Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) p-ISSN: 2355-7699
Vol. 13, No. 2, April 2026, hlm. 273-282 e-ISSN: 2528-6579

GREY WOLF OPTIMIZER UNTUK SELEKSI FITUR PADA DETEKSI KECACATAN PERANGKAT LUNAK

Muhammad Zain Fawwaz Nuruddin Siswantoro[*,1], Darlis Herumurti [2]

[1]Universitas Internasional Semen Indonesia, Gresik, [2] Institut Teknologi Sepuluh Nopember, Surabaya
Email: [1]muhammad.siswantoro@uisi.ac.id, [2]darlis@its.ac.id
[*]Penulis Korespondensi

(Naskah masuk: 09 Januari 2024, diterima untuk diterbitkan: 17 November 2025)

Abstrak

Deteksi kecacatan perangkat lunak berperan penting dalam mengidentifikasi komponen yang berpotensi bermasalah sebelum kegagalan terjadi. Meskipun prediksi kecacatan dapat mengoptimalkan waktu pengembangan, menurunkan biaya, dan meningkatkan kepuasan pelanggan, efektivitasnya sangat bergantung pada pemilihan fitur yang relevan untuk model klasifikasi. Penelitian ini mengevaluasi penerapan Grey Wolf Optimizer (GWO) untuk seleksi fitur pada deteksi kecacatan perangkat lunak menggunakan Support Vector Machine (SVM), serta membandingkan kinerjanya dengan Genetic Algorithm (GA), Particle Swarm Optimization (PSO), dan Firefly Algorithm (FFA). Evaluasi dilakukan pada 12 dataset dari NASA Metrics Data Program (NASA MDP) dengan penerapan Synthetic Minority Oversampling Technique (SMOTE) untuk menangani ketidakseimbangan kelas. Hasil menunjukkan bahwa seleksi fitur berbasis GWO secara konsisten meningkatkan performa SVM dibandingkan penggunaan semua fitur, serta secara signifikan mengungguli metode berbasis GA, PSO, dan FFA pada seluruh dataset.

Kata kunci: *seleksi fitur, deteksi kecacatan perangkat lunak, GWO, SVM*

GRAY WOLF OPTIMIZER FOR FEATURE SELECTION IN SOFTWARE DEFECT DETECTION

Abstract

Software defect detection plays a crucial role in identifying potentially faulty components before failures occur. While defect prediction can optimize development time, reduce costs, and enhance customer satisfaction, its effectiveness largely depends on selecting the most relevant features for classification models. This study evaluates the application of the Grey Wolf Optimizer (GWO) for feature selection in software defect detection using a Support Vector Machine (SVM) and compares its performance with Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Firefly Algorithm (FFA). The evaluation, conducted on 12 datasets from the NASA Metrics Data Program (NASA MDP), incorporates the Synthetic Minority Oversampling Technique (SMOTE) to address class imbalance. Results show that GWO-based feature selection consistently improves SVM performance over using all features and significantly outperforms GA-, PSO-, and FFA-based approaches across all datasets.

Keywords: *feature selection, software defect detection, GWO, SVM*

1. PENDAHULUAN

Deteksi kecacatan perangkat lunak merupakan proses untuk mengidentifikasi kecacatan dalam perangkat lunak sebelum memasuki tahap pengujian. Dalam penelitian ini, cacat perangkat lunak didefinisikan sebagai ketidaksesuaian antara perilaku aktual perangkat lunak dengan spesifikasi atau kebutuhan yang diharapkan, yang dapat menyebabkan kegagalan fungsi atau hasil yang salah (IEEE, 2010). Deteksi kecacatan berkontribusi pada optimalisasi waktu pengembangan, pengurangan biaya, penurunan pekerjaan ulang, serta peningkatan kepuasan pelanggan, yang pada akhirnya menghasilkan perangkat lunak yang lebih konsisten dan dapat diandalkan. Oleh karena itu, prediksi kecacatan memiliki peran krusial dalam mencapai kualitas perangkat lunak dengan mengambil pembelajaran dari kesalahan-kesalahan sebelumnya (Rawat & Dubey, 2012).

Selama dekade terakhir, model klasifikasi telah banyak digunakan untuk mendeteksi kecacatan perangkat lunak secara otomatis. Hal ini karena model klasifikasi dapat secara otomatis mempelajari pola dari data kecacatan dalam perangkat lunak yang pernah terjadi sebelumnya. Pola ini kemudian akan digunakan oleh model untuk mendeteksi kecacatan

perangkat lunak yang lain (Siswantoro & Yuhana, 2023). Beberapa model klasifikasi telah berhasil digunakan dalam mendeteksi kecacatan perangkat lunak secara efektif pada penelitian sebelumnya, seperti k-Nearest Neighbors (k-NN) (Hammad dkk., 2019), Naïve Bayes (NB) (Ji dkk., 2019), Support Vector Machine (SVM) (Rong dkk., 2016), Artificial Neural Network (ANN) (Jayanthi & Florence, 2019) dan Decision Tree (DT) (Marian dkk., 2016).

Salah satu faktor penting yang menentukan performa model klasifikasi dalam mendeteksi kecacatan perangkat lunak adalah pemilihan fitur input. Dataset yang digunakan untuk melatih model klasifikasi pendeteksi kecacatan perangkat lunak biasanya memiliki banyak fitur. Namun, tidak semua fitur tersebut berkontribusi secara signifikan terhadap identifikasi kecacatan perangkat lunak. Proses seleksi fitur akan membantu mengidentifikasi subset fitur yang paling relevan dan informati serta mengurangi dimensi data. Dengan mengeliminasi fitur yang kurang relevan, seleksi fitur dapat meningkatkan akurasi deteksi dan mengoptimalkan waktu pelatihan model (Balogun dkk., 2020).

