Jurnal Maklumatika

Vol. 4, No. 1, Juli 2017 ISSN : 2407-5043

PREDIKSI SOFTWARE DEFECT PENGGUNAAN OPTIMASI GENETIC ALGORITHM DENGAN ADOPSI MODEL MLP DAN SVM

Puput irfansyah[1], **Syamsiah**[2], **Agus darmawan**[3]

[1,2,3] Program Studi Teknik Informatika, Universitas Indraprasta PGRI

Jl.Nangka No 58 Tanjung Barat Jagakarsa Jakarta Selatan, 12530

Email : irfandot@gmail.com[1], ncham.unindra08@gmail.com[2] , agus.darmawan@ymail.com[3]

Abstrak

Dalam Pengembangan teknologi perusahaan atau institusi diberbagai bidang memerlukan software untuk membantu proses bisnis mereka agar dapat berjalan dengan cepat, tepat, efektif dan efisien. Tentunya software yang digunakan haruslah mempunyai standar kualitas yang baik agar tujuan dari perusahaan atau institusi tersebut dapat terpenuhi. Maka dari itu diperlukan *software-software* yang tidak memiliki kesalahan / error (*defect*). Kesalahan biasanya disebabkan oleh kesalahan manusia dalam melakukan pembuatan model untuk para pengembangan software. Dari beberapa penelitian sebelumnya model yang paling baik untuk melakukan prediksi *software defect* adalah MLP dan SVM. Pada Penelitian ini melakukan pemilihan model terhadap adalah MLP dan SVM dilakukan pengujian dengan Kurva ROC dan Confusion Matrix dan dilakukan pula Optimasi yang di gunakan *Genetic Algorithm* akan diterapkan untuk pemilihan variable pada Algoritma terpilih. Setelah itu akan dilakukan lagi pengujian dengan Kurva ROC dan Confusion Matrix untuk mencari model mana yang menghasilkan tingkat akurasi paling tinggi dalam prediksi *software defect*. Hasil akurasi yang diperoleh membuktikan bahwa SVM + GA memiliki tingkat akurasi yang lebih tinggi dibandingkan MLP+GA. dengan menabahkan optimasi *Genetic Algorithm* menghasilkan persentase akurasi 95,02% dan dengan nilai AUC (*Area Under Curve*) sebesar 0,964 dengan demikian algoritma SVM + GA dioptimisasi dengan *Genetic Algorithm* dapat memprediksi *Software Defect* dengan lebih baik.

***Kata kunci** : MLP dan SVM, Optimasi ,Genetic Algorithm, Confusion Matrix, Kurva ROC*

1.PENDAHULUAN

1.1. Latar Belakang

Perusahaan atau institusi diberbagai bidang memerlukan *software* untuk membantu proses bisnis mereka agar dapat berjalan dengan efektif dan efisien. Tentunya *software* yang digunakan haruslah mempunyai standar kualitas yang baik agar tujuan dari perusahaan atau institusi tersebut dapat terpenuhi. Permintaan akan *software* yang berkualitas untuk mendukung kinerja perusahaan atau institusi meningkat dari tahun ke tahun. Seperti yang ditulis Fenton dan dikutip Gayatrri bahwa atribut-atribut pada kualitas *software* adalah *reliability, functionality, fault proneness, reusability,* dan *comprehensibility.* Diantara atribut dari kualitas *software*, *fault proneness* merupakan masalah penting, karena dapat digunakan untuk menilai kualitas akhir dari *software*, memperkirakan standar dan kepuasan pelanggan. *Fault proneness* adalah probabilitas kesalahan yang terdapat pada *software* (Pai & Dugan, 2007:675). *Fault proneness* merupakan salah satu atribut dalam menilai software yang menjadi perhatian karena dapat menjadi alat ukut kesalahan/*error* (*defect*). Perusahaan atau institusi pasti memerlukan *software* yang memiliki kesalahan sedikit atau tanpa kesalahan agar investasinya di bidang teknologi informasi tidak menjadi sia-sia. Jumlah *defect* pada *software* dapat digunakan untuk mengukur kualitas pengembang *software* dan mengatur proses *software* (Song, et al. 2006: 69).

Banyak Perusahaan besar masih bertanya bagaimana mereka dapat memprediksi kualitas *software* mereka sebelum digunakan meskipun sudah banyak penelitian besar berusaha untuk menemukan jawaban pertanyaan ini selama 30 tahun terakhir. Pengembangan *software* yang besar dan sistem yang rumit merupakan tantangan tersendiri (Lessmann, *et al*, 2008:485). Banyak cara ditempuh pada saat pembuatan *software* ,agar dapat berjalan dan berfungsi dengan baik. Akan tetapi

cara-cara yang dilakukan mungkin saja masih bisa terjadi kesalahan pada *software* yang dibuat. Diantaranya dengan melakukan prediksi kesalahan pada saat proses pembuatan *software* tersebut. Prediksi *software defect* diharapkan dapat mengurangi adanya kesalahan pada *software* sehingga tujuan dari perusahaan dapat tercapai dengan sempurna.

