Nigerian Journal of Operations Research (NIJOR)

https://nijor.org.ng/
pISSN: 2141-7938
eISSN: 3156-5956

Jaya-Optimized Support Vector Regression for Software Effort Estimation using Mutual Information Feature Selection

Asaju La'aro Bolaji[1], Jephthah Job Agyo[1, *], Ishaku Wreford Andrew[1],

[1] Department of Computer Science, Faculty of Computing and Information Systems, Federal University Wukari, Wukari, Taraba State, Nigeria.

ARTICLE INFO

Article history:
Received 19 August2026
Revised form 1 September 2026
Accepted 9 September 2026
Available online 27 September 2026

Keywords:

Software Effort Estimation; Support Vector Regression; Jaya Algorithm; Mutual Information; Feature Selection

ABSTRACT

Software effort estimation remains one of the most challenging tasks in project management, prompting researchers to explore advanced machine learning techniques for improving prediction accuracy. While many existing studies focus either on feature selection or model optimization, only a limited number have investigated the combined impact of these two approaches. To address this gap, this study proposes **SVR-JAYA FS**, a hybrid model that integrates Mutual Information-based feature selection with Jaya algorithm-based hyperparameter optimization for Support Vector Regression (SVR) in software effort estimation. The model was evaluated on twelve benchmark datasets from the PROMISE and GITHUB repositories using ten performance metrics and compared with a standard SVR model, an optimized SVR-JAYA FS model, SSA-BPNN, and six classical regression techniques. Results show that SVR-JAYA FS consistently outperforms the standard SVR across all datasets, achieving R^2 improvements of up to 99.45%, while also reducing execution time by as much as 55%. Compared with SSA-BPNN and classical regression methods, the proposed model demonstrated superior or competitive performance on most datasets. The findings indicate that integrating feature selection and hyperparameter optimization significantly improves both prediction accuracy and computational efficiency, particularly for datasets with a larger number of features, making SVR-JAYA FS a robust and practical approach for software effort estimation.

[*] *Corresponding author.*
E-mail address: jephthahjob@outlook.com

https://doi.org/10.67868/nijorxxxxx

1. Introduction

Software has become a very important part of modern society, influencing nearly every aspect of our daily lives. The accurate estimation of the effort required for software development has become an important concern in software engineering for both researchers and practitioners, this is because software projects continue to increase in size and complexity (Sahu *et al.*, 2021). When it comes to project planning, particularly in determining cost of development, resource requirements, and project

schedules, Software Effort Estimation (SEE) plays a significant role (Kassaymeh *et al.,* 2021; Schwalbe, 2015). Consequently, SEE has remained an active area of study, resulting in the emergence of numerous predictive techniques aimed at improving estimation accuracy (Kassaymeh *et al.,* 2021). This study contributes to this growing domain of knowledge by exploring an effective predictive approach for SEE.

There are generally three categories of prediction methods: statistical-based, experience-based, and machine learning (ML) based methods (Yasmin *et al.,* 2024). Experience-based methods are based on human domain expert judgment drawn from years of experience (Meharunnisa *et al.,* 2023). Statistical methods use historical data and mathematical equations, such as regression and parametric models, to predict effort. The issue with this method is that its accuracy often declines with dynamic datasets (Mustyala *et al.,* 2025). While ML techniques such as Artificial Neural Networks (ANN), Backpropagation Neural Networks (BPNN), and others exploit input-output data relationships (Priya *et al.,* 2024).

Support Vector Regression (SVR) is widely utilized due to its strong generalization capability and flexibility; however, its performance is highly dependent on the appropriate hyperparameter tuning and feature selection. Accordingly, this study adopts SVR as the base prediction model and integrates a JAYA algorithm for hyperparameter tuning with a feature selection technique to enhance estimation accuracy.

Optimization and model-improvement techniques, such as parameter tuning and feature selection, are generally used to improve the efficiency and performance of predictive models. Metaheuristic algorithms, in particular, have been used to improve machine learning models across a wide range of prediction tasks. For example, Bolaji *et al.* (2018) applied the Fireworks Algorithm (FWA) to train artificial neural networks for pattern classification tasks, the network's weights and biases were optimized using FWA. Xue *et al.* (2020) improved grouting-power forecasting by combining SVR with an improved JAYA algorithm (JAYA), which optimized the SVR's hyperparameters using tent chaotic mapping and Lévy flights. Al Bakal *et al.* (2022) applied the Crow Search Algorithm to fine-tune SVR hyperparameters, outperforming grid search in accuracy and efficiency. Shobha *et al.* (2024) joined SVR and CatBoost with Particle Swarm Optimization (PSO) and Grey Wolf Optimizer (GWO) to improve landslide susceptibility prediction. In a similar machine learning study, Baba *et al.* (2026) introduced a feature selection framework based on variance analysis and k-means clustering for network intrusion detection that lowered model training and inference durations, notably for XGBoost and Random Forest models. These studies highlight the use of optimization and feature-selection strategies to enhance machine learning models in different domains, giving a broader rationale for using JAYA, a metaheuristic algorithm to optimize SVR hyperparameters and feature selection in this study.

While previous studies on software effort estimation have largely investigated feature selection and hyperparameter optimization independently, limited attention has been given to their combined application. This study addresses this gap by exploring the impact of integrating both JAYA and the mutual information feature selection technique within a Support Vector Regression (SVR) model to improve model's prediction performance.

2. Literature Review

Long et al. (2026) proposed a hybrid deep learning approach for Software Effort Estimation (SEE) by integrating an Inception Network with an Enhanced Banyan Tree Growth Optimizer (EBTGO). To extract hierarchical features from high-dimensional software project data the Inception Network was used, while EBTGO was used to fine tune the model hyperparameters to improve

predictive performance. Using the Maxwell and COCOMO81 datasets, the proposed approach was evaluated and was reported to outperform existing state-of-the-art methods across the evaluated performance metrics, including Mean Absolute Error (MAE) and Mean Magnitude of Relative Error (MMRE). The findings demonstrate the potential of combining deep learning architectures with metaheuristic optimization for improving software effort estimation.

Badana *et al.* (2025) proposed a hybrid deep learning framework for Software Effort Estimation (SEE) that integrates metaheuristic optimization with a Multilayer Long Short-Term Memory (M-LSTM) network. The study stressed the need of precise effort estimation for optimal project scheduling, resource allocation, and software development cost and time savings. The authors applied data preprocessing and feature extraction techniques to improve the quality of the input data and remove non-informative information and capture relevant statistical characteristics, respectively. Before performing effort estimation using the M-LSTM model, they further developed a Modified Chaotic Enriched Jaya with Moth Flame Optimization (MCEJ-MFO) algorithm for high-precision feature selection. Experimental results on the China and Maxwell datasets produced MSE values of 0.2825 and 0.2285, respectively, showing the application of hybrid metaheuristic optimization and deep learning to software effort estimation.

A hybrid approach that combines Particle Swarm Optimization (PSO) with Linear Regression (LR) to improve software effort estimation accuracy is proposed (Jayadi *et al.*, 2024). Recognizing the limitations of conventional LR in producing reliable estimates, the usage of PSO is employed to optimize the regression model and evaluate the approach using the COCOMO81, COCOMO_Nasa_v1, and COCOMO_Nasa_v2 datasets. The results showed that the PSO-optimized LR model consistently outperformed the standard LR model across all three datasets.