Beberapa peneliti menggunakan metode optimisasi metaheuristik yang terispirasi dari perilaku alam untuk mencari subset fitur yang paling berpengaruh terhadap pendeteksian kecacatan perangkat lunak. Metode optimisasi yang umum digunakan untuk seleksi fitur diantaranya Genetic Algorithms (GA) (Kumar & Das, 2023; Wahono & Herman, 2014), Particle Swarm Optimization (PSO) (Malhotra dkk., 2021), dan Firefly algorithm (FFA) (Anbu & Anandha Mala, 2019). Metode tersebut mencari kombinasi fitur terbaik dengan berevolusi mengikuti pola tertentu untuk meaksimumkan performa model klasifikasi.

Wahono, dkk (2014) mengusulkan kombinasi GA dan teknik *bagging* untuk meningkatkan akurasi prediksi kecacatan perangkat lunak. GA digunakan untuk pemilihan fitur, sedangkan teknik bagging digunakan untuk mengatasi masalah ketidakseimbangan jumlah observasi pada kelas positif dan kelas negatif. Sepuluh model klasifikasi, diantaranya Logistic Regression (LR), Linear Discriminant Analysis (LDA), NB, k-NN, K*, ANN, SVM, DT, C4.5, CART, dan Random Forest (RF), digunakan dalam penenlitian ini untuk mendeteksi kecacatan perangkat lunak pada sembilan datset dari projek NASA *Metrict Data Program* (NASA MDP). Fungsi objektif yang digunakan dalam penelitian ini adalah kombinasi linier antara akurasi dan invers dari *feature cost*. Hasil eksperimen menunjukkan bahwa pemilihan fitur menggunakan GA dan teknik *bagging* dapat meningkatkan UAC-ROC model klasifikasi pada sebagian dataset saja.

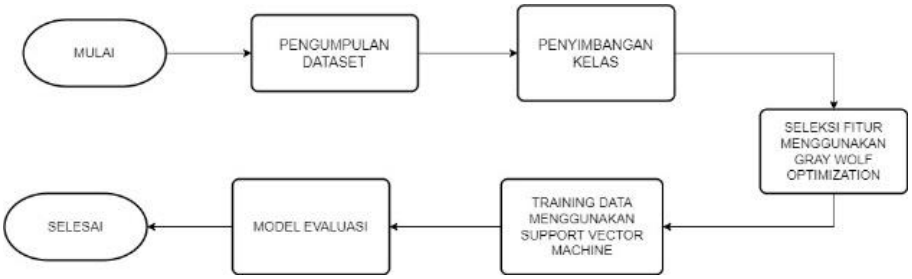
Kumar, dkk (2023) juga mengusulkan metode pemilihan fitur berbasis GA untuk mendeteksi kecacatan perangkat lunak. Tiga model klasifikasi, yaitu k-NN, DT, dan NB, digunakan untuk mendeteksi kecacatan perangkat lunak pada 12 dataset dari NASA MDP. Dalam penelitian ini *prediction error* digunakan sebagai fungsi objektif yang dicari nilai minimumnya oleh GA. Hasil eksperimen menunjukkan bahwa metode pemilihan fitur yang diusulkan berhasil mengingkatan akurasi ketiga model pada semua dataset.

Malhotra, dkk (2021) mengusulkan metode pemilihan fitur menggunakan PSO biner untuk memprediksi kecacatan perangkat lunak. Pada penelitian *Binary Cross-Entropy* digunakan sebagai fungsi obejektif yang diminimumkan oleh PSO biner. Hasil eksperimen pada model ANN menunjukkan bahwa metode yang diusulkan dapat meningkatkan UAC-ROC model ANN dalam memprediksi kecacatan perangkat lunak. Namun demikian metode ini hanya digunakan pada empat dataset NASA MDP.

Anbu dan Mala (2019) menggunakan FFA untuk mencari subset fitur terbaik dalam memprediksi kecacatan perangkat lunak. SVM, NB dan k-NN digunakan untuk mengklasifikasikan fitur terpilih dalam penelitian ini. Fungsi objektif yang digunakan dalam penelitian ini adalah jumlahan terboboti dari proporsi fitur terpilih dan invers *F measure*. Meskipun metode yang diusulkan dapat meningkatkan performa ketiga model, namun metode ini hanya dievaluasi menggunakan satu dataset dari NASA MDP.

Grey Wolf Optimizer (GWO) adalah algoritma optimisasi yang terinspirasi oleh perilaku kelompok serigala abu-abu dalam mencari mangsa (Mirjalili dkk., 2014). Kemampuan GWO terletak pada ketangkasannya dalam menyesuaikan parameter dan mencapai solusi optimum dalam berbagai jenis masalah optimisasi. Algoritma ini memanfaatkan langkah-langkah pencarian serigala yang cerdas dan koordinatif, seperti pemilihan serigala alpha, beta, dan delta yang menggambarkan peran pemimpin dalam kelompok. Dengan pendekatan ini, GWO mampu menyelesaikan masalah kompleks dengan cepat dan efisien, membuatnya menjadi pilihan yang menarik untuk berbagai aplikasi, termasuk dalam konteks pemilihan fitur dalam deteksi kecacatan perangkat lunak. GWO telah terbukti efektif dalam menangani berbagai tantangan optimisasi, dan daya adaptasinya membuatnya menjadi alat yang handal untuk mencapai solusi optimal dalam berbagai domain.

Penelitian ini bertujuan untuk mengusulkan penggunaan GWO dalam seleksi fitur untuk model SVM pada deteksi kecacatan perangkat lunak. Penelitian ini akan menilai sejauh mana GWO dapat meningkatkan akurasi klasifikasi SVM dan seberapa efisien algoritma ini dalam memilih subset fitur yang relevan dari dataset. Penelitian ini berfokus pada perbandingan kinerja empat metode optimisasi metaheuristik populer (GWO, GA, PSO, dan FFA) dalam seleksi fitur untuk deteksi kecacatan perangkat lunak berbasis SVM.