Beberapa Peneliti mencoba melakukan pendugaan terhadap *defect software* dengan menggunakan 1 metode data mining dan mereka mendapatkan hasil yang cukup akurat. Seperti yang dilakukan oleh Gayatri dengan *Decision Tree*, Ganesh J. Pai dengan menggunakan metode *Bayesian Network*, Khoshgoftaar dengan *Classification Trees*, dan yang lainnya. Kemudian peneliti lain mencoba melakukan ekperimen lain dengan menggunakan beberapa *dataset*, hasilnya sangat tidak relevan jika menggunakan dengan 1 metode. Eksperimen dengan beberapa metode dan menggunakan beberapa *dataset* menghasilkan kesimpulan yang beragam. Tidak ada metode yang paling akurat pada semua *dataset* (Lessmann, *et al*, 2008:485).Begitu pula pernyataan Qinbao Song, bahwa pemilihan metode yang tepat pada *dataset* yang berbeda-beda, proses evaluasi, dan proses penentuan keputusan sangatlah penting (Song & liu, 2011). Oleh karena itu belum ditemukan metode yang tepat pada *dataset* yang berbeda untuk prediksi *software defect*.

Dari beberapa penelitian yang sudah dilakukan, terdapat beberapa metode pada beberapa *dataset* yang popular untuk prediksi *software defect*. Ada yang melakukan seleksi atribut ataupun klasifikasi atribut, kemudian melakukan komparasi terhadap metode yang digunakan dan mendapatkan hasil evaluasinya. Dari semua model yang telah diteliti, belum ada model yang menghasilkan akurasi yang sangat tepat pada prediksi *Software Defect*. Meskipun dengan proses model yang berbeda, tetap saja belum ada model yang dapat menjadi acuan untuk prediksi *defect software*. *Dataset* yang mereka gunakan juga belum sepenuhnya sama. Penelitian oleh Tim Menzies dan rekannya pada tahun 2007 menggunakan NASA *dataset* dari Promise Repository, sedangkan penelitian oleh Stefan Lessmann dan rekannya, Khoshgoftaar dan rekannya, Qinbao Song dan rekannya menggunakan NASA *dataset* dari MDP *Repository* (Shepperd, et al. 2011). Penggunaan *dataset* yang berasal dari *repository* yang berbeda juga dapat menghasilkan akurasi yang berbeda.

Bedasarkan hasil Penelitian yang sudah dilakukan sebelumnya untuk memprediksi *software defect*, terdapat beberapa algoritma data mining yang cukup akurat untuk beberapa *dataset*, yang akan menjadi panutan pada penelitian ini. adopsi Algoritma yang digunakan adalah *Multi Layer Perceptron* (MLP) dan SVM. Kedua algoritma ini dipilih karena dari beberapa penelitian sebelumnya memiliki tingkat akurasi yang cukup tinggi pada beberapa *dataset* (Lesman, et all, 2008:485). Kedua algoritma ini akan dicari tingkat akurasinya dan dikomparasi untuk mencari algoritma mana yang terbaik dalam memprediksi *software defect*.

Untuk meningkatkan akurasi dalam penelitian ini digunakan pemilihan variabel atau yang sering didengar dengan *Feature Selection*. Salah satu metode yang sering digunakan adalah metode *Genetic Algorithm* (GA). *Genetic Algorithm* proses menggabungkan metodologi evaluasi yang secara heuristik/natural. Tujuan dari pemilihan *variable* adalah mengidentifikasi variabel yang sama pentingnya dalam *dataset*, kemudian membuang variabel lain yang nilainya tidak relevan dan berlebihan (Maimon & Rockach, 2010:84). Dengan adanya pemilihan variabel membuat metode lebih cepat dan lebih efektif karena tidak menggunakan variabel yang tidak relevan dan berlebihan. Terlebih lagi hasil dengan pemilihan variabel memungkinkan dapat meningkatkan akurasi dalam pengklasifikasian data yang dalam hal ini diharapkan dapat meningkatkan tingkat akurasi dari algoritma *Multi Layer Perceptron* (MLP) dan *Support Vector Machine* (*SVM*) dalam memprediksi *Software Defect.*

1.2. Rumusan Masalah

Adapun rumusan masalah dalam penelitian sebagai berikut :

1. Algoritma data mining manakah antara *Multi Layer Perceptron (MLP)* dan *Support Vector Machine* (*SVM*) yang dioptimisasi dengan *Genetic Algorithm* dapat digunakan untuk *Software Defect* ?
2. Diantara *Multi Layer Perceptron (MLP)* dan *Support Vector Machine* (*SVM*) yang dibahas dalam penelitian ini algoritma manakah yang menjadi model algoritma terpilih dan memiliki akurasi dan performa terbaik dalam memprediksi *Software Defect* ?

