE. (Khoshgoftaar, Gao, Napolitano, & Wald, 2013). ROS meningkatkan ukuran kelas minoritas dengan mensintesis sampel baru atau langsung mereplikasi secara acak dataset training (Yu et al., 2013). SMOTE merupakan pendekatan oversampling yang menciptakan sample sintetik (Chawla, Lazarevic, & Lawrence). SMOTE dan ROS meningkatkan jumlah kelas mayoritas sehingga meningkatkan waktu prediksi. RUS memiliki waktu prediksi yang lebih cepat dibandingkan ROS dan SMOTE (Park, Oh, & Pedrycz, 2013). Pengaplikasian RUS pada algoritma prediksi cacat *software* bisa mengurangi pengeluaran biaya (Arar & Ayan, 2015). Metode RUS lebih mudah dan cepat diaplikasikan dibandingkan metode yang lainnya (Yu et al., 2013).

Pada penelitian ini, mengusulkan penggunaan level data yaitu *Random Under Sampling* untuk penanganan ketidakseimbangan kelas pada prediksi cacat *software* berbasis *Neural Network*. Tujuan dari penelitian ini adalah menerapkan metode *Random Under Sampling* untuk menangani ketidakseimbangan kelas (*class imbalance*) pada *Neural Network* agar dapat meningkatkan kinerja prediksi cacat *software*.

2 PENELITIAN TERKAIT

Penelitian yang dilakukan oleh Zheng (Zheng, 2010) melakukan penelitian untuk mengurangi biaya dengan meningkatkan kinerja *Neural Network* untuk prediksi cacat *software* dengan boosting. *Software* yang berkualitas tinggi dapat diproduksi pada waktu sesuai anggaran. Dataset yang digunakan adalah empat dataset dari NASA. Setiap dataset dipilih berdasarkan *five-fold x validation*, satu bagian menjadi

data *testing* dan bagian yang lain menjadi data *training*. Atribut yang digunakan adalah 20 atribut pada setiap dataset. Dataset tersebut diolah menggunakan *Neural Network* dan *Boosting*. *Boosting* menyebabkan iterasi sebanyak 20 kali. Output layer didapat dari jumlah kelas. Hasil penelitian ini menunjukkan bahwa perbaikan dengan *boosting* lebih baik dalam efisiensi algoritma *Neural Network*. Model yang dilakukan (Zheng, 2010).

Penelitian yang dilakukan oleh Yu, Tang, Shen, Yang, dan Yang, J. (Yu et al., 2013), melakukan penelitian untuk meningkatkan akurasi prediksi cacat *software* mengkombinasikan SVM dengan cara modifikasi Adaboost dengan random under sampling (RUS). SVM memiliki kelemahan didalam mengukur kelas minoritas dan penentuan keputusan final dataset training, RUS memiliki efektifitas yang tinggi untuk mengatasi masalah pada kelas mayoritas dan minoritas pada SVM sedangkan Dataset yang digunakan adalah TargetATP. Teknik umum yang digunakan adalah *10-fold cross-validation* untuk pembagian dataset. Satu bagian dijadikan data *testing* dan sisanya menjadi data *training*. Pertama dataset dilakukan pemilihan fitur dengan logistic PSSM dan PSS kemudian diproses menggunakan SVM dan RUS. Hasilnya menunjukkan AUC terendah 0.75 dan terbesarnya 0.86.

Penelitian yang dilakukan oleh Wahono, Suryana dan Ahmad (Wahono, Herman, et al., 2014) mengatakan bahwa penelitian prediksi cacat *software* telah menjadi topik yang penting dalam bidang *software engineering*. Kinerja model prediksi cacat *software* berkurang secara signifikan, dikarenakan dataset yang digunakan mengandung ketidakseimbangan kelas (*imbalance class*). Seleksi fitur digunakan dalam *machine learning* ketika melibatkan dataset berdimensi tinggi dan atribut yang masih mengandung noise. Penelitian ini menerapkan optimasi metaheuristik untuk menemukan solusi optimal dalam seleksi fitur, secara signifikan mampu mencari solusi berkualitas tinggi dengan jangka waktu yang wajar. Pada penelitian ini, diusulkan kombinasi metode optimasi metaheuristik dan teknik bagging untuk meningkatkan kinerja prediksi cacat *software*. Metode optimasi metaheuristik (algoritma genetikan dan particle swarm optimizaion) diterapkan untuk menangani pemilihan fitur, dan teknik bagging digunakan untuk menangani ketidakseimbangan kelas.

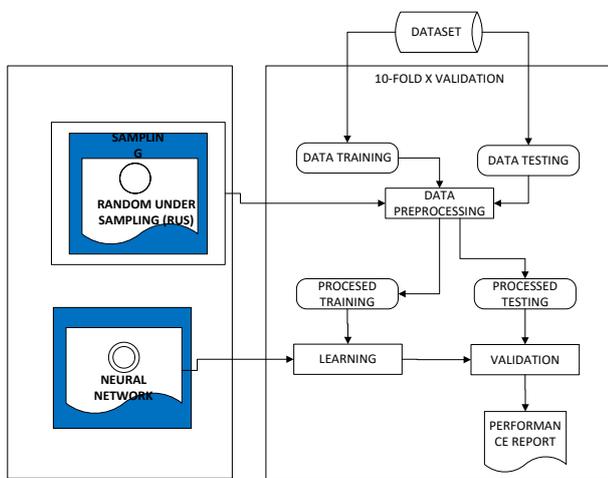
Penelitian ini menggunakan 9 NASA MDP dataset dan 10 algoritma pengklasifikasian yang dikelompokkan dalam 5 tipe. Yaitu pengklasifikasian statistik tradisional (Logistic Regression (LR), Linear Discriminant Analysis (LDA), dan Naïve Bays (NB)), Nearest Neighbors (k- Nearest Neighbor (k-NN) dan K*), *Neural Network* (Back Propagation (BP)), Support Vector Machine (SVM), dan Decision Tree (C45, Classification an Regression Tree (CART), dan Random Forest (RF)). Hasilnya menunjukkan bahwa metode yang diusulkan dapat memberikan peningkatan yang mengesankan hasil perbandingan, disimpulkan bahwa tidak ada perbedaan yang signifikan antara optimasi PSO dan Algoritma Genetika ketika digunakan sebagai seleksi fitur untuk sebagian besar pengklasifikasian pada model prediksi cacat *software*.

Penelitian yang dilakukan oleh Arar, dan Ayan (Arar & Ayan, 2015), untuk prediksi cacat *software* mengkombinasikan *Neural Network* (NN) dan novel *Artificial Bee Colony* (ABC) algoritma yang digunakan dalam penelitian ini. Aplikasi yang digunakan adalah WEKA. Teknik umum yang digunakan adalah *10-fold cross-validation* untuk pembagian dataset. Satu bagian dijadikan data *training* dan sisanya menjadi data *testing*. ABC digunakan untuk optimalisasi bobot sehingga kinerja *Neural Network* diharapkan meningkat. *The False*

Positive Rate (FPR) dan False Negatif Rate (FNR) dikalikan parametrik koefisien. Dataset yang digunakan adalah NASA MDP spacecraft. Dataset NASA dilakukan 10 kali *foldcross-validation*. Neural Network (NN) dengan optimasi novel Artificial Bee Colony (ABC) menghasilkan efisiensi biaya lebih baik dibandingkan Neural Network (NN).

3 METODE YANG DIUSULKAN

Metode yang diusulkan pada penelitian ini yaitu untuk meningkatkan kinerja *Neural Network back propagation* dengan metode *Random Under Sampling* untuk menangani ketidakseimbangan kelas (*class imbalance*) pada prediksi cacat *software*. Selanjutnya untuk validasi menggunakan *10-fold cross validation*. Hasil pengukuran kinerja algoritma dengan menggunakan uji *Wilcoxon* untuk mengetahui perbedaan kinerja model setelah dan sebelum diterapkan model resampling. Selanjutnya juga dilakukan pengujian kinerja model algoritam *Neural Network* dengan algoritma pengklasifikasi lain menggunakan uji *Freidmen (Freidmen test)*. Model kerangka pemikiran metode yang diusulkan ditunjukkan pada Gambar 1



Gambar 1. Kerangka Pemikiran Model yang Diusulkan

Neural Network (NN) atau jaringan syaraf tiruan dibuat oleh Warren McCulloch dan Walter Pitts (1943) dan dianggap sebagai basis Neural Network modern saat ini (Gorunescu, 2011). Neural Network (NN) telah digunakan dibanyak pengenalan pola aplikasi (Zheng, 2010). Neural Network adalah model penalaran berdasarkan otak manusia Yao (1993) dalam (Zheng, 2010). Neural Network (NN) terdiri dari sebuah set pengolahan informasi unit kunci. Network ini terdiri dari tiga lapisan yaitu input (minimal satu), tersembunyi (*hidden*) dan output.

Pada penelitian ini menggunakan algoritma *backpropagation*. Algoritma *backpropagation* bekerja melalui proses secara iteratif menggunakan data *training*, membandingkan nilai prediksi dari jaringan dengan setiap data yang terdapat pada data *training* (Han et al., 2012). Langkah pembelajaran dalam algoritma *backpropagation* adalah sebagai berikut Kahraman, Bayindir, & Sagioglu (2012).

1. Menginisialisasi bobot jaringan secara acak (biasanya antara -0.1 sampai 1.0)
2. Menghitung input untuk simpul berdasarkan nilai input dan bobot jaringan tersebut untuk setiap data training, berdasarkan rumus:

$$\text{Input } j = \sum_{i=1}^n O_i W_{ij} + \theta_j$$

Keterangan:

- O_i = Output simpul I dari layer sebelumnya
 - W_{ij} = Bobot relasi dari simpul I pada layer sebelumnya ke simpul j
 - θ_j = Bias (sebagai pembatas)
3. Berdasarkan langkah sebelumnya kemudian bangkitkan output untuk simpul menggunakan fungsi aktivasi sigmoid :

$$F = \frac{1}{1 + e^{-\text{input}}}$$

4. Menghitung nilai *error* antara nilai sesungguhnya dengan nilai prediksi menggunakan rumus:

$$\text{Error}_j = \text{Output}_j \cdot (1 - \text{Output}_j) \cdot (\text{Target}_j - \text{Output}_j)$$

Keterangan:

- Output_j = Output aktual dari simpul j
- Target_j = Nilai target yang sudah diketahui pada data training

5. Setelah nilai *error* dihitung, selanjutnya dibalik ke layer sebelumnya (*backpropagated*). Untuk menghitung nilai *error* pada hidden layer, menggunakan rumus

$$\text{Error}_j = \text{Output}_j \cdot (1 - \text{Output}_j) \cdot \sum_{k=1}^n \text{Error}_k W_{jk}$$

Keterangan :

- Output_j = Output aktual dari simpul j
- Error_k = Error Simpul k
- W_{jk} = Bobot relasi dari simpul j ke simpul k pada layer berikutnya

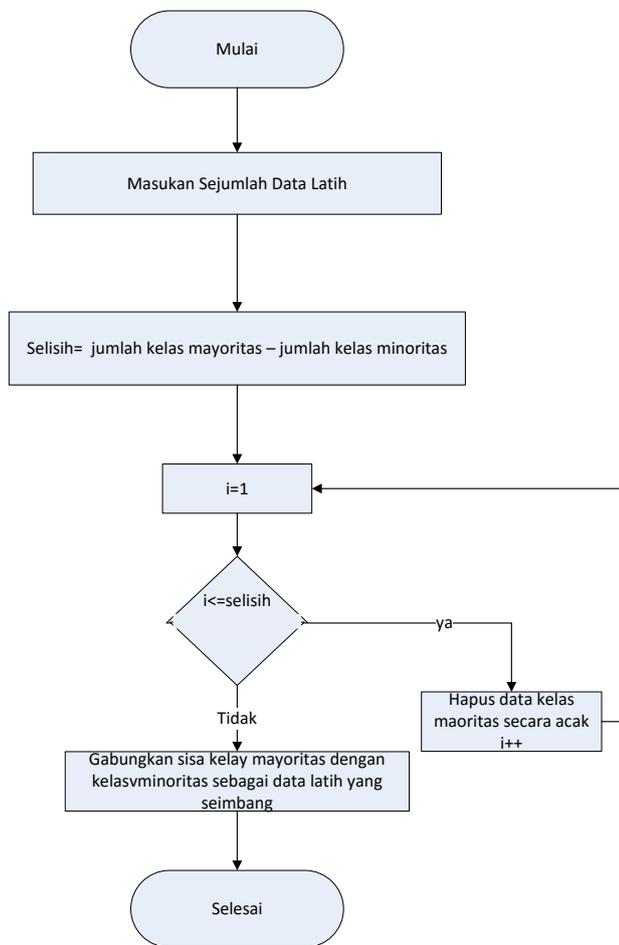
6. Setelah nilai *error* dihitung kemudian bobot relasi diperbahari menggunakan rumus

$$W_{ij} = W_{ij} + I \cdot \text{Error}_j \cdot \text{Output}_i$$

Keterangan :

- W_{ij} = Bobot relasi dari unit I pada layer sebelumnya ke unit j
- I = Learning rate
- Error_j = Error pada output layer simpul j
- Output_i = Output dari simpul i

Random under sampling (RUS) menghitung selisih antara kelas mayoritas dan minoritas kemudian dilakukan perulangan selisih hasil perhitungan, selama perulangan data kelas mayoritas dihapus secara acak, sehingga jumlah kelas mayoritas sama dengan minoritas (Saifudin & Wahono, 2015). Langkah pertama pada *Random Under Sampling* adalah pemilihan dataset kemudian dihitung selisih antara kelas mayoritas dan minoritas, jika masih terdapat selisih antara jumlah kelas maka dataset kelas mayoritas akan dihapus secara acak sampai jumlah kelas mayoritas sama dengan kelas minoritas. RUS dapat lebih efektif dan cepat dalam proses pelatihan prediksi imbalance class sebuah cacat *software*. Flowchart RUS terlihat pada Gambar 1.



Gambar 1 Flowchart Algoritma *Random Under Sampling* (RUS)

Dalam penelitian ini menggunakan data sekunder yaitu NASA (*National Aeronautics and Space Administration*) MDP (*Metrics Data Programs*) repository sebagai *software matrices* yang merupakan dataset yang sudah umum digunakan para peneliti dalam membangun model kecacatan *software* (Hall et al., 2012). Dataset NASA terdapat pada repository MDP dan PROMISE. Peneliti memilih dataset NASA MDP karena sudah diperbaiki sebelumnya dengan menghilangkan data null oleh Martin Shepperd (Liebchen & Shepperd, 2008). Dalam penelitian ini menggunakan dataset yang sudah diperbaiki yaitu CM1, KC1, KC3, MC2, MW1, PC1, PC2, PC3, PC4 dan PC5 dengan spesifikasi yang ditujukan pada Tabel 1.

Pengolahan dataset awal yaitu melakukan normalisasi data. Normalisasi data dilakukan sesuai fungsi aktivasi yang digunakan, dalam penelitian ini digunakan fungsi *binary sigmoid*, data harus dinormalisasikan dalam *range* 0 sampai 1 (Jong Jek Siang, 2009). Maka, pada data sinoptik yang ada dilakukan *transform* data dengan rumus sebagai berikut:

$$X^i = \frac{0.8(x-a)}{b-a} + 0.1$$

Keterangan : X^i = nilai transform
 X = nilai asli
 a = nilai minimal
 b = nilai maksimal

Proses pengujian metode dimulai dari pembagian dataset dengan metode *10-fold cross validation* yaitu membagi dataset menjadi dua segmen, segmen pertama digunakan sebagai data

training dan segmen kedua digunakan sebagai data testing untuk mevalidasi model (Witten, Frank, & Hall, 2011). Selanjutnya diterapkan tahapan evaluasi menggunakan *Area Under Curve (AUC)* untuk mengukur hasil akurasi indikator dari performa model prediksi.

Tabel 1 Dataset NASA MDP Repository

Nama Atribut	CM1	KC1	KC3	MC2	MW1	PC1	PC2	PC3	PC4	PC5
LOC_BLANK	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
LOC_CODE_AND_COMMENT	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
LOC_COMMENTS	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
LOC_EXECUTABLE	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
LOC_TOTAL	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
NUMBER_OF_LINES	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
HALSTEAD_CONTENT	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
HALSTEAD_DIFFICULTY	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
HALSTEAD_EFFORT	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
HALSTEAD_ERROR_EST	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
HALSTEAD_LENGTH	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
HALSTEAD_LEVEL	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
HALSTEAD_PROG_TIME	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
HALSTEAD_VOLUME	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
NUM_OPERANDS	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
NUM_OPERATORS	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
NUM_UNIQUE_OPERANDS	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
NUM_UNIQUE_OPERATORS	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
CYCOMATIC_COMPLEXITY	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
CYCOMATIC_DENSITY	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
DESIGN_COMPLEXITY	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
ESSENTIAL_COMPLEXITY	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BRANCH_COUNT	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
CALL_PAIRS	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
CONDITION_COUNT	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
DECISION_COUNT	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
DECISION_DENSITY	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
DESIGN_DENSITY	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
EDGE_COUNT	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
ESSENTIAL_DENSITY	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
GLOBAL_DATA_COMPLEXITY			✓	✓						✓
GLOBAL_DATA_DENSITY			✓	✓						✓
MAINTENANCE_SEVERITY	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MODIFIED_CONDITION_COUNT	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MULTIPLE_CONDITION_COUNT	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
NODE_COUNT	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
NORMALIZED_CYCOMATIC_COMPLEXITY	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
PARAMETER_COUNT	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
PERCENT_COMMENTS	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
PATHOLOGICAL_COMPLEXITY										
Defective	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Jumlah attribute	38	22	40	40	37	38	37	38	38	39
Jumlah Modul	3	116	14	12	264	679	722	105	127	169
	7	2	4	4				3	0	4
Jumlah modul yang cacat	42	29	36	44	27	55	14	13	17	45
	4							0	6	8
Presentase modul yang cacat	12,84%	25,30%	18,56%	35,48%	10,23%	8,10%	2,22%	12,35%	13,86%	27,04%
Jumlah modul yang tidak cacat	2	86	14	80	237	624	706	92	109	123
	5	8	8					3	4	6

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*, *GMean*, 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 terkait adalah sebagai berikut:

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

$$\text{Precision /PPV} = \frac{TP}{TP + FP}$$

$$\text{NPV} = \frac{TN}{TN + FN}$$

$$\text{Sensitivitas/ recall/} = \frac{TP}{TP + FN}$$

$$\text{Specificity / TN}_{\text{rate}} = \frac{TN}{TN + FP}$$

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

$$FN_{rate} = \frac{FN}{FN + TP}$$

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

$$G\text{-Mean} = \sqrt{sensitivitas * specificity}$$

Cara untuk menentukan model mana yang memiliki kinerja terbaik, maka dibutuhkan satu ukuran yang mewakili kinerja dari setiap model. *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). *Area Under Curve* (AUC) digunakan untuk mengukur hasil kinerja model prediksi dapat dilihat. Hasilnya dapat dilihat dari hasil *confusion matrix* secara manual dengan perbandingan klasifikasi menggunakan curva *Receiver Operating Characteristic* (ROC). ROC menghasilkan dua garis dengan bentuk *false positives* sebagai garis horisontal *true positives* sebagai garis vertical (Vercellis, 2011). Grafik antara sensitivitas (*true positive rate*), pada sumbu Y dengan 1-spesifitas pada sumbu X (*false positive rate*) disebut kurva ROC. Tarik-menarik antara sumbu Y dengan sumbu X digambarkan oleh curva ROC (Saifudin & Wahono, 2015).

$$\text{Rumus Area Under Curve (AUC)} = \frac{1 + TP_{rate} - FP_{rate}}{2}$$

AUC dihitung berdasarkan rata-rata perkiraan bidang bentuk trapesium untuk kurva yang dibentuk oleh TP_{rate} dan FP_{rate} (Park et al., 2013). Dalam pengklasifikasian keakuratan dari test diganostik menggunakan Area Under Curve (AUC) (Gorunescu, 2011) dapat dilihat melalui Tabel 2.

Tabel 2 Nilai AUC, Keterangan dan Diagnostik

Nilai AUC	Klasifikasi	Simbol
0.90 – 1.00	<i>Excellent classification</i>	↑
0.80 – 0.90	<i>Good classification</i>	↗
0.70 – 0.80	<i>Fair classification</i>	→
0.60 – 0.70	<i>Poor classification</i>	↘
< 0.60	<i>Failure</i>	↓

Menurut Demsar (2006) pada uji perbandingan dua klasifikasi pada beberapa dataset terhadap dua uji bisa menggunakan dataset parametric atau nonparametric. Dalam metode pembelajaran machine learning sebaiknya memakai nonparametric karena jumlah atribut pada dataset yang berbeda dan penyebaran data yang cenderung tidak normal (Demsar, 2006). Uji Wilcoxon Signed-Ranked merupakan uji jji statistic non parametrik, alternatif dari uji t berpasangan (Demsar, 2006) untuk sampel yang tidak terdistribusi normal. Dengan kata lain uji peringkat bertanda Wilcoxon merupakan uji statistik non parametrik yang digunakan untuk dua sampel berpasangan dengan sampel yang tidak terdistribusi normal.

$$R+ = \sum di > 0 \text{ rank } (di) + 1/2 \sum di = 0 \text{ rank } (di)$$

$$R- = \sum di < 0 \text{ rank } (di) + 1/2 \sum di = 0 \text{ rank } (di)$$

Untuk dataset lebih dari 25 maka uji statistiknya

$$z = \frac{T - 1/4N(N+1)}{\sqrt{1/24N(N+1)(2N+1)}}$$

Perbandingan statistik dari pengklasifikasi yang baik adalah menggunakan tes non parametrik yang terdiri dari uji *Wilcoxon signed ranked* untuk perbandingan dari dua pengklasifikasi dan uji Friedman dengan uji *Post Hoc* yang sesuai untuk perbandingan lebih dari satu pengklasifikasi dengan beberapa dataset (Demsar, 2006).

