Memprediksi Waktu Memperbaiki *Bug* dari Laporan *Bug* Menggunakan Klasifikasi *Random Forest*

Nur Fajri Azhar[1)] dan Siti Rochimah[2)]
[1, 2)]Teknik Informatika, Institut Teknologi Sepuluh Nopember
Kampus ITS Sukolilo, Surabaya, 60111
Telp : (031) 5992526, 5994251
e-mail: faxjri@gmail.com [1)], siti@its-sby.edu [2)]

Abstrak

Pengembang perangkat lunak harus memiliki rencana dalam pengaturan biaya pengembangan perangkat lunak. Perbaikan perangkat lunak dalam fase pemeliharaan sistem dapat disebabkan oleh bug. Bug adalah kerusakan yang terjadi pada perangkat lunak yang tidak sesuai dengan kebutuhan perangkat lunak. Bug perangkat lunak dapat memiliki waktu yang cepat atau lama dalam perbaikan yang bergantung dari tingkat kesulitannya. Pengembang dapat dibantu oleh rekomendasi model prediksi dan memberikan bahan pertimbangan waktu perbaikan bug.

Dalam penelitian ini, penulis akan menggunakan praproses penyaringan dataset, algoritma random forest untuk pembangunan pendekatan prediksi dan 10-fold cross validation untuk menghitung akurasi. Random forest digunakan karena memiliki kelebihan dalam hal akurasi jika digunakan dengan dataset berjumlah besar. Metode dalam penelitian ini memperoleh akurasi dengan rentang antara 70%-79%. Metode dalam penelitian ini memiliki akurasi yang lebih baik dibandingkan dengan metode decision tree, random forest, dan naïve bayes.

Kata kunci: *Prediksi, Waktu perbaikan bug, Random Forest.*

Abstract

Software developers must have a plan in the regulation of software development costs. Improvements in the maintenance phase of the software system can be caused by a bug. Bug is the damage caused to the software that is incompatible with the needs of the software. Software bugs can have a fast time or a long time in the corrective depends on the level of difficulty. Developers may be assisted by predictive models and provide recommendations for consideration time bug fixes.

In this study, the author will use preprocessing filtering datasets, random forest algorithms for the development of predictive approach and 10-fold cross validation to calculate the accuracy. Random Forests used because it has advantages in terms of accuracy when used with large amounts dataset. The method in this research obtain accuracy with a range between 70% -79%. The method in this study has a better accuracy than the decision tree method, random forest, and naïve Bayes.

Keywords*: Prediction, Bug Fix-Time, Random Forest.*

1. Pendahuluan

Pengembang perangkat lunak harus memiliki rencana dalam pengaturan biaya pengembangan perangkat lunak. Pengembang dapat menggolongkan sebuah perangkat lunak yang berhasil atau tidak setelah dikembangkan berdasarkan dengan rencana yang telah dibuat. Pengembang perangkat lunak akan menerima berbagai laporan tentang kegagalan sistem, permintaan fitur, dan perbaikan sistem dalam fase pemeliharaan sistem. Laporan kerusakan dalam pemeliharaan perangkat lunak salah satunya adalah *bug*. *Bug* adalah kerusakan yang terjadi pada perangkat lunak yang tidak sesuai dengan kebutuhan perangkat lunak. *Bug* perangkat lunak dapat diperbaiki oleh pengembang pemeliharaan perangkat lunak dengan biaya yang bergantung pada waktu perbaikannya. *Bug* perangkat lunak dapat memiliki waktu yang singkat atau lama dalam perbaikan yang tergantung dari tingkat kesulitannya. *Bug* juga dapat diketahui waktu perbaikannya dengan menggunakan model prediksi waktu perbaikan *bug*. Pengembang dapat dibantu oleh rekomendasi model prediksi dan memberikan bahan pertimbangan waktu perbaikan *bug*.

Penelitian pada bidang model prediksi ini telah dilakukan oleh beberapa peneliti untuk menghitung atau memprediksi waktu pengerjaan *bug*. Giger memprediksi waktu perbaikan *bug* menggunakan pohon keputusan (*decision tree*) menjadi kategori cepat atau lambat, hasil penelitian yang dikerjakan memiliki akurasi 60%-70%. Giger mengusulkan untuk mengksplorasi model prediksi untuk mendapatkan tingkat akurasi yang lebih baik. Giger mengusulkan beberapa model prediksi, yaitu menggunakan *Naïve Bayes* dan hutan acak (*Random Forest*) [1]. Abdelmoez telah meneliti *Naïve Bayes* dan penelitiannya menghasilkan prediksi dengan akurasi mencapai 71% untuk kategori *bug* cepat [2]. Vijayakumar dan Bhuvaneswari melakukan penelitian yang membuat kategori berapa banyak usaha yang dibutuhkan untuk mengerjakan sebuah *bug*. Vijayakumar dan Bhuvaneswari menghasilkan eksperimen beberapa model prediksi yang dilakukan dan akurasi yang di dapatkan masih membutuhkan peningkatan kualitas [3]. Alenezi melakukan penelitian untuk pemilihan prioritas pada laporan bug, laporan *bug* diteliti dengan melakukan pelatihan dataset. Penelitian oleh Alenezi mengeluarkan hasil hutan acak memiliki tingkat akurasi yang lebih tinggi dari pada pohon keputusan dan *naïve bayes* [4]. Hutan Acak telah diterapkan oleh Marks yang memiliki sedikit perbedaan. Perbedaan pada penelitian Marks terletak pada pengelompokan interval waktu. Penelitian oleh Marks menghasilkan akurasi mencapai 65% [5].