2. METODOLOGI

Salah satu penerapan dalam metodologi pengumpulan data dalam penelitian ini adalah studi pustaka, studi pustaka bermanfaat agar menghindari pembuatan ulang, mengidentifikasi metode yang pernah dilakukan serta untuk mengetahui peneliti lain yang mempunyai area yang sama dalam bidang ini. Dalam metodologi ini juga membandingkan penemuan-penemuan yang telah dilakukan oleh penelitian sebelumnya yang berhubungan dengan topik penelitian. Adapun literatur dalam penelitian ini sebagai berikut:

1. Penelitian ini dilakukan oleh Atac Deniz Oral, Ayse Basar Bener pada tahun (2007) Peneliti melakukan penelitian untuk membangun dengan framework untuk prediksi defect software. Mereka menggunakan static code atribut dalam memprediksi software defect. *Dataset* yang digunakan NASA *dataset* berasal dari MDP Repository. Peneliti menggunakan 7 *dataset*, yaitu CM1, PC1, PC3, PC4, LP1, LP2, LP3. Peneliti menggunakan 4 metode (*Multi Layer Perceptron* (MLP), Naïve Bayes, Classification by Voting Feature Intervals (VFI), dan *The Ensemble* (Penggabungan 2 algoritma)) untuk melakukan prediksi. Peneliti juga menggunakan 10-fold cross validasi untuk mengukur hasil prediksi
2. Pada penelitian lainnya yang dilakukan oleh Tracy Hall (Hall, et al, 2011:11) dengan melakukan *review* literatur pada prediksi kesalahan pada *Software Engineering*. Peneliti mengumpulkan semua jurnal dari Januari 2000 sampai Desember 2010 yang berhubungan dengan prediksi *Software Defect* sejumlah 208. Identifikasi jurnal tersebut dilakukan untuk menentukan model dan metode yang akan dikembangkan dan digunakan. Data yang digunakan berasal dari semua jurnal yang melakukan penelitian dengan menggunakan NASA *dataset*. Penelitian yang menggunakan *dataset* yang sama diharapkan mampu menghasilkan *review* literatur yang sesuai. Penentuan kriteria dalam melakukan penelitian dengan kriteria prediksi, kriteria variabel data, kriteria model yang diusulkan, kriteria data. Hasilnya dari 208 jurnal tersebut, 36 jurnal berhasil sesuai dengan kriteria tersebut dan sisanya tidak sesuai dengan kriteria yang sudah ditentukan. Dari 36 jurnal yang diteliti, menghasilkan model dan metode yang lebih dominan dan menghasilkan akurasi yang cukup dalam melakukan prediksi *Software Defect* yakni metode *Naive Bayes* dan *Logistic Regression*. Kombinasi variabel bebas yang diteliti juga sangat berpengaruh. Dan juga seleksi fitur yang sudah digunakan pada metode menghasilkan performa yang baik.

Berdasarkan hasil literatur Penelitian yang sudah dilakukan sebelumnya untuk memprediksi *software defect*, terdapat beberapa algoritma data mining yang cukup akurat untuk beberapa *dataset*, yang akan menjadi panutan pada penelitian ini. adopsi Algoritma yang digunakan adalah *Multilayer Perceptron* (MLP) dan SVM. Kedua algoritma ini dipilih karena dari beberapa penelitian sebelumnya memiliki tingkat akurasi yang cukup tinggi pada beberapa *dataset* (Lessmann, et all ,2008). Kedua algoritma ini akan dicari tingkat akurasinya dan dikomparasi untuk mencari algoritma mana yang terbaik dalam memprediksi *software defect*.

Penelitian ini termasuk kedalam jenis penelitian Terapan karena berfokus untuk mencari solusi tentang masalah *software defect*. Tujuan dari penelitian ini adalah pemecahan masalah sehingga hasil penelitian bisa langsung diterapkan dan dapat dimanfaatkan untuk kepentingan bersama.

3. HASIL DAN PEMBAHASAN

3.1. Algoritma SVM (*SupportVectorMachine*)

Pehitungan manual metode ***Support Vector Machine*** (SVM) menggunakan *sample dataset* yang ada setelah dilakukan pemilihan variabel yaitu menggunakan dari salah satu dataset (CM1) sehingga didapat Karnel Model dari *rafid miner* dengan bobot dari masing masing atribut terlihat pada tabel 1 dibawah ini.

Tabel 1.Nilai Bobot *Support Vector Machine*

No	Atribute	Weight
1.	**No**	**-1.015**
2.	**Loc**	**0.079**
3.	**N**	**-0.069**
4.	**lOCode**	**0.314**
5.	**lOComment**	**-0.291**
6.	**lOBlank**	**-0.473**
7.	**lOCodeAndComment**	**-0.287**
8.	**uniq_Op**	**-0.088**
9.	**uniq_Opnd**	**-0.088**
10.	**total_Op**	**-0.113**
11.	**total_Opnd**	**-0.242**

Bias (offset): 1.782

3.2. Algoritma GA (*Genetic Algorithm*)

Dengan menggunakan optimasi *genetic algorithm* (GA), attribute akan diberikan bobot sehingga model yang terbentuk dapat lebih baik. Bobot *attribute* yang diberikan oleh algoritma optimasi *genetic algorithm* (GA) dapat dilihat pada tabel 2 dibawah ini.

