TESIS

METODE PRAPROSES UNTUK PENINGKATAN AKURASI PREDIKSI WAKTU PERBAIKAN BUG BERDASARKAN LAPORAN BUG

Mochammad Arief Ridwan
NRP. 05111650010051 / 5116201051

DOSEN PEMBIMBING

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

PROGRAM MAGISTER

RUMPUN MATA KULIAH REKAYASA PERANGKAT LUNAK

DEPARTEMEN INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI DAN KOMUNIKASI

INSTITUT TEKNOLOGI SEPULUH NOPEMBER

SURABAYA, 2018

LEMBAR PENGESAHAN TESIS

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

oleh:
MOCHAMMAD ARIEF RIDWAN
NRP.05111650010051 / 5116201051

Dengan judul:
Metode Praproses untuk Peningkatan Akurasi Prediksi Waktu Perbaikan *Bug* Berdasarkan Laporan *Bug*

Tanggal Ujian :
Periode Wisuda : 2018 Genap

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)

Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.
NIP. 197512202001122002
(Penguji 2)

Dr. Radityo Anggoro, S.Kom., M.Sc.
NIP.198410162008121002
(Penguji 3)

Dekan FTIK,

Dr. Agus Zainal Arifin, S. Kom., M. Kom.
NIP. 19720809 199512 1 001

[Halaman ini sengaja dikosongkan]

Metode Praproses untuk Peningkatan Akurasi Prediksi Waktu Perbaikan *Bug* Berdasarkan Laporan *Bug*

Nama Mahasiswa : Mochammad Arief Ridwan
NRP : 0511650010051/5116 201 051
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.

Beberapa penelitian yang telah dilakukan tentang prediksi waktu perbaikan bug menggunakan berbagai algoritma klasifikasi yang sudah ada dengan dataset yang bersifat bebas dan dapat diakses atau diunduh dari situs perangkat lunak. Klasifikasi dari penelitian yang sudah ada menggunakan banyak dataset dengan rentang waktu yang amat bervariasi, hal tersebut dapat menyebabkan turunnya akurasi dari model prediksi yang dibangun.

Dalam penelitian ini penulis berpendapat bahwa dataset awal sebaiknya dilakukan praproses terlebih dahulu dengan metode yang diusulkan untuk dapat meningkatkan akurasi dari prediksi. Model praproses yang diusulkan adalah partisi dataset berdasarkan rentang waktu tiap data.

Hasil dari tiap kelompok data dirata-rata sehingga dapat diperoleh akurasi akhir untuk dibandingkan dengan akurasi tanpa menggunakan praproses dataset. Dengan menggunakan model praproses yang diusulkan, terbukti mampu meningkatkan akurasi dari beberapa pengujian dengan dataset yang digunakan. Peningkatan akurasi tertinggi sebesar 21,24% sedangkan terendah 1,36%.

Kata Kunci: *Bug, Prediksi, Waktu perbaikan, Partisi.*

[Halaman ini sengaja dikosongkan]

Preprocess Method to Improve Accuracy of Predicting Bug Fixing Time Based on Bug Report

Name : Mochammad Arief Ridwan
Student ID : 0511650010051/5116 201 051
Supervisor : Dr. Ir. Siti Rochimah, M.T.

ABSTRACT

Software developers should have a plan in setting up software development costs. Software repairs in the system maintenance phase can be caused by bugs. Bugs are malfunctions that occur in software that does not meet the needs of the software. The software bug can have a fast or long time in the repair depending on the difficulty level. Developers can be assisted by predictive model recommendations and provide time-out considerations for bug fixes.

Some research has been done on the predicted time of bug fixes using various existing classification algorithms with free datasets that can be accessed or downloaded from the software site. The classification of existing research uses multiple datasets with varying time ranges, which can lead to a decrease in the accuracy of the predicted model built.

In this study the authors argue that the initial dataset should be pre-processed first with the proposed method to be able to improve the accuracy of the prediction. The proposed pre-processing model is the dataset partition based on the time range of each data.

The results of each data group are averaged so that final accuracy can be obtained for comparison with accuracy without using dataset preprocess. Using the proposed pre-process model proved to improve the accuracy of some tests with the dataset used. The highest accuracy increase was 21.24% while the lowest was 1.36%.

Keywords*: Bug, Prediction, Fixing time, Partition.*

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Puji syukur atas kehadirat Allah SWT, karena atas limpahan dan karunia-Nya lah penulis dapat menyelesaikan buku thesis penelitian saya ini dengan baik dan lancar. Terima kasih juga tidak lupa penulis ucapkan kepada seluruh pihak yang telah membantu penulis dalam penyusunan buku thesis ini, tanpa bantuan dari seluruh pihak yang telah berkontribusi dalam penyusunan buku thesis ini mungkin buku ini tidak akan selesai.

Pada kesempatan ini, penulis menyampaikan ucapan terima kasih yang sebesar-besarnya kepada:

1. Allah SWT atas limpahan rahmat, karunia, serta ilmu-Nya sehingga penulis dapat menyelesaikan thesis ini dengan baik.
2. Kedua orang tua penulis yang selalu memberi dukungan moril, materil, dan selalu mendoakan penulis untuk bisa menyelesaikan studi.
3. Ibu Dr. Ir. Siti Rochimah, M.T. selaku dosen pembimbing penulis yang telah mempercayai saya menjadi anak bimbing beliau, memberikan bantuan dan motivasi pada penulis sehingga dapat menyelesaikan buku thesis ini.
4. Bapak Daniel Oranova Siahaan, S.Kom. M.Sc. PD.Eng., Ibu Dr.Eng. Chastine Fatichah S.Kom. M.Kom., dan Bapak Radityo Anggoro S.Kom. M.Kom. selaku dosen penguji yang telah memberi saran, masukan, dan koreksi dalam thesis ini.
5. Nur Fajri Azhar selaku penyedia dataset dan yang memberikan inspirasi terhadap judul thesis ini.
6. Kepada Dewi Maya Fitriana yang selalu menyemangati dikala penulis sedang dalam kondisi buntu dalam mengerjakan thesis ini.

7. Teman-teman seperjuangan yang gagal lulus 3 semester, Risyanggi Azmi Faizin dan Achmad Saiful semoga bisa lulus bareng semester 4 ini sehingga bisa bebas dari SPP semester depan.
8. “Geng Sosialita” yang telah memberi warna pada perkuliahan ini.
9. Teman-teman PSM ITS yang masih memberi kesempatan kepada penulis untuk bergabung dalam kompetisi luar negeri walaupun sedang menempuh studi S2.
10. Alumni geng DTK yang sempat mendukung proses penelitian thesis.
11. Geng ITS Insane Workout yang sebenarnya tidak ada hubungannya dalam thesis ini, tetapi telah membantu penulis agar dapat hidup lebih sehat.
12. Rekan-rekan seangkatan 2016 S2 Informatika ITS yang telah menemani penulis selama 2 tahun dalam penempuh studi magister.
13. Serta pihak-pihak lain yang tidak dapat disebutkan satu persatu.

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

Surabaya, 25 April 2018

Mochammad Arief Ridwan

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

BAB 1
PENDAHULUAN

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

1.1 Latar Belakang

Setiap proses pengembangan perangkat lunak pasti memiliki mekanisme dan tahapan dalam pemeliharaan. Idealnya pemeliharaan dilakukan saat perangkat lunak sudah memasuki tahap rilis, walaupun sebenarnya tidak selalu seperti demikian. Tahap pemeliharaan ini bertujuan untuk menghilangkan bug pada perangkat lunak. Bug adalah kerusakan yang terjadi pada perangkat lunak yang tidak sesuai dengan kebutuhan perangkat lunak. Bug yang muncul pada perangkat lunak dapat diperbaiki oleh pengembang dengan mempertimbangkan biaya dan waktu pengerjaannya. Semakin lama waktu pengerjaan bug maka akan berpengaruh pada semakin tingginya biaya perbaikan bug pada perangkat lunak. Waktu pengerjaan bug sebenarnya dapat diprediksi dengan menggunakan model prediksi yang sudah ada untuk membantu estimasi dari waktu yang dibutuhkan untuk perbaikan bug.

Metode yang digunakan dalam prediksi waktu perbaikan bug adalah klasifikasi. Klasifikasi yang digunakan ada berbagai macam algoritma, penelitian oleh Giger menggunakan *decision tree* untuk melakukan prediksi dengan dataset Eclipse JDT, Eclipse Platform, Mozilla Core, Mozilla Firefox, Gnome GStreamer, dan Gnome Evolution. Menurut Giger akurasi dari prediksi yang diperoleh dapat ditingkatkan lagi dengan metode Naïve Bayes dan K-NN (Giger Emanuel, 2010).

Penelitian lain oleh Abdelmoez menggunakan Naïve Bayes dan K-NN untuk melakukan prediksi. Penelitian Abdelmoez menggunakan dataset Eclipse JDT, Mozilla Firefox, Gnome GStreamer, dan Gnome Evolution. Penelitian Abdelmoez menggunakan 17 atribut untuk tiap dataset, sama halnya dengan Giger

namun pemberian *class* untuk tiap data Abdelmoez menggunakan kuartil untuk pemberian *class* pada data sehingga dapat meningkatkan akurasi dari prediksi oleh Giger (Abdelmoez W, 2012).

Selain penelitian dengan mengubah metode klasifikasi, penelitian tentang atribut yang digunakan dalam klasifikasi telah dilakukan. Alenezi melakukan penelitian tentang penggunaan atribut pada laporan *bug*. Laporan *bug* diteliti dengan membandingkan penggunaan atribut deskriptif dan non-deskriptif. Atribut dekriptif adalah atribut yang bersifat tidak terstruktur berupa penjelasan tekstual dari *bug* yang ditemukan, sedangkan non-deskriptif adalah atribut terstruktur dari data bug (Alenezi, 2013). Alenezi menggunakan 3 metode klasifikasi untuk melihat pengaruh atribut yang digunakan dalam klasifikasi, yaitu Naïve Bayes, *Decision Tree,* dan *Random Forest*. Hasil dari penelitian Alenezi adalah atribut non deskriptif memberikan akurasi yang lebih tinggi dibandingkan atribut deskriptif.

Vijayakumar dan Bhuwaneswari (2014) melakukan penelitian yang membuat kategori berapa banyak usaha yang dibutuhkan untuk mengerjakan sebuah *bug*. Penelitian tersebut menghasilkan eksperimen beberapa model prediksi yang masih membutuhkan peningkatan akurasi.

Penelitian lain oleh Nur menggunakan *random forest* untuk melakukan prediksi waktu perbaikan bug dengan dataset Firefox dan Eclipse. Pemberian *class* pada penelitian Nur juga menggunakan pembagian kuartil untuk tiap dataset yang digunakan (Nur Fajri Azhar, 2016). Salah satu kekurangan yang mungkin dapat diperbaiki adalah tiap penelitian prediksi waktu perbaikan *bug* yang dilakukan memiliki dataset yang sangat banyak dengan rentang waktu yang sangat jauh. Hal ini yang mendasari penulis untuk membuat penelitian tentang praproses prediksi waktu perbaikan *bug*.

Penelitian ini mengusulkan metode praproses untuk mengelompokkan laporan *bug* berdasarkan waktu perbaikan. Dataset awal yang memiliki banyak data dikelompokkan berdasarkan waktu perbaikan yang memiliki kemiripan. Dengan memberikan suatu *threshold* pada batas kelompok dataset, kemudian akan dilakukan klasifikasi pada tiap pecahan dataset tersebut. Hasil akurasi yang didapat

dari masing-masing pecahan dataset tersebut akan dirata-rata sebagai akurasi akhir dari penelitian untuk tiap dataset asli. Dengan menggunakan beberapa metode klasifikasi untuk prediksi, akan dilihat pula pengaruhnya pada tiap metode klasifikasi yang digunakan dalam prediksi. Jika memang dapat meningkatkan akurasi pada tiap metode, maka diharapkan model prediksi yang diusulkan dapat digunakan sebagai salah satu model yang lebih baik untuk prediksi waktu perbaikan *bug*.

1.2 Perumusan Masalah

Rumusan masalah yang diangkat dalam penelitian ini adalah sebagai berikut :

1. Apakah penggolongan data bug berpengaruh terhadap klasifikasi laporan perbaikan *bug* ?
2. Bagaimana cara menentukan masuk golongan manakah laporan *bug* untuk tiap laporan ?
3. Apakah metode yang diusulkan akan selalu meningkatkan akurasi dari prediksi ?

1.3 Tujuan Penelitian

Penelitian ini bertujuan untuk menilai pengaruh praproses pengelompokan dataset laporan *bug* pada prediksi waktu perbaikan *bug.* Pendekatan yang digunakan dalam penelitian ini diharapkan dapat menghasilkan prediksi yang lebih akurat dari metode tanpa praproses.

1.4 Manfaat Penelitian

Mafaat penelitian ini diharapkan dapat digunakan oleh pengembang perangkat lunak untuk melakukan prediksi waktu pengerjaan *bug* pada perangkat lunak, sehingga pengembang dapat mengoptimalkan 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 praproses dataset sebelum dilakukan klasifikasi.
2. Melakukan penggolongan dataset *bug* untuk mengetahui apakah berpengaruh terhadap akurasi prediksi waktu perbaikan *bug*.

1.6 Batasan Masalah

Batasan masalah pada penelitian ini adalah :

1. Penelitian ini mengusulkan sebuah metode praproses dalam prediksi waktu perbaikan *bug*, tidak fokus pada bagaimana perbaikan *bug* dapat dilakukan.
2. Dataset yang digunakan bersifat *open source*, dan sudah tersedia bebas.

BAB 2
KAJIAN PUSTAKA

Pada bab ini akan dijelaskan tentang pustaka yang terkait dengan landasan penelitian. Pustaka yang terkait adalah *bug* perangkat lunak, dataset, dan penelitian sebelumnya yang terkait.

2.1 Dasar Teori

2.1.1 *Bug* Perangkat Lunak

Bug adalah salah satu masalah dalam pengembangan perangkat lunak yang sering terjadi. Penyebab dari munculnya *bug* dapat bervariasi mulai dari kesalahan pada tahap konstruksi perangkat lunak maupun kesalahan dalam perancangan. Selain dari kesalahan manusia, *bug* juga dapat muncul akibat perubahan ruang lingkup perangkat lunak. Semakin banyak jumlah *bug* pada perangkat lunak akan berakibat mengganggu kinerja dari perangkat lunak tersebut. Perangkat lunak yang mengalami banyak *bug* akan digolongkan sebagai *buggy* atau cacat.

2.1.2 *Dataset*

Penelitian ini menggunakan dataset yang berasal dari laporan pengguna perangkat lunak pada Bugzilla. Bugzilla memiliki sekumpulan data laporan *bug* yang tersimpan dengan baik dan data dapat diakses untuk berbagai kebutuhan terkait. Penelitian ini akan menggunakan data-data laporan dari pengguna untuk dijadikan dataset, dan laporan akan dibandingkan dari dua perangkat lunak *open source*. Perangkat lunak yang digunakan adalah Eclipse dan Firefox. Data laporan yang akan digunakan pada penelitian berasal mulai tahun 2004 hingga 2016.

Dataset yang digunakan pada penelitian ini memiliki beberapa atribut. Hasil dari Bugzilla memungkinkan kita mendapatkan banyak atribut, namun dalam penelitian ini hanya akan digunakan 9 atribut. Hal ini karena pada penelitian Alenezi (2013) telah dilakukan peneletian mengenai pengaruh atribut tekstual yang

dapat menurunkan akurasi dari prediksi waktu perbaikan *bug*. Atribut yang digunakan dalam penelitian dapat dilihat pada Tabel 2.1.