Gambar 1. Alur Penelitian

Meskipun masing-masing metode telah digunakan secara terpisah dalam literatur, belum ada studi yang membandingkan kinerjanya secara sistematis pada seluruh dataset NASA MDP dengan penanganan ketidakseimbangan kelas menggunakan SMOTE. Perbandingan ini penting untuk memberikan panduan berbasis bukti bagi peneliti dan praktisi dalam memilih metode seleksi fitur yang optimal untuk konteks data serupa.

2. METODE PENELTIAN

Metode yang digunakan dalam penelitian ini mencakup pemilihan dataset, penyeimbangan kelas, seleksi fitur dengan *Grey Wolf Optimizer* (GWO), penggunaan *Support Vector Machine* (SVM) sebagai model klasifikasi, serta evaluasi model yang telah dilatih menggunakan fitur terpilih. Gambaran alur penelitian ini dapat dilihat pada Gambar 1.

2.1 Pengumpulan Data

Untuk mengevaluasi metode pemilihan fitur yang diusulkan, penelitian ini menggunakan 12 dataset dari projek NASA *Metrict Data Program* (NASA MDP) yang terdiri dari CM1, JM1, KC1, KC3, MC1, MC2, MW1, PC1, PC2, PC3, PC4, dan PC5 (Gray dkk., 2012). Dataset ini dipilih karena banyak digunakan dalam penelitian sebelumnya sehingga memungkinkan perbandingan kinerja dengan studi terdahulu serta memastikan replikasi penelitian oleh pihak lain. Selain itu, dataset NASA MDP memiliki keragaman karakteristik (jumlah fitur, jumlah instance, dan distribusi kelas) yang memadai untuk menguji ketahanan metode yang diusulkan. Dataset ini difokuskan pada pengumpulan data empiris terkait dengan kecacatan perangkat lunak di berbagai proyek yang dilakukan oleh NASA.

Informasi yang tercakup meliputi jumlah cacat yang teridentifikasi dalam setiap projek, ukuran kode sumber, dan upaya yang dilakukan dalam pengembangan perangkat lunak. Para peneliti seringkali memanfaatkan dataset NASA MDP ini untuk menyelidiki bagaimana berbagai metrik perangkat lunak berhubungan dengan kemunculan cacat perangkat lunak. Tabel 1 menampilkan ringkasan 12 dataset NASA MDP yang digunakan dalam penelitian ini.

Penelitian ini menggunakan dataset yang telah melalui pemrosesan awal seperti dilaporkan dalam (Shepperd dkk., 2013). Pemrosesan pertama dilakukan untuk menghilangkan data yang bermasalah seperti data dengan nilai fitur yang bertentangan atau tidak masuk akal. Kemudian dilanjutkan dengan menghilangkan data yang tidak bermasalah tetapi tidak dapat membantu meningkatkan akurasi prediksi kecacatan perangkat lunak, seperti fitur yang nilainya kostan, dan data yang berulang atau tidak konsisten. Setiap dataset memiliki jumlah fitur dan jumlah observasi yang bervariasi dengan dua jenis output catat dan tidak cacat.

Tabel 1. Ringkasan 12 Dataset NASA MDP

Dataset	Fitur	Observasi		
		Total	Cacat	Tidak Cacat
CM1	37	327	42	285
JM1	21	7782	1672	6110
KC1	21	1183	314	869
KC3	39	194	36	158
MC1	38	1988	46	1942
MC2	39	125	44	81
MW1	37	253	27	226
PC1	37	705	61	644
PC2	36	745	16	729
PC3	37	1077	134	943
PC4	37	1287	177	1110
PC5	38	1711	471	1240

2.2 Penyeimbangan Kelas

Berdasarkan data yang diberikan pada Tabel I, terlihat adanya ketidakseimbangan yang signifikan antara jumlah observasi pada kelas cacat dan tidak cacat pada dataset NASA MDP. Jumlah observasi pada kelas tidak cacat jauh lebih dominan dibandingkan dengan kelas cacat. Untuk mengatasi ketidakseimbangan ini, diperlukan strategi khusus dalam pengolahan data sebelum digunakan dalam melatih model klasifikasi. Penelitian ini menggunakan metode *Synthetic Minority Oversampling Technique* (SMOTE) (Chawla dkk., 2002) untuk mengatasi masalah kelas yang tidak seimbang.

SMOTE adalah metode oversampling yang digunakan untuk menangani ketidakseimbangan kelas pada dataset, khususnya pada tugas klasifikasi di mana kelas minoritas (dalam penelitian ini, kelas cacat) memiliki jumlah observasi yang lebih sedikit. SMOTE bekerja dengan membuat sampel-sampel sintetis baru pada kelas minoritas dengan tahapan sebagai berikut:

1. Pilih observasi pada kelas minoritas sebagai titik awal untuk membuat observasi sintetis.
2. Pilih tetangga-tetangga dari observasi tersebut dalam ruang fitur. Jumlah tetangga yang dipilih adalah parameter yang dapat diatur sebelumnya.
3. Untuk setiap tetangga yang dipilih, observasi sintetis baru dibuat dengan menggunakan perbedaan antara fitur pada observasi minoritas awal dan tetangga tersebut. Perbedaan ini kemudian dikalikan dengan bilangan random dalam interval (0,1] dan hasilnya ditambahkan ke observasi awal.
4. Observasi sintetis yang baru dibuat kemudian digabungkan dengan dataset asli. Proses ini diulang sebanyak yang diperlukan untuk mencapai tingkat *oversampling* yang diinginkan.

2.3 Seleksi Fitur dengan *Grey Wolf Optimizer*

Grey Wolf Optimizer (GWO) merupakan algoritma optimasi yang terinspirasi dari perilaku sosial serigala abu-abu (*Canis lupus*) dalam mencari mangsa. Dikembangkan pada tahun 2014 oleh Mirjalili et al. (Mirjalili dkk., 2014), GWO dirancang untuk menyelesaikan masalah optimasi dengan cara mensimulasikan tiga tingkatan hierarki dalam kelompok serigala abu-abu, yaitu serigala alpha (α), beta (β), dan delta (δ). Serigala α sebagai pemimpin tertinggi, β sebagai pemimpin kedua, dan δ sebagai pemimpin ketiga. Selain itu, terdapat serigala omega (ω) yang pergerakannya dalam mencari mangsa mengikuti serigala α, β, dan δ. Posisi setiap serigala diwakili oleh solusi kandidat dalam ruang pencarian.