Pada laporan *bug* perangkat lunak sering ditemui beberapa kesalahan, seperti kesalahan pada isi laporan yang tidak sesuai, atau duplikat laporan. Lamkanfi mengusulkan dalam hasil penelitiannya, laporan *bug* dianalisis sebelum digunakan sebagai dataset model prediksi untuk meningkatkan akurasi [6]. Laporan *bug* dianalisis dengan melakukan penyaringan data yang dianggap bias, laporan *bug* yang termasuk dalam kategori ini yaitu laporan yang memakan waktu pengerjaan hanya beberapa menit atau lebih 100 hari.

Dalam penelitian ini, penulis akan menggunakan metode hutan acak seperti yang diusulkan oleh Giger untuk mengeksplorasi model prediksi waktu perbaikan *bug.* Hutan acak digunakan karena memiliki kelebihan dalam hal akurasi jika digunakan dengan dataset uji berjumlah besar. Penelitian ini diharapkan dapat menghasilkan peningkatan akurasi prediksi dengan melakukan penyaringan laporan *bug*, sehingga penelitian ini dapat digunakan sebagai salah satu pertimbangan model prediksi.

2. Tinjauan Pustaka

Bug adalah salah satu masalah pada perangkat lunak yang sering terjadi. *Bug* merupakan sebuah kesalahan, cacat, kegagalan atau kerusakan dalam sebuah perangkat lunak dan sistem akan berjalan tidak sesuai dengan harapan. *Bug* perangkat lunak dapat terjadi dikarenakan kesalahan pengembang yang tidak di sengaja. Pengembang dapat melakukan kesalahan baik dalam sumber kode (*source code)* atau desain perangkat lunak itu sendiri. *Bug* perangkat lunak muncul dapat dimungkinan karena kerangka (*framework)* dan sistem operasi yang digunakan. Beberapa kasus *bug* dapat disebabkan oleh penyusun (*compiler)* menghasilkan kode yang salah. Perangkat lunak yang memiliki *bug* dalam jumlah besar dapat terganggu kinerja fungsinya, dan perangkat lunak akan digolongkan sebagai *buggy* atau cacat.

Hutan acak merupakan teknik mesin pembelajaran (*machine learning)* yang akan digunakan dalam penelitian ini. Hutan acak di perkenalkan oleh Leo Breiman dengan penelitiannya yang terbit pada tahun 2001 [7]. Hutan Acak adalah sebuah metode pembelajaran *ensemble* yang menghasilkan beberapa pohon keputusan setiap pada saat data dilatih. Hutan acak bekerja dengan menumbuhkan banyak pohon klasifikasi, sehingga pohon-pohon ini dapat diibaratkan sebagai hutan, lalu untuk objek baru diklasifikasikan dari masukan *vector*. Metode ini akan menempatkan masukan *vector* ke masing-masing pohon yang ada di dalam hutan. Setiap pohon akan memberikan klasifikasi dan penilaian untuk kelas yang dilatih. Hutan akan memilih klasifikasi yang memiliki penilaian paling banyak dari keseluruhan pohon yang ada.

2.1. Laporan *Bug*

Laporan *bug* akan digunakan sebagai dataset pada penelitian ini. Laporan *bug* yang akan digunakan dalam penelitian ini akan diambil dari Bugzilla[1]. Laporan *bug* pada Bugzilla terdiri dari deskripsi, lampiran, dan dependensi. Beberapa atribut yang terdapat dalam laporan adalah tanggal pembuatan laporan, identitas pengirim laporan, produk, komponen, prioritas, dan tingkat kerusakan, dan ada beberapa atribut yang dapat berganti status seperti penerima, kondisi saat ini, dan resolusi akhir.

Penelitian ini akan menggunakan data-data laporan dari pengguna untuk dijadikan dataset, dan laporan akan dibandingan dari dua perangkat lunak sumber terbuka (*open source*). Perangkat lunak sumber terbuka yang akan digunakan laporan *bug* sebagai dataset adalah Mozilla Firefox dan Eclipse. Data laporan yang akan di gunakan pada penelitian ini berasal dari tahun 2015 hingga 2016. Dataset yang akan digunakan pada penelitian ini memiliki beberapa atribut yang dapat dilihat pada tabel 1.

[1]Bugzilla http://www.bugzilla.org/

Table 1. Atribut yang di ekstrak dari laporan bug untuk di gunakan dalam dataset

Atribut	Deskripsi
Perangkat	Perangkat hardware, e.g., PC, Mac
Sistem_Operasi	Sistem operasi yang di gunakan
Komponen	Komponen yang terkena *bug*
Ditugaskan	Pengembang yang ditugaskan
Kerasnya	Kondisi dari *bug* e.g., *trivial,critical*
Produk	Komponen asal perangkat lunak yang digunakan
Prioritas	Prioritas dari *bug*
Resolusi	Resolusi terkahir pada laporan *bug*
Waktu_Perbaikan	Waktu yang dibutuhkan untuk memperbaiki *bug*

Pada laporan *bug* yang dilaporkan oleh pengguna terdapat laporan *bug* yang terlihat mencolok. Laporan *bug* terlihat mencolok dengan mencatatkan waktu perbaikan yang sangat singkat atau sangat lama. Laporan *bug* yang mencolok dapat dikatakan kurang "normal" dalam siklus hidupnya. Masalah laporan *bug* mencolok telah di observasi oleh Lamkafi. Lamkafi mendapatkan banyak *bug* yang dilaporkan dengan informasi yang salah, sehingga developer memilih untuk menutup *bug* tersebut. Pengerjaan laporan ini hanya akan memakan waktu satu hari bahkan hanya dalam beberapa menit. Laporan *bug* mencolok ini membuat banyaknya catatan waktu pengerjaan perbaikan menjadi sangat cepat dan dapat menurunkan performa metode klasifikasi[6]. Menerapkan penyaringan dataset bertujuan untuk mengurangi tingkat kesalahan pada pendekatan baru prediksi dan dapat meningkatkan tingkat akurasi waktu perbaikan *bug*.

