Peningkatan Akurasi Prediksi Waktu Perbaikan Bug dengan Pendekatan Partisi Data

Mochammad Arief Ridwan*, Siti Rochimah

Institut Teknologi Sepuluh Nopember

Naskah Diterima : 12 Maret 2018; Diterima Publikasi : 24 April 2018

DOI : 10.21456/vol8iss1pp76-83

Abstract

Software developers need to 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 several datasets of varying time ranges, the results obtained from a very variable time span are assessed to be further enhanced by partitioning over time ranges prior to classification. With partitions based on the repair timeframe, improved accuracy has been made with some classification methods. The results obtained after performing trials on multiple datasets are an increase for the majority of the datasets used. There is also a decrease in accuracy in some tests performed with a particular dataset.

Keywords : Prediction; Bug, Fixing time; Partition

Abstrak

Pengembang perangkat lunak perlu 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 beberapa dataset dengan rentang waktu yang amat bervariasi, hasil yang didapatkan dari rentang waktu yang amat bervariasi tersebut dinilai dapat lebih ditingkatkan lagi dengan melakukan partisi berdasarkan rentang waktu sebelum melakukan klasifikasi. Dengan partisi berdasarkan rentang waktu perbaikan, telah didapatkan peningkatan akurasi dengan beberapa metode klasifikasi. Hasil yang didapatkan setelah melakukan uji coba pada beberapa dataset adalah adanya peningkatan untuk mayoritas dataset yang digunakan. Terdapat juga penurunan akurasi pada beberapa pengujian yang dilakukan dengan dataset tertentu.

Keywords: Prediksi; Bug; Waktu perbaikan; Partisi.

1. Pendahuluan

Setiap proses pengembangan perangkat lunak memiliki mekanisme pemeliharaan. Idealnya pemeliharaan dilakukan saat perangkat lunak sudah memasuki tahap rilis. 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 *et al.* (2010) menggunakan *decision tree* untuk melakukan prediksi dengan dataset Eclipse JDT, Eclipse Platform, Mozilla Core, Mozilla Firefox, Gnome GStreamer, dan Gnome Evolution. Menurut

*Penulis korespondensi: aridwan05@mhs.if.its.ac.id

Giger akurasi dari prediksi yang diperoleh dapat ditingkatkan lagi dengan metode Naïve Bayes dan K-NN.

Penelitian lain oleh Abdelmoez *et al.* (2012) 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, namun menggunakan kuartil untuk pemberian *class* pada data sehingga dapat meningkatkan akurasi dari prediksi oleh Giger.

Selain penelitian dengan mengubah metode klasifikasi, penelitian tentang atribut yang digunakan dalam klasifikasi telah dilakukan. Alenezi dan Banitaan (2013) 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 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 dan Siti (2016) 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. 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 pengaruh partisi data berdasarkan waktu perbaikan terhadap akurasi prediksi waktu perbaikan bug.

Penelitian ini mengusulkan metode partisi dataset 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 partisi 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.

2. Kerangka Teori

2.1. Dataset

Dataset yang digunakan pada penelitian ini memiliki 10 atribut yang diambil dari Bugzilla untuk perangkat lunak Firefox dan Eclipse. Hal ini karena pada penelitian Alenezi (2013) telah dilakukan penelitian mengenai pengaruh atribut tekstual yang dapat menurunkan akurasi dari prediksi waktu perbaikan bug. Atribut yang digunakan dalam penelitian dapat dilihat pada Tabel 1.

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*.

2.2. 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 partisi yang diusulkan terhadap akurasi klasifikasi.

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 *et al*, 2006). Persamaan (1) adalah fungsi posterior pada Naive Bayes.

