TESIS- KI142502

MEMPREDIKSI WAKTU MEMPERBAIKI *BUG* DARI LAPORAN *BUG* MENGGUNAKAN KLASIFIKASI HUTAN ACAK

NUR FAJRI AZHAR
NRP. 5115201004

DOSEN PEMBIMBING
Dr. Ir. Siti Rochimah, M.T.
NIP. 196810021994032001

PROGRAM MAGISTER
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2017

[Halaman ini sengaja dikosongkan]

THESIS- KI142502

PREDICTING BUG FIX-TIME FROM BUG REPORT USING RANDOM FOREST CLASSIFICATION

NUR FAJRI AZHAR
NRP. 5115201004

SUPERVISOR:
Dr. Ir. Siti Rochimah, M.T.
NIP. 196810021994032001

MASTER PROGRAM
THE EXPERTISE OF SOFTWARE ENGINEERING
DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2017

[Halaman ini sengaja dikosongkan]

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Komputer (M.Kom.)
di
Institut Teknologi Sepuluh Nopember Surabaya

oleh:
Nur Fajri Azhar
Nrp. 5115201004

Dengan judul :
MEMPREDIKSI WAKTU MEMPERBAIKI BUG DARI LAPORAN BUG MENGGUNAKAN KLASIFIKASI HUTAN ACAK

Tanggal Ujian : 10-1-2017
Periode Wisuda : 2016 Gasal

Disetujui oleh:

Dr. Ir. Siti Rochimah, M.T
NIP. 196810021994032001
(Pembimbing 1)

Daniel Oranova Siahaan, S.Kom, M.Sc, PD.Eng.
NIP. 197411232006041001
(Penguji 1)

Nurul Fajrin Ariyani, S.Kom, M.Sc
NIP. 198607222015042003
(Penguji 2)

Fajar Baskoro, S.Kom, M.T
NIP. 197404031999031002
(Penguji 3)

an. Direktur Program Pascasarjana
Asisten Direktur

Prof. Dr. Ir. Tri Widjaja, M.Eng.
NIP. 19611021 198603 1 001

Direktur Program Pasca Sarjana,

Prof.Ir.Djauhar Manfaat, M.Sc., Ph.D.
NIP. 196012021987011001

[Halaman ini sengaja dikosongkan]

MEMPREDIKSI WAKTU MEMPERBAIKI *BUG* DARI LAPORAN *BUG* MENGGUNAKAN KLASIFIKASI HUTAN ACAK

Nama Mahasiswa : Nur Fajri Azhar

NRP Mahasiswa : 5115201004

Pembimbing : Dr. Ir. Siti Rochimah, M.T.

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*.

Penelitian pada bidang model prediksi ini telah dilakukan oleh beberapa peneliti untuk memprediksi waktu perbaikan *bug*. Hasil yang didapatkan dari penelitian sebelumnya masih membutuhkan peningkatan akurasi. Beberapa algoritma telah diusulkan oleh para peneliti untuk dieksplorasi.

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 rata-rata dengan 72.55%. Metode dalam penelitian ini memiliki akurasi yang lebih baik dibandingkan dengan metode *decision tree* dan *naïve bayes*.

Kata kunci : Waktu perbaikan *bug*¸Prediksi, Hutan acak.

[Halaman ini sengaja dikosongkan]

PREDICTING BUG FIX-TIME FROM BUG REPORT USING RANDOM FOREST CLASSIFICATION

By	: Nur Fajri Azhar
Student Identity Number	: 5115201004
Supervisor	: Dr. Ir. Siti Rochimah, M.T.

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.

Research in the field of predictive models has been done by some researchers to predict the timing of bug fixes. The results obtained from previous studies still need to improve accuracy. Several algorithms have been proposed by researchers to be explored.

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 obtains accuracy with a average 72.55%. The method in this study has a better accuracy than the decision tree and naïve Bayes.

Key Words : Bug Fix-Time, Prediction, Random Forest.

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Dengan memanjatkan puji syukur kehadirat Allah SWT atas limpahan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan tesis yang berjudul "MEMPREDIKSI WAKTU MEMPERBAIKI BUG DARI LAPORAN BUG MENGGUNAKAN KLASIFIKASI HUTAN ACAK" dapat terselesaikan sesuai dengan tujuan dan waktu yang diharapkan.

Pada kesempatan ini, penulis menyampaikan ucapan terima kasih yang sebesar besar nya kepada :

1. Allah SWT atas limpahan rahmat, karunia serta ilmu Nya sehingga penulis dapat menyelesaikan Tesis ini dengan baik.
2. Kedua orang tua saya Bapak Muslimin dan Sri Margiyati yang selalu memberi bantuan dan dukungan baik secara moril maupun materil kepada penulis agar senantiasa diberi kelancaran dalam menyelesaikan Tesis ini., serta adik saya Zulkhair Asyari dan Fauzi Al-Hasyr yang saya sayangi.
3. Ibu Dr. Ir. Siti Rochimah, M.T. selaku Dosen Pembimbing penulis yang telah memberikan kepercayaan, perhatian, bimbingan, bantuan dan motivasi kepada penulis dalam proses menyelesaikan tesis ini.
4. Bapak Daniel Oranova Siahaan, S.Kom, M.Sc, PD.Eng, Ibu Nurul Fajrin Ariyani, S.Kom, M.Sc, Bapak Fajar Baskoro S.Kom, M.T selaku dosen penguji yang telah memberikan saran, arahan, dan koreksi dalam tesis ini.
5. Bapak dan Ibu dosen Jurusan Teknik Informatika ITS yang telah banyak memberikan ilmu dan bimbingan.
6. Seluruh staf dan karyawan Teknik Informatika ITS yang telah banyak memberikan kelancaran administrasi akademik.
7. Teman-teman seperjuangan Rosetya Septiyawan, Fawwaz Ali, M. Sonhaji, Wawan Gunawan, Andreyan, Dika Rizky Yunianto, Didih.
8. Rekan-rekan angkatan 2015 Pasca Sarjana Teknik Informatika ITS yang telah menemani perjuangan selama 1,5 tahun ini atas dukungan terhadap pengerjaan tesis ini.
9. Serta pihak-pihak lain yang namanya tidak dapat penulis sebutkan satupersatu.

Penulis menyadari bahwa Tesis ini masih jauh dari kesempurnaan dan banyak kekurangan. Untuk itu dengan segala kerendahan hati penulis mengharapkan kritik dan saran yang membangun dari para pembaca.

Surabaya, Januari 2017

Nur Fajri Azhar

DAFTAR ISI

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

BAB I

PENDAHULUAN

Pada bab ini akan dijelaskan mengenai beberapa hal dasar dalam penelitian yang meliputi latar belakang, perumusan masalah, tujuan, manfaat, kontribusi penelitian, dan batasan masalah.

1.1 Latar Belakang

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 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*)

(Giger, Pinzger, & Gall, 2010). Abdelmoez telah meneliti *Naïve Bayes* dan penelitiannya menghasilkan prediksi dengan akurasi mencapai 71% untuk kategori *bug* cepat (Abdelmoez, Kholief, & Elsalmy, 2012). 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 (Vijayakumar, 2014). M. Alenezi melakukan penelitian untuk pemilihan prioritas pada laporan bug, laporan *bug* diteliti dengan melakukan pelatihan dataset. Penelitian oleh M.Alenezi mengeluarkan hasil hutan acak memiliki tingkat akurasi yang lebih tinggi dari pada pohon keputusan dan *naïve bayes* (Alenezi, 2013). Hutan Acak telah diterapkan oleh Marks (2011) yang memiliki sedikit perbedaan. Perbedaan pada penelitian Marks terletak pada pengelompokan interval waktu. Penelitian oleh Marks menghasilkan akurasi mencapai 65% (Marks and Hassan 2011).

Pada laporan *bug* perangkat lunak sering ditemui beberapa kesalahan, seperti kesalahan pada isi laporan yang tidak sesuai, atau duplikasi laporan. Lamkanfi mengusulkan dalam hasil penelitiannya, laporan *bug* dianalisis sebelum digunakan sebagai dataset model prediksi untuk meningkatkan akurasi (Lamkanfi & Demeyer, 2012). 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 dari 100 hari.

Dalam penelitian ini, penulis akan menggunakan metode hutan acak seperti yang diusulkan oleh Giger (2010) 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 presentasi akurasi prediksi yang cukup tinggi dengan melakukan penyaringan laporan *bug*, sehingga penelitian ini dapat digunakan sebagai salah satu pertimbangan model prediksi.

1.2 Perumusan Masalah

Penelitian ini adalah prediksi waktu perbaikan *bug* menggunakan klasifikasi hutan acak dengan optimasi praproses penyaringan dataset untuk peningkatan akurasi waktu perbaikan. Penelitian ini diharapkan dapat mengatasi masalah akurasi yang terjadi jika menggunakan dataset yang berjumlah besar. Berdasarkan hal tersebut maka rumusan masalah penelitian ini dapat dijelaskan menjadi beberapa butir, yaitu sebagai berikut:

1. Bagaimana cara melakukan klasifikasi laporan perbaikan *bug* perangkat lunak menggunakan metode klasifikasi hutan acak?
2. Bagaimana melakukan penyaringan laporan *bug* yang akan dijadikan dataset?
3. Bagaimana cara melakukan evaluasi dari metode yang diusulkan?

1.3 Tujuan

Penelitian ini bertujuan untuk membuat pendekatan baru dalam prediksi waktu pengerjaan sebuah *bug* yang dilaporkan oleh pengguna. Penelitian dilakukan dengan menggunakan metode hutan acak dan menggolongkan *bug* yang dilaporkan menjadi kategori berdasarkan waktu perbaikan yang di peroleh. Pendekatan yang digunakan dalam penelitian ini, diharapkan dapat menghasilkan pendekatan prediksi yang lebih baik dari metode yang digunakan pada penelitian sebelumnya.

1.4 Manfaat

Penelitian ini diharapkan dapat digunakan oleh para pengembang untuk melakukan prediksi waktu pengerjaan *bug* pada perangkat lunak mereka, sehingga pengembang dapat memaksimalkan penggunaan usaha dan biaya dalam hal pemeliharaan perangkat lunak.

1.5 Kontribusi Penelitian

Kontribusi dalam penelitian ini terkait dengan model prediksi waktu pengerjaan *bug* adalah sebagai berikut:

1. Mengusulkan sebuah pendekatan prediksi waktu pengerjaan *bug* yang menerapkan metode hutan acak sebagai metode untuk melakukan perhitungan.
2. Penerapan langkah penyeleksian dataset dengan menyaring laporan *bug* yang akan digunakan.

1.6 Batasan Masalah

Batasan Masalah dalam penelitian ini adalah sebagai berikut:

1. Penelitian ini membuat model prediksi waktu pengerjaan *bug*, Pekerjaan tidak fokus kepada pengerjaan perbaikan *bug* yang dilaporkan.
2. Dataset laporan *bug* yang digunakan dalam penelitian ini menggunakan laporan *bug* yang telah tersedia di Bugzilla dalam perangkat lunak sumber terbuka (*open-source).*

BAB II

KAJIAN PUSTAKA

Pada Bab ini akan dijelaskan mengenai pustaka yang terkait dengan dasar dalam penelitian yang meliputi *bug* perangkat lunak, dataset dan Penelitian terkait.

2.1 *Bug* Perangkat Lunak

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.

Pada perangkat lunak dapat memiliki laporan terperinci tentang *bug*, laporan *bug* terperinci lebih di kenal sebagai laporan bug *(bug report),* laporan cacat (*defect report),* laporan masalah *(trouble report),* permintaan perubahan *(change request).*

2.1.1 Laporan *Bug*

Laporan *bug* akan digunakan sebagai dataset pada penelitian ini. Laporan *bug* yang akan digunakan dalam penelitian ini akan diambil dari Bugzilla. 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. Pada tabel 2.1 menunjukkan contoh laporan *bug* perangkat lunak yang telah dikerjakan.

Tabel 2.1 Contoh laporan bug yang telah dikerjakan

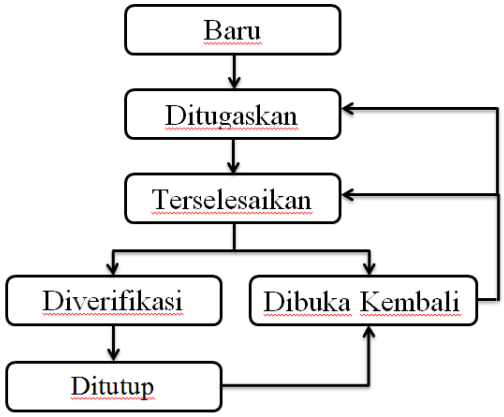
Bug ID	Perangkat	Sistem_Operasi	Ditugaska	Pelapor	produk	kerasnya	prioriti	kompone	resolusi	waktu_dibuka	waktu_selesai
492035	PC	Windows 7	4diac-inb	brian.bro	JDT	normal	P3	Core	DUPLICAT	19-04-16 13:51	19-04-16 14:01
491728	PC	Linux	akurtako	akurtakov	Platform	normal	P3	UI	FIXED	14-04-16 14:29	14-04-16 15:28

2.1.2 Bugzilla

Bugzilla adalah sebuah sistem pelacakan *bug.* Bugzilla adalah alat yang bekerja sebagai pelacak *bug* (*bugtracker).* Bugzilla bersifat perangkat lunak sumber terbuka didirikan Netscape Communication yang dimulai oleh Terry Weissman pada tahun 1998. Bugzilla digunakan untuk mengelola dan memfasilitasi proses laporan *bug*. Bugzilla juga dapat digunakan untuk pengembangan dan mempertahankan sistem perangkat lunak secara efektif. Beberapa perangkat lunak sumber terbuka yang menggunakan sistem ini adalah Mozilla dan eclipse sebagai sistem pelacakan *bug*. Informasi data *bug* pada bugzilla dapat digunakan untuk meneliti beberapa peristiwa dalam pengembangan perangkat lunak, salah satu kegunaannya adalah membuat model prediksi untuk waktu pengerjaan *bug*.

2.1.3 Daur Hidup *Bug*

Laporan *bug* memiliki beberapa kondisi yang berbeda dalam siklus hidupnya. Berdasarkan dari keterangan penggunaan Bugzilla, pada gambar 2.2 dapat dilihat ketika laporan *bug* baru diajukan pada perangkat lunak, *bug* tersebut akan di berikan status sebagai keadaan yang baru. Setelah *bug* yang telah didaftarkan pada *triage* didaftarkan ke pada pengembang perbaikan, *bug* akan berubah statusnya menjadi ditugaskan. Setelah *bug* selesai dikerjakan, status *bug* akan berubah pada laporan menjadi salah satu dari terselesaikan, diverifikasi, atau ditutup. Laporan *bug* akan memiliki resolusi yang ditandai dengan beberapa status. Status resolusi dalam laporan akan digunakan untuk menyimpan cara laporan tersebut diselesaikan.



Gambar 2.2 Daur hidup laporan *bug*

Hasil resolusi *bug* akan ditandai sebagai telah diperbaiki jika terjadi pengubahan kode dasar. Hasil resolusi *bug* akan ditandai sebagai duplikat jika terjadi laporan *bug* ganda. Hasil resolusi *bug* akan ditandai sebagai tidak di perbaiki atau cacat jika bug tidak bisa di perbaiki atau karena bukan sebuah *bug*. Hasil resolusi *bug* akan ditandai sebagai dibuka kembali jika telah selesai diperbaiki tetapi akan dikerjakan lagi. Beberapa *bug* yang belum diperbaiki dengan benar dapat dibuka kembali, sehingga pada penelitian ini jenis laporan *bug* tersebut tidak akan digunakan. Laporan *bug* yang dibuka kembali dapat membuat waktu perbaikan *bug* menjadi bias

2.2 Analisis data

Penelitian ini akan menggunakan dataset berasal dari laporan pengguna pada Bugzilla. Bugzilla memiliki sekumpulan data laporan yang tersimpan dengan baik dan data dapat diakses untuk berbagai kepentingan yang terkait. Penelitian ini akan menggunakan data- data laporan dari pengguna untuk dijadikan dataset, dan laporan akan dibandingan dari dua perangkat lunak sumber terbuka. 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. Berdasarkan hasil yang dilakukan dalam studi literatur, Penelitian ini akan

menggunakan beberapa atribut yang dianggap paling mempengaruhi tingkat akurasi prediksi. Atribut tersebut dapat dilihat pada tabel 2.2.

Tabel 2.2 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 untuk memperbaiki *bug*
Pelapor	Nama pelapor *bug* yang memiliki kerusakan
Produk	Komponen asal perangkat lunak yang digunakan
Kerasnya	Kondisi dari *bug* e.g., *trivial,critical*
Prioritas	Prioritas dari *bug*
Resolusi	Resolusi terkahir pada laporan *bug*
Waktu_Perbaikan	waktu yang dibutuhkan untuk memperbaiki *bug*

Pada penelitian Giger (2010) mengambil kesimpulan bahwa atribut ditugaskan, pelapor, dan bulan *bug* dibuka merupakan atribut yang paling berpengaruh dalam waktu perbaikan *bug*. Penelitian ini akan menggunakan atribut ditugaskan dan pelapor berdasarkan penelitian Giger (2010), sedangkan bulan *bug* dibuka tidak digunakan karena akan digabung bersama atribut waktu sebagai *class* dari dataset. Pada penelitian oleh Marks (2011) menyimpulkan atribut Tahun, Produk, Minggu, dan komponen adalah yang paling berpengaruh pada dataset Firefox, dan kerasnya (Severity), Jumlah dari CCed, dan produk *fix time* paling berpengaruh pada dataset Eclipse. Penelitian ini akan menggunakan atribut Produk, komponen, dan kerasnya berdasarkan penelitian marks (2011). Atribut yang berisi dengan informasi waktu akan digabungkan semua menjadi *class* yaitu Waktu_Perbaikan. Atribut Perangkat, Sistem_Operasi, Prioritas dan Resolusi akan digunakan karena merupakan informasi utama dari laporan *bug*.

