KAJIAN KINERJA METODE FUZZY k-NEAREST NEIGHBOR PADA PREDIKS CACAT SOFTWARE

Rimbun Siringoringo

Universitas Methodist Indonesia
Jl. Hang Tuah No. 8 Medan
rimbun.ringo@gmail.com

Abstract

This research examines the model of Fuzzy k-Nearest Neighbor (Fk-NN) to predict software defects. Software defects based on three dataset, CM1, JM1 and KC1. Datasets are derived from the promise repository. Feature selection and data normalization applied at the stage of data pre-process. Feature selection based on Correlation-Based Feature Selection (CFS), and data normalized based on min-max method. This research applies Fk-NN method to predict software defects. Performance prediction consisted of five aspects: accuracy, sensitivity and precision. Testing techniques applied in this study is 10-fold Cross Validation. To get the best performance, we applying the varied value of k and m. Range for K value is [1, 9] and m values [1.0, 1.9]. The best performance for CM1 dataset was obtained on a combination of value [k, m] = [9, 1.0]. For JM1 dataset, the best performance was obtained on a combination of value [k, m] = [9, 1.9]. For KC1 dataset, the best performance was obtained on a combination of value [k, m] = [9, 1.5]. The results of this study indicate that the results and performance classification with Fk-NN method highly depends on the parameters k and m, the selection of appropriate parameters will yield the expected performance

Keywords : *fuzzy k-nearest neighbor, correlation-based feature selection, cacat software*

I. PENDAHULUAN

Cacat atau *fault* pada perangkat lunak adalah kesalahan aktual yang terdapat pada kode program yang mengakibatkan sebuah perangkat lunak tidak bekerja sebagaimana mestinya. Cacat seringkali tidak terdeteksi pada tahap *testing* dan *debugging* perangkat lunak, melainkan pada saat diimplementasikan pada sistem operasi tertentu. Keberadaan cacat tentu saja menjadi hal yang tidak diinginkan, karena dapat menyebabkan kegagalan perangkat lunak, meningkatkan biaya perawatan, mengabaikan aspek keselamatan manusia (Devi *et al*, 2011).

Pengujian software merupakan tahapan yang krusial untuk memastikan kelayakan dan kualitas sebuah perangkat lunak tertentu. Tahapan ini bertujuan untuk menemukan sebanyak mungkin *flaws* pada software sebelum di lanjutkan ke tahap produksi (Shuai, 2013). Metode Pengujian perangkat lunak yang umum diterapkan adalah metode konvensional yaitu menerapkan sistem *tracking failure* atau menelusuri kesalahan dengan sistem *test case* atau kasus perkasus. Metode ini membutuhkan biaya yang mahal dan membutuhkan hampir 50 % dari jadwal pengembangan perangkat lunak (Saifuddin, 2015).

II. PENELITIAN TERKAIT

Machine learning atau mesin pembelajaran sebagai bagian dari *data mining* memberikan banyak sumbangsih pada penelitian terhadap cacat siftware Penelitian terhadap prediksi cacat perangkat lunak merupakan salah satu kajian penelitian yang telah banyak dilakukan oleh peneliti. Naidu & Geethanjali (2013) menerapkan algoritma ID3 untuk mengklasifikasi cacat software. Penelitian tersebut difokuskan pada penemuan banyaknya jumlah cacat dalam rangka mereduksi waktu dan biaya. Kualitas model yang diterapkan diukur berdasarkan kepadatan cacat atau *defect density*. Shan *et al* (2012) membangun model prediksi cacat perangkat lunak berbasis *Support Vector Machine* (SVM). Pada penelitian tersebut SVM dikombinasikan dengan *Locally Linear Embedding* (LLE) untuk menghilangkan redundansi data . Parameter SVM dioptimasi dengan teknik *grid search* untuk meningkatkan kinerja SVM. Hasil penelitian menunjukkan bahwa Model LLE-SVM menghasilkan performa dan akurasi yang lebih baik jika dibandingkan dengan model SVM saja. Song *et al* (2010) melakukan kajian pengujian perangkat lunak berbasis *machine learning* yaitu *defect-pronenes*. Pada penelitian tersebut dibangun Skema pengujian dua tingkat yakni evaluasi kemudian prediksi. Skema yang ditawarkan lebih efektif. Singh & Verma (2014) menerapkan metode *Clustering Base Classification* (CBC) untuk memprediksi cacat software pada NASA *benchmak dataset*. Metode tersebut tebukti memiliki *superior probability detect* sebesar 83.3 %.

