Perbandingan Algoritma Klasifikasi untuk Prediksi Cacat *Software* dengan Pendekatan CRISP-DM

Nurtriana Hidayati [1], Joko Suntoro [2], Galet Guntoro Setiaji[3]

[1] Prodi Sistem Informasi, Fakultas Teknologi Informasi dan Komunikasi, Universitas Semarang
Jl. Soekarno Hatta - Semarang
[1] anna@usm.ac.id

[2] [3] Prodi Teknik Informatika, Fakultas Teknologi Informasi dan Komunikasi, Universitas Semarang
Jl. Soekarno Hatta - Semarang
[2] jokosuntoro@usm.ac.id
[3] gallet@usm.ac.id

Abstrak

Proses prediksi cacat *software* merupakan bagian terpenting dalam sebuah pengujian kuliatas *software* sering juga disebut dengan *software quality* yang bertujuan untuk mengetahui mutu *software* dalam pemenuhan kebutuhan fungsional dan kinerjanya. Metode *machine learning* mempunyai kinerja lebih baik untuk menemukan cacat *software* daripada metode manual. Algoritma klasifikasi dalam *machine learning* yang pernah digunakan untuk prediksi cacat *software* antara lain k-Nearest Neighbor (k-NN), Naïve Bayes (NB) dan Decision Tree (CART). Dalam penelitian ini akan dibandingkan kinerja antara algoritma - algoritma klasifikiasi yaitu k-NN, NB, dan CART untuk prediksi cacat *software* dengan pendekatan CRISP-DM. CRISP-DM merupakan model proses data mining dengan 6 tahapan yaitu: *Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, dan Deployment* dalam menentukan perbandingan algoritma klasifikasi dalam memprediksi cacat software. *Software* Matrix yang digunakan pada penelitian ini adalah tujuh dataset dari NASA MDP. Hasil penelitian menunjukkan bahwa nilai rata-rata akurasi algoritma CART lebih baik daripada algoritma k-NN dan NB dengan nilai 0,867. Sedangkan nilai rata-rata akurasi algoritma k-NN dan NB masing-masing 0,859 dan 0,778.

Kata kunci: Prediksi Cacat Software, k-Nearest Neighbor, Naïve Bayes, CART, CRISP-DM, Rekayasa Perangkat Lunak, Data Mining, Machine Learning

Abstract

Process of predicting software defects is the most important part in a software quality test, often also called software quality, which aims to determine the quality of the software in meeting functional and performance requirements. Machine learning methods are better for finding software defect than manual methods. Classification algorithms in machine learning that have been used for software defect prediction include k-Nearest Neighbor (k-NN), Naïve Bayes (NB) dan Decision Tree (CART). In this study, we will compare the performance between classification algorithms, namely k-NN, NB, and CART for software defect prediction of CRISP-DM. CRISP-DM is a data mining process model with six stages, namely: Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, and Deployment in determining the comparison of classification algorithms in predicting software defects. Software Matrix used in this study is seven datasets from NASA MDP. The results showed that the average value of the CART algorithm is better than the k-NN and NB algorithms with a value of 0.867. While the average value of the accuracy of the k-NN and NB algorithms is 0.859 and 0.778, respectively.

Keywords: *Prediction of Software Defects, k-Nearest Neighbor, Naïve Bayes, CART, CRISP-DM, Software engineering, Data Mining, Machine Learning*

DOI: 10.34128/jsi.v7i2.313 *Received: June 7th, 2021*
Accepted: September 2nd, 2021

1. PENDAHULUAN

Software quality assurance meningkat secara drastis [1] seiring dengan meningkatnya kebutuhan *software* untuk kebutuhan industri dan akademik [2]. Bagian terpenting dalam *software* quality adalah prediksi cacat *software* [3]. Prediksi cacat *software* memanfaatkan pengukuran matriks pengujian *software* untuk dilakukan klasifikasi yang dapat memperkirakan kualitas modul program. Secara umum hasil pengujian *software* dibagi menjadi dua kelas, yaitu *software* rentan cacat dan *software* tidak cacat [4].

Kerugian dari segi biaya dan waktu dapat ditimbulkan jika *software* yang dibangun mengandung cacat *software* [5]. Estimasi biaya sejumlah $60 billion dikeluarkan karena sistem mengandung cacat *software* dalam studi National Institute of Standards and Technology [6]. Selain itu menurut Jones [7], biaya untuk memperbaiki cacat *software* pada tahapan desain bisa mencapai 60 kali lipat dari biaya pembuatan *software* dan apabila cacat *software* ditemukan saat *software* sudah dirilis maka biaya perbaikan cacat *software* bisa mencapai 100 kali lipat dari biaya pembuatan *software*.