2.2.1 Penyaringan Data Laporan *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 Lamkanfi. Lamkanfi mendapatkan banyak *bug* yang dilaporkan dengan informasi yang salah, sehingga pengembang memilih untuk menutup *bug* tersebut (Lamkanfi, 2012). 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.

Berdasarkan masalah laporan *bug* mencolok, maka penelitian ini akan melakukan penyaringan data laporan *bug* yang akan digunakan sebagai data pelatihan. Penulis bertujuan dalam menerapkan hal ini adalah untuk mengurangi tingkat kesalahan pada pendekatan baru prediksi. Tujuan lain yang diharapkan penelitian ini dapat meningkatkan tingkat akurasi waktu perbaikan *bug*.

2.3 Hutan Acak

Hutan acak merupakan teknik mesin pembelajaran (*machine learning)* yang akan digunaan dalam penelitian ini. Hutan acak di perkenalkan oleh Leo Breiman dengan penelitiannya yang terbit pada tahun 2001 (Breiman, 2001). 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.

Setiap pohon akan tumbuh seperti yang dikutip dari Breiman (2001) sebagai berikut :

1. Jika jumlah dalam kasus pada set latihan adalah N, maka sampel kasus N secara acak dengan pergantian dari data asli. Contoh ini akan menjadi set latihan untuk pohon yang tumbuh.
2. Jika ada variabel masukan M, sejumlah m<<M ditentukan sehingga pada setiap *node*, variable m yang di pilih secara acak dari M dan perpecahan terbaik untuk m ini digunakan untuk berbagi *node*. Nilai m tetap konstan selama pertumbuhan hutan.
3. setiap pohon akan tumbuh sebesar dan sejauh mungkin.

Pada penelitian Breiman (2001) dijelaskan tingkat kesalahan yang terjadi pada hutan tergantung pada dua hal yaitu sebagai berikut:

1. Korelasi antara dua pohon yang berada di hutan. Meningkatnya korelasi dapat juga meningkatkan tingkat kesalahan atau *error* pada hutan.
2. Kekuatan individu pada masing-masing pohon dalam hutan. Sebuah pohon dengan tingkat kesalahan yang rendah adalah penggolong (*classifier*) yang kuat. Meningkatnya kekuatan individu pohon dapat mengurangi tingkat kesalahan pada hutan.

2.4 Penelitian Terkait

Beberapa penelitian yang telah dilakukan terkait dengan model prediksi waktu perbaikan *bug* pada laporan *bug* perangkat lunak telah kaji adalah sebagai berikut:

2.4.1 Metode

Pada penelitian yang telah dilakukan memiliki beberapa metode yang berbeda. Metode tersebut terdiri dari *naïve bayes*, dan yang lainnya adalah kNN, pohon keputusan, dan hutan acak. Detail dari metode penelitian yang telah dilakukan akan dijelaskan sebagai berikut:

2.4.1.1 Naïve bayes

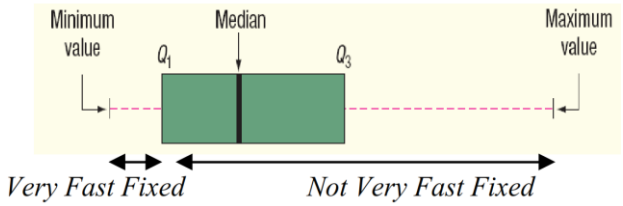
Pada penelitian yang dikerjakan oleh Abdelmoez (2012), klasifikasi pada model prediksi menggunakan *naïve bayes*. Pertama mengkelompokan kelas dataset menjadi 6 bagian dengan pembagian kuartil Q1,Q2, dan Q3. Kategori

waktu perbaikan *bug* yaitu cepat, lambat, sangat cepat, tidak sangat cepat, tidak sangat lambat, dan sangat lambat.

Pada penelitian AbdelMoez (2012), kelas cepat dan lambat dibagi dengan menggunakan median waktu dari perbaikan *bug*. Pembagian kelas ini dapat ditentukan dengan persamaan 2.1 dan 2.2.

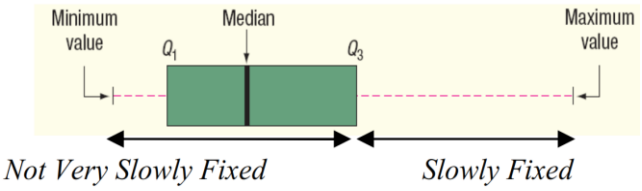
- Cepat = waktu perbaikan < median (Q2) (2.1)
- Lambat = waktu perbaikan > median (Q3) (2.2)

Pengkategorian *bug* dilanjutkan menjadi sangat cepat dan tidak sangat cepat. Pembagian ini menggunakan aturan sangatCepat = waktu perbaikan <Q1 dan tidakSangatCepat=waktu perbaikan > Q1. Visualisasi pembagian kelas sangat cepat dan tidak sangat cepat dapat dilihat pada gambar 2.2.



Gambar 2.1 Visualisasi pengkaegorian kelas *bug* 1 (Abdelmoez, 2012)

Pada gambar 2.3 pengkategorian *bug* menjadi sangat cepat dan tidak sangat cepat. Pembagian ini menggunakan aturan sangatLambat=Waktu perbaikan<Q3 dan tidakSangatLambat = waktu perbaikan > Q3.



Gambar 2.2 Visualisasi pengkategorian kelas *bug* 2 (Abdelmoez, 2012)

Penelitian yang dilakukan oleh Lamkafi (2012) menunjukkan penggunaan metode *naïve bayes* yang sama seperti penelitian oleh AbdelMoez (2012). Lamkafi (2012) 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.

Abdelmoez melanjutkan penelitiannya (2013) dengan menerapkan cara peningkatan akurasi berdasarkan penelitian Lamkafi (2012). Penggolongan kategori laporan *bug* dan metode klasifikasi masih sama seperti pada penelitiannya sebelumnya (Abdelmoez, Kholief, & Elsalmy, 2013).

2.4.1.2 Pohon Keputusan (*Decision Tree*)

Giger (2010) 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).

2.4.1.3 Hutan Acak

Marks (2011) 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) dan (1,3 tahun). Penelitian oleh Marks menganalisis dan mengidentifikasi atribut yang paling penting untuk digunakan (Marks & Hassan, 2011).

2.4.1.4 kNN-*based*

Zhang (2013) mengusulkan menggunakan kNN untuk memprediksi waktu perbaikan *bug*. Pada penelitian ini memprediksi *bug* menjadi kategori lambat (diatas treshold waktu) dan cepat (dibawah threshold waktu). Zhang et. al. (2013) juga mendefinisikan metrik jarak baru untuk mengukur kesamaan antara dua laporan *bug*, dan menerapkan teknik kNN untuk menentukan upaya yang diperlukan untuk *bug* baru berdasarkan asumsi bahwa dua *bug* yang sama membutuhkan upaya perbaikan *bug* yang mirip (Zhang, Gong, & Versteeg, 2013).

2.4.2 Dataset

Dataset yang digunakan pada penelitian banyak yang berasal dari perangkat lunak open source, seperti Mozilla firefox, eclipse, dan gnome. Laporan *bug* perangkat lunak open source dapat diakses melalui Bugzilla. Penelitian oleh Zhang (2013) menggunakan dataset yang berasal dari program komersial atau yang bersifat pribadi. Berdasarkan pada penelitian Giger (2010), data-data ini dapat meningkatkan tingkat akurasi prediksi. Data lengkap tentang dataset yang telah digunakan dapat dilihat pada tabel 2.3.

Tabel 2.3 Dataset yang telah digunakan pada penelitian sebelumnya

Peneliti	Dataset	Atribut
Abdelmoez [1] [2]	Eclipse JDT (Oct 2001-2007) Mozilla (Apr.2001-July 2008) Gnome Gstreamer (Apr.2002-Aug 2008) Gnome evolution (Jan.1999-July 2008)	Component,Reporter, Assignee, Priority, Severity, Platform, OS, Resolution, Status, hToLastFix, nrActivities, nrComments, hOpenedBeforeNextRelease, monthOpened, yearOpened, Milestone, nrPeopleCC
Zhang [3]	real maintenance data from three CA Technologies projects.	Tidak diperlihatkan dalam paper.
Lamkafi [4]	Eclipse Platform (Oct. 2001 - Oct. 2007) Eclipse PDE (Oct. 2001 - Oct. 2007) Eclipse JDT (Oct. 2001 - Oct. 2007) Eclipse CDT (Oct. 2001 - Oct. 2007) Eclipse GEF (Oct. 2001 - Oct. 2007) Mozilla Core (Mar. 1997 - Jul. 2008) Mozilla Bugzilla (Mar. 2003 - Jul. 2008) Mozilla Firefox (Jul. 1999 - Jul. 2008) Mozilla Thunderbird (Jan. 2000 - Jul. 2008) Mozilla Seamonkey (Nov. 1995 - Jul. 2008)	day opened month opened year opened platform op sys assigned to reporter severity priority component fixtime
Marks [5]	Eclipse Mozilla	(1) the location of the bug (Product, Version,Component, Operating system, Open

		bugs, Fixed bugs, Project fix-time) (2) the reporter of the bug (Reporter type, Reporter popularity, Reporter fix-time, Reporter requests) (3) the description of the bug (Severity Priority,Interest, Time, Bugs open in project, Has comments, Has target milestone, Length of bug description, Amount of code, Readability, Severity fix-time, Priority fix-time)
Giger [6]	Eclipse JDT (Oct. 2001 – Oct. 2007) Eclipse Platform (Oct. 2001 – Aug. 2007) Mozilla Core (Mar. 1997 – June 2008) Mozilla Firefox (Apr. 2001 – July 2008) Gnome GStreamer (April 2002 – Aug.2008) Gnome Evolution (Jan. 1999 – July 2008)	monthOpened, yearOpened, platform, os, reporter, assignee, milestone, nrPeople, priority, severity, OpenedBeforeNextRelease, resolution, status, hToLastFix, nrActivities, nrComments.
Pamela [7]	Mozilla Project (1998-2010) - Firefox - Seamonkey - Thunderbird Eclipse (Oct.2001 to March 2010) Chrome Project (Aug. 2008 – Nov 2010)	Tidak diberikan dalam paper.

2.4.3 Akurasi Prediksi

Setiap penelitian menghasilkan tingkat akurasi yang beragam. Akurasi pada penelitian dapat dipengaruhi oleh metode, dataset, atau atribut yang digunakan. Pada tabel 2.4 dapat dilihat hasil akurasi dari semua penelitian yang terkait.

Tabel 2.4 Hasil prediksi akurasi

Peneliti	Akurasi
Abdelmoez [1]	*Precision* 57% - 64%
Abdelmoez [2]	Rata-rata 71%
Zhang [3]	*Fmeasure* 72.45%.
Lamkafi [4]	AUC [0.623, 0.733]
Marks [5]	65%
Giger [6]	60%-70%
Pamela [7]	30% - 49%.

Metode *naïve bayes* yang diterapkan oleh Abdelmoez et al. (2012) mendapatkan presisi prediksi yang bervariasi antara 57% sampai 64%, Namun ketika *output set* dibagi oleh 1 atau 3 kuartil, dan hasil prediksi menjadi berkurang. Penelitian oleh Lamkafi (2012) menghasilkan 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]. Hasil yang didapat pada penelitian Lamkafi menunjukkan peningkatan kualitas prediksi yang dicapai.

Abdelmoez et al. (2013) melanjutkan studinya dengan menyaring data laporan dan bereksperimen dengan *threshold* untuk menghapus *outlier*. Mereka menemukan bahwa penghapusan '*outlier* ringan' yang disebut *Inner fence* (IF), dengan IF = Q3 + 1,5 * (Q3 - Q1), mencapai rata-rata 71% klasifikasi prediksi yang benar. Marks (2011) menggunakan metode hutan acak mendapatkan akurasi sekitar 65%. Pamela (2010) menggunakan *Multivariate Regression Testing* mendapatkan hasil yang menunjukkan kekuatan prediksi berkisar antara 30% sampai 49%.

Zhang (2012) menggunakan metode berbasis kNN untuk mengklasifikasikan waktu yang dibutuhkan (cepat atau cepat) untuk memperbaiki *bug*. Evaluasi hasil menunjukkan penelitian menggunakan kNN mendapatkan nilai rata-rata *Fmeasure* sebesar 72.45% (Zhang, 2012).Pada penelitian oleh Giger (2010) 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 (Giger, 2010).

2.4.4 Evaluasi Akurasi

Penelitian oleh Abdelmoez (2012) mendapatkan akurasi yang bervariasi antara 57% sampai 64%. Hasil prediksi menjadi berkurang, ketika *output set* dibagi oleh 1 atau 3 kuartil, ketepatan klasifikasi sangat menurun untuk kelas target yang lebih kecil. Hal ini dikarenakan dataset yang digunakan memiliki banyak *outlier* atau data yang terlihat kurang normal.

Pada penelitian yang dilakukan oleh Lamkafi (2012) menjelaskan tentang perbedaan pada data yang kurang normal. Contoh laporan *bug* yang kurang normal adalah *bug* yang memiliki catatan waktu pengerjaan sangat cepat dan hanya akan memakan waktu satu hari bahkan hanya dalam beberapa menit dan juga sebaliknya. Hasil yang didapatkan oleh lamkafi terbukti dapat meningkatkan akurasi model prediksi dan diterapkan oleh Abdelmoez (2013). Hasil peningkatan akurasi model prediksi menunjukkan banyak perbaikan kualitas prediksi yang dapat dicapai.

Penelitian oleh Marks (2011) mendapatkan akurasi yang mencapai 65%, dan menemukan hasil bahwa atribut laporan *bug creation date*, dan *bug location* mempunyai dampak yang kuat pada resolusi waktu perbaikan, sementara atribut *priority* tidak memiliki pengaruh yang signifikan.

Pada paper yang dikerjakan oleh Pamela (2010) menunjukkan data projek komersial tidak sama seperti pada data perangkat lunak sumber terbuka yang tidak dipengaruhi oleh reputasi *bug-opener* (Bhattacharya, 2010). Metode yang menggunakan *regression analysis* menunjukkan atribut laporan *bug* yang sebelumnya digunakan untuk membangun model prediksi waktu perbaikan *bug* tidak selalu berkorelasi dengan waktu perbaikan *bug*. Akurasi prediksi yang didapatkan oleh Giger (2010) sekitar 60%-70%. Giger (2010) juga menyatakan bahwa dengan menggunakan semua data atribut pada laporan dapat meningkatkan akurasi sedikit lebih tinggi.

Tabel 2.5 Hasil studi komparasi

Item Perbandingan	**Penelitian**					
	E. Giger (2010)	W. Abdelmoez (2012)	A.Lamkanfi(2012)	w. AbdelMoez (2013)	M. Alenezi (2013)	Marks (2011)
Algoritma Klasifikasi	*Decision tree*	*naïve bayes*	*naïve bayes*	*naïve bayes*	*Decision tree, naïve bayes, Random Forest*	*Random Forest*
Masalah	*Bug* mana yang harus diperbaiki lebih dahulu & Berapa lama waktu yang di butuhkan untuk memperbaikinya	*Bug* mana yang harus diperbaiki lebih dahulu & Berapa lama waktu yang di butuhkan untuk memperbaikinya	Kualitas model prediksi dapat dipengaruhi dengan *factor bug report* yang memiliki *outliers*.	Kualitas model prediksi dapat dipengaruhi dengan *factor bug report* yang memiliki *outliers*.	Memberikan prioritas kepada *bug report*.	Mempelajari atribut yang paling penting dalam laporan *bug*
Dataset	17 atribut *Bug* report dari 6 *software sistem* -Eclips -Mozilla -Gnome	17 atribut *Bug report* dari 4 *software sistem* -Eclips -Mozilla -Gnome	11 atribut 5 *software sistem* di Eclips dan Mozilla	17 atribut *Bug report* dari 4 *software sistem* -Eclips -Mozilla -Gnome	*Feature set-1(textual desc.)* *-tokenization* *-Vector space representation.* *Feature set-2 Metadata feature; Componen, OS, severity*	Mozilla & Eclipse menggunakan 3 atribut dimensi (Lokasi *bug*, *reporter bug*, deskripsi *bug*)
Filtering bug report	Tidak ada	Tidak ada	Ada	Ada	Tidak ada	Tidak Ada
Seleksi fitur	Tidak ada	Tidak ada	Tidak ada	Tidak ada	Ada	Ada
Kelebihan	60-70% tepat pengklasifikasian Dan dapat meningkat 5 sampai 10% dengan data *post-submission.*	Dapat membagi sampai 6 kategori jenis *bug*, dengan akurasi yang cukup baik	Peningkatan hasil dari akurasi cukup besar, dan pengirisan dalam jumlah *bug report* yang tidak dibutuhkan.	Meningkatnya recall dan persentasi peningkatan akurasi prediksi sampai 90%	*Decision tree & Random forest* lebih baik hasilnya dibandingkan *naïve bayes.*	menghasilkan atribut paling penting yang digunakan. akurasi mencapai 65%.