Tabel 1 Atribut yang digunakan dalam penelitian

No.	Atribut	Deskripsi
1	Perangkat	Perangkat *hardware*
2	Sistem Operasi	Sistem operasi yang digunakan
3	Komponen	Komponen yang terkena *bug*
4	Ditugaskan	Pengembang yang ditugaskan untuk memperbaiki *bug*
5	Pelapor	Nama pelapor yang menemukan *bug*
6	Produk	Komponen perangkat lunak yang digunakan
7	Kerasnya	Kondisi dari *bug*
8	Prioritas	Prioritas dari *bug*
9	Resolusi	Resolusi terakhir pada laporan *bug*
10	Waktu Perbaikan	Waktu yang dibutuhkan untuk memperbaiki *bug*

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)} \quad (1)$$

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.

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 and Norvig, 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.

Metode *decision tree* 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).

$$E(S) = \sum_{i=1}^{c} -p_i log p_i \quad (2)$$

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 (3).

$$D = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2} \quad (3)$$

Nilai D merupakan nilai jarak dari data ke 1 dengan data ke 2, dimana *x* dan *y* merupakan nilai dari atribut masing-masing data. Dengan menentukan nilai K, akan dicari sejumlah K dengan jarak terdekat sebagai *class* prediksi.

Usulan penggunaan K-NN dalam prediksi waktu perbaikan bug telah dilakukan oleh Zhang *et al.* (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.

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.

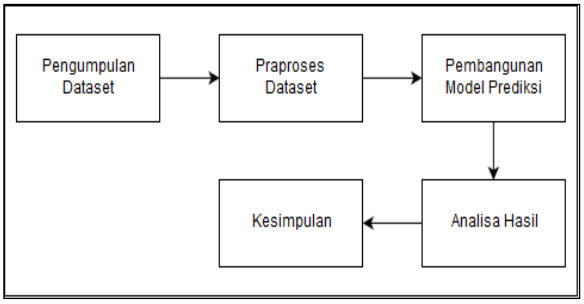
2.3. K-Cross Fold Validation

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.

3. Metode

Secara umum tahapan penelitian dapat dilihat pada Gambar 1. Tahap awal adalah pengumpulan dataset dengan menggunakan Bugzilla pada perangkat lunak Firefox dan Eclipse. Pada tahap pengumpulan dataset akan digunakan periode pengamatan tahun 2004 – 2016 untuk dataset Firefox, sedangkan untuk Eclipse menggunakan periode pengamatan tahun 2010-2016.



Gambar 1. Tahapan umum penelitian

Tahap selanjutnya dari penelitian ini adalah penerapan partisi dataset yang diusulkan. Untuk masing-masing dataset akan dilakukan partisi sehingga untuk satu dataset akan memiliki beberapa partisi dataset yang populasinya lebih kecil jika dibandingkan dataset hasil pengumpulan dataset.

Setelah dilakukan partisi untuk tiap dataset maka akan dilajutkan dengan pembangunan model prediksi untuk tiap partisi dataset yang ada. Pembangunan model prediksi menggunakan metode klasifikasi Naive Bayes, *decision tree*, *random forest*, dan K-NN berdasarkan penelitian Romi (2016) tentang prediksi kecacatan perangkat lunak.

Setelah model prediksi telah dibuat, maka akan dilakukan pengujian dengan menggunakan *K-Cross fold validation* untuk mendapatkan akurasi dari model yang sudah dibuat. Masing-masing partisi dataset akan dihitung akurasinya dan akan dibandingkan dengan akurasi tanpa partisi.

3.1. 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.

Penelitian ini menggunakan dataset dari laporan *bug* seperti yang terdapat pada Tabel 2. 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.

3.2. Partisi Dataset

Praproses yang diusulkan pada penelitian ini adalah pembagian dataset menjadi beberapa partisi yang kemudian dilakukan pembangunan model prediksi. Akan ada beberapa tahapan dalam partisi yang diusulkan, yaitu tahap pengurutan dataset, penentuan *threshold* partisi data, dan tahap pemberian *class* untuk tiap data dalam dataset. Masing-masing tahapan dari partisi dataset akan dijelaskan pada sub-sub bab dibawah. Alur partisi dan penerapan data secara keseluruhan dapat dilihat pada Gambar 2.