An advanced Software Reliability Growth Model (SRGM) that incorporates environmental factors and dependent failures to improve software reliability prediction is developed by Hussain et al. (2024). The model employs a non-homogeneous Poisson process (NHPP) framework, with the Cuckoo Search (CS) algorithm used for parameter estimation. Experimental results, supported by two software defect datasets and simulation studies, showed that the proposed model outperformed existing NHPP-based models and remained effective under varying operating conditions.

Kassaymeh *et al.* (2021) proposed a software reliability prediction model that combines the Salp Swarm Algorithm (SSA) with a Backpropagation Neural Network (BPNN). The primary aim of the study was to optimize the network's weights and biases using SSA, thereby improving the predictive capability of the BPNN. Experimental results demonstrated that the hybrid SSA-BPNN model consistently achieved better performance than the conventional BPNN across all evaluated datasets. Sakhrawi *et al.* (2021) conducted a systematic mapping study (SMS) to analyze the application of machine learning (ML) techniques in software enhancement effort prediction (SEME). 30 primary studies published between 1995 and 2020 were covered by the review, revealing that relatively few studies have focused on SEME compared to development effort prediction. SEME was framed as a regression problem by most of the reviewed works, with Support Vector Regression (SVR), statistical regression, and stochastic gradient boosting (SGB) being the most common approaches.

3. Methodology

3.1 Research Design

This study adopts an experimental and quantitative research design to evaluate the effect of combining mutual information (MI)-based feature selection and Jaya algorithm-based hyperparameter optimization on the predictive performance of Support Vector Regression (SVR) for software effort estimation (SEE). Three model configurations are compared under controlled conditions using twelve

benchmark datasets: SVR (Baseline): trained on all features with default hyperparameters; SVR-JAYA: trained on all features with Jaya-optimized hyperparameters; and SVR-JAYA FS (Proposed): trained on MI-selected features with Jaya-optimized hyperparameters. The experimental procedure follows three phases: (i) data collection and preprocessing, (ii) feature selection and model development, and (iii) performance evaluation and comparative analysis. Fig. 1 shows the overall model development process, which provides a step-by-step representation of the three stages.

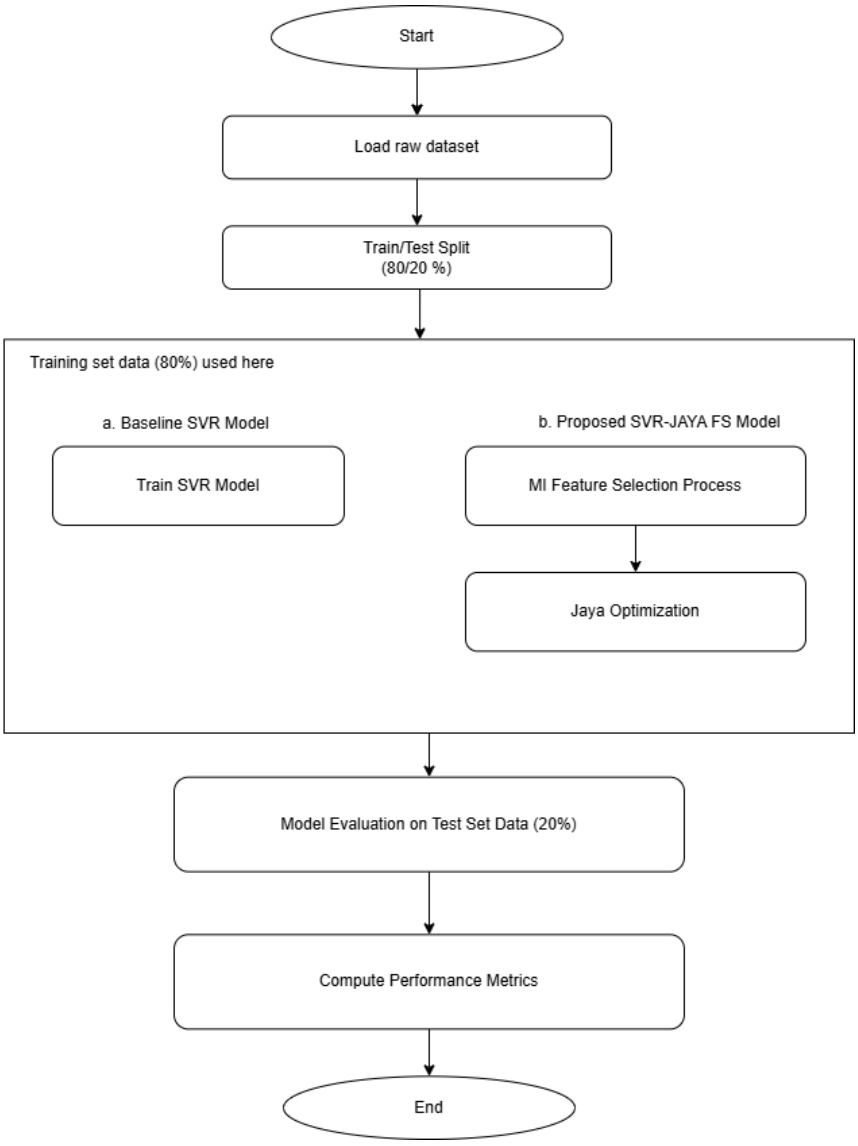


Fig. 1: Unified SVR-MI-Jaya model development process

3.2 Tools and Technologies

For this study, the following were used: Python (version 3.11.4) with Scikit-learn library, Pandas, and NumPy. The experiment was carried out on a 64-bit operating system, an x64-based processor HP laptop with the following specifications: 4 GB of RAM, Intel(R) Core(TM) i3-5005U CPU 2.00GHz, Intel(R) HD Graphics 5500 (128 MB), 932 GB HDD TOSHIBA MQ04ABF100.

3.3 Data Splitting Strategy

Before normalization, feature selection, or hyperparameter optimization, each dataset was divided into training (80%) and holdout test (20%) subsets using a fixed random state (random_state = 42) to guaranty an objective assessment and avoid information leakage between model training and testing. A three-fold cross-validation scheme was used independently at two stages within the training partition: (1) during Mutual Information-based feature selection to determine the ideal number of features (k) by assessing candidate feature subsets solely on training folds, and (2) during Jaya-based hyperparameter optimization to assess each candidate SVR configuration's fitness (MSE). The holdout test set was only used for final model evaluation and was not accessed during feature selection or hyperparameter tuning; all scaling, ranking, and model-fitting activities were carried out solely within the training partition.

3.4 Datasets

These datasets were obtained from two repositories: PROMISE and GitHub. As shown in Table 1 below, the datasets vary in size and complexity, with each entry detailing the dataset name, number of features, dimensions, time unit of the predicted project, and the source.