[Halaman ini sengaja dikosongkan]

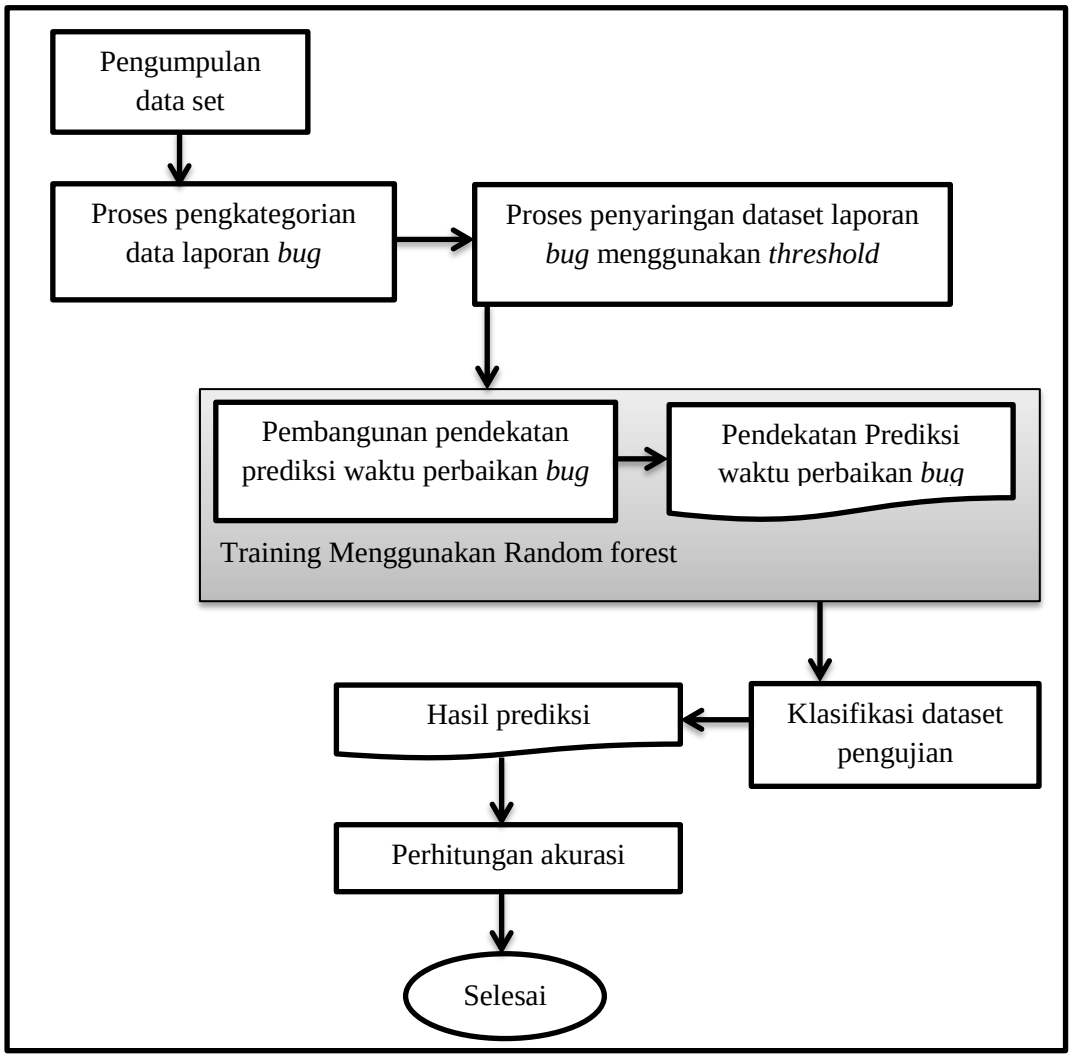
BAB III

METODOLOGI PENELITIAN

Bab ini akan menjelaskan metodologi penelitian yang akan digunakan pada penelitian ini.

3.1 Metode Penelitian

Pada tahap desain akan dibahas tentang perancangan dari solusi yang diusulkan. Alur desain yang akan digunakan dapat dilihat dalam gambar 3.1. Pada alur desain pemodelan terdapat tahap penyaringan dataset dan penggunaan hutan acak sebagai algoritma untuk melatih dataset.



Gambar 3.1 Alur desain Pemodelan

3.1.1 Pengumpulan Dataset

Dataset yang di gunakan dalam penelitian ini adalah laporan dari pengguna yang berasal dari Bugzilla. Data laporan *bug* berasal dari perangkat lunak Eclipse dan Firefox. Dataset Firefox akan diambil berdasarkan dari tahun 2004 sampai 2016 dengan jumlah 79.393 data. Dataset Eclipse akan diambil berdasarkan dari tahun 2010 sampai 2016 dengan jumlah 35.460 data.

Tabel 3.1 Dataset yang akan digunakan

No.	Dataset	Jumlah Data	Periode pengamatan
1.	Firefox 1	16.495	Jan. 2004 - Des. 2007
2.	Firefox 2	27.438	Jan. 2008 - Des. 2016
3.	Eclipse 1	17.499	Jan. 2010 – Des. 2012
4.	Eclipse 2	17.961	Jan. 2013 – Des. 2016

Penelitian ini akan menggunakan dataset dari laporan *bug* seperti pada tabel 3.1. Dataset yang berasal dari Firefox dan Eclipse akan dibagi menjadi dua. Pada dataset Firefox akan dibagi dari periode pengamatan pada tahun 2004-2007 dan 2008 - 2016. Pada dataset Eclipse akan dibagi dari periode pengamatan pada tahun 2010-2012 dan 2013-2016. Pembagian dataset berdasarkan periode pada tabel 3.1 bertujuan untuk memberikan beberapa sekenario yang akan digunakan pada pengujian.

3.1.2 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 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 3.1 akan memperoleh waktu perbaikan *bug* dengan mengurangi waktu perbaikan dengan waktu *bug* dibuka.

Tabel 3.2 Contoh potongan dataset eclipse 2016 yang telah dihitung waktu perbaikan

Bug ID	Perangkat	Sistem_Operasi	Ditugaska	Pelapor	produk	kerasnya	priorit	kompone	resolusi	waktu_dibuka	waktu_selesai	Waktu_Pe
492035	PC	Windows 7	4diac-inb	brian.bro	JDT	normal	P3	Core	DUPLICAT	19-04-16 13:51	19-04-16 14:01	0
445456	PC	Linux-GTK	abuzila	abuzila	Platform	normal	P3	SWT	FIXED	30-09-14 5:03	21-09-16 12:14	17335
478618	PC	Linux	abuzila	snjezana.	Platform	normal	P3	SWT	FIXED	29-09-15 7:49	04-05-16 9:16	5233
37322	PC	Windows 2000	aeschli	akiezun	JDT	normal	P3	UI	WONTFIX	07-05-03 9:40	10-06-16 5:11	114787
43082	All	All	aeschli	preuss	JDT	enhancem	P3	UI	WORKSFC	15-09-03 7:52	04-07-16 14:23	112230
103420	PC	Windows XP	agarcher	amwright	Platform	normal	P3	User Assis	INVALID	11-07-05 18:56	03-08-16 8:17	96973
486016	PC	Linux	akurtako	akurtakov	Platform	normal	P3	User Assis	FIXED	18-01-16 3:23	18-01-16 4:41	1
491728	PC	Linux	akurtako	akurtakov	Platform	normal	P3	UI	FIXED	14-04-16 14:29	14-04-16 15:28	0
496890	PC	Linux	akurtako	akurtakov	Platform	normal	P3	User Assis	FIXED	28-06-16 1:17	28-06-16 5:26	4

$$WaktuPerbaikan(bug_i)=waktu\_selesai(bug_i)-waktu\_dibuka(bug_i) \quad (3.1)$$

Persamaan 3.1 adalah rumus untuk memperoleh waktu perbaikan *bug*. Waktu perbaikan *bug* akan didapatkan dengan mengurangi waktu selesai *bug* diperbaiki dengan waktu *bug* dibuka. Semua *bug* dalam laporan akan memiliki waktu perbaikan yang akan di masukkan kedalam dataset untuk dilakukan latihan.

Pada tabel 3.2 menunjukkan potongan dataset eclipse 2016 yang telah dihitung waktu perbaikannya. Waktu perbaikan pada tabel 3.2 menggunakan persamaan 3.1 dengan satuan jam. Pada laporan *bug* kolom pertama tabel 3.2 dengan ID 492035 memiliki waktu perbaikan 0 jam. Waktu perbaikan 0 jam pada laporan *bug* seperti pada *bug* dengan ID 492035 dapat dikarenakan duplikasi dari laporan *bug* yang sudah pernah dilaporkan lebih dahulu.

3.1.2.1 Pengkategorian dataset waktu perbaikan

Bug dibagi menjadi 2 kategori dalam dalam laporan, yaitu cepat dan lambat. Giger et al. telah mengusulkan cara ini pada penelitiannya (Giger, 2010). Kategori *bug* akan didapatkan dengan menggunakan nilai median waktu perbaikan dari seluruh data laporan yang digunakan. Nilai median akan didapatkan dengan persamaan 3.2. Pada rumus median, X adalah letak data dan n adalah jumlah keseluruhan data. Kedua kategori akan diberikan kepada *bug* dengan cara persamaan 3.3 berikut (Giger, 2010):

$$Median = X_{(n+1)/2} \quad (3.2)$$

$$KategoriBug = \begin{cases} Cepat: WaktuPerbaikan \leq Median \\ Lambat: WaktuPerbaikan > Median \end{cases} \quad (3.3)$$

Berdasarkan pada persamaan 3.3, *Bug* akan di kategorikan sebagai *bug* yang cepat jika waktu perbaikannya sama atau lebih cepat dari pada median waktu perbaikan. *Bug* akan dikategorikan sebagai lambat jika didapatkan waktu perbaikan lebih lama dari median waktu perbaikan. Waktu perbaikan pada laporan *bug* menggunakan hitungan jam.

Penelitian ini akan melakukan percobaan dalam pembagian *class* waktu perbaikan dengan membandingkan metode pembagian dengan kuartil (Abdelmoez, 2012) dan berdasarkan waktu (Marks, 2011). Pada percobaan *class* pertama akan menggunakan metode kuartil yang akan dibagi berdasarkan Q1, Q2, dan Q3. Pada percobaan *class* kedua akan menggunakan 4 kategori waktu dibawah 720 jam (0-1 bulan), 720 jam – 2160 jam (1-3 bulan), 2160 jam – 8640 jam (3 bulan – 1 tahun), dan lebih dari 8640 jam (1 tahun - ∞). Kategori waktu memiliki perbedaan dari penelitian marks (2011) yang hanya menggunakan 3 *class* dan percobaan ini menambahkan kategori waktu (0-1 bulan).

$$Kategori\ 1 = \text{Waktu perbaikan } \leq Q1 \tag{3.4}$$
$$Kategori\ 2 = \text{Waktu perbaikan } > Q1 \ \& \text{ Waktu perbaikan } \leq Q2 \tag{3.5}$$
$$Kategori\ 3 = \text{Waktu perbaikan } > Q2 \ \& \text{ Waktu perbaikan } \leq Q3 \tag{3.6}$$
$$Kategori\ 4 = \text{Waktu perbaikan } > Q3 \tag{3.7}$$

$$Kategori\ 1 = \text{Waktu perbaikan } < 720 \text{ jam} \tag{3.8}$$
$$Kategori\ 2 = \text{Waktu perbaikan } > 720 \ \& \text{ Waktu perbaikan } < 2160 \tag{3.9}$$
$$Kategori\ 3 = \text{Waktu perbaikan } > 2160 \ \& \text{ Waktu perbaikan } < 8640 \tag{3.10}$$
$$Kategori\ 4 = \text{Waktu perbaikan } 8640 \text{ jam} - \ \infty \tag{3.11}$$

Pada percobaan *class* pertama akan menemukan waktu perbaikan *bug* berdasarkan pada persamaan 3.1 dan dilanjutkan pengkategorian menggunakan persamaan 3.4, 3.5, 3.6, dan 3.7. Pencarian kuartil Q1, Q2, dan Q3 akan dibahas pada subbab 3.1.3. Kategori 1 akan menjadi waktu perbaikan paling cepat. Kategori 2 akan menjadi waktu perbaikan cepat. Kategori 3 akan menjadi waktu perbaikan lambat. Kategori 4 akan menjadi waktu perbaikan paling lambat. Pengkategorian *class* menggunakan kuarti akan membuat persebaran *class* menjadi merata karena akan membagi dataset berdasarkan *threshold* dari jumlah laporan pada setiap dataset. Pada percobaan *class* kedua akan menemukan waktu perbaikan *bug* berdasarkan pada persamaan 3.1 dan dilanjutkan dengan pengkategorian menggunakan persamaan 3.8, 3.9, 3.10, 3.11. Penggunaan kategori *class* berdasarkan waktu memiliki tujuan untuk mempermudah

mengenali waktu perbaikan dari berbagai dataset dan mengetahui secara pasti berapa waktu perbaikan.

Langkah pertama dalam tahap pengelompokkan ini yaitu dengan mengurutkan data laporan *bug* dari waktu perbaikan terendah hingga tertinggi seperti pada tabel 3.3. Pada tabel 3.3 adalah pengurutan waktu perbaikan dataset pada tabel 3.2. Pengurutan waktu perbaikan bertujuan untuk mempermudah proses pengkategorian dan langkah praproses selanjutnya yaitu penyaringan dataset.

Langkah kedua adalah merubah waktu perbaikan menjadi kategori 1,2,3 atau 4 seperti contoh pada tabel 3.4. Perubahan nama kategori dirubah berdasarkan pada kedua jenis *class* dengan persamaan 3.4, 3.5, 3.6, 3.7. dan 3.8, 3.9, 3.10, 3.11. Atribut waktu_Perbaikan pada laporan *bug* akan menjadi *class* dalam proses prediksi. Langkah ketiga adalah menghapus atribut *bug* ID,waktu_dibuka, dan waktu_selesai. Langkah ketiga dilakukan karena atribut ini tidak akan digunakan dalam dataset seperti pada tabel 3.4.

Tabel 3.4 Contoh potongan dataset eclipse 2016 yang telah diurutkan waktu perbaikan

Bug ID	Perangkat	Sistem_Operasi	Ditugaska	Pelapor	produk	kerasnya	prioritas	kompone	resolusi	waktu_dibuka	waktu_selesai	Waktu_Pe
492035	PC	Windows 7	4diac-inbo	brian.broo	JDT	normal	P3	Core	DUPLICAT	19-04-16 13:51	19-04-16 14:01	0
491728	PC	Linux	akurtako	akurtakov	Platform	normal	P3	UI	FIXED	14-04-16 14:29	14-04-16 15:28	0
486016	PC	Linux	akurtako	akurtakov	Platform	normal	P3	User Assis	FIXED	18-01-16 3:23	18-01-16 4:41	1
496890	PC	Linux	akurtako	akurtakov	Platform	normal	P3	User Assis	FIXED	28-06-16 1:17	28-06-16 5:26	4
478618	PC	Linux	abuzila	snjezana.p	Platform	normal	P3	SWT	FIXED	29-09-15 7:49	04-05-16 9:16	5233
445456	PC	Linux-GTK	abuzila	abuzila	Platform	normal	P3	SWT	FIXED	30-09-14 5:03	21-09-16 12:14	17335
103420	PC	Windows XP	agarcher	amwright	Platform	normal	P3	User Assis	INVALID	11-07-05 18:56	03-08-16 8:17	96973
43082	All	All	aeschli	preuss	JDT	enhancem	P3	UI	WORKSFO	15-09-03 7:52	04-07-16 14:23	112230
37322	PC	Windows 2000	aeschli	akiezun	JDT	normal	P3	UI	WONTFIX	07-05-03 9:40	10-06-16 5:11	114787

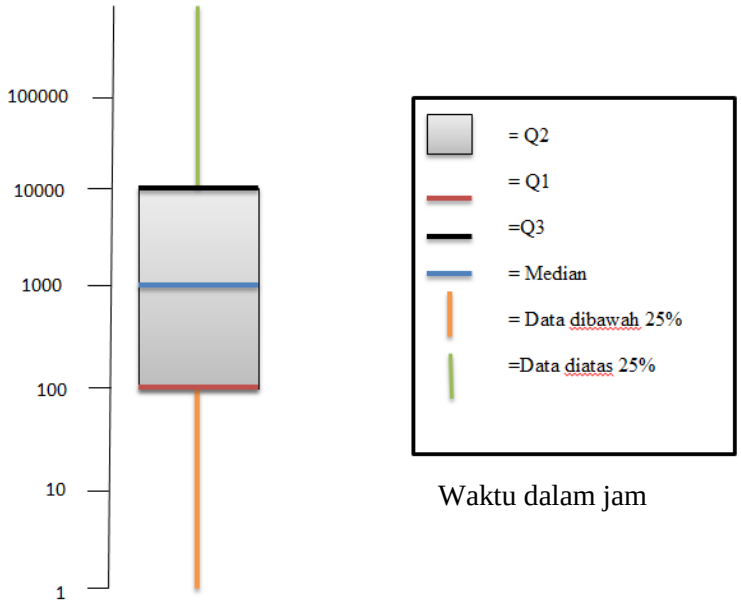
Tabel 3.3 Contoh potongan dataset eclipse 2016 yang telah dikategorikan

Perangkat	Sistem_O	Ditugaskan	Pelapor	produk	kerasnya	prioritas	kompo	resolusi	Waktu_Perbaikan
PC	Windows	4diac-inbox	brian.broo	JDT	normal	P3	Core	DUPLICAT	Kategori 1
PC	Linux	akurtako	akurtakov	Platform	normal	P3	UI	FIXED	Kategori 1
PC	Linux	akurtako	akurtakov	Platform	normal	P3	User A	FIXED	Kategori 1
PC	Linux	akurtako	akurtakov	Platform	normal	P3	User A	FIXED	Kategori 1
PC	Linux	abuzila	snjezana.p	Platform	normal	P3	SWT	FIXED	Kategori 2
PC	Linux-GTK	abuzila	abuzila	Platform	normal	P3	SWT	FIXED	Kategori 2
PC	Windows	agarcher	amwright	Platform	normal	P3	User A	INVALID	Kategori 2
All	All	aeschli	preuss	JDT	enhancem	P3	UI	WORKSFO	Kategori 2
PC	Windows	aeschli	akiezun	JDT	normal	P3	UI	WONTFIX	Kategori 2

3.1.3 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). Dalam laporan *bug* dapat terlihat perbedaan mencolok antara *bug* yang mencatatkan waktu perbaikan sangat cepat atau sangat lama. Perbedaan mencolok dapat dikatakan kurang "normal" atau bias dalam siklus hidup sebuah laporan *bug*. Banyak *bug* yang dilaporkan dengan informasi yang salah atau duplikasi dengan laporan lain, sehingga membuat pengembang untuk memilih laporan *bug* 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 tabel 3.5 menampilkan contoh dataset yang memiliki data laporan *bug* yang kurang normal. Pada *bug* ID 492035 adalah salah satu laporan *bug* yang pengerjaannya sangat cepat dikarenakan duplikasi dari laporan *bug* yang lain. Pada *bug* ID 103420 adalah laporan *bug* yang kurang normal dikarenakan memakan waktu terlalu lama dan ternyata hasil pengerjaannya *invalid*.