Pada penelitian ini, algoritma *Fuzzy K-Nearest Neighbor* (FKNN) digunakan untuk membangun model prediksi cacat perangkat lunak secara otomatis berbasis *machine learning*. FKNN memiliki banyak kelebihan jika dibandingkan dengan metode *machine learning* standar seperti *K-Nearest Neighbor* (KNN) atau *Support Vector Machine* (SVM). FK-NN merupakan salah satu algoritma klasifikasi yang terdapat pada teknik Machine Learning (ML). FK-NN merupakan hasil pengembangan dari algoritma sebelumnya yaitu K-Nearest Neighbor (K-NN). Metode FK-NN menggabungkan prinsip *fuzzy* ke dalam metode K-NN (Keller *et al*, 1985), sehingga FK-NN dapat mengatasi sifat ambiguitas data. Dengan FK-NN, setiap *instance* memiliki derajat keanggotaan pada setiap kelas sehingga akan memberikan kekuatan pada *instance* tersebut. Penelitian terdahulu yang menerapkan FK-NN memberi kesimpulan bahwa FK-NN berhasil memperbaiki kinerja dan performa dari K-NN. FK-NN mampu menangani tingkat kepadatan (*density*) data latih yang tidak merata (Shang *et al*, 2010), dimana kondisi tingkat kepadatan data latih yang tidak merata dapat menurunkan tingkat presisi atau akurasi hasil klasifikasi. Hal ini diperkuat oleh Derrac *at al* (2014) dalam penelitiannya yang menyatakan bahwa FK-NN memiliki akurasi yang lebih tinggi hampir pada semua masalah klasifikasi. Chen *et al* (2013) menerapkan FKNN dalam memprediksi penyakit parkinson. Dengan metode 10-*fold crocc validation* diperoleh akurasi sebesar 96,07 %. Bakry *et al* (2015) menerapkan metode FK-NN pada masalah klasifikasi *big data*. Dengan pendekatan MapReduce diperoleh performa klasifikasi yang lebih baik. Shang *et al* (2006) menerapkan FKNN pada

masalah klasifikasi teks. Dari hasil ekperimen tersebut diperoleh bahwa metode FKNN memiliki hasil yang lebih efektif dan lebih *feasible* jika dibandingkan dengan KNN yang klasik.

Pada penelitian ini akan dibangun sebuah model prediksi cacat perangkat lunak berbasis metode FKNN. Keefektifan model yang diusulkan dalam penelitian ini akan diukur berdasarkan pendekatan *accuracy*, *sensitivity*, *specificity* dan AUC dengan objek ekperimen adalah *software metrics data* set yang diperoleh data NASA dataset repository.

III. TINJAUAN PUSTAKA

3.1. Konsep K-Nearest Neighbor

Menurut Han *et al* (2012), metode *K-Nearest Neighbor* (K-NN) merupakan metode klasifikasi klasik yang paling sederhana. K-NN melakukan klasifikasi terhadap objek berdasarkan data pembelajaran yang jaraknya paling dekat (*nearest*) dengan objek yang duji. Metode K-NN menggunakan prinsip ketetanggaan *(neighbor)* untuk memprediksi *class* yang baru. Jumlah tetangga yang dipakai adalah sebanyak K tetangga.

Prosedur K-NN dapat dijelaskan sebagai berikut. Misalkan terdapat sebuah data uji *z = (x', y')* dan data latih *d = (x', x),* dengan *x'* ada atribut data uji, dan *y'* adalah label *class* data uji yang belum diketahui. Selanjutnya dihitung jarak antara data uji ke setiap data latih, kemudian mengambil K tetangga terdekat pertama. Setelah mengambil K tetangga terdekat pertama kemudian dihitung jumlah data yang mengikuti kelas yang ada dari K tetangga tersebut. Kelas dengan data terbanyak yang mengikutinya menjadi kelas pemenang yang diberikan sebagai label kelas pada data uji y' (Han *et al* , 2012).

Algoritma K-Nearest Neighbor :

1. z = (x',y'), adalah data uji dengan vektor x' dan label kelas y' yang belum diketahui
2. Hitung jarak eucledian d(x',x), jarak diantara data uji z ke setiap vektor data latih menggunakan persamaan (2.1), kemudian simpan dalam D

$$d_i = \sqrt{\sum_{i=1}^{p}(x'_i - x_i)^2} \qquad (2.1)$$

3. Pilih $D_z \subseteq D$, yaitu K tetangga terdekat dari z
4. Berikan label kelas dari data latih tetangga terdekat yang jumlahnya mayoritas atau terbesar menggunakan persamaan (2.2)
5.

$$y' = \arg max \sum (x_i y_i) \in D_z \qquad (2.2)$$

3.2. Konsep Fuzzy K-Nearest Neighbor

Pada teori himpunan *fuzzy*, sebuah data dikatakan memiliki nilai keanggotaan pada setiap kelas. Hal ini bisa diartikan sebuah data dapat dimiliki oleh kelas yang berbeda dengan nilai derajat keanggotaan pada interval [0,1]. Teori himpunan *fuzzy* yang menggeneralisasi teori *k-Nearest Neighbor* klasik dengan pendefinisian nilai keanggotaan sebuah data pada masing-masing kelas diperoleh dengan persamaan (2.3) (Keller, 1985).