Table 1: SEE datasets summary

Dataset	**Features**	**N (samples)**	**Unit**	**Source**
Albrecht	7	24	Months	PROMISE
China	17	499	Hours	PROMISE
Cosmic	11	42	Months	PROMISE
COCOMO 81	17	63	Months	PROMISE
COCOMONASA I	17	60	Months	PROMISE
COCOMONASA II	24	93	Months	PROMISE
Desharnais	12	81	Hours	GitHub
Kemerer	6	15	Months	PROMISE
Kitchenham	7	145	Months	PROMISE
Maxwell	27	62	FP	PROMISE
Miyazaki	8	48	Months	PROMISE
USP05	8	203	Months	GitHub

Source: tera-PROMISE Category: Effort (openscience.us)
N denotes the total number of samples (observations) in each dataset

It is worthy to mentioned that two preprocessing steps are applied to all datasets before model training:

i. Normalization: Min-Max scaling is applied to all numerical features, mapping values to the [0, 1] range to prevent high-magnitude features from dominating the learning process.

ii. Categorical Encoding: Categorical attributes were converted to numeric form using one-hot encoding to ensure compatibility with SVR.

3.5 Mutual Information

Mutual Information (MI) is a non-parametric measure of dependency between two random variables; in this case, a feature X_i and the target variable Y. It quantifies how much information the feature contributes to predicting the target, regardless of linearity. Mathematically, it is defined as:

$$I(X_i;Y) = \int_Y \int_{X_i} p\,(x_i,y)\log\left(\frac{p(x_i,y)}{p(x_i)p(y)}\right) dx_i\,dy \quad (1)$$

Where:

$p(x_i,y)$ is the joint probability density of X_i and Y

$p(x_i)$ and $p(y)$ are the marginal probabilities of X_i and Y, respectively

The insight behind this expression is that if X_i and Y are independent, then

$$p(x_i,y) = p(x_i)p(y), \quad (2)$$

and thus $I(X_i;Y) = 0$.

Conversely, the more predictive the feature is for the target the higher the MI value.

The major steps are as follows: Initialization, Cross-Validation for Optimal Feature Count, Determining the Optimal Number of Features: The optimal number of features k^* is chosen as:

$$k^* = \text{argmin}_k \overline{MSE_k} \quad (3)$$

where $\overline{MSE_k}$ is the mean cross-validation MSE gotten for feature count k. This makes sure that the feature subset that yields the lowest validation error is selected.

As stated in Section 3.3, this cross-validation is performed exclusively within the training data.

3.6 Formulation of SVR

Given a training dataset:

$$D = (x_i, y_i)_{i=1}^N \quad (4)$$

where $x_i \in \mathbb{R}^d$ represents the feature vector, and $y_i \in \mathbb{R}$ the corresponding target, the SVR function is defined as:

$$f(x) = \langle w, \phi(x)\rangle + b \quad (5)$$

where:

w is the weight vector,

$\phi(x)$ is a nonlinear transformation mapping the input into a higher-dimensional feature space

b is the bias term.

The aim is to find w and b such that the predictions are within a deviation of ϵ from the actual values while maintaining model flatness (i.e., minimizing$\| w \|^2$).

SVR optimization problem can be expressed as:

$$\min_{w,b,\xi_i,\xi_i^*} \frac{1}{2} \| w \|^2 + C \sum_{i=1}^{N} (\xi_i + \xi_i^*)$$

subjecte to:

$$\begin{aligned} y_i - \langle w, \phi(x_i) \rangle - b &\leq \epsilon + \xi_i \\ \langle w, \phi(x_i) \rangle + b - y_i &\leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* &\geq 0, \quad i = 1, \ldots, N \end{aligned} \tag{6}$$

where:

$C > 0$ is the regularization parameter controlling the trade-off between model flatness and tolerance for deviations

ξ_i, ξ_i^* are slack variables representing deviations outside the ϵ-tube.

In this study, the Radial Basis Function (RBF) kernel is employed, defined as:

$$K(x_i, x_j) = \exp(-\gamma \| x_i - x_j \|^2) \tag{7}$$

where γ is the kernel parameter that controls the influence of individual training sample.

A preliminary analysis was performed to determine suitable ranges of values for the SVR hyperparameters. For the regularization parameter C, a structured range of values [1, 5, 10, 20, 30, …, 100] was evaluated. For the epsilon (ε) and kernel coefficient (γ) parameters, several values within the intervals 0.01 - 0.5 and 0.001 - 0.05, respectively were sampled. Multiple SVR parameter configurations were tested using different combinations of these values, and the configuration C = 100, ε = 0.1, and γ = 0.05 was selected based on its consistent performance across datasets.

3.7 Jaya Algorithm for Hyperparameter Optimization

The Jaya algorithm is a parameter-free population-based metaheuristic that iteratively improves candidate solutions by moving them toward the best solution and away from the worst. The update rule for each candidate hyperparameter is:

$$X_{new} = X_{cur} + r_1(X_{best} - |X_{cur}|) - r_2(X_{worst} - |X_{cur}|) \tag{8}$$

Where X_{cur} represents the current hyperparameter value, X_{best} is the best-performing solution in the population, X_{worst} is the worst-performing solution in the population and lastly r_1 and r_2 are random numbers between 0 and 1.

Support Vector Regression (SVR) is effective in capturing non-linear patterns in software effort data, but its ability to make predictions depends on how well its hyperparameters are tuned, which is a problem well suited for metaheuristic optimization. Because of its distinctive parameter-free architecture, the Jaya Algorithm was chosen among potential algorithms such as Particle Swarm Optimization (PSO), Differential Evolution (DE), and Gray Wolf Optimizer (GWO). Jaya algorithm

requires only population size and generation limit to be defined, in contrast to other metaheuristics algorithms that require careful calibration of algorithm-specific control factors, such inertia weights or crossover rates. This eliminates parameter misconfiguration risks while ensuring robust, computationally efficient SVR hyperparameter convergence.

Based on the preliminary analysis for selecting the base configurations of the SVR parameters presented in section 3.6 above, the Jaya search bounds were defined as C ∈ [1, 1000], ε ∈ [0.01, 0.5], and γ ∈ [0.0001, 0.1]. The upper bound for C was extended beyond the manually evaluated range to allow Jaya to explore larger values that were not covered by the preliminary grid. Similarly, the search interval for γ was expanded from the manually sampled range to reduce the risk of unnecessarily restricting the optimization to the initially explored region. The ε interval was retained at [0.01, 0.5] because variations in ε within this range did not produce substantial changes in model performance during the preliminary analysis. Thus, the final search space was empirically informed by the preliminary parameter analysis while providing Jaya with sufficient freedom to explore configurations beyond those identified through manual search.

Algorithm 1: Jaya Algorithm for SVR Parameter Optimization

Step 1: Initialize the population P consisting of N candidate hyperparameter sets $X_1, X_2, \ldots, X_n$.

Step 2: Evaluate the fitness $f(X_i)$ of each candidate X_i by training the SVR model and computing its MSE performance metric.

Step 3: Determine the best X_{best} and worst X_{worst} solutions based on evaluation scores.

Step 4: Update each candidate solution using equation 8

Step 5: Repeat steps 2-4 until convergence criteria (∈) are met or a predefined number of iterations is reached.

Step 6 Return the best hyperparameter set X_{best}.

3.8 Performance Evaluation Metric

To evaluate the prediction accuracy of the SVR model, several error and accuracy metrics were used. Let X_i denote the actual value, Y_i the predicted value, $\bar{X}$ the mean of actual values, and n the number of data points.

