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* menunjukan 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 projek, 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 menunjukan 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

Copyright © 2015 IlmuKomputer.Com

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 dalan *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 meningkatan jumlah kelas mayorita 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 datasaet 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 menunjukan 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 pebagian 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 menunjukan 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 menerapan 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 dikelompokan dalam 5 tipe. Yaitu pengklasfikasian 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 Decission Tree (C45, Classification an Regression Tree (CART), dan Random Forest (RF)). Hasilnya menunjukan 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*.

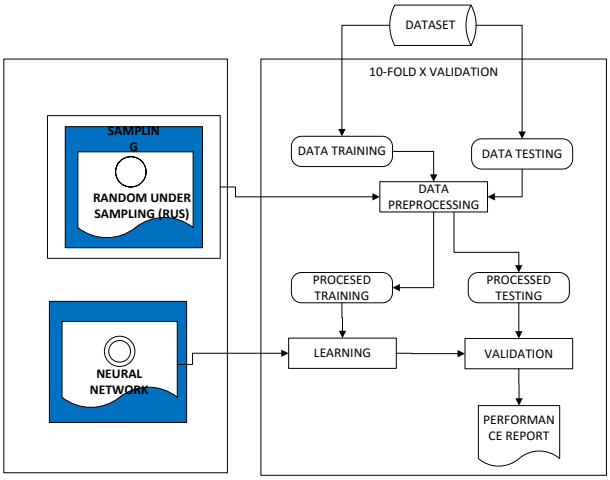
Penelitian yang dilakukan oleh Arar, dan Ayan (Arar & Ayan, 2015), untuk prediksi cacat *software* mengkombinasikan *Neural Network* (NN) dan novel *Artificial Bee Colony* (ABC) algoritma yang digunakan dalam penelitian ini. Aplikasi yang digunakan adalah WEKA. Teknik umum yang digunakan adalah *10-fold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

Copyright © 2015 IlmuKomputer.Com

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 ditunjukan pada Gambar 1



Gambar 1. Kerangka Pemikiran Model yang Diusulkan

Neural Network (NN) atau jaringan syaraf tiruan dibuat oleh Warren McCulloh 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 *backporpagation.* 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, & Sagiroglu (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:

   $$\text{Input } j = \sum^{n}_{i=1} 0_i \, W_{ij} + \theta_i$$

   Keterangan:
   Oi = *Output* simpul I dari layer sebelumnya
   Wij =Bobot relasi dari simpul I pada layer sebelumnya ke sempul j
   Θj = Bias (sebagai pembatas)
3. Berdasarkan langkah sebeblumnya 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:

   $$\text{Error}j = \text{Output}_j \, . \, (1 - \text{Output}_j).(\text{Target}_j - \text{Output}_j)$$

   Keterangan:
   Outputj = *Output* aktual dari simpul j
   Targetj = Nilai target yang sudah diketahuo pada data training
5. Setelah nilai error dihitung, selanjutnya dibalik ke layer sebelumnya (*backppropagated*). Untuk menghitung nilai error pada hidden layer, menggunakan rumus

   $$\text{Error}j = \text{Output}_j \, . \, (1 - \text{Output}_j) \sum^{n}_{k=1} \text{Error}_k W_{jk}$$