Gambar 3.2 Visualisasi kuartil pada data laporan *bug*

Tabel 3.5 Contoh potongan dataset eclipse 2016 yang memiliki laporan bias

Bug ID	Perangkat	Sistem_Operasi	Ditugaska	Pelapor	produk	kerasnya	priorit	kompone	resolusi	waktu_dibuka	waktu_selesai	Waktu_Pe
492035	PC	Windows 7	4diac-inb	brian.bro	JDT	normal	P3	Core	DUPLICAT	19-04-16 13:51	19-04-16 14:01	0
491728	PC	Linux	akurtako	akurtakov	Platform	normal	P3	UI	FIXED	14-04-16 14:29	14-04-16 15:28	0
486016	PC	Linux	akurtako	akurtakov	Platform	normal	P3	User Assis	FIXED	18-01-16 3:23	18-01-16 4:41	1
496890	PC	Linux	akurtako	akurtakov	Platform	normal	P3	User Assis	FIXED	28-06-16 1:17	28-06-16 5:26	4
478618	PC	Linux	abuzila	snjezana.	Platform	normal	P3	SWT	FIXED	29-09-15 7:49	04-05-16 9:16	5233
445456	PC	Linux-GTK	abuzila	abuzila	Platform	normal	P3	SWT	FIXED	30-09-14 5:03	21-09-16 12:14	17335
103420	PC	Windows XP	agarcher	amwright	Platform	normal	P3	User Assis	INVALID	11-07-05 18:56	03-08-16 8:17	96973
43082	All	All	aeschli	preuss	JDT	enhancem	P3	UI	WORKSFO	15-09-03 7:52	04-07-16 14:23	112230
37322	PC	Windows 2000	aeschli	akiezun	JDT	normal	P3	UI	WONTFIX	07-05-03 9:40	10-06-16 5:11	114787

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 (2012). 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. *Bug* yang termasuk dalam rentang waktu akan disaring atau dihapus dari dataset.

Pembentukan *treshold* akan dilakukan dengan membentuk bagian yang seimbang yaitu kuartil. Visualisasi kuartil dapat dilihat pada gambar 3.2. 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 (R. L. Scheaffer, M. Mulekar, 2010). Pada penelitian ini akan menyaring data laporan pada data dibawah dengan persamaan 3.13.

$$Letak\ Q_i = \frac{i\ (n+1)}{4} \tag{3.12}$$

$$X = ½ * \text{Q1} \tag{3.13}$$

Mendapatkan nilai kuartil 1,2, dan 3 akan dihitung menggunakan persamaan 3.12, dimana i adalah kuartil ke i, dan n adalah jumlah data.

3.1.4 Pembangunan Pendekatan Baru 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. Weka dapat mengimplementasikan banyak algoritma termasuk hutan acak. Algoritma *pseudocode* dari hutan acak dapat dilihat pada gambar 3.3.

Pada gambar 3.3 menunjukkan pseudocode algoritma hutan acak bekerja. Langkah pertama adalah untuk setiap pohon di hutan, kita memilih sebuah sampel *bootstrap* dari S, dimana S menunjukkan *i bootstrap.*

Algoritma 1 hutan acak

Prasyarat : A *training set* **S** := $(x_1,y_1),\ldots,(x_n,y_n)$,
Fitur **F**, dan jumlah pohon dalam hutan **B**

```
1   Function HutanAcak (S,F)
2     H ← ∅
3     for i ∈ 1,…, B do
4                 S (i) ← A contoh bootstrap dari S
5                 hi ← RandomizedTreeLearn (S (i),F)
6                 Hi ← H ∪ {hi}
7     end for
8     Return H
9   end function
10  function RandomizedTreeLearn (S,F)
11    Pada setiap node:
12                f ← bagian (subset) yang sangat kecil dari F
13                Dipisah dari fitur terbaik di f
14    return The learned tree
15  end function
```

Gambar 3.3 Pseudocode hutan acak (Breiman, 2010)

Langkah berikutnya adalah belajar pohon keputusan menggunakan algoritma pembelajaran pohon keputusan yang telah dimodifikasi. Algoritma pohon keputusan yang dimodifikasi adalah sebagai berikut: Pada setiap node dari pohon, bukannya memeriksa semua kemungkinan *feature-split*, tetapi secara acak memilih beberapa subset dari fitur $f \subseteq F$, dimana F adalah serangkaian fitur. Node kemudian dibagi pada fitur terbaik di *f* daripada F, dan dalam prakteknya *f* jauh lebih kecil dari F. Pemutusan untuk di mana fitur untuk dibagi seringkali menjadi aspek komputasi yang paling mahal dari pembelajaran pohon keputusan. Mempersempit set fitur, kita dapat mempercepat belajar dari pohon secara drastis.

3.1.5 Alat ukur

Uji coba selanjutnya akan mengukur kinerja atau performa model prediksi yang akan dilakukan perhitungan akurasi dengan menggunakan 10-*fold cross validation*, Ketelitian (P), Sensitivitas (R) dan Akurasi (A) yang dapat dilihat pada

persamaan 3.14,3.15, 3.16, dan 3.17. *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 3.14. setelah itu akan dilakukan iterasi sebanyak *K* kali. Pada penelitian ini akan menggunakan *K* =10. Pada iterasi ke k, subset Bk menjadi *test set*, sedangkan *subset* yang lain menjadi *training set*.

$$\cup_{k=1}^{K} B_k = \{1, \dots, n\} \; dan \; B_j \cap B_k = \emptyset \; (j \neq k) \tag{3.14}$$

$$P = \frac{True\ Positif}{(True\ Positif + False\ Positif)} \tag{3.15}$$

$$R = \frac{True\ Positif}{(True\ Positif + False\ Negatif)} \tag{3.16}$$

$$A = \frac{(TP + TN)}{(TP + FP + FN + TN)} \tag{3.17}$$

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.

3.1.6 Metode pada Evaluasi

Evaluasi akan dilakukan dengan membandingkan hasil presisi, sensitivitas dan akurasi metode yang diusulkan dengan dengan metode pohon keputusan dan *naïve bayes*. Setiap metode pembanding akan menggunakan hasil presisi,

sensitivitas, dan akurasi dari persamaan 3.15, 3.16, dan 3.17. Metode pohon keputusan dan *naïve bayes* akan menggunakan dataset yang sama seperti digunakan oleh pengujian metode yang diusulkan, yaitu Mozilla firefox dan Eclipse.Metode pohon keputusan dan *naïve bayes* akan menggunakan jumlah fitur dan kelas yang sama. Hasil presisi, sensitivitas, dan akurasi akan dibandingkan terhadap hasil perhitungan metode yang di usulkan.

[Halaman ini sengaja dikosongkan]

BAB IV

PENGUJIAN DAN EVALUASI

Pada bab ini akan dibahas tentang evaluasi hasil pengujian. Pembahasan pertama adalah lingkungan pengujian yang terdiri dari perangkat keras dan perangkat lunak. Pembahasan kedua adalah skenario pengujian dan evaluasi dari setiap skenario untuk mendapatkan kinerja terbaik melalui perbandingan performa akurasi, sensitivitas, presisi. Pembahasan terakhir adalah tentang analisa hasil pengujian.

4.1 Lingkungan Uji Coba

Lingkungan pengujian pada penelitian ini meliputi spesifikasi perangkat keras yang ditunjukkan pada Tabel 4.1 dan spesifikasi perangkat lunak yang ditunjukkan pada Tabel 4.2.

Tabel 4.1 Spesifikasi Perangkat Keras

No	Nama	Spesifikasi
1.	Prosesor	Intel(R) Core(TM) i7-4710HQ CPU @ 2.50Hz
2.	Memori	8.00 GB
3.	Hard disk	1 Terra
4.	VGA	NVIDIA GeForce 840M 4GB

Tabel 4.2 Spesifikasi Perangkat Lunak

No	Nama	Spesifikasi
1.	Sistem Operasi	Windows 8.1 Pro 64 bit
2.	Proses Data	Ms. Excel 2010
3.	Weka	v. 3.8

4.2 Pengujian Pendekatan

Penelitian ini menggunakan dataset yang berasal dari Bugzilla. Pengujian akan menggunakan empat kelompok dataset seperti yang dijelaskan pada subbab 3.1.1. pengujian akan dilakukan menjadi dua tahap, yaitu praproses yang

melibatkan pemilihan *class*, penyaringan dataset dan proses klasifikasi menggunakan weka dengan metode hutan acak.

4.2.1 Pra Proses

Pada tahap ini akan dilakukan praproses seperti yang telah dijelaskan pada sub bab 3.1.2 dan 3.1.3. Proses pertama yang akan dilakukan adalah pengelompokkan kelas pada setiap data laporan *bug* didalam dataset. Proses kedua adalah menyaring laporan dengan menghitung letak kuartil sebagai batas dengan persamaan 3.12 dan menghitung jumlah data yang akan dihapus dengan menggunakan persamaan 3.13.

4.2.1.1 Pembagian *Class*

Pada bagian sub bab ini akan melakukan percobaan untuk melihat hasil dari penggunaan dataset dengan *class* yang berbeda. Kelompok dataset 1 akan menggunakan kuartil sebagai *threshold* data dan menggunakan persamaan 3.4, 3.5, 3.6, dan 3.7. Kelompok dataset 2 akan menggunakan pembagian *class* berdasarkan waktu yang di tentukan dengan menerapkan persamaan 3.8, 3.9, 3.10, 3.11.

Tabel 4.3 Jumlah pengkategorian setiap *class* kelompok dataset 1

No.	Dataset	Kategori 1	Kategori 2	Kategori 3	Kategori 4	Total
1.	Firefox 1	4124	4124	4124	4123	16.495
2.	Firefox 2	6860	6860	6860	6858	27.438
3.	Eclipse 1	4375	4375	4375	4374	17.499
4.	Eclipse 2	4490	4490	4491	4490	17.961

Pada tabel 4.3 menunjukkan dataset firefox 1 memiliki 4124 data pada kategori 1, kategori 2, dan kategori 3, pada kategori 4 memiliki 4123 data. Dataset firefox 2 memiliki 6860 data pada kategori 1, kategori 2, dan kategori 3, pada kategori 4 memiliki 6858 data. Pada dataset Eclipse 1 memiliki 4375 data pada kategori 1, kategori 2, dan kategori 3, pada kategori 4 memiliki 4374 data. Pada dataset Eclipse 2 memiliki 4490 data pada kategori 1, kategori 2, dan kategori 4, pada kategori 3 memiliki 4490 data. Dataset pada kelompok 1 terlihat pada tabel

4.3 menampilkan pembagian *class* sangat merata pada dataset.

Tabel 4.4 Jumlah pengkategorian setiap *class* kelompok dataset 1

No.	Dataset	0-1 bulan	1-3 bulan	3 bulan – 1 tahun	1 tahun - ∞	Total
1.	Firefox 1	1437	634	1000	13424	16.495
2.	Firefox 2	8498	5549	7040	6351	27.438
3.	Eclipse 1	8347	2395	2484	4273	17.499
4.	Eclipse 2	8364	2384	2876	4337	17.961

Pada tabel 4.4 menampilkan jumlah data pada masing-masing *class* pada kelompok dataset 2. Pada dataset Firefox 1 memiliki data 1437 pada *class* 0-1 bulan, 634 data pada *class* 1-3 bulan, 1000 data pada *class* 3 bulan - 1 tahun, dan 13424 data pada *class* 1 tahun - ∞. Pada dataset Firefox 2 memiliki data 8498 pada *class* 0-1 bulan, 5549 data pada *class* 1-3 bulan, 7040 data pada *class* 3 bulan - 1 tahun, dan 6351 data pada *class* 1 tahun - ∞. Pada dataset Eclipse 1 memiliki data 8364 pada *class* 0-1 bulan, 2395 data pada *class* 1-3 bulan, 2484 data pada *class* 3 bulan - 1 tahun, dan 4337 data pada *class* 1 tahun - ∞.

4.2.1.2 Penyaringan dataset

Pada tahapan penyaringan dataset akan dilakukan seperti yang telah dijelaskan pada sub bab 3.1.3. Langakah awal dalam penyaringan adalah dengan mencari letak posisi data (X) dengan menghitung kuartil. Penyaringan data akan dilakukan dengan menerapkan persamaan 3.7. Data yang akan disaring terdapat pada data yang berada dibawah kuartil 1, yaitu data yang memiliki nilai waktu perbaikan yang memiliki nilai lebih rendah dari data yang terletak pada data kuartil 1. Hasil perhitungan kuartil dan penyaringan dapat dilihat pada tabel 4.4.

Tabel 4.5 Data letak kuartil dan jumlah hasil penyaringan dataset

No.	Dataset	Q1	Q2	Q3	Saring (data)	Total Dataset
1.	Firefox 1	X_{4124}	X_{8248}	X_{12372}	2.062	14.434
2.	Firefox 2	X_{6860}	X_{13720}	X_{20580}	3.430	24.009

3.	Eclipse 1	X_{4375}	X_{8750}	X_{13125}	2.187	15.312
4.	Eclipse 2	X_{4490}	X_{8981}	X_{13471}	2.245	15.716

Tabel 4.4 menunjukkan letak data kuartil (X) dan jumlah data yang harus dikurangi pada bagian kuartil 1 (Q1). Pada dataset firefox 1 akan dikurangi 2.062 data. Pada dataset firefox 2 akan dikurangi 3.430 data. Pada dataset Eclipse 1 akan dikurangi 2.187 data. Pada dataset Eclipse 2 akan dikurangi 2.245 data. Jumlah data akan disaring pada dataset yang berada dibawah data kuartil 1. Jumlah dataset setelah penyaringan pada dataset Firefox 1 adalah 14.434. Jumlah dataset setelah penyaringan pada dataset Firefox 2 adalah 24.009. Jumlah dataset setelah penyaringan pada dataset Eclipse 1 adalah 15.312. Jumlah dataset setelah penyaringan pada dataset Eclipse 2 adalah 15.716.

4.2.2 Hasil Pengujian

Pada sub bab ini akan menunjukkan hasil pengujian yang telah dilakukan penulis. Pengujian dilakukan dengan menghitungan pendekatan prediksi menggunakan metode hutan acak. Hasil yang didapatkan akan berasal dari hasil, presisi, sesitivitas (*recall*), dan akurasi. Pengujian akan dibagi menjadi 2 skenario. Pengujian skenario pertama akan melakukan pengujian terhadap pembagian *class* yang dipilih dan hasilnya akan memilih kelompok *class* yang lebih baik untuk digunakan pada scenario kedua. Pengujian scenario kedua akan dilakukan dengan dataset dari Mozilla firefox dan Eclipse sebelum penyaringan dengan yang sudah disaring pada tahap praproses.