Tabel 2. Pembobotan *Attribute Genetic Algorithm(GA)*

No	Atribute	Weight
1.	log	0.0
2.	n	0.0
3.	IOCode	1.0
4.	IOComment	0.0
5.	IOBlack	1.0
6.	IOCodeAndIOComment	0.0
7.	uniq_op	1.0
8.	uniq_opnd	0.0
9.	total_Op	0.0
10.	total_Opnd	0.0

3.3. *Multi Layer Perceptron* (MLP)

Pehitungan manual metode *Multi Layer Perceptron* menggunakan *sample dataset* yang ada setelah dilakukan pemilihan variabel yaitu menggunakan dari salah satu *dataset.*

Berikut ini adalah *neural net* yang dihasilkan dari data training dengan menggunakan *multilayerperceptron* pada *tool Rapid Miner*.

Tabel 3. Nilai pada Hidden 1

No	Node 1	Node 2	Node 3	Node 4	Node 5	Node 6
1.	no: -0.308	no: -5.729	no: -2.008	no: 8.163	no: -0.116	no: -0.116
2.	loc: 0.011	loc: 1.565	loc: 0.499	loc: 1.540	loc: 0.264	loc: 0.264
3.	n: 0.501	n: -3.774	n: -1.047	n: 0.757	n: 0.277	n: 0.277
4.	lOCode: -0.124	lOCode: 3.179	lOCode: 1.086	lOCode: -1.585	lOCode: 0.346	lOCode: 0.346
5.	lOComment: 0.026	lOComment: 0.732	lOComment: -0.114	lOComment : 4.534	lOComment : 0.329	lOComment : 0.329
6.	lOBlank: 0.858	lOBlank: -4.139	lOBlank: -1.325	lOBlank: 3.892	lOBlank: 0.247	lOBlank: 0.247
7.	lOCodeAndComment: 0.305	lOCodeAndComment: 5.579	lOCodeAndComment: 2.195	lOCodeAndComment: -1.915	lOCodeAndComment: 0.344	lOCodeAndComment: 0.344
8.	uniq_Op: 0.522	uniq_Op: -5.119	uniq_Op: -1.590	uniq_Op: 0.903	uniq_Op: 0.086	uniq_Op: 0.086
9.	uniq_Opnd: -0.045	uniq_Opnd: -2.720	uniq_Opnd: -1.222	uniq_Opnd: 2.569	uniq_Opnd: 0.294	uniq_Opnd: 0.294
10.	total_Op: 0.509	total_Op: -3.431	total_Op: -0.959	total_Op: 0.718	total_Op: 0.290	total_Op: 0.290
11.	total_Opnd: 0.536	total_Opnd: -4.372	total_Opnd: -1.299	total_Opnd: 0.683	total_Opnd: 0.245	total_Opnd: 0.245
12.	Threshold: -0.314	Threshold: -5.557	Threshold: -2.226	Threshold: 1.869	Threshold: -0.322	Threshold: -0.322
13.	no: -0.308	no: -5.729	no: -2.008	no: 8.163	no: -0.116	no: -0.116
14	loc: 0.011	loc: 1.565	loc: 0.499	loc: 1.540	loc: 0.264	loc: 0.264

3.4. Uji Model

Dalam penulisan ini misalkan, metode yang digunakan, yaitu algoritma *Support Vector Machine, Multi Layer Perceptron* dan *Genetic algorithms + Support Vector Machine*, kemudian dilakukan komparasi Ketiga dan mengukur metode mana yang paling akurat. Metode klasifikasi bisa dievaluasi berdasarkan beberapa kriteria seperti tingkat akurasi, kecepatan, kehandalan, skalabilitas, dan interpretabilitas.

3.5. Pengujian Model

Model yang telah dibentuk diuji tingkat akurasinya dengan memasukan data uji yang berasal dari data *training*. Karena data yang didapat dalam penelitian ini setelah proses *preprocessing* hanya 401 data maka digunakan metode *cross validation* untuk menguji tingkat akurasi. Untuk nilai akurasi model untuk metode SVM sebesar 91.52% , metode *MLP* sebesar 94.27% dan GA +SVM 95,02

3.6. *Confusion Matrix* algoritma SVM

Tabel 4 adalah perhitungan akurasi data training menggunakan algoritma SVM Diketahui dari 401 data training, dengan menggunakan metode algoritma SVM didapat klafikasi 42 data prediksi *True* sesuai memang *True*, 8 datap rediksi *True* ternyata *False*, didapat klasifikasi 26 data prediksi *False* ternyata malah *True*, dan 325 data prediksi *False* memang sesuai dengan *False*.