Table 2: Evaluation Metrics and Formulars

Metric	Formular	
Mean Squared Error (MSE)	$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(X_i - Y_i)^2$	(9)
Root Mean Squared Error (RMSE)	$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(X_i - Y_i)^2}$	(10)
Relative Absolute Error (RAE)	$\text{RAE} = \frac{\sum_{i=1}^{n}\lvert X_i - Y_i\rvert}{\sum_{i=1}^{n}\lvert X_i - \bar{X}\rvert}$	(11)

Metric	Formular	
Root Relative Squared Error (RRSE)	$\text{RRSE} = \sqrt{\frac{\sum_{i=1}^{n}(X_i - Y_i)^2}{\sum_{i=1}^{n}(X_i - \bar{X})^2}}$	(12)
Mean Absolute Error (MAE)	$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} \lvert X_i - Y_i \rvert$	(13)
Mean Magnitude of Relative Error (MMRE)	$\text{MMRE} = \frac{1}{n}\sum_{i=1}^{n} \frac{\lvert X_i - Y_i \rvert}{X_i}$	(14)
Median Magnitude of Relative Error (MdMRE)	$\text{MdMRE} = \text{median}\left(\frac{\lvert X_i - Y_i \rvert}{X_i}\right)$	(15)
Variance Accounted For (VAF)	$\text{VAF} = \left(1 - \frac{\text{Var}(X - Y)}{\text{Var}(X)}\right) \times 100\%$	(16)
Coefficient of Determination (R^2)	$R^2 = 1 - \frac{\sum_{i=1}^{n}(X_i - Y_i)^2}{\sum_{i=1}^{n}(X_i - \bar{X})^2}$	(17)
Euclidean Distance (ED)	$\text{ED} = \sqrt{\sum_{i=1}^{n}(X_i - Y_i)^2}$	(18)
Manhattan Distance (MD)	$\text{MD} = \sum_{i=1}^{n} \lvert X_i - Y_i \rvert$	(19)

3.9. Sensitivity of MMRE to Epsilon Value for the Proposed Model

A preliminary single-run sensitivity assessment was performed on the USP05 dataset, which was chosen because of its substantially right-skewed target distribution, containing near-zero actual values (with min value = 0.50, max value = 400.00), by assessing MMRE over a range of epsilon values in order to establish an adequate smoothing factor (Table 3). The extreme MMRE surpassing 16 million was generated by the unsmoothed setup ($\varepsilon = 1\times10^{-10}$). This was drastically decreased to about 6.0 when ε was increased to 0.001; subsequent increases to 0.005 and 0.01 produced relatively little more change (1.99 and 1.92, respectively), suggesting that MMRE stabilizes once ε reaches this range. In order to avoid distorting MMRE for datasets lacking near-zero targets, ε = 0.01 was chosen since it falls inside this stable area while remaining tiny.

Table 3: Sensitivity of MMRE to Epsilon Value (USP05, Single Run)

Epsilon (ε)	MMRE
1×10^{-10}	16,098,375.00
0.001	5.9968
0.005	1.9879
0.01	1.9173

4. Results and discussions

The experiment on each dataset was carried out 30 different times, evaluated using multiple performance metrics, and the best results in terms of the fitness function (MSE metric) are selected

and used for comparison against other methods. The results observed are discussed in detail in the following sections.

4.1 Comparison Between the Three Experimental Configurations

The comparative analysis results presented in Table 4 show the performance between the SVR-JAYA model and SVR-JAYA FS model across all twelve software datasets. An important insight that emerged from a careful observation of Table 1 (Dataset Summary) alongside the experimental results in Table 4 is the relationship between dataset dimensionality and the effectiveness of Mutual Information-based feature selection. The mixed performance between the two models, as observed in Table 4, is directly explained through this relationship. Generally, many datasets with 10 or more features (China, Cocomo 81, Cocomonasa I, Maxwell) showed equal or improved performance under feature selection, whereas datasets with 8 or fewer features (Albrecht, Kemerer, Miyazaki, USP05) performed poorly when features were removed. However, three datasets exhibited a different pattern. Although the Cocomonasa II and Desharnais datasets contain 24 and 12 features, respectively, both experienced a decline in predictive performance. In contrast, the Kitchenham dataset, which has only 7 features, showed improved performance.

The poor performance of SVR-JAYA FS on COCOMO NASA II is partly due to feature selection. From 52 predictors (after applying one-hot encoding), Mutual Information (MI) selected only ~24-28 features per run. MI scores varied widely: features like *equivphyskloc* scored ~0.69, followed by *center_5*, *time_xh*, *center_2*, *tool_n*, and *acap_n*, while several features scored zero. Although this reduced dimensionality, some low-MI features likely carry complementary information that SVR needs when combined with other predictors. Their removal may have limited the model's ability to capture complex relationships in COCOMO NASA II, contributing to the poor performance of SVR-JAYA FS model.

This finding shows that feature selection can either add value or reduce value depending on the presence of redundant, noisy, or irrelevant attributes. A reduction in computational cost achieved by SVR-JAYA FS can also be observed in Table 3. Even in cases where feature selection performed quite bad, it reduced the execution time. Notable examples include: the China dataset (execution time was reduced from 1,990.76s to 542.67s, about 73% time improvement); the Cosmic dataset (reduction from 95.99s to 47.44s, about 51.0% improvement).

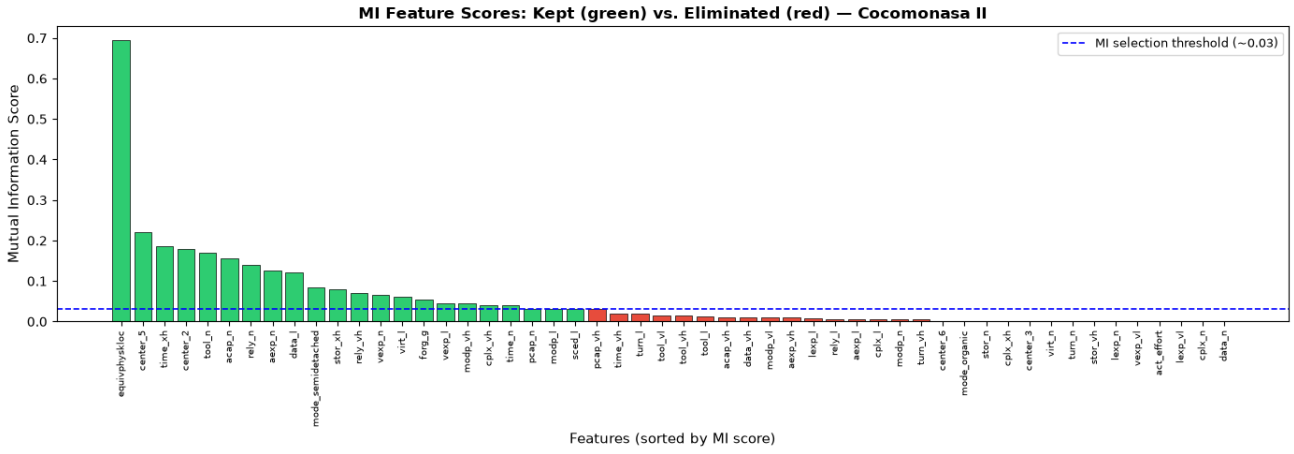


Fig. 2: MI Feature Selection for Cocomonasa II Dataset - Kept vs. Eliminated Features