2.2. Penelitian Terkait

Pada penelitian yang dikerjakan oleh Abdelmoez, klasifikasi pada model prediksi menggunakan naïve bayes. Pertama mengkelompokan kelas dataset menjadi 6 bagian dengan pembagian kuartil Q1,Q2, dan Q3. Kategori waktu perbaikan bug yaitu cepat, lambat, sangat cepat, tidak sangat cepat, tidak sangat lambat, dan sangat lambat. Kelas cepat dan lambat dibagi dengan menggunakan median waktu dari perbaikan bug. Pembagian kelas ini dapat ditentukan dengan persamaan 1 dan 2. Hasil precision prediksi yang didapatkan bervariasi antara 57% sampai 64% [2].

$$\text{Cepat} = \text{waktu perbaikan} < \text{median (Q2)} \ldots (1)$$
$$\text{Lambat} = \text{waktu perbaikan} > \text{median (Q3)} \ldots (2)$$

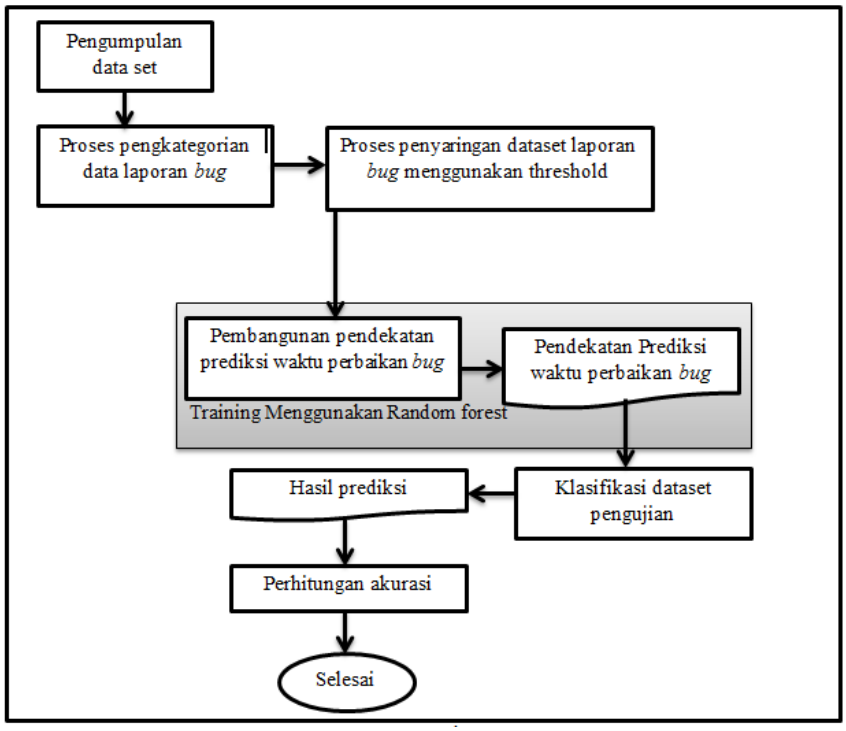
Penelitian yang dilakukan oleh Lamkafi [6] menunjukkan penggunaan metode naïve bayes yang sama seperti AbdelMoez [2]. Lamkafi meneliti cara meningkatkan akurasi prediksi pada prediksi waktu perbaikan *bug* dan mengklasifikasi dengan menggunakan naïve bayes. Perbedaan sedikit dari penelitian lamkafi adalah pembagian kategori kelas yang hanya menggunakan kategori cepat dan lambat saja. Hasil yang didapat pada penelitian Lamkafi menunjukkan peningkatan kualitas prediksi yang dicapai dengan nilai *Area Under the Receiver Operating Characteristic Curve* (AUC) berkisar antara [0.641, 0.722] untuk analisis sebelum melakukan penyaringan data laporan, dan nilai AUC setelah penyaringan data menjadi rentang antara [0.623, 0.733] [6]. Abdelmoez melanjutkan penelitiannya [8] dengan menerapkan cara peningkatan akurasi berdasarkan penelitian Lamkafi [6] dan mendapatkan akurasi mencapai rata-rata 71%. Penggolongan kategori laporan *bug* dan metode klasifikasi masih sama seperti pada penelitiannya sebelumnya.

Giger mengkategorikan laporan *bug* menjadi dua kelompok, yaitu cepat dan lambat. Kategori tersebut dipisahkan dengan median waktu perbaikan dari dataset yang digunakan. Pohon Keputusan digunakan untuk dua kasus, pertama hanya menggunakan data awal (*reporter, date, nextRelease, hToLatFix*), dan menggunakan kedua data awal dan data *post-submission* (*assignee, platform*, sistem operasi, prioritas, *severity*, status, komentar, *milestone* dan lainnya). Penelitian oleh Giger menghasilkan prediksi sebesar 60%-70% dengan tanpa menggunakan data post-submission. Akurasi prediksi dengan menggunakan pohon keputusan lebih baik hingga 10%-20% dari pada menggunakan klasifikasi acak [1].