Cara kerja GWO melibatkan pembaruan secara iteratif posisi serigala berdasarkan formula matematika yang memodelkan interaksi sosial dalam kelompok. Misalkan $\mathbf{Y}(i)$ dan $\mathbf{Y}_m(i)$ masing-masing adalah posisi serigala dan mangsa pada iterasi ke-i, maka posisi serigala pada iterasi selanjutnya dapat dimodelkan menggunakan (1), (2), (3), dan (4),

$$\mathbf{A} = 2a\mathbf{r}_1 - a \quad (1)$$

$$\mathbf{C} = 2\mathbf{r}_2 \quad (2)$$

$$\mathbf{D} = |\mathbf{C}.\mathbf{Y}_m(i) - \mathbf{Y}(i)| \quad (3)$$

$$\mathbf{Y}(i+1) = \mathbf{Y}_m(i) - \mathbf{A}.\mathbf{D} \quad (4)$$

dengan a adalah konstanta yang nilainya berkurang secara linier dari 2 ke 0 selama proses iterasi, serta $\mathbf{r}_1$ dan $\mathbf{r}_2$ adalah vektor random yang elemennya berasal dari interval [0,1].

Pada kenyataannya posisi mangsa yang merupakan solusi terbaik tidak diketahui. Oleh karena itu, pergerakan serigala pada setiap iterasi akan mengikuti pergerakan serigala α, β, dan δ. Hal ini didasarkan pada asumsi bahwa posisi serigala α, β, dan δ berada paling dekat ke mangsa. Sehingga pergerakan serigala pada setiap iterasi dapat dimodelkan menggunakan (5), (6), dan (7),

$$\mathbf{D}_s = |\mathbf{C}_s.\mathbf{Y}_s(i) - \mathbf{Y}(i)| \quad (5)$$

$$\widehat{\mathbf{Y}}_s = \mathbf{Y}_s(i) - \mathbf{A}_s.\mathbf{D}_s \quad (6)$$

$$\mathbf{Y}(i+1) = \frac{1}{3}\left(\widehat{\mathbf{Y}}_\alpha + \widehat{\mathbf{Y}}_\beta + \widehat{\mathbf{Y}}_\delta\right) \quad (7)$$

dengan s = α, β, δ. $\mathbf{A}_s$ dan $\mathbf{C}_s$ masing-masing adalah vektor koefsien yang dibangkitkan menggunakan (1) dan (2). $\mathbf{Y}_s(i)$ adalah posisi dari serigala α, β, dan δ pada iterasi ke-i.

Misalkan terdapat N fitur dalam dataset, maka posisi serigala dapat dinyatakan sebagai vektor $\mathbf{Y} = (y_1, y_2, \dots, y_N)$ dengan $y_j \in [0,1], j = 0,1,2, \dots, N$. Setiap elemen $\mathbf{Y}$ mewakili kehadiran atau ketidakhadiran fitur tertentu dalam subset fitur yang dipilih. Jika $y_j \geq 0.5$, maka fitur ke-j terpilih masuk ke dalam subset yang dipertimbangkan, selain itu maka fitur ke-j tidak termasuk ke dalam subset yang dipertimbangkan. Pada setiap iterasi, serigala α, β, dan δ dipilih dari populasi yang posisinya paling dekat ke mangsa, yaitu vektor $\mathbf{Y}$ yang nilai fungsi objektifnya paling kecil pertama, kedua, dan ketiga. Pada penelitian ini, nilai fungsi objektif $\mathbf{Y}$ didefinisikan sebagai negatif dari akurasi klasifikasi model yang dilatih menggunakan subset fitur terpilih berdasarkan posisi $\mathbf{Y}$. Jumlah populasi serigala yang digunakan pada setiap iterasi adalah 10 dan maksimum iterasi adalah 20. Penelitian ini menggunakan implementasi GWO dari pustaka MEALPY (Van Thieu & Mirjalili, 2023), sebuah Pustaka sumber terbuka untuk algoritma metaheuristik dalam bahasa Python.

2.4 *Support Vector Machine*

Penelitian ini menggunakan *Support Vector Machine* (SVM) untuk mendeteksi kecacatan perangkat lunak berdasarkan subset fitur yang terpilih menggunakan GWO. SVM adalah suatu metode klasifikasi dalam pembelajaran mesin yang sangat efektif, terutama dalam menangani masalah klasifikasi biner. SVM bertujuan untuk menemukan *hyperplane* terbaik yang memisahkan dua kelas dalam ruang fitur. *Hyperplane* ini dipilih sedemikian rupa sehingga memiliki margin maksimum, yaitu jarak terdekat antara titik-titik pelatihan dan *hyperplane*.

Persamaan *hyperplane* dari SVM untuk kasus klasifikasi biner linear dinyatakan seperti pada (8),

$$y(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i\, y_i K(\mathbf{x}, \mathbf{x}_i) + b \quad (8)$$

dengan $\mathbf{x}$ adalah fitur dari observasi, α_i dan b masing-masing adalah bobot dan bias yang dihitung selama proses pelatihan, y_i dan $\mathbf{x}_i$ masing-masing adalah label kelas dan fitur pada data pelatihan, $K(\mathbf{x},\mathbf{x}_i)$ adalah fungsi kernel yang menghitung produk skalar antara fitur dari observasi dan fitur pada data pelatihan, dan N adalah jumlah data pelatihan. Sebuah observasi akan diklasifikasikan sebagai kelas postif jika $y(\mathbf{x}) > 0$, jika tidak maka sebagai kelas negatif (Bishop, 2006).