4.HASIL EKSPERIMEN

Hasil eksperimen *Neural Network* yang dilakukan dengan menggunakan 10 dataset NASA MDP (CM1, KC1, KC3, MC2, MW1, PC1, PC2, PC3, PC4 dan PC5). Model yang diuji adalah model *Neural Network* (NN) dan *Neural Network* dengan RUS (NN+RUS). Hasil pengukuran model prediksi *cacat software* dengan *Neural Network* dapat dilihat pada Tabel 3 dan *Neural Network* dengan RUS (NN+RUS) pada Tabel 4

Tabel 3 Hasil Eksperimen Neural Network

Datase t	Accur acy	Recall	Specifi city	PPV	NPV	F- Measu re	G- Mean	AUC
CM1	84.1%	42.9%	91.7%	47.7%	89.8%	0.467	0.62	0.78
KC1	82.4%	38.4%	90.7%	44%	88.6%	0.41	0.59	0.76
KC3	83%	47.9%	93.3%	67.7%	85.9%	0.56	0.66	0.82
MC2	78.9%	46.2%	98.6%	96%	72.0%	0.62	0.67	0.84
MW1	71.5%	74.3%	71.1%	28.3%	94.7%	0.4	0.72	0.84
PC1	89.7%	43.5%	94.4%	44.8%	94.2%	0.44	0.64	0.8
PC2	89.2%	41.7%	90.8%	13.5%	97.8%	0.2	0.61	0.77
PC3	37.5%	91.3%	29.4%	16.3%	95.7%	0.2	0.51	0.71
PC4	85.8%	37.6%	93.3%	46.7%	90.6%	0.41	0.59	0.77
PC5	96.6%	45.3%	98.2%	43.9%	98.3%	0.44	0.66	0.83

Tabel 4 Hasil Eksperimen Model NN+RUS

Datase t	Accur acy	Recall	Specifi city	PPV	NPV	F- Measu re	G- Mean	AU C
CM1	82.3%	46%	88.8%	42.6%	90.1%	0.44	0.64	0.79
KC1	82.4%	40%	90.5%	44.2%	89.9%	0.43	0.42	0.77
KC3	82%	47.7%	92%	63.6%	85.7%	0.54	0.66	0.82
MC2	79.2%	51.9%	98.6%	96.4%	74.2%	0.68	0.72	0.87
MW1	52.1%	100.9%	44.7%	21.7%	100.9%	0.35	0.67	0.84
PC1	87.8%	44.9%	92.2%	37.3%	94.2%	0.41	0.65	0.81
PC2	69.4%	75%	69.2%	7.73%	98.8%	0.2	0.73	0.84
PC3	61%	73%	58.5%	20.8%	93.4%	0.33	0.66	0.77
PC4	84.6%	41.4%	91.3%	42.5%	90.9%	0.43	0.62	0.78
PC5	96.0%	54.7%	97.3%	38.8%	98.6%	0.46	0.74	0.88

Untuk lebih jelasnya perbandingan perbandingan Akurasi, Recall, F-Measure, G-Mean dan AUC sebagai berikut:

Tabel 5. Hasil Pengukuran Akurasi Model Prediksi Cacat Software

Dataset	NN	NN+RUS
CM1	84%	82.3%
KC1	82.4%	82.4%
KC3	83%	82%
MC2	78%	79.2%
MW1	71%	52%
PC1	89%	88%
PC2	89%	70%
PC3	37.5%	61%
PC4	85.8%	86%
PC5	96%	96.0%

Berdasarkan Tabel 5 perbandingan sensitivitas NN dan NN+RUS menunjukkan nilai akurasi terbaik terdapat pada dataset PC5 dengan nilai yang sama antara *Neural Network* dan *Neural Network* dengan *Random Under Sampling* yaitu 96% sedangkan nilai akurasi terendah terdapat pada dataset PC3 dengan model *Neural Network* yaitu 37.5%.

Tabel 6 Hasil Pengukuran Sensitivitas Model Prediksi Cacat Software

Dataset	NN	NN+RUS
CM1	42.00%	46%
KC1	38.4%	40%
KC3	47.70%	47.7%
MC2	46.2%	51.92%
MW1	74.3%	100.%
PC1	43.5%	44.9%
PC2	41.7%	75%
PC3	91.3%	73%
PC4	37.6%	41.40%
PC5	45.3%	54.7%

Berdasarkan Tabel 6 nilai sensitivitas terbaik adalah *Neural Network* dengan *Random Under Sampling* dimana terdapat delapan dataset (CM1, KC1, MC2, MW1, PC1, PC2, PC4, dan PC5) memiliki nilai lebih tinggi dibandingkan model *Neural Network*. Sedangkan satu dataset (KC3) memiliki nilai yang sama antara kedua model dan satu dataset (PC3) menundukkan nilai sensitivitas yang terbaik pada model *Neural Network*. Nilai tertinggi terdapat pada dataset MW1 yaitu 100% pada model *Neural Network* dengan *Random Under Sampling* sedangkan dataset yang memiliki nilai sensitivitas terendah adalah KC1 yaitu 38.4% pada model *Neural Network*.

Tabel 7 Hasil Pengukuran F-Measure Model Prediksi Cacat Software

Dataset	NN	NN+RUS
CM1	0.467	0.44
KC1	0.41	0.43
KC3	0.56	0.54
MC2	0.62	0.68
MW1	0.4	0.35
PC1	0.44	0.41
PC2	0.2	0.2
PC3	0.2	0.33
PC4	0.41	0.43
PC5	0.44	0.46

Berdasarkan Tabel 7 menunjukkan hasil *f-measure* yang baik pada penggunaan *Neural Network* dengan *Random Under Sampling* dimana terdapat lima dataset (KC1, MC2, PC3, PC4, dan PC5) yang menunjukkan nilai *f-measure* yang terbaik. Satu dataset (PC2) memiliki nilai yang sama pada kedua model. Sedangkan empat dataset yang tersisa (CM1, KC3, MW1 dan PC1) menunjukkan model *Neural Network* memiliki nilai *f-measure* tertinggi. Nilai *f-measure* yang terbesar terdapat pada dataset MC2 (0.68) pada model *Neural Network* dengan *Random Under Sampling* sedangkan yang terkecil terdapat pada dataset PC2 untuk kedua model dan PC3 untuk model *Neural Network*.

Tabel 8 Hasil Pengukuran G-Mean Model Prediksi Cacat Software

Dataset	NN	NN+RUS
CM1	0.62	0.64
KC1	0.59	0.42
KC3	0.66	0.66
MC2	0.67	0.72
MW1	0.72	0.67
PC1	0.64	0.65
PC2	0.61	0.73
PC3	0.51	0.66
PC4	0.59	0.62
PC5	0.66	0.74

Berdasarkan Tabel 8 didapatkan hasil nilai *G-Mean* yang terbaik pada model *Neural Network* dengan *Random Under Sampling*, dimana terdapat tujuh dataset (CM1, MC2, PC1, PC2, PC3, PC4, dan PC5). Satu dataset (KC3) memiliki *G-Mean* yang sama pada kedua model sedangkan dua dataset lainnya (KC1 dan MW1) menunjukkan nilai tertinggi pada model *Neural Network*. Nilai *G-Mean* tertinggi terdapat pada dataset PC2 (0.73) dan yang terendah pada dataset KC1 (0.42) keduanya berasal dari model *Neural Network* dengan *Random Under Sampling*.

Tabel 9 Perbandingan AUC NN dan NN+RUS

Dataset	NN	NN+RUS
CM1	0.78	0.79
KC1	0.76	0.77
KC3	0.82	0.814
MC2	0.84	0.87
MW1	0.84	0.84
PC1	0.8	0.81
PC2	0.77	0.84
PC3	0.71	0.77
PC4	0.77	0.78
PC5	0.83	0.88

Berdasarkan Tabel 9 didapatkan hasil AUC terbaik pada model *Neural Network* dengan *Random Under Sampling*, dimana terdapat delapan dataset (CM1, KC3, MC2, PC1, PC2, PC3, PC4 dan PC5). Satu dataset (MW1) memiliki nilai yang sama sedangkan satu dataset lainnya (KC3) menunjukkan model *Neural Network* memiliki AUC terbaik. Nilai AUC terbesar terdapat pada dataset PC5 (88) dari model *Neural Network* dengan *Random Under Sampling* sebaliknya nilai terkecil terdapat pada dataset PC3 dengan model *Neural Network*.

Signifikansi penggunaan *Random Under Sampling* untuk peningkatan kinerja *Neural Network* pada prediksi cacat software dapat diketahui menggunakan uji statistic. Uji statistik yang aman dan sesuai dengan algoritma machine learning dapat dilakukan dengan uji nonparametrik (Demsar, 2006), terdiri dari uji peringkat bertanda Wilcoxon dan uji friedman.

Pada uji peringkat bertanda *Wilcoxon (Wilcoxon Signed Ranged test)* satu sisi untuk sisi bawah ditetapkan hipotesis sebagai berikut:

H0: $\mu_{NN} \geq \mu_{NN+RUS}$ (Model NN+RUS tidak lebih baik dari model NN).

H1: $\mu_{NN} \leq \mu_{NN+RUS}$ (Model NN+RUS lebih baik dari model NN).

Rekap untuk uji Wilcoxon berdasarkan AUC dapat dilihat pada Tabel 10

Tabel 10 Rekap Uji Peringkat Bertanda Wilcoxon untuk Model NN dan NN+RUS

Pengukuran	Jumlah			Rata-rata Peringkat		P-Value	Hasil ($\alpha < 0,05$)
	N	P	T	N	P		
Akurasi	4	4	2	2.9	3.1	0.3	Not Sig. ($P > 0.05$), Model NN+RUS tidak lebih baik dari NN
Sensitivitas	1	8	1	0.73	4.32	0.025	Sig. ($P < 0.05$) , maka model NN+RUS lebih baik dari NN
F-Measure	4	5	1	1.76	2.35	0.02	Sig. ($P < 0.05$) , maka model NN+RUS lebih baik dari NN
G-Mean	2	7	1	1.09	4.76	0.377	Not Sig. ($P > 0.05$), maka model NN+RUS tidak lebih baik dari NN
AUC	1	8	1	0.814	6.5	0.002	Sig. ($P < 0.05$) , maka model NN+RUS lebih baik dari NN

Pada uji Friedman satu sisi untuk sisi dibawah ditetapkan hipotesis sebagai berikut:

$H_0: \mu_{NN} = \mu_{NN+RUS}$ (Tidak ada perbedaan antara model NN dengan NN+RUS)

$H_1: \mu_{NN} \neq \mu_{NN+RUS}$ (Ada perbedaan antara model NN dengan NN+RUS)

Rekap perbandingan AUC pada model prediksi cacat software dapat dilihat pada Tabel 11, hasil uji Friedman dapat dilihat pada Tabel 12 dan Nilai P-value pada Tabel 13

Tabel 11 Perbandingan AUC pada model prediksi cacat software

	C MI	KC 1	KC 3	M C2	M WI	PC 1	PC 2	PC 3	PC 4	PC 5
NN	0.78	0.76	0.82	0.84	0.83	0.87	0.77	0.71	0.77	0.83
NN+RUS	0.79	0.77	0.82	0.87	0.84	0.81	0.84	0.77	0.78	0.88
NN+Boosting	0.78	0.765	0.816	0.842	0.83	0.82	0.73	0.78	0.88	0.84
NN+ABC	0.77	0.8	0.816	0.82	0.83	0.82	0.73	0.71	0.71	0.83
NN+ROS	0.786	0.762	0.82	0.844	0.832	0.883	0.72	0.77	0.74	0.84
NN+SMOTE	0.77	0.76	0.799	0.85	0.83	0.878	0.71	0.77	0.83	0.83
NB	0.76	0.76	0.72	0.812	0.828	0.85	0.73	0.74	0.825	0.8
NB+RUS	0.78	0.756	0.73	0.83	0.84	0.87	0.738	0.745	0.81	0.88
LR	0.72	0.68	0.8	0.86	0.87	0.87	0.71	0.75	0.88	0.88
C45	0.736	0.614	0.731	0.805	0.805	0.816	0.736	0.736	0.736	0.736

Tabel 12 Hasil Uji Friedman

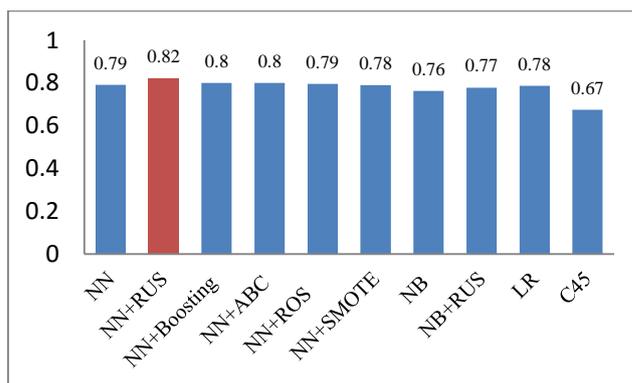
Q (Observed value)	43.723
Q (Critical value)	16.919
DF	9
p-value	< 0.0001
Alpha	0.05

Tabel 13 Nilai P-value Berdasarkan Uji Friedman

	NN	NN+RUS	NN+Boosting	NN+ABC	NN+ROS	NN+SMOTE	NB	NB+RUS	LR	C45
NN	1	0.032	0.973	0.989	0.986	1.000	0.996	1.000	1.000	0.232
NN+RUS	0.032	1	0.518	0.420	0.444	0.011	0.001	0.090	0.082	< 0.001
NN+Boosting	0.973	0.518	1	1.000	1.000	0.887	0.518	0.998	0.997	0.006
NN+ABC	0.989	0.420	1.000	1	1.000	0.936	0.619	0.999	0.999	0.011
NN+ROS	0.986	0.444	1.000	1.000	1	0.925	0.594	0.999	0.999	0.009
NN+SMOTE	1.000	0.011	0.887	0.936	0.925	1	1.000	0.999	1.000	0.200
NB	0.996	0.001	0.518	0.619	0.594	1.000	1	0.999	0.999	0.821
NB+RUS	1.000	0.090	0.998	0.999	0.999	0.999	0.999	1	1.000	0.099
LR	1.000	0.082	0.997	0.999	0.999	1.000	0.999	1.000	1	0.109
C45	0.232	0.001	0.006	0.011	0.009	0.420	0.821	0.099	0.109	1

Berdasarkan Tabel 13 NN+RUS memiliki perbedaan yang signifikan dengan NN, NN+SMOTE, NB, dan C45 karena nilai alpha < 0.05.

Nilai AUC dihitung rata-ratanya agar dapat dibuat grafik perbandingan antar model. Rata-rata AUC terlihat pada Gambar 2.



Gambar 2 Grafik Perbandingan Rata-Rata AUC NN+RUS dengan Model Lainnya

Berdasarkan Gambar 2 rata-rata AUC tertinggi terdapat pada model NN+RUS yaitu 0.82.

Tabel 14 Nilai Rata-Rata AUC, Keterangan dan Diagnostik

	Rata-Rata AUC	Klasifikasi	Simbol
NN	0.792	Fair classification	→
NN+RUS	0.817	Good classification	↗
NN+Boosting	0.8	Fair classification	→
NN+ABC	0.8	Fair classification	→
NN+ROS	0.7957	Fair classification	→
NN+SMOTE	0.7897	Fair classification	→
NB	0.7625	Fair classification	→
NB+RUS	0.7779	Fair classification	→
LR	0.787	Fair classification	→
C45	0.6746	Poor classification	→

Berdasarkan Tabel 4.102 didapatkan NN, NN+Boosting, NN+ABC, NN+ABC, NN+ROS, NN+SMOTE, NB, NB+RUS, dan LR termasuk *fair classification* sedangkan C45 termasuk *poor classification* dan NN+RUS termasuk *good classification*.

5. KESIMPULAN

Hasil eksperimen pada penelitian ini mendapatkan nilai AUC terbesar terdapat pada model NN+RUS yaitu 0.88 pada dataset PC5. Rata-rata AUC NN mengalami peningkatan sebesar 0.02 setelah diterapkan RUS. Hasil uji Wilcoxon dan Friedman menunjukkan bahwa bahwa AUC NN+RUS memiliki perbedaan yang signifikan dengan NN dengan p-value wilcoxon = 0.002 dan p-value friedman = 0.003 ($p < 0.005$). Hasil perbandingan model yang diusulkan dengan metode lain didapatkan nilai AUC terbesar pada model NN+RUS, NB dan LR (0.88) pada dataset PC5 dengan rata-rata AUC terbesar pada NN+RUS (0.82). Menurut uji *friedman* terdapat perbedaan AUC yang signifikan antara NN+RUS dengan NN, NN+SMOTE, NB, dan C45 karena nilai p-value < 0.0001 .

Dari pengujian diatas maka dapat disimpulkan bahwa penggunaan metode NN+RUS mampu menangani ketidakseimbangan kelas dataset pada prediksi cacat *software* berbasis *Neural Network* dengan menghasilkan AUC lebih tinggi dibandingkan metode NN yang tidak menggunakan RUS.

DAFTAR PUSTAKA

- Arar, Ö. F., & Ayan, K. (2015). *Software defect prediction using cost-sensitive neural network. Applied Soft Computing*, 1–15. <http://doi.org/10.1016/j.asoc.2015>
- Chen, H., Zhang, J., Xu, Y., Chen, B., & Zhang, K. (2012). Performance comparison of artificial *Neural Network* and logistic regression model for differentiating lung nodules on CT scans. *Xpert Systems with Applications*, (11503–11509), 39(13).
- Chawla, Lazarevic, & Lawrence. (n.d.). *No Title SMOTEBoost: Improving Prediction of the Minority Class in Boosting. Principles and Practice of Knowledge Discovery in Database. Dubrovnik* (pp. 107–119).
- Demsar, J. (2006). Statistical Comparisons of Classifiers over Multiple Data Sets. *The Journal of Machine Learning Research*, 1–30.
- Gao, K., Khoshgoftaar, T., & Wald, R. (2014). Combining Feature Selection and Ensemble Learning for *Software Quality Estimation. The Twenty-Seventh International Flairs Conference*, 47–52.
- Gorunescu, F. (2011). *Data Mining: Concepts, Models, and Techniques. Springer*.
- Hall, T., Beecham, S., Bowes, D., Gray, D., & Counsell, S. (2012). A systematic literature review on fault prediction performance in *software engineering. IEEE Transactions on Software Engineering*, 38(03), 1276–1304. <http://doi.org/10.1109/TSE.2011.103>
- Han, J., Kamber, M., & Pei, J. (2012). *Data Mining Concepts and Techniques No Title*. San Fransisco: Morgan Kauffman.
- Harrington, P. (2012). *Machine Learning in Action. Manning Publications Co*.
- Hastie, T., Tibshirani, R., & Friedman, J. (2011). *The elements of statistical learning: data mining, interence, and prediction. Springer. Springer*.
- Jones, C., & Bonsignour, O. (2012). *The Economics of Software Quality*.
- Jong, J. S. (2009). *Jaringan Syaraf Tiruan & Pemrogramannya Menggunakan MATLAB*. Yogyakarta: Andi Yogyakarta.
- Journal, I., & Models, I. (2014). *A Study Of Application Of Neural Network Technique On Software Repositories* Ana Maria Bautista, Tomas San Feliu Research methodology, 3(3).
- Khoshgoftaar, T. M., Gao, K. G. K., & Seliya, N. (2010). Attribute Selection and Imbalanced Data: Problems in *Software Defect Prediction. Tools with Artificial Intelligence (ICTAI), 2010 22nd IEEE International Conference on, 1. http://doi.org/10.1109/ICTAI.2010.27*
- Korada, N. K., Kumar, N. P., & Deekshitulu, Y. (2012). Implementatioan Of Naive Bayesian Classifier and Ada-Boost Aloritm Using Maiz Expert System. *International Journal Of Information Sciences And Techniques (IJIST) Vol.2, No.3, 63-75, 2(3), 63–75*.
- Liebchen, G. a. & Shepperd, M. (2008). Data sets and data quality in *software engineering. Proceedings of the 4th International Workshop on Predictor Models in Software Engineering - PROMISE*, 39.
- Park, B. J., Oh, S. K., & Pedrycz, W. (2013). The design of polynomial function-based *Neural Network* predictors for detection of *software defects. Information Sciences*, 229, 40–57. <http://doi.org/10.1016/j.ins.2011.01.026>
- Park, B., Oh, S., & Pedrycz, W. (2013). The design of polynomial function-based *Neural Network* predictors for detection of *software defects. Information Sciences*, 229, 40–57. <http://doi.org/10.1016/j.ins.2011.01.026>
- Pressman, R. S. (2010). *Software Engineering A Practitioner's Approach Sevent Edition* (p. (p. 895)). New York, NY: McGraw-Hill Companies, Inc.
- Rianto, H., Pascasarjana, P., Ilmu, M., Tinggi, S., Informatika, M., Komputer, D. A. N., & Mandiri, N. (2015). *Resampling Logistic Regression Untuk Penanganan Ketidakseimbangan Data Skala Besar Pada Prediksi Cacat Software*.
- Saifudin, A., & Wahono, R. S. (2015). Penerapan Teknik Ensemble untuk Menangani Ketidakseimbangan Kelas pada Prediksi Cacat *Software, 1(1)*.
- Setiyorini, T., Pascasarjana, P., Ilmu, M., Tinggi, S., Informatika, M., Komputer, D. a N., & Mandiri, N. (2014a). Penerapan Metode Bagging Untuk Mengurangi Data Noise Pada *Neural Network Untuk Estimasi Kuat Tekan Beton Penerapan Metode Bagging Untuk Mengurangi Data Noise Pada Neural Network Untuk, 1(1), 36–41*.
- Shepperd, M., Song, Q., Sun, Z., & Mair, C. (2013). Data quality: Some comments on the NASA *software defect datasets. IEEE Transactions on Software Engineering*, 39, 1208–1215. <http://doi.org/10.1109/TSE.2013.11>
- Wahono, R. S. (2015). A Systematic Literature Review of *Software Defect Prediction : Research Trends , Datasets , Methods and Frameworks, 1(1)*.
- Wahono, R. S., & Herman, N. S. (2014). Genetic feature selection for *software defect prediction. Advanced Science Letters*, 20(1), 239–244. <http://doi.org/10.1166/asl.2014.5283>
- Wahono, R. S., Herman, N. S., & Ahmad, S. (2014). *Neural Network Parameter Optimization Based on Genetic Algorithm for Software Defect Prediction. Advanced Science Letters*, 20(10), 1951–1955. <http://doi.org/10.1166/asl.2014.5641>
- Wahono, R. S., & Suryana, N. (2013). Combining particle swarm optimization based feature selection and bagging technique for *software defect prediction. International Journal of Software Engineering and Its Applications*, 7(5), 153–166. <http://doi.org/10.14257/ijseia.2013.7.5.16>
- Wahono, R. S., Suryana, N., & Ahmad, S. (2014). Metaheuristic Optimization based Feature Selection for *Software Defect Prediction. Journal of Software*, 9(5), 1324–1333. <http://doi.org/10.4304/jsw.9.5.1324-1333>
- Wang, B. X., & Japkowicz, N. (2010). Boosting support vector machines for imbalanced data sets. *Knowledge and Information Systems*, 25, 1–20. <http://doi.org/10.1007/s10115-009-0198-y>
- Witten, I. H., Frank, E., & Hal, M. A. (2011). *Data Mining Practical Mechine Learning Tools and Techniques Third Edition* (3rd ed.). Elsevier Inc.
- Wu, X., & Kumar, V. (2010). No Title. *Taylor & Francis Grop*, 5, 158.

