# PENGKOMBINASIAN POHON KEPUTUSAN DATA PENCILAN KELAS DAN POHON KEPUTUSAN DATA NORMAL UNTUK PENINGKATAN AKURASI PREDIKSI CACAT PERANGKAT LUNAK

**Utomo Pujianto[1], Daniel Oranova Siahaan[2]**
[1]Jurusan Teknik Informatika, Fakultas Teknik, Universitas Muhammadiyah Gresik
Jl. Sumatra 101, Gresik Kota Baru, Gresik
[2]Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember
Kampus ITS Sukolilo, Surabaya
Email: utomo.pujianto@yahoo.co.id[1], danielos@cs.its.ac.id[2]

**ABSTRAK**

*Keberadaan pencilan dalam set data prediksi cacat perangkat lunak memunculkan dua pilihan penanganan, yaitu apakah tetap digunakan dengan mengabaikan keberadaannya sebagai pencilan, ataukah dihapuskan dari set data uji coba. Pengabaian keberadaan pencilan dalam set data uji menghasilkan akurasi hasil prediksi yang secara signifikan lebih rendah dibandingkan dengan jika pencilan tersebut dihapuskan dari set data uji coba. Dalam paper ini diusulkan utilisasi pencilan kelas dalam set data prediksi cacat perangkat lunak untuk meningkatkan kinerja sistem prediksi cacat perangkat lunak. Metode yang diusulkan terdiri dari proses prediksi pencilan kontekstual dari set data uji, menggunakan model prediksi berbasis pohon keputusan tunggal. Selanjutnya dilakukan dengan proses prediksi label kelas cacat perangkat lunak menggunakan dua buah model prediksi berbasis pohon keputusan tunggal, yang masing-masing berfungsi untuk melakukan prediksi cacat perangkat lunak terhadap subset data normal dan subset data pencilan. Hasil pengujian terhadap lima set data NASA dari repository PROMISE menunjukkan bahwa metode yang diusulkan memiliki kinerja akurasi yang lebih baik dibandingkan penggunaan algoritma J48 yang mengabaikan keberadaan pencilan, maupun yang menghapuskan pencilan dari set data sampel pengujian.*

***Kata Kunci:*** *Prediksi Cacat Perangkat Lunak, Pohon Keputusan, Pencilan*.

## 1. PENDAHULUAN

Dalam setiap siklus pengembangan perangkat lunak, pengujian perangkat lunak merupakan salah satu fase yang banyak menyita sumber daya utama proyek, yaitu waktu, biaya dan tenaga. Salah satu cara yang dapat ditempuh untuk dapat mengoptimalkan sumber daya yang ada adalah memprioritaskan pengalokasikan sumber daya proyek hanya pada modul-modul perangkat lunak yang memiliki potensi besar mengandung cacat (*defect*). Untuk itulah diperlukan suatu metode untuk dapat memprediksi apakah sebuah modul perangkat lunak berpotensi mengandung cacat atau tidak.

Permasalahan prediksi cacat perangkat lunak termasuk dalam ranah permasalahan klasifikasi. Oleh karena itulah sejumlah penelitian sebelumnya mengusulkan metode-metode klasifikasi untuk menghasilkan sistem prediksi cacat perangkat lunak dengan kinerja terbaik. Sebagian metode yang diusulkan berasal dari metode-metode klasifikasi dalam domain pembelajaran mesin (*machine learning*), seperti misalnya *expectation–maximization*, *naïve bayes*, dan *random forests* [8]. Dari sekian banyak metode yang pernah diusulkan, sejumlah metode berbasis pohon keputusan, baik pohon keputusan tunggal maupun ansambel pohon keputusan, menunjukkan kinerja yang sangat baik seperti misalnya ditunjukkan dalam [2], [6], dan [12].

Akan tetapi, kinerja yang baik dari metode-metode berbasis pohon keputusan juga belum dapat dimaksimalkan, karena masih dipengaruhi oleh adanya pencilan dalam sampel data penelitian prediksi cacat perangkat lunak. Dua penelitian yang membahas tentang mekanisme deteksi dan penanganan terhadap pencilan adalah [1] dan [2]. Dalam kedua penelitian tersebut, digunakan sejumlah nilai ambang (*threshold*) yang bersifat statis dari atribut-atribut data untuk membedakan antara data normal dan data pencilan.