3.2.1. Pengurutan Dataset

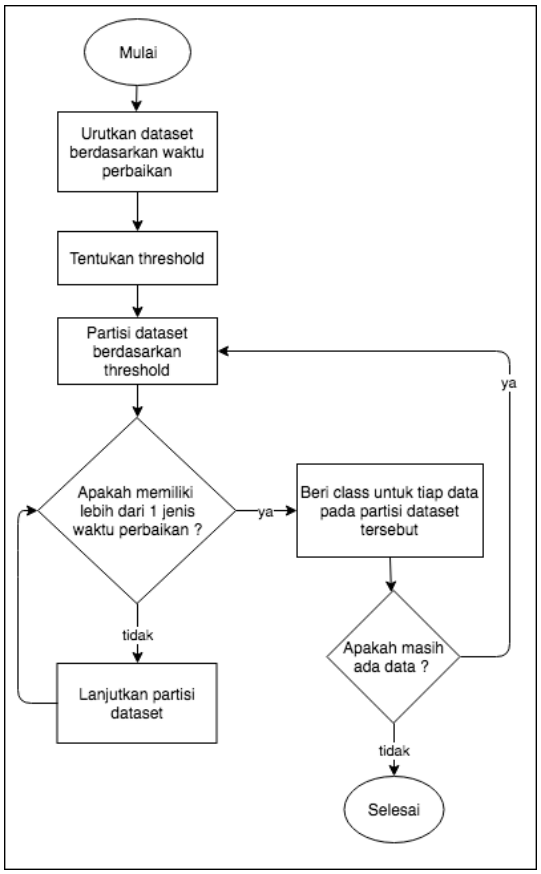
Dataset yang akan diolah untuk pertama akan diurutkan berdasarkan atribut waktu perbaikan. Atribut waktu perbaikan adalah waktu yang dibutuhkan untuk memperbaiki *bug* pada perangkat lunak berupa satuan jam mulai dari yang tercepat hingga terlama.

Tabel 1. Hasil pengumpulan dataset

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

3.2.2. Penentuan Threshold

Tahap penentuan threshold adalah menentukan batas partisi dataset yang akan digunakan. 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.



Gambar 2 Alur partisi keseluruhan

3.2.3. Pemberian Class

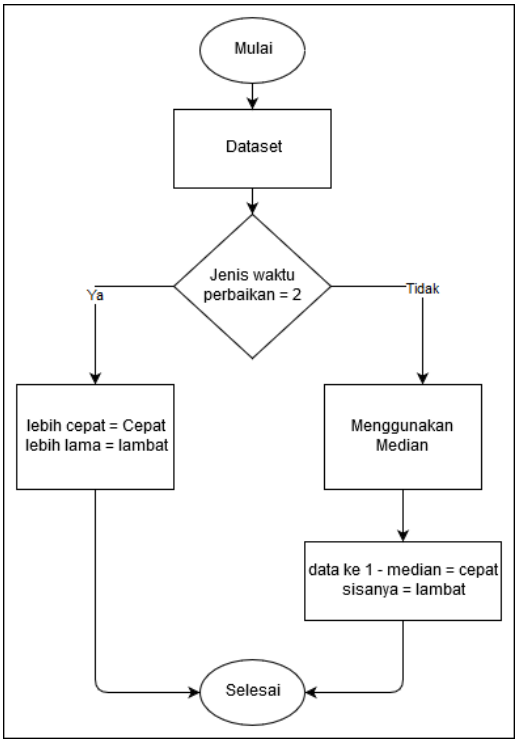
Tahap pemberian *class* merupakan prosedur penentuan *class* dari partisi dataset yang sudah dipotong berdasarkan *threshold*. Dari partisi dataset

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.