Metode klasifikasi *machine learning* adalah pendekatan paling populer untuk prediksi cacat *software* [8]. Metode *machine learning* mempunyai kinerja lebih baik untuk menemukan cacat *software* daripada metode manual, seperti review *source code* manual [9]. Prediksi cacat *software* dengan metode *machine learning* menghasilkan *software* lebih berkualitas. *Software* berkualitas diartikan sebagai *software* dengan lebih sedikit modul cacat [10].

Algoritma klasifikasi dalam *machine learning* yang pernah digunakan untuk prediksi cacat *software* antara lain: k-Nearest Neighbor [11], Naïve Bayes [4], [12], dan Decision Tree [13]–[15]. Algoritma k-NN merupakan klasifikasi berdasarkan kemiripan antara data training dan data testing [16] dengan penghitungan Euclidean *distance* [17]. Algoritma Naïve Bayes adalah salah satu algoritma klasifikasi berdasarkan teorema Bayesian pada statistika [18] yang digunakan untuk memprediksi probabilitas keanggotaaan suatu kelas. Sedangkan algoritma Decision Tree berdasarkan penghitungan split dan entropy, pola yang terbentuk dari algoritma Decision Tree berbentuk pohon keputusan [19].

Dataset NASA MDP adalah salah satu *software* metrix yang digunakan oleh para peneliti untuk prediksi cacat *software* [3], [20]. Dataset NASA MDP adalah metriks berbasis modul, dimana modul adalah unit terkecil dari fungsionalitas. Modul dalam dataset NASA MDP dikumpulkan dari atribut kode statis yang didefinisikan oleh McCabe dan Halstead sebagai perdiktor cacat dalam prediksi cacat *software* [21]. Dataset NASA MDP bersifat publik sehingga mudah digunakan, dapat digeneralisasi, dan dapat diuji ulang [22].

Dalam penelitian ini akan dibandingkan kinerja antara algoritma-algoritma klasifikiasi yaitu k-Nearest Neighbor, Naïve Bayes, dan Decision Tree (CART) untuk klasifikasi dalam bidang prediksi cacat *software* dengan digunakan dataset NASA MDP sebagai obyek penelitian. Tujuan penelitian ini adalah mengetahui algoritma klasifikasi terbaik untuk prediksi cacat *software* dengan pendekatan CRISP-DM (Cross Industry Standard Process Data Mining).

2. TINJAUAN PUSTAKA

2.1 Algoritma k-Nearest Neighbor

Algoritma k-Nearest Neighbor (k-NN) digunakan untuk melakukan klasifikasi terhadap objek berdasarkan pembelajaran yang jaraknya paling dekat dengan objek [23]. Pengukuran jarak terdekat pada penelitian ini digunakan metode Euclidean *distance* [24]. Formula penghitungan Euclidean *distance* dapat dilihat pada persamaan 1. Dimana $d(x, y)$ adalah jarak antara data x ke data y, x_i adalah data testing ke-i, dan y_i adalah data training ke-i.

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \quad (1)$$

Perhatikan penggunaan bahasa. Gunakan Bahasa Indonesia yang baku untuk ragam ilmiah. Jika Anda menggunakan istilah asing yang belum diserap ke dalam Bahasa Indonesia, tuliskan *italic* (miring). Jika istilah tersebut sudah terserap ke dalam Bahasa Indonesia atau sudah lazim di dunia informatika, seperti monitor, tidak perlu Anda tulis miring.

2.2 Algoritma Naïve Bayes

Algoritma Naïve Bayes adalah salah satu algoritma klasifikasi berdasarkan teorema Bayesian pada statistika [5]. Algoritma Naïve Bayes dapat digunakan untuk memprediksi probabilitas keanggotaaan suatu kelas [25]. Teorema Bayesian menghitung nilai posterior probability $P(H|X)$ menggunakan probabilitas $P(H)$, $P(X)$, dan $P(X|H)$ [26]. Penghitungan algoritma Naïve Bayes untuk tipe data nominal menggunakan persamaan 2. Apabila dataset bertipe numerik, maka digunakan penghitungan distribusi Gaussian [27]. Penghitungan distribusi Gaussian dapat dilihat dari persamaan 3, dimana dihitung terlebih dahulu nilai rata-rata μ sesuai persamaan 4, dan standard deviasi σ sesuai persamaan 5.