Tabel 4. *Confussion Matrix* data training Untuk Algoritma *SVM*

accuracy: 91.52% +/- 3.58% (mikro: 91.52%)

	true true	true false	class precision
pred. true	42	8	84.00%
pred. false	26	325	92.59%
class recall	61.76%	97.60%	

Perhitungan nilai akurasi dari *confusion matrix* tersebut adalah sebagai berikut:

$$akurasi = \frac{(TP + FN)}{(TP + TN + FN + FP)}$$

$$= \frac{(42 + 325)}{(42 + 8 + 325 + 26)}$$

$$= 0.9152 = \mathbf{91.52\%}$$

$$precision = \frac{FN}{(FN + FP)}$$

$$= \frac{325}{(325 + 26)}$$

$$= 0.9259 = 92$$

$$recall = \frac{FN}{(FN + TN)}$$

$$= \frac{325}{(325 + 8)}$$

$$= 0.9760 = \mathbf{97,60\%}$$

Performance Vector pada rafidminer

PerformanceVector

```
PerformanceVector:
accuracy: 91.52% +/- 3.58% (mikro: 91.52%)
ConfusionMatrix:
True:   true    false
true:   42      8
false:  26      325
```

Gambar 1. *Text View* Model *Confusion Matrix* untuk algoritma *SVM*

3.7. *Confusion Matrix* algoritma GA

Tabel 5 adalah perhitungan akurasi data training menggunakan algoritma GA model yang di pilih adalah SVM karena proses pengolah data lebih cepat.Diketahui dari 401 data training, dengan menggunakan metode algoritma GA didapat klasifikasi 53 data prediksi *true* sesuai memang *true*, 5 prediksi *true* ternyata memang *false*, didapat klasifikasi 15 data prediksi *false* ternyata *true*, dan 328

data prediksi *false* memang sesuai dengan *false*.

Tabel 5. *Confussion Matrix* data training untuk Algoritma *GA Model SVM*

accuracy: 95.02% +/- 5.68% (mikro: 95.01%)

	true true	true false	class precision
pred. true	53	5	91.38%
pred. false	15	328	95.63%
class recall	77.94%	98.50%	

Nilai akurasi dari *confusion matrix* tersebut adalah sebagai berikut:

$$akurasi = \frac{(TP + FN)}{(TP + TN + FN + FP)} = \frac{(53 + 328)}{(53 + 5 + 328 + 15)} = 0.9502 = \mathbf{95.02\%}$$

$$precision = \frac{FN}{(FN + FP)} = \frac{328}{(328 + 15)} = 0.9563 = \mathbf{95}$$

$$recall = \frac{FN}{(FN + TN)} = \frac{328}{(328 + 5)} = 0.9850 = \mathbf{98,50\%}$$

Performance Vector pada rafidminer

PerformanceVector

```
PerformanceVector:
accuracy: 95.02% +/- 5.68% (mikro: 95.01%)
ConfusionMatrix:
True:   true    false
true:   53      5
false:  15      328
```

Gambar 2. *Text View* Model *Confusion Matrix* untuk algoritma *GA model SVM*

3.8. Confusion Matrix algoritma MLP

Tabel 6 adalah perhitungan akurasi data training menggunakan algoritma MLP diketahui dari 401 data training, didapat klasifikasi 59 data prediksi *true* sesuai memang *true*, 14 prediksi *true* ternyata memang *false*,didapat klasifikasi 9 data prediksi *false* ternyata malah *true*, dan 319 data prediksi *false* memang sesuai dengan *false*.

Tabel 6. *Confussion Matrix* data training untuk Algoritma MLP

accuracy: 94.27% +/- 2.24% (mikro: 94.26%)

	true true	true false	class precision
pred. true	59	14	80.82%
pred. false	9	319	97.26%
class recall	86.76%	95.80%	

Nilai akurasi dari *confusion matrix* tersebut adalah sebagai berikut:

$$akurasi = \frac{(TP + FN)}{(TP + TN + FN + FP)} = \frac{(59 + 319)}{(59 + 14 + 319 + 9)} = 0.9427 = \mathbf{94,27\%}$$

$$precision = \frac{FN}{(FN + FP)} = \frac{319}{(319 + 9)} = 0.9726 = \mathbf{97}$$

$$recall = \frac{FN}{(FN + TN)} = \frac{319}{(319 + 14)} = 0.9580 = \mathbf{95,80\%}$$

Performance Vector pada *rafidminer*

PerformanceVector

```
PerformanceVector:
accuracy: 94.27% +/- 2.24% (mikro: 94.26%)
ConfusionMatrix:
True:   true    false
true:   59      14
false:  9       319
```