Table 4: Proposed SVR-JAYA FS vs SVR-JAYA Performance Results

Dataset	Model	MSE	RMSE	RAE	RRSE	MAE	MMRE	VAF (%)	R^2 (%)	ED	MD	Time (s)
Albrecht	SVR-JAYA	**0.0032**	**0.0547**	0.1746	**0.1540**	0.0418	0.6253	80.10	98.58	0.375	0.503	34.94
	SVR-JAYA FS	0.004	0.0631	**0.1544**	0.1854	**0.0417**	**0.3107**	**97.70**	**99.02**	**0.1411**	**0.2084**	**29.51**
China	SVR-JAYA	0.0002	0.0134	0.1571	0.1368	0.0074	0.4942	97.85	98.41	0.1473	0.7421	1990.76
	SVR-JAYA FS	**0.0001**	**0.012**	**0.1424**	**0.1192**	**0.0069**	**0.1869**	**98.60**	**99.34**	**0.1197**	**0.6932**	**542.67**
Cocomo 81	SVR-JAYA	0.0035	0.0514	1.4451	1.3189	0.0457	7.0181	-43.67	53.21	0.2535	0.7605	115.67
	SVR-JAYA FS	**0.0007**	**0.0274**	**0.659**	**0.5607**	**0.0215**	**0.9292**	**68.6459**	**86.49**	**0.0986**	**0.2796**	**59.80**
Cocomonasa I	SVR-JAYA	0.0565	**0.2084**	0.5383	0.5606	0.1652	0.7871	68.75	83.61	0.8361	1.9822	68.83
	SVR-JAYA FS	**0.045**	0.2376	**0.429**	**0.5004**	**0.1316**	**0.418**	**76.146**	**93.24**	**0.7347**	**1.5796**	**35.87**
Cocomonasa II	SVR-JAYA	**0.0008**	**0.0272**	**0.6910**	**0.6663**	**0.0224**	2.3851	**58.47**	**83.61**	**0.1304**	**0.4624**	75.22
	SVR-JAYA FS	0.0019	0.044	1.0757	1.017	0.0349	**1.3392**	21.5	52.75	0.1917	0.6639	**45.12**
Cosmic	SVR-JAYA	0.0007	0.0262	0.2553	0.196	**0.0119**	**0.3168**	**98.48**	**99.50**	0.0785	0.1975	87.93
	SVR-JAYA FS	**0.0004**	**0.0212**	**0.2146**	**0.1587**	0.0184	0.3313	97.6166	98.82	**0.0635**	**0.166**	**47.44**
Desharnais	SVR-JAYA	**0.0083**	**0.0910**	0.6467	**0.5562**	0.0744	**1.4568**	**65.79**	**84.83**	**0.3754**	1.2647	72.92
	SVR-JAYA FS	0.0112	0.1061	**0.6396**	0.6915	**0.0736**	1.5707	52.8	76.62	0.4373	**1.2509**	**46.91**
Kemerer	SVR-JAYA	**0.0046**	**0.068**	**0.6323**	**0.8103**	**0.0476**	**0.4081**	53.92	82.27	**0.1178**	**0.1453**	34.66
	SVR-JAYA FS	0.0124	0.1116	1.2783	1.3287	0.0979	0.5379	**59.44**	**93.42**	0.1933	0.2938	**30.37**
Kitchenham	SVR-JAYA	0.8102	0.4633	0.4519	0.6796	0.2192	83.74	87.93	99.36	1.9243	3.8522	143.83

Dataset	Model	MSE	RMSE	RAE	RRSE	MAE	MMRE	VAF (%)	R^2 (%)	ED	MD	Time (s)
Maxwell	SVR-JAYA FS	**0.074**	**0.272**	**0.2142**	**0.2054**	**0.1039**	**0.7499**	**95.88**	**99.49**	**1.4648**	**3.0132**	**72.32**
	SVR-JAYA	0.0684	0.2616	0.5916	0.6184	0.1612	**1.4453**	62.62	84.63	0.9431	2.0954	84.56
	SVR-JAYA FS	**0.0459**	**0.2143**	**0.5561**	**0.5067**	**0.1515**	1.9928	**74.33**	**90.71**	**0.7728**	**1.9695**	**38.27**
Miyazaki	SVR-JAYA	0.0011	0.0335	**0.6258**	**0.621**	0.0243	0.8704	24.47	62.02	0.1059	0.2426	68.62
	SVR-JAYA FS	**0.0004**	**0.0203**	0.8134	0.7715	**0.0154**	**0.4268**	**41.18**	**68.97**	**0.0642**	**0.154**	**45.15**
USP05	SVR-JAYA	0.0357	0.1889	0.6524	0.657	0.0781	1.472	57.32	76.47	1.2093	3.2017	**63.91**
	SVR-JAYA FS	**0.0342**	**0.185**	**0.5978**	**0.6437**	**0.0716**	**1.2053**	**58.57**	**76.53**	**1.1848**	**2.934**	75.20

4.2 Comparing the SVR-JAYA FS Method with Other State-of-the-Art Methods for SEE

To further assess the efficacy of the proposed method, a series of three comparative analyses was conducted against eleven state-of-the-art methodologies. In Table 5 the abbreviations corresponding to the methods under comparison are presented. It is important to note that the performance metrics for these methodologies are derived from the work of Kassaymeh (2021) in the publication titled "Salp Swarm Optimizer for Modelling Software Reliability Prediction Problems" and all values are presented in a normalized form. These metrics are based on experiments conducted with the identical set of twelve benchmark datasets utilized in this study.

Table 5: State-of-the-art comparative methods for SEE

Abbreviation	Method name
ATLM	Automatically transformed linear model
RR	Ridge Regression
LASSO	Least absolute shrinkage and selection operator Regression
ENR	ElasticNet Regression
RoR	Robust Regression
SWR	Stepwise Regression
EBA+	Extension for Estimation by Analogy method
ANGEL	Software tool for project prediction
Analogy-X	Mantel's correlation randomization test
SSA-BPNN	Salp Swarm Algorithm with Backpropagation Neural Network

Source: http://dec.bmth.ac.uk/ESERG/ANGEL/

The first comparison is between the proposed method alongside the SSA-BPNN methods, evaluated using all ten metrics. The second comparison is the proposed method in relation to six advanced techniques, specifically using the MAE performance measure.