$$P(H|X) = \frac{P(X|H)\ .P(H)}{P(X)} \tag{2}$$

$$g(x,\mu,\sigma) = \frac{1}{\sqrt{2\pi}.\sigma} exp^{\frac{-(x-\mu)^2}{2\sigma^2}} \tag{3}$$

$$\mu = \frac{\sum_i^n x_i}{n} \tag{4}$$

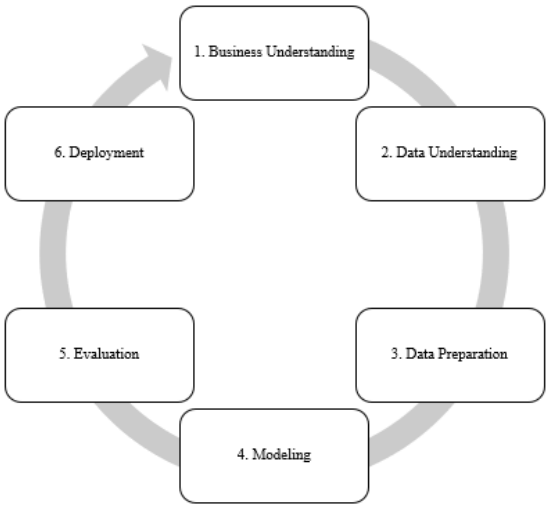
$$\sigma = \sqrt{\frac{\sum_i^n (x_i-\mu)^2}{n-1}} \tag{5}$$

2.3 Algoritma CART (Decision Tree)

Algoritma CART menggunakan penghitungan $IndexGini$ untuk pembentukan cabang. Sedangkan untuk pembentukan node, pada algoritma CART digunakan penghitungan $GiniGain$. Formula penghitungan $IndexGini$ dan $Gini\ Gain$ dapat dilihat pada persamaan 6 dan 7. Pola yang terbentuk dari algoritma CART berbentuk pohon keputusan (*decision tree*).

$$IndexGini = 1 - \sum_{i=1}^{k} p_i^2 \tag{6}$$

$$Gini\ Gain = Gini(A,S) - \sum_{i=1}^{n} \frac{|S_i|}{|S|} \times Gini(S_i) \tag{7}$$



Gambar 1. Tahapan CRISP-DM

2.4 Cross Industry Standard Process Data Mining (CRISP-DM)

Dalam penelitian ini digunakan pendekatan CRISP-DM. Gambar 1, menunjukkan 6 (enam) tahapan dalam CRIPS-DM. Tahapan CRISP-DM terdiri dari enam tahapan, yaitu: *business understanding, data understanding, data preparation, modeling, evaluation*, dan *deployment* [28], [29]. Tahapan *business understanding* berisi tentang menentukan tujuan bisnis, menilai situasi saat ini, dan menetapkan tujuan dilakukan data mining. *Data understanding*, adalah kegiatan persiapan, mengevaluasi persyaratan data, dan termasuk pengumpulan data. *Data preparation*, setelah data dikumpulkan, data-data tersebut perlu diidentifikasi, data perlu dipilih, dibersihkan, dan kemudian data tersebut dibangun ke dalam bentuk/format yang diinginkan. Data preparation, disebut juga dengan data preprocessing. *Modeling* adalah bentuk flowchart atau aplikasi dari algoritma untuk mencari, mengidentifkasi, dan menampilkan pola. *Evaluation*, digunakan untuk membantu pengukuran evaluasi pada model. Kita bisa mengukur model mana yang paling baik digunakan untuk proses data mining. Tahapan *deployment* digunakan untuk melakukan otomatisasi model atau pengembangan aplikasi, terintegrasi dengan sistem informasi manajemen atau operasional yang ada.

3. METODE PENELITIAN

3.1 *Business Understanding*

Tujuan penelitian ini adalah pemilihan model terbaik algoritma klasifikasi untuk prediksi cacat *software* pada *dataset* NASA MDP. Algoritma klasifikasi yang digunakan sebagai perbandingan adalah algoritma k-NN, NB, dan CART (decision tree). Metodologi penelitian yang digunakan dalam penelitian ini adalah metodologi eksperimen dengan pendekatan CRISP-DM. Menurut Dawson [22], penelitian eksprimen adalah uji coba yang dikontrol oleh peneliti sendiri untuk melakukan investigasi hubungan kausal (hubungan sebab-akibat). Eksperimen ini dilakukan dengan program bantu Rapidminer versi 7.4, Microsoft Excel 2013, SPSS versi 16, Geany versi 1.34.1, Google Chrome versi 73.0.3683.103, dan XAMPP versi 7.3.2.