Tabel 2.1 Atribut yang digunakan dalam penelitian

Atribut	**Deskripsi**
Perangkat	Perangkat *hardware*
Sistem Operasi	Sistem operasi yang digunakan
Komponen	Komponen yang terkena *bug*
Ditugaskan	Pengembang yang ditugaskan untuk memperbaiki *bug*
Pelapor	Nama pelapor yang menemukan *bug*
Produk	Komponen perangkat lunak yang digunakan
Kerasnya	Kondisi dari *bug*
Prioritas	Prioritas dari *bug*
Resolusi	Resolusi terakhir pada laporan *bug*
Waktu Perbaikan	Waktu yang dibutuhkan untuk memperbaiki *bug*

Atribut waktu perbaikan merupakan waktu perbaikan dari *bug* dalam satuan jam dan nantinya akan dijadikan *class* untuk prediksi. Penggunaan atribut tersebut tidak serta merta menggunakan atribut bertipe *numerical* menjadi sebuah *class* namun mengolahnya terlebih dahulu sebagai atribut *categorical*.

2.1.3 Metode Klasifikasi

Metode klasifikasi digunakan untuk prediksi *class* dari tiap data dalam dataset. Beberapa metode yang digunakan dalam penelitian adalah Naive Bayes, *decision tree*, K-NN, dan *random forest*. Tiap metode klasifikasi yang digunakan akan membuktikan bagaimana pengaruh praproses yang diusulkan terhadap akurasi klasifikasi.

2.1.4 *K-Cross Fold Validation*

Cross fold validation merupakan salah satu teknik untuk menilai / mengevaluasi keakuratan sebuah model prediksi yang dibangun berdasarkan dataset tertentu. Pembuatan model biasanya bertujuan untuk melakukan prediksi maupun klasifikasi terhadap suatu data baru yang boleh jadi belum pernah muncul di dalam dataset. Data yang digunakan dalam proses pembangunan model prediksi disebut dengan data latih, sedangkan data yang digunakan untuk melakukan validasi model disebut dengan data uji.

Metode evaluasi *cross fold validation* adalah dengan menentukan nilai K sebagai iterasi untuk menggunakan sebagian dari dataset sebagai data uji. Pengujian akan dilakukan sebanyak K yang akan ditentukan dengan pengambilan data uji secara acak. Data yang digunakan sebagai data uji tidak diikutsertakan dalam pembangunan model prediksi.

2.2 Studi Literatur

2.2.1 *Naive Bayes*

Metode Naive Bayes adalah salah satu metode klasifikasi yang tergolong mudah untuk implementasi dan mudah dipahami. Metode ini membandingkan peluang suatu data baru untuk tiap *class* yang tersedia berdasarkan atribut dari dataset. *Class* dengan peluang terbesar akan dijadikan *class* prediksi dari data uji tersebut (Kamber, 2006). Secara umum metode Naive Bayes memaksimalkan fungsi posterior, fungsi posterior dapat dilihat pada persamaan 2.1.

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)} \tag{2.1}$$

Nilai P(X) pada perhitungan fungsi posterior adalah sama untuk setiap *class*, sehingga pada fungsi tersebut yang perlu dimaksimalkan adalah nilai $P(X|C_i)P(C_i)$. Nilai $P(C_i)$ adalah peluang *class* ke i muncul pada data latih, sedangkan $P(X|C_i)$ adalah peluang data uji dengan atribut yang ditentukan muncul

pada data latih dengan *class* ke i. Nilai maksimal dari $P(C_i|X)$ akan menjadi *class* prediksi dari data uji.

Sebagai contoh implementasi penerapan Naive Bayes pada klasifikasi data dapat dilihat pada Tabel 2.2. Pada Tabel 2.2 terdapat dataset pelanggan dengan atribut *age, income*, *student*, dan *credit* dengan *class buy*.

Tabel 2.2 Contoh dataset untuk naive Bayes

No.	Age	Income	Student	Credit	Buy
1	Youth	High	No	Fair	No
2	Youth	High	No	Excellent	No
3	Middle-aged	High	No	Fair	Yes
4	Senior	Medium	No	Fair	Yes
5	Senior	Low	Yes	Fair	Yes
6	Senior	Low	Yes	Excellent	No
7	Middle-aged	Low	Yes	Excellent	Yes
8	Youth	Medium	No	Fair	No
9	Youth	Low	Yes	Fair	Yes
10	Senior	Medium	Yes	Fair	Yes
11	Youth	Medium	Yes	Excellent	Yes
12	Middle-aged	Medium	No	Excellent	Yes
13	Middle-aged	High	Yes	Fair	Yes
14	Senior	Medium	No	Excellent	No

Dari dataset pada Tabel 2.2 kita mencoba untuk melakukan klasifikasi untuk data uji X = (*age=youth, income=medium, student=yes, credit=fair*). Langkah selanjutnya adalah memaksimalkan fungsi $P(X|C_i)P(C_i)$ untuk i = 1, 2. Nilai $P(C_i)$ untuk masing-masing i adalah $\frac{9}{14}$ untuk P(*buy=yes*) dan $\frac{5}{14}$ untuk P(*buy=no*). Langkah selanjutnya adalah menghitung $P(X|C_i)$ untuk i = 1, 2 peluang kondisi yang ditentukan dari data uji dapat dihitung sebagai berikut

$$P(age = youth|buy = yes) = \frac{2}{9}$$

$$P(age = youth|buy = no) = \frac{3}{5}$$

$$P(income = medium|buy = yes) = \frac{4}{9}$$

$$P(income = medium|buy = no) = \frac{2}{5}$$

$$P(student = yes|buy = yes) = \frac{6}{9}$$

$$P(student = yes|buy = no) = \frac{1}{5}$$

$$P(credit = fair|buy = yes) = \frac{6}{9}$$

$$P(credit = fair|buy = no) = \frac{2}{5}$$

Setelah didapatkan hasil untuk probabilitas yang dibutuhkan maka langkah selanjutnya adalah menghitung nilai P(X|*buy=yes*) dan P(X|*buy=no*) dengan probabilitas yang didapatkan. Hasil probabilitas untuk P(X|*buy=yes*) = $\frac{2}{9}\frac{4}{9}\frac{6}{9}\frac{6}{9}$ = 0.044 sedangkan untuk P(X|*buy=no*) = $\frac{3}{5}\frac{2}{5}\frac{1}{5}\frac{2}{5}$ = 0.019. Langkah selanjutnya adalah mencari nilai maksimal antara P(X|*buy=yes*)P(*buy=yes*) dan P(X|*buy=no*)P(*buy=no*) yang akan dijadikan *class* prediksi.

P(X|*buy=yes*)P(*buy=yes*) = 0.028

P(X|*buy=no*)P(*buy=no*) = 0.007

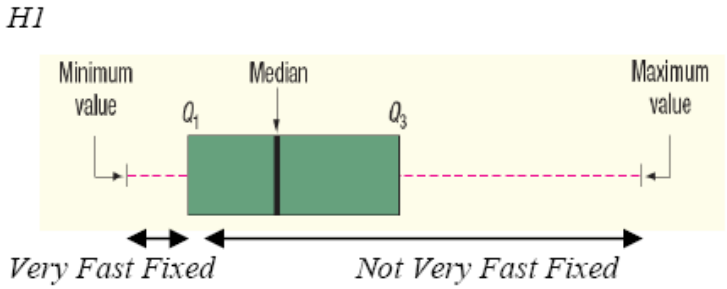
Setelah diketahui probabilitas *class* yang lebih besar maka ditentukan *class* prediksi dari data uji tersebut. Nilai P(X|*buy=yes*)P(*buy=yes*) lebih besar dibandingkan P(X|*buy=no*)P(*buy=no*) sehingga prediksi *class* untuk data X adalah *buy = yes*.

Pada penelitian Abdelmoez (2012) metode klasifikasi yang digunakan untuk prediksi adalah Naive Bayes. Penelitian tersebut menggunakan 6 *class* yaitu cepat, lambat, sangat cepat, tidak sangat cepat, sangat lambat, dan tidak sangat lambat. Pembagian pertama dari Abdelmoez menggunakan median untuk membagi dataset menjadi cepat dan lambat, pembagian *class* ini dapat ditentukan dengan persamaan 2.2 dan 2.3.

$$\text{Cepat} = \text{waktu perbaikan} < \text{median (Q2)} \tag{2.2}$$

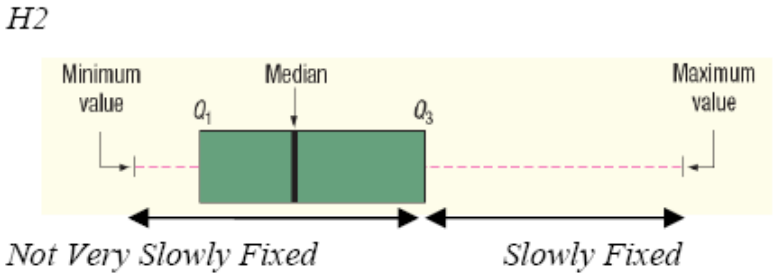
$$\text{Lambat} = \text{waktu perbaikan} > \text{median (Q2)} \tag{2.3}$$

Pembagian *class* selanjutnya menjadi sangat cepat dan tidak sangat cepat. Pembagian ini menggunakan aturan untuk waktu perbaikan < Q1 maka data tersebut dikategorikan sebagai sangat cepat, selain itu dikategorikan dalam tidak sangat cepat. Visualisasi pembagian *class* sangat cepat dan tidak sangat cepat dapat dilihat pada Gambar 2.1.



Gambar 2.1 Visualisasi pembagian kelas *bug* 1 Abdelmoez (2012)

Pembagian *class* selanjutnya menjadi sangat lambat dan tidak sangat lambat. Pembagian ini menggunakan aturan untuk waktu perbaikan > Q3 maka data tersebut dikategorikan sebagai sangat lambat, selain itu dikategorikan dalam tidak sangat lambat. Visualisasi pembagian *class* sangat lambat dan tidak sangat lambat dapat dilihat pada Gambar 2.2.



Gambar 2.2 Visualsisasi pembagian kelas *bug* 2 Abdelmoez (2012)

2.2.2 *Decision Tree*

Metode *decision tree* merepresentasi fungsi dengan masukan berupa vektor dari atribut yang memiliki nilai tertentu dan menghasilkan sebuah hasil tunggal berupa *class* (Stuart Russel, 2010). Metode *decision tree* merupakan salah satu metode yang cukup mudah untuk dipahami oleh manusia karena cukup representatif. Berdasarkan atribut yang ada dalam dataset, *decision tree* akan membangun sebuah *tree* untuk menentukan tergolong *class* mana sebuah data yang diujikan.

Langkah awal *decision tree* adalah dengan memilih atribut yang akan digunakan sebagai *root node* dari *decision tree*. Penentuan *root node* ini tentu menggunakan sebuah aturan yaitu perhitungan nilai *entropy*. Nilai *entropy* merupakan nilai yang digunakan untuk menentukan homogenitas suatu kumpulan data. Jika kumpulan data bersifat homogen total maka memiliki nilai *entropy* sebesar 0 dan jika kumpulan data terbagi sama rata memiliki nilai entropy sebesar 1. Untuk menentukan nilai *entropy* dari kumpulan data dapat menggunakan persamaan 2.4.

$$E(S) = \sum_{i=1}^{c} -p_i log p_i \quad (2.4)$$

Sebagai contoh akan dijelaskan dengan dataset latih untuk prediksi bermain golf atau tidak. Dataset tersebut dapat dilihat pada Tabel 2.3. Data latih tersebut memiliki 4 atribut bersifat *categorical* dan sebuah *class* prediksi dengan

nilai *yes* atau *no*. Dari data latih yang diberikan akan digunakan untuk memprediksi apakah kita bisa bermain golf atau tidak berdasarkan atribut yang ada.

Tabel 2.3 Contoh dataset untuk decision tree

Day	Outlook	Temp	Humidity	Wind	Play
1	Rainy	Hot	High	False	No
2	Rainy	Hot	High	True	No
3	Overcast	Hot	High	False	Yes
4	Sunny	Mild	High	False	Yes
5	Sunny	Cool	Normal	False	Yes
6	Sunny	Cool	Normal	True	No
7	Overcast	Cool	Normal	True	Yes
8	Rainy	Mild	High	False	No
9	Rainy	Cool	Normal	False	Yes
10	Sunny	Mild	Normal	False	Yes
11	Rainy	Mild	Normal	True	Yes
12	Overcast	Mild	High	True	Yes
13	Overcast	Hot	Normal	False	Yes
14	Sunny	Mild	High	True	No

Untuk menentukan *root node* pada data latih tersebut, perlu dilakukan perhitungan *entropy* untuk semua atribut yang ada. Selain dengan menghitung nilai *entropy* tiap atribut, juga diperlukan nilai *gain* untuk menentukan atribut mana yang akan dijadikan *root node*. Persamaan untuk menghitung nilai *entropy* pada tiap atribut dapat dilihat pada persamaan 2.5.

$$E(T, X) = \sum_{c \in X} P(c)E(c) \tag{2.5}$$

Nilai *entropy* dari data latih tersebut dapat dihitung dengan persamaan 2.4 sehingga menghasilkan nilai sebagai berikut :

$$Entropy(Play) = Entropy(5,9)$$
$$= Entropy(0.36,0.64)$$
$$= -(0.36 \log 0.36) - (0.64 \log 0.64)$$
$$Entropy(Play) = 0.94$$

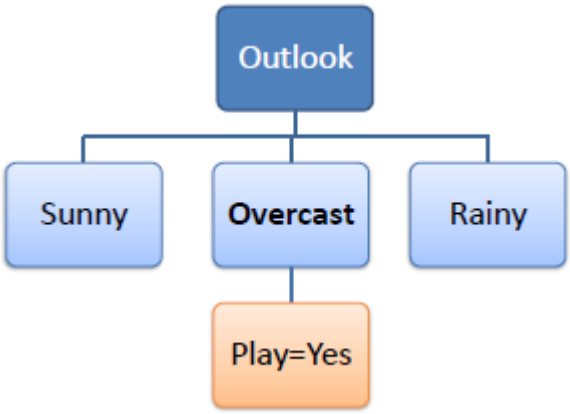
Nilai *entropy* pada *class* yang didapatkan akan digunakan untuk menghitung nilai *entropy* pada masing-masing atribut pada data latih dengan persamaan 2.5. Sebagai contoh akan dilakukan perhitungan nilai *entropy* pada atribut *outlook* dengan perhitungan sebagai berikut :

$$Entropy(Play, Outlook) = P(Sunny) * E(3,2) + P(Overcast) * E(4,0) + P(Rainy) * E(2,3)$$
$$Entropy(Play, Outlook) = (\frac{5}{14}) * 0.971 + (\frac{4}{14}) * 0 + (\frac{5}{14}) * 0.971$$
$$Entropy(Play, Outlook) = 0.693$$

Setelah didapatkan nilai entropy pada tiap atribut, selanjutnya akan dicari nilai *gain* untuk tiap atribut dengan persamaan 2.6. Sebagai contoh akan dilakukan perhitungan nilai *gain* pada atribut *outlook* sebagai berikut :

$$Gain(Play, Oulook) = E(Play) - E(Play, Outlook)$$
$$= 0.940 - 0.693$$
$$= 0.247$$

Setelah didapatkan hasil *gain* untuk tiap atribut, maka *root node* yang dipilih adalah atribut dengan nilai *gain* yang paling tinggi. Hasil dari *gain* untuk atribut *temp* = 0.029, *humidity* = 0.152, dan *windy* = 0.048 sehingga dapat diambil atribut *outlook* yang akan dijadikan *root node* dengan nilai *gain* tertinggi sebesar 0.247. Dengan didapatkannya sebuah atribut sebagai *root node* maka susunan *tree* awal yang dihasilkan dapat dilihat pada Gambar 2.3.



Gambar 2.3 Hasil tree awal decision tree

Pada Gambar 2.3 merupakan hasil *tree* awal pada metode *decision tree* atribut *outlook* merupakan *root node*, sedangkan *sunny, overcast,* dan *rainy* merupakan cabang atau *branch*. Cabang adalah variasi dari atribut yang terdapat pada level atas dari cabang tersebut. Cabang dengan nilai *entropy* = 0 akan memiliki *leaf node*, pada Gambar 2.3 berupa sebuah hasil *class* = *yes*. Untuk cabang yang belum memiliki *leaf node* maka dilakukan metode yang sama pada penentuan *node* untuk tingkat dibawahnya berdasarkan atribut yang tersisa dengan tetap mempertimbangkan nilai *entropy* dan *gain*.