3.3. Pembanguan Model Prediksi

Pembangunan model prediksi yang akan dilakukan menggunakan 4 metode klasifikasi. Implementasi dari algoritma klasifikasi akan menggunakan alat bantu yaitu WEKA tools versi 3.8.1.

WEKA tools 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. 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.



Gambar 3. Alur pemberian class pada dataset

3.4. 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 partisi lebih tinggi dibandingkan dengan tanpa partisi, maka dapat diambil kesimpulan bahwa metode partisi yang diusulkan dapat meningkatkan akurasi dari prediksi waktu perbaikan *bug*.

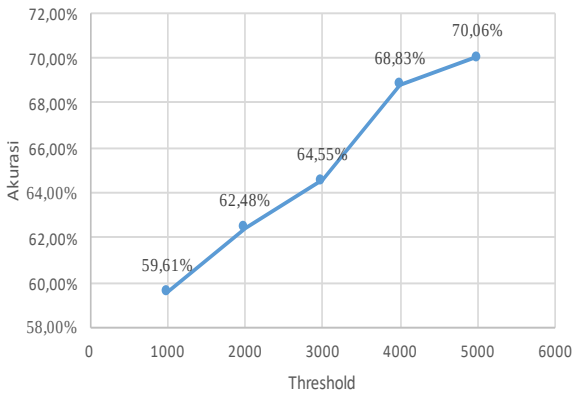
4. Hasil dan Pembahasan

4.1. Penentuan Threshold

Langkah pertama adalah penentuan *threshold* pada proses partisi dataset. Tahap penentuan *threshold* untuk mencari *threshold* yang digunakan dalam menjalankan algoritma partisi yang diusulkan. *Threshold* yang akan diuji cobakan adalah 1000, 2000, 3000, 4000, dan 5000. 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.

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 partisi yang diusulkan.



Gambar 4 Hasil akurasi penentuan threshold

4.2. Hasil Partisi Dataset

Partisi dataset sesuai metode partisi. Hasil dari partisi dataset juga akan dijabarkan dalam bentuk tabel. Berdasarkan hasil partisi dataset awal telah didapatkan beberapa partisi untuk masing-masing dataset. Jumlah partisi dataset berbeda untuk tiap dataset awal. Hal ini bergantung pada atribut waktu perbaikan dataset awal. Tabel 3 menunjukkan hasil partisi dan jumlah data untuk tiap partisi pada semua dataset yang digunakan.

Tabel 2 Hasil partisi dataset

Dataset	Partisi	Jumlah data
F1	1	5001
	2	5010
	3	6487
F2	1	5075
	2	5096
	3	5039
	4	9132
E1	1	5275
	2	5013
	3	7214
E2	1	5125
	2	5015
	3	7823
FG	1	5025
	2	5064
	3	5015
	4	5010
	5	5003
	6	5012
	7	5014
	8	5518
EG	1	8118
	2	5017
	3	5086
	4	5014
	5	5003
	6	7228

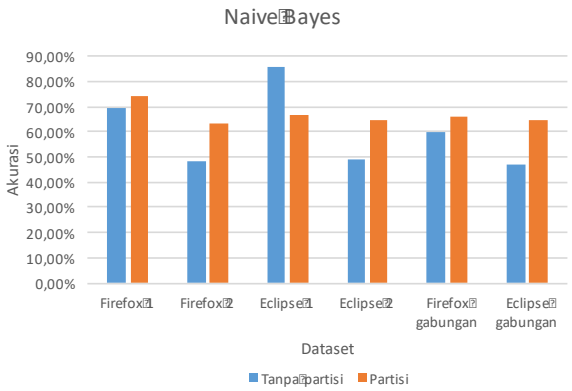
Setelah mendapatkan partisi untuk tiap dataset, selanjutnya dilakukan klasifikasi menggunakan 4 metode klasifikasi yaitu Naive Bayes, *decision tree*, *random forest*, dan K-NN. Hasil dari tiap partisi tersebut akan dirata-rata untuk mendapatkan akurasi akhir dari penggunaan metode partisi usulan dan dibandingkan dengan akurasi tanpa partisi.