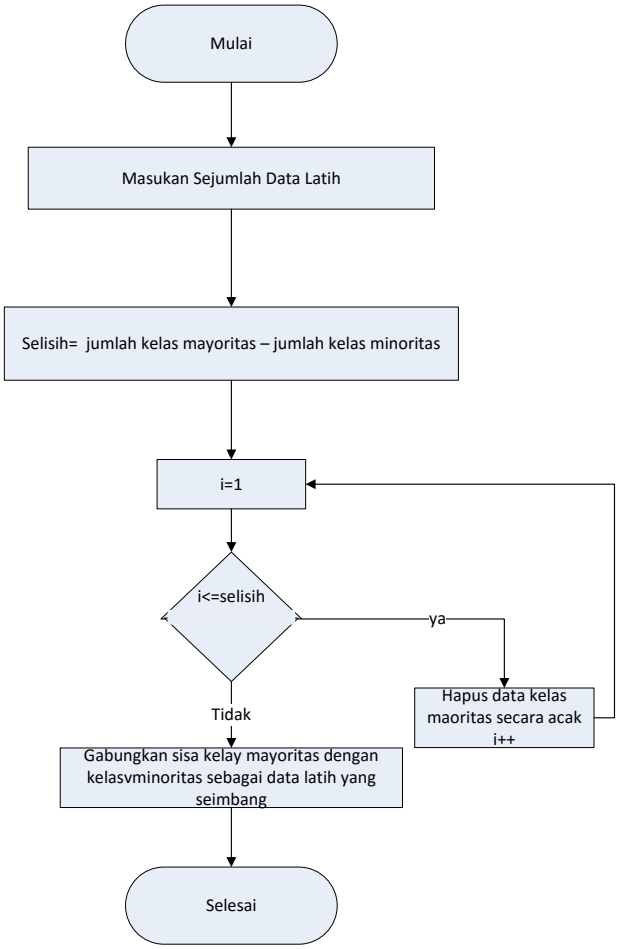
   Keterangan :
   $Output_j$ = *Output* aktual dari simpul j
   Errork = *Error* Simpul k
   Wjk = Bobot relasi dari simpul j ke simpul k pada layer berikutnya
6. Setelah nilai *error* dihitung kemudian bobot relasi diperbahari menggunakan rumus

   $$W_{ij} = W_{ij} + I \, . \, Error_j \, . \, Output_i$$

   Keterangan :
   Wij = Bobot relasi dari unit I pada layer sebelumnya ke unit j
   I = *Learning rate*
   Errorj = *Error* pada output layer simpul *j*
   Outputi = *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.

Copyright © 2015 IlmuKomputer.Com



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

Dalam penelitian ini menggunakan data sekunder yaitu NASA (*National Aeronautics and Space Administration*) MDP (*Metrics Data Programs*) *repository* sebagai *software matrics* 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 maximal

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 segemen kedua digunakan sebagai data testing untuk mevalidasi model (Witten, Frank, & Hall, 2011). Selanjutnya diterapkan tahapan eveluasi 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	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
CYCLOMATIC_COMPLEXITY	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
CYCLOMATIC_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_CYLOMATIC_COMPLEXITY	✓		✓	✓	✓	✓	✓	✓	✓	✓
PARAMETER_COUNT	✓		✓	✓	✓	✓	✓	✓	✓	✓
PERCENT_COMMENTS	✓		✓	✓	✓	✓	✓	✓	✓	✓
PATHOLOGICAL_COMPLEXITY										
Defective	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Jumlah attribute	**38**	**22**	**40**	**40**	**37**	**38**	**37**	**38**	**38**	**39**
Jumlah Modul	**327**	**1162**	**194**	**124**	**264**	**679**	**722**	**1053**	**1270**	**1694**
Jumlah modul yang cacat	**42**	**294**	**36**	**44**	**27**	**55**	**10**	**130**	**176**	**458**
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	**285**	**868**	**158**	**80**	**237**	**624**	**706**	**923**	**1094**	**1236**

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:

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

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

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

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

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

Copyright © 2015 IlmuKomputer.Com

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

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

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

G-Mean $= \sqrt{sensitivitas * specifity}$

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 perbadingan klasifikasi menggunakan curva *Receiver Operating Characteristic* (ROC). ROC menghasilkan dua garis dengan bentuk *false posiives* sebagai garis hosirontal *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).

Rumus Area Under Curve (AUC) = $\frac{1+TPrate-FPrate}{2}$

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	*Excellent classification*	
0.80 – 0.90	*Good classification*	
0.70 – 0.80	*Fair classification*	
0.60 – 0.70	*Poor classification*	
< 0.60	*Failure*	

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 ji 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 rank (di) + 1/2 $\sum$di=0 rank (di)
R- = $\sum$ di<0 rank (di) + 1/2 $\sum$di=0 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 perbadingan 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 diilhat pada Tabel 3 dan *Neural Network* dengan RUS (NN+RUS) pada Tabel 4

Tabel 3 Hasil Eksperimen Neural Network

Dataset	Accuracy	Recall	Specificity	PPV	NPV	F-Measure	G-Mean	AUC
CM1	84.1%	42.%	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.%	93.3%	67.7%	85.9%	0.56	0.66	0.82
MC2	78.%	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

Dataset	Accuracy	Recall	Specificity	PPV	NPV	F-Measure	G-Mean	AUC
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.%	44.7%	21.7%	100.%	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%