Dalam penelitian ini, digunakan fungsi kernel *radial basis function* (RBF) sebagai salah satu elemen kunci dalam SVM, seperti pada (9). Fungsi kernel RBF, juga dikenal sebagai Gaussian kernel, memungkinkan SVM untuk menangani kasus-kasus

non-linear dan dapat memetakan data ke dalam dimensi yang lebih tinggi. Fungsi RBF mengukur sejauh apa suatu observasi dari observasi-observasi lainnya dalam ruang fitur, dengan memperhitungkan *hyperparameter* γ. Penggunaan kernel RBF memungkinkan SVM untuk bekerja efektif dengan data yang memiliki distribusi yang kompleks, memungkinkan model untuk menyesuaikan dengan pola-pola non-linear dan meningkatkan kemampuan prediksi pada dataset yang lebih kompleks. Penelitian ini menggunakan implementasi SVM dari pustaka Scikit-learn (Pedregosa dkk., 2011) dengan standar *hyperparameter*.

$$K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|) \quad (9)$$

2.5 Evaluasi Model

Setiap dataset NASA MDP dibagi menjadi dua subset yang saling lepas yaitu data pelatihan (70%) dan data pengujian (30%) menggunakan sampling random terstratifikasi (Alpaydin, 2014). Data pelatihan kemudian digunakan untuk melatih model SVM menggunakan fitur terpilih, sedangkan data pengujian digunakan untuk mengevaluasi model yang terlah dilatih. Evaluasi model SVM dilakukan dengan menggunakan beberapa metrik kinerja klasifikasi yang umum digunakan untuk menunjukkan seberapa baik metode fitur seleksi yang diusulkan dapat meningkatkan kinerja model klasifikasi secara holistik.

Pertama-tama, akurasi klasifikasi digunakan sebagai indikator keseluruhan sejauh mana model mampu mengklasifikasikan data dengan benar. Misalkan *TP* adalah banyak observasi dalam data pengujian dari kelas positif (cacat) yang diklasifikasikan sebagai positif oleh model, *TN* adalah banyak observasi dalam data pengujian dari kelas negatif (tidak cacat) yang diklasifikasikan sebagai negatif oleh model, *FP* adalah banyak observasi dalam data pengujian dari kelas negatif yang diklasifikasikan sebagai positif oleh model, dan *FN* adalah banyak observasi dalam data pengujian dari kelas positif yang diklasifikasikan sebagai negatif oleh model, maka akurasi klasifikasi (*acc*) dapat dihitung menggunakan (10).

$$acc = \frac{TP + TN}{TP + TN + FP + FN} \quad (10)$$

Selanjutnya, *precision* dan *recall* digunakan sebagai ukuran tambahan tentang kemampuan model dalam mengidentifikasi kelas tertentu. *Precision* mengukur proporsi observasi yang benar-benar positif dari total observasi yang diprediksi positif oleh model dan dihitung menggunakan (11).

$$precision = \frac{TP}{TP + FP} \quad (11)$$

Kemudian, *recall* mengukur proporsi observasi yang benar-benar positif dari total observasi yang seharusnya positif dan dihitung menggunakan (12).

$$recall = \frac{TP}{TP + FN} \quad (12)$$

Terakhir, F_1 *score* adalah metrik yang mengkombinasikan *precision* dan *recall* untuk memberikan ukuran komprehensif tentang kinerja model dan dihitung menggunakan (13).

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (13)$$

3. HASIL DAN PEMBAHASAN

Pada penelitian ini telah dilakukan beberapa eksperimen untuk menguji performa model SVM dilatih menggunakan fitur terseleksi berbasis GWO pada 12 dataset NASA MDP. Selain itu model SVM juga dilatih menggunakan semua fitur asal dari setiap dataset. Sebagai perbandingan model SVM juga dilatih menggunakan fitur terseleksi berbasis *genetic algorithm* (GA) (Whitley, 1994), *particle swarm optimization* (PSO) (Kennedy & Eberhart, 1995), *firefly algorithm* (FFA) (Arora & Singh, 2013). Hasil eksperimen yang telah dilakukan disajikan pada Tabel 2 sampai dengan Tabel 5.

Tabel 2 menyajikan akurasi model SVM yang dilatih menggunakan semua fitur dan fitur terseleksi dalam mendeteksi kecacatan perangkat lunak. Dengan menggunakan semua fitur, akurasi model SVM berkisar antara 61.22% hingga 92.69% pada 12 dataset NASA MDO. Di lain pihak, pendekatan seleksi fitur berbasis GWO menonjol sebagai metode yang efektif, dengan akurasi yang secara konsisten lebih tinggi, yaitu antara 65.06% hingga 96.80%. Lebih lanjut akurasi model SVM yang dilatih menggunakan fitur terseleksi berbasis GWO juga lebih unggul daripada menggunakan fitur terseleksi berbasis GA, PSO, dan FFA.

Secara rata-rata, SVM yang dilatih dengan fitur terseleksi menggunakan GWO mencapai akurasi sebesar 83.40%, melampaui akurasi dengan semua fitur (78.08%) dan melampaui hasil dari algoritma GA (82.33%), PSO (81.45%), dan FFA (82.08%). Hasil ini menunjukkan bahwa GWO memiliki potensi yang kuat dalam meningkatkan kinerja SVM pada deteksi kecacatan perangkat lunak, memberikan kontribusi signifikan terhadap peningkatan akurasi model klasifikasi.

Tabel 3 menyajikan *precision* model SVM yang dilatih dengan semua fitur dan fitur terseleksi berbasis GWO, GA, PSO, dan FFA. *Precision* mengukur sejauh mana hasil positif dari model SVM benar-benar relevan. Dalam konteks ini, *precision* menyatakan persentase dari prediksi positif yang benar terhadap total prediksi positif.