- Yap, B. W., Ran, K., Rahman, H. A. A., Fong, S., Khairudin, Z., & Abdullah, N. N. (2014). No Title. *Electrical Engineering*, 285, 12–13.
- Yu, D., Hu, J., Tang, Z., Shen, H., Yang, J., & Yang, J. (2013). Neurocomputing Improving protein-ATP binding residues prediction by boosting SVMs with random under-sampling. *Neurocomputing*, 104, 180–190. <http://doi.org/10.1016/j.neucom.2012.10.012>
- Zamani, A. M., & Amaliah, B. (2012). Implementasi Algoritma Genetika pada Struktur Backpropagation *Neural Network* untuk Klasifikasi Kanker Payudara, 1.
- Zhang, Z.-Z., Chen, Q., Ke, S.-F., Wu, Y.-J., Qi, F., & Zhang, Y.-P. (2008). Ranking Potential Customers Based on Group-Ensemble. *International Journal of Data Warehousing and Mining*, 4(2), 79–89. <http://doi.org/10.4018/jdwm.2008040109>
- Zheng, J. (2010). Cost-sensitive boosting neural networks for software defect prediction. *Expert Systems with Applications*, 37(6), 4537–4543. <http://doi.org/10.1016/j.eswa.2009.12.056>

BIOGRAFI PENULIS



Erna Irawan. Memperoleh gelar Universitas BSI Bandung dan M.Kom dari Program Pasca Sarjana Program Studi Magister Ilmu Komputer STIMIK Nusa Mandiri, Jakarta. Saat ini bekerja sebagai dosen di Universitas BSI Bandung. Minat penelitiannya learning dan software engineering (rekayasa perangkat lunak).



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.

Integrasi Bagging dan Greedy Forward Selection pada Prediksi Cacat *Software* dengan Menggunakan Naïve Bayes

Fitriyani¹ dan Romi Satria Wahono²

¹Fakultas Ilmu Komputer, STMIK Nusa Mandiri
fitriyani.fitriyani@hotmail.co.id

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

Abstrak: Kualitas *software* ditemukan pada saat pemeriksaan dan pengujian. Apabila dalam pemeriksaan atau pengujian tersebut terdapat cacat *software* maka hal tersebut akan membutuhkan waktu dan biaya dalam perbaikannya karena biaya untuk estimasi dalam memperbaiki *software* yang cacat dibutuhkan biaya yang mencapai 60 Miliar pertahun. Naïve bayes merupakan algoritma klasifikasi yang sederhana, mempunyai kinerja yang bagus dan mudah dalam penerapannya, sudah banyak penelitian yang menggunakan algoritma naïve bayes untuk prediksi cacat *software* yaitu menentukan *software* mana yang masuk kategori cacat dan tidak cacat pada. Dataset NASA MDP merupakan dataset publik dan sudah banyak digunakan dalam penelitian karena sebanyak 64.79% menggunakan dataset tersebut dalam penelitian prediksi cacat *software*. Dataset NASA MDP memiliki kelemahan adalah kelas yang tidak seimbang dikarenakan kelas mayoritas berisi tidak cacat dan minoritas berisi cacat dan kelemahan lainnya adalah data tersebut memiliki dimensi yang tinggi atau fitur-fitur yang tidak relevan sehingga dapat menurunkan kinerja dari model prediksi cacat *software*. Untuk menangani ketidakseimbangan kelas dalam dataset NASA MDP adalah dengan menggunakan metode ensemble (bagging), bagging merupakan salah satu metode ensemble untuk memperbaiki ketidakseimbangan kelas. Sedangkan untuk menangani data yang berdimensi tinggi atau fitur-fitur yang tidak memiliki kontribusi dengan menggunakan seleksi fitur greedy forward selection. Hasil dalam penelitian ini didapatkan nilai AUC tertinggi adalah menggunakan model naïve bayes tanpa seleksi fitur adalah 0.713, naïve bayes dengan greedy forward selection sebesar 0.941 dan naïve bayes dengan greedy forward selection dan bagging adalah sebesar 0.923. Akan tetapi, dilihat dari rata-rata peringkat bahwa naïve bayes dengan greedy forward selection dan bagging merupakan model yang terbaik dalam prediksi cacat *software* dengan rata-rata peringkat sebesar 2.550.

Kata Kunci: Naïve Bayes, Greedy Forward Selection, Bagging, Ketidakseimbangan Kelas, Fitur-Fitur yang tidak Relevan.

1 PENDAHULUAN

Prediksi cacat *software* merupakan salah satu kegiatan penting pada fase atau tahap *testing* dalam *Software Development Life Cycle* (Arora, Tatarwal, & Saha, 2015). Terbukti pada tahun 2002, menurut NIST (*National Institute of Standards and Technology*) estimasi biaya cacat *software*

mencapai \$60 billion atau sekitar 60 miliar pertahun (Strate & Laplante, 2013) dan lebih dari 30 tahun prediksi cacat *software* merupakan topik yang penting dalam *software engineering*. Prediksi yang akurat pada kesalahan yang mungkin dapat terjadi dalam kode merupakan hal yang penting karena dapat membantu pada saat tahap pengujian, pengurangan biaya dan peningkatan kualitas perangkat lunak (Song, Jia, Shepperd, Ying, & Liu, 2010).

Kualitas dianggap sebagai isu penting dalam bidang *software engineering*, akan tetapi membangun kualitas perangkat lunak sangat membutuhkan biaya yang mahal untuk meningkatkan efektifitas dan efisiensi jaminan kualitas dan pengujian (Chang, Mu, & Zhang, 2011). Prediksi cacat *software* ini berupaya untuk meningkatkan kualitas perangkat lunak dan efisiensi pengujian dengan membangun model prediksi klasifikasi dari atribut kode untuk mengidentifikasi secara tepat pada modul rawan kesalahan (Lessmann, Baesens, Mues, & Pietsch, 2008). Saat ini penelitian tentang prediksi cacat *software* berfokus pada metode klasifikasi sebanyak 77.46%, 14.08% menggunakan metode estimasi dan sebanyak 1.41% menggunakan metode *clustering* dan asosiasi (Wahono, 2015).

Dataset NASA MDP merupakan data metrik perangkat lunak yang kerap kali digunakan dalam penelitian *software defect prediction* atau prediksi cacat *software*. Dataset NASA mudah diperoleh dan tersedia untuk umum karena sebanyak 64.79% penelitian menggunakan publik dataset dan 35.21% penelitian menggunakan dataset privat (Wahono, 2015).

Masalah dalam *software defect prediction* adalah *redundant data*, korelasi, fitur yang tidak relevan, *missing samples* dan masalah ini dapat membuat dataset tidak seimbang karena sulit untuk memastikan antara data cacat atau tidak cacat (Laradji, Alshayeb, & Ghouti, 2015). *Feature selection* (seleksi fitur atau atribut) dapat menangani untuk mengurangi masalah *redundant data* dan fitur yang tidak relevan. Seleksi fitur merupakan langkah yang penting dalam mesin pembelajaran (*machine learning*) (Laradji, Alshayeb, & Ghouti, 2015). Salah satu metode seleksi fitur adalah menggunakan greedy forward selection. Seleksi fitur dengan greedy forward selection sangat efisien, sederhana dan tidak seperti teknik seleksi fitur yang membutuhkan waktu lama dalam prosesnya (forward selection dan backward elimination). Greedy forward selection hanya memilih fitur yang berkontribusi dan dapat meningkatkan performa klasifikasi (Laradji, Alshayeb, & Ghouti, 2015).

Selain masalah *redundant data* dan fitur-fitur yang tidak relevan, pada dataset cacat *software* ditemukan dataset yang

tidak seimbang (*imbalance class*) karena data yang cacat jumlahnya lebih sedikit dibandingkan dengan data yang tidak cacat, sehingga data yang termasuk kelas mayoritas adalah tidak cacat dan data yang termasuk kelas minoritas adalah cacat. Untuk menangani masalah dalam *imbalance class* salah satunya adalah menggunakan teknik ensemble (boosting dan bagging). Bagging (bootstrap aggregating) merupakan teknik yang dapat meningkatkan klasifikasi dengan kombinasi klasifikasi secara acak pada dataset *training* dan bagging juga dapat mengurangi variansi dan menghindari *overfitting* (Wahono & Suryana, Combining Particle Swarm Optimization based Feature Selection and Bagging Technique for Software Defect, 2013).

Banyak penelitian yang telah dilakukan sebelumnya dan algoritma naive bayes merupakan model yang terbaik dibandingkan model lainnya seperti: logistic regression, neural network, random forest, decision tree, support vector machine dan k-nearest neighbor. Naive bayes merupakan algoritma klasifikasi yang sederhana dan mudah diimplementasikan sehingga algoritma ini sangat efektif apabila diuji dengan dataset yang tepat, terutama bila naive bayes dengan seleksi fitur, maka naive bayes dapat mengurangi *redundant* pada data (Witten, Frank, & Hall, 2011). Algoritma naive bayes termasuk dalam *supervised learning* dan salah satu algoritma pembelajaran tercepat yang dapat menangani sejumlah fitur atau kelas (Lee, 2015).

Pada penelitian ini untuk menangani masalah *redundant data* menggunakan seleksi fitur greedy forward selection dan untuk menangani *imbalance class* menggunakan teknik ensemble bagging, sedangkan algoritma klasifikasi yang digunakan dalam penelitian ini adalah algoritma naive bayes. Pada penelitian ini menggunakan dataset NASA (*National Aeronautics and Space Administration*) MDP (*Metrics Data Program*) repository sebagai *software metrics*.

2 PENELITIAN TERKAIT

Banyak penelitian tentang prediksi cacat *software* dan sudah banyak metode yang dalam penelitian tersebut menghasilkan kinerja yang baik, akan tetapi sebelum dilakukan penelitian adalah mengkaji metode-metode yang sudah menggunakan metode yang lain dalam penelitian sebelumnya, sehingga dapat mengetahui *state of the art* dalam penelitian yang membahas tentang fitur-fitur yang tidak relevan (*irrelevant features*) dan ketidakseimbangan kelas (*imbalanced class*).

Penelitian yang dilakukan oleh Song, Jia, Shepperd, Ying dan Liu bahwa prediksi cacat *software* merupakan topik penelitian yang penting dan lebih dari 30 tahun penelitian prediksi cacat *software* memfokuskan pada estimasi jumlah cacat pada sistem *software*, menemukan asosiasi atau hubungan antara cacat, mengklasifikasikan antara cacat dan tidak cacat (Song, Jia, Shepperd, Ying, & Liu, 2010). Pada penelitian ini, *data preprocessing* merupakan bagian yang penting dalam menangani *missing values*, *discretizing* atau transformasi untuk atribut *numeric* dan menangani masalah tersebut dapat menggunakan metode *log filtering*. Setelah proses *data processing* pada prediksi cacat *software*, dilakukan pemilihan atribut terbaik atau dapat disebut dengan *attribute selection*. Dalam penelitian ini untuk pemilihan atribut terbaik menggunakan *forward selection* dan *backward elimination* dan algoritma pembelajarannya yang digunakan adalah naive bayes, J48 dan OneR. Hasilnya dapat diketahui bahwa naive bayes dengan *log filtering* dan seleksi fitur (*forward selection* dan *backward elimination*) lebih baik daripada algoritma J48 dan OneR untuk meningkatkan AUC.

Pada penelitian (Khoshgoftaar, Hulse, & Napolitano, 2011) *noisy* dan *imbalance class* lebih efektif ditangani dengan teknik bagging dan boosting karena *imbalance class* dan *noise* dapat berpengaruh pada kualitas data dalam hal kinerja klasifikasi. Dataset dalam penelitian ini menggunakan Letter A, Nursey3, OptDigits8, Splice2, sedangkan metode dalam penelitian ini adalah menggunakan teknik data *sampling* seperti *Random Undersampling* (RUS) dan *Synthetic Minority Oversampling Technique* (SMOTE) yang menggabungkan dengan teknik bagging dan boosting sehingga penelitian ini menggunakan metode *Exactly Balanced Bagging* (EBBag), *Roughly Balanced Bagging* (RBBag), SMOTE dan Boosting (SMOTEBoost), RUS dan Boosting (RUSBoost), EBBag dan RBBag dengan *Replacement* (EBBagR dan RBBagR), EBBag dan RBBag tanpa *Replacement* (EBBagN dan RBBagN), SMOTE dan Boosting dengan *Reweighting* (SMOTEBoostW), RUS dan Boosting dengan *Reweighting* (RUSBoostW), SMOTE dan Boosting dengan *Resampling* (SMOTEBoostS), RUS dan Boosting dengan *Resampling* (RUSBoostS). Algoritma pembelajaran menggunakan C4.5D (*Default Parameter*) dan C4.5N (*disable pruning* dan *enable laplace smoothing*), naive bayes dan RIPPER. Hasil akhir dari penelitian ini adalah bahwa teknik bagging lebih baik tanpa menggunakan *replacement* untuk pembelajaran pada *noisy* dan *imbalance class*, walaupun teknik boosting lebih populer untuk menangani *imbalance class*, sehingga metode bagging lebih baik tanpa *replacement* untuk menangani *noisy* dan *imbalance class* dengan AUC tertinggi sebesar 0.947.

Penelitian yang dilakukan oleh Ma, Luo, Zeng dan Chen bahwa prediksi kualitas modul *software* merupakan hal yang sangat kritis, dalam penelitian ini (Ma, Luo, Zeng, & Chen, 2012) menggunakan algoritma klasifikasi naive bayes di weka dengan sebutan metode CC, *nearest neighbor filter* dengan sebutan NN-filter dan transfer naive bayes dengan sebutan TNB. Dataset dalam penelitian ini menggunakan dataset NASA: KC1, MC2, KC3, MW1, KC2, PC1, CM1. Hasilnya bahwa menggunakan algoritma TNB dapat meningkatkan kinerja menjadi lebih bagus dan dapat meningkatkan AUC sebesar 0.6236 pada KC1, 0.6252 pada MC2, 0.7377 pada KC3, 0.6777 pada MW1, 0.7787 pada KC2, 0.5796 pada PC1 dan 0.6594 pada CM1. AUC lebih meningkat dengan menggunakan algoritma TNB dan AUC dengan hasil AUC tertinggi sebesar 0.77 pada dataset KC2.

Pada penelitian (Wahono & Herman 2014), metaheuristic optimization untuk *feature selection* dapat menggunakan Genetic Algorithm (GA) dan Particle Swarm Optimization (PSO) dan untuk dapat lebih meningkatkan akurasi dari prediksi cacat *software* menggunakan metode bagging. Algoritma klasifikasi pada penelitian ini sebanyak 10 algoritma seperti: (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)). Dataset dalam penelitian ini menggunakan dataset NASA MDP dengan menggunakan 9 dataset sebagai berikut: CM1, KC1, KC3, MC2, MW1, PC1, PC2, PC3, PC4. Hasil akhir dari metode ini dapat menghasilkan peningkatan yang mengesankan pada banyak metode klasifikasi dan berdasarkan hasil perbandingan antara GA dan PSO tidak ada perbedaan yang signifikan ketika menggunakan *feature selection* pada banyak metode klasifikasi dalam prediksi cacat *software*. AUC rata-rata meningkat 25.99% untuk GA dan PSO meningkat 20.41%.

3 METODE YANG DIUSULKAN

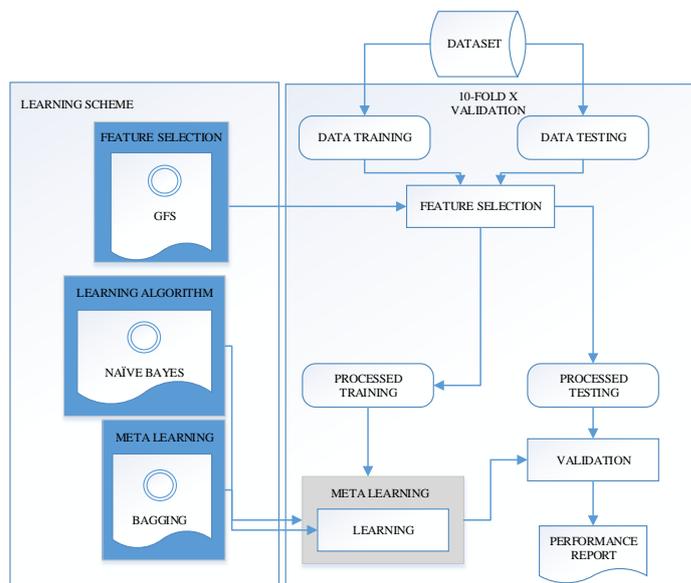
Penelitian ini dilakukan dengan mengusulkan model, melakukan eksperimen dengan menguji model yang diusulkan, evaluasi dan validasi, mengembangkan aplikasi. Pada penelitian ini dataset yang digunakan adalah dataset NASA (*National Aeronautics and Space Administration*) MDP (*Metrics Data Program*), dataset yang digunakan sebanyak 10 dataset yaitu: CM1, JM1, KC1, KC3, MC2, PC1, PC2, PC3, PC4 DAN PC5. Tabel 1 merupakan spesifikasi dari dataset NASA MDP.

Tabel 1 Spesifikasi Dataset NASA MDP

Atribut	NASA MDP									
	CM1	JM1	KC1	KC3	MC2	PC1	PC2	PC3	PC4	PC5
LOC_BLANK	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
LOC_CODE_AND_COMMENT	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
LOC_COMMENTS	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
LOC_EXECUTABLE	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
LOC_TOTAL	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
NUMBER_OF_LINES	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
HALSTEAD_CONTENT	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
HALSTEAD_DIFFICULTY	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
HALSTEAD_EFFORT	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
HALSTEAD_ERROR_EST	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
HALSTEAD_LENGTH	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
HALSTEAD_LEVEL	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
HALSTEAD_PROG_TIME	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
HALSTEAD_VOLUME	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
NUM_OPERANDS	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
NUM_OPERATORS	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
NUM_UNIQUE_OPERANDS	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
NUM_UNIQUE_OPERATORS	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
CYCOMATIC_COMPLEXITY	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
CYCOMATIC_DENSITY	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
DESIGN_COMPLEXITY	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
ESSENTIAL_COMPLEXITY	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BRANCH_COUNT	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
CALL_PAIRS	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
CONDITION_COUNT	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
DECISION_COUNT	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
DECISION_DENSITY	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
DESIGN_DENSITY	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
EDGE_COUNT	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
ESSENTIAL_DENSITY	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
GLOBAL_DATA_COMPLEXITY	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
GLOBAL_DATA_DENSITY	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MAINTENANCE_SEVERITY	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MODIFIED_CONDITION_COUNT	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MULTIPLE_CONDITION_COUNT	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
NODE_COUNT	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
NORMALIZED_CYCOMATIC	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
PARAMETER_COUNT	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
PERCENT_COMMENTS	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
PATHOLOGICAL_COMPLEXITY	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Defective	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Jumlah atribut	37	21	21	39	39	37	37	37	37	37
Jumlah modul	344	9591	2095	200	127	759	1586	1125	1399	17001
Jumlah modul cacat	42	1759	325	36	44	61	16	140	178	503
Presentase modul cacat	12%	18%	16%	18%	35%	8%	1%	12%	13%	3%
Jumlah modul tidak cacat	302	7834	1771	164	83	698	1569	985	1221	16498

Pada penelitian ini mengusulkan model Naïve Bayes dan seleksi fitur Greedy Forward Selection (NB+GFS), model yang diusulkan selanjutnya adalah model Naïve Bayes dengan Greedy Forward Selection dan Bagging (NB+GFS+BG), model bagging untuk menangani permasalahan ketidakseimbangan kelas (*imbalance class*). Gambar 1 merupakan kerangka penelitian yang diusulkan.

Eksperimen dilakukan dengan menguji model menggunakan aplikasi Rapidminer, hasil kinerja dari model di evaluasi menggunakan friedman test untuk dapat membandingkan model-model yang diusulkan sehingga dapat diketahui model yang terbaik pada penelitian prediksi cacat *software*. Validasi yang digunakan dalam penelitian ini menggunakan *10-fold cross validation*. Pengembangan aplikasi yang dilakukan menggunakan IDE (*Integrated Development Environment*) Netbeans dengan bahasa Java. Gambar 1 merupakan kerangka penelitian yang diusulkan.

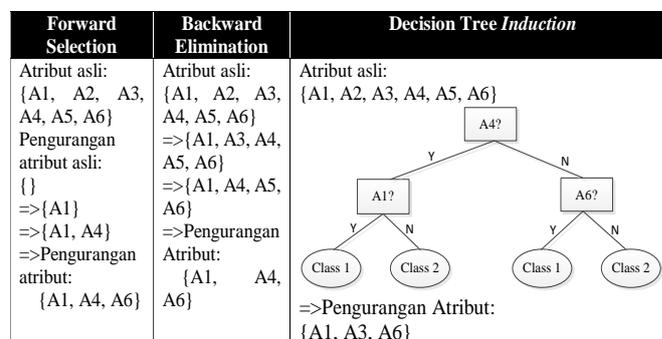


Gambar 1 Kerangka Penelitian

Pada model yang diusulkan dataset akan dibagi menjadi 10 bagian menggunakan *10-fold cross validation*, data bagian pertama menjadi data testing dan data bagian kedua sampai dengan data bagian kesepuluh menjadi data training. Selanjutnya dilakukan seleksi fitur dengan menggunakan Greedy Forward Selection (GFS), sehingga dihasilkan fitur-fitur relevan yang dapat meningkatkan kinerja dari model prediksi cacat *software*. Data training yang sudah dibentuk dengan *cross validation* akan dibuat data training baru secara acak oleh model bagging berbasis naive bayes sebanyak iterasi yang dimasukkan. Jika lebih banyak masuk dalam kategori cacat maka record tersebut di prediksi cacat, sebaliknya jika lebih banyak masuk dalam kategori tidak cacat, maka record tersebut di prediksi tidak cacat.

A. Seleksi Fitur Greedy Forward Selection

Subset selection merupakan metode *feature selection* (seleksi fitur), *subset selection* adalah menemukan *subset* terbaik (fitur), *subset* yang terbaik mempunyai jumlah dimensi yang paling berkontribusi pada akurasi. Seleksi fitur Seleksi fitur untuk menentukan atribut terbaik dan terburuk menggunakan *information gain* (Han, Kamber, & Pei, 2012). Seperti pada Gambar 2.



Gambar 2 Seleksi atribut Greedy
Sumber: (Han, Kamber, & Pei, 2012)

Algoritma greedy dengan seleksi subset atribut (Han, Kamber, & Pei, 2012) sebagai berikut:

1. Stepwise forward selection

Prosedur dimulai dengan himpunan kosong dari atribut sebagai set yang dikurangi, atribut yang terbaik

dari atribut asli ditentukan dan ditambahkan pada set yang kurang. Pada setiap iterasi berikutnya yang terbaik dari atribut asli yang tersisa ditambahkan ke set.

2. Stepwise backward elimination

Prosedur dimulai dengan *full* set atribut. Pada setiap langkah, teknik ini dapat menghilangkan atribut terburuk yang tersisa di dataset.

Atribut terbaik dan terburuk dapat ditentukan dengan menggunakan tes signifikansi statistik yang berasumsi bahwa atribut tidak saling berhubungan dengan satu sama lain (independen) (Han, Kamber, & Pei, 2012). Untuk langkah-langkah dalam menentukan evaluasi dalam pemilihan atribut dapat menggunakan *information gain* yang digunakan dalam pohon keputusan *decision tree* (Han, Kamber, & Pei, 2012).