Marks menerapkan hutan acak sebagai metode klasifikasi. Laporan *bug* pada penelitian ini dikategorikan menjadi 3 kategori. Kategori *bug* dibagi berdasarkan interval waktu sebagai berikut : (0,3 bulan), (3 bulan, 1 tahun), (1,3 tahun), (3 tahun, + ∞ tahun). Penelitian oleh Marks menganalisis dan mengidentifikasi atribut yang paling penting untuk digunakan dan mendapatkan akurasi sekitar 65% [5].

3. Metode Penelitian

Metode yang digunakan dalam penelitian ini yaitu menggunakan praproses pengolahan dataset dan algoritma Random Forest. Berikut alur metodenya:



Gambar 1. Alur metode

Pada gambar 1 dapat dilihat proses awal penelitian ini adalah melakukakan praproses dengan melakukan pengolahan dataset, yaitu menentukan kategori waktu perbaikan pada laporan *bug* dan melakukan penyaringan dataset. Langkah selanjutnya yaitu melakukan proses pembangunan pendekatan prediksi hingga perhitungan akurasi.

3.1. Pengumpulan Dataset

Dataset yang di gunakan dalam penelitian ini adalah laporan dari pengguna yang berasal dari Bugzilla. Data yang akan di gunakan berasal dari tahun 2015 dan 2016. Data laporan *bug* berasal dari perangkat lunak sumber terbuka Eclipse dan Firefox.

Tabel 2. Dataset yang akan digunakan

No.	Dataset	Jumlah Data
1.	Mozilla Firefox 2015	634
2.	Mozilla Firefox 2016	1.937
3.	Eclipse 2015 (Platform, JDT, PDE,E4)	2.121
4.	Eclipse 2016 (Platform, JDT, PDE,E4)	2.200

Penelitian ini akan menggunakan data *bug* seperti pada tabel 2. Dataset yang berasal dari Mozilla Firefox akan menggunakan data sejumlah 2.571 dan Eclipse sejumlah 4.321 yang berasal dari tahun 2015 dan 2016. Dataset yang akan digunakan pada penelitian ini memiliki atribut seperti pada tabel 1.

3.2. Pengkategorian dataset waktu perbaikan

Dataset yang di gunakan dalam penelitian ini diambil dari perangkat lunak Bugzilla sebagai sistem pelacakan *bug* mereka. Status akan di berikan kepada semua *bug* selama daur hidup *bug* terjadi. Sebuah *bug* yang dilaporkan akan diawali dengan status belum dikonfirmasi (*unconfirmed)*. Status belum dikonfirmasi akan berubah menjadi ditugaskan (*assigned*) ketika pengembang perbaikan ditugaskan

untuk memperbaiki. Ketika *bug* telah selesai di kerjakan statusnya akan berubah menjadi terselesaikan (*resolved)*. Pada penelitian ini, waktu perbaikan sebuah *bug* akan diperoleh dari waktu antara *bug* di laporkan dan waktu *bug* tersebut terselesaikan. Pada persamaan (3) akan memperoleh waktu perbaikan *bug* dengan mengurangi waktu selesai dengan waktu *bug* ditugaskan. Waktu selesai dan waktu *bug* ditugaskan terdapat pada setiap laporan *bug.*

$$\text{WaktuPerbaikan}(bug_i)=\text{waktu\_selesai}(bug_i)-\text{waktu\_ditugaskan}(bug_i) \ldots (3)$$

$$\text{Kategori 1}=\text{Waktu perbaikan} \leq 2.160 \text{ jam} \ldots (4)$$

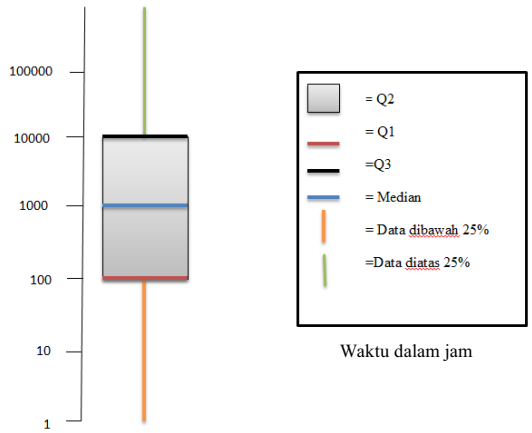
$$\text{Kategori 2}=\text{Waktu perbaikan} > 2.160 \text{ jam} \ldots (5)$$

Penelitian ini akan membuat dua kategori pada waktu perbaikan *bug,* yaitu kategori 1 dan kategori 2. Kategori 1 dan 2 bertujuan untuk mengelompokkan secara keseluruhan dari semua dataset dalam satuan jam yang telah ditentukan. Kategori 1 seperti pada persamaan (4) akan ditentukan waktu perbaikannya kurang dari 2.160 jam atau 0-3 bulan. Kategori 2 seperti pada persamaan (5) akan ditentukan waktu perbaikannya lebih dari 2.160 jam atau 3- ∞ bulan. Kategori *bug* ini akan digunakan sebagai c*lass* pada dataset. Waktu 2.160 jam ditentukan berdasarkan observasi dari penulis terhadap keempat dataset yang digunakan. Dataset yang digunakan pada penelitian ini memiliki rata-rata median data yang berada pada waktu 2-3 bulan.