Copyright © 2015 IlmuKomputer.Com

Berdasarkan Tabel 5 perbandingan sensitivitas NN dan NN+RUS menunjukan 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 dibadingkan model *Neural Network*. Sedangkan satu dataset (KC3) memiliki nilai yang sama antara kedua model dan satu dataset (PC3) menunkukan 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 menunjukan hasil *f-measure* yang baik pada penggunaan *Neural Network* dengan *Random Under Sampling* dimana terdapat lima dataset (KC1, MC2, PC3, PC4, dan PC5) yang menunjukan 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) menunjukan 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 terapat tujuh dataset (CM1, MC2, PC1. PC2, PC3, PC4, dan PC5). Satu dataset (KC3) memilki *G-Mean* yang sama pada kedua model sedangkan dua dataset lainnya (KC1 dan MW1) menunjukan 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, KC31, MC2, PC1, PC2, PC3, PC4 dan PC5). Satu dataset (MW1) memiliki nilai yang sama sedangkan satu dataset lainnya (KC3) menunjukan 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 pringkat bertanda *Wilcoxon (Wilcoxon Signed Rangked test)* satu sisi unutk 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).

Copyright © 2015 IlmuKomputer.Com

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 (α = < 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: μNN = μNN+RUS (Tidak ada perbedaan antara model NN dengan NN+RUS)

*H*1: μNN ≠ μNN+RUS (Ada perbedaan antara model NN dengan NN+RUS)

Rekap pebandingan 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*

	CM1	KC1	KC3	MC2	MW1	PC1	PC2	PC3	PC4	PC5
NN	0.78	0.76	**0.82**	0.84	0.83	0.8	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	0.8	0.82	0.73	0.78	0.84
NN+ABC	0.77	**0.8**	0.816	0.82	0.83	**0.82**	0.82	0.73	0.771	0.83
NN+ROS	0.786	0.762	**0.82**	0.844	0.832	0.8	0.783	0.72	0.77	0.84
NN+SMOTE	0.77	0.76	0.799	0.85	0.83	0.8	0.778	0.71	0.77	0.83
NB	0.76	0.76	0.72	0.712	0.728	0.75	0.783	0.74	0.8	0.825
NB+RUS	0.78	0.756	0.73	0.73	0.74	0.77	0.838	0.745	0.81	**0.88**
LR	0.72	0.68	0.8	0.86	0.8	0.77	0.81	0.75	0.8	**0.88**
C45	0.736	0.614	0.731	0.505	0.5	0.716	0.736	0.736	0.736	0.736

Tabel 12 Hasil Uji Friedman

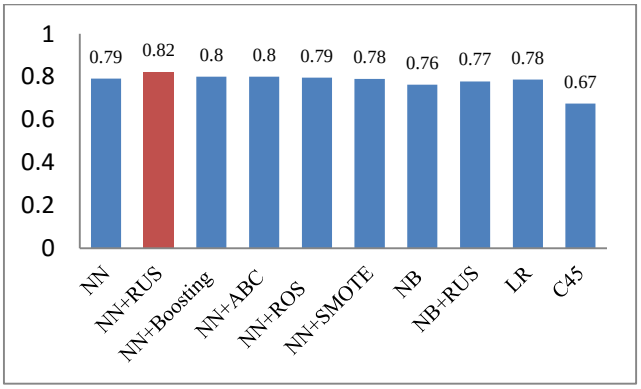
Q (Observed value)	43.723
Q (Critical value)	16.919
DF	9
p-value	< 0.0001
Alpha	0.05

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	1.000	1.000	0.420
NB	0.996	**0.001**	0.518	0.619	0.594	1.000	1	0.961	0.967	0.821
NB+RUS	1.000	0.090	0.998	0.999	0.999	1.000	0.961	1	1.000	0.099
LR	1.000	0.082	0.997	0.999	0.999	1.000	0.967	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+SMOSE, NB, dan C45 karena nilai alpha < 0.05.

Nilai AUC dihitung rata-ratanya agar dapat dibuat grafik pernabdingan 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*	→

Copyright © 2015 IlmuKomputer.Com

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 menunjukan 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 penggunaa metode NN+RUS mempu 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 TitleSMOTEBoost: 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 TecniquesNo 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 Algoritm 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.

Copyright © 2015 IlmuKomputer.Com

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.

Copyright © 2015 IlmuKomputer.Com