Nilai keanggotaan suatu data pada kelas sangat dipengaruhi oleh jarak data itu ke tetangga terdekatnya, semakin dekat ke tetangganya maka semakin besar nilai keanggotaan data tersebut pada kelas tetangganya, begitu pula sebaliknya. Jarak tersebut diukur dengan N dimensi atau fitur data. Apabila nilai *m* adalah 2 maka kontribusi masing-masing titik tetangga diberatkan dengan kebalikan dari jaraknya dengan titik yang diklasifikasikan, Jika m meningkat, tetangga adalah rata-rata yang berbobot dan jarak mereka dengan yang diklasifikasikan kurang berpengaruh.

Algoritma Fuzzy K-Nearest Neighbor :

1. Normalisasikan data menggunakan nilai terbesar dan terkecil data pada setiap fitur. Normalisasi data ditentukan berdasarkan persamaan (2.3).

$$x' = \frac{x - min_a}{max_a - min_a} \qquad (2.3)$$

Dengan *x'* adalah nilai hasil normalisasi, *x* adalah nilai yang asli, min_a adalah nilai minimim pada fitur a dan max_a adalah nilai maksimum pada fitur a

2. Cari K tetangga terdekat untuk data uji x menggunakan persamaan (2.1).
3. Hitung nilai keanggotaan *u*(*x*, *yi*) menggunakan persamaan (2.4) untuk setiap i, dimana 1 ≤ i ≤ C.

$$\mu_i(x) = \frac{\sum_{j=1}^{k}\mu_{ij}\left(\frac{1}{|x - x_j|^{2/m-1}}\right)}{\sum_{j=1}^{k}\left(\frac{1}{|x - x_j|^{2/m-1}}\right)} \qquad (2.4)$$

dengan :

$\mu_{i(x)}$: nilai keanggotaan data *x* ke kelas *ci*
μ_{ij} : nilai keanggotaan data tetangga
m : bobot pangkat (*weight exponent*) yang besarnya *m* > 1.
k : jumlah tetangga terdekat yang digunakan

4. Ambil nilai terbesar menggunakan persamaan (2.5) untuk semua i, dengan 1 ≤ i ≤ C, C adalah jumlah kelas.

$$v = \arg max\ (\mu_i(x)) \qquad (2.5)$$

5. Berikan label kelas v ke data uji *x* yaitu y_i

3.3. Seleksi Fitur

Banyak penelitian yang menelaah tentang peran penting seleksi fitur pada masalah klasifikasi. Tidak ketinggalan pada masalah yang berhubungan dengan cacat perangkat lunak, seleksi fitur menjadi hal yang sering diterapkan. Beberapa penelitian penerapan seleksi fitur pada masalah cacat perangkat lunak yaitu Khoshgoftaar dan Gao (2009) menerapkan metode *Random Under Sampling* (RUS), Wahono dan Suryana (2014) menerapkan metode *Genetic Feature Selection*, Agarwal dan Tomar (2014) menerapkan metode *F-Score*.

3.4. *Correlation-Based Feature Selection*

Metode *Correlation-based Feature Selection* (CFS) merupakan salah satu algoritma yang banyak digunakan dalam rangka seleksi fitur pada data dengan dimensi fitur

yang besar. Teknik ini termasuk ke dalam kategori seleksi fitur yang mengevaluasi subset dari atribut. Teknik ini mempertimbangkan kegunaan atribut individual untuk memprediksi *class* dan juga level inter-korelasi di antara mereka.

CFS mengharuskan atribut numerik didiskritkan terlebih dahulu sebelum menggunakan symmetrical uncertainty untuk mengestimasi derajat asosiasi antara dua fitur diskrit. Pada Weka, teknik ini tersedia dalam class *weka.attribeteSelection .CfsSubsetEval* dan dipasangkan dengan metode pencarian *Forward Selection.*

3.5. Matrik konfusi

Matrik konfusi atau c*onfusion matrix* adalah salah satu metode yang sering digunakan untuk mengevaluasi performa atau kinerja klasifikasi. Matrik konfusi didasarkan pada tabel yang menyatakan jumlah data uji yang benar diklasifikasikan dan jumlah data uji yang salah diklasifikasikan (Tan *et al,* 2012). Tabel 2.1 merupakan tabel Matrik konfusi untuk klasifikasi biner.

Tabel 1. *Confusion matrix* untuk klasifikasi biner

		Kelas prediksi		
		yes	no	Total
Kelas Aktual	yes	TP	FN	P
	no	FP	TN	N
	Total	P'	N'	P+N

Keterangan :

True Positive (TP)	:	Jumlah dokumen dari kelas 1 yang benar diklasifikasikan sebagai kelas 1.
True Negative (TN)	:	Jumlah dokumen dari kelas 0 yang benar diklasifikasikan sebagai kelas 0
False Positive (FP)	:	Jumlah dokumen dari kelas 0 yang salah diklasifikasikan sebagai kelas 1.
False Negative (FN),	:	Jumlah dokumen dari kelas 1 yang salah diklasifikasikan sebagai kelas 0.