3.3. Penyaringan Data Laporan *Bug*

Pada laporan *bug* yang dikirim oleh pengguna dapat terjadi beberapa kesalahan informasi. Kesalahan informasi yang dikirim oleh pengguna bisa seperti komponen *bug* (contoh : Komponen umum yang bukan milik komponen UI). Laporan *bug* dapat terlihat perbedaan mencolok antara *bug* yang mencatatkan waktu perbaikan sangat cepat atau sangat lama. Perbedaan mencolok dapat dikatakan kurang "normal" dalam siklus hidup sebuah laporan *bug*. Banyak *bug* yang dilaporkan dengan informasi yang salah sehingga membuat pengembang untuk memilih laporan ditutup. Penutupan sebuah *bug* membuat banyaknya catatan waktu pengerjaan pada laporan *bug* menjadi sangat cepat dan hanya akan memakan waktu satu hari bahkan hanya dalam beberapa menit.

Pada tahap ini akan dilakukan penyaringan pada data laporan *bug* yang akan dijadikan dataset. Penyaringan dilakukan dengan membuat *threshold* yang telah di usulkan oleh Lamkanfi [6]. Penyaringan akan dilakukan langkah pertama dengan membuat median waktu pengerjaan dan menginvestigasi waktu tercepat dan waktu paling lama dalam laporan *bug*. Langkah kedua akan menentukan rentang waktu laporan *bug* paling cepat dan paling lama yang dianggap bias dengan *treshold*. *Bug* yang termasuk dalam rentang waktu akan disaring atau dihapus dari dataset.



Gambar 2 contoh visualisasi kuartil pada data laporan *bug*

Pembentukan *treshold* akan dilakukan dengan membentuk bagian yang seimbang yaitu kuartil seperti yang dapat dilihat pada gambar 2. Kuartil rendah (Q1) akan mengacu pada 25% data yang berada pada bagian bawah. Kuartil atas (Q3) akan mengacu pada 25% data yang berada pada bagian atas. Kuartil dalam (Q2) akan mengacu pada 50% dari data diantara Q1 dan Q3 [9]. Pada penelitian ini akan menyaring data laporan pada data dibawah dengan persamaan (6) [6].

X=½*Q1 … (6)

$$Letak\ Q_i = \frac{i\,(n+1)}{4} \quad \ldots(7)$$

Mendapatkan nilai kuartil 1,2, dan 3 akan dihitung menggunakan persamaan (7), dimana i adalah kuartil ke i, dan n adalah jumlah data.

3.4. Pembangunan Pendekatan Prediksi

Pembangunan pendekatan prediksi akan menggunakan algoritma hutan acak. Dalam penelitian ini akan memprediksi kategori waktu apa yang akan dilabel dari sebuah laporan *bug*. Latihan akan di lakukan lebih dahulu dari dataset yang telah diolah sebelumnya. Penelitian ini akan melakukan percobaan dengan menggunakan alat (*tool*) WEKA. Alat ini dapat mengimplementasikan banyak algoritma termasuk hutan acak.

3.4. Metode Evaluasi

Uji coba selanjutnya akan mengukur kinerja atau performa model prediksi yang akan dilakukan perhitungan akurasi dengan menggunakan 10-*fold cross validation,* Presisi (*Precision*), Sensitivitas (*Recall*) dan Akurasi (*Accuracy*) yang dapat dilihat pada persamaan (8), (9), (10), dan (11). *K-fold cross validation* adalah sebuah teknik intensif komputer yang menggunakan keseluruhan data sebagai *training set* dan *test set*. Seluruh data secara acak dibagi menjadi *K* buah *subset* Bk dengan ukuran yang sama dimana Bk merupakan himpunan bagian dari {1,…,n} dan dilanjutkan pada persamaan (8). setelah itu akan dilakukan iterasi sebanyak *K* kali. Pada penelitian ini akan menggunakan *K* =10. Pada iterasi ke k, subset Bk menjadi *test set*, sedangakan *subset* yang lain menjadi *training set*.

$$\cup_{k=1}^{K} B_k = \{1, \ldots, n\}\ dan\ B_j \cap B_k = \emptyset\ (j \neq k) \quad \ldots(8)$$

$$P = \frac{True\ Positif}{(True\ Positif + False\ Positif)} \ldots(9)$$

$$R = \frac{True\ Positif}{(True\ Positif + False\ Negatif)} \ldots(10)$$

$$A = \frac{(TP+TN)}{(TP+FP+FN+TN)} \ldots(11)$$

Nilai *True Positive* (TP) merupakan jumlah data laporan *bug* yang berhasil diklasifikasikan oleh metode klasifikasi sebagai *bug* yang sesuai dengan *class* nya. Nilai *True Negative* (TN) merupakan jumlah data laporan *bug* yang berhasil diklasifikasikan oleh metode klasifikasi sebagai *bug* yang tidak sesuai dengan *class* nya. Nilai *False Negative* (FN) merupakan jumlah data laporan *bug* yang diklasifikasikan oleh metode klasifikasi sebagai *bug* yang sesuai dengan *class* nya. Nilai *False Positive* (FP) merupakan jumlah data laporan *bug* yang diklasifikasikan oleh metode klasifikasi sebagai *bug* yang tidak sesuai dengan *class* nya. Melakukan uji coba mengenai akurasi dengan (11), beberapa langkah akan dilakukan untuk memperlihatkan perbandingan nilai presisi, sensitivitas dan akurasi antara dataset dari setiap metode.