Giger (2010) membuat kategori laporan *bug* menjadi dua kelompok, yaitu cepat dan lambat. Kategori tersebut dipisah dengan median waktu perbaikan dari dataset yang digunakan. *Decision tree* digunakan untuk dua kasus, pertama hanya menggunakan data awal (*reporter, date, nextRelease, hToLatFix*), dan menggunakan kedua data awal dan *post-submission* (*assignee, platform, OS, priority, severety, status,* dan lainnya). Hasil dari penelitian tersebut adalah bahwa kasus kedua memiliki hasil yang lebih baik dibandingkan dengan kasus pertama.

2.2.3 *K – Nearest Neighbour*

Metode K-NN adalah metode klasifikasi dengan menentukan banyaknya tetangga terdekat untuk suatu data dengan data lainnya. Dengan jumlah tetangga yang ditentukan, akan dicari jarak tiap data dengan seluruh data yang ada. K-tetangga terdekat akan menjadi kelas prediksi dari metode K-NN. Untuk

perhitungan jarak dari data menggunakan *euclidean distance* dengan rumus seperti pada persamaan 2.6.

$$D = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2} \tag{2.6}$$

Metode K-NN pada dasarnya lebih cocok untuk klasifikasi data bertipe numerik, sehingga untuk kasus data non numerik perlu dilakukan sebuah proses *encoding* sehingga dapat diproses. Proses *encoding* yang dilakukan dinamakan *binary encoding* dimana pada dataset akan dibuat sebuah atribut *dummy* yang merepresentasikan nilai awal dataset. Contoh *binary encoding* dapat dilihat pada Tabel 2.4.

Tabel 2.4 Binary encoding pada data trend

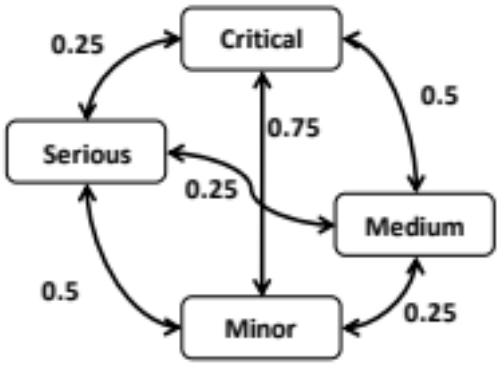
Categorical	**Encoded**		
Trend	Trend_Up	Trend_Down	Trend_Flat
Up	1	0	0
Up	1	0	0
Down	0	1	0
Flat	0	0	1
Down	0	1	0
Up	1	0	0
Flat	0	0	1
Down	0	1	0
Up	1	0	0
Flat	0	0	1
Down	0	1	0

Tabel 2.4 merupakan salah satu contoh implementasi dari *binary encoding* pada dataset dengan atribut *trend* bertipe non numerik dengan nilai *up, down,* atau

flat. Pada dataset tersebut akan ditambahkan atribut *dummy* yang merepresentasikan nilai awal dari atribut *trend*, karena memiliki 3 variasi nilai maka akan ditambahkan 3 atribut *dummy*. Semakin banyak variasi dari nilai atribut maka akan semakin banyak atribut *dummy* yang akan ditambahkan dan tentunya juga mempengaruhi hasil dari klasifikasi.

Usulan penggunaan K-NN dalam prediksi waktu perbaikan *bug* telah dilakukan oleh (Zhang Hongyu, 2013). Penelitian tersebut menggunakan *threshold* waktu dalam menentukan kategori *bug* yang ada yaitu cepat dan lambat. Penelitian tersebut juga beranggapan bahwa dua buah *bug* yang memiliki kemiripan juga memiliki waktu perbaikan yang cenderung hampir sama.

Penelitian Zhang menggunakan dataset CA dengan perhitungan jarak antar data menggunakan *euclidean distance* yang didefinisikan ulang oleh Zhang. Jarak antar *bug* pada penelitian Zhang memperhitungkan prioritas dari *bug* yang ada yaitu, *critical, serious, medium,* dan *minor.* Definisi dari jarak berdasarkan prioritas yang diusulkan Zhang dapat dilihat pada Gambar 2.4. Sebagai contoh jarak antara *critical* dan *minor* adalah 0.75.



Gambar 2.4 Definisi jarak berdasarkan prioritas

Jarak yang didefinisikan Zhang digunakan untuk perhitungan jarak antar data pada klasifikasi K-NN untuk data yang berbeda prioritas. Perhitungan jarak *euclidean* pada penelitian Zhang dapat dilihat pada persamaan 2.7.

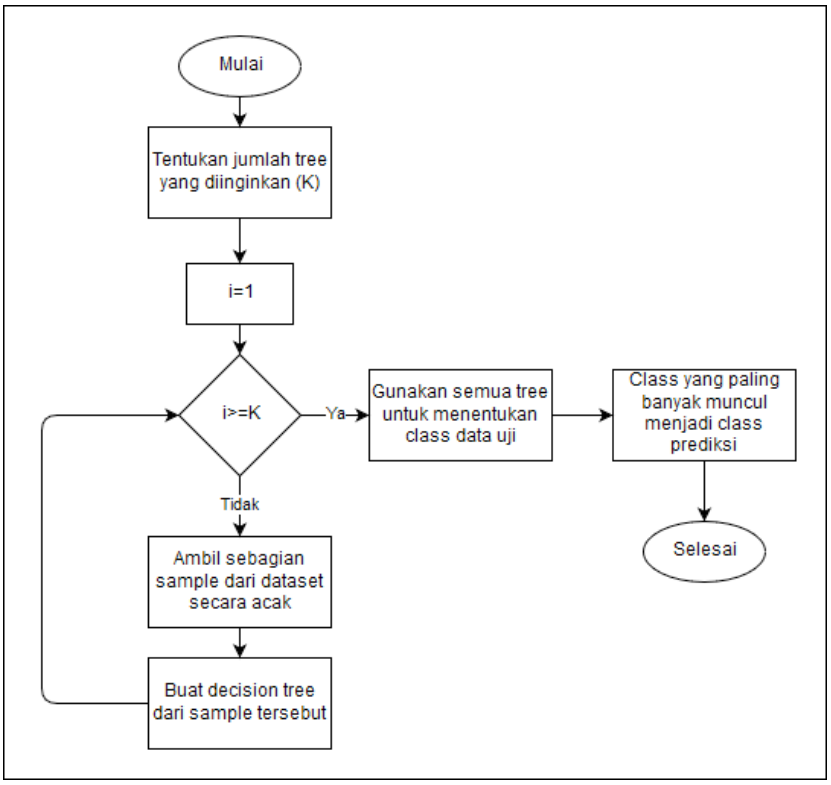
$$d_R(a,\beta) = \sqrt{d_s^2(a,\beta) + \sum_{i=1}^{N_f} d^2(a_i,\beta_i)} \quad (2.7)$$

Dengan α dan β adalah laporan *bug* yang berbeda, N_f adalah jumlah fitur, α_i merupakan fitur ke-i dari *bug* α, $d(\alpha_i,\beta_i)$ merupakan jarak dari fitur ke-i antara *bug* α dan β, $d_s(\alpha,\beta)$ adalah jarak berdasarka prioritas yang dicari menggunakan pemetaan pada Gambar 2.4.

2.2.4 *Random Forest*

Metode *random forest* pertama kali diusulkan oleh Breiman (2001). *Random forest* merupakan algoritma pengembangan dari *decision tree*, dikatakan *random forest* karena menggunakan beberapa *tree* sehingga terbentuk seperti semacam *forest* (hutan) dari beberapa *tree* tersebut. Penggunaan awal dari *random forest* yaitu menentukan banyaknya pohon yang akan digunakan, dari sebuah pohon akan menghasilkan sebuah prediksi untuk 1 data dan hasil prediksi tersebut bisa berbeda-beda untuk tiap pohon. Hasil terbanyak akan dijadikan hasil akhir prediksi untuk 1 data tersebut, hal ini dilakukan untuk seluruh data pada dataset yang ada.

Untuk metode *decision tree* yang digunakan seperti penjelasan pada sub bab sebelumnya mengenai *decision tree*, dimana pertimbangan *entropy* dan *gain* dalam menentukan *node*. Secara umum *random forest* menerapkan algoritma seperti yang dapat dilihat pada Gambar 2.5.



Gambar 2.5 alur algoritma random forest

Penelitian menggunakan *random forest* telah dilakukan oleh Nur (2016) dengan dataset Firefox dan Eclipse. Nur menggunakan metode pembagian kelas berdasarkan kuartil dari dataset. Kuartil 1 merupakan kelas kategori 1, kuartil 2 merupakan kelas kategori 2, kuartil 3 merupakan kelas kategori 3, dan kuartil 4 merupakan kelas kategori 4. Semakin tinggi kategori berarti semakin lama waktu perbaikan dari *bug* tersebut.

2.2.5 *WEKA Tools*

Untuk melakukan berbagai pengujian dataset pada penelitian ini, penulis menggunakan alat bantu berupa perangkat lunak WEKA Tools. WEKA dapat mengimplementasikan banyak algoritma yang digunakan untuk prediksi, yaitu Naive Bayes, *decision tree,* K-NN, dan *random forest*. WEKA juga mampu mengimplementasikan metode pengujian klasifikasi yang dilakukan, karena pada penelitian ini menggunakan K-*cross fold* validation maka pada penggunaan WEKA

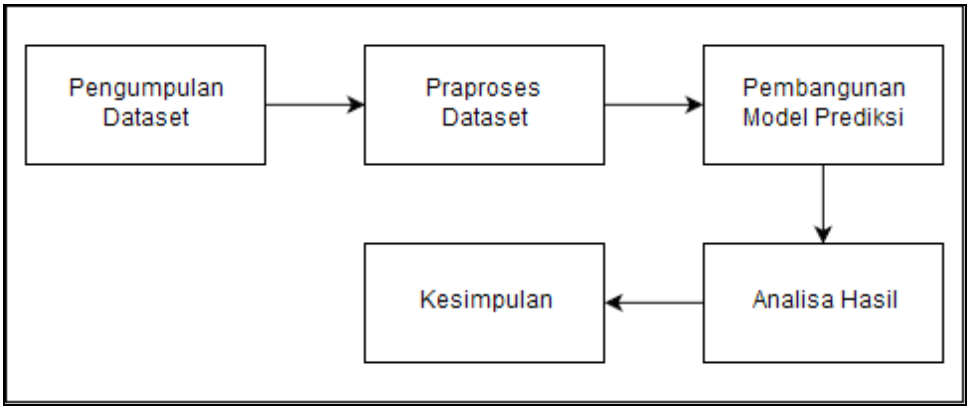
juga akan diatur menggunakan metode tersebut untuk evaluasi hasil prediksi. Dengan menggunakan WEKA, akan mempermudah proses prediksi dan evaluasi dari penelitian yang dilakukan, hasil dari prediksi juga dapat terlihat secara jelas pada tampilan antarmuka yang disediakan oleh WEKA. Versi WEKA yang digunakan adalah WEKA 3.8.1.

[Halaman ini sengaja dikosongkan]

BAB 3
METODE PENELITIAN

3.1 Tahapan Penelitian

Tahapan penelitian dibutuhkan agar tujuan yang diharapkan dari penelitian ini dapat tercapai. Langkah awal dari penelitian ini adalah pengumpulan dataset seperti yang ditunjukkan pada Gambar 3.1. Pengumpulan dataset digunakan untuk mempelajari dataset yang digunakan agar hasil yang didapatkan lebih akurat. Langkah selanjutnya adalah perancangan dan penerapan praproses yang diusulkan. Dalam penelitian ini, praproses yang diusulkan adalah penggunaan partisi dari dataset yang akan diproses. Tahap pembangunan model prediksi merupakan tahapan setelah dilakukan praproses untuk tiap partisi dataset yang telah dibuat. Pembangunan model prediksi yang dilakukan menggunakan berbagai metode klasifikasi yang telah dijelaskan pada Bab 2. Hasil dari tahap pengujian kemudian akan dianalisis untuk mendapatkan kesimpulan dari metode praproses yang telah diusulkan, apakah pengaruhnya dengan prediksi tanpa metode praproses.



Gambar 3.1 Diagram alur penelitian

3.2 Pengumpulan Dataset

Dataset yang digunakan 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 dari tahun 2004 sampai 2016 dengan

jumlah data sebanyak 43.933 data. Dataset Eclipse akan diambil berdasarkan laporan dari tahun 2010 sampai 2016 dengan jumlah 35.460 data.

Tabel 3.1 Dataset yang digunakan

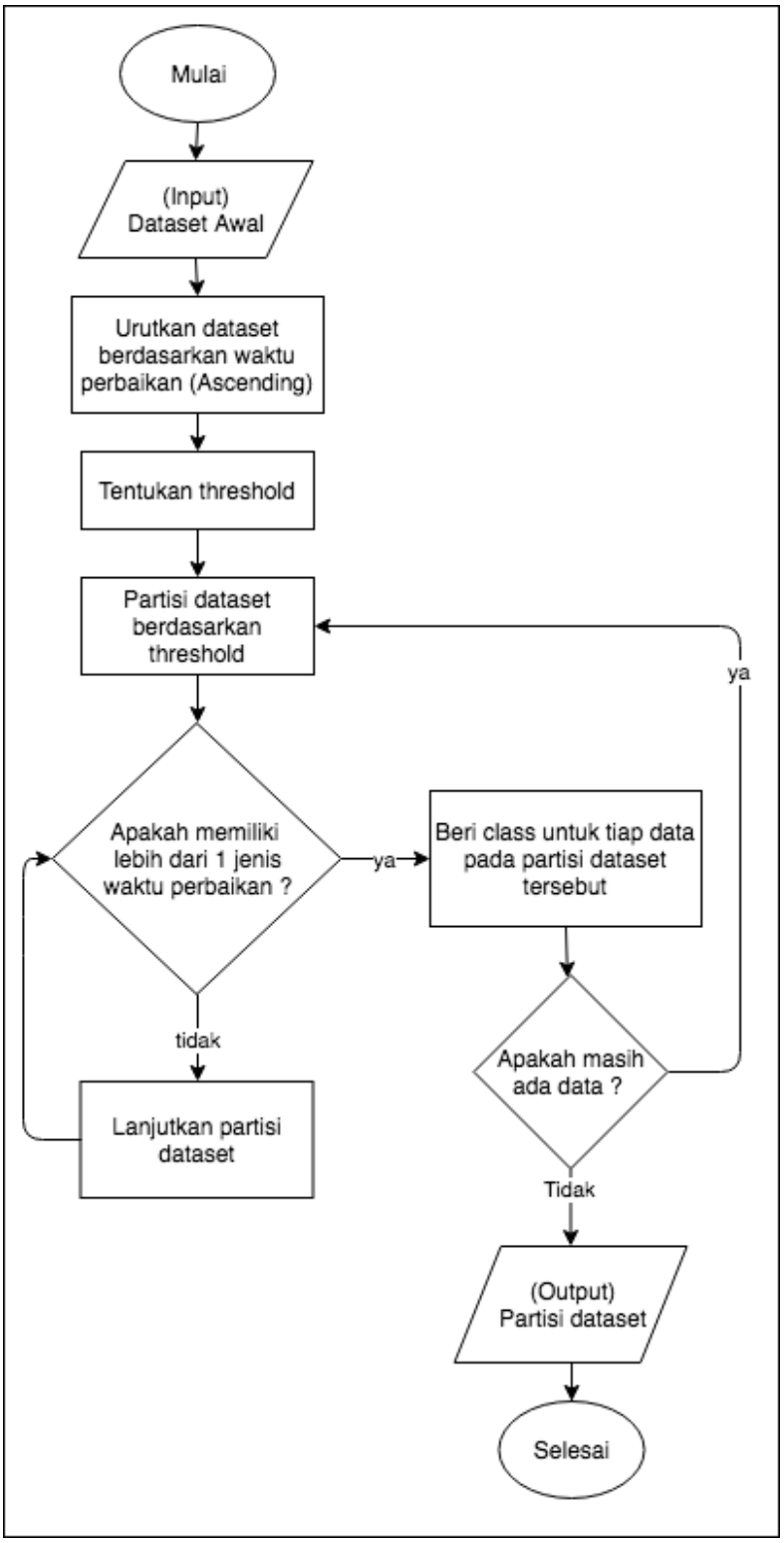
No	Dataset	Jumlah Data	Periode Pengamatan
1	Firefox 1	16.495	2004-2007
2	Firefox 2	27.438	2008-2016
3	Firefox Gabungan	43.933	2004-2016
4	Eclipse 1	17.499	2010-2012
5	Eclipse 2	17.961	2013-2016
6	Eclipse Gabungan	35.460	2010-2016

Penelitian ini menggunakan dataset dari laporan *bug* seperti yang terdapat pada Tabel 3.1. Dataset Firefox dan Eclipse diambil dalam rentang tertentu, kemudian akan dilakukan model prediksi terhadap dataset dalam rentang tersebut. setelah masing-masing rentang diperoleh hasil, akan diujicobakan jika data tersebut digabungkan.