4.2.1 SVR-JAYA FS vs SSA-BPNN

Table 6: SVR-JAYA FS vs SSA-BPNN: SEP Performance Results

Dataset	Model	MSE	RMSE	RAE	RRSE	MAE	MMRE	VAF (%)	R^2 (%)	ED	MD
Albrecht	SSA-BPNN	0.0161	0.1271	0.3153	0.3652	0.0986	**0.1685**	86.67	86.66	3.6971	5.9146
	SVR-JAYA FS	**0.004**	**0.0631**	**0.1544**	**0.1854**	**0.0417**	0.3107	**97.70**	**99.02**	**0.1411**	**0.2084**
China	SSA-BPNN	0.0030	0.0546	0.3795	0.3792	0.0292	0.7453	85.70	85.62	0.6684	4.3815
	SVR-JAYA FS	**0.0001**	**0.012**	**0.1424**	**0.1192**	**0.0069**	**0.1869**	**98.5979**	**99.34**	**0.1197**	**0.6932**
Cocomo 81	SSA-BPNN	0.0160	0.1266	0.6870	**0.5528**	0.1007	**0.1010**	**69.67**	69.44	0.5519	1.9124
	SVR-JAYA FS	**0.0007**	**0.0274**	**0.659**	0.5607	**0.0215**	0.9292	68.6459	**86.49**	**0.0986**	**0.2796**
Cocomonasa I	SSA-BPNN	**0.0074**	**0.0861**	**0.3602**	**0.2993**	**0.0702**	1.6424	**91.04**	91.04	**0.3654**	**1.2635**
	SVR-JAYA FS	0.045	0.2121	0.429	0.5004	0.1316	**0.418**	76.1461	**93.24**	0.7347	1.5796
Cocononasa II	SSA-BPNN	0.0103	0.1013	**0.3664**	**0.4307**	0.0659	1.7229	**81.47**	**81.45**	0.5360	1.8446
	SVR-JAYA FS	**0.0019**	**0.044**	1.0757	1.017	**0.0349**	**1.3392**	21.5	52.75	**0.1917**	**0.6639**
Cosmic	SSA-BPNN	**1.34E-07**	**3.66E-04**	**9.19E-04**	**9.90E-04**	0.0287	**4.10E-04**	**100.00**	**99.99**	**7.32E-04**	**1.15E-03**
	SVR-JAYA FS	0.0004	0.0212	0.2146	0.1587	**0.0184**	0.3313	97.6166	98.82	0.0635	0.166
Desharnais	SSA-BPNN	0.0157	0.1251	**0.5098**	**0.4984**	0.0978	**0.8231**	**75.18**	75.16	0.6131	2.3470
	SVR-JAYA FS	**0.0112**	**0.1061**	0.6396	0.6915	**0.0736**	1.5707	52.8	**76.62**	**0.4373**	**1.2509**
Kemerer	SSA-BPNN	**1.62E-04**	**0.0127**	**0.0338**	**0.0335**	**0.0117**	**0.1086**	**99.8882**	**99.8877**	**0.0285**	**0.0583**

Kitchenham	SVR-JAYA FS	0.0124	0.1116	1.2783	1.3287	0.0979	0.5379	59.44	93.42	0.1933	0.2938
	SSA-BPNN	**0.0229**	**0.1512**	0.6721	0.6949	**0.1019**	1.8379	51.71	51.7055	**1.0030**	4.4849
Maxwell	SVR-JAYA FS	0.074	0.272	**0.2142**	**0.2054**	0.1039	**0.7499**	**95.88**	**99.49**	1.4648	**3.0132**
	SSA-BPNN	**0.0068**	**0.0824**	**0.3881**	**0.3504**	**0.0651**	**1.0482**	**87.7261**	87.7239	**0.3593**	**1.2362**
Miyazaki	SVR-JAYA FS	0.0459	0.2143	0.5561	0.5067	0.1515	1.9928	74.33	**90.71**	0.7728	1.9695
	SSA-BPNN	0.0035	0.0593	**0.2928**	**0.2444**	0.0423	0.5992	**94.06**	**94.03**	0.2295	0.6341
USP05	SVR-JAYA FS	**0.0004**	**0.0203**	0.8134	0.7715	**0.0154**	**0.4268**	41.18	68.97	**0.0642**	**0.154**
	SSA-BPNN	**0.0144**	**0.1201**	1.2103	0.9145	**0.0605**	8.7496	16.38	16.36	**0.9303**	3.6278
	SVR-JAYA FS	0.0342	0.185	**0.5978**	**0.6437**	0.0716	**1.2053**	**58.57**	**76.53**	1.1848	**2.934**

The results in Table 6 reveals that SVR-JAYA FS demonstrates clear superiority on Albrecht, China, and Cocomo 81, and achieves competitive results on Desharnais, Kitchenham, Miyazaki, and USP05 in terms of error-based metrics. However, SSA-BPNN outperforms the proposed model on the remaining five datasets.

4.2.2 SVR-JAYA FS vs six state-of-the-art methods

Note: Unlike Table 4 and Table 6, the MAE values reported here are presented on the normalized percentage scale. Specifically, they represent the mean of all 30 individual MAE values multiplied by 100, whereas Tables 4 and 6 report a single MAE value from the 30 independent runs. Consequently, the figures are not directly comparable. sourced from Kassaymeh (2021)

Table 7: Mean MAE (%) of SVR-JAYA FS and state-of-the-art methods across the twelve benchmark datasets, reported as the average performance over 30 independent runs.

Dataset	ATLM	RR	LASSO	ENR	RoR	SWR	SSA-BPNN	SVR-JAYA FS
Albrecht	15.2	15.0	16.8	14.3	15.0	91.6	**9.9**	13.9
China	33.3	33.2	32.9	30.7	38.4	81.3	2.9	**1.5**
Cocomo 81	25.5	17.8	11.1	36.6	11.2	43.4	10.1	**2.6**
Cocomo Nasa I	32.3	35.1	53.7	46.5	30.9	**5.5**	7.0	13.5
Cocomo Nasa II	50.2	49.6	77.1	63.6	48	32.4	6.6	**4.4**
Cosmic	49.6	49.6	34.5	49.6	51.2	51.2	2.87	**2.1**
Desharnais	41.6	43.4	67.8	36.3	42.4	72.0	9.8	**7.8**
Kemerer	33.7	16.8	15.4	41.3	35.6	35.4	**1.2**	10.9
Kitchenham	**6.4**	8.0	8.3	7.8	12.6	17.3	10.2	21.6
Maxwell	17.8	17.8	17.2	30.1	21.7	18.1	**6.5**	16.9
Miyazaki	10.5	12.0	12.9	12.9	21.4	22.7	4.2	**2.9**
USP05	**5.2**	5.7	84.7	30.6	87.9	36.6	6.0	15.2

Against the six regression-based methods (ATLM, RR, LASSO, ENR, RoR, SWR) evaluated using Mean MAE, SVR-JAYA FS achieved the best MAE on six out of twelve datasets. On datasets such as China, Cocomo 81, Cosmic, and Desharnais, the improvements over classical regression methods are substantial, with SVR-JAYA FS reducing MAE by 80-95% compared to some baselines. Hence, the proposed method is deemed suitable for implementation in software effort estimation processes over unoptimized regression-based methods.

4.3 Statistical Significance Analysis of Model Performance

A Wilcoxon Signed-Rank Test was conducted using the Mean Squared Error (MSE) achieved by the proposed SVR-JAYA FS model and SSA-BPNN model across the twelve SEE benchmark datasets to assess whether the performance difference between the models is statistically significant. A non-parametric paired test was considered appropriate given the small sample size (n = 12). The hypotheses tested were as follows:

H_0 (Null Hypothesis): There is no statistically significant difference in MSE between SVR-JAYA FS and SSA-BPNN across the twelve SEE datasets.

H_1 (Alternative Hypothesis): There is a statistically significant difference in MSE between SVR-JAYA FS and SSA-BPNN across the twelve SEE datasets.

The test was conducted at a significance level of $\alpha = 0.05$, two-tailed.

Table 8: Descriptive statistics of MSE across twelve SEE datasets (n = 12)

Model	N	Mean	Std. Dev.	Median	Minimum	Maximum
SSA-BPNN	12	0.009689	0.007357	0.008850	0.0000001	0.022900
SVR-JAYA FS	12	0.019183	0.024608	0.007600	0.000100	0.074000

Table 8 presents the descriptive statistics of the MSE values obtained by both models across the twelve datasets.