4. Hasil dan Pembahasan

Pengujian akan dilakukan menjadi dua tahap, yaitu praproses yang melibatkan pengolahan dataset dan proses klasifikasi menggunakan weka dengan metode hutan acak. Evaluasi akan dilakukan dengan membandingkan hasil dari metode yang diusulkan dengan metode yang pernah dilakukan dengan metode pohon keputusan, hutan acak, dan naïve bayes + penyaringan *outlier*.

4.1. Praproses

Pada tahap ini akan dilakukan praproses seperti yang telah dijelaskan pada bab 2 yaitu pengkategorian waktu perbaikan pada laporan *bug* dan penyaringan dataset. Proses yang akan dilakukan adalah pengelompokkan kategori dan menjadikan kategori sebagai *class* pada setiap data laporan *bug* didalam dataset dan menyaring laporan dengan menentukan kuartil sebagai batas. Pengkategorian akan dilakukan dengan menggunakan persamaan (3),(4),(5) pada setiap laporan *bug*. Proses pertama adalah menentukan waktu perbaikan dengan menggunakan persamaan (3). Proses kedua adalah menentukan kelompok laporan *bug* berdasarkan waktu perbaikan dengan menggunakan persamaan (4) atau (5).

Tabel 3. Hasil pengkategorian *class* dataset dari laporan *bug*

No.	Dataset	Kategori 1(data)	Kategori 2(data)	Total Data
1.	Mozilla Firefox 2015	284	350	634
2.	Mozilla Firefox 2016	1086	851	1937
3.	Eclipse 2015 (Platform, JDT, PDE,E4)	1268	853	2121
4.	Eclipse 2016 (Platform, JDT, PDE,E4)	1223	977	2200

Pada tabel 3 menunjukkan dataset firefox tahun 2015 memiliki 284 data pada kategori 1 dan 350 data pada kategori 2. Dataset firefox tahun 2016 memiliki 1086 data pada kategori 1 dan 851 data pada kategori 2. Pada dataset Eclipse memiliki data kategori 1 hampir sama, yaitu 1268 pada tahun 2015 dan 1223 pada tahun 2016, kategori 2 memiliki data 853 pada tahun 2015 dan 977 pada tahun 2016. Dataset firefox tahun 2016 dan kedua dataset eclipse menunjukkan bahwa kategori 1 memiliki data yang lebih banyak dibandingkan dengan kategori 2. Kategori 1 memiliki data lebih banyak karena terdapat data bias yang cukup banyak.

Pada tahapan penyaringan dataset akan dilakukan seperti yang telah dijelaskan pada bab 2. Langkah awal dalam penyaringan adalah dengan mencari letak posisi data (X) dengan menghitung kuartil 1 pada persamaan (7). Penyaringan data akan dilakukan dengan menerapkan persamaan (6). Hasil perhitungan kuartil dan penyaringan dapat dilihat pada tabel 4.

Table 4. Data letak kuartil dan jumlah data hasil penyaringan dataset

No.	Dataset	Q1	Penyaringan (data)	Total Data
1.	Mozilla Firefox 2015	X_{159}	79	555
2.	Mozilla Firefox 2016	X_{489}	241	1696
3.	Eclipse 2015 (Platform, JDT, PDE,E4)	X_{265}	265	1856
4.	Eclipse 2016 (Platform, JDT, PDE,E4)	X_{550}	275	1925

Tabel 4 menunjukkan letak data kuartil (X) dan jumlah data yang harus dikurangi pada bagian kuartil 1 (Q1) dari posisi data dengan nilai terendah. Pada dataset firefox 2015 akan dikurangi 79 data. Pada dataset firefox 2016 akan dikurangi 241 data. Pada dataset Eclipse 2015 akan dikurangi 265 data. Pada dataset Eclipse 2016 akan dikurangi 275 data. Jumlah data akan disaring pada dataset yang berada dibawah data kuartil 1. Jumlah dataset setelah penyaringan adalah 555 untuk firefox 2015, 1686 untuk firefox 2016, 1856 untuk eclipse 2015, dan 1925 untuk eclipse 2016.

4.2. Hasil Pengujian

Pengujian pada pendekatan prediksi ini menggunaan metode hutan acak. Hasil yang didapatkan akan berasal dari hasil TP (*True postif)*, presisi, sesitivitas (*recall*), dan akurasi. Pengujian dilakukan dengan dataset dari Mozilla firefox dan Eclipse yang sudah disaring pada tahap praproses.