Seleksi atribut menggunakan *information gain* adalah memilih gain tertinggi dan formula yang digunakan adalah sebagai berikut dimana langkah yang pertama adalah dengan mencari nilai *entropy* sebagai berikut:

$$Info(D) = - \sum_{i=1}^m p_i \log_2(p_i)$$

$Info(D)$ adalah untuk mengetahui nilai dari *entropy*, kemudian jika suatu atribut mempunyai nilai yang berbeda-beda maka menggunakan formula sebagai berikut:

$$Info_A(D) = \sum_{j=1}^v \frac{|D_j|}{|D|} \times Info(D_j)$$

Kemudian untuk mendapatkan hasil *gain*, dimana formulanya menjadi:

$$Gain(A) = Info(D) - Info_A(D)$$

B. Bagging

Teknik ensemble merupakan teknik yang sukses untuk menangani dataset yang tidak seimbang meskipun tidak secara khusus dirancang untuk masalah data yang tidak seimbang (Laradji, Alshayeb, & Ghouti, 2015). Teknik bagging merupakan salah satu teknik ensemble dan teknik ini pada klasifikasi memisahkan data *training* ke dalam beberapa data *training* baru dengan *random sampling* dan membangun model berbasis data *training* baru (Wahono & Suryana, 2013). Algoritma bagging untuk klasifikasi (Liu & Zhou, 2013):

Input: Data set $D = \{(x_i, y_i)\}_{i=1}^n$
 Base learning algorithm \mathcal{L}
 The number of iterations T
 1. for $t=1$ to T do
 2. $h_t = \mathcal{L}(D, \mathcal{D}_{bs})$ /* \mathcal{D}_{bs}
 is the bootstrap distribution */
 3. end for
 Output: $H(x) = \max_y \sum_{t=1}^n I(h_t(x) = y)$
 /* $I(x)=1$ if x is
 true, and 0 otherwise */

C. Naïve bayes

Klasifikasi menggunakan naïve bayes dapat menghasilkan akurasi yang tinggi dan cepat ketika diaplikasikan pada data yang besar (Han, Kamber, & Pei, 2012). Pengklasifikasi naïve bayes berpendapat bahwa nilai dari atribut pada kelas tertentu tidak bergantung (*independence*) pada nilai atribut lainnya, pendapat ini dapat disebut *class-conditional independence* sehingga perhitungannya dapat dibuat lebih sederhana dan disebut “naif (naïve)” (Han, Kamber, & Pei, 2012).

Persamaan naïve bayes dengan menggunakan distribusi gaussian karena dataset NASA MDP merupakan data tipe numerik. Pada distribusi gaussian, dihitung mean μ dan standar deviasi σ pada semua atribut, berikut adalah persamaan yang digunakan:

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

Pengukuran kinerja model menggunakan confusion matrix, *confusion matrix* merupakan alat untuk menganalisa seberapa baik kinerja dari pengklasifikasi dapat mengenali tupel dari kelas yang berbeda (Han, Kamber, & Pei, 2012). Confusion matrix memberikan penilaian kinerja klasifikasi berdasarkan objek dengan benar atau salah (Gorunescu, 2011). Confusion matrix merupakan matrik 2 dimensi yang menggambarkan perbandingan antara hasil prediksi dengan kenyataan.

Berikut adalah persamaan model confusion matrix:

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

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

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

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

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

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

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

F-measure mengkombinasikan *recall* atau *sensitivity* dan *precision* sehingga menghasilkan metrik yang efektif untuk pencarian kembali informasi dalam himpunan yang mengandung masalah ketidakseimbangan.

Pada kelas yang tidak seimbang karena kelas minoritas mendominasi sehingga pengukuran kinerja yang tepat menggunakan Area Under the ROC (*Receiver Operating Characteristic*) Curve (AUC), *f-measure*, *Geometric Mean (G-Mean)*. Area Under ROC Curve (AUC) digunakan untuk memberikan metrik numerik single untuk dapat membandingkan kinerja dari model, nilai AUC berkisar dari 0 sampai 1 dan model yang lebih baik prediksinya adalah yang mendekati nilai 1 (Gao, Khoshgoftaar, & Wald, 2014). Berikut adalah persamaan model AUC:

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

Pedoman umum yang digunakan untuk klasifikasi akurasi sebagai berikut:

1. 0.90-1.00 = *excellent classification*
2. 0.80-0.90 = *good classification*
3. 0.70-0.80 = *fair classification*
4. 0.60-0.70 = *poor classification*
5. 0.50-0.60 = *failure*

Kurva ROC adalah grafik antara sensitivitas (true positive rate) pada sumbu Y dengan 1- spesifisitas pada sumbu X (false positive rate), curve ROC ini seakan-akan menggambarkan tawar-menawar antara sumbu Y atau sensitivitas dengan sumbu X atau spesifisitas. Kurva ROC biasanya digunakan dalam pembelajaran dan data mining, nilai dalam kurva ROC dapat menjadi evaluasi sehingga dapat membandingkan algoritma. Dalam klasifikasi kurva ROC merupakan teknik untuk memvisualisasikan, mengatur dan memilih klasifikasi berdasarkan kinerja dari algoritma (Gorunescu, 2011).

Setelah dilakukan evaluasi terhadap model yang diusulkan, tahap selanjutnya adalah menganalisis model yang diusulkan dengan menggunakan metode statistik uji t (*t-test*) dan uji friedman (*friedman test*).

Uji t sampel berpasangan (*paired-samples t-test*) merupakan prosedur yang digunakan untuk membandingkan rata-rata dari dua variabel, variabel sebelum dan sesudah menggunakan model. Uji t dapat disebut juga dengan z-test dalam statistik dan pengujian ini dilakukan untuk mengetahui apakah ada perbedaan yang signifikan pada dua variabel yang diuji.

Algoritma untuk perbandingan performa z-test (Gorunescu, 2011) sebagai berikut:

Input:

- Masukkan bagian P_1 (akurasi klasifikasi) untuk sampel pertama (model M_1);
- Masukkan bagian P_2 (akurasi klasifikasi) untuk sampel kedua (model M_2);
- Masukkan ukuran sampel pada N_1 untuk sampel pertama;
- Masukkan ukuran sampel pada N_2 untuk sampel kedua;

Kemudian $p - level$ dihitung berdasarkan nilai t untuk perbandingan dengan persamaan berikut ini:

$$|t| = \frac{\sqrt{N_1 \cdot N_2} \cdot |P_1 - P_2|}{\sqrt{p \cdot q}}$$

Dimana:

$$p = \frac{P_1 \cdot N_1 + P_2 \cdot N_2}{N_1 + N_2}, q = 1 - p, \text{ dan untuk } N_1 + N_2 - 2$$

Output:

Level signifikansi untuk akurasi yang berbeda dari dua model.

Dalam statistik jika diketahui $p > 0.05$ berarti tidak ada perbedaan yang signifikan, akan tetapi jika diketahui nilai $p < 0.05$ maka ada perbedaan yang signifikan antara dua model.

4 HASIL EKSPERIMEN

Eksperimen pada penelitian ini menggunakan laptop LENOVO G470 dengan prosesor Intel Celeron CPU B800 @ 1.50 GHz, memori (RAM) 2.00 GB dan sistem operasi Windows 8.1 Pro 32-bit. Aplikasi yang dikembangkan menggunakan IDE (Integrated Development Environment) NetBeans menggunakan bahasa Java. Aplikasi yang digunakan dalam penelitian ini adalah Rapidminer, hasil kinerja dari model selanjutnya di analisis menggunakan menggunakan Microsoft Excel dan XLSTAT.

Hasil pengukuran pada penelitian ini dapat dilihat pada Tabel 1, dimana Tabel 1 adalah hasil akurasi dari model Naïve Bayes (NB), model Naïve Bayes dan Greedy Forward Selection (NB+GFS), model Naïve Bayes dengan Greedy Forward Selection dan Bagging (NB+GFS+BG). Tabel 2 merupakan hasil pengukuran sensitifitas model NB, NB+GFS dan NB+GFS+BG. Tabel 3 menunjukkan hasil pengukuran f-measure model NB, NB+GFS, NB+GFS+BG. Hasil penguruan g-mean dapat dilihat pada Tabel 4 dan Tabel 5 merupakan hasil AUC pada model NB, model NB+GFS dan model NB+GFS+BG.

Tabel 1 Hasil Akurasi

Model	CM1	JM1	KC1	KC3	MC2	PC1	PC2	PC3	PC4	PC5
NB	82.56%	81.31%	82.25%	79.00%	72.37%	88.54%	95.59%	33.53%	87.42%	96.63%
NB+GFS	86.94%	81.48%	83.64%	82.00%	75.64%	91.84%	96.66%	84.00%	83.99%	97.21%
NB+GFS+BG	85.76%	81.58%	83.35%	83.50%	74.87%	91.18%	96.91%	83.56%	86.77%	97.35%

Tabel 2 Hasil Sensitifitas

Model	CM1	JM1	KC1	KC3	MC2	PC1	PC2	PC3	PC4	PC5
NB	33.33%	94.86%	37.54%	36.11%	35.83%	36.07%	18.75%	90.00%	94.10%	44.73%
NB+GFS	34.67%	95.25%	33.54%	33.33%	43.18%	36.07%	6.25%	37.86%	87.39%	33.80%
NB+GFS+BG	28.57%	95.97%	33.85%	41.67%	50%	55.56%	12.50%	36.43%	91.07%	29.62%

Tabel 3 Hasil F-measure

Model	CM1	JM1	KC1	KC3	MC2	PC1	PC2	PC3	PC4	PC5
NB	0.55	0.95	0.58	0.57	0.57	0.58	0.43	0.48	0.94	0.66
NB+GFS	0.39	0.89	0.39	0.40	0.55	0.42	0.04	0.37	0.91	0.42
NB+GFS+BG	0.33	0.90	0.39	0.48	0.58	0.45	0.20	0.36	0.92	0.40

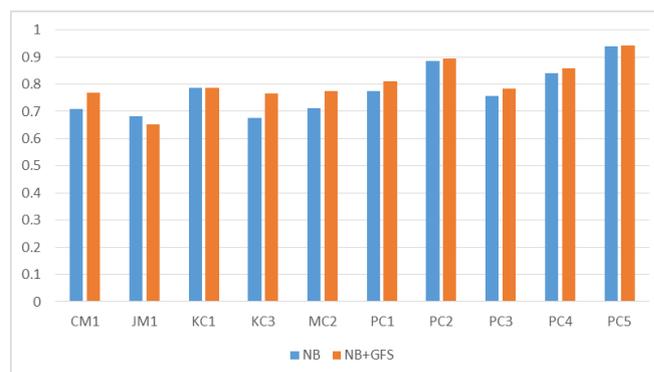
Tabel 4 Hasil G-mean

Model	CM1	JM1	KC1	KC3	MC2	PC1	PC2	PC3	PC4	PC5
NB	0.546	0.461	0.578	0.567	0.551	0.579	0.425	0.472	0.625	0.66
NB+GFS	0.57	0.44	0.56	0.56	0.63	0.59	0.25	0.59	0.73	0.58
NB+GFS+BG	0.52	0.42	0.56	0.62	0.66	0.71	0.35	0.57	0.72	0.54

Tabel 5 Hasil AUC

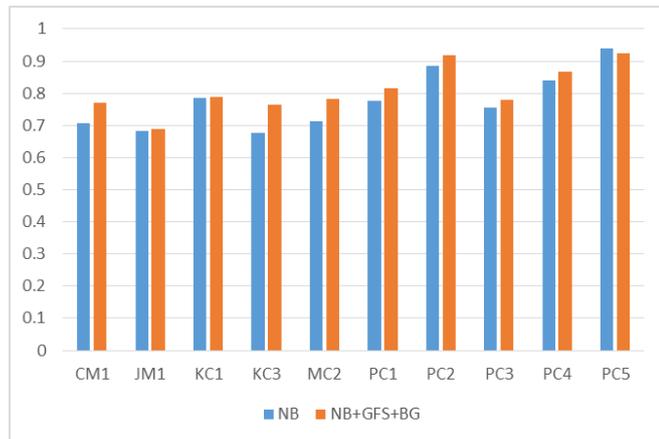
Model	CM1	JM1	KC1	KC3	MC2	PC1	PC2	PC3	PC4	PC5
NB	0.708	0.683	0.786	0.677	0.712	0.775	0.885	0.756	0.84	0.94
NB+GFS	0.769	0.653	0.788	0.765	0.775	0.81	0.894	0.784	0.86	0.941
NB+GFS+BG	0.771	0.688	0.788	0.763	0.781	0.817	0.918	0.778	0.866	0.923

Gambar 3 merupakan grafik perbandingan AUC model Naïve Bayes (NB) dengan model Naïve Bayes dan Greedy Forward Selection (NB+GFS).



Gambar 3 Grafik AUC Model NB dan NB+GFS

Gambar 4 menunjukkan grafik perbandingan AUC antara model Naïve Bayes (NB) dengan model Naïve Bayes dan Greedy Forward Selection dengan Bagging (NB+GFS+BG).



Gambar 4 Perbandingan AUC Model NB dan NB+GFS+BG

Hasil dari pengukuran kinerja, selanjutnya di analisis menggunakan uji t (*t-test*) untuk mengetahui model yang terbaik. Uji t dilakukan dengan membandingkan dua model dan mengukur p-value, jika p-value < nilai alpha (0.05), maka ada perbedaan yang signifikan antara dua model yang dibandingkan. Sebaliknya, jika p-value > nilai alpha, maka tidak ada perbedaan yang signifikan.

Uji t dilakukan pada AUC dengan menggunakan metode statistik untuk menguji hipotesa pada Naïve Bayes (NB) dengan Naïve Bayes dan Greedy forward selection (NB+GFS).

H_0 : Tidak ada perbedaan nilai rata-rata AUC NB dengan NB+GFS.

H_1 : Ada perbedaan antara nilai rata-rata AUC NB dengan NB+GFS.

Tabel 6 Uji t Model NB dan Model NB+GFS

	NB	NB+GFS
Mean	0.7762	0.8039
Variance	0.007838178	0.006342767
Observations	10	10
Pearson Correlation	0.917706055	
Hypothesized Mean Difference	0	
df	9	
t Stat	-2.487968197	
P(T<=t) one-tail	0.017268476	
t Critical one-tail	1.833112933	
P(T<=t) two-tail	0.034536952	
t Critical two-tail	2.262157163	

Pada Tabel 6 dapat diketahui bahwa nilai rata-rata AUC dari model NB+GFS lebih tinggi dibandingkan model NB sebesar 0.8039. Dalam uji beda statistik nilai alpha ditentukan sebesar 0.05, jika nilai p lebih kecil dibanding alpha ($p < 0.05$) maka H_0 ditolak dan H_1 diterima sehingga ada perbedaan yang signifikan antara model yang dibandingkan, akan tetapi jika nilai p lebih besar dibandingkan nilai alpha ($p > 0.05$) maka H_0 diterima dan H_1 ditolak sehingga tidak ada perbedaan yang signifikan antara model yang dibandingkan.

Pada Tabel 6 dapat dilihat bahwa nilai $P(T \leq t)$ adalah 0.03 dan ini menunjukkan bahwa nilai p lebih kecil dibandingkan nilai alpha ($0.03 < 0.05$) sehingga hipotesis H_0 ditolak dan H_1 diterima. Hipotesis H_1 diterima berarti ada perbedaan signifikan antara model NB dan model NB+GFS sehingga model NB+GFS membuat peningkatan ketika dibandingkan dengan modek NB.

Tabel 7 Uji t Model NB dan Model NB+GFS+BG

	NB	NB+GFS+BG
Mean	0.7762	0.8093
Variance	0.007838178	0.005397344
Observations	10	10
Pearson Correlation	0.936299769	
Hypothesized Mean Difference	0	
df	9	
t Stat	-3.221564827	
P(T<=t) one-tail	0.005231557	
t Critical one-tail	1.833112933	
P(T<=t) two-tail	0.010463114	
t Critical two-tail	2.262157163	

Pada Tabel 7 dapat diketahui bahwa nilai rata-rata AUC dari model NB+GFS+BG lebih tinggi dibandingkan model NB sebesar 0.8093. Dalam uji beda statistik nilai alpha ditentukan sebesar 0.05, jika nilai p lebih kecil dibanding alpha ($p < 0.05$) maka H_0 ditolak dan H_1 diterima sehingga ada perbedaan yang signifikan antara model yang dibandingkan, akan tetapi jika nilai p lebih besar dibandingkan nilai alpha ($p > 0.05$) maka H_0 diterima dan H_1 ditolak sehingga tidak ada perbedaan yang signifikan antara model yang dibandingkan.

Pada Tabel 7 dapat dilihat bahwa nilai $P(T \leq t)$ adalah 0.01 dan ini menunjukkan bahwa nilai p lebih kecil dibandingkan nilai alpha ($0.01 < 0.05$) sehingga hipotesis H_0 ditolak dan H_1 diterima. Hipotesis H_1 diterima berarti ada perbedaan signifikan antara model NB dan model NB+GFS+BG sehingga model NB+GFS+BG membuat peningkatan ketika dibandingkan dengan model NB.

Selanjutnya dilakukan uji friedman untuk mengetahui model yang terbaik, pada Tabel 8 menunjukkan hasil p-value dari uji friedman. Nilai yang dibekalkan berarti nilai tersebut lebih kecil dari 0.05. Tabel 9 merupakan signifikansi dari model yang dibandingkan, model NB+GFS ada perbedaan yang signifikan dengan model NB dan model NB+GFS+BG ada perbedaan yang signifikan dengan model NB.

Tabel 8 P-Value Uji Friedman

	NB	NB+GFS	NB+GFS+BG
NB	1	0.049	0.007
NB+GFS	0.049	1	0.780
NB+GFS+BG	0.007	0.780	1

Tabel 9 Signifikan Model Uji Friedman

	NB	NB+GFS	NB+GFS+BG
NB	No	Yes	Yes
NB+GFS	Yes	No	No
NB+GFS+BG	Yes	No	No

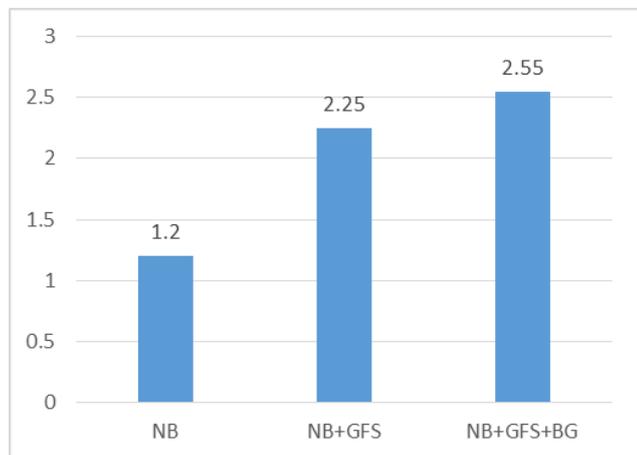
Tabel 10 Table of pairwise differences

	NB	NB+GFS	NB+GFS+BG
NB	0	-1.050	-1.350
NB+GFS	1.050	0	-0.300
NB+GFS+BG	1.350	0.300	0

Critical difference: 1,0481

Peringkat dari rata-rata diperoleh dari perbandingan model dan diketahui bahwa nilai peringkat rata-rata dibagi

dengan jumlah sampel yang digunakan. Perbedaan peringkat dari rata-rata masing-masing model dibandingkan dengan *Critical Difference* yang dapat dilihat pada Tabel 10. *Critical Difference* dapat dihitung menggunakan persamaan $CD = q_{\alpha, \infty, L} \sqrt{\frac{L(L+1)}{12K}}$ dan dalam eksperimen ini dihasilkan nilai *Critical Difference* sebesar 1.0481. Jika nilai pada Tabel 10 lebih besar dibandingkan dengan nilai *Critical Difference* maka model tersebut mempunyai perbedaan yang signifikan dibandingkan model lainnya.



Gambar 5 Rata-Rata Peringkat Uji Friedman

Pada Gambar 5 merupakan rata-rata peringkat dari uji friedman, diketahui bahwa model NB+GFS+BG mempunyai rata-rata peringkat tertinggi, kemudian diikuti dengan model NB+GFS dan model NB mempunyai rata-rata peringkat terendah.

Selanjutnya untuk mengetahui model seleksi fitur yang terbaik dalam menangani fitur-fitur yang tidak relevan, dilakukan perbandingan model dengan seleksi fitur lain. Model yang dibandingkan yaitu: Naïve Bayes (NB), Naïve Bayes dan Bagging (NB+BG), Naïve Bayes dan Greedy Forward Selection (NB+GFS), Naïve Bayes dengan Greedy Forward Selection dan Bagging (NB+GFS+BG), Naïve Bayes dan Genetic Feature Selection (NB+GAFS), Naïve Bayes dan Forward Selection (NB+FS), Naïve Bayes dan Backward Elimination (NB+BE). Tabel 11 menunjukkan hasil AUC dari masing-masing model yang dibandingkan.

Tabel 11 Perbandingan AUC dengan Seleksi Fitur Lain

Model	CM1	JMI	KC1	KC3	MC2	PC1	PC2	PC3	PC4	PC5
NB	0.708	0.683	0.786	0.677	0.712	0.775	0.885	0.756	0.84	0.94
NB+BG	0.717	0.685	0.789	0.673	0.721	0.794	0.877	0.756	0.838	0.942
NB+GFS	0.769	0.653	0.788	0.765	0.775	0.81	0.894	0.784	0.86	0.941
NB+GFS+BG	0.771	0.688	0.788	0.763	0.781	0.817	0.918	0.778	0.866	0.923
NB+GAFS	0.757	0.635	0.79	0.719	0.758	0.79	0.75	0.792	0.857	0.952
NB+FS	0.601	0.612	0.799	0.749	0.707	0.742	0.824	0.583	0.812	0.886
NB+BE	0.717	0.659	0.768	0.734	0.275	0.799	0.908	0.808	0.885	0.938

Tabel 12 Pairwise Differences

	NB	NB+BG	NB+GFS	NB+GFS+BG	NB+GAFS	NB+FS	NB+BE
NB	0	-0.750	-2.200	-2.700	-1.250	0.750	-1.200
NB+BG	0.750	0	-1.450	-1.950	-0.500	1.500	-0.450
NB+GFS	2.200	1.450	0	-0.500	0.950	2.950	1.000
NB+GFS+BG	2.700	1.950	0.500	0	1.450	3.450	1.500
NB+GAFS	1.250	0.500	-0.950	-1.450	0	2.000	0.050
NB+FS	-0.750	-1.500	-2.950	-3.450	-2.000	0	-1.950
NB+BE	1.200	0.450	-1.000	-1.500	-0.050	1.950	0

Critical difference: 2,8483

Pada Tabel 12 menunjukkan bahwa nilai NB+GFS dan NB+GFS+BG lebih besar dibandingkan nilai *Critical Difference* berarti model NB+GFS dan model NB+GFS+BG mempunyai perbedaan yang signifikan ketika dibandingkan dengan model NB+FS. Tabel 13 diketahui bahwa nilai p (*p-value*) lebih kecil dari nilai alpha sehingga ada perbedaan signifikan antara model. Pada Tabel 13 dapat dilihat bahwa model NB+GFS dan model NB+GFS+BG mempunyai perbedaan yang signifikan dengan model NB+FS sehingga hipotesa H₁ diterima dan kesimpulannya adalah model NB+GFS dan model NB+GFS+BG ada peningkatan ketika dibandingkan dengan model NB+FS. Model yang mempunyai perbedaan yang signifikan dan mempunyai peningkatan dibandingkan model lainnya ditandai dengan angka yang dibelakan. Pada Tabel 14 merupakan hasil perbedaan signifikansi antara model yang dibandingkan, model yang berisi Yes merupakan model yang mempunyai perbedaan signifikan dengan model lainnya, sedangkan model yang berisi No merupakan model yang tidak mempunyai perbedaan signifikan terhadap model lainnya.