3.3 Praproses Dataset

Prarproses yang diusulkan pada penelitian ini adalah pembagian dataset menjadi beberapa partisi yang kemudian dilakukan pembangunan model prediksi. Akan ada beberapa tahapan dalam praproses yang diusulkan, yaitu tahap pengurutan dataset, penentuan *threshold* partisi data, dan tahap pemberian *class* untuk tiap data dalam dataset.

Alur praproses dan penerapan data secara keseluruhan dapat dilihat pada Gambar 3.2. Alur tersebut dibedakan menjadi beberapa tahap, yaitu : (1) Pengurutan dataset, (2) Penentuan *threshold*, (3) Pemberian *class* untuk tiap data pada dataset.



Gambar 3.2 Alur praproses

Metode praproses yang diusulkan membutuhkan proses untuk mengurutkan dataset berdasarkan waktu perbaikan, namun pada alur yang dijelaskan tidak dibahas secara spesifik metode pengurutan tersebut menggunakan algoritma apa, sehingga untuk implementasi dari praproses yang diusulkan dapat menggunakan algoritma pengurutan apapun. Sedangkan alur dalam bentuk *pseudocode* dapat dilihat pada Gambar 3.3.

```
Preprocess(D,t)
    1.  Sort by waktu_perbaikan(D)
    2.  For ( i from 0 to D)
    3.          j = i
    4.          part = 1
    5.          While (i <= t AND Z >1)
    6.                  Attribute_counter = []
    7.                  D[i].part = part
    8.                  Attribute_counter[D[i].waktu_perbaikan] = 1
    9.                  Z = Attribute_counter.count
    10.                 i++
    11.         part++
    12.         Class(D,j,i,Z)
    13. Return D
```

Gambar 3.3 Pseudocode metode praproses usulan

3.3.1 Pengurutan Dataset

Dataset yang akan diolah untuk pertama akan diurutkan berdasarkan atribut waktu perbaikan. Waktu perbaikan telah dijelaskan pada Bab 2 yaitu waktu yang dibutuhkan untuk memperbaiki *bug* pada perangkat lunak. Atribut waktu perbaikan berupa satuan jam mulai dari yang tercepat hingga terlama akan

diurutkan untuk memulai praproses. Contoh potongan dataset yang sudah diurutkan dapat dilihat pada Gambar 3.4.

Bug ID	Hardware	OS	Assignee	Reporter	Product	Severity	Priority	Component	Resolution	Jam
414922	x86	All	asaf	cbook	Firefox	blocker	P1	RSS Discover	FIXED	0
414759	All	All	nobody	dietrich	Firefox	blocker	P1	Session Rest	DUPLICATE	0
423736	x86	Mac OS X	gavin.sharp	sayrer	Firefox	normal	P1	Theme	FIXED	0
415202	All	All	myk	myk	Firefox	normal	P1	Preferences	FIXED	0
414919	x86	Linux	bugzilla	bugzilla	Firefox	normal	P2	Location Bar	FIXED	0
403470	All	All	alvolkov.bgs	slavomir.katu	NSS	blocker	P1	Libraries	FIXED	24
402669	All	All	nelson	nelson	NSPR	blocker	P1	NSPR	FIXED	24
365597	All	All	peterv	polidobj	Core	critical	P1	XSLT	FIXED	24
406840	x86	Linux	peterv	dbaron	Core	major	P1	DOM	FIXED	24
377397	All	All	asaf	dtownsend	Firefox	major	P1	Page Info Wi	FIXED	24
406852	x86	Linux	jonas	dbaron	Core	normal	P1	Plug-ins	FIXED	24

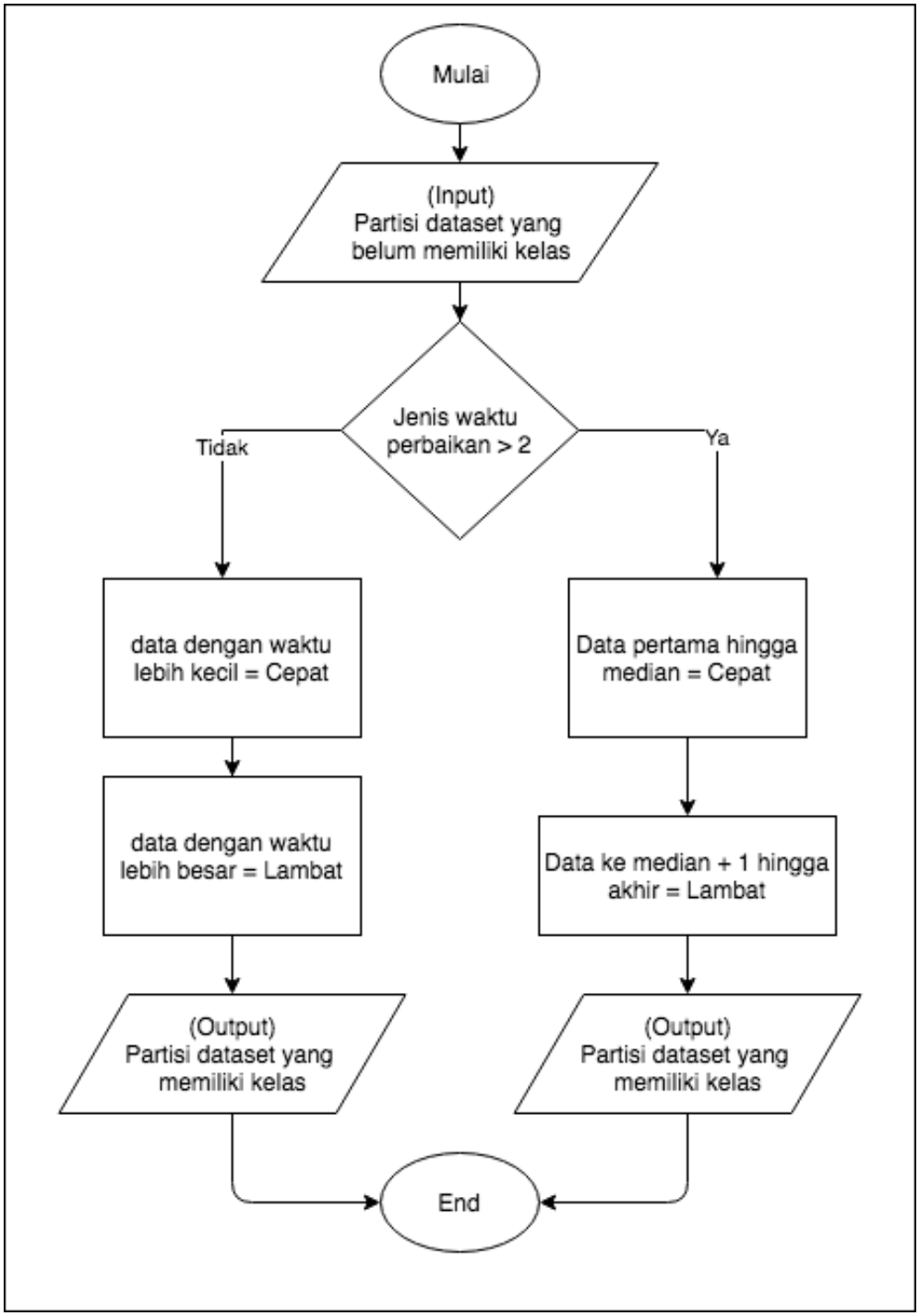
Gambar 3.4 Potongan dataset yang sudah terurut

3.3.2 Penentuan *Threshold*

Tahap penentuan *threshold* adalah menentukan batas partisi dataset yang akan digunakan dalam praproses. Pada penelitian ini akan menggunakan 5 uji coba *threshold* yaitu 1000, 2000, 3000, 4000, dan 5000. Uji coba tersebut akan dilakukan pada satu dataset yang kemudian akan dicari *threshold* mana yang dinilai lebih baik diantara yang lain. Setelah diketahui *threshold* mana yang lebih baik, untuk selanjutnya akan menggunakan *threshold* tersebut untuk dataset yang lain.

3.3.3 Pemberian *Class*

Tahap pemberian *class* merupakan prosedur penentuan class dari partisi dataset yang sudah dipotong berdasarkan *threshold*. Penentuan *class* yang digunakan pada penelitian ini berdasarkan median dari partisi dataset yang diperoleh. Dari partisi dataset yang diperoleh akan dilihat ada berapa perbedaan waktu perbaikan pada partisi tersebut. Jika ada lebih dari 2 jumlah waktu perbaikan maka akan diambil nilai mediannya, untuk data pertama hingga median akan mendapatkan kelas cepat, dan sisanya akan mendapatkan kelas lambat. Sedangkan jika hanya ada 2 waktu perbaikan, waktu yang lebih sedikit akan mendapatkan kelas cepat dan waktu perbaikan yang lebih lama akan mendapatkan kelas lambat. Untuk lebih jelasnya skema pemberian *class* dapat dilihat pada Gambar 3.5.



Gambar 3.5 Alur pemberian class pada dataset

3.4 Pembangunan Model Prediksi

Pembangunan model prediksi yang akan dilakukan menggunakan 4 metode klasifikasi seperti yang dibahas pada Bab 2 berdasarkan penelitian Romi (2016) tentang prediksi kecacatan perangkat lunak. Implementasi dari algoritma klasifikasi akan menggunakan alat bantu yaitu WEKA tools versi 3.8.1. Pada implementasi masing-masing algoritma klasifikasi akan menggunakan koefisien yang ditentukan saat akan menjalankan algoritma klasifikasi. Koefisien pada klasifikasi tersebut akan dijelaskan pada sub-sub bab dibawah ini :

3.4.1 Naive Bayes

Metode Naive Bayes yang digunakan pada penelitian ini menggunakan konfigurasi koefisien standar dari WEKA tools. Koefisien *batchSize* merupakan jumlah data yang akan diproses saat prediksi dilakukan, koefisien *numDecimalPlaces* merupakan jumlah angka dibelakang koma untuk hasil yang didapatkan, sedangkan *displayModelInOldFormat, doNotCheckCapabilities,* dan *debug* merupakan atribut yang digunakan untuk memunculkan informasi tambahan saat prediksi berlangsung. Atribut *useKernelEstimator* dan *useSupervisedDiscretization* digunakan apabila ada atribut bersifat numerik dari dataset yang digunakan sehingga untuk penelitian ini tidak berpengaruh

3.4.2 Decision Tree

Metode *decision tree* pada penelitian ini menggunakan nilai koefisien yang berbeda dari standar WEKA. Hal ini disebabkan untuk mempercepat waktu *running* dari algoritma. Koefisien yang diubah dari standar WEKA adalah *batchSize* dan *minNumObj*. Nilai awal standar WEKA adalah 100 untuk *batchSize* dan 2 untuk *minNumObj,* sedangkan yang digunakan dalam penelitian ini adalah 1 untuk *batchSize* dan 3 untuk *minNumObj*. Nilai *minNumObj* adalah jumlah minimum percabangan yang terjadi pada sebuah *node*. Semakin kecil nilai *minNumObj* akan semakin sedikit percabangan pada sebuah *node* sehingga butuh waktu proses yang semakin lama dibandingkan nilai *minNumObj* yang lebih besar.

3.4.3 Random Forest

Metode *random forest* yang digunakan dalam penelitian ini menggunakan koefisien yang berbeda dari standar WEKA. Hal ini juga merupakan usaha untuk mempercepat waktu *running* dari algoritma. Koefisien yang diubah dari standar WEKA adalah *bagSizePercent* dan *numIterations*. Nilai awal standar WEKA adalah 100 untuk *bagSizePercent* dan 100 untuk *numIterations,* sedangkan yang digunakan dalam penelitian ini adalah 1 untuk *bagSizePercent* dan 10 untuk *numIterations*. Nilai *bagSizePercent* adalah persentase dataset yang diambil untuk membuat sebuah *training tree*. Nilai *numIterations* adalah nilai yang menentukan banyaknya *tree* yang digunakan dalam *random forest*. Pada penelitian ini menggunakan 10 jumlah *tree*.

3.4.4 K-Nearest Neighbour

Metode *K - Nearest Neighbour* yang digunakan dalam penelitian ini menggunakan koefisien K=5, dengan perhitungan jarak menggunakan *euclidean distance*, selebihnya menggunakan standar WEKA dengan *binary encoding* sebagai proses mengubah tipe data non numerik menjadi numerik.

3.5 Analisa Hasil

Analisa hasil dari metode klasifikasi yang digunakan menggunakan *K-Cross Fold Validation* dengan menggunakan nilai K=10. Metode evaluasi hasil akan digunakan pada semua klasifikasi dan semua dataset. Dari evaluasi yang diperoleh akan didapatkan akurasi untuk masing-masing metode. Apabila akurasi dari prediksi dengan penggunaan praproses lebih tinggi dibandingkan dengan tanpa praproses, maka dapat diambil kesimpulan bahwa metode praproses yang diusulkan dapat meningkatkan akurasi dari prediksi waktu perbaikan *bug*.

BAB 4
HASIL DAN PEMBAHASAN

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 pengujian prediksi tanpa menggunakan praproses. Pembahasan ketiga adalah penentuan *threshold* untuk metode praproses yang diusulkan. Pembahasan keempat adalah pengujian menggunakan praproses yang diusulkan. Pembahasan kelima adalah analisa dan hasil pengujian.

4.1 Lingkungan Uji Coba

Subbab ini menjelaskan ruang lingkup uji coba yang digunakan dalam penelitian ini. Lingkungan pengujian meliputi spesifikasi perangkat keras dan spesifikasi perangkat lunak. Perangkat keras yang dimaksud adalah jenis *processor* dan *memory (RAM)* yang digunakan, kapasitas *hard disk*, dan VGA yang digunakan. Untuk perangkat lunak adalah sistem operasi serta *tools* untuk pengujian. Rincian tersebut dapat dilihat pada Tabel 4.1 dibawah ini.