3.2 *Data Understanding*

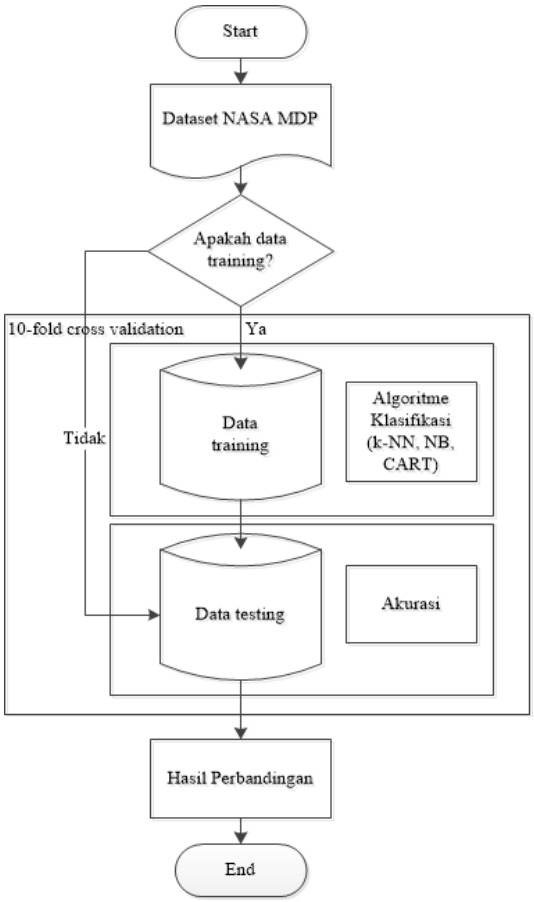
Dataset yang digunakan pada penelitian ini adalah tujuh *dataset* dari NASA MDP. Ketujuh *dataset* NASA MDP yang digunakan antara lain CM1, JM1, KC1, KC3, PC3, PC4 dan PC5. Tabel 1 menunjukkan spesifikasi dan atribut *dataset* NASA MDP. Spesifikasi dari attribut dataset NASA dibagi dalam 4 bagian yaitu: LOC count, attribut halstead, attribut McCabe, dan attribut miscellaneous. LOC count adalah jumlah baris kode dan komentar dari program. Attribut halsted adalah perhitungan operan 32 dan operator dalam program. Attribut McCabe adalah kompleksitas cyclomatic dalam program.

3.3 *Data Preparation*

Dataset NASA MDP yang digunakan pada penelitian ini telah dilakukan transformasi data. Metode transformasi data yang digunakan adalah teknik Min-Max. Hasil dari transformasi data yang digunakan pada penelitian ini dapat diunduh di https://github.com/klainfo/NASADefectDataset.

Tabel 1. Spesifikasi dan deskripsi *dataset* NASA MDP

Dataset	Jumlah Attribut	Modul	Sistem	Bahasa
CM1	38	327	Instrumen pesawat ruang angkasa	C
JM1	22	7.720	Sistem prediksi pendaratan realtime	C
KC1	22	1.162	Manajemen penyimpanan data lapangan	C++
KC3	40	194	Manajemen penyimpanan data lapangan	JAVA
MC2	40	124	Sistem panduan video	C, C++
PC1	38	679	*Software* penerbangan satelit	C
PC2	37	722	Simulator dinamis untuk sistem kontrol perilaku	C
PC3	38	1.053	*Software* penerbangan satelit	C
PC4	38	1.270	*Software* penerbangan satelit	C
PC5	39	1.694	Peningkatan keamanan sistem upgrade kokpit	C++



Gambar 2. Flowchart model yang diusulkan

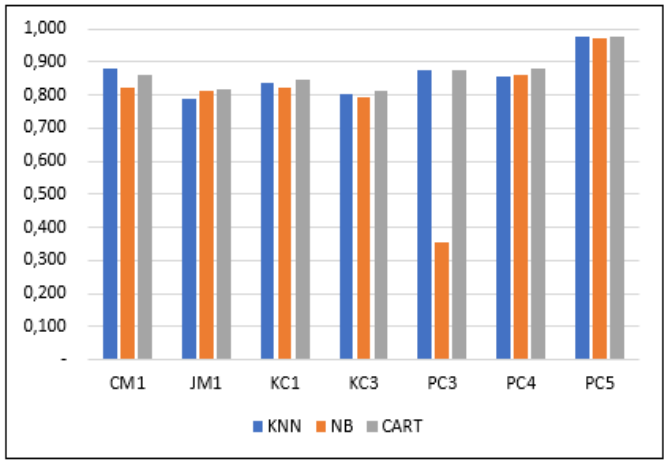
4. HASIL DAN PEMBAHASAN

Dalam penelitian ini akan dipilih model terbaik algoritma klasifikasi untuk prediksi cacat *software* pada *dataset* NASA MDP. Algoritma klasifikasi yang digunakan sebagai perbandingan adalah algoritma k-NN [11], NB [4], [12], dan CART [13]–[15]. Gambar 2 menunjukkan model yang diusulkan pada penelitian ini. Dataset NASA MDP akan dibagi menjadi dua bagian yaitu data training dan data testing menggunakan 10-f*old cross validation*. Data training digunakan untuk pemodelan data dengan algoritma k-NN, NB, dan CART. Data testing digunakan untuk pengukuran nilai akurasi masing-masing algoritma. Setelah diketahui nilai akurasi masing-masing algoritma, akan dilakukan uji validasi menggunakan penghitungan statistika, untuk mengetahui apakah masing-masing algoritma terdapat perbedaan signifikan.