Selanjutnya Han *et al* (2012), berdasarkan tabel di atas dapat dibuat beberapa formula untuk menentukan performa klasifikasi atau prediksi yaitu *accuracy*, *error rate*, *sensitifity*, *sfesifity* dan *precision* sebagaimana disajikan pada tabel 2.

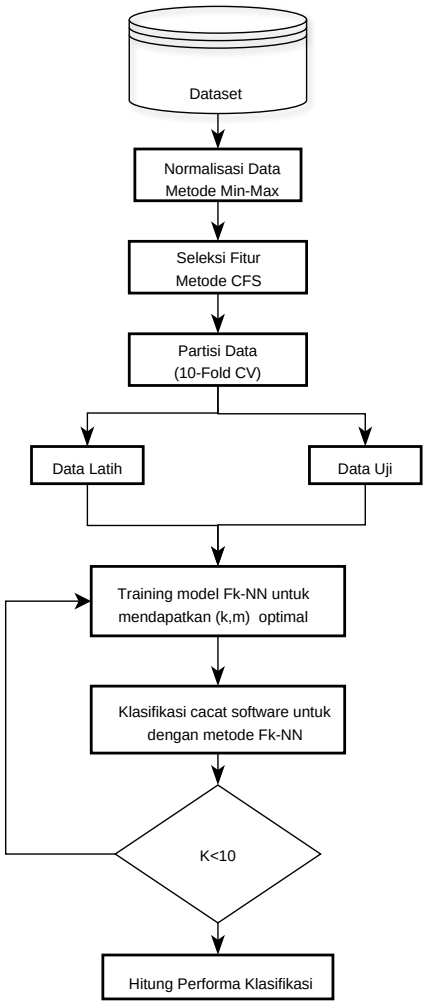
Tabel 2. Pengukuran performa prediksi dan klasifikasi

Kriteria	Formula	
Accuracy, recognition rate	$\frac{TP+TN}{P+N}$	(2.3)
Error rate, misclassification rate	$\frac{FP+FN}{P+N}$	(2.4)
Sensitivity, true positive rate	$\frac{TP}{P}$	(2.5)
Specificity, true negative rate	$\frac{TN}{N}$	(2.6)
precision	$\frac{TP}{TP+FP}$	(2.7)

IV. METODOLOGI

4.1. Prosedur Penyelesaian Masalah

Prosedur penyelesaian masalah yang diusulkan pada penelitian ini disajikan pada gambar 1 berikut.



Gambar 1. Model yang diusulkan

Pada penelitian ini, terdapat beberapa tahapan yang diusulkan pada model prediksi cacat software.

1) Seleksi fitur dataset dengan metode *Correlation-Based Feature Selection* (CFS). Seleksi fitur dataset dilakukan secara dinamis yaitu sebanyak fold partisi.
2) Normalisasi data set dengan metode Min-Max untuk mendapatkan nilai
3) Partisi dataset menjadi 10 partisi, masing-masing 10 partisi untuk training set dan 10 partisi untuk test set. Partisi ini dibangun dengan metode 10-fold cross validation.
4) Proses training model Fk-NN
5) Proses prediksi label cacat software berdasarkan dataset hasil pemrosesan awal CFS. Prediksi label ini dilakukan dengan metode FKNN. Penerapan metode FKNN dilakukan dengan penentuan parameter k yang bervariasi. Parameter k di cobakan adalah dengan range [1, 10]

4.2. *Dataset*

Pada penelitian ini digunakan tiga jenis dataset *software defec* yaitu MC1, JM1 dan KC1. Dataset tersebut bersumber dari *dataset repository* NASA MDP. Dataset

tersebut terdiri dari 22 atribut. Detail atribut tesebut dapat ditampilkan pada tabel berikut ini :

Tabel 3. Gambaran umum dataset

No	Dataset	Jlh Atribut	Bahasa	Jlh Instance
1	CM1	22	C	**490**
2	JM1	22	C++	**10880**
3	KC1	22	C	**2100**

Tabel berikut ini adalah daftar atribut dataset. Keseluruhan dataset terdiri atas 22 atribut, termasuk 1 atribut kelas.

Tabel 4. Daftar atribut dataset

No Atrib.	Atribut	Tipe data
1	loc	numerik
2	v(g)	numerik
3	ev(g)	numerik
4	iv(g)	numerik
5	n	numerik
6	v	numerik
7	l	numerik
8	d	numerik
9	i	numerik
10	e	numerik
11	b	numerik
12	t	numerik
13	lOCode	numerik
14	lOComment	numerik
15	lOBlank	numerik
16	locCodeAndComment	numerik
17	uniq_Op	numerik
18	uniq_Opnd	numerik
19	total_Op	numerik
20	total_Opnd	numerik
21	branchCount	numerik
22	defects	True, false

V. HASIL DAN PEMBAHASAN PENELITIAN

Pada tahapan pre-proses data, tahapan seleksi fitur (*feaure selection*) dan Trasformasi data.