Tabel 2. Akrurasi Model SVM Yang Dilatih Menggunakan Semua Fitur dan Fitur Terseleksi Berbasis GWO, GA, PSO, Dan FFA

Dataset	Akurasi (%)				
	Semua	GWO	GA	PSO	FFA
CM1	81,29	91,23	90,64	90,06	90,64
JM1	64,27	65,06	64,78	64,76	64,73
KC1	65,13	66,28	66,09	66,09	65,90
KC3	76,84	84,21	83,16	82,11	82,11
MC1	88,59	89,62	89,11	89,02	89,28
MC2	61,22	81,63	79,59	71,43	73,47
MW1	78,68	84,56	82,35	81,62	82,35
PC1	85,79	89,92	86,56	88,11	88,89
PC2	92,69	96,80	95,89	94,52	96,35
PC3	84,81	87,10	87,10	86,40	87,10
PC4	87,39	91,29	90,09	90,69	90,99
PC5	70,30	73,12	72,58	72,58	73,12
Rata-rata	78,08	83,40	82,33	81,45	82,08

Tabel 3. Precision Model SVM yang Dilatih Menggunakan Semua Fitur Dan Fitur Terseleksi Berbasis GWO, GA, PSO, Dan FFA

Dataset	Precision (%)				
	Semua	GWO	GA	PSO	FFA
CM1	83,12	91,52	91,49	91,36	90,74
JM1	65,16	66,03	65,89	65,65	65,83
KC1	67,16	68,33	67,97	68,07	67,81
KC3	78,11	84,64	84,21	83,39	83,39
MC1	88,60	89,64	89,20	89,04	89,30
MC2	61,03	81,94	79,52	71,97	73,75
MW1	79,17	85,13	82,45	81,62	82,78
PC1	86,39	90,63	87,23	89,36	89,94
PC2	93,68	97,02	96,23	94,85	96,51
PC3	85,25	87,60	87,48	87,09	87,25
PC4	88,03	91,77	90,73	91,16	91,59
PC5	70,35	73,12	73,16	72,58	73,13
Rata-rata	78,84	83,95	82,96	82,18	82,67

Dengan menggunakan semua fitur, *precision* model SVM berkisar antara 61.03% hingga 93.68% untuk 12 dataset NASA MDP. Seperti halnya pada akurasi, model SVM yang dilatih dengan fitur terseleksi menggunakan GWO menghasilkan *precision* yang lebih tinggi dari pada dilatih semua dengan semua fitur, yaitu berkisar antara 66.03% hingga 97.02%.

Secara konsisten, metode seleksi fitur berbasis GWO menunjukkan *precision* yang lebih baik dengan rata-rata 83.95%, dibandingkan dengan GA (82.96%), PSO (82.18%), dan FFA (82.67%). Temuan ini menunjukkan bahwa GWO tidak hanya efektif dalam meningkatkan akurasi, tetapi juga memberikan kontribusi positif dalam meningkatkan presisi model SVM, memperkuat validitas pendekatan seleksi fitur ini untuk meningkatkan kualitas deteksi kecacatan perangkat lunak.

Tabel 4 menggambarkan hasil *recall* model SVM yang dilatih dengan semua fitur dan fitur terseleksi menggunakan GWO, GA, PSO, dan FFA. *Recall*, atau juga dikenal sebagai *sensitivity*, mengukur kemampuan model untuk mengidentifikasi instan positif. Dengan kata lain, *recall* mengukur sejauh mana model dapat mengidentifikasi dan mengingat kembali instan positif yang sebenarnya. Seperti tampak pada Tabel 4, SVM dengan fitur terseleksi menggunakan GWO menghasilkan *recall* yang konsisten lebih tinggi, antara 65.06% hingga 96.80%, dibandingkan dengan SVM yang dilatih dengan semua fitur maupun menggunakan metode seleksi fitur lainnya. Rata-rata *recall* SVM dengan GWO mencapai 83.40%, melebihi *recall* dengan semua fitur (78.08%) dan melampaui recall dari algoritma seleksi fitur lainnya, yaitu GA (82.33%), PSO (81.45%), dan FFA (82.08%). Hasil ini mengindikasikan bahwa GWO efektif dalam meningkatkan kemampuan model untuk mengenali instan positif, yang penting untuk mengukur kesanggupan model dalam mendeteksi kecacatan perangkat lunak.

Tabel 5 menampilkan hasil F_1 *score* dari model SVM yang dilatih dengan semua fitur dan fitur terseleksi menggunakan GWO, GA, PSO, dan FFA. F1 Score menggabungkan informasi tentang *precision* dan *recall* untuk memberikan evaluasi yang holistik terhadap performa model klasifikasi. Dalam konteks ini, F_1 *score* memberikan nilai yang seimbang antara kemampuan model dalam memberikan prediksi yang tepat dan kemampuan model dalam mengidentifikasi instance positif yang sebenarnya. Hasil eksperimen menunjukkan bahwa SVM dengan fitur terseleksi menggunakan GWO memberikan F_1 *score* yang lebih tinggi secara konsisten dibandingkan dengan SVM yang dilatih dengan semua fitur atau menggunakan metode seleksi fitur lainnya. Rata-rata F_1 *score* SVM dengan GWO mencapai 83.25%, melebihi rata-rata dengan semua fitur (77.83%) dan melampaui rata-rata dari algoritma seleksi fitur lainnya. Hasil ini menegaskan bahwa

GWO memiliki kontribusi signifikan dalam meningkatkan keseimbangan antara *precision* dan *recall* pada deteksi kecacatan perangkat lunak, yang esensial dalam evaluasi kinerja model klasifikasi.