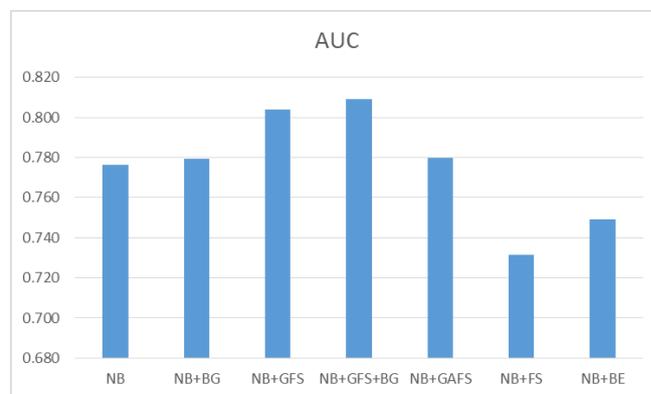
value) lebih kecil dari nilai alpha sehingga ada perbedaan signifikan antara model. Pada Tabel 13 dapat dilihat bahwa model NB+GFS dan model NB+GFS+BG mempunyai perbedaan yang signifikan dengan model NB+FS sehingga hipotesa H₁ diterima dan kesimpulannya adalah model NB+GFS dan model NB+GFS+BG ada peningkatan ketika dibandingkan dengan model NB+FS. Model yang mempunyai perbedaan yang signifikan dan mempunyai peningkatan dibandingkan model lainnya ditandai dengan angka yang dibelakan. Pada Tabel 14 merupakan hasil perbedaan signifikansi antara model yang dibandingkan, model yang berisi Yes merupakan model yang mempunyai perbedaan signifikan dengan model lainnya, sedangkan model yang berisi No merupakan model yang tidak mempunyai perbedaan signifikan terhadap model lainnya.

Tabel 13 Hasil P-Value Uji Friedman

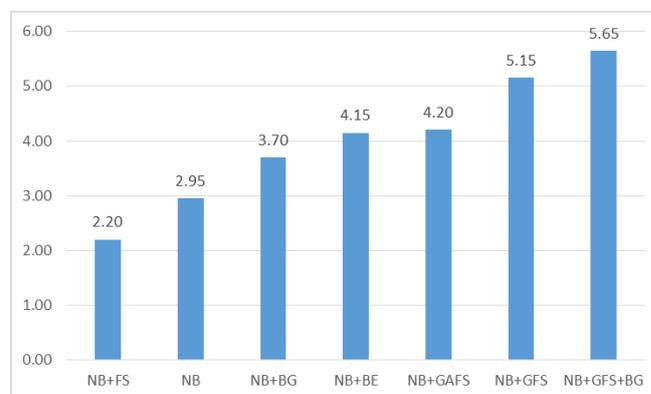
	NB	NB+BG	NB+GFS	NB+GFS+BG	NB+GAFS	NB+FS	NB+BE
NB	1	0.987	0.255	0.077	0.855	0.987	0.878
NB+BG	0.987	1	0.744	0.403	0.999	0.713	0.999
NB+GFS	0.255	0.744	1	0.999	0.958	0.037	0.946
NB+GFS+BG	0.077	0.403	0.999	1	0.744	0.007	0.713
NB+GAFS	0.855	0.999	0.958	0.744	1	0.371	1.000
NB+FS	0.987	0.713	0.037	0.007	0.371	1	0.403
NB+BE	0.878	0.999	0.946	0.713	1.000	0.403	1

Tabel 14 Perbedaan Signifikansi

	NB	NB+BG	NB+GFS	NB+GFS+BG	NB+GAFS	NB+FS	NB+BE
NB	No	No	No	No	No	No	No
NB+BG	No	No	No	No	No	No	No
NB+GFS	No	No	No	No	No	Yes	No
NB+GFS+BG	No	No	No	No	No	Yes	No
NB+GAFS	No	No	No	No	No	No	No
NB+FS	No	No	Yes	Yes	No	No	No
NB+BE	No	No	No	No	No	No	No



Gambar 6 Grafik AUC Mean Uji Friedman



Gambar 7 Grafik Rata-Rata Peringkat Uji Friedman

Pada Gambar 6 dapat dilihat bahwa model NB+GFS+BG merupakan model terbaik dalam penelitian ini dengan mempunyai rata rata AUC tertinggi dalam uji friedman

dan model NB+GFS merupakan model terbaik kedua yang diikuti oleh model NB+GAFS, NB+BG, NB, NB+BE dan NB+FS. Gambar 7 merupakan rata-rata peringkat dimana model NB+GFS+BG merupakan model terbaik yang menunjukkan bahwa seleksi fitur greedy forward selection dan bagging mampu meningkatkan kinerja model prediksi cacat *software* dengan memperbaiki model NB dan dapat menangani permasalahan ketidakseimbangan kelas (*imbalance class*) dan fitur-fitur yang tidak relevan (*irrelevant features*).

5. KESIMPULAN

Pengembangan *software* diperlukan agar menghasilkan *software* yang berkualitas. Kualitas *software* ditentukan dengan melakukan pemeriksaan dan pengujian, untuk menemukan cacat dalam *software* tersebut yang dapat menurunkan kualitas *software*. Dataset cacat *software* umumnya memiliki ketidakseimbangan kelas dan fitur-fitur yang tidak relevan yang dapat menyebabkan menurunnya kinerja dari algoritma pembelajaran. Salah satu algoritma pembelajaran yang digunakan dalam prediksi cacat *software* yaitu naïve bayes, yang merupakan algoritma terbaik untuk prediksi cacat atau tidaknya sebuah *software*. Pada penelitian ini mengusulkan model Naïve Bayes dan seleksi fitur Greedy Forward Selection (NB+GFS) untuk mengatasi permasalahan fitur-fitur yang tidak relevan, sedangkan untuk mengatasi permasalahan ketidakseimbangan kelas menggunakan model Naïve Bayes dengan Greedy Forward Selection dan Bagging (NB+GFS+BG). Hasil eksperimen dalam penelitian ini mendapatkan nilai AUC tertinggi pada model NB+GFS sebesar 0.941 dan untuk model NB+GFS+BG mendapatkan nilai AUC tertinggi sebesar 0.923 pada dataset PC5. Untuk menentukan model terbaik dalam penelitian prediksi cacat *software*, selanjutnya dilakukan friedman test. Hasil friedman test diketahui bahwa nilai rata-rata peringkat dengan tertinggi adalah model NB+GFS+BG, sehingga model NB+GFS+BG merupakan model terbaik dalam penelitian prediksi cacat *software*.

REFERENSI

- Arora, I., Tatarwal, V., & Saha, A. (2015). Open Issues in *Software Defect Prediction*. *Procedia Computer Science*, 906-912.
- Chang, R., Mu, X., & Zhang, L. (2011). *Software Defect Prediction Using Non-Negative Matrix Factorization*. *Journal of Software*, 2114-2120.
- Gao, K., Khoshgoftaar, T., & Wald, R. (2014). Combining Feature Selection and Ensemble Learning for *Software Quality Estimation*. *Twenty-Seventh International Florida Artificial Intelligence Research society Conference* (pp. 47-52). Association for the Advancement of Artificial Intelligence.
- Gorunescu, F. (2011). *Data Mining Concepts, Models and Techniques*. Berlin: Springer.
- Han, J., Kamber, M., & Pei, J. (2012). *Data Mining Concepts and Techniques*. Waltham: Elsevier.
- Khoshgoftaar, T. M., Hulse, J. V., & Napolitano, A. (2011). Comparing Boosting and Bagging Techniques with Noisy and Imbalanced Data. *IEEE Transactions on Systems, Man and Cybernetics*, 552-568.
- Laradji, I. H., Alshayeb, M., & Ghouti, L. (2015). *Software Defect Prediction Using Ensemble Learning on Selected Features*. *Information and Software Technology*, 388-402.
- Lee, C.-H. (2015). A Gradient Approach for Value Weighted Classification Learning in *Naive Bayes*. *Knowledge-Based Systems*, 1-9.
- 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. *Imbalanced Learning: Foundations, Algorithms, and Applications, First Edition*, 61-82.
- Ma, Y., Luo, G., Zeng, X., & Chen, A. (2012). Transfer Learning for Cross-Company *Software Defect Prediction*. *Information and Software Technology*, 248-256.
- Song, Q., Jia, Z., Shepperd, M., Ying, S., & Liu, J. (2010). A General *Software Defect-Proneness Prediction Framework*. *IEEE Transaction on Software Engineering*, 1-16.
- Strate, J. D., & Laplante, P. A. (2013). A Literature Review of Research in *Software Defect Reporting*. *IEEE Transactions on Reliability*, 444-454.
- Wahono, R. S. (2015). A Systematic Literature Review of *Software Defect Prediction: Research Trends, Datasets, Methods and Frameworks*. *Journal of Software Engineering*, 1-16.
- Wahono, R. S., & Suryana, N. (2013). Combining Particle Swarm Optimization based Feature Selection and Bagging Technique for *Software Defect*. *IJSEIA*, 153-166.
- Wahono, R. S., Suryana, N., & Ahmad, S. (2014). Metaheuristic Optimization based Feature Selection for *Software Defect Prediction*. *Journal of Software*, 1324-1333.
- Witten, I. H., Frank, E., & Hall, M. A. (2011). *Data Mining Practical Machine Learning Tools and techniques*. Burlington: Elsevier.

BIOGRAFI PENULIS



Fitriyani. Memperoleh gelar S.T pada bidang Sistem Informasi dari Universitas BSI Bandung dan gelar M.Kom pada bidang *software engineering* dari STMIK Nusa Mandiri. Staf pengajar di Universitas BSI Bandung. Minat penelitian saat ini meliputi *software engineering* 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.

Komparasi Metode *Machine Learning* dan *Non Machine Learning* untuk Estimasi Usaha Perangkat Lunak

Ega Kartika Adhitya, Romi Satria Wahono dan Hendro Subagyo
Fakultas Ilmu Komputer, Universitas Dian Nuwanto
 egaegi25gmail.com, romi@romisatriawahono.net, hendro.subagyo@gmail.com

Abstrak: Estimasi usaha perangkat lunak (EURL) adalah proses yang sangat penting dalam mewujudkan kesuksesan pelaksanaan suatu proyek perangkat lunak. Ada banyak metode EURL sehingga diperlukan pemilihan yang sesuai proyek yang akan dikembangkan agar menghasilkan estimasi yang akurat. Dataset yang sering digunakan untuk penelitian EURL adalah dataset Albercht dan Desherhanis, namun dataset tersebut memiliki atribut yang tidak relevan yang dapat mengurangi akurasi estimasi. Penelitian ini membandingkan metode machine learning (ML) dan non machine learning (non ML) untuk mengetahui metode mana yang paling baik untuk EURL serta penggunaan seleksi atribut *forward selection* (FS) dan *backward elimination* (BE) untuk menyelesaikan masalah atribut yang tidak relevan. ML terdiri dari algoritma *k-nearest neighbors* (kNN), *neural networks* (NN) dan *support vector machine* (SVM) sedangkan non ML terdiri dari metode *function point* (FP) dan *use case point* (UCP). Dari hasil eksperimen, kNN adalah metode terbaik pada ML dengan nilai RMSE 6.2 dan 9.4 untuk dataset Albercht dan Desherhanis. Sedangkan pada non ML, FP yang paling baik dengan nilai RMSE 4.6 dan 21.3 untuk dataset Albercht dan Desherhanis. Sementara hasil eksperimen berikutnya, kNN diintegrasikan dengan FS mendapat nilai RMSE 2.5 dan 6.4 untuk dataset Albercht dan Desherhanis lebih baik dari integrasi kNN dengan BE dengan nilai RMSE 7.5 baik untuk dataset Albercht maupun Desherhanis. Dari eksperimen, dapat disimpulkan bahwa metode ML lebih baik dari pada non ML dalam estimasi usaha perangkat lunak dengan kNN sebagai estimasinya. Integrasi kNN dengan FS juga terbukti mampu meningkatkan estimasi lebih baik daripada kNN standar maupun kNN dengan BE.

Kata kunci: estimasi usaha perangkat lunak, metode *machine learning*, metode *non-machine learning*, seleksi atribut

1 PENDAHULUAN

Perangkat lunak merupakan abstraksi fisik yang memungkinkan kita untuk berbicara dengan mesin perangkat keras (Dennis, 2012). Tanpa adanya perangkat lunak, maka perangkat keras yang telah diciptakan tidak akan dapat berguna atau berfungsi dengan optimal. Perangkat lunak tidak hanya berupa program komputer, tetapi juga berisi dokumentasi yang terkait dan data konfigurasi yang diperlukan untuk membuat sebuah program dapat beroperasi dengan benar.

Estimasi usaha perangkat lunak dibutuhkan karena dalam pengembangan perangkat lunak dibatasi oleh biaya dan jadwal yang telah ditentukan (Ian Sommerville, 2011) Salah satu kegiatan manajemen proyek perangkat lunak adalah kegiatan estimasi usaha perangkat lunak. Estimasi usaha perangkat lunak merupakan kegiatan memperkirakan berapa banyak sumber daya yang dibutuhkan untuk menyelesaikan sebuah rencana proyek (Ian Sommerville, 2011). Melakukan estimasi atau memperkirakan berapa jumlah tenaga dan waktu yang dibutuhkan untuk membuat sebuah sistem tidak mudah

dilakukan karena diperlukan pemahaman yang benar tentang sebuah pekerjaan.

Kesuksesan proyek pengembangan dipengaruhi oleh banyak hal, diantaranya dukungan eksekutif, keterlibatan pengguna dalam proyek, pengalaman manager proyek, tujuan bisnis yang jelas, infrastruktur perangkat lunak dan penggunaan metodologi pengembangan yang formal. Faktor lainnya merupakan faktor yang berkaitan dengan *timing* dan *scope* proyek, diantaranya *scope* yang minimal dan estimasi yang handal (Albrecht & Gaffney, 1983). Estimasi sumber daya, biaya, dan jadwal untuk pengembangan perangkat lunak membutuhkan pengalaman, akses ke informasi sejarah proyek yang baik, dan keberanian untuk melakukan prediksi kuantitatif sedangkan informasi kualitatif adalah semua fakta yang ada. Estimasi membawa risiko yang melekat dan risiko ini dapat menyebabkan ketidakpastian dalam proyek.

Estimasi usaha pengembangan perangkat lunak adalah proses melakukan estimasi usaha yang diperlukan untuk pengembangan perangkat lunak (Nunes, Constantine, & Kazman, 2011). Perusahaan-perusahaan pengembang perangkat lunak banyak yang dihadapkan dengan masalah estimasi biaya dan waktu yang diperlukan untuk proyek pengembangan perangkat lunak. Estimasi jadwal proyek sulit dilakukan karena banyak hal, misalnya perangkat lunak yang dikembangkan kemungkinan harus berjalan di lingkungan sistem yang sama sekali berbeda, menggunakan pengembangan dengan teknologi baru atau kompetensi orang-orang yang terlibat dalam proyek yang mungkin tidak diketahui. Ada begitu banyak ketidakpastian sehingga sulit untuk memperkirakan biaya dan waktu pengembangan secara akurat selama awal tahapan proyek. (Albrecht & Gaffney, 1983)

Dalam perkembangannya metode-metode untuk estimasi usaha pengembangan perangkat lunak, metode-metode tersebut di kelompokkan menjadi metode non machine learning dan machine learning (Shepperd & MacDonell, 2012), banyak metode estimasi telah diusulkan untuk estimasi usaha pengembangan perangkat lunak. Beberapa metode non-machine learning yang telah diusulkan dalam estimasi usaha perangkat lunak diantaranya *slim* (Adriano L I Oliveira, Braga, Lima, & Cornélio, 2010), *cocomo* (El-Sebakhy, 2011), *expert judgement* (Nassif, Capretz, & Hill, 1993), *function point* (FP) (Choy, Tang, & Tong, 2011) dan *use case point* (UCP) (Boehm & Papaccio, 1988).

Metode-metode non machine learning dalam estimasi usaha pengembangan perangkat lunak yang disebutkan sebelumnya merupakan metode konvensional yang memiliki tingkat keakuratan yang relatif rendah (Nunes, Constantine, & Kazman, 2011). Selain metode-metode non-machine learning di atas, belakangan ini dikembangkan estimasi dengan metode machine learning. Beberapa metode machine learning yang telah digunakan untuk estimasi usaha pengembangan perangkat lunak diantaranya adalah *case-based reasoning* (CBR) learning *k-nearest neighbor* (kNN) (Ian Sommerville,

2011), *neural networks* (NN), *support vector machine* (SVM) (Wen, Li, Lin, Hu, & Huang, 2012), *liner regression* (LR) (W. Wang & Zhou, 2012).

Namun dalam metode machine learning juga terdapat kekurangan pada tingkat keakuratan yang relatif rendah (Ian Sommerville, 2011) dataset yang digunakan diantaranya Cocomo, Desharnais, Maxwell dan Albercht, umumnya, estimasi usaha sangat sulit dilakukan dalam proyek perangkat lunak karena proyek perangkat lunak bersifat dinamis dan kesulitan menemukan proyek yang sangat mirip dengan proyek tersebut sebelumnya. Selain itu, sulit untuk menentukan metode estimasi yang cocok untuk proyek tersebut. Dataset *software effort* banyak memiliki atribut yang beragam. Misalnya dataset Albrecht memiliki 8 atribut, dataset desherhanis memiliki 8 atribut. Kinerja yang lebih baik dapat dicapai dengan menghilangkan beberapa atribut, yang dengan ini dapat menghilangkan atribut yang tidak relevan (Nassif, Capretz, & Ho, 2012). Seleksi atribut dapat digunakan untuk menentukan *subset of features* yang optimal, dimana hal ini dapat meningkatkan akurasi estimasi (S. Wang, Li, Song, Wei, & Li, 2011)

Seleksi atribut merupakan bagian penting untuk mengoptimalkan kinerja dari *classifier* (Mehmood, S. Palli, & Khan, 2014). Seleksi atribut dapat didasarkan pada pengurangan ruang fitur yang besar, misalnya dengan mengeliminasi atribut yang kurang relevan (Danger, Segura-Bedmar, Martínez, & Rosso, 2010). Penggunaan algoritma seleksi atribut yang tepat dapat meningkatkan *accuracy* Algoritma seleksi atribut dapat dibedakan menjadi dua tipe, yaitu *filter* dan *wrapper* (Danger et al., 2010). Contoh dari tipe *filter* adalah *information gain* (IG), *chi-square*, dan *log likelihood ratio*. Contoh dari tipe *wrapper* adalah *forward selection* dan *backward elimination* (Nassif et al., 2012). Hasil *precision* dari tipe *wrapper* lebih tinggi daripada tipe *filter*, tetapi hasil ini tercapai dengan tingkat kompleksitas yang besar. Masalah kompleksitas yang tinggi juga dapat menimbulkan masalah (W. Wang & Zhou, 2012).

Dari semua hasil penelitian yang sudah dilakukan belum ditemukan model yang paling tepat untuk estimasi usaha perangkat lunak. Maka dari itu akan dilakukan komparasi terhadap beberapa metode machine learning (kNN, SVM dan NN), dan metode non-machine learning (FP dan UCP), komparasi terhadap beberapa seleksi atribut (*forward selection*, *backward elimination*) dan melakukan integrasi dari hasil komparasi metode machine learning, non-machine learning dan seleksi atribut yang terbaik pada metode estimasi usaha perangkat lunak, sehingga di dapatkan model yang baik untuk estimasi usaha perangkat lunak.

2 PENELITIAN TERKAIT

Nassif (Nassif et al., 2012), melakukan penelitian tentang perbandingan akurasi yang dihasilkan oleh algoritma *neural network* dan *linear regression*, dan *linear regression* secara bersama-sama akan digunakan dengan *algorithmic* model, yaitu *use case point*, dataset yang digunakan Albercht dengan hasil MMR :LR: 39.2 dan NN: 40

Kocaguneli (Kocaguneli & Menzies, 2013) melakukan penelitian tentang perbandingan akurasi yang bertujuan untuk mengkarakterisasi isi penting dari data *software effort estimation* (SEE), yaitu, sedikitnya jumlah fitur dan contoh yang diperlukan untuk menangkap informasi dalam data yang ada. Jika isi penting sangat kecil, maka informasi yang terkandung harus sangat singkat dan nilai tambah skema pembelajaran yang kompleks harus minimal, dataset yang digunakan cocomo dengan hasil RMSE kNN : 41.42

Adriano (Adriano L.I. Oliveira, Braga, Lima, & Cornélio, 2010), melakukan penelitian tentang analisis industri perangkat lunak, manajer proyek biasanya mengandalkan pengalaman mereka sebelumnya untuk memperkirakan jumlah atau waktu yang dibutuhkan untuk setiap proyek perangkat lunak. Keakuratan perkiraan tersebut merupakan faktor kunci untuk aplikasi yang efisien bagi sumber daya manusia. Metode yang digunakan yaitu teknik pembelajaran mesin seperti fungsi radial basis (RBF) jaringan saraf, *multi-layer perceptron* (MLP) jaringan saraf, regresi dukungan vektor (SVR), mengantongi prediktor dan pohon regresi berbasis, untuk memperkirakan perangkat lunak. Beberapa karya telah menunjukkan bahwa tingkat akurasi estimasi usaha software sangat tergantung pada nilai-nilai parameter metode ini. Selain itu, telah ditunjukkan bahwa pemilihan fitur masukan juga mungkin memiliki pengaruh penting pada akurasi estimasi, dataset yang digunakan desherhanis dengan hasil RMSE SVM 83.33

Nuno (Nunes et al., 2011a), telah mengusulkan beberapa metode pengukuran dengan ukuran function al dan biaya model estimasi, terutama pada poin fungsi analisis (FPA) dan Cocomo. Keduanya menganggap bahwa pengembang dapat memperoleh pengukuran ukuran dan perkiraan dari sejarah data proyek dan karakteristik proyek saat ini. Dengan orientasi objek, menggunakan kasus yang muncul sebagai sebuah teknik yang dominan untuk persyaratan penataan. Teknik ini telah dikomparasikan ke dalam *unified modeling language* (UML) dan *unified process* dan menjadi standar sesungguhnya untuk persyaratan peragaan, dataset yang digunakan privat dengan hasil UCP 85.33

Felfernig (Dubois, Rasovska, & De Guio, 2009), melakukan fungsi point analysis (FPA) untuk berorientasi objek, pengembangan estimasi usaha perangkat lunak. Dalam makalah ini menggunakan fungsi point analysis (FPA) (Papatheocharous & Andreou, 2009) dapat diterapkan untuk estimasi upaya pengembangan sistem konfigurasi berbasis pengetahuan. FPA didasarkan pada user dan berpusat tampilan pada perangkat lunak dan *platform-independen*. Metode tersebut pertama kali diusulkan oleh Alberth (Albrecht & Gaffney, 1983) dengan tujuan untuk memberikan upaya untuk ukuran perangkat lunak bersama dengan aturan penghitungan telah diadaptasi beberapa kali, dataset yang digunakan albrecht dengan hasil FP 73.33

Seleksi fitur, baru-baru ini digunakan di bidang rekayasa perangkat lunak untuk meningkatkan akurasi, model biaya perangkat lunak. Ide di balik memilih subset paling informatif fitur yang tersedia berasal dari hipotesis bahwa mengurangi dimensi dataset secara signifikan akan mengurangi kompleksitas dan waktu yang diperlukan untuk mencapai perkiraan menggunakan teknik pemodelan tertentu.