Tabel 4.1 Rincian lingkungan uji coba

Jenis Lingkungan	**Rincian**	
Perangkat Keras	*Processor*	Intel Core i5 2.40 Ghz
	Memory (RAM)	4 GB
	Hard disk	121 GB
	VGA	Intel Iris 1536 MB
Perangkat Lunak	Sistem Operasi	Mac OS High Sierra 10.13
	Tools	WEKA 3.8.1

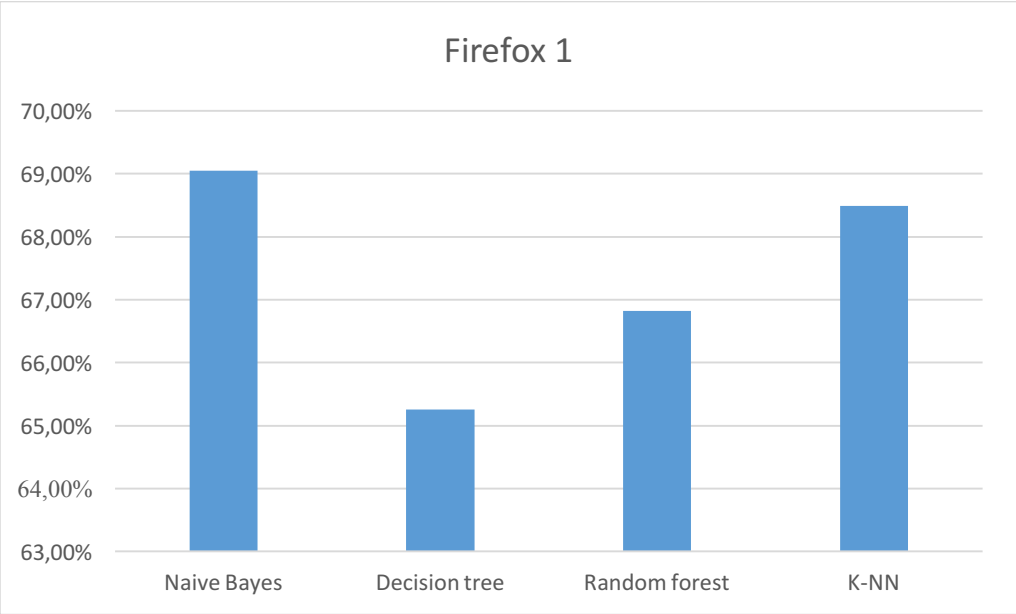
4.2 Pengujian Tanpa Praproses

Pengujian tanpa praproses ini akan mencoba mengambil hasil akurasi dari prediksi untuk semua dataset yang digunakan. Metode prediksi yang digunakan

untuk tiap dataset adalah Naive Bayes, *decision tree*, *random forest*, dan K-NN. Untuk hasil pada tiap dataset dapat dilihat pada sub-sub bab dibawah ini :

4.2.1 Dataset Firefox 1

Pada dataset Firefox 1 akan dilakukan pengujian dengan 4 metode klasifikasi yang sudah ditentukan. Hasil yang didapat dengan melakukan prediksi menggunakan WEKA tools dapat dilihat pada Gambar 4.1.

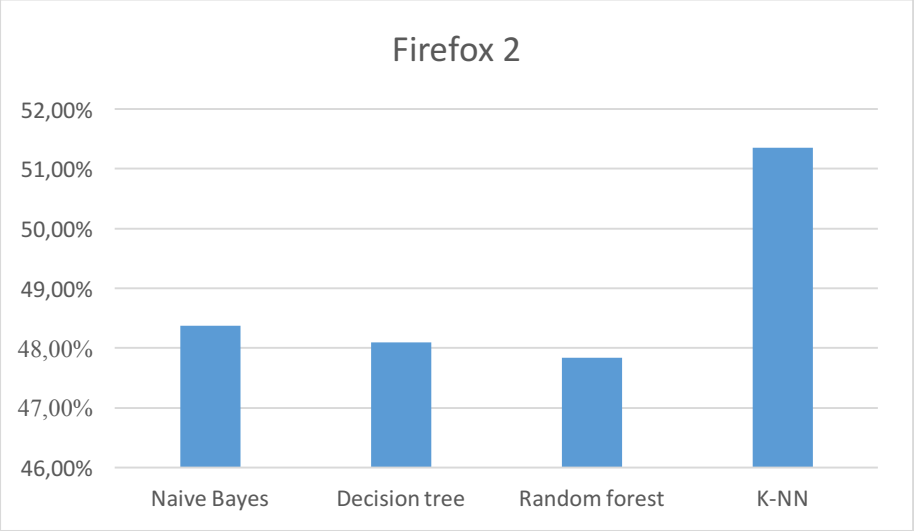


Gambar 4.1 Hasil akurasi prediksi dataset Firefox 1

Pengujian tanpa praproses untuk dataset Firefox 1 menghasilkan akurasi sebesar 69,05% untuk metode Naive Bayes, 65,25% untuk *decision tree*, 66,82% untuk *random forest*, dan 68,49% untuk K-NN. Akurasi tertinggi dari dataset Firefox 1 diperoleh dengan metode Naive Bayes, sedangkan akurasi terendah diperoleh dengan metode *decision tree*.

4.2.2 Dataset Firefox 2

Pada dataset Firefox 2 akan dilakukan pengujian dengan 4 metode klasifikasi yang sudah ditentukan. Hasil yang didapat dengan melakukan prediksi menggunakan WEKA tools dapat dilihat pada Gambar 4.2.

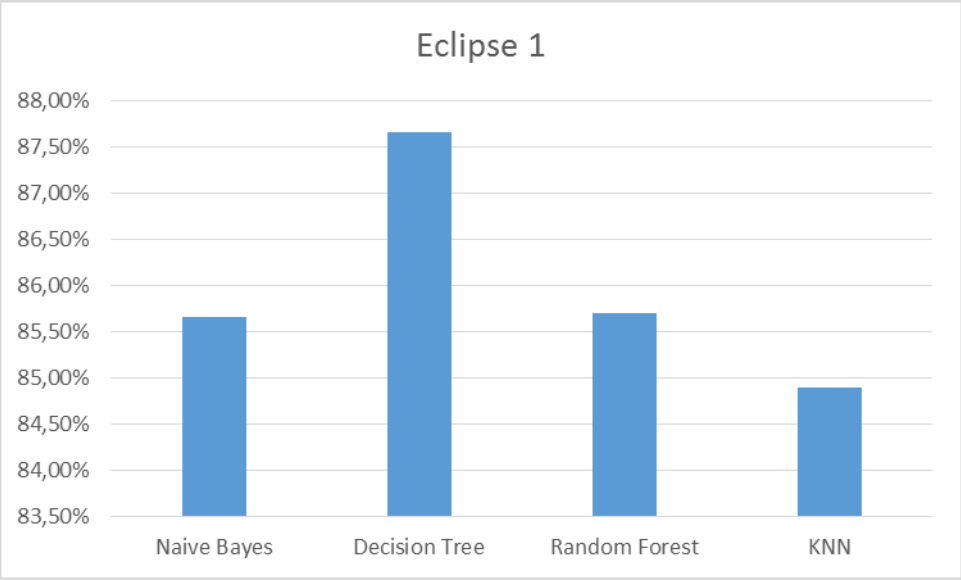


Gambar 4.2 Hasil akurasi prediksi dataset Firefox 2

Pengujian tanpa praproses untuk dataset Firefox 2 menghasilkan akurasi sebesar 48,37% untuk metode Naive Bayes, 48,10% untuk *decision tree*, 47,48% untuk *random forest*, dan 51,36% untuk K-NN. Akurasi tertinggi dari dataset Firefox 2 diperoleh dengan metode K-NN, sedangkan akurasi terendah diperoleh dengan metode *random forest*.

4.2.3 Dataset Eclipse 1

Pada dataset Eclipse 1 akan dilakukan pengujian dengan 4 metode klasifikasi yang sudah ditentukan. Hasil yang didapat dengan melakukan prediksi menggunakan WEKA tools dapat dilihat pada Gambar 4.3.

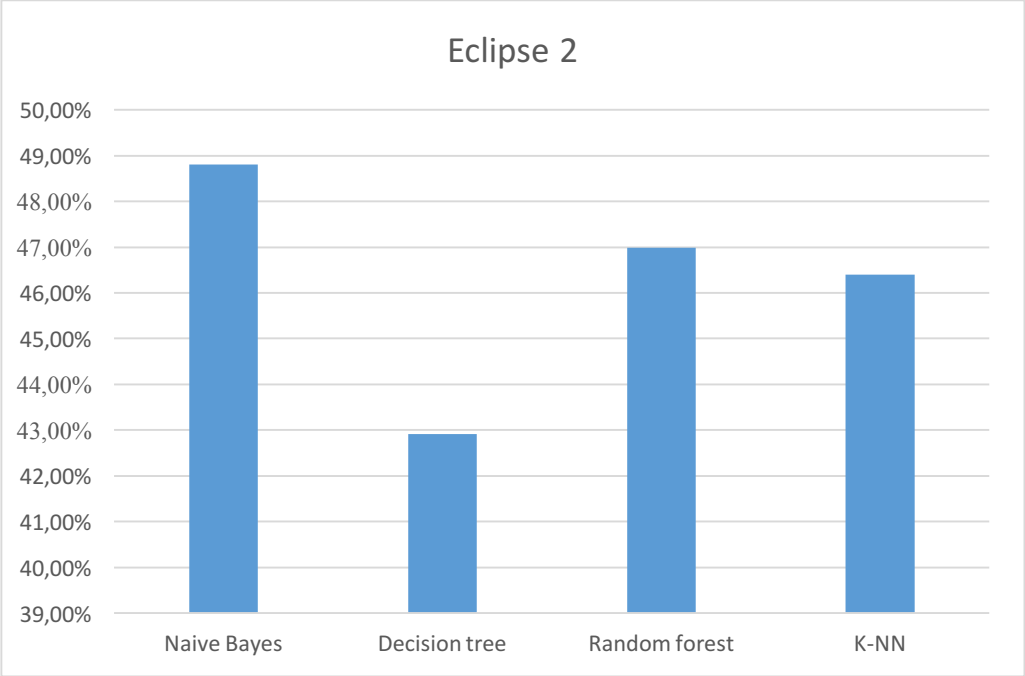


Gambar 4.3 Hasil akurasi prediksi dataset Eclipse 1

Pengujian tanpa praproses untuk dataset Eclipse 1 menghasilkan akurasi sebesar 85,66% untuk metode Naive Bayes, 87,65% untuk *decision tree*, 85,70% untuk *random forest*, dan 84,89% untuk K-NN. Akurasi tertinggi dari dataset Eclipse 1 diperoleh dengan metode *decision tree*, sedangkan akurasi terendah diperoleh dengan metode K-NN.

4.2.4 Dataset Eclipse 2

Pada dataset Eclipse 2 akan dilakukan pengujian dengan 4 metode klasifikasi yang sudah ditentukan. Hasil yang didapat dengan melakukan prediksi menggunakan WEKA tools dapat dilihat pada Gambar 4.4.

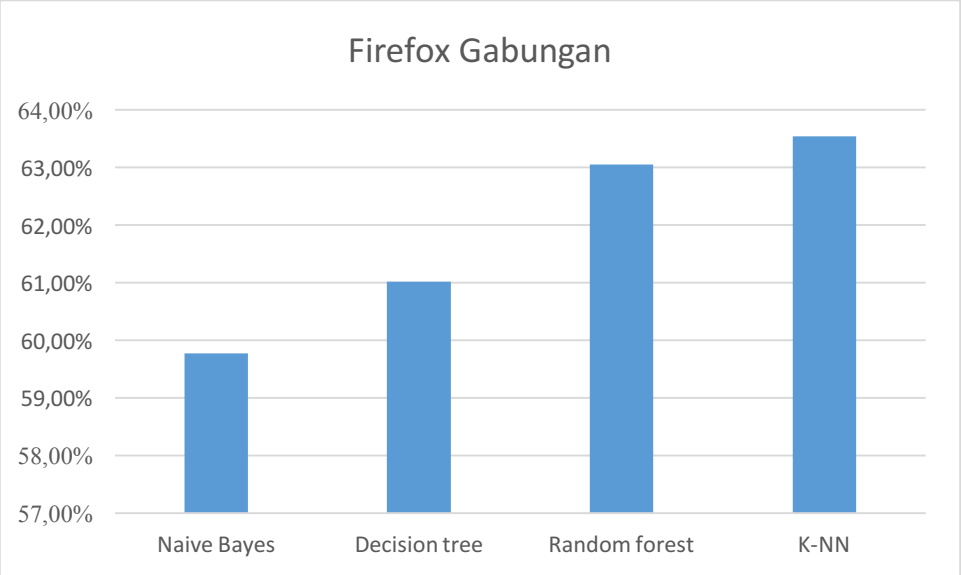


Gambar 4. 4 Hasil akurasi prediksi dataset Eclipse 2

Pengujian tanpa praproses untuk dataset Eclipse 2 menghasilkan akurasi sebesar 48,81% untuk metode Naive Bayes, 42,91% untuk *decision tree*, 46,99% untuk *random forest*, dan 46,40% untuk K-NN. Akurasi tertinggi dari dataset Eclipse 2 diperoleh dengan metode Naive Bayes, sedangkan akurasi terendah diperoleh dengan metode *decision tree*.

4.2.5 Dataset Firefox Gabungan

Pada dataset Firefox gabungan akan dilakukan pengujian dengan 4 metode klasifikasi yang sudah ditentukan. Hasil yang didapat dengan melakukan prediksi menggunakan WEKA tools dapat dilihat pada Gambar 4.5.

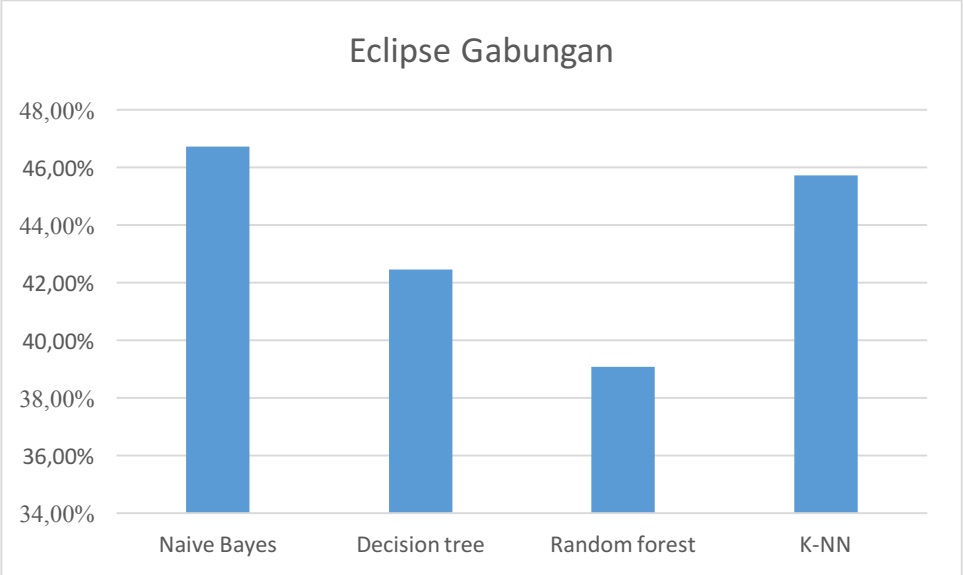


Gambar 4.5 Hasil prediksi akurasi dataset Firefox gabungan

Pengujian tanpa praproses untuk dataset Firefox gabungan menghasilkan akurasi sebesar 59,78% untuk metode Naive Bayes, 61,02% untuk *decision tree*, 63,05% untuk *random forest*, dan 63,54% untuk K-NN. Akurasi tertinggi dari dataset Firefox gabungan diperoleh dengan metode K-NN, sedangkan akurasi terendah diperoleh dengan metode Naive Bayes.

4.2.6 Dataset Eclipse Gabungan

Pada dataset Eclipse gabungan akan dilakukan pengujian dengan 4 metode klasifikasi yang sudah ditentukan. Hasil yang didapat dengan melakukan prediksi menggunakan WEKA tools dapat dilihat pada Gambar 4.6.