Karena menggunakan nilai ambang yang bersifat statis, maka model-model prediksi yang dihasilkan dari penelitian tersebut tidak cukup fleksibel untuk dapat diterapkan dalam kasus-kasus set data prediksi cacat perangkat lunak lainnya. Hal ini disebabkan masing-masing proyek perangkat lunak memiliki karakteristik tersendiri yang belum tentu sesuai dengan karakteristik set data yang dijadikan sampel dalam dua penelitian tersebut.

Permasalahan utilisasi sampel data penelitian juga perlu mendapatkan perhatian dari kedua penelitian tersebut. Karena pencilan yang telah dapat

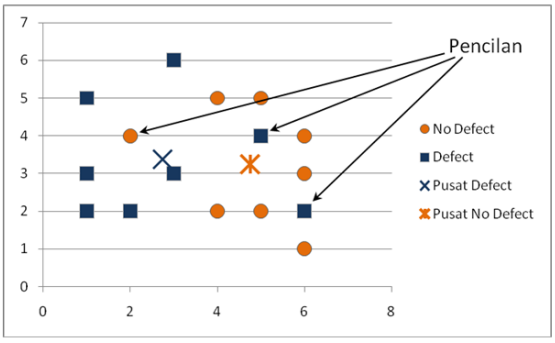
dideteksi tidak digunakan lagi dalam proses pembangunan model prediksi cacat perangkat lunak dengan metode-metode yang dipilih, yaitu *naïve bayes*, J48, dan *random forest*. Hal ini tentunya mengurangi tingkat utilisasi data dari sampel data penelitian. Selain itu, tidak ada seorangpun yang dapat menjamin bahwa kasus-kasus khusus yang saat itu tercatat sebagai pencilan dalam set data penelitian, tidak dapat terjadi kembali di masa yang akan datang sebagai sebuah pola yang kuat.

Walaupun hingga saat ini masih terjadi perdebatan tentang perlu-tidaknya mengikutsertakan data pencilan dalam proses analisis data, resiko tersebut sudah selayaknya dihindari. Sehingga hal ini merupakan tantangan bagi para peneliti untuk dapat meningkatkan kinerja prediksi cacat perangkat lunak, dengan tetap memaksimalkan utilisasi sampel data penelitian, yang di dalamnya juga terkandung pencilan. Hal ini didukung oleh Boetteicher [3], yang menyatakan bahwa pemilihan data pelatihan dan pengujian yang tepat akan dapat meningkatkan kinerja metode prediksi cacat perangkat lunak. Dalam paper ini diusulkan sebuah metode ansambel pohon keputusan untuk meningkatkan kinerja sistem prediksi cacat perangkat lunak dengan tetap memaksimalkan utilisasi data sampel penelitian.

## 2. MODEL, ANALISIS, DESAIN, DAN IMPLEMENTASI

Secara garis besar, metode prediksi cacat perangkat lunak yang diusulkan dalam paper ini terdiri dari dua bagian penting, yaitu prediksi pencilan dan prediksi cacat perangkat lunak. Kedua macam proses prediksi tersebut dilakukan dengan dua tahapan, yaitu tahapan induksi dan tahapan deduksi. Tahapan pertama, yaitu tahapan induksi dilakukan dengan langkah-langkah sebagai berikut:

1. Melakukan normalisasi data.
2. Menentukan *centroid* dari masing-masing klaster kelas. Karena masing-masing set data yang digunakan dalam penelitian ini hanya memiliki dua macam kelas, yaitu kelas label *defect = true* dan kelas label *defect = false*, maka penentuan *centroid* dilakukan dengan menghitung nilai rata-rata untuk masing-masing atribut dari seluruh anggota tiap klaster kelas.
3. Menghitung jarak *euclidean* terhadap kedua buah *centroid* untuk setiap data dalam set data. Pencilan kontekstual diperoleh dengan membandingkan nilai jarak masing-masing data terhadap *centroid* klaster kelas *defect = true* dan *centroid* klaster kelas *defect = false*. Sebuah data akan dinyatakan sebagai pencilan kontekstual dari kelas *defect = true* jika jarak *euclidean* data tersebut terhadap *centroid* klaster kelas *defect = true* lebih besar dari jarak euclidean data tersebut terhadap *centroid* klaster kelas *defect = false*. Begitu juga sebaliknya, data akan dinyatakan sebagai pencilan kontekstual dari kelas *defect = false* jika jarak *euclidean* data tersebut terhadap *centroid* klaster kelas *defect = false* lebih besar dari jarak *euclidean* data tersebut terhadap *centroid* klaster kelas *defect = true*. Ilustrasi dari penentuan pencilan kontekstual berdasarkan *centroid* kelas dapat dilihat pada Gambar 1.
4. a) Berdasarkan langkah ketiga, dilakukan pembangunan dua buah model prediksi berbentuk pohon keputusan. Pohon keputusan yang pertama dibangun dari sub set data yang diidentifikasi sebagai pencilan. Sedangkan pohon keputusan yang kedua dibangun dari sub set data yang diidentifikasi sebagai data normal. Setiap pohon keputusan dibangun menggunakan teknik induksi dalam algoritma J48, yang merupakan implementasi algoritma C4.5 dalam bahasa pemrograman Java.
   b) Berdasarkan langkah ketiga, dilakukan pembentukan set data baru hasil modifikasi set data sebelumnya, dengan menambahkan atribut baru, yaitu atribut pencilan. Atribut ini bertipe nominal, dengan pilihan nilai "*yes*" jika data tersebut adalah pencilan dan bernilai "*no*" jika bukan pencilan. Set data yang telah dimodifikasi kemudian digunakan untuk membangun model pohon keputusan yang nantinya akan digunakan dalam proses prediksi pencilan. Pohon keputusan untuk keperluan prediksi pencilan ini juga dihasilkan dengan metode induksi dari algoritma J48.




**Gambar 1.** Ilustrasi Deteksi Pencilan Berbasis Centroid Kelas

Tahapan deduksi terdiri dari dua proses utama, yaitu prediksi pencilan dan prediksi cacat perangkat lunak. Proses prediksi pencilan dilakukan dengan menggunakan pohon keputusan yang dihasilkan dari langkah 4b tahapan induksi. Keluaran yang dihasilkan dari proses prediksi pencilan kemudian dibagi menjadi dua kategori, yaitu data uji yang diprediksi sebagai pencilan dan data uji yang diprediksi sebagai data normal. Kedua sub set data

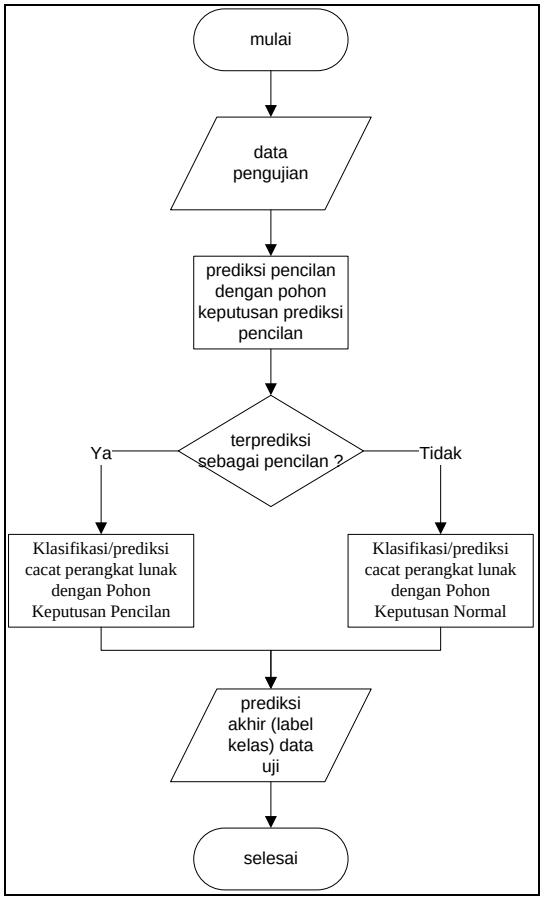
uji tersebut akan dijadikan sebagai masukan untuk proses prediksi cacat perangkat lunak. Data-data uji yang diprediksi sebagai pencilan kemudian diproses menggunakan pohon keputusan 'pencilan' yang dihasilkan dari langkah 4a tahapan induksi. Sedangkan data-data uji yang diprediksi sebagai data normal diproses menggunakan pohon keputusan normal yang juga dihasilkan dari langkah 4a tahapan induksi. Keluaran dari proses prediksi dengan pohon keputusan normal dan pohon keputusan pencilan adalah prediksi label kelas *defect* dari masing-masing data uji.