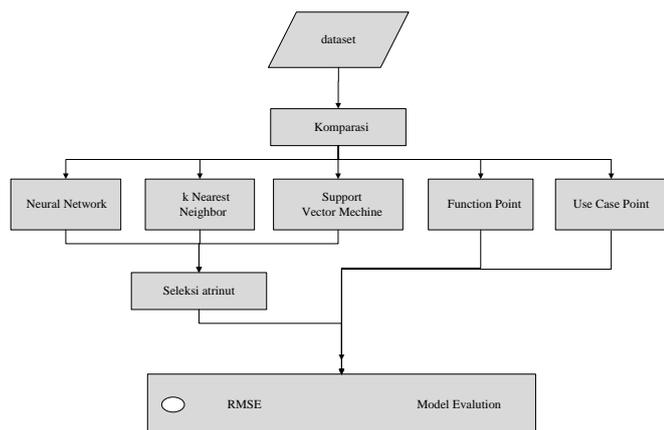
3 METODE YANG DIUSULKAN

Peneliti mengusulkan untuk mengkomparasi metode machine learning (kNN, SVM, NN) dan non machine learning (FP dan UCP) dan menambahkan algoritma seleksi atribut (Forward Selection dan Backward Elimination). Gambar 1 menunjukkan komparasi algoritma klasifikasi dan seleksi atribut yang diusulkan

Model yang diusulkan dalam penelitian ini mulai dari pengolahan dataset hingga menghasilkan model dan selanjutnya model akan diuji dengan menggunakan dataset testing. Dataset akan dilakukan pengujian dengan menggunakan algoritma berbeda secara bergantian. Hasil dari pengujian ini akan dilakukan komparasi algoritma mana yang

paling akurat. Langkah-langkah tersebut akan dilakukan sebagai berikut:

- 1 Dataset untuk komparasi dan Model yang terbentuk akan langsung diuji dengan dataset testing yang terbentuk, dan nilai akurasi model akan dirata-ratakan.
- 2 Membandingkan hasil performa antara algoritma *support vector machine*, *k- nearest neighbor*, *neural network*, *function point* dan *use case point*. Hasil RMSE metode machine learning dan metode non-machine learning ini akan dibandingkan. Nilai RMSE yang paling rendah merupakan metode paling akurat.
- 3 Dilanjutkan dengan menambahkan seleksi atribut pada metode yang paling akurat.



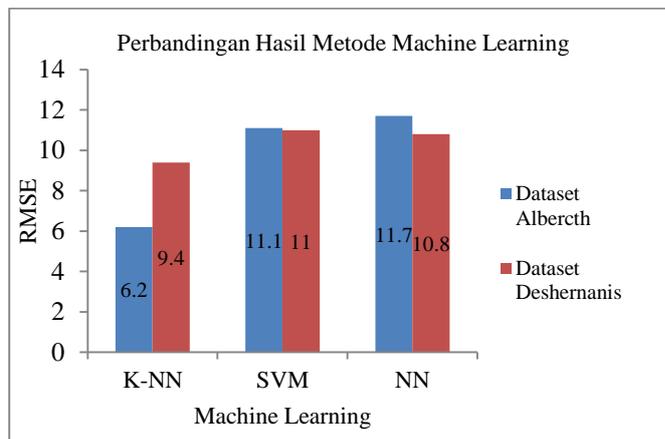
Gambar 1. Komparasi metode machine learning dan non machine learning serta seleksi atribut.

4 HASIL EKSPERIMEN

Penelitian yang dilakukan menggunakan komputer dengan sistem operasi Microsoft Windows 7 Professional 64-bit. Aplikasi yang digunakan adalah RapidMiner 5.2. Data penelitian ini menggunakan dataset Albercth dan dataset Desherhanis.

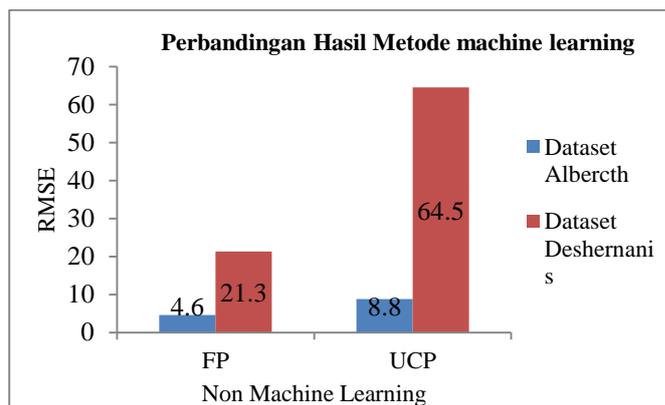
Pada penelitian pertama yang menggunakan metode machine learning kNN, NN dan SVM, dapat dilihat hasil komparasi RMSE pada Gambar 2. Dari Gambar 2 dapat kita ketahui bahwa K-NN mempunyai nilai RSME yang paling baik, dengan nilai RMSE 6.2 untuk dataset Albercth dan 9.4 untuk dataset Desherhanis. NN menghasilkan RMSE 11.7 poin untuk dataset Albercth dan 10.8 untuk dataset Desherhanis . SVM menghasilkan RMSE 11.1 untuk dataset Albercth dan 11 untuk dataset Desherhanis. Dari hasil tersebut, maka dapat kita simpulkan kNN merupakan metode machine learning yang baik. Gambar 2 merupakan grafik komparasi RMSE metode machine learning.

Dari penelitian dari Wen (Wen, Li, Lin, Hu, & Huang, 2012a), Nassif (Nassif, Capretz, & Ho, 2011), dan Kocaguneli (Kocaguneli & Menzies, 2013), juga melakukan penelitian dihasilkan hal yang berbeda hal ini dikarenakan beberapa faktor yang yang mempengaruhi dari hasil tersebut salah satu peneliti Wen (Wen et al., 2012), juga mendapatkan hasil untuk nilai estimasi usaha perangkat lunak untuk metode machine learning adalah kNN.



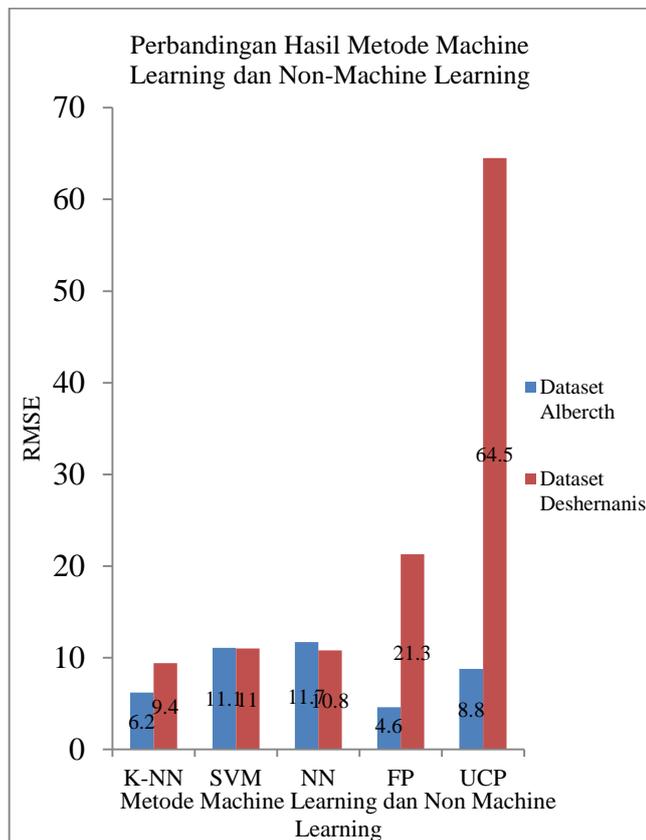
Gambar 2 Grafik Komparasi RMSE Metode Machine Learning

Pada penelitian kedua yang menggunakan metode non-machine learning FP dan UCP, dapat dilihat hasil komparasi RMSE pada Gambar 3. Dari Gmabar 3 dapat ketahui bahwa FP mempunyai nilai RSME yang paling baik, dengan nilai RMSE 4.6 poin untuk dataset Albercth dan 21.3 poin untuk dataset Desherhanis. UCP menghasilkan RMSE 8.8 poin untuk dataset Albercth dan 64.3 poin untuk dataset Desherhanis. Dari hasil tersebut, maka dapat kita simpulkan FP merupakan metode non-machine learning yang baik. Gambar 3 merupakan grafik komparasi RMSE metode non- machine learning.



Gambar 3. Grafik Komparasi RMSE Metode Non-Machine Learning

Penelitian ketiga adalah komparasi hasil metode machine learning dan non-machine learning terbaik. Pada penelitian pertama, dikomparasikan tiga metode machine learning yaitu kNN, NN dan SVM. Didapatkan hasil terbaik adalah kNN. Pada penelitian kedua, dikomparasikan dua metode non machine learning yaitu FP dan UCP. Didapatkan hasil terbaik adalah kNN. Maka pada penelitian ketiga ini dikomparasikan antara kNN dan FP dan menghasilkan kNN sebagai metode terbaik untuk estimasi perangkat lunak. Gambar 4 merupakan hasil komparasi metode machine learning dan non machine learning terbaik untuk estimasi perangkat lunak



Gambar 4. Perbandingan Hasil Metode Machine Learning dan Non-Machine Learning

Penelitian keempat dapat dilihat pada Tabel 1 merupakan tabel pengujian seleksi atribut terbaik, dapat dianalisa untuk seleksi atribut terbaik untuk setiap algoritma klasifikasi. Untuk kNN, lebih baik menggunakan *feature* backward elimination dan forward selection.

Metode seleksi atribut backward elimination dan forward selection merupakan alternative untuk mengurangi kemungkinan adanya multikolinearitas dalam model yang dihasilkan. Prosedur ini tidak selalu mengarahkan ke model yang terbaik, mengingat kita hanya mempertimbangkan sebuah subset kecil dari semua model-model yang mungkin. Sehingga resiko melewatkan atau kehilangan model terbaik akan bertambah seiring dengan penambahan jumlah variabel bebas.

Tabel 1. Pengujian Seleksi Aribut Untuk kNN

Seleksi Atribut Untuk K-NN		
	Backward Elimination	Forward Selection
Dataset Albercth	7.5	2.5
Dataset Deshernanis	7.5	6.4

Berdasarkan metode seleksi atribut dapat dibedakan menjadi dua tipe, yaitu *filter* dan *wrapper* [17]. Contoh dari tipe *filter* adalah *information gain* (IG), *chi-square*, dan *log likelihood ratio*. Contoh dari tipe *wrapper* adalah *forward selection* dan *backward elimination* [18]. Hasil akurasi dari tipe *wrapper* lebih tinggi daripada tipe *filter*, tetapi hasil ini tercapai dengan tingkat kompleksitas yang besar. Tabel 3 dan hasil penelitian dapat disimpulkan bahwa seleksi atribut FS mendapatkan hasil yang terbaik. Dengan menggunakan kNN untuk dua dataset baik dataset Albercth dan dataset Deshernanis, algoritma

seleksi atribut FS mendapatkan hasil yang paling baik untuk digunakan pada estimasi usaha perangkat lunak.

5 KESIMPULAN

SVM didapatkan kNN dengan hasil terbaik dengan nilai RMSE yang paling baik, dengan nilai RMSE 6.2 untuk dataset Albercth dan 9.4 untuk dataset Deshernanis. Hasil dari metode non ML antara FP dan UCP didapatkan FP mempunyai nilai RSME yang paling baik, dengan nilai RMSE 4.6 untuk dataset Albercth dan 21.3 untuk dataset Deshernanis. Dapat disimpulkan hasil dari komparasi antara kNN dan FP dan menghasilkan kNN sebagai metode terbaik untuk estimasi perangkat lunak.

Seleksi atribut FS mendapatkan hasil yang terbaik. Dengan menggunakan kNN untuk dua dataset baik dataset Albercth dan dataset Deshernanis, algoritma Seleksi Atribut FS mendapatkan hasil yang paling baik untuk digunakan pada estimasi usaha perangkat lunak.menghasilkan kNN dengan Seleksi Atribut FS sebagai metode terbaik untuk estimasi perangkat lunak.

REFERENSI

Albrecht, A. J., & Gaffney, J. E. (1983). Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation. *IEEE Transactions on Software Engineering*, SE-9(6), 639–648. doi:10.1109/TSE.1983.235271

Boehm, B. W., & Papaccio, P. N. (1988). Understanding and controlling software costs. *IEEE Transactions on Software Engineering*. doi:10.1109/32.6191

Choy, S. K., Tang, M. L., & Tong, C. S. (2011). Image segmentation using fuzzy region competition and spatial/frequency information. *IEEE Transactions on Image Processing: A Publication of the IEEE Signal Processing Society*, 20(6), 1473–84. doi:10.1109/TIP.2010.2095023

Danger, R., Segura-Bedmar, I., Martínez, P., & Rosso, P. (2010). A comparison of machine learning techniques for detection of drug target articles. *Journal of Biomedical Informatics*, 43(6), 902–13. doi:10.1016/j.jbi.2010.07.010

Dennis, A. (2012). *Systems Analysis and Design: An Applied Approach*. Wiley; 5 edition (January 18, 2012). Retrieved from <http://www.philadelphia.edu.jo/it/cs/syllabus/731332.pdf>

Dubois, S., Rasovska, I., & De Guio, R. (2009). Towards an automatic extraction of Generalized System of Contradictions out of solutionless Design of Experiments. In *3rd IFIP Working Conference on Computer Aided Innovation (CAI): Growth and Development of CAI*. doi:10.1007/978-3-642-03346-9

El-Sebakhy, E. a. (2011). Functional networks as a novel data mining paradigm in forecasting software development efforts. *Expert Systems with Applications*, 38(3), 2187–2194. doi:10.1016/j.eswa.2010.08.005

Ian Sommerville. (2011). *Software engineering 9. New York 1992*. Addison-Wesley; 9 edition (March 13, 2010). doi:10.1109/MC.1987.1663532.

Kocaguneli, E., & Menzies, T. (2013). Software effort models should be assessed via leave-one-out validation. *Journal of Systems and Software*, 86(7), 1879–1890. doi:10.1016/j.jss.2013.02.053

Mehmood, A., S. Palli, A., & Khan, M. N. A. (2014). A Study of Sentiment and Trend Analysis Techniques for Social Media Content. *International Journal of Modern Education and Computer Science*, 6(December), 47–54. doi:10.5815/ijmecs.2014.12.07

Nassif, A. B., Capretz, L. F., & Hill, R. (1993). A Regression Model with Mamdani Fuzzy Inference System for Early Software Effort Estimation Based on Use Case Diagrams, 615–620.

Nassif, A. B., Capretz, L. F., & Ho, D. (2011). Estimating Software Effort Based on Use Case Point Model Using Sugeno Fuzzy

Inference System. *2011 IEEE 23rd International Conference on Tools with Artificial Intelligence*, 393–398. doi:10.1109/ICTAI.2011.64

Nassif, A. B., Capretz, L. F., & Ho, D. (2012). Estimating Software Effort Using an ANN Model Based on Use Case Points. *2012 11th International Conference on Machine Learning and Applications*, 7, 42–47. doi:10.1109/ICMLA.2012.138

Nunes, N., Constantine, L., & Kazman, R. (2011a). IUCP: Estimating interactive-software project size with enhanced use-case points. *IEEE Software*, 28, 64–73. doi:10.1109/MS.2010.111

Nunes, N., Constantine, L., & Kazman, R. (2011b). IUCP: Estimating interactive-software project size with enhanced use-case points. *IEEE Software*. doi:10.1109/MS.2010.111

Oliveira, A. L. I., Braga, P. L., Lima, R. M. F., & Cornélio, M. L. (2010). GA-based method for feature selection and parameters optimization for machine learning regression applied to software effort estimation. *Information and Software Technology*, 52(11), 1155–1166. doi:10.1016/j.infsof.2010.05.009

Oliveira, A. L. I., Braga, P. L., Lima, R. M. F., & Cornélio, M. L. (2010). GA-based method for feature selection and parameters optimization for machine learning regression applied to software effort estimation. *Information and Software Technology*, 52(11), 1155–1166. doi:10.1016/j.infsof.2010.05.009

Papatheocharous, E., & Andreou, A. S. (2009). Hybrid Computational Models for Software Cost Prediction: An Approach Using Artificial Neural Networks and Genetic Algorithms, 87–100.

Shepperd, M., & MacDonell, S. (2012, August). Evaluating prediction systems in software project estimation. *Information and Software Technology*. Elsevier B.V. doi:10.1016/j.infsof.2011.12.008

Wang, S., Li, D., Song, X., Wei, Y., & Li, H. (2011). A feature selection method based on improved fisher's discriminant ratio for text sentiment classification. *Expert Systems with Applications*, 38(7), 8696–8702. doi:10.1016/j.eswa.2011.01.077

Wang, W., & Zhou, Z. H. (2012). Learnability of multi-instance multi-label learning. *Chinese Science Bulletin*, 57, 2488–2491. doi:10.1007/s11434-012-5133-z



Hendro Subagyo. Menyelesaikan program S1 (B.Eng) dan S2 (M.Eng) pada jurusan Ilmu Komputer dan Informasi Matematik di The University of Electro-Communications, Tokyo, Jepang pada tahun 1999 dan 2001. Di Indonesia berstatus sebagai peneliti di Pusat Dokumentasi Informasi Ilmiah. Lembaga Ilmu Pengetahuan. Memiliki minat pada sistem operasi, pemrograman dan bahasa pemrograman (khususnya Java dan Real-Time Java) dan komputer aritmatika. Tema penelitian saat ini adalah software engineering dan ubiquitous computing.

BIOGRAFI PENULIS



Ega Kartika Adhitya. Menyelesaikan pendidikan S1 Perikanan di Universitas Diponegoro, Semarang, S2 Magister Teknik Informatika di Universitas Dian Nuswantoro Semarang. Saat ini menjadi Mahasiswa di Semarang. Minat penelitian saat ini adalah software engineering.



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 asosiasi ilmiah ACM, PMI dan IEEE Computer Society.

Integrasi Pareto Fitness, Multiple-Population dan Temporary Population pada Algoritma Genetika untuk Pembangkitan Data Test pada Pengujian Perangkat Lunak

Mohammad Reza Maulana, Romi Satria Wahono dan Catur Supriyanto

Fakultas Ilmu Komputer, Universitas Dian Nuswantoro

reza.stmikwp@gmail.com, romi@romisatriawahono.net, catur@research.dinus.ac.id

Abstrak: Pengujian perangkat lunak memerlukan biaya yang mahal dan sering kali lebih dari 50% biaya keseluruhan dalam pengembangan perangkat lunak digunakan dalam tahapan ini. Untuk mengurangi biaya proses pengujian perangkat lunak secara otomatis dapat digunakan. Hal yang sangat penting dalam pengujian perangkat lunak secara otomatis adalah proses menghasilkan data tes. Pengujian secara otomatis yang paling efektif dalam menekan biaya adalah pengujian *branch coverage*. Salah satu metode yang banyak digunakan dan memiliki kinerja baik adalah algoritma genetika (AG). Salah satu permasalahan AG dalam menghasilkan data tes adalah ketiga target cabang dipilih memungkinkan tidak ada satupun individu yang memenuhi kriteria. Hal ini akan menyebabkan proses pencarian data tes memakan waktu lebih lama. Oleh karena itu di dalam penelitian ini diusulkan integrasi *pareto fitness*, *multiple-population* dan *temporary population* di dalam proses pencarian data tes dengan menggunakan AG (AG-PFMPTP). *Multiple-population* diusulkan untuk menghindari *premature convergence*. Kemudian *pareto fitness* dan *temporary population* digunakan untuk mencari beberapa data tes sekaligus, kemudian mengevaluasinya dan memasukkan ke dalam *archive temporary population*. Dari hasil pengujian yang telah dilakukan rata-rata generasi metode AG-PFMPTP secara signifikan lebih sedikit dalam menghasilkan data tes yang dibutuhkan dibandingkan metode AG standar ataupun AG dengan *multiple-population* (AG-MP) pada semua benchmark program yang digunakan. Hal tersebut menunjukkan metode yang diusulkan lebih cepat dalam mencari data tes yang dibutuhkan.

Kata Kunci: pengujian perangkat lunak, pembangkitan data tes, algoritma genetika

1 PENDAHULUAN

Pengujian perangkat lunak merupakan bagian yang tidak terpisahkan dari rekayasa perangkat lunak (Anand et al., 2015). Pengujian perangkat lunak merupakan tahap yang sangat penting dalam siklus pengembangan perangkat lunak yang bertujuan untuk memastikan tingkat kualitas perangkat lunak yang dihasilkan pada tingkatan tertentu (Ferrer, Kruse, Chicano, & Alba, 2015). Proses pengujian perangkat lunak memerlukan biaya yang mahal (Anand et al., 2015)(P McMinn, 2004)(Alakeel, 2014), sering kali lebih dari 50% biaya total proses pengembangan perangkat lunak digunakan dalam proses ini (Anand et al., 2015). Untuk mengatasi hal tersebut pengujian perangkat lunak secara otomatis dapat dilakukan untuk mengurangi biaya (Anand et al., 2015)(Ferrer et al., 2015)(Díaz, Tuya, & Blanco, 2003)(Hermadi, Lokan, & Sarker, 2014), dan mendapatkan hasil pengujian yang lebih meyakinkan (Díaz et al., 2003)(Hermadi et al., 2014).