4.2.2.1 Hasil Pengujian Skenario 1 kelompok kelas dataset 1 dan 2

Pada tahap ini akan melakukan pengujian terhadap dataset yang menggunakan aturan pembagian *class* yang berbeda. Kelompok *class* 1 menggunakan *threshold* kuartil sebagai aturan pembagian. Kelompok *class* 2 menggunakan pembagian berdasarkan waktu perbaikan sebagai aturan. Pengujian akan menggunakan presisi dan sensitivitas sebagai perbandingan hasil.

Tabel 4.6 Hasil pengujian kelompok kelas dataset 1

No.	Dataset	Presisi	Sensitivitas
1.	Firefox 1	0.713	0.717
2.	Firefox 2	0.535	0.537
3.	Eclipse 1	0.689	0.696
4.	Eclipse 2	0.697	0.709

Pada tabel 4.6 memperlihatkan hasil dataset Firefox 1 mendapatkan presisi 0.713 dan sensitifitas 0.717. Pada dataset Firefox 2 mendapatkan presisi 0.535 dan sensitifitas 0.537. Pada dataset Eclipse 1 mendapatkan presisi 0.689 dan sensitifitas 0.696. Pada dataset Eclipse 2 mendapatkan presisi 0.697 dan sensitifitas 0.709.

Tabel 4.7 Hasil pengujian kelompok kelas dataset 2

No.	Dataset	Presisi	Sensitivitas
1.	Firefox 1	0.819	0.851
2.	Firefox 2	0.535	0.537
3.	Eclipse 1	0.714	0.738
4.	Eclipse 2	0.713	0.736

Pada tabel 4.6 memperlihatkan hasil dataset Firefox 1 mendapatkan presisi 0.713 dan sensitifitas 0.717. Pada dataset Firefox 2 mendapatkan presisi 0.535 dan sensitifitas 0.537. Pada dataset Eclipse 1 mendapatkan presisi 0.714 dan sensitifitas 0.738. Pada dataset Eclipse 2 mendapatkan presisi 0.713 dan sensitifitas 0.736. Dataset Firefox 1 mendapatkan hasil yang baik dari pada dataset lainnya. Penurunan presisi, sensitivitas, dan akurasi terjadi pada dataset lainnya. Penurunan Presisi dan Sensitivitas disebabkan karena pada dataset firefox 2 memiliki penyebaran data yang kurang baik sehingga presisi dan sensitivitas mendapatkan hasil kurang baik. Pada dataset Eclipse mendapatkan hasil presis dan sensitivitas yang cukup baik.

Tabel 4.8 Perbandingan performa pembagian *class*

No.	Evaluasi	Dataset	Kelompok 1	Kelompok 2	Selisih
1.	Presisi	Firefox 1	0.713	**0.819**	0.106
		Firefox 2	0.535	0.535	0
		Eclipse 1	0.689	**0.714**	0.025
		Eclipse 2	0.697	**0.713**	0.016
2.	Sensitivitas	Firefox 1	0.717	**0.851**	0.134
		Firefox 2	0.537	0.537	0
		Eclipse 1	0.696	**0.738**	0.042
		Eclipse 2	0.709	**0.736**	0.027

Pada tabel 4.8 Menunjukkan hasil perbandingan dari presisi dan sensitivitas kelompok dataset 1 yang dengan menggunakan pembagian *class* kuartil, dan kelompok dataset 2 yang dengan menggunakan pembagian *class* waktu. Hasil yang didapatkan dari perbandingan pada tabel 4.8 adalah dataset kelompok 2 mendapatkan yang lebih baik. Presisi dari kelompok 2 lebih baik pada dataset Firefox 1 dengan selisih 0.106, Eclipse 1 dengan selisih 0.025, dan Eclipse 2 dengan selisih 0.016. Sensitivitas dari kelompok 2 lebih baik pada dataset Firefox 1 dengan selisih 0.134, Eclipse 1 dengan selisih 0.042, dan Eclipse 2 dengan selisih 0.027. Hasil dari tabel 4.8 menunjukkan kelompok 2 lebih baik, maka pengujian selanjutnya akan menggunakan pembagian *class* berdasarkan kelompok 2.

4.2.2.2 Hasil Pengujian Skenario 2 Penyaringan Dataset

Pada tahap ini akan melakukan pengujian skenario 2, yaitu melakukan perhitungan dengan hutan acak terhadap semua dataset. Pengujian akan menghasilkan presisi, sensitivitas, dan akurasi. Pengujian ini akan membandingkan hasil perhitungan terhadap setiap dataset dari hasil perhitungan sebelum penyaringan dan hasil perhitungan sesudah penyaringan.

Tabel 4.9 Hasil pengujian setiap dataset sebelum penyaringan

No.	Dataset	Presisi	Sensitivitas	Akurasi (%)
1.	Firefox 1	0.819	0.851	85.05

2.	Firefox 2	0.535	0.537	55.08
3.	Eclipse 1	0.714	0.738	73.75
4.	Eclipse 2	0.713	0.736	73.62

Pada tabel 4.9 menunjukkan hasil pengujian yang telah dilakukan penulis. Pada dataset firefox 1 mendapatkan hasil yang cukup baik dengan presisi 0.819, sensitivitas 0.851, dan akurasi mencapai 85.05%. Pada dataset firefox 2 mengalami penurunan performa dengan presisi 0.535 dan sensitivitas mendapatkan hasil 0.537 dengan akurasi 55.08%. Pada dataset Eclipse 1 hasil performa dengan presisi 0.714 dan sensitivitas mendapatkan hasil 0.738 dengan akurasi 73.75%. Pada dataset Eclipse 2 hasil performa dengan presisi 0.713 dan sensitivitas mendapatkan hasil 0.736 dengan akurasi 73.62%.

Penurunan Performa pada dataset dikarenakan terdapat banyaknya penyebaran data pada setiap atribut terhadap *class* cukup merata, sehingga dapat menurunkan performa metode klasifikasi. Dataset Firefox yang berisi 27.439 mendapatkan hasil yang kurang baik karena banyak data pada atribut terdistribusi dan saling berkorelasi dengan data yang lain. Pada dataset Firefox 2 kelemahan terdapat pada *class* (1-3 bulan) yang memiliki banyak kesalahan klasifikasi dengan 1902 data benar dan 3647 data salah, hasil performa mendapatkan presisi 0.441 dan sensitivitas 0.343. Pada dataset Eclipse 1 dan 2 memiliki kelemahan yang sama pada *class* (1-3 bulan) dan (3 bulan – 1 tahun). Pada Eclipse 1 *class* (1-3 bulan) mendapatkan hasil yang kurang baik dengan presisi 0.415 dan sensitivitas 0.319, sedangkan (3 bulan – 1 tahun) mendapatkan presisi 0.440 dan sensitivitas 0.319. Pada Eclipse 2 *class* (1-3 bulan) mendapatkan hasil yang kurang baik dengan presisi 0.390 dan sensitivitas 0.301, sedangkan (3 bulan – 1 tahun) mendapatkan presisi 0.454 dan sensitivitas 0.359.

Tabel 4.10 Hasil pengujian setiap dataset sesudah penyaringan

No.	Dataset	Presisi	Sensitivitas	Akurasi (%)
1.	Firefox 1	0.794	0.829	82.88
2.	Firefox 2	0.556	0.559	55.91

3.	Eclipse 1	0.797	0.799	79.85
4.	Eclipse 2	0.703	0.716	71.57

Pada tabel 4.10 menampilkan hasil dari performa perhitungan metode hutan acak dengan melakukan praproses penyaringan dataset. Pada dataset Firefox 1 mendapatkan hasil presisi dengan nilai 0.794, sensitivitas dengan 0.829 dan akurasi sebesar 82.88%. Pada dataset Firefox 2 mendapatkan hasil performa dengan presisi 0.556, sensitivitas 0.559 dan akurasi 55.91%. Pada dataset Eclipse 1 mendapatkan hasil performa cukup baik dengan presisi 0.797, sensitivitas 0.799 dan akurasi 79.85%. Pada dataset Eclipse 2 mendapatkan presisi 0.703, sensitivitas 0.716 dan akurasi 71.57%.

Tabel 4.11 Perbandingan performa metode dengan menggunakan penyaringan

No.	Evaluasi	Dataset	Tanpa Penyaringan	Penyaringan	Selisih
1.	Presisi	Firefox 1	**0.819**	0.794	0.025
		Firefox 2	0.535	**0.556**	0.021
		Eclipse 1	0.714	**0.797**	0.083
		Eclipse 2	**0.713**	0.703	0.01
2.	Sensitivitas	Firefox 1	**0.851**	0.829	0.022
		Firefox 2	0.537	**0.559**	0.022
		Eclipse 1	0.738	**0.799**	0.061
		Eclipse 2	**0.736**	0.716	0.02
3.	Akurasi	Firefox 1	**85.05**	82.88	2.17
		Firefox 2	55.08	**55.91**	0.83
		Eclipse 1	73.75	**79.85**	6.1
		Eclipse 2	**73.62**	71.57	2.05

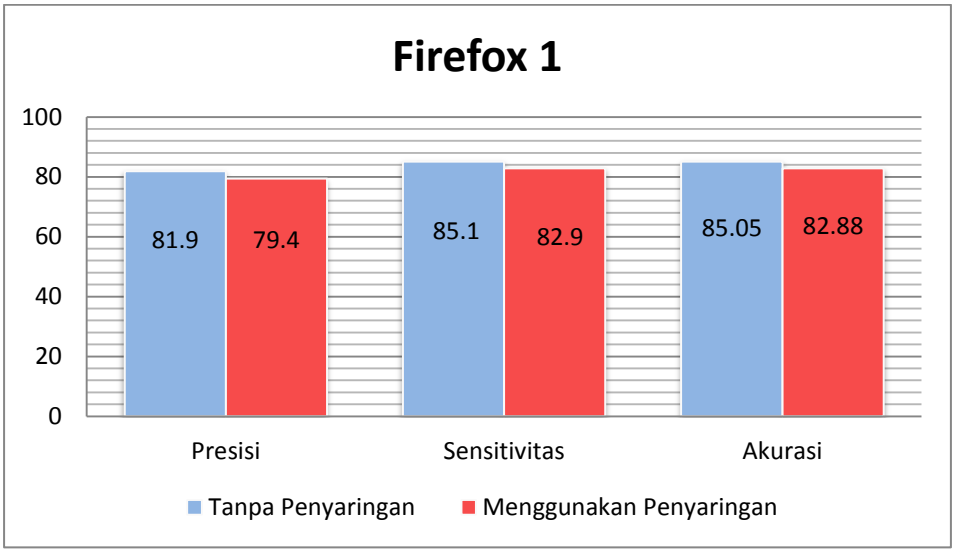
Pada tabel 4.11 menunjukkan hasil perbandingan performa metode antara menggunakan praproses penyaringan dengan tanpa menggunakan praproses penyaringan. Pada dataset Firefox 1 mendapatkan hasil yang menurun ketika digunakan praproses penyaringan. Dataset Firefox 1 mendapatkan hasil presisi

tanpa penyaringan dengan 0.819 dan menurun sebanyak 0.025 menjadi 0.794. Dataset Firefox 1 mendapatkan hasil sensitivitas tanpa penyaringan dengan 0.851 dan menurun sebanyak 0.022 menjadi 0.829. Dataset Firefox 1 mendapatkan hasil akurasi tanpa penyaringan dengan 85.05% dan menurun sebanyak 2.17% menjadi 82.88%.

Pada tabel 4.11 dataset Firefox 2 mendapatkan hasil yang meningkat dari hasil perhitungan metode sebelum menggunakan praproses penyaringan dan setelah menggunakan praproses penyaringan. Dataset Firefox 2 mendapatkan hasil presisi tanpa penyaringan dengan 0.535 dan meningkat sebanyak 0.021 menjadi 0.556. Dataset Firefox 2 mendapatkan hasil sensitivitas tanpa penyaringan dengan 0.537 dan meningkat sebanyak 0.022 menjadi 0.559. Dataset Firefox 2 mendapatkan hasil akurasi tanpa penyaringan dengan 55.08% dan meningkat sebanyak 0.83 % menjadi 55.91%.

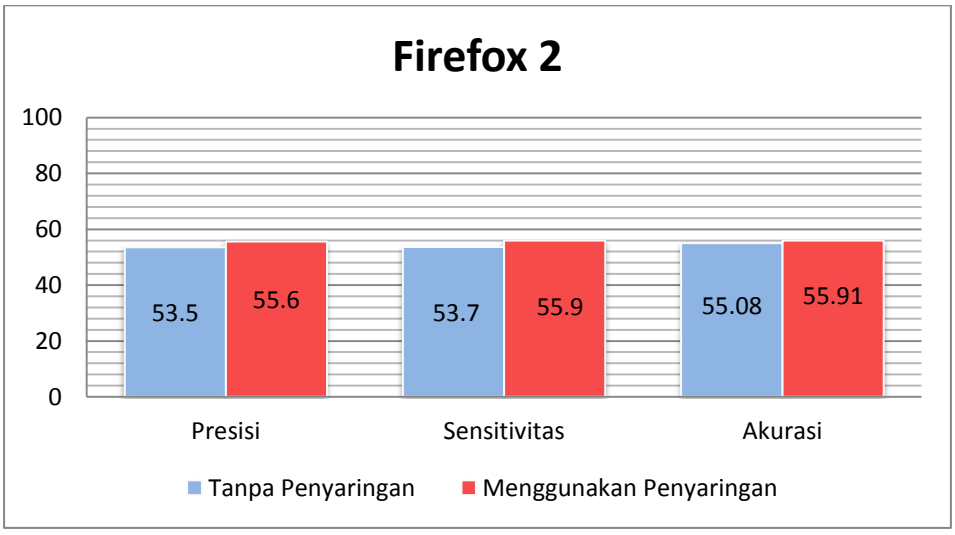
Pada tabel 4.11 dataset Eclipse 1 mendapatkan hasil yang meningkat dari hasil perhitungan metode sebelum menggunakan praproses penyaringan dan setelah menggunakan praproses penyaringan. Dataset Eclipse 1 mendapatkan hasil presisi tanpa penyaringan dengan 0.714 dan meningkat sebanyak 0.083 menjadi 0.797. Dataset Eclipse 1 mendapatkan hasil sensitivitas tanpa penyaringan dengan 0.738 dan meningkat sebanyak 0.061 menjadi 0.799. Dataset Eclipse 1 mendapatkan hasil akurasi tanpa penyaringan dengan 73.75% dan meningkat sebanyak 6.1% menjadi 79.85%.

Pada tabel 4.11 dataset Eclipse 2 mendapatkan hasil yang menurun dari hasil perhitungan metode sebelum menggunakan praproses penyaringan dan setelah menggunakan praproses penyaringan. Dataset Eclipse 2 mendapatkan hasil presisi tanpa penyaringan dengan 0.713 dan menurun sebanyak 0.01 menjadi 0.703. Dataset Eclipse 2 mendapatkan hasil sensitivitas tanpa penyaringan dengan 0.736 dan menurun sebanyak 0.02 menjadi 0.716. Dataset Eclipse 2 mendapatkan hasil akurasi tanpa penyaringan dengan 73.62% dan menurun sebanyak 2.05% menjadi 71.57%.



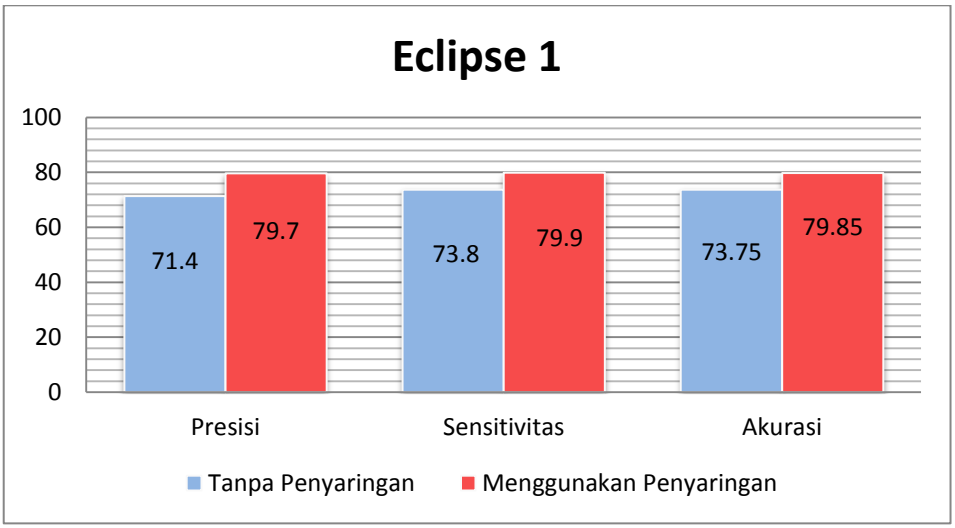
Gambar 4.1 Grafik hasil perbandingan pengujian penyaringan dataset firefox 1

Pada gambar 4.1 menampilkan grafik hasil dari perbandingan pengujian penyaringan dataset firefox 1. Pada grafik 4.1 nilai presisi dan sensitivitas di diubah dengan mengkalikan nilai presisi dan sensitivitas dengan angka 100. Pengubahan angka bertujuan untuk mempermudah untuk melihat perbandingan pada grafik. Penurunan performa pada dataset Firefox 1 sebesar 2.17% karena pada saat melakukan penyaringan data sebanyak 2062 data, hanya terdapat 217 data yang waktu pengerjaannya 0 jam. Dari 217 data hanya terdapat 13 duplikat dan 8 *invalid*. Data yang disaring pada dataset Firefox 1 terdapat 1845 data penting yang seharusnya tidak terhapus.