Tabel 4. Recall Model SVM yang Dilatih Menggunakan Semua Fitur Dan Fitur Terseleksi Berbasis GWO, GA, PSO, Dan FFA

Dataset	Precision (%)				
	Semua	GWO	GA	PSO	FFA
CM1	81,29	91,23	90,64	90,06	90,64
JM1	64,27	65,06	64,78	64,76	64,73
KC1	65,13	66,28	66,09	66,09	65,90
KC3	76,84	84,21	83,16	82,11	82,11
MC1	88,59	89,62	89,11	89,02	89,28
MC2	61,22	81,63	79,59	71,43	73,47
MW1	78,68	84,56	82,35	81,62	82,35
PC1	85,79	89,92	86,56	88,11	88,89
PC2	92,69	96,80	95,89	94,52	96,35
PC3	84,81	87,10	87,10	86,40	87,10
PC4	87,39	91,29	90,09	90,69	90,99
PC5	70,30	73,12	72,58	72,58	73,12
Rata-rata	78,08	83,40	82,33	81,45	82,08

Tabel 5. F_1 Score Model SVM yang Dilatih Menggunakan Semua Fitur Dan Fitur Terseleksi Berbasis GWO, GA, PSO, Dan FFA

Dataset	Precision (%)				
	Semua	GWO	GA	PSO	FFA
CM1	81,00	91,21	90,59	89,97	90,64
JM1	63,51	64,31	63,93	64,04	63,88
KC1	64,30	65,51	65,38	65,34	65,16
KC3	76,40	84,10	82,93	81,82	81,82
MC1	88,59	89,62	89,10	89,02	89,28
MC2	60,96	81,63	79,52	71,42	73,47
MW1	78,62	84,52	82,30	81,62	82,32
PC1	85,65	89,82	86,43	87,94	88,74
PC2	92,64	96,80	95,88	94,51	96,34
PC3	84,80	87,09	87,10	86,38	87,10
PC4	87,28	91,23	90,01	90,63	90,92
PC5	70,22	73,12	72,31	72,55	73,09
Rata-rata	77,83	83,25	82,12	81,27	81,90

Untuk memastikan bahwa perbedaan performa antar metode berbeda secara signifikan, dilakukan uji Friedman dengan $\alpha = 0{,}05$ pada seluruh metrik yang digunakan dalam evaluasi (akurasi, presisi, recall, dan F1-score). Hasil uji Friedman dapat dilihat pada Tabel 6. Dari Tabel 6 dapat dilihat bahwa semua metrik memiliki p-value < 0,05, hal ini menunjukkan bahwa terdapat perbedaan signifikan di antara metode yang diuji.

Tabel 6. Hasil uji Friedman untuk semua metrik evaluasi

Metrik	Chi-square	p-value
Akurasi	45,6190	$3{,}76\times10^{-8}$
Precision	44,3809	$1{,}05\times10^{-7}$
Recall	45,6190	$3{,}76\times10^{-8}$
F1-score	43,9048	$1{,}16\times10^{-7}$

Untuk mengetahui hasil seleksi fitur menggunakan GWO berbeda secara signifikan dengan metode lain, dilakukan uji post-hoc Wilcoxon berpasangan dengan koreksi Holm yang hasilnya dapat dilihat pada Tabel 7. Dari Tabel 7 dapat dilihat bahwa hasil perbandingan GWO dengan metode lain memiliki p-value kurang dari 0,05 pada semua metrik. Hasil ini menunjukkan bahwa hasil pemilihan fitur menggunakan GWO terbukti lebih baik secara signifikan dibanding menggunakan semua fitur, serta hasil seleksi fitur menggunakan GA, PSO, dan FFA pada seluruh metrik evaluasi.

Tabel 8 menyajikan jumlah fitur yang terseleksi oleh GWO, GA, PSO, dan FFA untuk berbagai dataset. Hasil eksperimen menunjukkan variasi yang signifikan dalam jumlah fitur yang terseleksi untuk setiap metode. Secara rata-rata, GWO menghasilkan subset fitur yang lebih kompak dengan rata-rata 14.75. Dengan demikian, secara rata-rata GWO berhasil mereduksi lebih dari 50% fitur yang ada di dataset asal. Di lain pihak, rata-rata jumlah fitur yang terpilih dengan menggunakan GA, PSO, dan FFA masing-masing mencapai 18.17, 18.25, dan 17.42. Hasil ini mencerminkan keunggulan GWO dalam mengidentifikasi subset fitur yang lebih kecil namun tetap informatif untuk meningkatkan kinerja model SVM pada deteksi kecacatan perangkat lunak. Efisiensi pengurangan dimensi ini dapat membantu mengoptimalkan waktu komputasi dan memperjelas interpretabilitas model tanpa mengorbankan kualitas prediksi.

Tabel 7. Hasil uji post-hoc Wilcoxon dengan koreksi Holm untuk semua metrik evaluasi

Metode 1	Metode 2	p-value			
		Akurasi	Precision	Recall	F1-score
GWO	Semua fitur	0,0049	0.0049	0.0049	0.0049
GWO	GA	0,0167	0.0059	0.0167	0.0049
GWO	PSO	0,0029	0.0029	0.0029	0.0029
GWO	FFA	0,0202	0.0049	0.0202	0.0039

Tabel 8. Jumlah Fitur Terseleksi Berbasis GWO, GA, PSO, Dan FFA

Dataset	Precision (%)				
	Semua	GWO	GA	PSO	FFA
CM1	37	19	23	17	15
JM1	21	9	10	9	10
KC1	21	13	10	10	12
KC3	39	16	15	17	19
MC1	38	21	18	24	37
MC2	39	13	15	20	19
MW1	37	13	24	18	17
PC1	37	13	25	22	12
PC2	36	16	21	20	9
PC3	37	13	14	20	20
PC4	37	21	18	21	25
PC5	38	10	25	21	14
Rata-rata	34,75	14,75	18,17	18,25	17,42