Dalam pengujian perangkat lunak secara otomatis, pembangkitan data tes merupakan hal yang paling utama (P McMinn, 2004)(Lakhotia, McMinn, & Harman, 2009)(Mao,

2014)(Mao, Xiao, Yu, & Chen, 2015)(Ribeiro, 2008). Pembangkitan data tes adalah proses identifikasi serangkaian uji kasus untuk memenuhi kecukupan data tes yang dipilih (Pachauri & Srivastava, 2013). Pembangkitan data tes menguraikan proses pencarian data tes terbaik untuk kriteria tes tertentu.

Terdapat beberapa pendekatan yang digunakan dalam membangkitkan data tes pada pengujian perangkat lunak secara otomatis, diantaranya pengujian white-box (pengujian struktural) (Mao, 2014)(Mao et al., 2015)(Phil McMinn, Harman, Lakhotia, Hassoun, & Wegener, 2012) dan pengujian black-box (pengujian fungsional) (Ferrer et al., 2015)(Vos et al., 2013). Dibandingkan pengujian fungsional, pengujian struktural memegang peranan penting karena lebih efektif dalam mengurangi biaya dalam proses pengujian perangkat lunak (Mao et al., 2015).

Saat ini banyak metode yang digunakan untuk membangkitkan data tes. Metode pencarian berbasis Metaheuristik atau yang dikenal dengan Search-Based Software Testing (SBST) banyak digunakan dalam penelitian berdasarkan *survey* yang telah dilakukan oleh McMinn (Phil McMinn, 2011). Metode ini dapat membangkitkan data tes dengan hasil cakupan yang tinggi dan mempunyai kemampuan dalam memperlihatkan kesalahan program yang diuji (Mao et al., 2015). Selain itu berdasarkan *survey* yang dilakukan Ali et al. (Ali, Briand, Hemmati, & Panesar-Walawege, 2010) menyatakan bahwa SBST lebih unggul dalam aspek efektifitas biaya dibandingkan dengan pembangkitan data tes secara tradisional. Meskipun memiliki banyak keunggulan, SBST mempunyai tantangan dalam penentuan *fitness function* yang akan berpengaruh pada efek cakupan hasil dan pencarian yang lebih efisien (Mao et al., 2015). *fitness function* digunakan dalam menyesuaikan arah pencarian data tes dan membangun hasil cakupan.

Beberapa algoritma SBST yang digunakan dalam membangkitkan data tes diantaranya simulated annealing (SA) (Cohen, Colbourn, & Ling, 2003)(Patil & Nikumbh, 2012), ant colony optimization (ACO) (Mao et al., 2015), dan algoritma genetika (AG) (Pachauri & Srivastava, 2013)(Alshraideh, Mahafzah, & Al-Sharaeh, 2011)(Praveen Ranjan Srivastava & Kim, 2009)(Miller, Reformat, & Zhang, 2006)(Aleti & Grunske, 2015)(Yao & Gong, 2014). Penggunaan algoritma SA mempunyai kelebihan dalam memecahkan optimasi pada masalah program yang kompleks (Mao et al., 2015). Walaupun SA memiliki kelebihan, algoritma tersebut memiliki kelemahan dalam hal kecepatan konvergensi (Mao, 2014). Srivastava et al. (P R Srivastava & Baby, 2010) mengadopsi algoritma ACO untuk menghasilkan urutan tes berdasarkan pengujian perangkat lunak, hal ini digunakan untuk memberikan hasil cakupan pengujian yang tinggi. Akan tetapi ACO memiliki kelemahan seperti pada algoritma SA, ACO memerlukan waktu yang lama dalam menghasilkan konvergensi. Algoritma genetika mempunyai kelebihan dalam menangani program dengan skala besar dan efektif dalam melakukan pencarian dibandingkan dengan metode pencarian

yang lain (Phil McMinn, 2011)(Harman & McMinn, 2010)(Yao & Gong, 2014).

Di dalam banyak penelitian yang dirangkum oleh Bueno (Bueno, Jino, & Wong, 2014), algoritma genetika digunakan untuk menghasilkan data tes untuk pengujian *statement coverage*, *path coverage*, *fault based coverage*, dan *branch coverage*. Diantara jenis pengujian tersebut, *branch coverage* merupakan pengujian yang paling efektif dalam menekan biaya dalam proses pengujian (Mao, 2014). *Branch coverage* merupakan pengujian yang dilakukan untuk menguji kondisi percabangan di dalam *source code* program. Salah satu permasalahan yang ada dalam menghasilkan data test untuk menguji setiap kondisi percabangan di dalam program dengan metode algoritma genetika adalah ketika cabang dipilih untuk target cakupan pengujian memungkinkan tidak ada satupun individu (individu mengkodekan data tes masukan) di dalam populasi yang memenuhi kriteria (Pachauri & Srivastava, 2013)(Alshraideh et al., 2011)(Miller et al., 2006). Kriteria dalam hal ini adalah hasil data tes yang didapatkan tidak dapat digunakan untuk mengeksekusi target cabang yang ditentukan. Apabila cabang tersebut tidak terlewati hal ini menyebabkan *source code* yang ada di dalam cabang tersebut tidak dapat dieksekusi untuk diuji.

Pada penelitian yang dilakukan (Alshraideh et al., 2011) mengusulkan model *multi-population* dalam melakukan pencarian beberapa target cakupan pengujian secara bersamaan. Dalam setiap tahap pencarian, tidak hanya satu kandidat target yang dicari, melainkan beberapa target sekaligus. Hal ini dilakukan untuk menghindari optimal lokal jika hanya satu target yang dicari. *Multi-population* juga akan mencegah terjadinya *premature convergence* yang akan mengakibatkan hasil pencarian tidak bisa beragam (Cantu-paz, 1997). Metode yang diimplementasikan menunjukkan pencarian terhadap target menjadi lebih singkat, jumlah eksekusi yang diperlukan untuk cakupan target lebih banyak, dan lebih efektif dalam proses pencarian jika dibandingkan dengan yang hanya menggunakan *single-population* (Alshraideh et al., 2011). Meskipun demikian *multiple-population* belum banyak berkontribusi terhadap tercapainya pemenuhan target yang dapat tereksekusi.

Pada penelitian lain yang dilakukan (Pachauri & Srivastava, 2013) mengusulkan model perbaikan proses pencarian dengan cara mengurutkan target yang akan dieksekusi. Selain mengurutkan target, peneliti mengusulkan teknik penyimpanan beberapa individu terbaik di dalam *memory* yang pada iterasi berikutnya akan menggantikan individu yang paling buruk. Penggunaan elitism juga diimplementasikan oleh peneliti untuk membandingkan dan menggantikan sebagian atau keseluruhan individu di dalam populasi yang dihasilkan untuk mendapatkan populasi terbaik. Hasil dari metode yang diusulkan menunjukkan peningkatan pencarian target yang lebih baik. Walaupun metode yang diusulkan memberikan hasil yang baik, masih terdapat kemungkinan untuk diperluas dan ditingkatkan performa dalam menghasilkan data tes.

Seperti yang sudah dijelaskan sebelumnya *fitness function* memegang peranan penting dalam proses pencarian target (Mao et al., 2015). *Fitness function* yang lebih baik memungkinkan proses pencarian target akan semakin lebih baik. Terdapat konsep penghitungan nilai fitness pada kasus *multiobjective* di dalam proses pencarian di dalam algoritma genetika yang dinamakan *pareto fitness*. Hasil penelitian menunjukkan perbaikan *fitness function* tersebut dapat meningkatkan kinerja algoritma genetika (Elaoud, Loukil, & Teghem, 2007)(Ferrer, Chicano, & Alba, 2012).

Berdasarkan hal tersebut skema penggabungan *multi-population* dan *pareto fitness* akan diusulkan pada penelitian ini. Di dalam penelitian ini peneliti juga mengusulkan metode *temporary population* yang nantinya digunakan dalam penyimpanan beberapa populasi terbaik untuk beberapa target cabang sekaligus. Diharapkan dengan penggabungan metode tersebut dapat meningkatkan kinerja algoritma genetika dalam proses pencarian dan pemenuhan target pengujian akan menjadi lebih baik.

Paper ini disusun dengan urutan sebagai berikut. Pada bagian 2, penelitian terkait akan dijelaskan. Pada bagian 3 metode yang diusulkan akan dipaparkan. Selanjutnya bagian 4 akan disajikan perbandingan hasil eksperimen metode yang diusulkan dengan metode yang lain. Kemudian pada bagian terakhir akan disampaikan kesimpulan dari penelitian yang dilakukan.

2 PENELITIAN TERKAIT

Penelitian pembangkitan data tes menggunakan metode algoritma genetika sudah banyak dilakukan. Banyak peneliti menggunakan metode algoritma genetika karena kemampuan dari metode ini lebih baik dibandingkan dengan metode lain. Pachauri & Srivastava (Pachauri & Srivastava, 2013) mengusulkan model pencarian data tes menggunakan algoritma genetika dengan memanfaatkan metode pengurutan *branch* yang akan dicari, kemudian mengimplementasikan *memory* dan *elitism*. Metode perbaikan yang diusulkan adalah dalam proses pencarian data tes, yaitu dengan cara mengurutkan target yang akan dieksekusi. Selain mengurutkan target, peneliti mengusulkan teknik penyimpanan beberapa individu terbaik di dalam *memory* yang pada iterasi berikutnya akan menggantikan individu yang paling buruk. Penggunaan elitism juga diimplementasikan oleh peneliti untuk membandingkan dan menggantikan sebagian atau keseluruhan individu di dalam populasi yang dihasilkan untuk mendapatkan populasi terbaik. Hasil dari metode yang diusulkan menunjukkan peningkatan pencarian target yang lebih baik. Walaupun metode yang diusulkan memberikan hasil yang baik, masih terdapat kemungkinan untuk diperluas dan ditingkatkan performa dalam menghasilkan data tes.

Alshraideh et al. mengusulkan proses pembangkitan data tes menggunakan *multiple-population* pada algoritma genetika (Alshraideh et al., 2011). Dalam setiap tahap pencarian, tidak hanya satu kandidat target yang dicari, melainkan beberapa target sekaligus. Hal ini dilakukan untuk menghindari optimal lokal jika hanya satu target yang dicari. *Multi-population* juga akan mencegah terjadinya *premature convergence* yang akan mengakibatkan hasil pencarian tidak bisa beragam. Metode yang diimplementasikan menunjukkan pencarian terhadap target menjadi lebih singkat, jumlah eksekusi yang diperlukan untuk cakupan target lebih banyak, dan lebih efektif dalam proses pencarian jika dibandingkan dengan yang hanya menggunakan *single-population*. Begitupun Yao et al. (Yao & Gong, 2014) mengusulkan pembangkitan data tes menggunakan algoritma genetika dengan memanfaatkan *individual sharing* di dalam *multiple-population*. Metode yang diusulkan Yao et al. berbeda dengan *multiple-population* tradisional. Tujuan utama dari strategi yang diusulkan adalah untuk memperluas jangkauan pencarian setiap populasi dengan berbagi individu, sehingga dapat meningkatkan efisiensi algoritma.

Ferrer et al. mengusulkan proses pembangkitan data tes menggunakan konsep *Multiobjective Evolutionary based on Pareto Efficiency* dengan menggunakan metode *Multi-*

Objective Cellular algoritma genetika (MOCeCell) (Ferrer et al., 2012). Dalam pendekatan metode yang diusulkan, solusi untuk masalah ini adalah tes *suite*, yaitu, satu set data tes. Tes *suite* dievaluasi sesuai dengan dua tujuan. Tujuan pertama mengevaluasi cakupan yang diperlukan, secara umum, mengeksekusi tes *suite* kedalam program yang diuji. Tujuan kedua adalah evaluasi hitungan sederhana dari jumlah data tes di dalam satu set tes *suite*. Jumlah test data yang sedikit dan mencakup semua *test suite* yang dibutuhkan akan menjadikan pengujian lebih efektif.

Di dalam penelitian ini, peneliti akan mengintegrasikan konsep *pareto fitness*, *multiple population* dan *temporary population* untuk meningkatkan kinerja algoritma genetika dalam melakukan pencarian data tes yang hasilnya akan digunakan untuk melakukan pengujian perangkat lunak.

3 METODE YANG DIUSULKAN

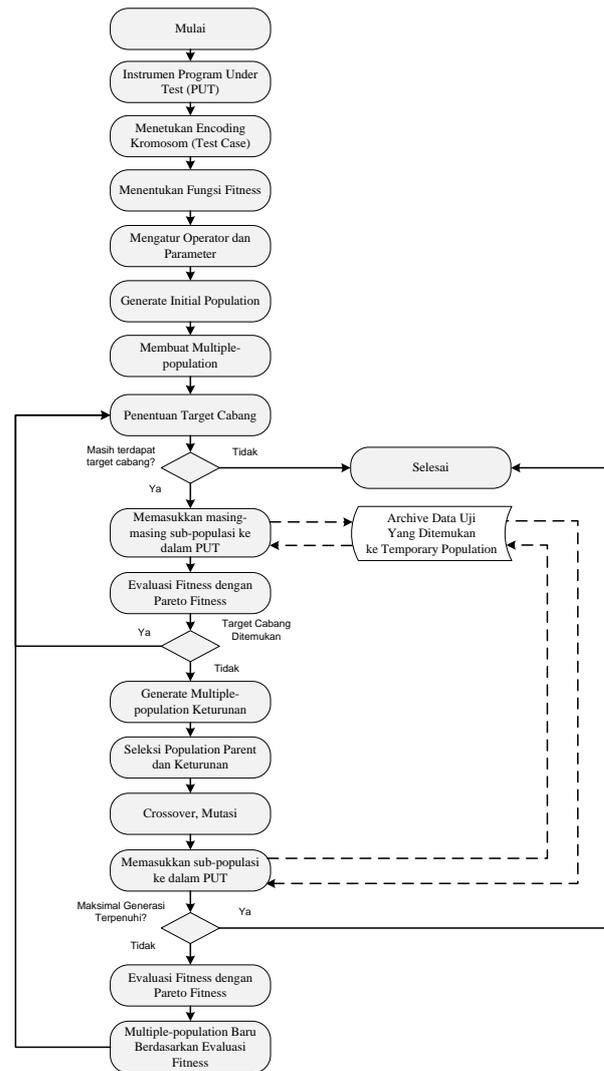
Metode yang diusulkan yaitu integrasi *pareto fitness*, *multiple-population* dan *temporary population* di dalam algoritma genetika untuk pembangkitan data tes perangkat lunak. *Pareto fitness* digunakan untuk menghitung masing-masing individu yang memungkinkan dengan menggunakan fungsi fitness yang berbeda. Hal ini dapat mempercepat pencarian target data tes. *Temporary population* digunakan untuk menyimpan individu yang sesuai dengan fungsi fitness yang memenuhi kriteria. *Temporary population* juga digunakan untuk membandingkan hasil individu sebelumnya dengan generasi yang sedang berjalan. Hal ini akan membantu pengurangan data tes yang sedang dicari, karena sebagian data tes sudah ditemukan. *Multiple-population* digunakan untuk menghindari *premature convergence* dan menjadikan individu lebih beragam.

Gambar 1 Menunjukkan alur dari metode yang diusulkan. langkah yang pertama adalah membuat instrumen program yang akan diuji (*Program Under Test (PUT)*). Instrumen tersebut diantaranya analisa kondisi yang ada di source code program tersebut. Instrumen ini nantinya akan berpengaruh pada penentuan nilai fitness. Setelah itu proses selanjutnya adalah menentukan proses *encoding* untuk mengkodekan kromosom yang akan dimasukkan di dalam populasi. Dalam penelitian ini digunakan *binary string encoding* untuk pengkodean kromosom. Proses penentuan fungsi fitness dilakukan selanjutnya dengan melihat instrumen yang telah dianalisa pada tahap awal.

Parameter dan operator algoritma genetika diatur untuk inisialiasi kebutuhan saat proses algoritma genetika berlangsung. Contoh pengaturan parameter adalah mengatur panjang dari kromosom, ukuran populasi dan bermacam-macam nilai probabilitas. Sedangkan contoh mengatur operator seperti pemilihan proses seleksi, *crossover* dan mutasi.

Proses selanjutnya yaitu membangkitkan populasi awal secara acak. Untuk metode yang diusulkan, proses selanjutnya adalah pembuatan *multiple-population*. Kemudian pemilihan target cabang ditentukan, jika target cabang masih ada akan dilanjutkan proses pencarian, jika tidak maka pencarian selesai. Dari masing-masing sub-populasi yang dihasilkan akan dievaluasi menggunakan *pareto fitness*. Setelah dilakukan evaluasi fitness selanjutnya masing-masing kromosom yang terdapat di dalam sub-populasi dimasukkan ke dalam PUT. Di bagian ini akan dilakukan pengecekan apakah kromosom sesuai dengan data tes yang dibutuhkan. Di dalamnya terdapat kemungkinan tiga kemungkinan, pertama tidak ada satupun data tes yang diharapkan diciptakan. Kedua,

hanya terdapat sebagian data tes yang memenuhi kriteria, kemudian dimasukkan ke dalam *archive* data tes untuk disimpan. Ketiga, semua data tes berhasil ditemukan, selanjutnya dimasukkan ke dalam *archive* data tes untuk disimpan. Pada bagian ini *temporary population* mengecek berdasarkan evaluasi fitness. Walaupun bukan individu yang sesuai target cabang, akan tetapi apabila memenuhi kriteria yang lain maka akan dilakukan penyimpanan ke dalam *archive temporary population*. Proses selanjutnya mengecek apakah cabang yang sudah dipilih sebelumnya dieksekusi. Jika dieksekusi maka akan ke proses penentuan target cabang lagi. Apabila tidak maka akan ke proses selanjutnya, yaitu pembuatan *multiple-population* keturunan.



Gambar 1. Metode yang Diusulkan

Setelah populasi keturunan baru tercipta, selanjutnya akan dilakukan proses pemilihan populasi *parent* dan populasi keturunan yang nantinya akan dilakukan proses *crossover*. Proses mutasi akan dilanjutkan setelah proses *crossover* selesai dikerjakan. Setelah itu sub-populasi keturunan yang sudah melalui operasi genetik dimasukkan ke dalam PUT. Jika terdapat individu yang sesuai dengan kriteria maka akan dilakukan penyimpanan ke *archive temporary population*. Langkah berikutnya melakukan pengecekan maksimal generasi yang sudah ditentukan. Apabila generasi saat ini sama dengan batas maksimal generasi, maka pencarian akan dihentikan, jika tidak maka proses berikutnya akan dikerjakan. Selanjutnya adalah proses evaluasi fitness menggunakan

pareto fitness. Setelah itu sub-populasi baru akan diciptakan berdasarkan evaluasi *fitness* yang sudah didapatkan. Proses pemilihan target cabang kembali dilakukan, dan mengulangi proses selanjutnya sampai target cabang sudah tidak ada atau batas maksimal generasi sudah terpenuhi.

4 HASIL EKSPERIMEN

Eksperimen dilakukan pada komputer dengan spesifikasi processor intel intel core 2 duo, 2 GB RAM, dan Sistem Operasi Windows 8. Lingkungan pengembangan program menggunakan Netbeans IDE 8 dengan bahasa pemrograman Java.

Di dalam penelitian ini menggunakan *benchmark* data program publik yang juga digunakan oleh peneliti lain seperti oleh Pachauri & Srivastava (Pachauri & Srivastava, 2013) dan Ferrer et al (Ferrer et al., 2012). Penggunaan *benchmark* program publik memungkinkan orang lain dapat memvalidasi hasil penelitian yang dilakukan. Pada penelitian ini menggunakan tiga *benchmark* program yang banyak digunakan oleh peneliti, yaitu *Myers Triangle*, *Michael Triangle* dan *Sthamer Triangle* (Pachauri & Srivastava, 2013)(Ferrer et al., 2012). Berikut ini merupakan informasi *benchmark* program yang digunakan:

- *Myers Triangle*. Program ini menggolongkan segitiga atas dasar sisi input sebagai non-segitiga atau segitiga, yaitu, sama kaki, sama sisi atau sisi tak sama panjang. Diberikan tiga input nilai di dalam parameter yang semuanya mewakili sisi segitiga. *Control flow graph* yang dihasilkan memiliki 14 *node* dengan predikat *node* berjumlah 6. Memiliki kondisi kesetaraan dengan operator DAN, yang membuat cabang sulit untuk dilewati.
- *Michael Triangle*. Program ini juga menggolongkan segitiga atas dasar sisi input sebagai non-segitiga atau segitiga, yaitu, sama kaki, sama sisi atau sisi tak sama panjang. Dibutuhkan tiga input nilai dengan semua dari ketiganya mewakili sisi segitiga tetapi dengan kondisi predikat yang berbeda. *Control flow graph* yang dihasilkan memiliki 26 *node* dengan jumlah predikat *node* 11.
- *Sthamer Triangle*. Seperti *benchmark* program sebelumnya, program ini juga menggolongkan segitiga atas dasar sisi input sebagai non-segitiga atau segitiga, yaitu, sama kaki, sama sisi, segitiga sudut kanan atau sisi tak sama panjang. *Control flow graph* yang dihasilkan memiliki 29 *node* dengan jumlah predikat *node* 13. Program ini memiliki kondisi persamaan dengan operator DAN dan operator relasional yang kompleks

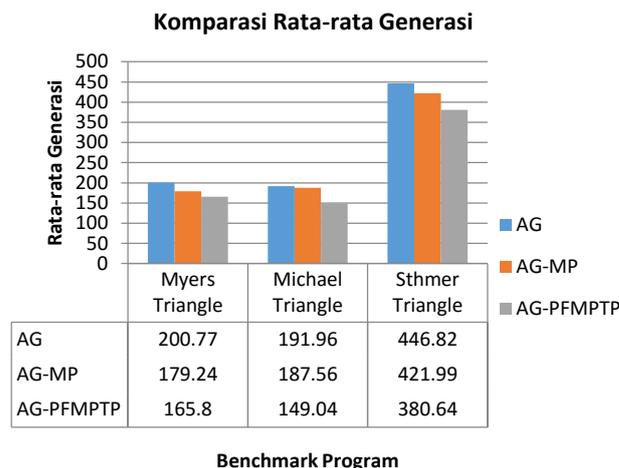
Pengujian akan dilakukan dengan masing-masing *benchmark* program yang digunakan. Penentuan parameter dan operator berdasarkan hasil beberapa percobaan yang dilakukan oleh peneliti. Dari beberapa percobaan didapatkan beberapa pengaturan yang cukup memadai untuk digunakan di dalam pengujian. Tabel 1 menunjukkan pengaturan parameter dan operator yang digunakan. Hal ini juga dilakukan oleh peneliti lain yaitu Yao & Gong (Yao & Gong, 2014).

Pengujian yang dilakukan bertujuan untuk mengetahui jumlah rata-rata generasi dan rata-rata *coverage*. Pengujian dilakukan sebanyak seratus kali untuk rata-rata generasi dan tiga puluh kali untuk rata-rata *coverage*.