**Gambar 2.** Tahapan Induksi dari metode prediksi cacat perangkat lunak yang diusulkan

## 3. SKENARIO UJI COBA

### 3.1 Set Data Uji Coba

Uji coba dalam penelitian ini menggunakan lima buah set data dari tim *Metrics Data Program* NASA yang tersimpan dalam repositori PROMISE. Kelima set data tersebut adalah CM1, JM1, KC1, KC2 dan PC1. Seluruh set data dalam uji coba memiliki daftar atribut berupa metrik perangkat lunak yang sama, yaitu dua puluh satu atribut bertipe kontinyu, ditambah satu atribut yang menjadi atribut kelas (*defect*). Tabel 1 memperlihatkan sejumlah informasi penting dari lima set data yang dipakai.




**Gambar 3.** Tahapan Deduksi dari metode prediksi cacat perangkat lunak yang diusulkan

**Tabel 1.** Informasi umum set data yang digunakan dalam uji coba.

| Set Data | Jumlah Data | | |
|---|---|---|---|
| | **Label kelas bernilai *false*** | **Label kelas bernilai *true*** | **Total** |
| CM1 | 449 | 49 | 498 |
| JM1 | 8779 | 2106 | 10885 |
| KC1 | 1783 | 326 | 2109 |
| KC2 | 415 | 107 | 522 |
| PC1 | 1032 | 77 | 1109 |

### 3.2 Metode Uji Coba

Metode Uji yang digunakan dalam penelitian ini adalah *Full Train Full Test* dan *Ten Fold Cross Validation*. Dengan *Full Train Full Test*, seluruh data dalam setiap set data digunakan sebagai data pelatihan sekaligus sebagai data uji. Sedangkan dengan *Ten Fold Cross Validation*, masing-masing set data akan dibagi menjadi sepuluh bagian, dimana

masing-masing bagian secara bergiliran akan digunakan sebagai data uji terhadap sembilan bagian lainnya yang digunakan sebagai data pelatihan.

Uji coba dalam penelitian ini dilakukan dalam dua macam skenario. Dalam Skenario I, data pencilan kelas, yang ditemukan menggunakan deteksi pencilan berbasis *centroid*, tidak digunakan dalam proses prediksi cacat perangkat lunak. Akurasi prediksi cacat perangkat lunak yang dihasilkan kemudian dibandingkan prediksi cacat perangkat lunak yang juga melakukan eksklusi pencilan kelas, yaitu yang ditemukan menggunakan deteksi pencilan berbasis *threshold* statis [2]. Sedangkan dalam Skenario II, data pencilan kelas, yang telah diidentifikasi menggunakan deteksi pencilan berbasis *centroid*, digunakan sebagai basis untuk membentuk model prediksi cacat perangkat lunak berupa pohon keputusan.

## 4. HASIL UJI COBA

**Tabel 2** Hasil Uji Coba Skenario I dengan Metode Uji *Full Train Full Test*

| Set Data | Deteksi Pencilan Berbasis Threshold Statis | | Deteksi Pencilan Berbasis Centroid | |
|---|---|---|---|---|
| | Akurasi J48 | Akurasi RF | Akurasi J48 | Akurasi RF |
| CM1 | 97,40% | 100% | 100% | 100% |
| JM1 | 94,82% | 99,71% | 99,55% | 99,99% |
| KC1 | 98,54% | 99,95% | 99,76% | 100% |
| KC2 | 98,71% | 100% | 99,29% | 100% |
| PC1 | 97,63% | 99,59% | 99,89% | 100% |

Ket: RF = *Random Forests*.

**Tabel 3** Hasil Uji Coba Skenario I dengan Metode Uji *Ten Fold Cross Validation*

| Set Data | Deteksi Pencilan Berbasis Threshold Statis | | Deteksi Pencilan Berbasis Centroid | |
|---|---|---|---|---|
| | Akurasi J48 | Akurasi RF | Akurasi J48 | Akurasi RF |
| CM1 | 94,80% | 96,22% | 97,71% | 98,98% |
| JM1 | 92,07% | 92,39% | 98,84% | 99,16% |
| KC1 | 96,81% | 97,29% | 99,40% | 99,52% |
| KC2 | 92,92% | 93,56% | 97,38% | 97,38% |
| PC1 | 96,81% | 97,01 | 99,34% | 99,45% |

Ket: RF = *Random Forests*.