Gambar 4.6 Hasil akurasi prediksi dataset Eclipse gabungan

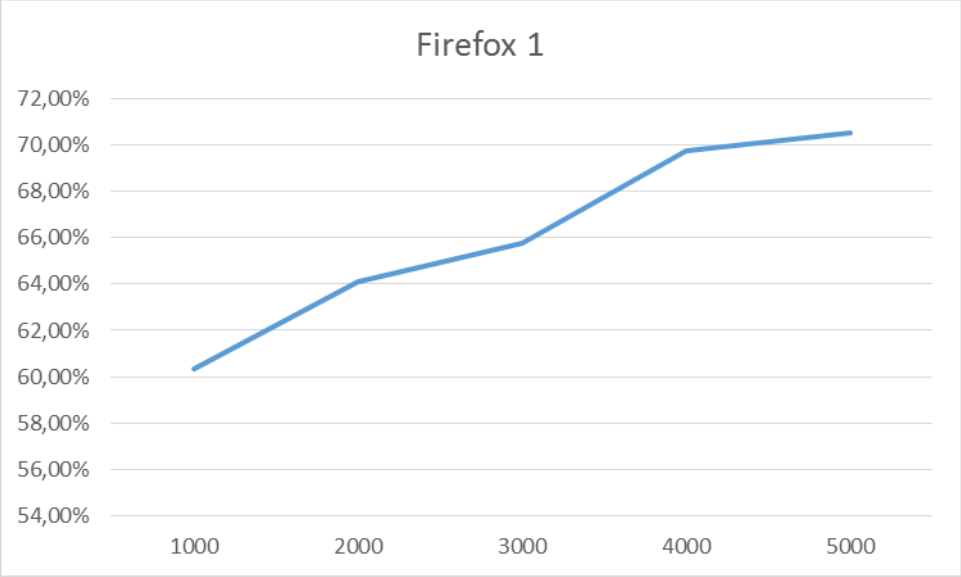
Pengujian tanpa praproses untuk dataset Eclipse gabungan menghasilkan akurasi sebesar 46,73% untuk metode Naive Bayes, 42,46% untuk *decision tree*, 39,08% untuk *random forest*, dan 45,73% untuk K-NN. Akurasi tertinggi dari dataset Firefox gabungan diperoleh dengan metode Naive Bayes, sedangkan akurasi terendah diperoleh dengan metode *random forest*.

4.3 Penentuan *threshold*

Tahap penentuan *threshold* untuk mencari *threshold* yang digunakan dalam menjalankan algoritma praproses yang diusulkan. *Threshold* yang akan diuji cobakan adalah 1000, 2000, 3000, 4000, dan 5000. Penggunaan nilai *threshold* dengan nilai maksimal sebesar 5000 bertujuan agar kelompok yang terbentuk dari dataset awal tidak terlalu banyak atau tidak terlalu sedikit. Karena populasi dari tiap dataset awal berkisar antara 16495 hingga 43393 maka pada penelitian ini menggunakan rentang uji *threshold* dari 1000 hingga 5000. Penggunaan *threshold* yang terlalu besar akan membuat pengelompokan menjadi sangat sedikit dan penggunaan threshold yang terlalu kecil akan membuat pengelompokan menjadi

sangat banyak. Masing-masing *threshold* akan diuji cobakan pada satu dataset yang ditentukan yaitu dataset Firefox 1.

Penggunaan *threshold* tersebut akan membagi dataset awal menjadi beberapa partisi. Partisi dataset tersebut akan berbeda-beda jumlahnya tergantung dari *threshold* yang ditentukan, semakin tinggi *threshold* maka akan semakin sedikit jumlah partisi dataset. Untuk tiap partisi dataset akan dilakukan metode klasifikasi yang telah ditentukan (Naive Bayes, *decision tree, random forest,* dan K-NN). Setelah didapatkan akurasi untuk semua partisi dataset dengan semua metode klasifikasi, maka akan didapatkan rata-rata dari akurasi untuk semua partisi dataset. Nilai rata-rata ini yang akan digunakan sebagai penentu *threshold* yang digunakan. *Threshold* dengan akurasi tertinggi akan digunakan untuk selanjutnya pada semua dataset awal. Hasil akurasi dari metode penentuan *threshold* dapat dilihat pada Gambar 4.7.



Gambar 4.7 Hasil akurasi mekanisme penentuan threshold

Dari hasil yang didapatkan saat diambil keputusan bahwa penggunaan *threshold* sebesar 5000 memberikan akurasi yang paling tinggi jika dibandingkan

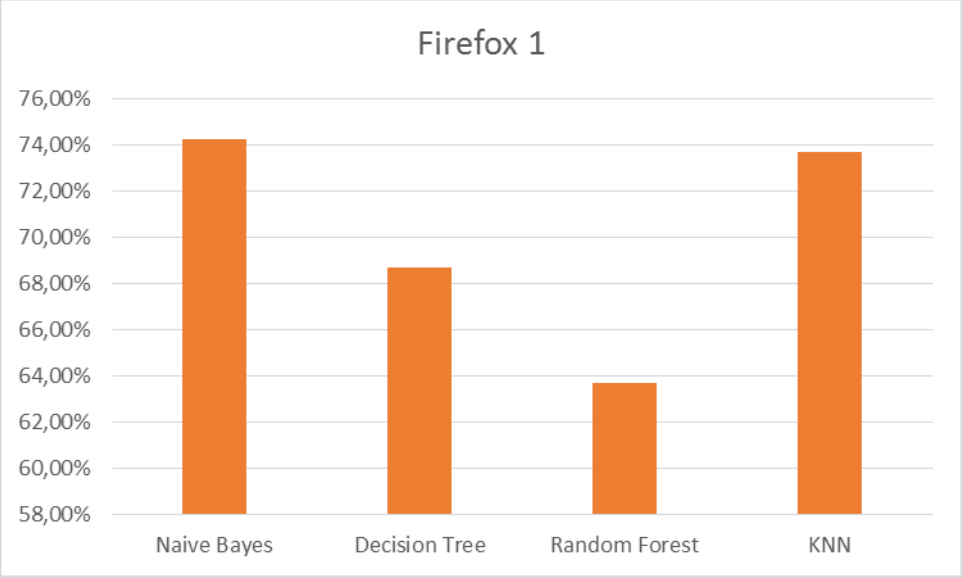
dengan *threshold* lainnya. Oleh karena itu untuk dataset lain, akan digunakan *threshold* 5000 untuk menjalankan metode praproses yang diusulkan.

4.4 Pengujian Menggunakan Praproses

Pengujian menggunakan praproses ini akan mencoba mengambil hasil akurasi dari prediksi untuk semua dataset yang digunakan setelah menggunakan praproses dengan *threshold* sebesar 5000. Metode prediksi yang digunakan untuk tiap dataset adalah Naive Bayes, *decision tree*, *random forest*, dan K-NN. Untuk hasil pada tiap dataset dapat dilihat pada sub-sub bab dibawah ini :

4.4.1 Dataset Firefox 1

Pada dataset Firefox 1 akan dilakukan pengujian dengan 4 metode klasifikasi yang sudah ditentukan. Hasil yang didapat dengan melakukan prediksi menggunakan WEKA tools dapat dilihat pada Gambar 4.8.



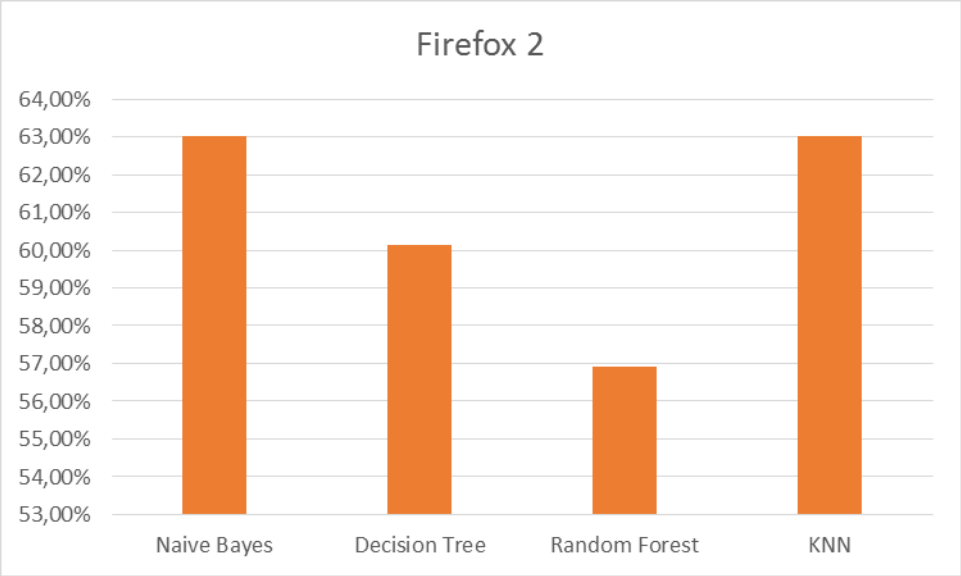
Gambar 4.8 Hasil akurasi prediksi dataset Firefox 1 dengan praproses

Pengujian dengan praproses untuk dataset Firefox 1 menghasilkan akurasi sebesar 74,22% untuk metode Naive Bayes, 68,67% untuk *decision tree*, 63,69%

untuk *random forest*, dan 73,66% untuk K-NN. Akurasi tertinggi dari dataset Firefox 1 diperoleh dengan metode Naive Bayes, sedangkan akurasi terendah diperoleh dengan metode *random forest*.

4.4.2 Dataset Firefox 2

Pada dataset Firefox 2 akan dilakukan pengujian dengan 4 metode klasifikasi yang sudah ditentukan. Hasil yang didapat dengan melakukan prediksi menggunakan WEKA tools dapat dilihat pada Gambar 4.9.

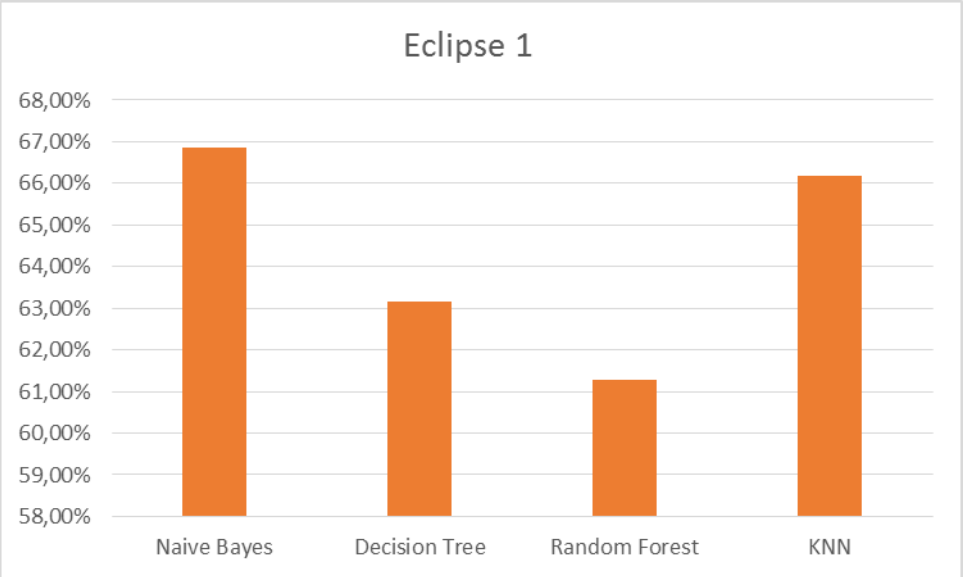


Gambar 4.9 Hasil akurasi prediksi dataset Firefox 2 dengan praproses

Pengujian dengan praproses untuk dataset Firefox 2 menghasilkan akurasi sebesar 63,02% untuk metode Naive Bayes, 60,15% untuk *decision tree*, 56,92% untuk *random forest*, dan 63,03% untuk K-NN. Akurasi tertinggi dari dataset Firefox 2 diperoleh dengan metode K-NN, sedangkan akurasi terendah diperoleh dengan metode *random forest*.

4.4.3 Dataset Eclipse 1

Pada dataset Eclipse 1 akan dilakukan pengujian dengan 4 metode klasifikasi yang sudah ditentukan. Hasil yang didapat dengan melakukan prediksi menggunakan WEKA tools dapat dilihat pada Gambar 4.10

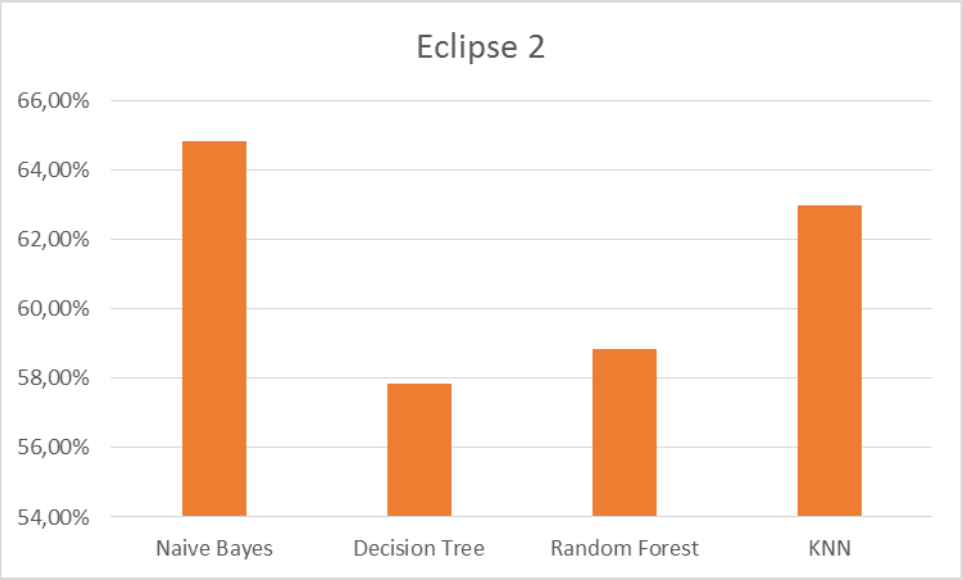


Gambar 4.10 Hasil akurasi prediksi dataset Eclipse 1 dengan praproses

Pengujian dengan praproses untuk dataset Eclipse 1 menghasilkan akurasi sebesar 66,86% untuk metode Naive Bayes, 63,16% untuk *decision tree*, 61,27% untuk *random forest*, dan 66,17% untuk K-NN. Akurasi tertinggi dari dataset Eclipse 1 diperoleh dengan metode Naive Bayes, sedangkan akurasi terendah diperoleh dengan metode *random forest*.

4.4.4 Dataset Eclipse 2

Pada dataset Eclipse 2 akan dilakukan pengujian dengan 4 metode klasifikasi yang sudah ditentukan. Hasil yang didapat dengan melakukan prediksi menggunakan WEKA tools dapat dilihat pada Gambar 4.11.

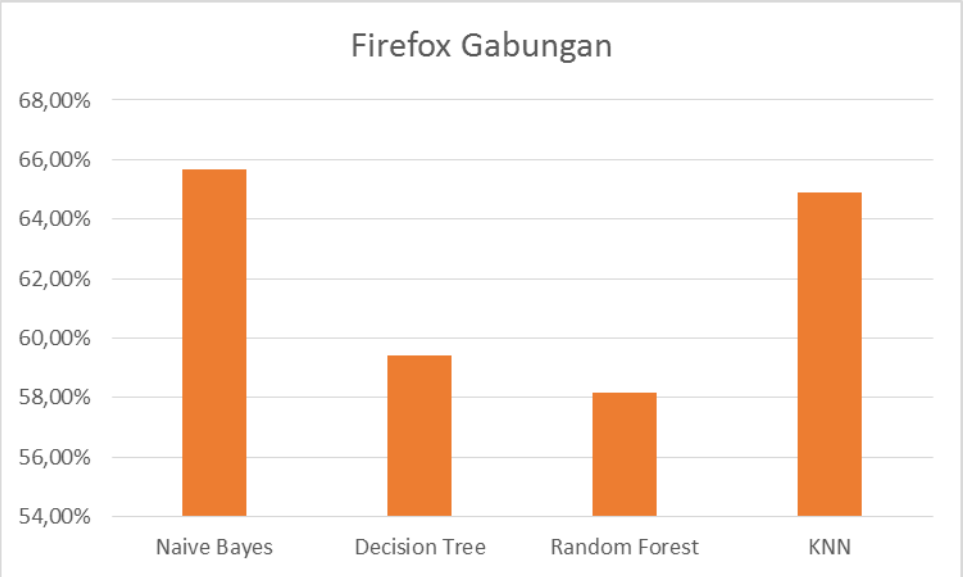


Gambar 4.11 Hasil akurasi prediksi dataset Eclipse 2 dengan praproses

Pengujian dengan praproses untuk dataset Eclipse 2 menghasilkan akurasi sebesar 64,85% untuk metode Naive Bayes, 57,85% untuk *decision tree*, 58,85% untuk *random forest*, dan 62,97% untuk K-NN. Akurasi tertinggi dari dataset Eclipse 2 diperoleh dengan metode Naive Bayes, sedangkan akurasi terendah diperoleh dengan metode *decision tree*.