4.3.1 Wilcoxon Signed-Rank Test

Because of the small sample size, Monte Carlo exact significance testing (based on 12 sampled tables, 99% confidence interval) was used in addition to the asymptotic approximation, as recommended when sample sizes are limited. Table 9 presents the test results.

Table 9: Wilcoxon Signed-Rank Test results comparing SVR-JAYA FS and SSA-BPNN on MSE

Statistic	Value
Negative Ranks (SVR-JAYA FS < SSA-BPNN)	6 (Mean Rank = 4.67, Sum = 28.00)
Positive Ranks (SVR-JAYA FS > SSA-BPNN)	6 (Mean Rank = 8.33, Sum = 50.00)
Ties	0
Z	−0.863
Asymp. Sig. (2-tailed)	0.388
Monte Carlo Sig. (2-tailed)	0.417 (99% CI: 0.050 – 0.783)
Effect size ($r = Z / \sqrt{N}$)	0.25 (small-to-medium)

The null hypothesis was therefore retained because the Wilcoxon Signed-Rank Test indicated no statistically significant difference in MSE between SVR-JAYA FS and SSA-BPNN across the twelve datasets ($Z = -0.863$, $p = 0.417$, Monte Carlo, two-tailed; $\alpha = 0.05$). SVR-JAYA FS achieved a lower MSE than SSA-BPNN on six of the twelve datasets, and a higher MSE on the remaining six, with the associated effect size ($r = 0.25$) indicating a small-to-medium effect that did not reach statistical significance at this sample size.

4.3.2 Friedman Test and Nemenyi Post-Hoc analysis across all benchmark models

To evaluate whether the observed differences in Mean Absolute Error (MAE) among all eight compared models are statistically significant, a Friedman test was conducted using the mean MAE (%) values achieved across the twelve SEE benchmark datasets (Table 7) for ATLM, RR, LASSO, ENR,

RoR, SWR, SSA-BPNN, and the proposed SVR-JAYA FS. The Friedman test ranks each model's performance within every dataset and evaluates whether the average ranks differ significantly across models, making it appropriate for comparing multiple related models over the same set of datasets.

H_0 (Null Hypothesis): There is no statistically significant difference in the MAE distributions of the eight compared models across the twelve SEE datasets.

H_1 (Alternative Hypothesis): At least one model differs significantly from the others in terms of MAE distribution across the twelve SEE datasets.

Mean Ranks

Table 10: Mean ranks of MAE (%) across all eight models over twelve SEE datasets (n = 12).

Model	Mean Rank
SSA-BPNN	2.08
SVR-JAYA FS	2.25
ATLM	4.38
RR	4.50
LASSO	5.21
ENR	5.29
RoR	5.92
SWR	6.38

Friedman Test Result

Table 11: Friedman test statistics across all eight benchmark models (n = 12).

Statistic	Value
N (datasets)	12
Test Statistic (χ^2)	35.420
Degrees of Freedom (df)	7
Asymptotic Sig. (2-tailed)	< 0.001

The null hypothesis was rejected because the Friedman test produced a statistically significant result (χ^2 (7) = 35.420, $p < 0.001$). This shows that the eight compared models do not perform equivalently in terms of MAE across the twelve SEE datasets. The two lowest best mean ranks were obtained by SSA-BPNN (mean rank = 2.08) and the proposed SVR-JAYA FS (mean rank = 2.25), clearly separated from the six classical regression baselines, having their mean ranks ranged from 4.38 (ATLM) to 6.38 (SWR).

A Nemenyi post-hoc test was conducted to identify which pairs of models differ significantly in mean rank because the Friedman test was significant. The Nemenyi test establishes a Critical

Difference (CD): two models are considered significantly different if the absolute difference between their mean ranks exceeds the CD.

Fig. 3 presents the resulting critical difference diagram. Models connected by a thick horizontal bar are not significantly different from one another; models not connected by a shared bar differ significantly in mean rank at $\alpha = 0.05$.

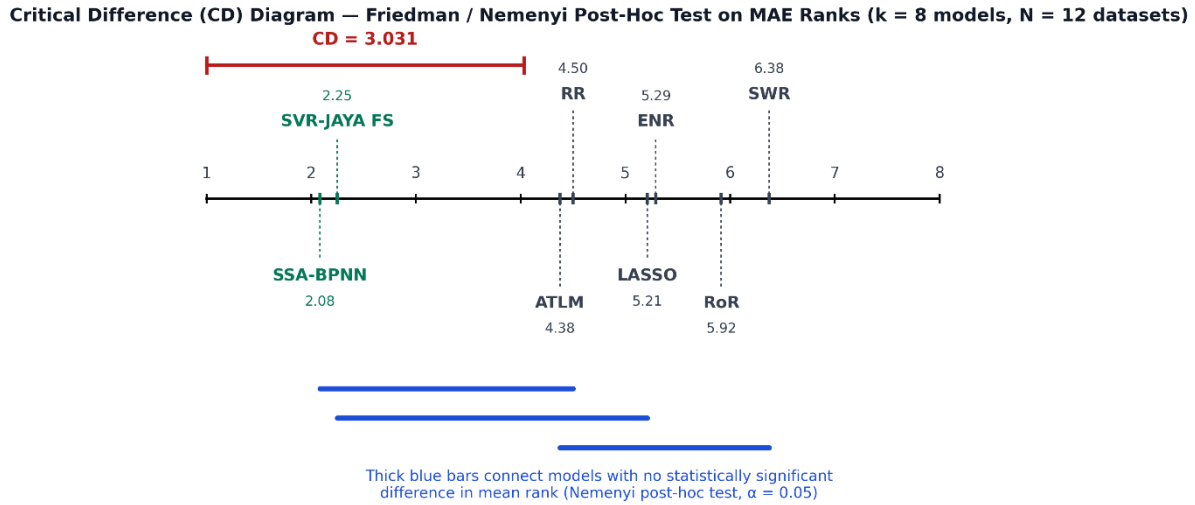


Fig 3: Critical difference (CD) diagram for the Friedman/Nemenyi post-hoc test on mean MAE ranks (k = 8, N = 12, CD = 3.031).

The CD diagram shows that SVR-JAYA FS is not significantly different in rank from SSA-BPNN ($|2.25 - 2.08| = 0.17$), ATLM (2.13), RR (2.25), or LASSO (2.96), as these differences all fall below the critical difference of 3.031. However, SVR-JAYA FS differs significantly from RoR (3.67) and SWR (4.13), and marginally from ENR (3.04, only fractionally above the CD threshold). These findings were corroborated by the Bonferroni-corrected pairwise comparisons produced alongside the Friedman test in SPSS, which similarly identified SVR-JAYA FS as significantly outperforming RoR (Adj. Sig. = 0.007) and SWR (Adj. Sig. = 0.001), with the comparison against ENR falling just short of significance under the more conservative Bonferroni correction (Adj. Sig. = 0.066), consistent with its borderline status in the Nemenyi test.

Taken together, the Friedman and Nemenyi results indicate that SVR-JAYA FS and SSA-BPNN form a distinct, top-performing group that is statistically indistinguishable from one another, and significantly outperforms the weaker classical baselines (RoR and SWR), while performing comparably to the stronger classical baselines (ATLM, RR, and LASSO) in terms of MAE rank across the twelve SEE datasets.