5.1. Seleksi Fitur

Seleksi fitur bertujuan untuk mereduksi dimensionalitas dataset dengan menyaring fitur-fitur tertentu. Dengan menerapkan algoritma CFS, dataset MC1, JM1 dan KC1 direduksi menjadi dataset dengan atribut yang lebih kecil. hasil reduksi untuk keseluruhan dataset di atas disajikan pada tabel berikut ini. Untuk dataset CM1, hasil seleksi fitur terdiri atas 8 atribut yaitu fitur [1,4,9,14,15,17,18]

Tabel 5. Hasil seleksi fitur dataset CM1

No Atribut	Fitur terpilih	Tipe data
1	loc	numerik
4	iv(g)	numerik
9	i	numerik
14	lOComment	numerik
15	lOBlank	numerik
17	uniq_Op	numerik
18	uniq_Opnd	numerik
22	defects	True, false

Untuk dataset JM1, hasil seleksi fitur terdiri atas 8 atribut yaitu fitur [1,2,3,4,9,14,15,16]

Tabel 6. Hasil seleksi fitur dataset JM1

No Atribut	Fitur terpilih	Tipe data
1	loc	numerik
2	v(g)	numerik
3	ev(g)	numerik
4	iv(g)	numerik
9	i	numerik
14	lOComment	numerik
15	lOBlank	numerik
16	locCodeAnd Comment	numerik
22	defects	True, false

Untuk dataset KC1, hasil seleksi fitur terdiri atas 8 fitur yaitu fitur [6,8,9,13,14,15,17,21]

Tabel 7. Hasil seleksi fitur dataset KC1

No Atribut	Fitur terpilih	Tipe data
6	v	numerik
8	d	numerik
9	i	numerik
13	lOCode	numerik
14	lOComment	numerik
15	lOBlank	numerik
17	uniq_Op	numerik
21	branchCount	numerik
22	defects	True, false

5.2. Normalisasi

Normalisasi bertujuan untuk mentransformasi data dalam rangka mengindari munculnya data numerik yang terlalu besar dan terlalu kecil. Keberadaan data numerik yang terlalu besar dan terlalu kecil menimbulkan kesulitan dalam perhitungan matematika. Dengan menerapkan metode normalisasi *min-max*, pada tabel berikut ini disajikan hasil normalisasi dataset. Fitur **loc** pada dataset CM1 dijadikan sebagai sample hasil normalisasi tersebut.

Tabel 8. Normalisasi dataset fitur **loc**

Instance	**Loc original**	**Loc normalisasi**
1	1	0,0000
2	24	0,0161
3	7	0,0000
4	25	0,0645
5	361	0,7258
6	76	0,1452
7	24	0,0323
8	18	0,0484
9	32	0,0161
10	17	0,0323
48	28	0,0323
49	8	0,0000
50	82	0,1452

5.3. Partisi dataset

Teknik pengujian yang diterapkan pada penelitian ini adalah teknik 10-*fold Cross Validation*.

Dataset CM1

Dataset CM1 terdiri atas 490 *record*, setiap fold terdiri atas 49 record data uji dan 441 data training.

Tabel 9. Skema k-fold cross validation dataset CM1

Fold	Data Latih	Data Uji	Total
Fold-1	441	49	490
Fold-2	441	49	490
Fold-3	441	49	490
Fold-4	441	49	490
Fold-5	441	49	490
Fold-6	441	49	490
Fold-7	441	49	490
Fold-8	441	49	490
Fold-9	441	49	490
Fold-10	441	49	490

Dataset JM1

Dataset JM1 terdiri atas **10880** *record*, setiap fold terdiri atas 1088 record data uji dan 9792 data training.

Tabel 10. Skema k-fold cross validation dataset JM1

Fold	Data Latih	Data Uji	Total
Fold-1	9792	1088	**10880**
Fold-2	9792	1080	**10880**
Fold-3	9792	1080	**10880**
Fold-4	9792	1080	**10880**
Fold-5	9792	1080	**10880**
Fold-6	9792	1088	**10880**
Fold-7	9792	1088	**10880**
Fold-8	9792	1088	**10880**
Fold-9	9792	1088	**10880**
Fold-10	9792	1088	**10880**

Dataset KC1

Dataset KM1 terdiri atas **2100** *record*, setiap fold terdiri atas 210 record data uji dan 1898 data latih.