Tabel 2 dan Tabel 3 memperlihatkan perbandingan akurasi prediksi cacat perangkat lunak antara dua metode sebagai hasil dari uji coba skenario I. Hasil pengujian dari Uji Coba Skenario I menunjukkan bahwa eksklusi data pencilan dimana data pencilan kelas dideteksi dengan menggunakan deteksi berbasis *centroid* menghasilkan prediksi cacat perangkat lunak yang lebih baik saat menggunakan algoritma J48 dan *Random Forests* dibandingkan jika data pencilan kelas dideteksi dengan menggunakan deteksi pencilan berbasis nilai ambang statis [2]. Sedangkan hasil dari Uji Coba Skenario II, seperti ditunjukkan pada Tabel 4 dan Tabel 5 memperlihatkan bahwa utilisasi pencilan kontesktual menghasilkan kinerja yang masih lebih baik jika dibandingkan dengan metode J48 dan *Random Forests* yang tetap menggunakan seluruh data sampel penelitian.

**Tabel 4** Hasil Uji Coba Skenario II dengan Metode Uji *Full Train Full Test*

| Set Data | Akurasi J48 | Akurasi Random Forests | Akurasi Metode yang diusulkan |
|---|---|---|---|
| CM1 | 91.57% | 99.60% | 95,38% |
| JM1 | 87.76% | 98.48% | 88,00% |
| KC1 | 90.99% | 98.58% | 94,59% |
| KC2 | 92.91% | 97.51% | 91,00% |
| PC1 | 96.84% | 99.28% | 97,39% |

**Tabel 5** Hasil Uji Coba Skenario II dengan Metode Uji *Full Train Full Test*

| Set Data | Akurasi J48 | Akurasi Random Forests | Akurasi Metode yang diusulkan |
|---|---|---|---|
| CM1 | 88,05% | 88,86% | 89,76% |
| JM1 | 79,78% | 80,95% | 82,05% |
| KC1 | 84,03% | 85,18% | 85,60% |
| KC2 | 81,16% | 82,82% | 83,91% |
| PC1 | 93,55% | 93,48% | 94,31% |

## 5. KESIMPULAN

Berdasarkan hasil-hasil yang diperoleh dari uji coba, didapatkan simpulan bahwa prediksi cacat perangkat lunak menggunakan metode ansambel pohon keputusan dengan deteksi pencilan berbasis *centroid* menghasilkan akurasi yang lebih baik dibanding metode J48 dan *Random Forests*.

## 6. DAFTAR PUSTAKA

[1] Alan, O., & Catal, C. (2009). An outlier detection algorithm based on object-oriented metrics thresholds. Proceedings of ISCIS 2009 conference, Guzelyurt, 14–16 September 2009, Northern Cyprus.

[2] Alan, O, Catal, C., (2011) “Thresholds based outlier detection approach for mining class outliers: An empirical case study on software measurement datasets”, Expert Systems with Applications, vol38, p3440–3445

[3] Boetticher, G. (2006). “Improving credibility of machine learner models in software

engineering". Advanced machine learner applications in software engineering. Series on software engineering and knowledge engineering. Hershey, PA, USA: Idea Group Publishing.

[4] Boetticher, G., Menzies, T., Ostrand, T., (2007) PROMISE Repository of empirical software engineering data http://promisedata.org/ repository, West Virginia University, Department of Computer Science,

[5] Catal C., (2011), Software fault prediction: A literature review and current trends, Expert Systems with Applications 38 (2011) 4626–4636

[6] Catal C, Diri, B., (2009), Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem, Information Sciences, vol 179, p1040–1058

[7] Chang, Chingpao-Pao, (2008), Software process improvement using multivariate statistical process control and action based defect prediction, Departement of computer science and information engineering, National Chen Kung University, Taiwan

[8] Koru, G., & Liu, H. (2005). Building effective defect prediction models in practice. IEEE Software, 22(6), 23–29.

[9] Lyu. M.R., (2007), "Software reliability engineering: A roadmap". In FOSE '07, IEEE Computer Society, p153–170, Washington, DC, USA,.

[10] Ma, Y., Guo, L., & Cukic, B. (2006). "A statistical framework for the prediction of faultproneness". Advances in machine learning application in software engineering. Idea Group Inc.. pp. 237–265.

[11] Hall, M., Hall, Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten I.H., (2009); The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1.Han.

[12] Wicaksono S.A., (2010), "Pembangunan Model Prediksi Defect Pengembangan Perangkat Lunak Menggunakan Metode Ansambel Decision Tree Dan Cost Sensitive Learning,", Master Thesis, Institut Teknologi Sepuluh Nopember, Surabaya.3:243-252