4.4.5 Dataset Firefox Gabungan

Pada dataset Firefox gabungan akan dilakukan pengujian dengan 4 metode klasifikasi yang sudah ditentukan. Hasil yang didapat dengan melakukan prediksi menggunakan WEKA tools dapat dilihat pada Gambar 4.12.

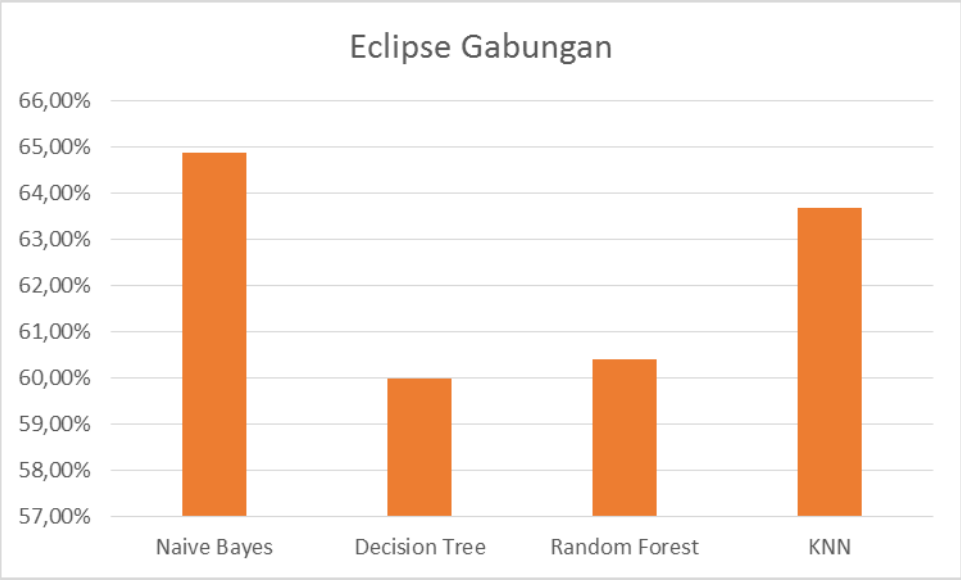


Gambar 4.12 Hasil akurasi prediksi dataset Firefox gabungan dengan praproses

Pengujian dengan praproses untuk dataset Firefox gabungan menghasilkan akurasi sebesar 65,68% untuk metode Naive Bayes, 59,43% untuk *decision tree*, 58,15% untuk *random forest*, dan 64,90% untuk K-NN. Akurasi tertinggi dari dataset Firefox gabungan diperoleh dengan metode Naive Bayes, sedangkan akurasi terendah diperoleh dengan metode *random forest*.

4.4.6 Dataset Eclipse Gabungan

Pada dataset Eclipse gabungan akan dilakukan pengujian dengan 4 metode klasifikasi yang sudah ditentukan. Hasil yang didapat dengan melakukan prediksi menggunakan WEKA tools dapat dilihat pada Gambar 4.13.



Gambar 4.13 Hasil akurasi prediksi dataset Eclipse gabungan dengan praproses

Pengujian dengan praproses untuk dataset Eclipse gabungan menghasilkan akurasi sebesar 64,87% untuk metode Naive Bayes, 59,97% untuk *decision tree*, 60,32% untuk *random forest*, dan 63,66% untuk K-NN. Akurasi tertinggi dari dataset Firefox gabungan diperoleh dengan metode Naive Bayes, sedangkan akurasi terendah diperoleh dengan metode *decision tree*.

4.5 Analisa Hasil dan Pembahasan

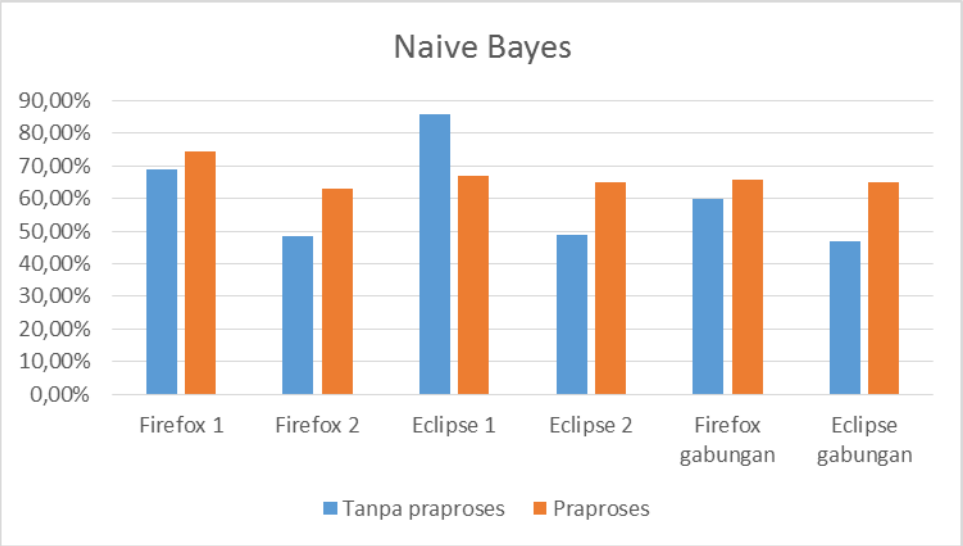
Berdasarkan hasil pengujian prediksi dari dataset yang digunakan, dapat dibandingkan akurasi antara pengujian menggunakan praproses dengan pengujian tanpa praproses. Untuk melihat perbandingan dari akurasi tanpa praproses dengan menggunakan praproses dapat dilihat lebih jelas pada Tabel 4.2.

Tabel 4.2 Perbandingan akurasi prediksi dataset

Dataset	Naive Bayes		Decision Tree		Random Forest		K-NN	
	Tanpa	Praproses	Tanpa	Praproses	Tanpa	Praproses	Tanpa	Praproses
Firefox 1	69,05%	**74,22%**	65,25%	**68,67%**	66,82%	**63,69%**	68,49%	**73,66%**
Firefox 2	48,37%	**63,02%**	48,10%	**60,15%**	47,84%	**56,92%**	51,35%	**63,03%**
Eclipse 1	85,66%	**66,86%**	87,65%	**63,16%**	85,70%	**61,27%**	84,89%	**66,17%**
Eclipse 2	48,81%	**64,85%**	42,91%	**57,85%**	46,99%	**58,85%**	46,40%	**62,97%**
Firefox gabungan	59,78%	**65,68%**	61,02%	**59,43%**	63,05%	**58,15%**	63,54%	**64,90%**
Eclipse gabungan	46,73%	**64,87%**	42,46%	**59,97%**	39,08%	**60,32%**	45,73%	**63,66%**

Hasil yang diperoleh dari uji coba pada seluruh dataset, dapat diambil kesimpulan bahwa mayoritas dari uji coba menggunakan praproses dapat menaikkan akurasi dari prediksi tanpa menggunakan praproses. Setelah dilakukan pengujian sebanyak 24 kali dapat dilihat bahwa 17 pengujian terbukti dapat meningkatkan akurasi prediksi sedangkan ada 7 pengujian dimana akurasi prediksi mengalami penurunan.

Hasil dari dataset Firefox 1 mengalami peningkatan akurasi untuk metode Naive Bayes, *decision tree,* dan K-NN, sedangkan mengalami penurunan akurasi untuk *random forest*. Hasil dari dataset Firefox gabungan mengalami peningkatan akurasi untuk metode Naive Bayes dan K-NN, sedangkan mengalami penurunan untuk *decision tree* dan *random forest.* Hasil dari dataset Firefox 2, Eclipse 2, dan Eclipse gabungan mengalami peningkatan akurasi untuk semua metode. Hasil dari dataset Eclipse 1 mengalami penurunan akurasi untuk semua metode. Grafik hasil akurasi untuk tiap metode pada dataset dapat dilihat pada Gambar 4.14 - 4.17.



Gambar 4.14 Grafik perbandingan akurasi metode Naive Bayes

Gambar 4.14 menunjukkan bahwa akurasi dari tiap pengujian metode Naive Bayes dengan dataset yang digunakan menghasilkan peningkatan akurasi untuk tiap dataset, terkecuali dataset Eclipse 1. Dataset Firefox 1 mengalami peningkatan sebanyak 5,17% , dataset Firefox 2 mengalami peningkatan sebanyak 14,65%, dataset Eclipse 1 mengalami penurunan sebanyak 18,80%, dataset Eclipse 2 mengalami peningkatan sebanyak 16,04%, dataset Firefox gabungan mengalami peningkatan sebanyak 5,90%, dan dataset Eclipse gabungan mengalami peningkatan sebanyak 18,14%.

Dataset Eclipse 1 menjadi satu-satunya dataset yang mengalami penurunan akurasi jika dibandingkan dengan dataset lainnya untuk metode Naive Bayes. Penyebab turunnya akurasi pada dataset Eclipse 1 adalah adanya variasi dari atribut yang cukup sedikit jika dibandingkan dengan dataset yang lain. Sedikitnya variasi atribut menandakan dataset yang digunakan sudah cukup homogen untuk dilakukan klasifikasi, sehingga akurasi sebelum praproses menjadi sangat tinggi jika dibandingkan dengan praproses. Jumlah variasi atribut dataset awal dapat dilihat pada Tabel 4.3.

Tabel 4.3 Variasi atribut dari dataset awal tanpa praproses

Dataset	Atribut									Jumlah
	Hardware	OS	Assignee	Reporter	Product	Severity	Priority	Component	Resolution	
F1	8	33	601	3686	28	7	5	192	8	16495
F2	11	35	1196	3326	6	7	6	217	6	27438
E1	6	28	206	4606	5	7	5	28	6	17499
E2	6	31	403	4628	4	7	5	28	6	17961
Fg	12	53	1663	6269	33	7	5	386	8	40833
Eg	6	35	459	8227	5	7	5	29	6	35460

Tabel 4.3 merepresentasikan variasi dari dataset awal sebelum praproses. Dari dataset awal tersebut dilakukan praproses sehingga menjadi beberapa partisi untuk tiap dataset awal. Untuk lebih jelas dapat dilihat seperti Tabel 4.4 dibawah.

Tabel 4.4 Variasi atribut dataset setelah praproses

Dataset	Part	Atribut										Jumlah
		Hardware	OS	Assignee	Reporter	Product	Severity	Priority	Component	Resolution	Jam	
f1	1	8	25	337	1389	24	7	5	167	8	1018	5000
f1	2	8	28	367	1643	19	7	5	126	8	642	5009
f1	3	8	30	355	1449	17	7	5	115	6	914	6486
f2	1	11	26	488	1136	6	7	5	178	6	16	5074
f2	2	7	20	592	935	6	7	5	173	6	42	5095
f2	3	8	23	628	1001	6	7	5	173	6	104	5038
f2	4	8	28	790	1751	6	7	5	167	6	1733	9131
e1	1	6	24	109	1809	5	7	5	27	6	5	5274
e1	2	4	19	121	1386	4	7	5	24	6	65	5012
e1	3	6	27	183	2468	5	7	5	26	6	2359	7213
e2	1	6	23	149	1773	4	7	5	25	6	5	5124
e2	2	6	20	204	1251	4	7	5	25	6	62	5014
e2	3	6	28	333	2558	4	7	5	27	6	2694	7823
fg	1	11	32	513	1267	19	7	5	269	6	12	5025
fg	2	10	28	625	1074	23	7	5	252	6	5036	5064
fg	3	9	29	630	1074	18	7	5	253	6	62	5014
fg	4	9	28	664	1169	20	7	5	251	6	168	5009
fg	5	10	35	669	1527	21	7	5	258	8	487	5002
fg	6	11	39	542	1694	24	7	5	205	8	737	5011
fg	7	10	28	370	1523	21	7	5	124	7	278	5013
fg	8	9	33	403	1392	20	7	5	136	6	1028	5517
eg	1	6	29	185	2806	5	7	5	28	6	2	8117
eg	2	6	23	189	1618	4	7	5	25	6	10	5016
eg	3	6	21	213	1433	4	7	5	26	6	30	5085
eg	4	6	23	226	1453	4	7	5	24	6	100	5013
eg	5	6	24	224	1634	5	7	5	27	6	398	5002
eg	6	6	27	294	2788	5	7	5	26	6	7174	7227

Tabel 4.4 merupakah variasi atribut dataset yang sudah dilakukan praproses. Sebagai contoh dataset F1 dan part 1 adalah dataset Firefox 1 untuk partisi pertama, begitu pula untuk part 2 dan 3. Untuk E1 adalah dataset Eclipse 1, kemudian Fg dan Eg adalah Firefox gabungan dan Eclipse gabungan. Nilai pada kolom atribut merupakan variasi dari atribut yang terdapat pada partisi tersebut, sedangkan kolom jumlah merupakan populasi pada partisi tersebut.

Berdasarkan Tabel 4.3 atribut yang masih memiliki banyak variasi adalah *assignee* dan *reporter*. Pada dataset Eclipse 1 memiliki variasi *assignee* yang paling sedikit, sehingga dataset Eclipse 1 merupakan yang paling homogen dalam atribut tersebut. Hal ini menyebabkan akurasi dari klasifikasi tanpa praproses pada dataset Eclipse 1 menjadi yang paling tinggi jika dibandingkan dengan dataset lain pada metode Naive Bayes.

Penyebab tingginya akurasi saat dilakukan klasifikasi tanpa praproses pada dataset Eclipse 1 adalah karena variasi dari atribut yang cenderung lebih sedikit dibandingkan dengan dataset lainnya. Hal ini menyebabkan pada saat perhitungan fungsi posterior untuk atribut tersebut akan memiliki peluang yang lebih tinggi jika dibandingkan dengan memiliki lebih banyak variasi. Tingginya peluang tersebut akan mempengaruhi hasil prediksi serta dapat mempengaruhi akurasi prediksi saat dilakukan evaluasi.

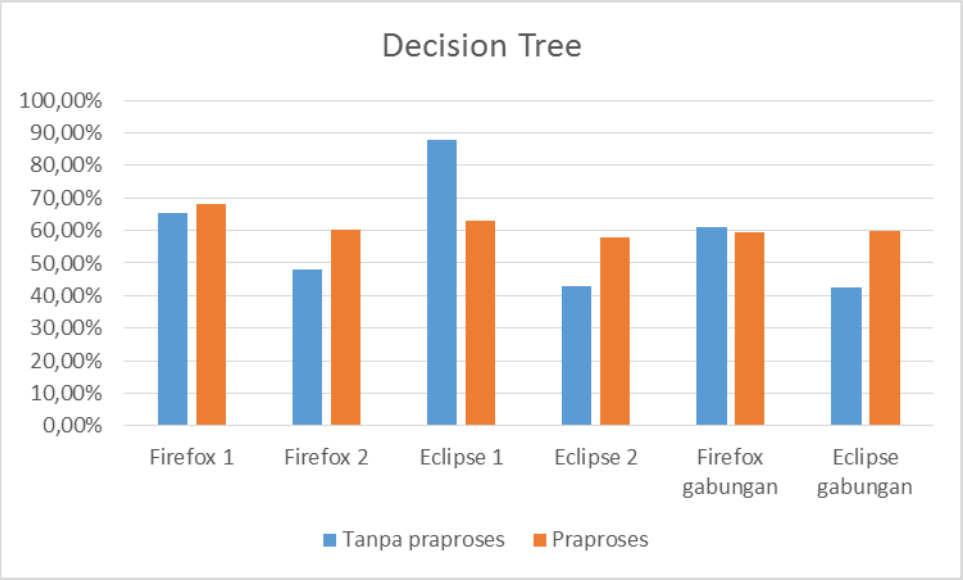
Untuk membuktikan pengaruh dari atribut yang digunakan pada pengujian maka dilakukan penilaian atribut pada 6 dataset awal. Metode yang digunakan adalah *oneRAttributeEval* yaitu dengan melakukan penilaian terhadap masing-masing atribut pada dataset yang selanjutnya diklasifikasi menggunakan *oneRClassifier*. Atribut dengan akurasi tertinggi pada klasifikasi *oneR* menunjukkan bahwa atribut tersebut paling berpengaruh terhadap klasifikasi.

