IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 9 February 2025, accepted 18 February 2025, date of publication 24 February 2025, date of current version 28 February 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3544625

RESEARCH ARTICLE

Using Permutation-Based Feature Importance for Improved Machine Learning Model Performance at Reduced Costs

ADAM KHAN[1], ASAD ALI[1], JAHANGIR KHAN[1], FASEE ULLAH[1,2,3], AND MUHAMMAD FAHEEM[4,5], (Member, IEEE)

[1]Department of Computer Science and IT, Sarhad University of Science & Information Technology, Peshawar 25000, Pakistan
[2]Institute of Emerging Digital Technologies (EDiT), Universiti Teknologi PETRONAS, Seri Iskandar 32160, Malaysia
[3]Center for Cyber Physical Systems (C2PS), Department of Computer and Information Sciences, Universiti Teknologi PETRONAS, Seri Iskandar 32160, Malaysia
[4]School of Computing Technology and Innovations, University of Vaasa, 65200 Vaasa, Finland
[5]VTT Technical Research Center of Finland Ltd., 02150 Espoo, Finland

Corresponding author: Muhammad Faheem (muhammad.faheem@uwasa.fi)

The authors gratefully acknowledge the support provided by their affiliated Universities and Research Institutes to complete to this study.

ABSTRACT In Software Quality Assurance (SQA), predicting defect-prone software modules is essential for ensuring software reliability and consistency. This task is commonly achieved through Machine Learning (ML) techniques, but improving model performance typically incurs significant computational costs. These high computational costs and uncertain payoffs make most Software engineering researchers reluctant to optimize ML models. This creates a need for novel techniques that can achieve near-optimal performance of hyperparameter settings while maintaining the computational efficiency of default settings. To address this, we employed five ML models, Decision Tree, Ranger, Random Forest, Support Vector Machine, and k-nearest Neighbors, and optimized their parameters using the random search technique. Our experiments covered six diverse Software Fault Prediction (SFP) datasets, encompassing various software features, application domains, and defect patterns, to evaluate the approach's generalizability and effectiveness. Moreover, the Permutation Feature Importance (PFI)-based model-agnostic method was employed to identify the top ten features most critical for model accuracy and efficiency. These selected features were used to retrain the ML models without hyperparameters (default settings) to determine whether similar performance could be achieved at low computational cost. The results show an average accuracy improvement of 77.39% and a 92.02% reduction in computational cost. The most important case attained a 99.25% accuracy improvement and a 96.77% cost reduction. Such results clearly show that PFI-based feature selection is capable of high performance at a fraction of computational cost, offering an efficient solution for software engineers to optimize ML models.

INDEX TERMS Model-agnostic techniques, permutation feature importance (PFI), software fault prediction (SFP), predictive accuracy, machine learning (ML), computational cost, default settings, hyperparameter.

I. INTRODUCTION

The Software Quality Assurance (SQA) team is critical in delivering defect-free software systems. One of their key techniques is to predict which software system modules (e.g., classes, methods, etc.) are likely defects-prone; for that, they need highly accurate Machine Learning (ML) models. To achieve precise predictions of faults or bugs, prior research reported the need to optimize the parameters (hyperparameter) of the ML models [1], [2], [3], [4]. While some studies suggest that hyperparameter ML techniques might not consistently outperform default settings [5], [6], While some

The associate editor coordinating the review of this manuscript and approving it for publication was Joao Bernardo Ferreira Sequeiros.

studies suggest that hyperparameter ML techniques might not consistently outperform default settings [2], [7], [8], [9]. in the majority of cases, Software Faults Prediction (SFP) studies report that parameter tuning significantly improves the predictive performance [2], [10], [11], [12]. However, hyperparameter optimization involves high computational overheads; even when optimized settings outperform the default ones, the additional time and resources consumed in computation become a challenge. Therefore, researchers in the SFP domain avoid hyperparameter tuning mainly because of its time-consuming nature and uncertain payoffs [2], [10], [11], [12]. For instance, Chakkrit et al. [2] found that 80% of the 50 most cited SFP studies prefer default settings, underscoring the preference for simplicity and computational efficiency. This emphasizes the need for substitute approaches that keep predictive accuracy while reducing computational costs. The existing literature explains the decisions of complex ML models using intrinsic interpretability and post-hoc explanations. Intrinsic interpretability focuses on models that are inherently easy to understand [13]. In contrast, post-hoc explanations simplify the black-box models, such as deep neural networks, without showing their internal working [14], utilizing techniques like feature attribution, text explanations, and visualizations [15].

Post-hoc explanations can be divided into model-agnostic and model-specific methods. Model-agnostic methods, such as Local Interpretable Model-Agnostic Explanations (LIME) [16] and SHAP (Shapley Additive Explanations) [17] are widely used in Explainable Artificial Intelligence (XAI). These approaches provide interpretability across diverse models. For instance, SHAP leverages Shapley Values, a game-theoretic approach, to fairly distribute contributions among players in a cooperative game. In the context of feature selection, the model output is the ''game,'' and the features act as ''players'' contributing to that output.