Tabel 2. Nilai akurasi antar algoritma klasifikasi

No	Dataset	k-NN	NB	CART
1	CM1	**0,878**	0,823	0,861
2	JM1	0,786	0,814	**0,817**
3	KC1	0,838	0,824	**0,845**
4	KC3	0,805	0,795	**0,815**
5	PC3	**0,874**	0,355	**0,874**
6	PC4	0,858	0,859	**0,880**
7	PC5	**0,977**	0,974	**0,977**
Rata-Rata		0,859	0,778	**0,867**

p-ISSN: 2460-173X
e-ISSN: 2598-5841



Gambar 3. Diagram nilai akurasi antar algoritma klasifikasi

4.1 *Evaluation*

Evaluasi yang digunakan dalam penelitian ini adalah pengukuran hasil akurasi masing-masing algoritma klasifikasi (k-NN, NB, dan CART) untuk prediksi cacat software. Hasil akurasi pada penelitian ini dapat dilihat pada Tabel 2 dan Gambar 3, terlihat bahwa nilai rata-rata akurasi algoritma CART lebih baik daripada algoritma k-NN dan NB dengan nilai 0,867. Sedangkan nilai rata-rata akurasi algoritma k-NN dan NB masing-masing 0,859 dan 0,778.

Setelah didapatkan nilai akurasi masing-masing algoritma, langkah selanjutnya adalah dilakukan uji validasi statistik, agar diketahui adakah perbedaan signifikan antar algoritma. Uji validasi statistik yang digunakan adalah uji Friedman test. Friedman test digunakan untuk mengetahui perbedaan signifikan lebih dari dua metode [30]. Dalam penelitian ini ditetapkan nilai α sebesar 0,05, jika nilai p-value lebih besar dari nilai α maka tidak ada perbedaan signifikan antar algoritma. Sedangkan jika nilai p-value lebih kecil dari nilai α maka terdapat perbedaan signifikan antar algoritma. Tabel 3 menunjukkan hasil uji Friedman test pada penelitian ini, nilai p-value pada penelitian ini adalah sebesar 0,021 (kurang dari nilai α), sehingga dapat disimpulkan terdapat perbedaan signifikan antar algoritma.

Dalam uji Friedman test tidak diketahui algoritma mana saja yang saling berbeda secara signifikan, sehingga perlu dilakukan uji lanjutan atau uji post hoc. Uji post hoc yang digunakan pada penelitian ini adalah Nemenyi post hoc [30], [31]. Pada uji Nemenyi post hoc, secara umum kinerja kedua klasifikasi dikatakan berbeda secara signifikan jika nilai *difference* (diff) berdasarkan penghitungan *average rank* (AR) lebih kecil daripada nilai *critical difference* (CD). Tabel 4 menunjukkan nilai *average rank* antar algoritma. Sedangkan penghitungan nilai CD berdasarkan pada persamaan 8, dengan nilai K = 3, N = 7, dan q_α = 2,343, sehingga nilai CD pada penelitian ini adalah 1,252.

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}} \tag{8}$$

Setelah diketahui nilai AR antar algoritma dan nilai CD, tahapan selanjutnya adalah dilakukan penghitungan *Pairwase Comparison* berdasarkan nilai *difference* (diff). Jika nilai diff lebih besar daripada nilai CD, maka dapat disimpulkan bahwa terdapat perbedaan signifikan. Namun sebaliknya jika nilai diff lebih kecil daripada nilai CD, maka tidak terdapat perbedaan signifikan. Tabel 5 menunjukkan nilai diff penghitungan Pairwase Comparison, dan Tabel 6 menunjukkan nilai *significant difference*. Terlihat bahwa terdapat perbedaan signifikan antara algoritma NB dan algoritma CART. Namun tidak ada perbedaan antara algoritma k-NN dan algoritma NB, maupun algoritma k-NN dan algoritma CART.