4.3. Analisa Hasil

Analisa hasil dan pembahasan terhadap pengujian tanpa partisi dengan pengujian menggunakan metode partisi dataset. Setelah dilakukan pengujian dari seluruh dataset yang digunakan, dapat dibandingkan akurasi antara pengujian menggunakan partisi dengan pengujian tanpa partisi. Hasil yang didapatkan akan disajikan pada beberapa grafik dibawah. Hasil pertama yang akan dibahas adalah hasil dari metode Naive Bayes yang dapat dilihat pada Gambar 5.

Gambar 5 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 dilakukan partisi menjadi sangat tinggi jika dibandingkan dengan setelah dilakukan partisi. Variasi atribut dataset awal dapat dilihat pada Tabel 4.



Gambar 5 Hasil perbandingan akurasi metode Naïve Bayes

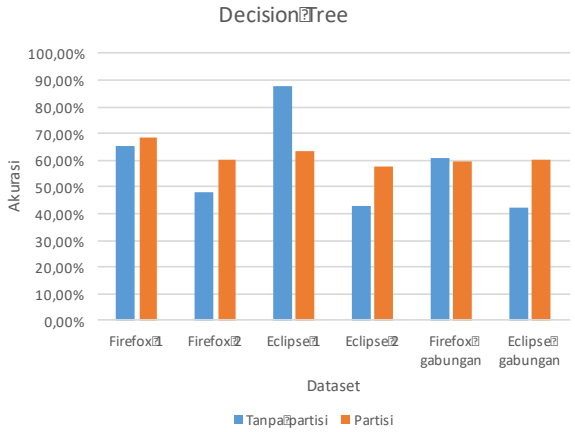
Tabel 2 Variasi dari dataset awal tanpa partisi

	Atribut								
Dataset	1	2	3	4	5	6	7	8	9
F1	8	33	601	3686	28	7	5	192	8
F2	11	35	1196	3326	6	7	6	217	6
E1	6	28	206	4606	5	7	5	28	6
E2	6	31	403	4628	4	7	5	28	6
Fg	12	53	1663	6269	33	7	5	386	8
Eg	6	35	459	8227	5	7	5	29	6

Berdasarkan Tabel 4 atribut yang masih memiliki banyak variasi adalah atribut 3 (*assignee*) dan atribut 4 (*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 partisi 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. Pembahasan selanjutnya adalah metode *decision tree* yang dapat dilihat pada Gambar 6.



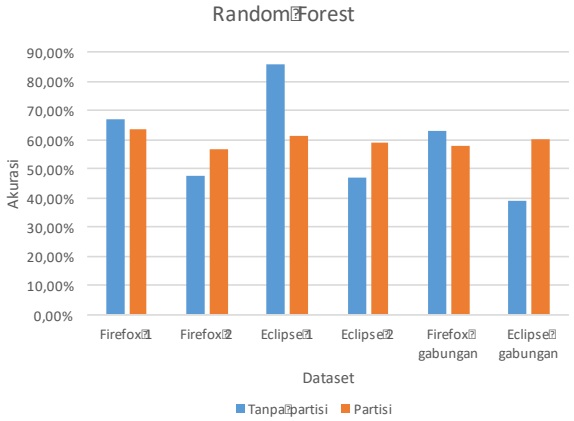
Gambar 6. Hasil perbandingan akurasi metode decision tree

Gambar 6 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 bahwa metode *decision tree* menggunakan perhitungan nilai *entropy* 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*. Pembahasan selanjutnya adalah metode *random forest* yang dapat dilihat pada Gambar 7.