4. KESIMPULAN

Artikel ini menyajika penggunaan Grey Wolf Optimizer (GWO) dalam seleksi fitur pada model Support Vector Machine (SVM) untuk deteksi kecacatan perangkat lunak. Dalam eksperimen menggunakan 12 dataset dari projek NASA MDP diperoleh hasil bahwa model SVM yang dilatih menggunakan fitur yang diseleksi oleh GWO memiliki performa yang lebih baik, dalam hal akurasi, precision, recall, dan F_1 score daripada yang dilatih dengan semua fitur pada dataset dalam mendeteksi kecacatan perangkat lunak. Secara rata-rata fitur yang diseleksi menggunakan GWO dapat meningkatkan performa model SVM lebih dari 5%. Selain itu GWO juga menghasilkan fitur terpilih yang lebih baik jika dibandingkan dengan metode seleksi fitur berbasis Genetic Algorithms (GA), Particle Swarm Optimization (PSO), dan Firefly algorithm (FFA) secara signifikan. Namun demikian hasil penelitian ini berlaku pada dataset NASA MDP dengan karakteristik tertentu. Untuk memastikan generalisasi hasil, penelitian lanjutan pada domain dan dataset berbeda sangat disarankan. Selanjutnya untuk lebih meningkatkan performa model SVM dalam mendetksi kecacatan perangkat lunak, penggunaaan GWO untuk mengoptimasi hyperparameter model SVM juga perlu dipertimbangkan.

DAFTAR PUSTAKA

ALPAYDIN, E. 2014. *Introduction to machine learning* (2nd ed.). MIT press.

ANBU, M., & ANANDHA MALA, G. S. 2019. Feature selection using firefly algorithm in software defect prediction. *Cluster Computing*, *22*, 10925–10934.

ARORA, S., & SINGH, S. 2013. The firefly optimization algorithm: convergence analysis and parameter selection. *International Journal of Computer Applications*, *69*(3).

BALOGUN, A. O., BASRI, S., MAHAMAD, S., ABDULKADIR, S. J., ALMOMANI, M. A., ADEYEMO, V. E., AL-TASHI, Q., MOJEED, H. A., IMAM, A. A., & BAJEH, A. O. 2020. Impact of feature selection methods on the predictive performance of software defect prediction models: an extensive empirical study. *Symmetry*, *12*(7), 1147.

BISHOP, C. M. 2006. *Pattern recognition and machine learning*. springer.

CHAWLA, N. V, BOWYER, K. W., HALL, L. O., & KEGELMEYER, W. P. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, *16*, 321–357.

GRAY, D., BOWES, D., DAVEY, N., SUN, Y., & CHRISTIANSON, B. 2012. Reflections on the NASA MDP data sets. *IET software*, *6*(6), 549–558.

HAMMAD, M., ALQADDOUMI, A., & AL-OBAIDY, H. 2019. Predicting software faults based on k-nearest neighbors classification. *International Journal of Computing and Digital Systems*, *8*(5), 462–467.

IEEE. 2010. *IEEE standard classification for software anomalies* (IEEE Std 1044-2009). IEEE.

JAYANTHI, R., & FLORENCE, L. 2019. Software defect prediction techniques using metrics based on neural network classifier. *Cluster Computing*, *22*, 77–88.

JI, H., HUANG, S., WU, Y., HUI, Z., & ZHENG, C. 2019. A new weighted naive Bayes method

based on information diffusion for software defect prediction. *Software Quality Journal*, *27*(3), 923–968.

KENNEDY, J., & EBERHART, R. 1995. Particle swarm optimization. *Proceedings of ICNN'95-international conference on neural networks*, *4*, 1942–1948.

KUMAR, H., & DAS, H. 2023. Software Fault Prediction using Wrapper based Feature Selection Approach employing Genetic Algorithm. *2022 OPJU International Technology Conference on Emerging Technologies for Sustainable Development (OTCON)*, 1–7.

MALHOTRA, R., SHAKYA, A., RANJAN, R., & BANSHI, R. 2021. Software defect prediction using Binary Particle Swarm Optimization with Binary Cross Entropy as the fitness function. *Journal of Physics: Conference Series*, *1767*(1), 12003.

MARIAN, Z., MIRCEA, I.-G., CZIBULA, I.-G., & CZIBULA, G. 2016. A novel approach for software defect prediction using fuzzy decision trees. *2016 18th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, 240–247.

MIRJALILI, S., MIRJALILI, S. M., & LEWIS, A. 2014. Grey wolf optimizer. *Advances in engineering software*, *69*, 46–61.

PEDREGOSA, F., VAROQUAUX, G. G., GRAMFORT, A., MICHEL, V., THIRION, B., GRISEL, O., BLONDEL, M., PRETTENHOFER, P., WEISS, R., DUBOURG, V., VANDERPLAS, J., PASSOS, A., COURNAPEAU, D., BRUCHER, M., PERROT, M., & DUCHESNAY, É. 2011. Scikit-learn: Machine learning in Python. *Journal of machine learning research*, *12*, 2825–2830.

RAWAT, M. S., & DUBEY, S. K. 2012. Software defect prediction models for quality improvement: a literature study. *International Journal of Computer Science Issues (IJCSI)*, *9*(5), 288.

RONG, X., LI, F., & CUI, Z. 2016. A model for software defect prediction using support vector machine based on CBA. *International Journal of Intelligent Systems Technologies and Applications*, *15*(1), 19–34.

SHEPPERD, M., SONG, Q., SUN, Z., & MAIR, C. 2013. Data quality: Some comments on the nasa software defect datasets. *IEEE Transactions on Software Engineering*, *39*(9), 1208–1215.

SISWANTORO, M. Z. F. N., & YUHANA, U. L. 2023. Software Defect Prediction Based on Optimized Machine Learning Models: A Comparative Study. *Teknika*, *12*(2), 166–172.

VAN THIEU, N., & MIRJALILI, S. 2023. MEALPY: An open-source library for latest meta-heuristic algorithms in Python. *Journal of Systems Architecture*, *139*, 102871. https://doi.org/https://doi.org/10.1016/j.sysarc.2023.102871

WAHONO, R. S., & HERMAN, N. S. 2014. Genetic feature selection for software defect prediction. *Advanced Science Letters*, *20*(1), 239–244.

WHITLEY, D. 1994. A genetic algorithm tutorial. *Statistics and computing*, *4*, 65–85.

Halaman ini sengaja dikosongkan