Tabel 5. Hasil pengujian metode yang diusulkan

No.	Dataset	TP	Presisi	Sensitivitas	Akurasi
1.	Mozilla Firefox 2015	0.796	0.794	0.796	79.6396
2.	Mozilla Firefox 2016	0.702	0.702	0.702	70.1651
3.	Eclipse 2015 (Platform, JDT, PDE,E4)	0.794	0.799	0.794	79.3642
4.	Eclipse 2016 (Platform, JDT, PDE,E4)	0.731	0.740	0.731	73.1429

Pada tabel 5 menunjukkan hasil pengujian yang telah dilakukan penulis. Pada dataset firefox 2015 mendapatkan hasil yang cukup baik dengan TP 0.796, presisi 0.794, sensitivitas 0.796, dan akurasi mencapai 79.6396%. Pada dataset firefox 2016 mengalami penurunan performa tetapi tidak terlalu signifikan dengan TP, presisi dan sensitivitas mendapatkan hasil sama 0.702 dengan akurasi 70.1651%. Pada dataset eclipse 2015 mendapatkan performa yang cukup baik dengan Tp 0.794, presisi 0.799, sensitivitas 0.794 dan akurasi mencapai 79.3642%. Pada dataset eclipse pada tahun 2016 mendapatkan hasil TP 0.731, Presisi 0.740, sensitivitas 0.731 dan akurasi 73.1429%. Data yang terlihat pada tabel 5 dapat disimpulkan metode ini mendapatkan hasil yang cukup baik pada dataset tahun 2015, tetapi pada dataset tahun 2016 mengalami sedikit penurunan performa.

4.2. Evaluasi

Pada subbab ini dibahas tentang hasil pengujian dan evaluasi dari setiap pengujian yang telah dilakukan. Evaluasi hasil pengujian akan menggunakan dataset yang sama, yaitu firefox 2015, firefox 2016, eclipse 2015 dan eclipse 2016. Hasil klasifikasi dari setiap dataset akan dibandingkan dengan hasil klasifikasi menggunakan tiga metode yang pernah dilakukan, yaitu pohon keputusan [1], hutan acak[5], dan *naïve bayes* + penyaringan *outlier* [8]. Perhitungan performa menggunakan nilai Presisi (P), sensitivitas (S), dan akurasi (A).

Tabel 6 Perbandingan performa antar metode

Metode	Mozilla Firefox 2015			Mozilla Firefox 2016			Eclipse 2015			Eclipse 2016		
	P	S	A	P	S	A	P	S	A	P	S	A
Pohon keputusan	0.721	0.719	71.92	0.688	0.674	67.42	0.792	0.79	79.02	0.648	0.59	59.22
Hutan Acak	0.608	0.625	62.46	0.673	0.68	67.99	0.707	0.719	71.95	0.659	0.706	70.63
Naïve bayes	0.828	0.829	82.88	0.707	0.7	70.05	0.733	0.727	72.68	0.727	0.69	69.03
Metode usulan	0.794	0.796	79.63	0.702	0.702	70.16	0.799	0.794	79.36	0.740	0.731	73.14

Pada tabel 6 menunjukkan nilai presisi dan sensitivitas metode pohon keputusan, hutan acak, naïve bayes, dan metode yang diusulkan pada dataset firefox 2015. Nilai presisi tertinggi didapatkan oleh naïve bayes dengan 0.828. Nilai sensitifitas tertinggi didapatkan oleh naïve bayes. Metode yang diusulkan mendapatkan nilai yang lebih rendah pada presisi 0.74 dan sensitifitas 0.731. Metode yang diusulkan mendapat nilai yang lebih rendah karena jumlah data yang digunakan pada firefox 2015 hanya berjumlah 634. Performa metode yang diusulkan diketahui tidak bekerja lebih baik dari pada naïve bayes jika digunakan kepada dataset yang berjumlah sedikit. Nilai akurasi tertinggi didapatkan oleh metode naïve bayes dengan 82.88%. Nilai akurasi metode yang diusulkan mendapat nilai 79.64%. Metode yang diusulkan mendapatkan persentasi akurasi lebih kecil dari pada *naïve bayes* sebesar 3.24%. Akurasi metode yang diusulkan mendapatkan akurasi yang lebih rendah dari pada *naïve bayes* karena mengalami kesalahan klasifikasi data sebanyak 113. Faktor banyaknya jumlah dataset juga mempengaruhi performa metode yang diusulkan.

Pada dataset firefox 2016 menunjukkan nilai presisi tertinggi didapatkan oleh naïve bayes sebesar 0.707. Nilai presisi metode yang diusulkan mendapatkan nilai 0.702. Nilai presisi antara naïve bayes dengan metode yang diusulkan diketahui memiliki performa sama kuat dan hanya berselisih 0.005. Nilai sensitifitas tertinggi didapatkan oleh metode yang diusulkan sebesar 0.702. nilai sensitifitas metode yang diusulkan hanya selisih sebesar 0.002 terhadap naïve bayes. Performa presisi dan sensitivitas metode yang diusulkan terlihat seimbang dan lebih baik daripada metode naïve bayes. Nilai akurasi tertinggi didapatkan oleh metode yang diusulkan dengan nilai 70.17%. Nilai akurasi metode yang diusulkan hanya selisih sebesar 0.12% dengan naïve bayes. Performa metode naïve bayes dan metode yang diusulkan pada firefox 2016 memiliki performa tinggi yang sama dibandingkan dengan kedua metode lainnya. Pada dataset firefox 2016 menggunakan data sebanyak 1937 disimpulkan akan mendapatkan performa yang lebih tinggi jika menggunakan langkah praproses penyaringan dataset.