5. Conclusion and recommendation

There are four key conclusions that results of this study led to. First, SVR predictive accuracy across all datasets consistently and substantially improves as a result of the Jaya-based hyperparameter optimization, confirming it as a critical component regardless of dataset characteristics. The second conclusion is that, MI-based feature selection improves accuracy on high-dimensional datasets (for this study, datasets with ten or more features, e.g., China, Cocomo 81, Kitchenham) by removing irrelevant features, but it degrades performance on low-dimensional datasets (for this study, datasets with fewer than ten features, e.g., Albrecht, Kemerer, Miyazaki), where each feature carries unique

predictive signal. The third conclusion is that SVR-JAYA FS delivers consistent computational gains, by reducing execution time by over 40% on average across several datasets, this is a practical advantage for time-sensitive project planning. Lastly, it is concluded that the proposed SVR-JAYA FS model when compared against some state-of-the-art methods is very competitive, achieving superior MAE on seven of ten datasets versus classical regression methods, and remaining competitive with SSA-BPNN despite it having a simpler architecture to the SSA-BPNN.

The following recommendations are proposed based on the findings of this study. This study used only the Jaya algorithm for the optimization of SVR hyperparameter, future research should conduct systematic comparisons between Jaya and other competitive metaheuristics algorithms. Future research should also investigate other feature selection methods that account for feature interactions, potentially yielding better feature subsets than MI-based feature selection methods.

Author Contributions

Conceptualization, J.J.A; methodology, J.J.A; software, I.W.A and J.J.A; validation, J.J.A and A.L.B; formal analysis, J.J.A and A.L.B; investigation, J.J.A; resources, A.L.B; data curation, J.J.A; writing—original draft preparation, J.J.A; writing—review and editing, A.L.B and I.W.A; supervision, A.L.B; project administration, A.L.B. All authors have read and agreed to the published version of the manuscript.

Funding

This research received no external funding

Data Availability Statement

The dataset used for this research were obtained from two repositories: PROMISE and GitHub.
Source: [tera-PROMISE Category: Effort (openscience.us)]

Conflicts of Interest

The authors confirm that they have no financial interests or personal relationships that could have influenced the work presented in this paper.

Acknowledgement

This research was not funded by any grant

REFERENCES

Al Bakal, S. M., Alrefaee, S. D., & Algamal, Z. Y. (2022). Hyper parameters selection by using metaheuristic algorithm for improving support vector regression. *Mathematical Statistician and Engineering Applications (MSEA).* https://doi.org/10.17762/msea.v71i3.496

Baba, U., Garba, E. J., Ahmadu, A. S., & Lapina, M. (2026). Towards Real-Time Network Intrusion Detection: An Enhanced Feature Selection Framework Using Variance Analysis and K-Means Clustering. Nigerian Journal of Operations Research, 3(2), 211-234. https://doi.org/10.67868/2dkjpz49

Badana, M., Kumar, B.K., Sundari, M.R. et al. A Novel Hybrid Deep Learning Framework with Metaheuristic Optimization for Accurate Software Effort Estimation. SN COMPUT. SCI. 6, 916 (2025). https://doi.org/10.1007/s42979-025-04459-3

Bolaji, A. L. A., Ahmad, A. A., & Shola, P. B. (2018). Training of neural network for pattern classification using fireworks algorithm. *International Journal of System Assurance Engineering and Management, 9*(1), 208-215.

Hussain, A., Oraibi, Y., Mashikhin, Z., Jameel, A., Tashtoush, M., & Az-Zo, E. A. (2024). New Software Reliability Growth Model: Piratical Swarm Optimization-Based Parameter Estimation in Environments with Uncertainty and Dependent Failures. *Statistics, Optimization & Information Computing,* 13(1), 209–221. https://doi.org/10.19139/soic-2310-5070-2109

Jayadi, P., Ahmad, K. A., Cahyo, R. Z. D., & Aldida, J. D. (2024). Particle Swarm Optimization-based Linear Regression for Software Effort Estimation. *Journal of Information System, Technology and Engineering*, 2(2), 261–268. https://doi.org/10.61487/jiste.v2i2.69

Kassaymeh, S., Abdullah, S., Al-Laham, M., Alweshah, M., Al-Betar, M. A., & Othman, Z. (2021). Salp swarm optimizer for modeling software reliability prediction problems. *Neural Processing Letters*. 53, 6, 4451-4487. https://doi.org/10.1007/s11063-021-10607-6

Long, W., Qixin, Z. & Luxia, Y. Software effort estimation based on inception network optimized by enhanced banyan tree growth optimizer. Sci Rep 16, 143 (2026). https://doi.org/10.1038/s41598-025-29028-9

Meharunnisa, M. S., Saqlain, M., Abid, M., Awais, M., & Stević, Ž. (2023). Analysis of software effort estimation by machine learning techniques. *Ingénierie des Systèmes d'Information, 28*(6), 1445-1457. https://doi.org/10.18280/isi.280602

Mustyala, S., Manchala, P., & Bisi, M. (2025). Case studies: Machine learning approaches for software development effort estimation. In T. R. Benala, S. Dehuri, R. Mall, & M. N. Favorskaya (Eds.), *Boosting software development using machine learning* (Artificial Intelligence-Enhanced Software and Systems Engineering, Vol. 7). Springer. https://doi.org/10.1007/978-3-031-88188-6_5

Priya Varshini, A. G., Anitha Kumari, K., Sneha, E., Sampathkumar, M., & Janarthanababu, D. (2024). Deep artificial neural network for software effort estimation. In E. S. Gopi & P. Maheswaran (Eds.), *Proceedings of the International Conference on Machine Learning, Deep Learning and Computational Intelligence for Wireless Communication* (Signals and Communication Technology). Springer. https://doi.org/10.1007/978-3-031-47942-7_52

Sahu, K., Alzahrani, F. A., Srivastava, R. K., & Kumar, R. (2021). Evaluating the impact of prediction techniques: Software reliability perspective. *Computers, Materials & Continua, 67*(2), 1471–1488. https://doi.org/10.32604/cmc.2021.014868

Sakhrawi, Z., Sellami, A., & Bouassida, N. (2021). Software enhancement effort prediction using machine-learning techniques: A systematic mapping study. *SN Computer Science, 2*(6), Article 472. https://doi.org/10.1007/s42979-021-00872-6

Schwalbe, K. (2015). *Information technology project management* (8th ed.). Cengage Learning.

Shobha Ajin, R., Segoni, S., & Fanti, R. (2024, October 22). Optimization of SVR and CatBoost models using metaheuristic algorithms to assess landslide susceptibility. *Scientific Reports*, 14(1):24851. doi: 10.1038/s41598-024-72663-x. PMID: 39438526; PMCID: PMC11496660.

Xue, L., Zhu, Y., Guan, T., Ren, B., Tong, D., & Wu, B. (2020, October 17). Grouting power prediction using a hybrid model based on support vector regression optimized by an improved Jaya algorithm. MDPI. https://doi.org/10.3390/app10207273

Yasmin, A., Haider, W., Daud, A., & Banjar, A. (2024). Machine learning-basedManuscript software effort estimation using development-centric features for crowdsourcing platform. *Intelligent Data Analysis, 28*(1), 183–204. https://doi.org/10.3233/IDA-237366