Tabel 11. Skema k-fold cross validation dataset KC1

Fold	Data Latih	Data Uji	Total
Fold-1	1898	210	**2100**
Fold-2	1898	210	**2100**
Fold-3	1898	210	**2100**
Fold-4	1898	210	**2100**
Fold-5	1898	210	**2100**
Fold-6	1898	210	**2100**
Fold-7	1898	210	**2100**
Fold-8	1898	210	**2100**
Fold-9	1898	210	**2100**
Fold-10	1899	210	**2100**

5.4. Pengujian performa klasifikasi

Pada pengujian ini, performa klasifikasi yang diuji terdiri atas *accuracy*, *sensitivity* dan *precision*. Pengujian ini didasarkan pada kombinasi parameter *k* dan *m*, dimana nilai *k* adalah [1,3,5,7, 9] dan nilai *m* adalah [1, 1.2, 1.5, 1.7, 1.9]

Dataset CM1

Tabel 13. Performa Klasifikasi dataset CM1

Parameter		**Performa metrik**		
k	***m***	***accuracy***	***sensitivity***	***precision***
1	1.0	0,9597	0,9741	0,9821
1	1.2	0,9597	0,9741	0,9821
1	1.5	0,9597	0,9741	0,9821
1	1.7	0,9597	0,9741	0,9821
1	1.9	0,9597	0,9741	0,9821
3	1.0	0,9578	0,9637	0,9910
3	1.2	0,9578	0,9656	0,9888
3	1.5	0,9578	0,9698	0,9843
3	1.7	0,9578	0,9698	0,9843
3	1.9	0,9578	0,9698	0,9843
5	1.0	0,9679	0,9701	0,9955
5	1.2	0,9679	0,9721	0,9933
5	1.5	0,9598	0,9719	0,9843
5	1.7	0,9618	0,9720	0,9865
5	1.9	0,9618	0,9720	0,9865
7	1.0	0,9679	0,9700	0,9955

7	1.2	0,9699	0,9721	0,9955
7	1.5	0,9618	0,9720	0,9865
7	1.7	0,9598	0,9719	0,9843
7	1.9	0,9618	0,9720	0,9865
9	1.0	**0,9699**	0,9701	**0,9977**
9	1.2	**0,9699**	0,9721	0,9955
9	1.5	0,9618	0,9720	0,9865
9	1.7	0,9618	0,9720	0,9865
9	1.9	0,9678	**0,9742**	0,9910

Dari tabel 12 di atas, nilai *accuracy* tertinggi pada dataset CM1 diproleh sebesar 0.9699 yaitu pada kombinasi [k,m] = [7, 1.2], [9,1.0] dan [9, 1.2]. Nilai *sensitivity* tertinggi adalah 0,9742 pada kombinasi [k,m] =[9, 1.9]. Nilai *precision* tertinggi diproleh sebesar 0,9977 pada kombinasi [k,m]=[9,1.0]. Secara keseluruhan, performa terbaik diperoleh pada kombinasi nilai [k,m] =[9, 1.0]

Dataset JM1

Tabel 15. Performa Klasifikasi dataset JM1

Parameter		**Performa metrik**		
k	***m***	***accuracy***	***sensitivity***	***precision***
1	1.0	0,7899	0,9001	0,8319
1	1.2	0,7899	0,9001	0,8319
1	1.5	0,7899	0,9001	0,8319
1	1.7	0,7899	0,9001	0,8319
1	1.9	0,7899	0,9001	0,8319
3	1.0	0,8299	0,9034	0,8836
3	1.2	0,8297	0,9031	0,8837
3	1.5	0,8288	0,9003	0,8857
3	1.7	0,8279	0,9021	0,8824
3	1.9	0,8300	0,9048	0,8821
5	1.0	0,8356	0,8994	0,8963
5	1.2	0,8353	0,8991	0,8963
5	1.5	0,8344	0,8968	0,8979
5	1.7	0,8349	0,9020	0,8922
5	1.9	0,8378	0,9058	0,8917
7	1.0	0,8499	0,9001	0,9155
7	1.2	0,8496	0,8998	0,9155
7	1.5	0,8476	0,8965	0,9170
7	1.7	0,8474	0,9034	0,9080
7	1.9	0,8494	**0,9063**	0,9072
9	1.0	0,8511	0,8969	0,9213
9	1.2	0,8508	0,8966	0,9213
9	1.5	0,8501	0,8948	**0,9225**
9	1.7	0,8514	0,9035	0,9132
9	1.9	**0,8533**	0,9060	0,9129

Dari tabel 15 di atas, nilai *accuracy* tertinggi pada dataset JM1 sebesar 0,853 pada kombinasi [k,m] = [9,1.9]. Nilai *sensitivity* tertinggi diproleh sebesar 0,9063 yaitu pada kombinasi [k,m] =[7, 1.9]. Nilai precision tertinggi diproleh sebesar 0,9225 yaitu pada kombinasi [k,m]=[9,1.5]. Secara keseluruhan, performa terbaik pada dataset JM1 diperoleh pada kombinasi nilai [k,m] =[9, 1.9]