Pada dataset eclipse 2015 menunjukkan performa pohon keputusan, hutan acak, naïve bayes, dan metode yang diusulkan memiliki performa hampir sama yang mencapai diatas 0.7. Presisi tertinggi didapatkan oleh metode yang diusulkan sebesar 0.799 yang diikuti dengan pohon keputusan sebesar 0.792. Nilai sensitivitas tertinggi didapatkan oleh metode yang diusulkan sebesar 0.794 dan diikuti oleh metode pohon keputusan sebesar 0.79. Performa antara metode yang diusulkan dengan hutan acak terlihat lebih tinggi, hasil performa metode yang diusulkan dengan melakukan praproses penyaringan dataset terlihat menghasilkan performa yang lebih baik dibandingkan dengan hasil yang tidak menggunakan langkah praproses. Metode yang diusulkan mendapatkan nilai akurasi tertinggi dengan nilai 79.36%. Akurasi metode yang diusulkan terlihat hanya selisis sedikit lebih banyak 0.34% dengan pohon keputusan. Pohon keputusan terllihat bekerja sangat baik pada dataset ini. Performa akurasi yang menggunakan praproses pada naïve bayes dan metode yang diusulkan telihat bekerja lebih baik dari pada hutan acak.

Nilai presisi tertinggi pada dataset eclipse 2016 didapatkan oleh metode yang diusulkan sebesar 0.74. Nilai sensitivitas tertinggi didapatkan oleh metode yang diusulkan sebesar 0.731. Performa metode yang diusulkan terlihat lebih tinggi dari pada metode yang lain. Akurasi tertinggi didapatkan oleh metode yang diusulkan dengan 73.14%. Metode hutan acak memiliki performa tertinggi kedua dengan 70.63%. Pada dataset 2016 terlihat hutan acak memiliki akurasi yang bagus dan terlihat meningkat oleh metode yang diusulkan dengan praproses penyaringan.

5. Simpulan

Analisa hasil pengujian telah didapatkan dari proses pengujian serta perbandingannya dengan metode yang telah diajukan pada penelitian sebelumnya. Metode yang diusulkan yaitu hutan acak ditambah penyaringan dataset memiliki hasil akurasi mencapai rata-rata pada setiap dataset yang digunakan sebesar 75.57%. Akurasi ini merupakan nilai tertinggi dari penelitian yang telah dilakukan sebelumnya dengan rentang antara 70%-79%. Nilai *true positive* yang dihasilkan juga memiliki rata-rata yang cukup baik dengan nilai 0.755. Nilai presisi yang didapatkan didapatkan dengan metode yang diusulkan mendapat rata-rata 0.758. dan sensitivitas rata-rata 0.755.

Metode hutan acak ditambah penyaringan dataset memiliki nilai presisi terbaik setelah naïve bayes, tetapi perbedaan hanya tidak terlalu signifikan yang hanya terpaut 2 angka dibelakang koma. Metode yang diusulkan memiliki nilai sensitivitas yang baik dibandingkan dengan metode lainnya. Metode hutan acak ditambah penyaringan dataset memiliki nilai akurasi yang baik dibandingkan dengan metode pohon keputusan, hutan acak, dan naïve bayes ditambah penyaringan dataset pada penggunaan dataset firefox 2016, eclipse 2015 dan eclipse 2016. Pada dataset firefox 2015 hasil akurasi naïve bayes lebih unggul daripada metode yang diusulkan. Naïve bayes memiliki akurasi yang lebih tinggi jika digunakan terhadap dataset yang memiliki data yang sedikit dibandingkan dengan hutan acak, namun perbedaan akurasi tidak terlalu signifikan dan hanya terpaut sekitar 3%.

Prediksi waktu perbaikan *bug* perangkat lunak yang dilakukan pada penelitian ini adalah berdasarkan pada metode hutan acak yang ditambahkan praproses penyaringan dataset yang memiliki data bias. Prediksi waktu perbaikan *bug* kedepan dapat dikembangkan dan diterapkan pada projek perangkat lunak yang komersial. Kedepannya diharapkan tingkat akurasi dapat ditingkatkan lagi dengan cara atau metode yang lebih baik lagi.

Daftar Pustaka

[1] E. Giger, M. Pinzger, and H. C. Gall, "Predicting the Fix Time of Bugs," in *RSSE '10 Proceedings of the 2nd International Workshop on Recommendation Systems for Software Engineering*, 2010, pp. 52–56.

[2] W. Abdelmoez, M. Kholief, and F. M. Elsalmy, "Bug Fix-Time Prediction Model Using Naïve Bayes Classifier," in *2012 22nd International Conference on Computer Theory and Applications (ICCTA)*, 2012, no. October, pp. 13–15.

[3] K. Vijayakumar, "How much effort needed to fix the bug□? A data mining approach for effort estimation and analysing of bug report attributes in Firefox," in *2014 International Conference on Intelligent Computing Applications*, 2014.

[4] M. Alenezi, "Bug Reports Prioritization□: Which Features and Classifier to Use□?," in *2013 12th International Conference on Machine Learning and Applications*, 2013.

[5] L. Marks and A. E. Hassan, "Studying the Fix-Time for Bugs in Large Open Source Projects Categories and Subject Descriptors," in *Promise '11 Proceedings of the 7th International Conference on Predictive Models in Software Engineering*, 2011, p. No. 11.

[6] A. Lamkanfi and S. Demeyer, "Filtering Bug Reports for Fix-Time Analysis," in *2012 16th European Conference on Software Maintenance and Reengineering*, 2012, pp. 379–384.

[7] L. E. O. Breiman, "Random Forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[8] W. Abdelmoez, M. Kholief, and F. M. Elsalmy, "Improving Bug Fix-Time Prediction Model by Filtering out Outliers," in *2013 The International Conference on Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE)*, 2013, pp. 359–364.

[9] J. T. M. R. L. Scheaffer, M. Mulekar, *Probability and Statistics for Engineers*, 5th ed. Duxbury Press, 2010.