Gambar 7 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%.



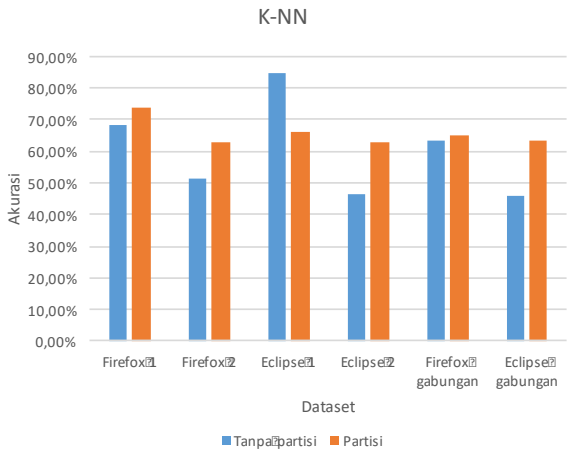
Gambar 7. Hasil perbandingan akurasi metode random forest

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* dari masing-masing atribut. Pembahasan selanjutnya adalah metode K-NN yang dapat dilihat pada Gambar 8.

Gambar 8 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 dilakukan, dapat dilihat bahwa variasi dari atribut pada sebuah dataset akan berpengaruh pada hasil prediksi untuk metode Naive Bayes, *decision tree*, *random forest*, dan K-NN.



Gambar 8. Hasil perbandingan akurasi metode K-NN

5. Kesimpulan

Berdasarkan hasil dari pengujian terhadap seluruh dataset yang dilakukan dapat disimpulkan bahwa metode partisi yang diusulkan dapat meningkatkan akurasi untuk beberapa kasus pengujian. Peningkatan akurasi yang terjadi dipengaruhi oleh variasi dari atribut pada dataset yang digunakan.

Keterbatasan pada penelitian ini adalah pada penentuan *threshold* untuk partisi yang sebatas uji coba nilai 1000, 2000, 3000, 4000, dan 5000. Saran untuk kedepannya agar dapat menggunakan suatu *threshold* yang nilainya lebih pasti.

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.

Daftar Pustaka

Nur, F.A., Siti, R., 2016. Memprediksi Waktu Perbaikan Bug dari Laporan Bug Menggunakan Klasifikasi Random Forest. Jurnal Sistem dan Informatika 11 (2), 156-164.

Norvig, P., Russel, S., 2010. Artificial Intelligence a Modern Approach. Pearson, New Jersey, MA.

Jiawei, H., Kamber, Micheline, 2006. Data Mining: Concepts and Technique. Elsevier, Waltham, MA.

Gall, H.C., Giger, E., Pinzger, M., 2010. Predicting the fix time of bugs, *Proceedings of the 2nd International Workshop on Recommendation System for Software Engineering*, Cape Town, May 4, 52-56.

Abdelmoez, W., Elsalmy, F.M., Kholief, M., 2013. Bug fix-time prediction model using naïve bayes classifier. *International Conference on Computer Theory and Application*, Alexandria, October 13, 167-172.

Alenezi, M., Banitaan, S., 2013. Bug reports prioritization : Which features and classifier to use? *International Conference on Machine Learning and Applications*, Miami, April 4, 112-116.

Gong, L., Versteeg, S., Zhang, H., 2013. Predicting bug fixing-time: an empirical study of commercial software projects. *Proceedings of the 2013 Internatioanal Conference on Software Engineering*, San Francisco, May 18, 1042-1051.

Romi, S. W., 2016. A Systematic Literature Review of Software Defect Prediction: Research Trends, Dataset, Methods and Frameworks. Journal of Software Engineering, 1 (2), 1-16.

Breiman, L., 2001. Random Forest, Machine Learning, 45 (1), 5-32.

Bhuwaneswari, V., 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. *International Conference on Intelligent Computing Applications*, Coimbatore, March 6, 335-339.