p-ISSN: 2460-173X
e-ISSN: 2598-5841

Tabel 3. Hasil uji *Friedman Test* antar algoritma

N	**7**
Chi-Square	7,692
DF	2
p-value	0,021
α	**0,050**

Tabel 4. *Average Rank* antar algoritma

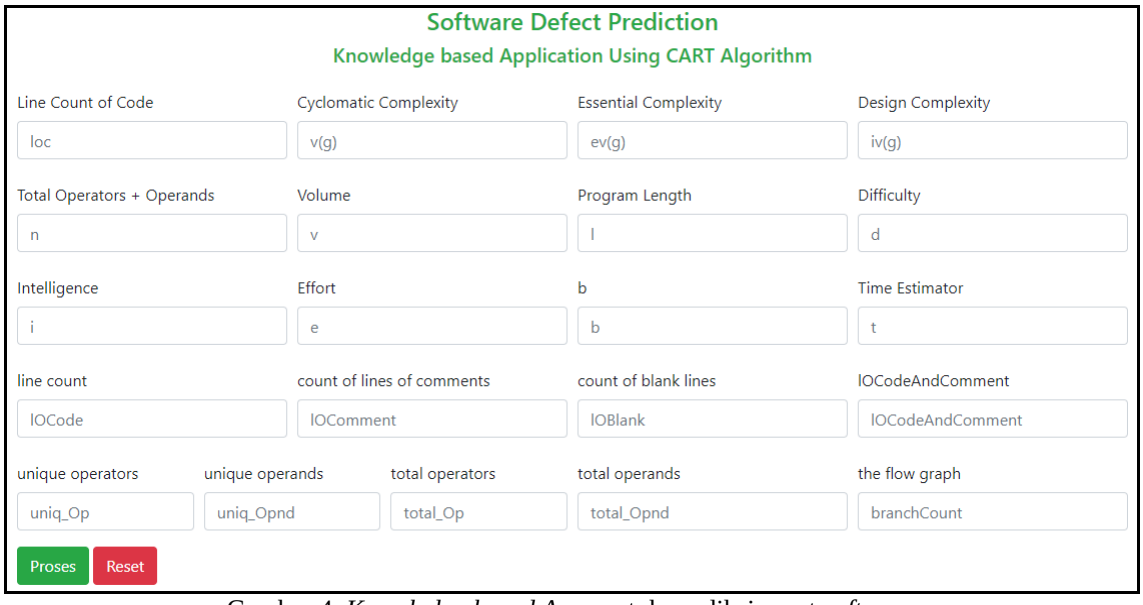
No	Dataset	k-NN	NB	CART
1	CM1	1	3	2
2	JM1	3	2	1
3	KC1	2	3	1
4	KC3	2	3	1
5	PC3	1,5	3	1,5
6	PC4	3	2	1
7	PC5	1,5	3	1,5
Average Rank		2,000	2,714	1,286

Tabel 5. *Pairwase Comparison* berdasarkan *difference*

	k-NN	NB	CART
k-NN	0,000	0,714	0,714
NB	0,714	0,000	**1,429**
CART	0,714	**1,429**	0,000
CD:	1,252		

Tabel 6. *Significant Difference*

	k-NN	NB	CART
k-NN	FALSE	FALSE	FALSE
NB	FALSE	FALSE	**TRUE**
CART	FALSE	**TRUE**	FALSE



Gambar 4. *Knowledge based Apps* untuk prediksi cacat *software*

Gambar 4 di atas menunjukan aplikasi yang dibangun dalam tahapan *deployment*. Bahasa pemrograman yang digunakan pada tahapan *deployment* adalah bahasa pemrograman PHP.

4.2 *Deployment*

Nilai rata-rata akurasi algoritma CART lebih baik daripada algoritma k-NN dan NB. Setelah dilakukan uji validasi dengan menggunakan Friedman test, didapatkan hasil terdapat perbedaan signifikan antara ketiga algoritma, dengan nilai p-value sebesar 0,021. Sehingga pada tahapan deployment akan dibangun *knowledge* based application untuk prediksi cacat *software* menggunakan algoritma CART.

5. KESIMPULAN

Hasil penelitian menunjukkan nilai rata-rata akurasi algoritma CART lebih baik daripada algoritma k-NN dan NB dengan nilai 0,867. Sedangkan nilai rata-rata akurasi algoritma k-NN dan NB masing-masing 0,859 dan 0,778. Terdapat perbedaan secara signifikan antara ketiga algoritma ketika diuji Friedman test dengan nilai p-value sebesar 0,021. Penelitian ini menghasilkan kesimpulan bahwa pemodelan algoritma CART memiliki nilai akurasi terbaik untuk prediksi cacat software.