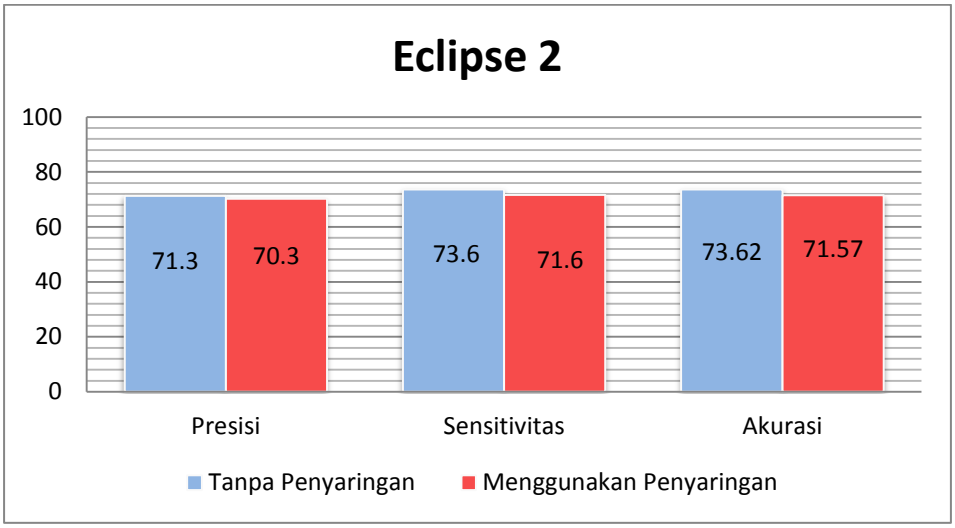
Gambar 4.2 Grafik hasil perbandingan pengujian penyaringan dataset firefox 2

Pada gambar 4.2 menampilkan grafik hasil dari perbandingan pengujian penyaringan dataset firefox 2. Dataset Firefox 2 mengalami peningkatan presisi, sensitivitas dan akurasi setelah dilakukan penyaringan dataset. Penyaringan dataset sebanyak 3430 data terdapat 1259 data yang memiliki waktu 0 jam perbaikan yang terdiri dari 424 duplikat dan 284 *invalid*. Penyaringan data pada dataset Firefox telah menghapus data 781 duplikat, 135 *wont fix*, dan 439 *invalid*. Penyaringan data meningkatkan akurasi sebesar 0.83%.



Gambar 4.3 Grafik hasil perbandingan pengujian penyaringan dataset Eclipse 1

Pada gambar 4.3 menampilkan grafik hasil dari perbandingan pengujian penyaringan dataset Eclipse 1. Dataset Eclipse 1 mengalami peningkatan presisi, sensitivitas dan akurasi setelah dilakukan penyaringan dataset. Penyaringan data pada dataset Eclipse 1 sebanyak 2187 data. Pada dataset Eclipse 1 terdapat 3443 data yang memiliki waktu 0 jam perbaikan yang terdiri dari 1112 duplikat dan 379 *invalid*. Penyaringan data pada dataset Eclipse 1 telah menghapus lebih dari separuh dari data yang dapat menyebabkan bias pada dataset. Penyaringan data meningkatkan akurasi pada dataset Eclipse 1 sebesar 6.1%.



Gambar 4.4 Grafik hasil perbandingan pengujian penyaringan dataset Eclipse 1

Pada gambar 4.4 menampilkan grafik hasil dari perbandingan pengujian penyaringan dataset Eclipse 2. Penurunan performa pada dataset Eclipse 2 sebesar 2.05% karena pada saat melakukan penyaringan data sebanyak 2245 data, hanya terdapat 630 data yang memiliki resolusi duplikat. Dari 2245 data yang di disaring pada dataset hanya terdapat 241 *invalid*. Data yang disaring pada dataset Eclipse 2 terdapat 1374 data penting yang seharusnya tidak terhapus.

Penurunan hasil performa dari praproses penyaringan diakibatkan karena pada saat proses penyaringan terdapat banyak data yang seharusnya tidak ikut di saring. Penyaringan data pada dataset harus hanya yang duplikasi dan memiliki waktu perbaikan 0 jam untuk mendapatkan peningkatan performa presisi, sensitivitas, dan akurasi dari praproses.

4.3 Evaluasi Hasil Pengujian

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 1 yang terisi 16.496 data, firefox 2 yang terisi 27.439 data, eclipse 1 yang terisi 17.499 data dan eclipse 2 yang terisi 17.961 data. Hasil klasifikasi dari setiap dataset akan dibandingkan dengan hasil klasifikasi menggunakan dua metode yaitu pohon keputusan dan *Naïve bayes.* Presisi, sensitivitas , dan akurasi akan dijadikan sebagai perbandingan performa dari setiap metode. Metode yang diusulkan akan menggunakan hasil dari

pengujian menggunakan praproses dan metode hutan acak

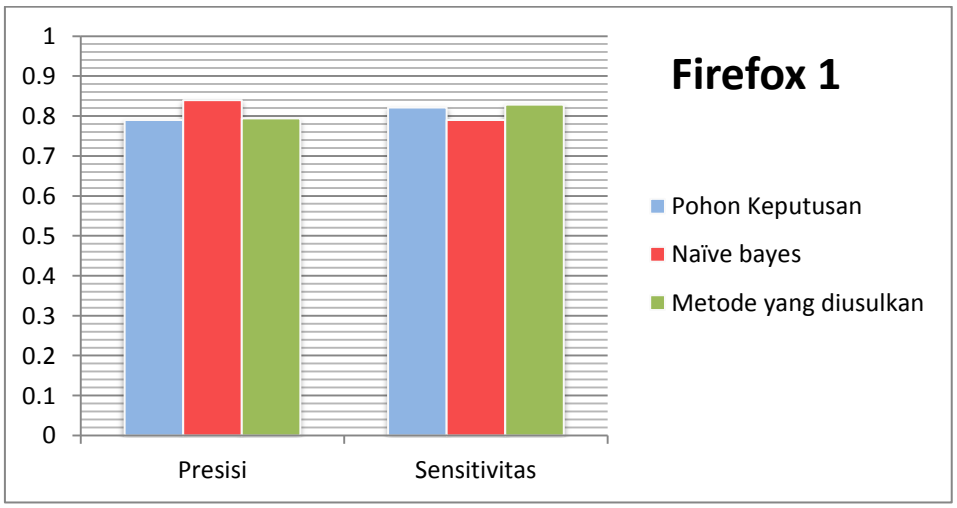
4.3.1 Evaluasi Hasil Pengujian pada Dataset Firefox 1

Pada tabel 4.12 menunjukkan nilai presisi dan sensitivitas metode pohon keputusan, *naïve bayes*, dan metode yang diusulkan pada dataset firefox 1. Nilai presisi tertinggi didapatkan oleh *naïve bayes* dengan 0.840. Nilai sensitifitas tertinggi didapatkan oleh metode yang diusulkan dengan 0.829. Metode yang diusulkan mendapatkan nilai akurasi yang lebih tinggi daripada metode pohon keputusan dan *naïve bayes* dengan 82.88%.

Tabel 4.12 Perbandingan performa metode pada dataset firefox 1

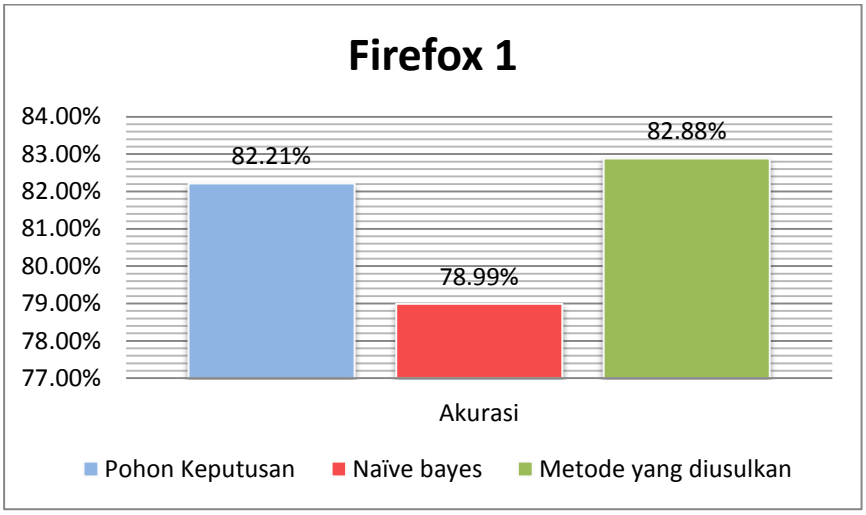
Metode	Firefox 1		
	Presisi	sensitivitas	Akurasi
Pohon keputusan	0.790	0.822	82.21
Naïve bayes	**0.840**	0.790	78.99
Metode usulan	0.794	**0.829**	**82.88**

Pada tabel 4.12 menunjukkan performa metode yang diusulkan bekerja lebih baik pada sensitivitas dan akurasi. Pada presisi *naïve bayes* unggul pada tingkat presisi pada *class* kategori 4 yang mencapai 0.984, sehingga tingkat presisi rata-rata dapat melampau metode yang diusulkan yang hanya mendapat 0.901. Nilai akurasi tertinggi didapatkan oleh metode yang diusulkan dengan 82.88%. Nilai akurasi metode yang diusulkan mendapat nilai selisih 0.67% lebih banyak dari akurasi pohon keputusan.



Gambar 4.5 Grafik hasil evaluasi pengujian dataset firefox 1

Pada gambar 4.5 menunjukkan keunggulan metode yang diusulkan terhadap metode pohon keputusan pada perhitungan presisi dan sensitivitas. Metode yang diusulkan mendapatkan hasil presisi yang lebih sedikit dibandingkan dengan metode *naïve bayes*. Selisih hasil presisi metode yang diusulkan dengan *naïve bayes* hanya sebesar 0.046. Selisih yang didapatkan dari metode yang diusulkan dengan *naïve bayes* diketahui sangat kecil yaitu dua angka dibelakang koma. Perbedaan performa metode yang diusulkan dan pohon keputusan mengungguli *naïve bayes*.



Gambar 4.6 Grafik hasil evaluasi akurasi dataset firefox 1

Metode yang diusulkan mendapatkan persentasi akurasi lebih baik dari pada pohon keputusan dan *naïve bayes* seperti yang dilihat pada gambar 4.6. Pada gambar 4.6 terlihat perbedaan akurasi yang didapatkan dari setiap metode. Metode yang diusulkan memiliki akurasi lebih banyak 0.67 % dari pohon keputusan dan 3.89% dari pada *naïve bayes*.

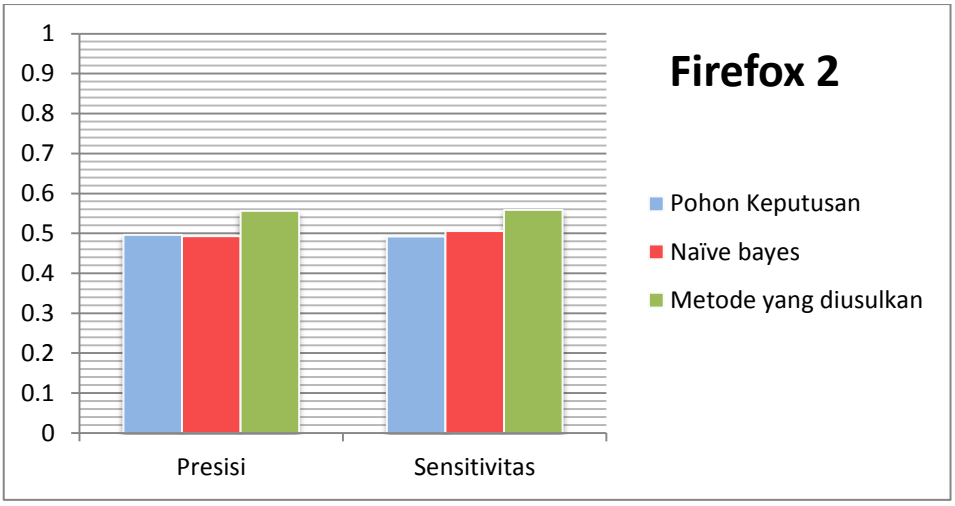
4.3.2 Evaluasi Hasil Pengujian pada Dataset Firefox 2

Tabel 4.13 Perbandingan performa metode pada dataset firefox 2

Metode	Firefox 2		
	Presisi	sensitivitas	Akurasi
Pohon keputusan	0.496	0.492	49.20
Naïve bayes	0.493	0.506	50.55
Metode usulan	**0.556**	**0.559**	**55.91**

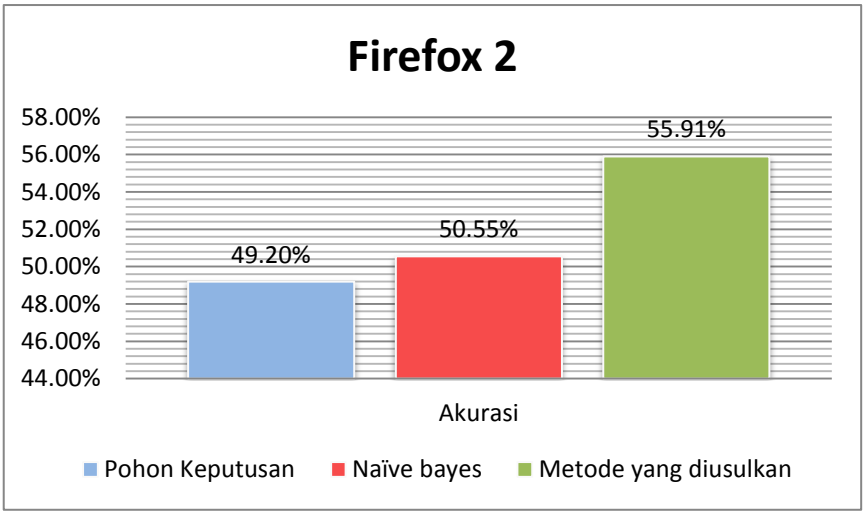
Pada tabel 4.13 dataset firefox 2 menunjukkan nilai presisi tertinggi didapatkan oleh metode yang diusulkan sebesar 0.556. Nilai sensitivitas metode

yang diusulkan mendapatkan nilai 0.559. Nilai presisi tertinggi didapatkan oleh metode yang diusulkan dan diikuti oleh pohon keputusan lalu *naïve bayes*. Nilai sensitifitas tertinggi didapatkan oleh metode yang diusulkan dan selanjutnya *naïve bayes* dengan 0.506. Akurasi tertinggi didapatkan oleh metode yang diusulkan dengan 55.91 dan selanjutnya *naïve bayes* dengan 50.55 dan pohon keputusan 49.20.



Gambar 4.7 Grafik hasil evaluasi pengujian dataset firefox 2

Performa presisi dan sensitivitas metode yang diusulkan pada gambar 4.7 terlihat lebih baik daripada metode pohon keputusan dan *naïve bayes*.Pada gambar 4.7 menunjukkan hasil presisi dan sensitivitas dari metode *naïve bayes* dan metode pohon keputusan hampir sama.



Gambar 4.8 Grafik hasil evaluasi akurasi dataset firefox 2

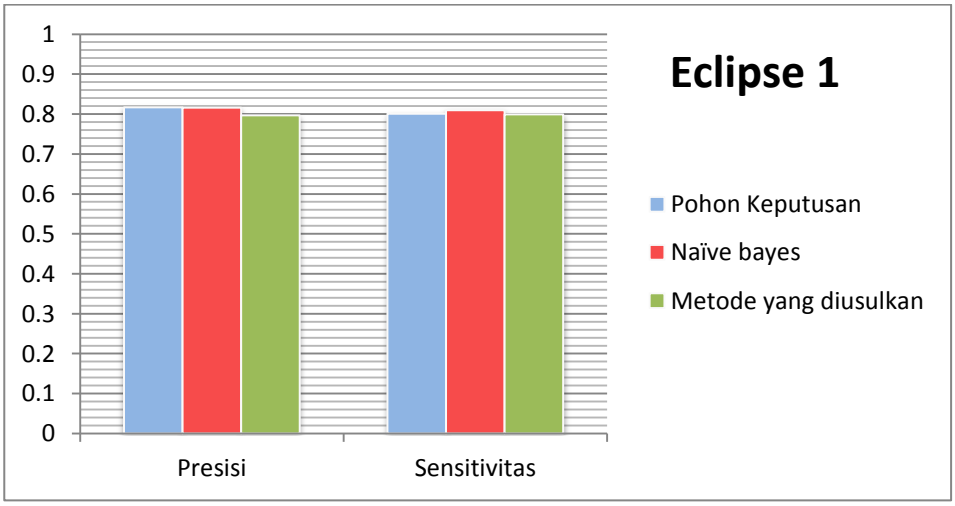
Pada gambar 4.8 menunjukkan nilai akurasi tertinggi didapatkan oleh metode yang diusulkan dengan nilai 55.91%. Nilai akurasi metode yang diusulkan hanya selisih sebesar 5.36% dengan *naïve bayes*. Performa metode *naïve bayes* dan metode yang diusulkan pada dataset firefox 2 memiliki performa yang lebih tinggi dari pada pohon keputusan. Pada dataset firefox 2 menggunakan data sebanyak 27.439 mendapatkan performa yang menurun dari pada dataset Firefox 1.