Dataset KC1

Tabel 17. Performa Klasifikasi dataset KC1

Parameter		**Performa metrik**		
k	***m***	***accuracy***	***sensitivity***	***precision***
1	1.0	0,8023	0,9566	0,8026
1	1.2	0,8023	0,9566	0,8026
1	1.5	0,8023	0,9566	0,8026
1	1.7	0,8023	0,9566	0,8026
1	1.9	0,8023	0,9566	0,8026
3	1.0	0,8686	0,9588	0,8828
3	1.2	0,8682	0,9582	0,8828
3	1.5	0,8663	0,9542	0,8845
3	1.7	0,8696	0,9556	0,8873
3	1.9	0,8691	0,9550	0,8873
5	1.0	0,9180	0,9601	0,9423
5	1.2	0,9175	0,9590	0,9428
5	1.5	0,9218	0,9593	0,9479
5	1.7	0,9213	0,9593	0,9473
5	1.9	0,9199	0,9582	0,9467
7	1.0	0,9256	0,9594	0,9523
7	1.2	0,9265	0,9590	0,9540
7	1.5	0,9294	0,9587	0,9579
7	1.7	0,9298	0,9592	0,9579
7	1.9	0,9303	0,9597	0,9579
9	1.0	0,9436	**0,9628**	0,9708
9	1.2	0,9445	0,9619	0,9731
9	1.5	**0,9479**	0,9600	**0,9793**
9	1.7	0,9474	0,9605	0,9781
9	1.9	0,9474	0,9610	0,9776

Dari tabel di atas, nilai *accuracy* tertinggi diproleh sebesar 0,9479 yaitu pada kombinasi [k,m] =[1, 1.2]. Nilai sensitivity tertinggi diproleh sebesar 0,9628 yaitu pada kombinasi [k,m] =[9, 1.0]. Nilai precision tertinggi diproleh sebesar 0,9793 yaitu pada kombinasi [k,m]=[9,1.5]. Secara keseluruhan, performa terbaik diperoleh pada kombinasi nilai [k,m] =[9, 1.5]

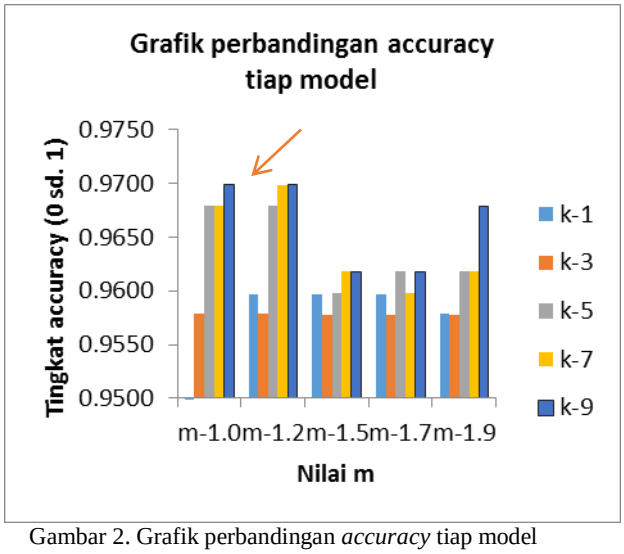
Pada tabel berikut disajikan performa maksimum dan minimum untuk setiap data set

Tabel 14. Nilai performa maksimum, minimum dan rata-rata pada dataset

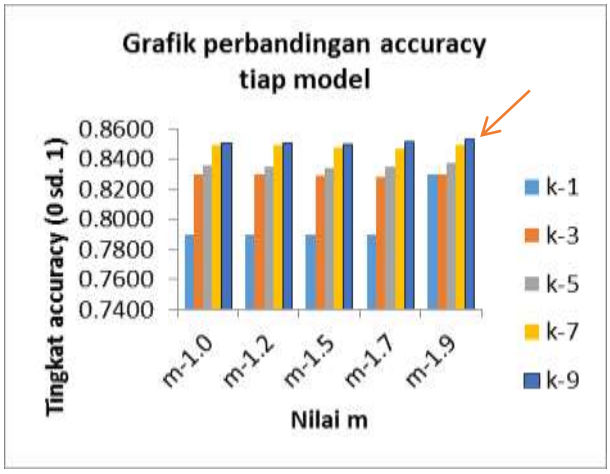
Dataset	**Performa**	***accuracy***	***sensitivity***	***precision***
CM1	MAX	0,9699	0,9742	0,9977
	MIN	0,9578	0,9637	0,9821
KC1	MAX	0,8533	0,9063	0,9225