DAFTAR PUSTAKA

[1] T. M. Khoshgoftaar and E. B. Allen, "A practical classification-rule for software-quality models," *IEEE Trans. Reliab.*, vol. 49, no. 2, pp. 209–216, 2000, doi: 10.1109/24.877340.

[2] I. H. Laradji, M. Alshayeb, and L. Ghouti, "Software defect prediction using ensemble learning on selected features," *Inf. Softw. Technol.*, vol. 58, pp. 388–402, 2015, doi: 10.1016/j.infsof.2014.07.005.

[3] K. Gao, T. M. Khoshgoftaar, and R. Wald, "Combining Feature Selection and Ensemble Learning for Software Quality Estimation," in *The Twenty-Seventh International Flairs Conference*, 2014, pp. 47–52.

[4] Q. Song, Z. Jia, M. Shepperd, S. Ying, and J. Liu, "A General Software Defect-Proneness Prediction Framework," *IEEE Trans. Softw. Eng.*, vol. 37, no. 3, pp. 356–370, 2010.

[5] J. Suntoro, F. W. Christanto, and H. Indriyawati, "Software Defect Prediction Using AWEIG + ADACOST Bayesian Algorithm for Handling High Dimensional Data and Class Imbalanced Problem," *Int. J. Inf. Technol. Bus.*, vol. 1, no. 1, pp. 36–41, 2018.

[6] J. D. Strate and P. A. Laplante, "A Literature Review of Research in Software Defect Reporting," *IEEE Trans. Reliab.*, vol. 62, no. 2, pp. 444–454, 2013.

[7] C. Jones and O. Bonsignour, *The Economics of Software Quality*. Boston: Pearson Education, Inc., 2012.

[8] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, "Benchmarking Classification Models for Software Defect Prediction: A Proposed Framework and Novel Findings," *IEEE Trans. Softw. Eng.*, vol. 34, no. 4, pp. 485–496, 2008.

[9] T. Menzies, Z. Milton, B. Turhan, B. Cukic, Y. Jiang, and A. Bener, "Defect prediction from static code features: current results, limitations, new approaches," *Autom. Softw. Eng.*, vol. 17, no. 4, pp. 375–407, 2010.

[10] Ö. F. Arar and K. Ayan, "Software defect prediction using cost-sensitive neural network," *Appl. Soft Comput.*, vol. 33, pp. 263–277, 2015, doi: 10.1016/j.asoc.2015.04.045.

[11] B. Turhan, T. Menzies, A. B. Bener, and J. Di Stefano, "On the relative value of cross-company and within-company data for defect prediction," *Empir. Softw. Eng.*, vol. 14, no. 5, pp. 540–578, 2009, doi: 10.1007/s10664-008-9103-7.

[12] C. Catal and B. Diri, "Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem," *Inf. Sci. (Ny).*, vol. 179, no. 8, pp. 1040–1058, 2009, doi: 10.1016/j.ins.2008.12.001.

[13] E. Arisholm, L. C. Briand, and E. B. Johannessen, "A systematic and comprehensive