4.3.3 Evaluasi Hasil Pengujian pada Dataset Eclipse 1

Tabel 4.14 Perbandingan performa metode pada dataset eclipse 1

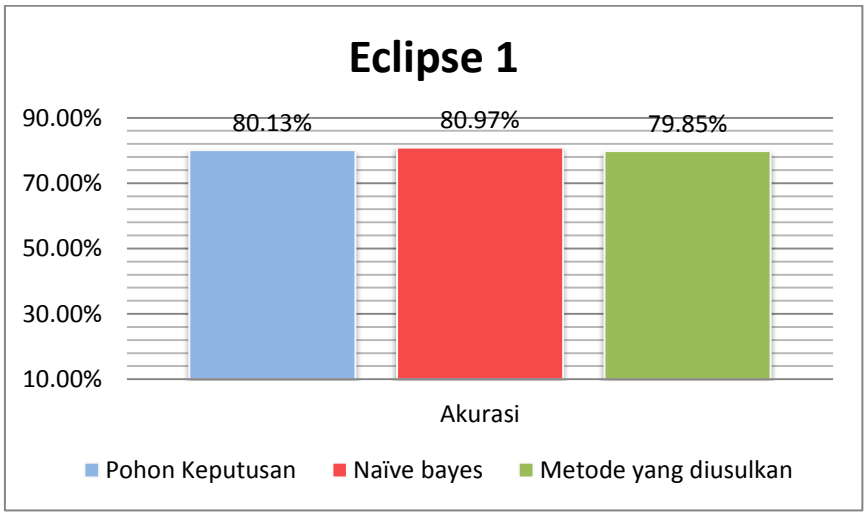
Metode	Eclipse 1		
	Presisi	sensitivitas	Akurasi
Pohon keputusan	**0.817**	0.801	80.13
Naïve bayes	0.816	**0.810**	**80.97**
Metode usulan	0.797	0.799	79.85

Pada tabel 4.14, dataset Eclipse 1 menunjukkan performa pohon keputusan, *naïve bayes*, dan metode yang diusulkan memiliki performa hampir sama yang mencapai diatas 0.7. Presisi tertinggi didapatkan oleh metode pohon keputusan sebesar 0.817 yang diikuti dengan *naïve bayes* sebesar 0.792. Nilai sensitivitas tertinggi didapatkan oleh metode *naïve bayes* sebesar 0.810 dan diikuti oleh metode pohon keputusan sebesar 0.801.



Gambar 4.9 Grafik hasil evaluasi pengujian dataset eclipse 1

Pada gambar 4.9 menunjukkan performa pohon keputusan,, *naïve bayes*, dan metode yang diusulkan. Metode yang diusulkan telihat di gambar 4.9 mendapatkan hasil yang lebih rendah dari metode lain. Performa antara metode yang diusulkan memiliki hasil yang lebih rendah, tetapi tidak berbeda secara signifikan. Metode yang diusulkan memiliki presisi 0.797 dengan selisih hanya 0.02 dari metode pohon keputusan. Metode yang diusulkan memiliki sensitivitas 0.799 dengan selisih hanya 0.011 dari metode *naïve bayes*. Metode yang diusulkan mendapatkan nilai performa yang lebih rendah dapat dikarenakan sensitivitas pada kategori 3 mendapatkan hasil hanya 0.485.



Gambar 4.10 Grafik hasil evaluasi akurasi dataset eclipse 1

Pada gambar 4.10 menunjukkan metode yang diusulkan mendapatkan nilai akurasi yang cukup baik dengan nilai 79.85%. Akurasi metode yang diusulkan terlihat lebih rendah dibandingkan metode lain tetapi tidak berbeda secara signifikan. Metode yang diusulkan terlihat bekerja cukup baik pada dataset ini. Performa dari metode yang diusulkan lebih rendah karena data pada *class* kategori 2 dan kategori 3 memiliki kesalah klasifikasi yang cukup banyak, hal ini karena kedua kategori tersebut memiliki data yang saling berkorelasi. Kategori 2 yang terdeteksi sebagai kategori 3 sebanyak 863 data, sedangakan kategori 3 yang terdeteksi sebagai kategori 2 sebanyak 919. Atribut yang memiliki banyak kesamaan antara kategori 1 dan 2 adalah atribut prioritas.

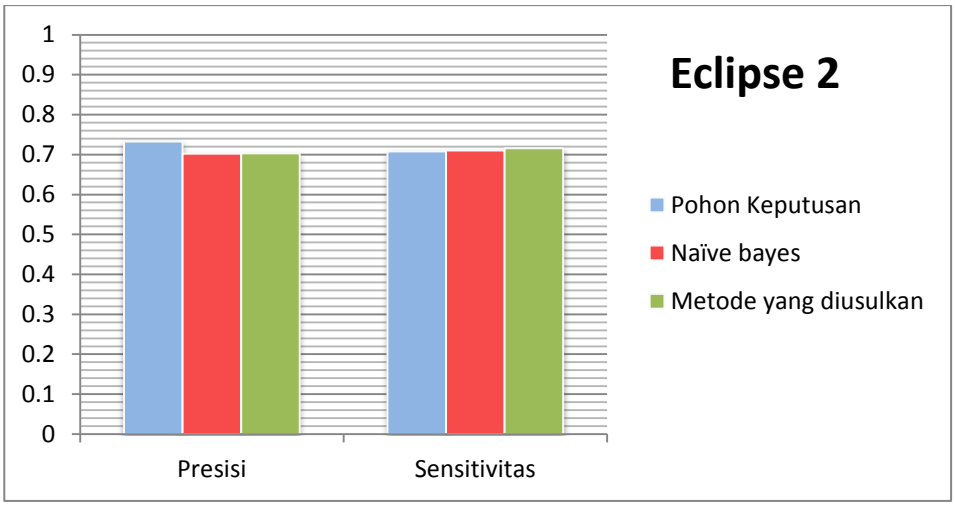
4.3.4 Evaluasi Hasil Pengujian pada Dataset Eclipse 2

Pada tabel 4.15, Nilai presisi tertinggi dataset Eclipse 2 didapatkan oleh metode pohon keputusan sebesar 0.703. Metode yang diusulkan mendapatkan hasil presisi kedua dengan nilai 0.733 lebih rendah dari metode pohon keputusan dengan selisih 0.03. Nilai sensitivitas tertinggi didapatkan oleh metode yang diusulkan sebesar 0.716 dan metode *naïve bayes* menempati nilai tertinggi kedua dengan selisih nilai 0.006.

Tabel 4.15 Perbandingan performa metode pada dataset eclipse 2

Metode	Eclipse 2		
	Presisi	sensitivitas	Akurasi
Pohon keputusan	**0.733**	0.708	70.82
Naïve bayes	0.702	0.710	71.03
Metode usulan	0.703	**0.716**	**71.57**

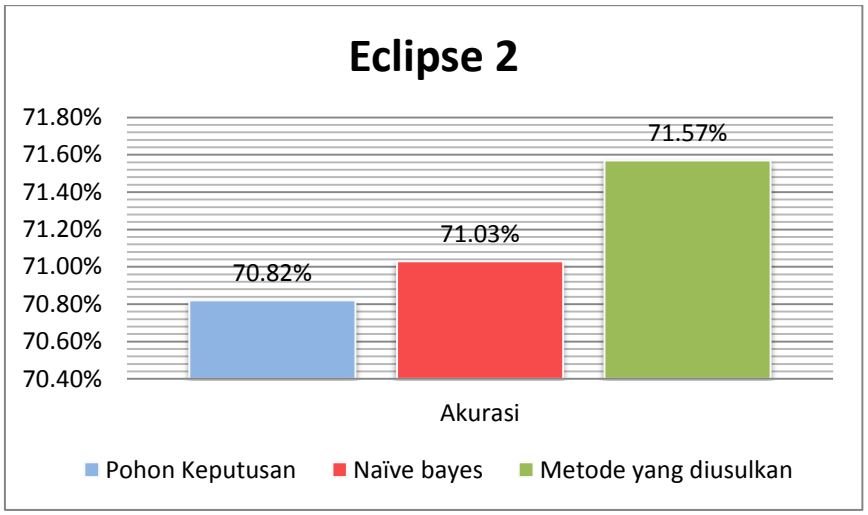
Performa metode yang diusulkan terlihat lebih tinggi dari pada metode metode pohon keputusan dan *naïve bayes*. Performa akurasi dari metode yang diusulkan terlihat pada tabel 4.15 lebih unggul sebesar 71.57% dan selisih lebih besar 0.54% dari hasil akurasi *naïve bayes*.



Gambar 4.11 Grafik hasil evaluasi pengujian dataset eclipse 2

Pada gambar 4.11 menunjukkan performa pohon keputusan, *naïve bayes*, dan metode yang diusulkan. Metode yang diusulkan telihat di gambar 4.11 mendapatkan hasil yang lebih baik dari metode lain pada sensitivitas. Presisi dari metode pohon keputusan memiliki hasil yang lebih baik. Presisi dari metode yang

diusulkan sedikit dibawah dari metode pohon keputusan karena presisi yang didapat dari kategori 2 sangat rendah dengan 0.394.



Gambar 4.12 Grafik hasil evaluasi akurasi dataset eclipse 2

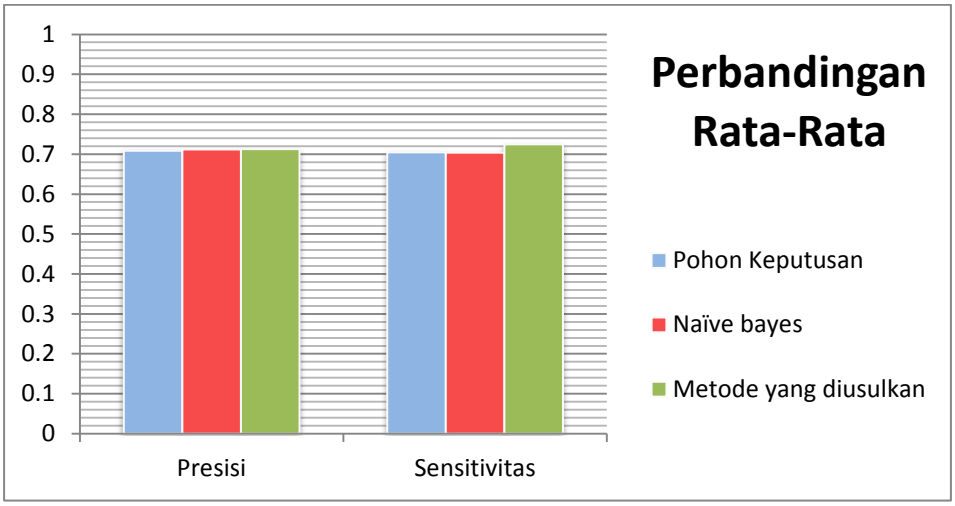
Pada gambar 4.12 menunjukkan akurasi tertinggi didapatkan oleh metode yang diusulkan dengan 71.57%. Metode *naïve bayes* memiliki performa tertinggi kedua dengan 71.03%. Pada dataset Eclipse 2 metode yang diusulkan terlihat pada gambar 4.12 mendapatkan akurasi yang lebih baik dari pada *naïve bayes* dan pohon keputusan.

4.3.5 Perbandingan Hasil Evaluasi

Tabel 4.16 Perbandingan performa rata-rata metode

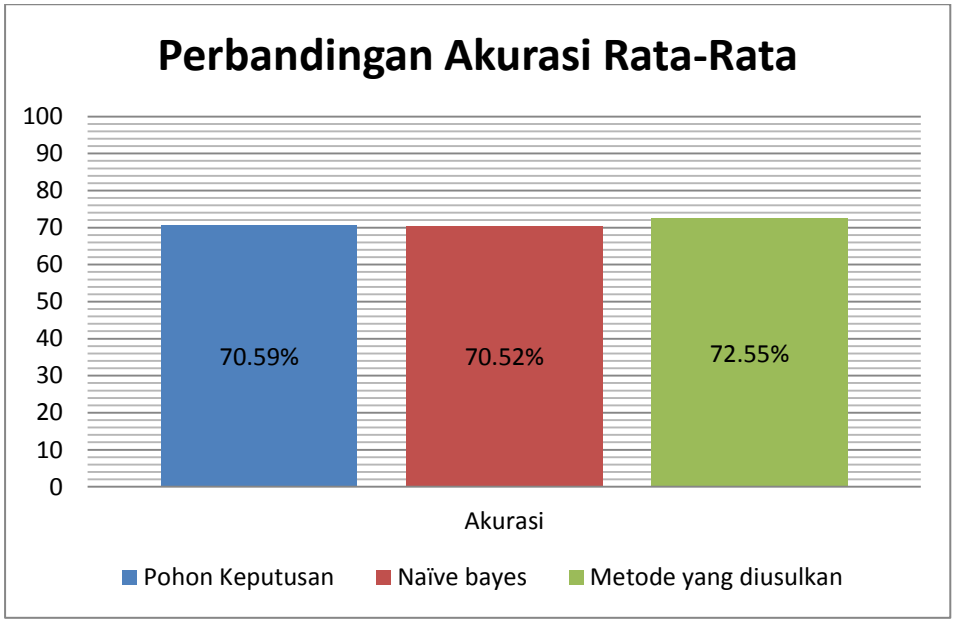
Metode	Perbandingan Rata- Rata		
	Presisi	sensitivitas	Akurasi
Pohon keputusan	0.709	0.705	70.59
Naïve bayes	0.712	0.704	70.52
Metode usulan	**0.7125**	**0.725**	**72.55**

Pada tabel 4.16 menunjukkan performa rata-rata dari metode pohon keputusan, naïve bayes, dan metode yang diusulkan. Metode yang diusulkan memiliki performatertinggi dari presisi, sensitivitas, dan akurasi. Presisi tertinggi didapatkan oleh metode yang diusulkan sebesar 0.7125 yang diikuti dengan metode naïve bayes sebesar 0.712. Nilai sensitivitas tertinggi didapatkan oleh metode yang diusulkan sebesar 0.725 dan diikuti oleh metode pohon keputusan sebesar 0.705.



Gambar 4.13 Grafik hasil rata – rata evaluasi pengujian

Pada gambar 4.13 menunjukkan hasil rata-rata presisi dan sensitivitas dari setiap metode yang dievaluasi dengan menggunakan dataset dari firefox 1, firefox 2, eclipse 1 dan eclipse 2. Pada gambar 4.13 menampilkan hasil metode keputusan memiliki rata-rata hasil presisi dan sensitivitas tertinggi dari pada metode pohon keputusan dan *naïve bayes*. Hasil tertinggi kedua dimiliki oleh metode *naïve bayes* pada presisi. Dari gambar 4.13 dapat disimpulkan ketiga metode bekerja dengan baik dan metode yang diusulkan memiliki hasil presisi dan sensitivitas yang lebih baik.

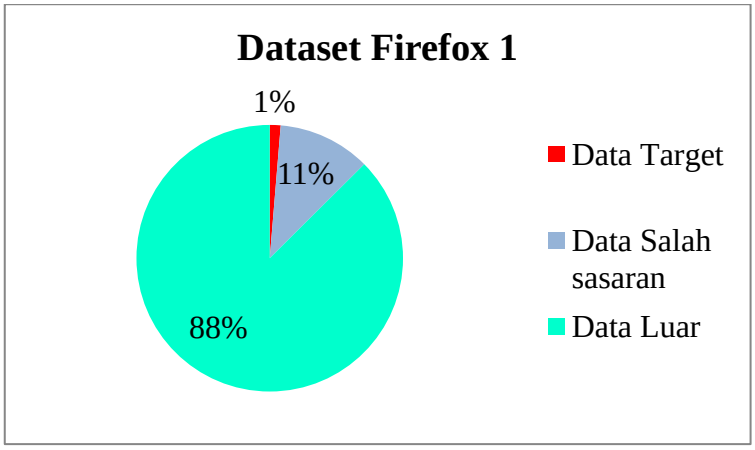


Gambar 4.14 Hasil rata-rata evaluasi akurasi

Pada gambar 4.14 menunjukkan hasil rata-rata persentasi akurasi dari setiap metode yang dievaluasi dengan menggunakan dataset dari firefox 1, firefox 2, eclipse 1 dan eclipse 2. Metode pohon keputusan mendapatkan rata-rata akurasi dengan 70.59%, Metode *naïve bayes* mendapatkan akurasi rata-rata 70.52%. Metode yang diusulkan dengan metode hutan acak ditambah dengan praproses penyaringan dataset mendapatkan hasil 72.55%. Berdasarkan pada gambar 4.14, performa metode yang diusulkan pada penelitian ini mendapatkan hasil yang lebih baik dibandingkan dengan performa dari metode pohon keputusan dan *naïve bayes*.