Gambar 3. *Text View* Model *Confusion Matrix* untuk algoritma *MLP*

3.9. *Confusion Matrix* Komparasi

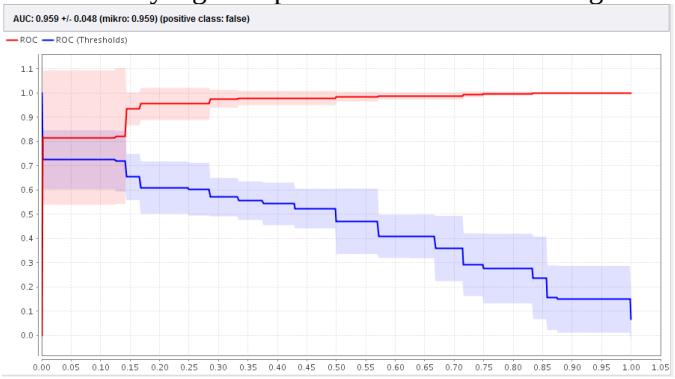
Dari tiga table *confusion matrix*, selanjutnya dilakukan perhitungan nilai *accuracy*, *precision*, dan *recall*. Perbadingan nilai *accuracy*, *precision*, dan *recall* yang telah dihitung untuk metode *SVM*, *GA+svm*, dan *MLP* dapat dilihat pada Tabel 7

Tabel 7. Komparasi Nilai A*ccuracy, Precision*, dan *Recall*

	SVM	*GA+svm*	*MLP*
Accuracy	91.52%	95.02%	94.27%
Precision	92.59%	95.63%	97.26%
Recall	97.60%	98.50%	95.80%

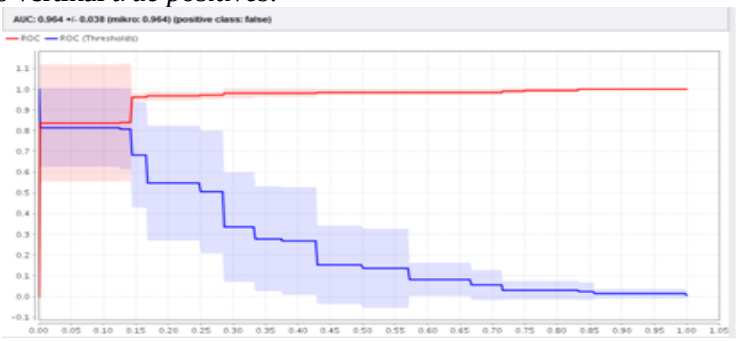
3.10. Kurva ROC

Hasil perhitungan divisualisasikan dengan kurva ROC. Perbandingan ketiga metode komparasi bisa dilihat pada Gambar 4 yang merupakan kurva ROC untuk algoritma SVM.



Gambar 4. Kurva ROC dengan algoritma SVM

Kurva ROC pada gambar 5 mengekspresikan *confusion matrix* dari Tabel 7 Garis horizontal adalah *false positives* dan garis vertikal *true positives*.



Gambar 5. Kurva ROC dengan Metode *Genetic algorithms(GA)*

Seperti terlihat pada Gambar 4, Gambar 5¸dan Gambar 6¸ area di bawah kurva pada Gambar 6 paling luas diantara ketiga metode

Gambar 6. Kurva ROC dengan Metode *Neural Network*

Pebandingan hasil perhitungan nilai AUC untuk metode *SVM, GA* +SVM, MLP dapat dilihat pada Tabel 8

Tabel 8. Komparasi Nilai AUC

	SVM	*GA+SVM*	*MLP*
AUC	0.959	0.964	0.960

3.11. Analisis Hasil Komparasi

Model yang dihasilkan dengan metode SVM, *GA +SVM*, dan *MLP* diuji menggunakan metode *Cross Validation*, terlihat perbandingan nilai *accuracy*, *precision*, *sensitivity*, dan *recall* pada Tabel 9, untuk metode *GA* memiliki nilai *accuracy*, *precision*, dan *recall* yang paling tinggi, diikuti dengan metode *SVM*, dan yang terendah adalah *MLP*.

Tabel 9. Komparasi Nilai A*ccuracy* dan AUC

	SVM	***GA***	***MLP***
Accuracy	**91.52%**	95.02%	94.27%
AUC	0.959	0.964	0.960

Tabel 9 membandingkan *accuracy* dan AUC dari tiap metode. Terlihat bahwa nilai *accuracy* GA paling tinggi begitu pula dengan nilai AUC-nya. Untuk metode *MLP* dan *SVM* juga menunjukan nilai yang sesuai. Untuk klasifikasi *data mining*, nilai AUC dapat dibagi menjadi beberapa kelompok