investigation of methods to build and evaluate fault prediction models," *J. Syst. Softw.*, vol. 83, no. 1, pp. 2–17, 2010, doi: 10.1016/j.jss.2009.06.055.
[14] N. Gayatri, S. Nickolas, and A. V. Reddy, "Feature Selection Using Decision Tree Induction in Class level Metrics Dataset for Software Defect Predictions," 2010.
[15] Z. Sun, Q. Song, and X. Zhu, "Using coding-based ensemble learning to improve software defect prediction," *IEEE Trans. Syst. Man, Cybern. Part C (Applications Rev.*, vol. 42, no. 6, pp. 1806–1817, 2012, doi: 10.1109/TSMCC.2012.2226152.
[16] V. Kotu and B. Deshpande, *Predictive Analytics and Data Mining: Concepts and Practice with Rapidminer*. Waltham: Morgan Kaufmann Publishers is an imprint of Elsevier, 2015.
[17] H. Liu and S. Zhang, "Noisy data elimination using mutual k-nearest neighbor for classification mining," *J. Syst. Softw.*, vol. 85, no. 5, pp. 1067–1074, 2012, doi: 10.1016/j.jss.2011.12.019.
[18] J. Suntoro, *Data Mining Algoritma dan Implementasi Menggunakan Bahasa Pemrograman PHP*. Jakarta: Elex Media Komputindo, 2019.
[19] J. Suntoro and C. N. Indah, "Average Weight Information Gain Untuk Menangani Data Berdimensi," *J. Buana Inform.*, vol. 8, no. 3, pp. 131–140, 2017.
[20] R. S. Wahono, N. Suryana, and S. Ahmad, "Metaheuristic Optimization based Feature Selection for Software Defect Prediction," *J. Softw.*, vol. 9, no. 5, pp. 1324–1333, 2014, doi: 10.4304/jsw.9.5.1324-1333.
[21] R. S. Wahono, N. S. Herman, and S. Ahmad, "A Comparison Framework of Classification Models for Software Defect Prediction," *Adv. Sci. Lett.*, vol. 20, no. 10–11, pp. 1945–1950, 2014, doi: 10.1166/asl.2014.5640.
[22] C. Dawson, *Projects in Computing and Information Systems*. Canada: Pearson Education, 2009.
[23] B. Lantz, *Machine Learning with R*. Birmingham: Packt Publishing Ltd, 2013.
[24] A. A. Aburomman and M. B. I. Reaz, "A novel SVM-kNN-PSO ensemble method for intrusion detection system," *Appl. Soft Comput.*, vol. 38, pp. 360–372, 2016, doi: 10.1016/j.asoc.2015.10.011.
[25] J. Han, M. Kamber, and J. Pei, *Data Mining Concepts and Techniques Third Edition*. Waltham: Morgan Kaufmann Publishers is an imprint of Elsevier, 2012.
[26] M. Kantardzic, *Data Mining: Concepts, Models, Methods, and Algorithms Second Edition*. Canada: Wiley-IEEE Press, 2011.
[27] D. Ryu and J. Baik, "Effective multi-objective naïve Bayes learning for cross-project defect prediction," *Appl. Soft Comput.*, vol. 49, pp. 1062–1077, 2016, doi: 10.1016/j.asoc.2016.04.009.
[28] J. Suntoro, A. Ilham, and H. A. D. Rani, "New Method Based Pre-Processing to Tackle Missing and High Dimensional Data of CRISP-DM Approach," *J. Phys. Conf. Ser.*, vol. 1471, no. 1, 2020.
[29] M. North, *Data Mining for the Masses*. Global Text Project, 2012.
[30] J. Demšar, "Statistical Comparisons of Classifiers over Multiple Data Sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, 2006, doi: 10.1016/j.jecp.2010.03.005.
[31] H.-L. Dai, "Class imbalance learning via a fuzzy total margin based support vector machine," *Appl. Soft Comput.*, vol. 31, pp. 172–184, 2015, doi: 10.1016/j.asoc.2015.02.025.

Biodata Penulis

Nurtriana Hidayati lahir di Rembang, Jawa Tengah pada 22 Desember 1983 adalah seorang dosen sebuah perguruan tinggi swasta di kota Semarang, Jawa Tengah dengan bernama Universitas Semarang pada program studi S1 Sistem Informasi yang bernaung pada jurusan Teknologi Informasi dalam Fakultas Teknologi Informasi dan Komunikasi (FTIK). Sejak 2010 sudah selama 10 tahun ini mengampu pada bidang ilmu sesuai dengan pendidikan terakhir S2 pada Megister Sistem Informasi Universitas Diponegoro (UNDIP) dengan gelar M.Kom. Sistem

Informasi memiliki ruang lingkup ilmu yang sangat penting sehingga selain mengajar pada program studi sistem informasi juga mengampu pada program studi S1 Teknik Informastika. Untuk menciptakan lulusan mahasiswa jurusan TI saya mengajarkan beberapa matakuliah diantaranya analisa dan perancangan sistem informasi, basis data, data mining dan rekayasa perangkat lunak.

Joko Suntoro lahir di Semarang pada tanggal 31 Juli 1989. Penulis sekarang bekerja sebagai pengajar di Jurusan Teknologi Informasi Universitas Semarang. Beberapa matakuliah yang diampu penulis diantaranya data mining, data warehouse, arsitektur perushaan, rekayasa perangkat lunak dan pengujian sistem. Penulis Memperoleh gelar S.Kom. pada jurusan Teknik Informatika di Universitas Semarang tahun 2015, dan gelar M.Kom. pada jurusan Magister Teknik Informatika di Universitas Dian Nuswantoro tahun 2016. Selain aktif sebagai pengajar, penulis juga aktif melakukan penelitian.

Galet Guntoro Setiaji, lahir di Semarang 10 April 1983, Penulis bekerja di UPT. Pangkalan Data Universitas Semarang dan juga mengajar di Fakultas Teknologi Informasi Universitas Semarang, Beberapa matakuliah yang diampu adalah pemrograman web dan pemrograman web framework, Meraih gelar Sarjana Komputer (S.Kom) di Universitas Stikubank Semarang tahun 2006, Kemudian mendapatkan gelar Magister Komputer (M.Kom) di Universitas Dian Nuswantoro tahun 2016. Saat ini bekerja sebagai dosen Program Studi Teknik Informatika Universitas Semarang.