Setelah dilakukan uji seleksi fitur pada 6 dataset awal diperoleh hasil rangking atribut untuk tiap dataset yang telah ditotal perolehan rangkingnya. Untuk lebih jelas dapat dilihat pada Tabel 4.5.

Tabel 4.5 Hasil seleksi fitur dataset

Rangking	Atribut	Jumlah rangking
1	Assignee	10
2	Reporter	13
3	Component	23
4	OS	24
5	Resolution	31
6	Product	36
7	Hardware	42
8	Priority	43
9	Severity	49

Berdasarkan hasil yang didapatkan dari pengujian menggunakan *oneRAttributeEval* pada ke 6 dataset awal menunjukkan bahwa atribut *assignee* merupakan yang paling berpengaruh terhadap klasifikasi. Hal tersebut terbukti dari jumlah rangking pada atribut tersebut yang cukup rendah menandakan pentingnya atribut tersebut dalam klasifikasi.

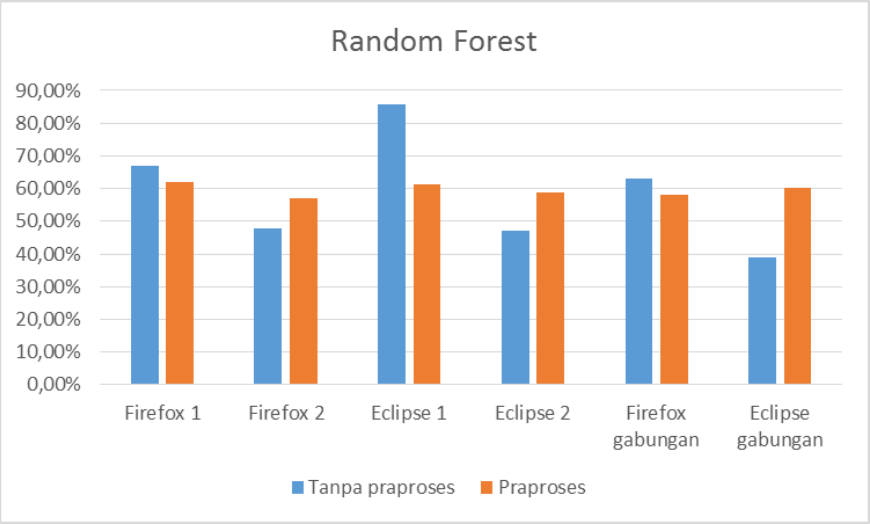


Gambar 4.15 Grafik perbandingan akurasi decision tree

Gambar 4.15 menunjukkan bahwa akurasi dari tiap pengujian metode *decision* tree dengan dataset yang digunakan menghasilkan peningkatan akurasi untuk tiap dataset terkecuali Eclipse 1 dan Firefox gabungan. Dataset Firefox 1 mengalami peningkatan sebanyak 3,42%, dataset Firefox 2 mengalami peningkatan sebanyak 12,05%, dataset Eclipse 1 mengalami penurunan sebanyak 24,49%, dataset Eclipse 2 mengalami peningkatan sebanyak 14,49%, dataset Firefox gabungan mengalami penurunan sebanyak 1,59%, dan dataset Eclipse gabungan mengalami peningkatan sebanyak 17,51%.

Pada percobaan menggunakan metode *decision tree* penurunan yang cukup signifikan juga terjadi pada dataset Eclipse 1. Penyebab tingginya akurasi tanpa praproses pada metode *decision tree* yang digunakan juga dampak dari variasi dari atribut dataset Eclipse 1 yang cenderung lebih sedikit dibandingkan dataset yang lainnya.

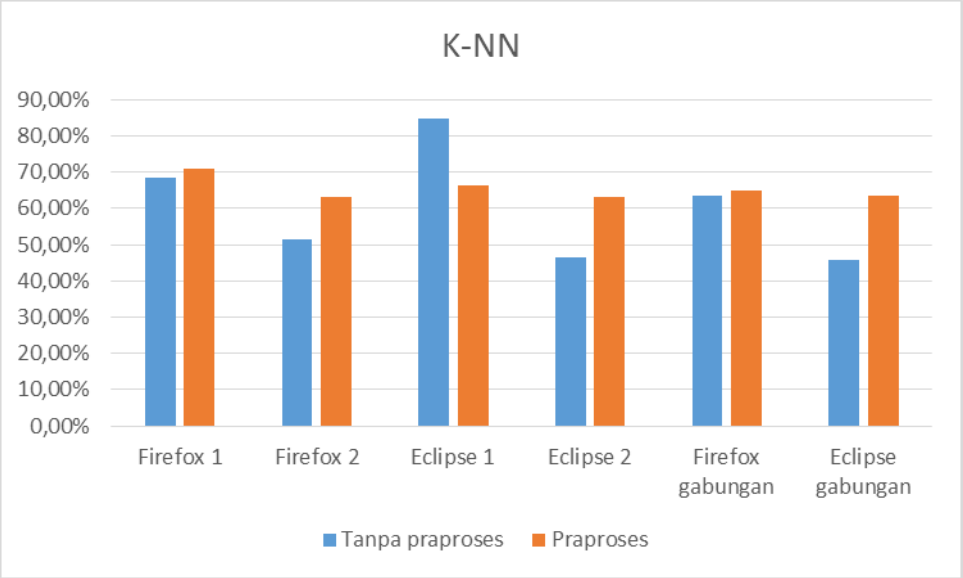
Sebelumnya telah dijelaskan pada tinjauan pustaka bahwa metode *decision tree* menggunakan perhitungan nilai *entropy* dan *gain* dalam menentukan *tree*. Sedangkan nilai-nilai tersebut sangat dipengaruhi oleh variasi jenis data untuk tiap atribut. Semakin banyak variasi nilai pada atribut tertentu akan mempengaruhi hasil dari perhitungan *entropy* atribut tersebut untuk menentukan *node*.



Gambar 4.16 Grafik perbandingan akurasi random forest

Gambar 4.16 menunjukkan bahwa akurasi dari tiap pengujian metode *random forest* dengan dataset yang digunakan menghasilkan peningkatan akurasi untuk dataset Firefox 2, Eclipse 2 dan Eclipse gabungan. Dataset Firefox 1 mengalami penurunan sebanyak 3,13%, dataset Firefox 2 mengalami peningkatan sebanyak 9,08%, dataset Eclipse 1 mengalami penurunan sebanyak 24,43%, dataset Eclipse 2 mengalami peningkatan sebanyak 11,86%, dataset Firefox gabungan mengalami penurunan sebanyak 4,90%, dan dataset Eclipse gabungan mengalami peningkatan sebanyak 21,24%.

Pada percobaan menggunakan metode *random forest* memiliki kemiripan hasil dengan metode *decison tree*. Kesamaan yang terlihat jelas yaitu penurunan cukup signifikan pada dataset Eclipse 1, dan peningkatan yang cukup signifikan pada dataset Firefox 2, Eclipse 2, dan Eclipse gabungan. Hal ini disebabkan karena metode *random forest* merupakan pengembangan dari metode *decision tree*, dimana pada *random forest* menggunakan banyak *tree* sehingga penentuan dari *node* untuk masing-masing *tree* tetap mempertimbangkan nilai *entropy* dan *gain* dari masing-masing atribut.



Gambar 4.17 Grafik perbandingan akurasi K-NN

Gambar 4.17 menunjukkan bahwa akurasi dari tiap pengujian metode K-NN dengan dataset yang digunakan menghasilkan peningkatan akurasi untuk seluruh dataset terkecuali Eclipse 1. Dataset Firefox 1 mengalami peningkatan sebanyak 5,17%, dataset Firefox 2 mengalami peningkatan sebanyak 11,68%, dataset Eclipse 1 mengalami penurunan sebanyak 18,72%, dataset Eclipse 2 mengalami peningkatan sebanyak 16,57%, dataset Firefox gabungan mengalami peningkatan sebanyak 1,36%, dan dataset Eclipse gabungan mengalami peningkatan sebanyak 17,93%.

Pada percobaan dengan metode K-NN memiliki kemiripan dengan metode Naive Bayes dimana hanya mengalami penurunan pada dataset Eclipse 1. Hal ini dapat disebabkan karena variasi dari atribut yang cenderung lebih sedikit dibandingkan dengan dataset lainnya. Seperti yang telah dijelaskan pada tinjauan pustaka bahwa variasi dari atribut akan dijadikan sebuah atribut *dummy* yang merepresentasikan atribut awal dengan cara *binary encoding*, maka dengan banyaknya variasi atribut akan mempengaruhi hasil prediksi serta akurasi.

Dari percobaan 4 metode yang telah dianalisa hasilnya, dapat disimpulkan bahwa variasi dari atribut pada sebuah dataset akan berpengaruh pada hasil prediksi untuk metode Naive Bayes, *decision tree, random forest,* dan K-NN. Banyaknya variasi dari atribut membuat hasil prediksi menjadi tidak akurat jika dibandingkan dengan variasi berjumlah lebih sedikit. Dataset Eclipse 1 merupakan dataset dengan variasi atribut yang cenderung lebih sedikit jika dibandingkan dengan dataset lainnya. Sehingga pada percobaan tanpa praproses sudah menunjukkan hasil yang cukup baik jika dibandingkan dengan praproses. Terbukti pada tiap percobaan dengan praproses untuk semua metode yang digunakan, dataset Eclipse 1 selalu mengalami penurunan akurasi.

Dalam penelitian yang sudah dilakukan perlu dilakukan prediksi dini untuk menentukan sebuah dataset apakah akan mengalami penurunan akurasi atau peningkatan akurasi. Dengan adanya prediksi dini, akan membuat klasifikasi yang

dilakukan akan semakin akurat. Untuk menentukan prediksi dini tersebut perlu dilakukan beberapa penelitian dengan dataset yang lebih bervariasi.

BAB 5
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil uji coba dari metode usulan memberikan beberapa kesimpulan yaitu :

1. Metode praproses yang diusulkan dapat mengelomopkkan data sehingga meningkatkan akurasi dari prediksi waktu perbaikan laporan *bug* untuk 17 uji coba.
2. Hasil dari pengujian yang dilakukan belum dapat disimpulkan kapan perlu menggunakan praproses atau tidak, perlu dilakukan penelitian dengan lebih banyak jenis dataset perangkat lunak.
3. Variasi dari atribut suatu dataset dapat berpengaruh terhadap akurasi prediksi dari metode klasifikasi yang digunakan, terutama pada salah satu atribut yang dinilai penting dalam klasifikasi.

5.2 Saran

Kekurangan dari penelitian ini adalah metode usulan masih bergantung pada sebuah angka *threshold* yang belum pasti nilainya. *Threshold* yang digunakan dalam penelitian ini masih sebatas uji coba dengan rentang 1000, 2000, 3000, 4000, dan 5000. Untuk kedepannya dapat memungkinkan penggunaan *threshold* yang pasti nilainya. Selain adanya keterbatasan pada penentuan *threshold*, pada penelitian ini juga masih belum dapat disimpulkan kapan metode partisi ini perlu diterapkan dan kapan sebaiknya tidak diterapkan. Untuk menjawab hal tersebut perlu dilakukan penelitian menggunakan lebih banyak jenis dataset perangkat lunak.

Beberapa kasus dapat menurunkan akurasi dari prediksi, untuk kasus yang dapat menurunkan akurasi prediksi pada praproses masih belum dapat diatasi secara otomatis. Untuk mengetahui apakah ada indikasi akurasi prediksi dapat menurun harus dilakukan pengamatan secara manual. Untuk kedepannya diharapkan dapat

dilakukan prediksi dini apakah dataset yang digunakan terindikasi menurun akurasinya apabila dilakukan praproses.

DAFTAR PUSTAKA

Abdelmoez W, M. K. a. F. M. E., 2012. *Bug Fix-Time Prediction Model Using Naive Bayes Classifier.* s.l., s.n.

Alenezi, M., 2013. *Bug Reports Prioritization : Which Features and Classifier to Use ?.* s.l., s.n.

Giger Emanuel, M. P. a. H. C. G., 2010. Predicting the Fix Time of Bugs. *ACM,* pp. 52-56.

Kamber, J. H. a. M., 2006. *Data Mining: Concepts and Technique.* s.l.:Elsevier.

Nur Fajri Azhar, S. R., 2016. Memprediksi Waktu Memperbaiki Bug dari Laporan Bug Menggunakan Klasifikasi Random Forest. *Jurnal Sistem dan Informatika,* Volume 11, p. 1.

Stuart Russel, P. N., 2010. *Artificial Intelligence a Modern Approach.* 3 penyunt. New Jersey: Pearson.

Zhang Hongyu, L. G. S. V., 2013. *Predicting Bug-Fixing Time: An Empirical Study of Commercial Software Projects.* San Francisco, s.n.

Breimann, L., 2001. *Random Forest,* Machine Learning, 5-32.

Bhuwaneswari, V., V. 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.* pp. 335-339

Romi, S. W., 2016. *A Systematic Literature Review of Software Defect Prediction: Research Trends, Dataset, Methods and Frameworks.* Volume 1, pp. 1-16

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS

Penulis, Mochammad Arief Ridwan, lahir di Surabaya 31 Oktober 1994. Penulis merupakan anak ketiga dari 3 bersaudara dari pasangan suami istri Hadi Surono dan Endang Murnia. Penulis juga merupakan penikmat musik, penggemar olahraga, dan video game.

Penulis menempuh pendidikan formal di SD Negeri Wedoro 1 Sidoarjo (2001-2006), SMP Negeri 22 Surabaya (2006-2009), SMA Negeri 15 Surabaya (2009-2012), S1 Teknik Informatika Institut Teknologi Sepuluh Nopember Surabaya dengan bidang minat Rekayasa Perangkat Lunak (2012-2016). Pada tahun 2016 penulis melanjutkan pendidikan Magister di tempat yang sama yaitu Institut Teknologi Sepuluh Nopember dengan jurusan yang sama pula. Saat menempuh pendidikan S2 penulis sempat mengerjakan proyek aplikasi mobile untuk internal ITS berbasis iOS dengan nama aplikasi ePerkantoran. Pada studi pasca sarjana, penulis juga mengambil bidang minat Rekayasa Perangkat Lunak.

Selama menjalani kuliah di Institut Teknologi Sepuluh Nopember penulis juga aktif dalam beberapa organisasi mahasiswa yaitu sebagai staff Departemen Hubungan Luar Himpunan Mahasiswa Teknik Computer – Informatika (2013-2014), staff Departemen Pengembangan Sumber Daya Musik Paduan Suara Mahasiswa ITS (2013-2014), Sekretaris Departemen Pengembangan Sumber Daya Musik Paduan Suara Mahasiswa ITS (2014-2015). Selain aktif berorganisasi penulis juga beberapa kali mengikuti kompetisi bersama Paduan Suara Mahasiswa ITS baik tingkat nasional maupun internasional.