0.90-1.00 = klasifikasi sangat baik

0.80-0.90 = klasifikasi baik

0.70-0.80 = klasifikasi cukup

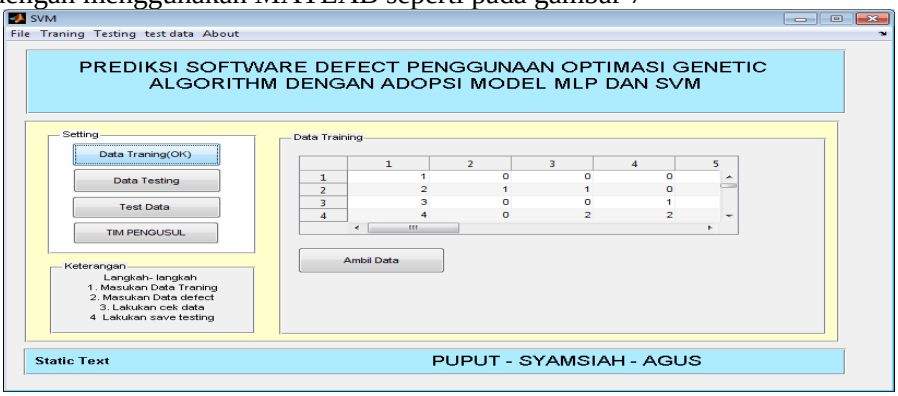
0.60-0.70 = klasifikasi buruk

0.50-0.60 = klasifikasi salah

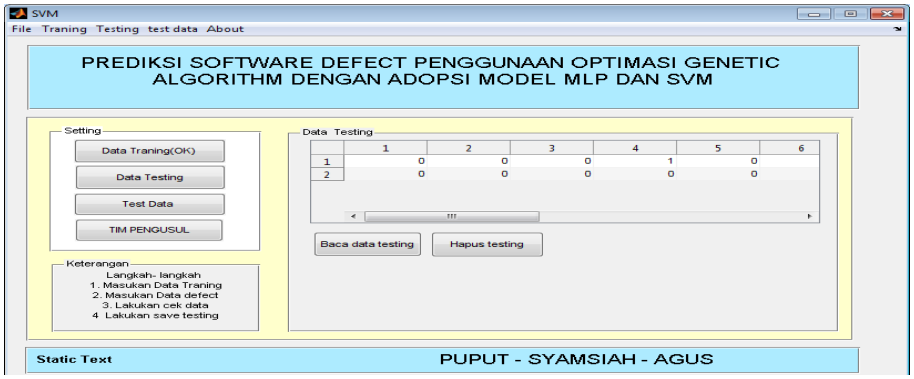
Berdasarkan pengelompokkan di atas dan Tabel IV.10 maka dapat disimpukan bahwa metode *SVM +GA* dan *MLP+GA* termasuk klasifikasi cukup karena memiliki nilai AUC antara 0.70-0.80.

3.12. Hasil Algoritma terpilih

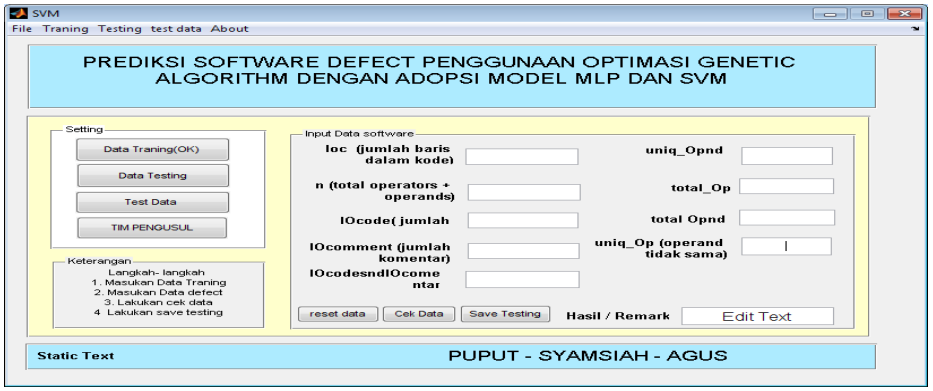
Berdasarkan hasil perbandingan akurasi pada tabel IV.10 algoritma terpilih sebagai algoritma terbaik dalam klasifikasi penentuan *Software Difect* yaitu algoritma *SVM +GA* yang memiliki tingkat akurasi tertinggi dengan persentase 95.02% dilakukan penerapan pada data baru (tabel IV 9).Hasil klasifikasi dari algoritma GA+SVM diterapkan kedalam pembuatan aplikasi untuk klasifikasi *software defect* dengan menggunakan MATLAB seperti pada gambar 7



Gambar 7. Aplikasi untuk Klasifikasi penentuan software defect form data training



Gambar 8. Aplikasi untuk Klasifikasi software defect form Data Testing



Gambar 9. Aplikasi untuk Klasifikasi software defect from Test Data

Gambar 10. Aplikasi untuk Klasifikasi form Profil peneliti

Pada aplikasi untuk klasifikasi software defectpada gambar 10 dihasilkan klasifikasi true dan false input data Anggota koperasi pada program tersebut sesuai dengan atribut yang dibutuhkan, kemudian klik tombol Cek data, maka secara otomatis tampil hasil klasifikasi software defectyang **true** atau **false**

3.13. Analisis Aplikasi (SQA)

Untuk mengetahui kualitas dari *Aplikasi Klasifikasi penentuan software defect* maka digunakan *Software Quality Assurance* (SQA). Komponen dari SQA yang digunakan pada pengujian ini dapat di lihat pada tabel 10 :

Tabel 10. *Metric of Software Quality Assurance* (SQA)

No	Metrik	Deskripsi	Bobot
1	Accuracy	Ketepatan perhitungan	0.25
2	Completeness	Kelengkapan kebutuhan	0.15
3	Operability	Kemudahan untuk dioperasikan	0.25
4	Simplicity	Kemudahan untuk difahami	0.1
5	Training	Kemudahan pembelajaran	0.25

Ada 5 komponen dalam *software quality assurance* (SQA) yang digunakan. Dari 5 komponen tersebut akan dibuat 5 pertanyaan untuk angket yang akan disebarkan kepada 5 orang pengamat yang merupakan user yang diambil secara acak.