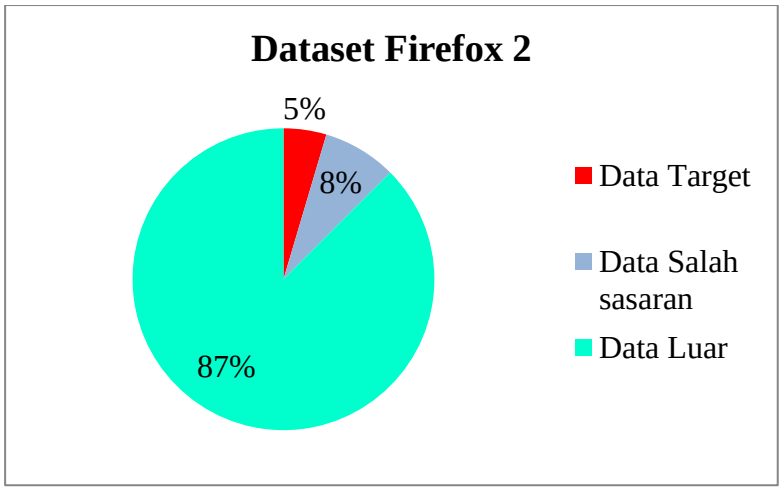
4.4 Analisis Penyaringan Dataset

Penyaringan dataset pada penelitian ini bertujuan untuk meningkatkan hasil akurasi yang didapatkan ketika melakukan perhitungan klasifikasi. Pada sub bab ini akan dilakukan analisis yang terjadi pada pengujian sebelum penyaringan dan sesudah penyaringan. Penyaringan yang dilakukan pada dataset Firefox 1 sebanyak 2064 data dari jumlah keseluruhan sebanyak 16.495 data. Penyaringan yang dilakukan pada dataset Firefox 2 sebanyak 3430 data dari jumlah keseluruhan sebanyak 27.438 data. Penyaringan yang dilakukan pada dataset Eclipse 1 sebanyak 2187 data dari jumlah keseluruhan sebanyak 17.499 data. Penyaringan yang dilakukan pada dataset Eclipse 1 sebanyak 2245 data dari jumlah keseluruhan sebanyak 17.961 data. Penyaringan terhadap data pada setiap dataset adalah 12.5% dengan sasaran data bias yang terletak pada bagian Q1 dataset.



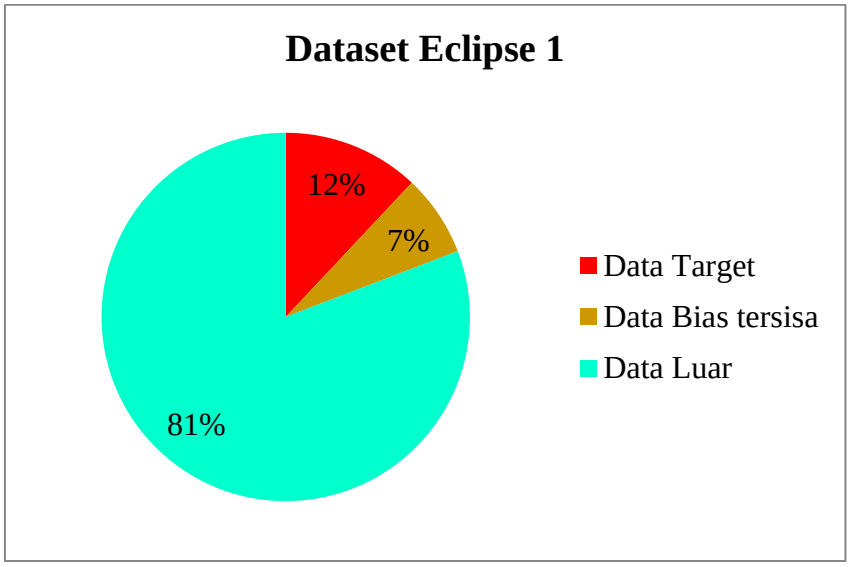
Gambar 4.15 Grafik persentasi data proses penyaringan pada dataset Firefox 1

Pada dataset Firefox 1, proses penyaringan seperti pada gambar 4.15 menampilkan total persentasi dari data. Pada dataset Firefox 1 terdapat 88% data yang bukan target penyaringan atau data Luar. Pada gambar 4.15 menampilkan dari 12.5% yang di saring, hanya 1.31% tepat sasaran pada target penyaringan yang sesungguhnya harus di saring. Penyaringan pada dataset Firefox 1 terjadi penyaringan yang salah sasaran sebanyak 11.19%. Data salah sasaran adalah data yang dalam dataset yang seharusnya tidak tersaring. Pada Dataset Firefox 1 terjadi penurunan akurasi sebesar 2.17%.



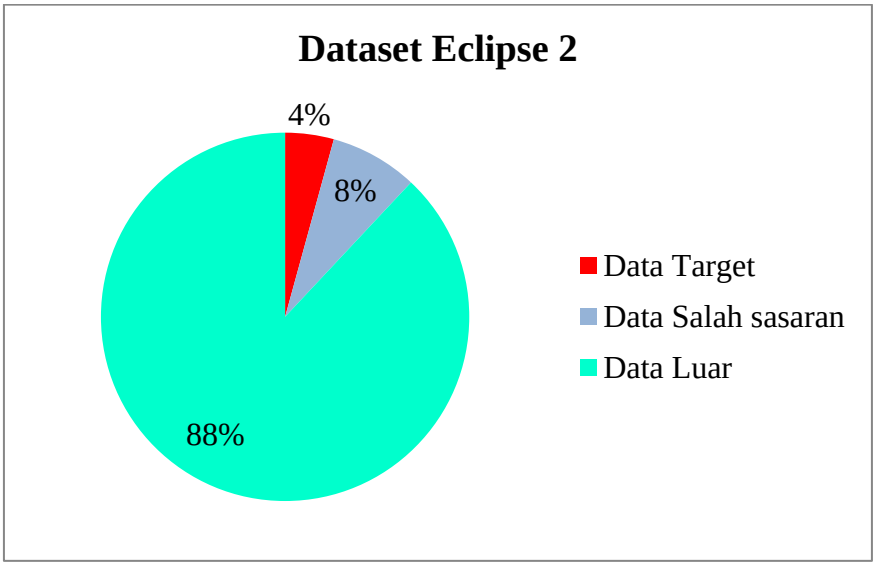
Gambar 4.16 Grafik persentasi data proses penyaringan pada dataset Firefox 2

Pada gambar 4.16 menampilkan proses penyaringan dan total persentasi data pada dataset Firefox 2. Pada dataset Firefox 2 terdapat 87% data yang bukan target penyaringan atau data Luar. Pada gambar 4.15 menampilkan dari 12.5% yang di saring dan hanya 5% atau 4.6% tepat sasaran pada target penyaringan yang sesungguhnya harus di saring. Penyaringan pada dataset Firefox 2 terjadi penyaringan yang salah sasaran sebanyak 8% atau lebih tepatnya 7.9%. Pada dataset Firefox 2 terdapat data bias yang lebih banyak dari pada dataset Firefox 1. Pada Dataset Firefox 2 terjadi peningkatan akurasi sebesar 0.83%.



Gambar 4.17 Grafik persentasi data proses penyaringan pada dataset Eclipse 1

Pada gambar 4.17 menampilkan proses penyaringan dan total persentasi data pada dataset Eclipse 1. Pada dataset Eclipse 1 telah memenuhi sasaran dalam data 12.5% yang dihapus memiliki data bias seluruhnya. Pada gambar 4.17 menampilkan ada 7% data bias yang masih tersisa dalam dataset. Pada dataset Eclipse 1 terjadi peningkatan akurasi sebanyak 6.1%



Gambar 4.18 Grafik persentasi data proses penyaringan pada dataset Eclipse 2

Pada gambar 4.18 menampilkan proses penyaringan dan total persentasi data pada dataset Eclipse 2. Pada dataset Eclipse 2 terdapat 88% data yang bukan

target penyaringan atau data Luar. Pada gambar 4.18 menampilkan dari 12.5% yang di saring dan hanya 4% tepat sasaran pada target penyaringan yang sesungguhnya harus di saring. Penyaringan pada dataset Eclipse 2 terjadi penyaringan yang salah sasaran sebanyak 8% atau lebih tepatnya 8.2%. Pada Dataset Eclipse 2 terjadi penurunan akurasi sebesar 2.05%.

Dari Pengujian terhadap dataset Firefox 1, Firefox 2, Eclipse 1 dan Eclipse 2 menggunakan praproses menghasilkan beberapa hasil. Pada hasil analisis dari penyaringan dataset, Rata-rata penyaringan pada pengujian telah memenuhi target sebanyak 5.6% dari total 12.5% data yang harus disaring. Dalam proses penyaringan terdapat kesalahan sasaran penyaringan sebanyak 6.9%. Hasil akurasi dapat meningkat jika lebih dari 4.5%-12% penyaringan yang tepat sasaran. Hasil akurasi dapat menurun jika mendapatkan dibawah 4% penyaringan yang tepat sasaran dan lebih dari 8% data yang salah sasaran. Penurunan hasil akurasi dapat simpulkan karena banyaknya data yang seharusnya penting termasuk dalam penyaringan data.

4.5 Analisis Hasil Pengujian

Analisa hasil pengujian telah didapatkan dari proses pengujian serta perbandingannya dengan metode yang telah diajukan pada penelitian sebelumnya. Hasil dari analisa adalah sebagai berikut:

1. Metode yang diusulkan yaitu hutan acak ditambah penyaringan dataset memiliki nilai presisi lebih baik dengan rata-rata 0.7125 lalu setelahnya diikuti oleh *naïve bayes* dengan 0.7125. Perbedaan hasil presisi dari semua metode tidak terlalu signifikan. Metode yang diusulkan memiliki nilai sensitivitas yang baik dibandingkan dengan metode lainnya dengan 0.725.
2. Metode hutan acak ditambah penyaringan dataset memiliki nilai akurasi yang baik dibandingkan dengan metode pohon keputusan dan *naïve bayes* pada penggunaan dataset firefox 1, firefox 2, dan eclipse 2.
3. Pada dataset Eclipse 1 hasil akurasi *naïve bayes* lebih unggul daripada metode yang diusulkan. Selisih akurasi pada dataset Eclipse 1 tidak terlalu berbeda secara signifikan dari metode yang diusulkan karena hanya terpaut 1.12%. Performa dari metode yang diusulkan lebih rendah karena data

pada *class* kategori 2 dan kategori 3 memiliki kesalah klasifikasi yang cukup banyak, hal ini karena kedua kategori tersebut memiliki data yang saling berkorelasi. Kategori 2 yang terdeteksi sebagai kategori 3 sebanyak 863 data, sedangakan kategori 3 yang terdeteksi sebagai kategori 2 sebanyak 919. Atribut yang memiliki banyak kesamaan antara kategori 1 dan 2 adalah atribut prioritas.

4. Hasil pada dataset firefox 2 menggunakan data sebanyak 27.439 mendapatkan performa yang menurun dari pada dataset Firefox 1. Penurunan akurasi dapat dikarenakan penyebaran data yang tidak baik sehingga dapat membuat metode klasifikasi terlihat menghasilkan performa yang kurang baik.
5. Metode yang diusulkan yaitu hutan acak ditambah penyaringan dataset memiliki hasil akurasi mencapai rata-rata pada setiap dataset yang digunakan sebesar 72.55%. Akurasi dari keseluruhan dataset yang digunakan menghasilkan akurasi dengan rentang antara 55%-82%. Akurasi ini merupakan nilai tertinggi dari penelitian yang telah dilakukan sebelumnya. Nilai presisi yang didapatkan didapatkan dengan metode yang diusulkan mendapat nilai tertinggi dengan rata-rata 0.7125. Nilai sensitivitas yang didapatkan oleh metode yang diusulkan adalah yang tertinggi dengan rata-rata 0.725.

BAB V

PENUTUP

Bab ini akan membahas kesimpulan dari hasil penelitian dan saran untuk penelitian selanjutnya berdasarkan hasil dan evaluasi pengusian pada bab 4.

5.1 Kesimpulan

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. Pada penelitian ini menggunakan empat dataset yang berasal dari perangkat lunak sumber terbuka Firefox dan Eclipse berjumlah 79.393 data. Dataset Firefox berjumlah 43.933 data dari pengamatan tahun 2004 - 2016. Dataset Eclipse berjumlah 35.460 data dari pengamatan tahun 2010 - 2016. Dari hasil penelitian yang telah dilakukan, maka dapat diambil kesimpulan sebagai berikut:

1. Penggunaan praproses yang berisi langkah penyaringan dataset terhadap data yang termasuk bias akan mengalami peningkatan akurasi. Langkah praproses ini baik terhadap metode hutan acak karena akan mengurangi jumlah data dalam atribut yang saling berkorelasi. Rata-rata penyaringan pada pengujian telah memenuhi target sebanyak 5.6% dari total 12.5% data yang harus disaring. Dalam proses penyaringan terdapat kesalahan sasaran penyaringan sebanyak 6.9%. Hasil akurasi dapat meningkat sebesar 0.83% - 6.1% jika penyaringan data pada dataset yang tepat sasaran lebih dari 4.5% hingga 12%. Hasil akurasi dapat menurun hingga 2.17% jika mendapatkan data penyaringan yang tepat sasaran dibawah 4% dan lebih dari 8% data yang salah sasaran. Penurunan hasil akurasi dapat simpulkan karena banyaknya data yang seharusnya penting termasuk dalam penyaringan data.
2. Metode yang diusulkan yaitu hutan acak ditambah penyaringan dataset memiliki hasil akurasi mencapai rata-rata pada setiap dataset yang digunakan sebesar 72.55%. Akurasi dari keseluruhan dataset yang

digunakan menghasilkan akurasi dengan rentang antara 55%-82%. Akurasi ini merupakan nilai tertinggi dari penelitian yang telah dilakukan sebelumnya. Nilai presisi yang didapatkan didapatkan dengan metode yang diusulkan mendapat nilai tertinggi dengan rata-rata 0.7125. Nilai sensitivitas yang didapatkan oleh metode yang diusulkan adalah yang tertinggi dengan rata-rata 0.725.

3. Metode yang diusulkan dengan menggunakan hutan acak memiliki performa yang lebih baik dibandingkan dengan metode pohon keputusan dan *naïve bayes*. Semakin baik hasil akurasi dapat dikarenakan teknik klasifikasi hutan acak dengan cara *vote*, karena semakin banyak data yang tidak berkorelasi maka hasil pemilihan hasil akan semakin baik.

5.2 Saran

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. Penyaringan dataset diharapkan dapat ditingkatkan persentasi keberhasilan agar mendapatkan peningkatan akurasi yang lebih optimal.

[Halaman ini sengaja dikosongkan]

DAFTAR PUSTAKA

1. Abdelmoez, W., Mohamed Kholief, and Fayrouz M. Elsalmy. 2012. "Bug Fix-Time Prediction Model Using Naïve Bayes Classifier." Pp. 13–15 in *2012 22nd International Conference on Computer Theory and Applications (ICCTA)*. IEEE.
2. Abdelmoez, W., Mohamed Kholief, and Fayrouz M. Elsalmy. 2013. "Improving Bug Fix-Time Prediction Model by Filtering out Outliers." Pp. 359–64 in *2013 The International Conference on Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE)*. IEEE.
3. Alenezi, Mamdouh. 2013. "Bug Reports Prioritization : Which Features and Classifier to Use ?" in *2013 12th International Conference on Machine Learning and Applications*. IEEE.
4. Bhattacharya, Pamela. 2010. "Bug-Fix Time Prediction Models : Can We Do Better ?" (3):1–4.
5. Breiman, L. E. O. 2001. "Random Forests." *Machine Learning* 45(1):5–32.
6. Giger, Emanuel, Martin Pinzger, and Harald C. Gall. 2010. "Predicting the Fix Time of Bugs." Pp. 52–56 in *RSSE '10 Proceedings of the 2nd International Workshop on Recommendation Systems for Software Engineering*. ACM.
7. Lamkanfi, Ahmed and Serge Demeyer. 2012. "Filtering Bug Reports for Fix-Time Analysis." Pp. 379–84 in *2012 16th European Conference on Software Maintenance and Reengineering*. IEEE.
8. Marks, Lionel and Ahmed E. Hassan. 2011. "Studying the Fix-Time for Bugs in Large Open Source Projects Categories and Subject Descriptors." P. No. 11 in *Promise '11 Proceedings of the 7th International Conference on Predictive Models in Software Engineering*. ACM.
9. R. L. Scheaffer, M. Mulekar, J. T. McClave. 2010. *Probability and Statistics for Engineers*. 5th ed. Duxbury Press.
10. Vijayakumar, K. 2014. "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*. IEEE.

11. Zhang, Hongyu, Liang Gong, and Steve Versteeg. 2013. "Predicting Bug-Fixing Time : An Empirical Study of Commercial Software Projects." 1042–51.

BIODATA PENULIS

Penulis, **Nur Fajri Azhar**, lahir di Pasir, 18 mei 1992. Biasa dipanggil dengan nama fajri. Anak pertama dari 3 bersaudara dan dibesarkan di kota Balikpapan, Kalimantan Timur. Penulis menempuh pendidikan formal di SD Negeri 001 Balikpapan (1998-2004), SMP NEGERI 3 Balikpapan (2004-2007), SMA Negeri 6 Balikpapan (2007-2010) Pada tahun 2010-2014 penulis melanjutkan studinya di jurusan Teknik Informatika, Fakultas Teknik Informatika, Universitas Muhammadiyah Malang. Pada tahun 2014-2016, penulis melanjutkan pendidikan Magister S2 di jurusan yang sama, yaitu Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya, Jawa Timur. Di Jurusan Teknik Informatika, penulis mengambil bidang minat Rekayasa Perangkat Lunak.