	MIN	0,7899	0,8948	0,8319
	MAX	0,9479	0,9628	0,9793
KC2	MIN	0,8023	0,9542	0,428

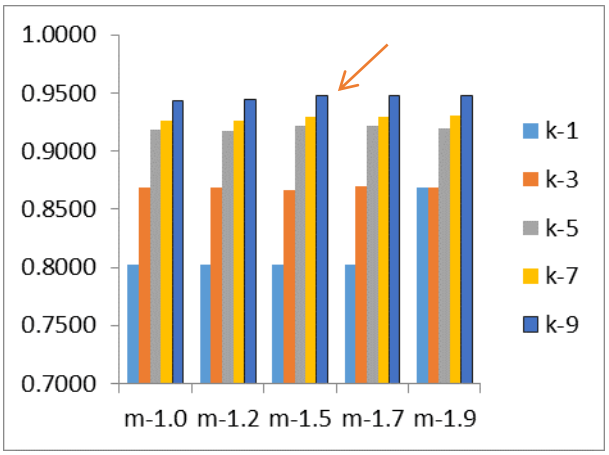
Pada gambar berikut, ditampilkan grafik perbandingan performa *accuracy* untuk tiap model. Untuk dataset CM1, *accuracy* terbaik diperoleh pada pasangan nilai [k,m] =[9, 1.0]. Untuk dataset JM1, *accuracy* terbaik diperoleh pada pasangan nilai [k,m] =[9, 1.9]. Untuk dataset KC1, *accuracy* terbaik diperoleh pada pasangan nilai [k,m] =[9, 1.5].



Gambar 2. Grafik perbandingan *accuracy* tiap model dataset CM1



Gambar 3. Grafik perbandingan *accuracy* tiap model dataset JM1



Gambar 4. Grafik perbandingan *accuracy* tiap model dataset KC1

VI. KESIMPULAN

Penelitian ini melakukan penelitian terhadap prediksi cacat software berdasis metode Fuzzy k-Nearest Neighbor (Fk-NN). Berdasarkan percobaan yang dilaksanakan diperoleh kesimpilan bahwa performa Fk-NN sangat dipengaruhi oleh nilai *k* dan *m* yang diterapkan. Nilai k dan m tidak dapat berdiri sendiri dalam memperbaiki performa yang diharapkan, nilai k dan m merupakan yang saling membutuhkan satu-sama lain

DAFTAR PUSTAKA

Devi, C., A., Surendiran, B. & Kannammal, K., E. 2011. A Study of feature selection methods for Software fault prediction model. *Proceedings of 1st International conference on network, intelligence and computing technology* : pp. 1-5.

Shuai, B. Li, H., Li, M., Zhang, Q. & Tang, C. 2013. *Software Defect Prediction Using Dynamic Support Vector Machine*. Proceeding of 9th International Conference on Computational Intelligence and Security : 260-264

Shan, C., Chen, B., Hu., H., Xue, J. & Li, N. 2012. *Software defect prediction model based on LLE and SVM.*

Saifudin, A. & Wahono, R., S. 2015. *Penerapan Teknik Ensemble untuk Menangani Ketidakseimbangan Kelas pada Prediksi Cacat SoftwareJournal of Software Engineering*. Journal of Software Engineering. 1(1) : 28-39

Naidu, M., S. & Geethanjali, N. *Classification Of Defects In Software Using Decision Tree Algorithm*. International Journal of Engineering Science and Technology (IJEST). 5(6) : 1332-1340

Agarwal, S. & Tomar, D. 2014. *A Feature Selection Based Model for Software Defect Prediction*. International Journal of Advanced Science and Technology. .65 (2014) : pp.39-58.

Singh, P., Verma, S., 2014. *An Efficient Software Fault Prediction Model using Cluster based Classification* . International Journal of Applied Information Systems. 7(3) : 35-42.

Song, Q., Jia, Z., Shepperd, M., Ying, S & Liu, J. 2010. *A General Software Defect-Proneness*

Prediction Framework. IEEE transactions on software engineering, 37 (3) : 356-370.

Keller, M. James, Michael R Gray, James A. Givens. 1985. *A Fuzzy K-Nearest Neighbor.IEEE Transactions On Sistem, Man And Cybernetics***,** *Vol. SMC-15 NO 4*

Shang, W., Huang, H., Zhu, H., Lian, W. & Wang , Z. 2010. *An improve kNN algorithm-FK-NN. Springer* :741-746.

Derrac at al (2014). Joaqu´ın Derrac, J., Chiclana, F., Garcıa, S., Francisco Herrera, F. 2014. *Evolutionary Fuzzy K-Nearest Neighbors Algorithm using Interval-Valued Fuzzy Sets*. Information Sciences: 1-28

Chen, H.,L, Huang, C., C., Xin-Gang Yu, X., G., Sun, X., Wang, G & Wang., S., J. 2013. *An efficient diagnosis system for detection of Parkinson's disease using fuzzy k-nearest neighbor approach*. Expert Systems with Applications. 40 (2013) 263–271

Bakry, M., E., Safwat., S. & Hegaji, O. 2015. *Big Data Classification using Fuzzy K-Nearest Neighbor*. International Journal of Computer Applications . *132 (10) : 0975 – 8887*

Article I. Han, J., Kamber, M. & Pei, J. 2012. *Data mining tecniques and concepts*. Morgan Kaufman publisher. Watham : USA

Wahono, R., S, Suryana, N, 2014. *Genetic Feature Selection for Software Defect Prediction. Advanced Science Letters*. 20 (2014) : 239–244