Tabel 11. Hasil Evaluasi Software Quality Assurance (SQA)

User	Skor Metrik					Skor
	1	2	3	4	5	
1	100	80	80	100	100	87
2	100	80	80	60	100	83
3	100	80	100	80	80	85
4	80	60	80	100	80	75
5	100	60	80	100	100	84
					Rata-Rata	82.8

$$\text{Skor1} = (100 * 0.25) + (80 * 0.15) + (80 * 0.25) + (100 * 0.1) + (100 * 0.25) = 87.00$$

$$\text{Skor2} = (100 * 0.25) + (80 * 0.15) + (80 * 0.25) + (60 * 0.1) + (100 * 0.25) = 83.00$$

$$\text{Skor3} = (100 * 0.25) + (80 * 0.15) + (100 * 0.25) + (80 * 0.1) + (80 * 0.25) = 85.00$$

$$\text{Skor4} = (80 * 0.25) + (60 * 0.15) + (80 * 0.25) + (100 * 0.1) + (80 * 0.25) = 75$$

$$\text{Skor5} = (100 * 0.25) + (60 * 0.15) + (80 * 0.25) + (100 * 0.1) + (100 * 0.25) = 84.00$$

$$\text{Rata} - \text{Rata} = \frac{87.00 + 83.00 + 85.00 + 75.00 + 84.00}{5} = 82.80$$

Skor rata-rata yang dihasilkan adalah 82.28 merupakan hasil skor yang baik, sehingga dapat disimpulkan bahwa kualitas perangkat lunak "*Aplikasi Klasifikasi penentuan software defect*" ini cukup baik.

4. KESIMPULAN

Dalam penelitian ini dilakukan pembuatan model baru yang diusulkan dengan metode SVM dan MLP di optimisasi dengan *Genetic Algorithm* yang dilakukan pada NASA *Dataset.* Kesimpulan dapat dapat dilihat sebagi berikut:

1. Model yang diusulkan dilakukan komparasi *Multi Layer Perceptron* dan SVM yang dioptimisasi dengan *Genetic Algorithm* umenghasilkan model yang sesuai dan paling akurat untuk melakukan prediksi *software defect*.
2. Berdasarkan hasil evaluasi dan validasi dapat dikesimpulan bahwa, algoritma SVM dengan Optimasi *Genetic Algorithm* memiliki akurasi dan performa terbaik secara rata-rata untuk semua *dataset* yaitu sebesar 95.02% dan nilai *Area Under Curve* (AUC) sebesar 0.964.

DAFTAR PUSTAKA

Hall, T., S. Beecham, D. Bowes, Gray D., and S. Counsell. 2011.*A Systematic Literature Review on Fault Prediction Performance in Software Engineering*. IEEE Transactions on Software Engineering.

Lessmann, Stefan, Bart Baesens, Christophe Mues, dan Swantje Pietsch. 2008.*Benchmarking Classification Models for Software Defect Prediction: A Proposed Framework and Novel Findings***.**IEEE Transactions on Software Engineering.

Maimon, Oded, dan Lior Rokach.. 2010. *Data Mining and Knowledge Discovery Handbook*. Israel: Springer Science and Business Media, 2010.

Menzies, Tim, Jeremy Greenwald, dan Art Frank. 2007. *Data Mining Static Code Attributes to Learn Defect Predictors***.**IEEE Transactions on Software Engineering,

Oral, Atac Deniz, dan Ayse Basar Bener. 2007.*Defect Prediction for Embedded Software*. IEEE.

Pai, Ganesh J., dan Joanne Bechta Dugan. 2007. *Empirical Analysis of Software Fault Content and Fault Proneness Using Bayesian Network*. IEEE Transactions on Software Engineering,

Shepperd, Martin, Qinbao Song, Zhongbin Sun, dan Carolyn Mair. 2011. *Data Quality: Some Comments on the NASA Software Defect Data Sets.*IEEE.

Song, Jia, Shepperd, Ying, dan Liu.. 2011. *A General Software Defect-Proneness Prediction Framework*. IEEE Transactions On Software Engineering,

Song, Qinbao, Martin Shepperd, Michelle Cartwright, dan Carolyn Mair. 2006. *Software Defect Association Mining and Defect Correction Effort Prediction*. IEEE Transactions on Software Engineering.