Tabel 1. Penentuan Parameter dan Operator

Parameter/ Operator	Benchmark Program		
	Myers	Michaels	Sthmer
Jumlah kromosom	60	60	60
Panjang bit	6	6	6
Range nilai data tes	0-64	0-64	0-64
Seleksi	Random	Random	Random
Crossover	Single	Single	Single
Survive probability	0.5	0.5	0.5
Mutation probability	0.15	0.15	0.15
Jumlah subpopulasi	6	6	6
Maks generasi	1000	1000	1000
Jumlah Generasi	200	200	500

Dari Gambar 2 dapat dilihat bahwa metode AG-MP memiliki generasi lebih sedikit dibandingkan metode AG dari pengujian yang dilakukan pada semua *benchmark* program. Pada *benchmark* program *Myers Triangle* rata-rata generasi berkurang sebesar 21,53. Sedangkan pada pengujian yang dilakukan menggunakan *benchmark* program *Michael Triangle* metode AG dan AG-MP tidak terlalu jauh berbeda nilai dari rata-rata generasinya yang hanya berkurang 4,4. Kemudian untuk *benchmark* program *Sthmer Triangle* nilai rata-rata generasi berkurang sebesar 24,83.

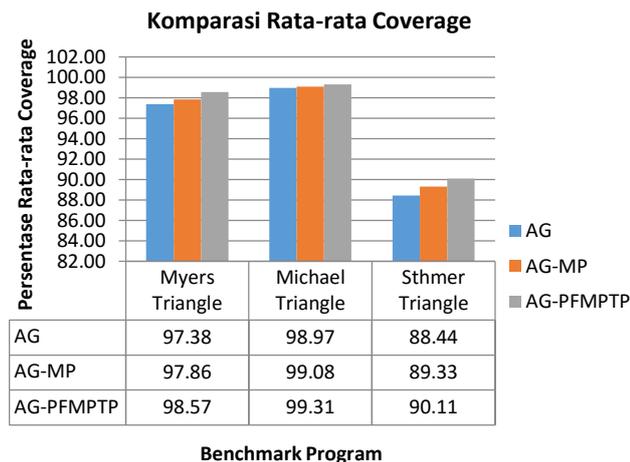


Gambar 2. Komparasi Rata-rata Generasi

Kemudian secara konsisten metode yang diusulkan yaitu metode AG-PFMPTP memiliki generasi yang paling sedikit dibandingkan dengan metode AG ataupun AG-MP pada semua *benchmark* program. Untuk *benchmark* program *Myers Triangle* metode AG-PFMPTP berkurang 34,97 untuk rata-rata generasinya dari metode AG. Sedangkan jika dibandingkan dengan metode AG-MP, rata-rata generasi metode AG-PFMPTP berkurang 13,44. Kemudian *benchmark* program *Michael Triangle* rata-rata generasi metode AG-PFMPTP berkurang 42,92 dari metode AG dan 38,52 dari metode AG-MP. Sedangkan untuk *benchmark* program *Sthmer Triangle* rata-rata generasi metode AG-PFMPTP berkurang 66,18 dari metode AG dan 41,35 dari metode AG-MP. Dari hasil komparasi tersebut, dapat disimpulkan bahwa metode AG-PFMPTP lebih efisien dalam mendapatkan data tes dibandingkan dengan kedua metode yang lainnya.

Dari Gambar 3 menunjukkan bahwa metode AG-MP memiliki nilai persentase *coverage* lebih besar dibandingkan metode AG dengan selisih 0,48% pada *benchmark* program *Myers Triangle*, selisih 0,11% pada *benchmark* program *Michael Triangle* dan selisih 0,89% pada *benchmark* program

Sthmer Triangle. Selisih terkecil terdapat pada benchmark program *Michael Triangle*. Kemudian metode AG-PFMPTP memiliki nilai persentase *coverage* lebih besar dibandingkan dengan metode AG. Metode AG-PFMPTP memiliki selisih persentase dari metode AG sebesar 1,19% untuk benchmark program *Myers Triangle*, 0,34% untuk benchmark program *Michael Triangle* dan 1,67% untuk benchmark program *Sthmer Triangle*. Begitupun jika dibandingkan dengan metode AG-MP, metode AG-PFMPTP lebih besar nilai persentase rata-rata *coverage* yang dihasilkan. Metode AG-PFMPTP memiliki selisih persentase dari metode AG sebesar 0,71% untuk benchmark program *Myers Triangle*, 0,23% untuk benchmark program *Michael Triangle* dan 0,78% untuk benchmark program *Sthmer Triangle*.



Gambar 3. Komparasi Rata-rata Coverage

Dari ketiga metode tersebut bisa dilihat tidak terlalu jauh peningkatan nilai persentase *coverage* yang dihasilkan. Jika dilihat pada hasil komparasi rata-rata *coverage* untuk benchmark program *Michael Triangle* sangat tipis sekali perbedaan nilai persentase dari ketiga metode. Walaupun demikian metode yang diusulkan, yaitu AG-PFMPTP memiliki nilai persentase *coverage* yang paling besar diantara dua metode yang lain. Hal tersebut menandakan metode AG-PFMPTP lebih banyak mencakup data tes yang dibutuhkan.

Untuk memvalidasi hasil evaluasi hasil pengujian metode AG, AG-MP dan AG-PFMPTP yang telah dilakukan signifikan berbeda atau tidak, maka akan dilakukan pengujian statistik uji beda. Sebelum dilakukan penentuan penggunaan metode uji beda yang digunakan, terlebih dahulu dilakukan uji normalitas data. Karena data yang didapatkan memiliki distribusi normal maka dilakukan pengujian parametrik dengan metode t-test. Uji normalitas data yang digunakan menggunakan metode Shapiro-Wilk (Razali & Wah, 2011).

Uji beda untuk rata-rata generasi bertujuan untuk mengetahui apakah ada perbedaan yang signifikan antara metode AG, AG-MP dan AG-PFMPTP dalam kecepatan proses pencarian data tes. Tabel 2 merupakan hasil t-test rata-rata generasi untuk masing-masing perbandingan metode. Masing-masing metode akan dibandingkan dengan metode lain. Metode AG akan dibandingkan dengan AG-MP, metode AG dengan AG-PFMPTP dan metode AG-MP dengan metode AG-PFMPTP.

Dari Tabel 2 dapat dilihat hasil Sig.(2-tailed) menghasilkan nilai yang berbeda untuk masing-masing perbandingan nilai. P-value dari masing-masing perbandingan dihitung dari hasil Sig.(2-tailed) dibagi dua. Dari hasil tersebut dapat disimpulkan:

1. Hasil uji beda rata-rata generasi metode AG dengan AG-MP menghasilkan nilai sig.(2-tailed) 0.116, oleh karena itu P-value yang dihasilkan adalah 0.056. Dari nilai tersebut dapat disimpulkan bahwa H0 diterima karena $0.056 > 0.5$ yang artinya tidak terdapat perbedaan yang signifikan antara metode AG dan AG-MP.
2. Hasil uji beda rata-rata generasi metode AG dengan AG-PFMPTP menghasilkan nilai sig.(2-tailed) 0.036, oleh karena itu P-value yang dihasilkan adalah 0.018. Dari nilai tersebut dapat disimpulkan bahwa H0 ditolak karena $0.018 < 0.05$ yang artinya terdapat perbedaan yang signifikan antara metode AG dan AG-PFMPTP.
3. Hasil uji beda rata-rata generasi metode AG-MP dengan AG-PFMPTP menghasilkan nilai sig.(2-tailed) 0.073, oleh karena itu P-value yang dihasilkan adalah 0.037. Dari nilai tersebut dapat disimpulkan bahwa H0 ditolak karena $0.037 < 0.05$ yang artinya terdapat perbedaan yang signifikan antara metode AG-MP dan AG-PFMPTP.

Tabel 2. Uji Beda Rata-rata Generasi

		Paired Differences				t	df	Sig. (2-tailed)	
		Mean	Std. Deviation	Std. Error	95% Confidence Interval of the Difference				
					Lower				Upper
		AG - AG_MP	AG - AG_PFMPTP	AG_MP - AG_PFMPTP					
Pair 1	AG - AG_MP	16.92000	10.96747	6.33207	-10.32469	44.16469	2.672	2	.116
Pair 2	AG - AG_PFMPTP	48.02333	16.21879	9.36392	7.73363	88.31303	5.129	2	.036
Pair 3	AG_MP - AG_PFMPTP	31.10333	15.36220	8.86937	-7.05849	69.26516	3.507	2	.073

Untuk hasil uji beda rata-rata generasi dapat dilihat bahwa untuk metode AG dan AG-MP tidak terdapat perbedaan yang signifikan. Dari penelitian Yao & Gong (Yao & Gong, 2014) menyatakan bahwa AG dengan menggunakan *multiple-population* signifikan berbeda dengan AG standar. Hal yang menyebabkan perbedaan ini dapat timbul dari pengaturan parameter & operator, sistem perangkat lunak, konversi program yang berbeda dan tujuan pengujian (Yao & Gong, 2014). Sedangkan hasil uji beda rata-rata generasi untuk metode AG-PFMPTP terdapat perbedaan yang signifikan baik dibandingkan dengan AG maupun AG-MP. Dengan melihat hasil uji beda tersebut dan komparasi metode yang telah dilakukan, maka metode AG-PFMPTP yang merupakan integrasi *pareto fitness*, *multiple-population* dan *temporary population* dapat meningkatkan kinerja dari algoritma genetika secara signifikan dalam melakukan proses pencarian data tes untuk pengujian perangkat lunak.

Selanjutnya untuk uji beda rata-rata *coverage* bertujuan untuk mengetahui apakah ada perbedaan yang signifikan antara metode AG, AG-MP dan AG-PFMPTP dalam mencakup data tes yang didapatkan berdasarkan batasan generasi tertentu. Tabel 3 merupakan hasil t-test rata-rata generasi untuk masing-masing perbandingan metode. Masing-masing metode akan dibandingkan dengan metode lain.

Metode AG akan dibandingkan dengan AG-MP, metode AG dengan AG-PFMPTP dan metode AG-MP dengan metode AG-PFMPTP.

Tabel 3. Uji Beda Rata-rata Coverage

		Paired Differences					t	df	Sig. (2-tailed)
		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference				
					Lower	Upper			
Pair 1	GA - GA_MP	-.49334	.38726	.22358	-1.45534	.46866	-2.207	2	.158
Pair 2	GA_PFM - PTP	-1.06732	.66947	.38652	-2.73038	.59573	-2.761	2	.110
Pair 3	GA_PFM - PTP	-.57398	.29968	.17302	-1.31844	.17047	-3.317	2	.080

Dari Tabel 3 dapat dilihat hasil Sig.(2-tailed) menghasilkan nilai yang berbeda untuk masing-masing perbandingan nilai. *P-value* dari masing-masing perbandingan dihitung dari hasil Sig.(2-tailed) dibagi dua seperti yang dilakukan pada uji beda rata-rata generasi. Dari hasil tersebut dapat disimpulkan:

1. Hasil uji beda rata-rata *coverage* metode AG dengan AG-MP menghasilkan nilai sig.(2-tailed) 0.158, oleh karena itu *P-value* yang dihasilkan adalah 0.079. Dari nilai tersebut dapat disimpulkan bahwa H_0 diterima karena $0.079 > 0.05$ yang artinya tidak terdapat perbedaan yang signifikan antara metode AG dan AG-MP.
2. Hasil uji beda rata-rata *coverage* metode AG dengan AG-PFMPTP menghasilkan nilai sig.(2-tailed) 0.110, oleh karena itu *P-value* yang dihasilkan adalah 0.055. Dari nilai tersebut dapat disimpulkan bahwa H_0 diterima karena $0.055 > 0.05$ yang artinya tidak terdapat perbedaan yang signifikan antara metode AG dan AG-PFMPTP.
3. Hasil uji beda rata-rata *coverage* metode AG-MP dengan AG-PFMPTP menghasilkan nilai sig.(2-tailed) 0.080, oleh karena itu *P-value* yang dihasilkan adalah 0.040. Dari nilai tersebut dapat disimpulkan bahwa H_0 ditolak karena $0.040 < 0.05$ yang artinya terdapat perbedaan yang signifikan antara metode AG-MP dan AG-PFMPTP.

Dari hasil uji beda rata-rata *coverage* dapat dilihat bahwa untuk metode AG dan AG-MP tidak terdapat perbedaan yang signifikan. Kemudian hasil uji beda rata-rata *coverage* untuk metode AG dan AG-PFMPTP tidak terdapat perbedaan yang signifikan pula. Sedangkan metode AG-MP dan AG-PFMPTP terdapat perbedaan yang signifikan. Dengan melihat hasil uji beda dan komparasi metode yang telah dilakukan, metode AG-PFMPTP yang merupakan integrasi *pareto fitness*, *multiple-population* dan *temporary population* secara signifikan lebih baik dibandingkan metode AG-MP dalam mencakup data tes yang dibutuhkan untuk pengujian perangkat lunak.

5 KESIMPULAN

Dari hasil penelitian yang sudah dilakukan dari tahap analisa permasalahan dan *literature review* sampai tahap

validasi hasil evaluasi pengujian, telah diketahui hasil peningkatan kinerja dari metode yang diusulkan. Integrasi *pareto fitness*, *multiple-population* dan *temporary population* (AG-PFMPTP) dapat meningkatkan kinerja algoritma genetika dalam pencarian data tes yang digunakan untuk pengujian perangkat lunak dibandingkan algoritma genetika standar (AG) dan algoritma genetika dengan menggunakan *multiple-population* (AG-MP).

Dari hasil pengujian rata-rata generasi secara konsisten metode yang diusulkan dapat memperkecil nilai rata-rata generasi dalam proses menghasilkan data tes pada semua benchmark program yang digunakan. Hal ini menunjukkan metode AG-PFMPTP lebih cepat dalam menemukan data tes dibandingkan metode AG ataupun AG-MP. Kemudian untuk hasil pengujian rata-rata *coverage* metode AG-PFMPTP mencakup lebih banyak data tes yang dibutuhkan dibandingkan metode AG dan AG-MP. Meskipun kenaikan nilai rata-rata *coverage* dari metode AG-PFMPTP tidak terlalu besar.

Pada proses validasi hasil menggunakan t-test untuk uji beda rata-rata generasi, metode AG-PFMPTP secara signifikan terdapat perbedaan jika dibandingkan dengan metode AG dan AG-MP. Hal ini menunjukkan metode AG-PFMPTP lebih cepat dalam mendapatkan data tes yang dibutuhkan dibandingkan dengan metode AG dan AG-MP. Kemudian untuk validasi uji beda rata-rata *coverage* didapatkan hasil bahwa metode AG-PFMPTP tidak terdapat perbedaan yang signifikan dalam mencakup data tes yang didapatkan jika dibandingkan dengan metode AG. Akan tetapi terdapat perbedaan yang signifikan apabila metode AG-PFMPTP dibandingkan dengan metode AG-MP.

REFERENSI

- Alakeel, A. (2014). Using Fuzzy Logic Techniques for Assertion-Based Software Testing Metrics. *The Scientific World Journal*. Retrieved from <http://www.hindawi.com/journals/tswj/aa/629430/>
- Aleti, A., & Grunske, L. (2015). Test data generation with a Kalman filter-based adaptive genetic algorithm. *Journal of Systems and Software*, 103, 343–352. <http://doi.org/10.1016/j.jss.2014.11.035>
- Ali, S., Briand, L. C., Hemmati, H., & Panesar-Walawege, R. K. (2010). A systematic review of the application and empirical investigation of search-based test case generation. *IEEE Transactions on Software Engineering*, 36(6), 742–762. <http://doi.org/10.1109/TSE.2009.52>
- Alshraideh, M., Mahafzah, B. a., & Al-Sharaeh, S. (2011). A multiple-population genetic algorithm for branch coverage test data generation. *Software Quality Journal*, 19, 489–513. <http://doi.org/10.1007/s11219-010-9117-4>
- Anand, S., Burke, E. K., Yueh, T., Clark, J., Cohen, M. B., Grieskamp, W., ... Zhu, H. (2015). The Journal of Systems and Software An orchestrated survey of methodologies for automated software test case generation Orchestrators and Editors , 86(2013), 1978–2001. <http://doi.org/10.1016/j.jss.2013.02.061>
- Bueno, P. M. S., Jino, M., & Wong, W. E. (2014). Diversity oriented test data generation using metaheuristic search techniques. *Information Sciences*, 259, 490–509. <http://doi.org/10.1016/j.ins.2011.01.025>
- Cantu-paz, E. (1997). A Survey of Parallel Genetic Algorithms. *Department of Computer Science and Illinois Genetic Algorithms Laboratory University of Illinois at Urbana-Champaign*, 10. Retrieved from <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.106>

- Cohen, M. B., Colbourn, C. J., & Ling, a. C. H. (2003). Augmenting simulated annealing to build interaction test suites. *14th International Symposium on Software Reliability Engineering, 2003*. *ISSRE* 2003. <http://doi.org/10.1109/ISSRE.2003.1251061>
- Díaz, E., Tuya, J., & Blanco, R. (2003). Automated software testing using a metaheuristic technique based on tabu search. *Automated Software Engineering*, Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1240327
- Elaoud, S., Loukil, T., & Teghem, J. (2007). The Pareto fitness genetic algorithm: Test function study. *European Journal of Operational Research*, 177(3), 1703–1719. <http://doi.org/10.1016/j.ejor.2005.10.018>
- Ferrer, J., Chicano, F., & Alba, E. (2012). Evolutionary algorithms for the multi-objective test data generation problem. *Software: Practice and Experience*, 42(11), 1331–1362. <http://doi.org/10.1002/spe.1135>
- Ferrer, J., Kruse, P. M., Chicano, F., & Alba, E. (2015). Search based algorithms for test sequence generation in functional testing. *Information and Software Technology*, 58, 419–432. <http://doi.org/10.1016/j.infsof.2014.07.014>
- Harman, M., & McMinn, P. (2010). A Theoretical and Empirical Study of Search Based Testing: Local, Global and Hybrid Search. *IEEE Transactions on Software Engineering*, 36(2), 226–247. <http://doi.org/http://dx.doi.org/10.1109/TSE.2009.71>
- Hermadi, I., Lokan, C., & Sarker, R. (2014). Dynamic stopping criteria for search-based test data generation for path testing. *Information and Software Technology*, 56(4), 395–407. <http://doi.org/10.1016/j.infsof.2014.01.001>
- Lakhotia, K., McMinn, P., & Harman, M. (2009). Automated test data generation for coverage: Haven't we solved this problem yet? *TAIC PART 2009 - Testing: Academic and Industrial Conference - Practice and Research Techniques*, 95–104. <http://doi.org/10.1109/TAICPART.2009.15>
- Mao, C. (2014). Generating Test Data for Software Structural Testing Based on Particle Swarm Optimization. *Arabian Journal for Science and Engineering*, 39, 4593–4607. <http://doi.org/10.1007/s13369-014-1074-y>
- Mao, C., Xiao, L., Yu, X., & Chen, J. (2015). Adapting ant colony optimization to generate test data for software structural testing. *Swarm and Evolutionary Computation*, 20, 23–36. <http://doi.org/10.1016/j.swevo.2014.10.003>
- McMinn, P. (2004). Search-based software test data generation: a survey. *Software Testing, Verification and Reliability*, 1–58. <http://doi.org/10.1002/stvr.v14:2>
- McMinn, P. (2011). Search-Based Software Testing: Past, Present and Future. *2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops*. <http://doi.org/10.1109/ICSTW.2011.100>
- McMinn, P., Harman, M., Lakhotia, K., Hassoun, Y., & Wegener, J. (2012). Input Domain Reduction through Irrelevant Variable Removal and Its Effect on Local, Global, and Hybrid Search-Based Structural Test Data Generation. *IEEE Transactions on Software Engineering*, 38(2), 453–477. <http://doi.org/10.1109/TSE.2011.18>
- Miller, J., Reformat, M., & Zhang, H. (2006). Automatic test data generation using genetic algorithm and program dependence graphs. *Information and Software Technology*, 48, 586–605. <http://doi.org/10.1016/j.infsof.2005.06.006>
- Pachauri, A., & Srivastava, G. (2013). Automated test data generation for branch testing using genetic algorithm: An improved approach using branch ordering, memory and elitism. *Journal of Systems and Software*, 86(5), 1191–1208. <http://doi.org/10.1016/j.jss.2012.11.045>
- Patil, M., & Nikumbh, P. J. (2012). Pair-wise Testing Using Simulated Annealing. *Procedia Technology*, 4, 778–782. <http://doi.org/10.1016/j.protcy.2012.05.127>
- Razali, N. M., & Wah, Y. B. (2011). Power comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling tests. *Journal of Statistical Modeling and Analytics*, 2(1), 21–33.
- Ribeiro, J. C. B. (2008). Search-based test case generation for object-oriented java software using strongly-typed genetic programming. *Proceedings of the 2008 GECCO Conference Companion on Genetic and Evolutionary Computation - GECCO '08*, 1819. <http://doi.org/10.1145/1388969.1388979>
- Srivastava, P. R., & Baby, K. (2010). Automated Software Testing Using Metahuristic Technique Based on an Ant Colony Optimization. *2010 International Symposium on Electronic System Design*, 235–240. <http://doi.org/10.1109/ISED.2010.52>
- Srivastava, P. R., & Kim, T. (2009). Application of Genetic Algorithm in Software Testing. *International Journal of Software Engineering and Its Applications*, 3(4), 87–96. Retrieved from http://www.sersc.org/journals/IJSEIA/vol3_no4_2009/6.pdf
- Vos, T. E. J., Lindlar, F. F., Wilmes, B., Windisch, A., Baars, A. I., Kruse, P. M., ... Wegener, J. (2013). Evolutionary functional black-box testing in an industrial setting. *Software Quality Journal*, 21, 259–288. <http://doi.org/10.1007/s11219-012-9174-y>
- Yao, X., & Gong, D. (2014). Genetic Algorithm-Based Test Data Generation for Multiple Paths via Individual Sharing. *Computational Intelligence and Neuroscience*, 2014, 1–12. <http://doi.org/10.1155/2014/591294>

BIOGRAFI PENULIS



Mohammad Reza Maulana. Memperoleh gelar S.Kom pada bidang ilmu komputer di Sekolah Tinggi Manajemen Informatika dan Komputer (STMIK) Widya Pratama dan M.Kom pada bidang ilmu komputer di Universitas Dian Nuswantoro. Saat ini menjadi pengajar di STMIK Widya Pratama Pekalongan. Minat penelitian pada bidang software engineering.



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. Menjadi pengajar dan peneliti di Fakultas Ilmu Komputer, Universitas Dian Nuswantoro. Merupakan pendiri dan CEO PT Brainmatics, sebuah perusahaan yang bergerak di bidang pengembangan software. Minat penelitian pada bidang software engineering dan machine learning. Profesional member dari asosiasi ilmiah ACM, PMI dan IEEE Computer Society.



Catur Supriyanto. Dosen di Fakultas Ilmu Komputer, Universitas Dian Nuswantoro, Semarang, Indonesia. Menerima gelar master dari Universiti Teknikal Malaysia Melaka (UTEM), Malaysia. Minat penelitiannya adalah *information retrieval*, *machine learning*, *soft computing* dan *intelligent system*.