In contrast, model-specific explanations are tailored to particular model designs. For instance, the feature importance(FI) function in tree-based models [13] provides explanations specific to decision trees. However, such methods lack the general applicability of model-agnostic approaches, making them unsuitable for diverse modeling scenarios. To overcome this limitation, model-agnostic approaches for FI, such as Permutation Feature Importance (PFI), have been developed to assess global feature relevance across different models. PFI was introduced by Breiman [18] initially designed for tree-based models, but has since been extended to broader applications [19], [20], [21] demonstrating its adaptability and effectiveness. Despite certain limitations, such as sensitivity to data shuffling [22], PFI remains a robust and computationally efficient baseline for feature selection. Its utility is notable in SFP, where it provides a reliable method for identifying critical features while maintaining model-agnostic applicability [23].

Given the computational efficiency and robustness of PFI, this study highlights whether feature selection using PFI can reduce the challenges posed by hyperparameter optimization in SFP. We aim to assess whether default settings of hyperparameters, when coupled with the most important features extracted using PFI, can reach comparable accuracy as fully optimized models (hyperparameters) while significantly reducing computation costs. Though PFI offers valuable insights into feature selection, its applicability in real-world SE domains is mainly unexplored. This study bridges this gap by using PFI to extract the most critical features from hyperparameter settings and applying them to default configurations. Therefore, our approach is a balanced trade-off between predictive accuracy and computational efficiency, highlighting the scalability and resource challenges of SFP research. By adopting this methodology, we aim to contribute to the advancement of XAI in SE applications. Our proposed approach is resource-efficient, scalable, and in line with industry requirements, offering a practical solution that aligns theoretical advancements with real-world needs. Through this study, we hope to make state-of-the-art XAI techniques impactful for researchers and practitioners alike.

A. RESEARCH QUESTION (RQ)

Based on the challenges discussed in the last section, this research study aims to evaluate the impact of parameter optimization in the SFP domain, considering the significance of the XAI technique in interpreting model behavior. Prior research underscores the importance of identifying the most effective optimization techniques based on their performance across diverse datasets. To assess whether hyperparameter optimization positively affects prediction accuracy and computational costs, we have formulated the following RQ:

RQ1: Can default settings achieve performance levels comparable to hyperparameter settings while significantly reducing computational costs?

Motivation for RQ 1: Most defect prediction models perform poorly with default parameter settings. Hence, hyperparameter optimization becomes essential for improving prediction accuracy. Nonetheless, most researchers avoid this process in industries because of the computational burden involved in optimization. This RQ will establish the feasibility of having default settings perform equally and hyperparameter settings using important features identified through model-agnostic techniques. With this approach, it will minimize the computational cost of models while maintaining their accuracy.

This study aims to provide researchers with a proper understanding of the valuable aspects and techniques involved in feature selection and resource-efficient optimization of SFP, paving the way for a more immediate and fast comprehension of this domain's critical inner components.

In this context, the key contributions of the study are listed below:

- To the best of our knowledge, this is the first study to explore the applicability of a model-agnostic PFI method and identify critical features whose selection improves the efficiency and effectiveness of the models.

- The study evaluates the performance of five ML models, namely Random Forest, K-Nearest Neighbors, Decision Tree, Support Vector Machine, and Ranger, on six SFP datasets, thus giving a more comprehensive comparison of improvements in the computational cost and accuracy.
- The study proposes an approach that utilizes PFI-selected features to enhance default model configurations and offers a cost-effective alternative to hyper-parameter tuning. This contribution provides SE researchers and practitioners a practical strategy to obtain near-optimal performance with the benefit of lowering computational costs.
- This study applies PFI to enhance model performance in SFP and contributes to the increasing Explainable AI research in SE, discussing performance consistency and resource constraints.

The rest of the paper is structured as follows: In Section II, we provide background and a review of related work. Section III is related to the study design; Section IV presents the results. Section V presents Discussions, Section VI addresses potential threats to the study, and the paper concludes with Section VII, summarizing conclusions and Section VIII outlining directions for future work.

II. BACKGROUND AND RELATED WORK

ML techniques have been widely applied to improve the SFP data, i.e., Tantithamthavorn et al. [2] demonstrated that automated parameter optimization across 18 open-source projects could enhance the area under the curve (AUC) by up to 40%. They also highlighted the importance of defining parameter search spaces, especially for sensitive classification techniques like decision trees, to avoid overfitting risks. Kang et al. [24] proposed parameter optimization for just-in-time software defect prediction using Harmony Search to improve random forest performance while addressing class imbalance issues.

In addition, other studies highlighted the significance of parameter optimization on defect prediction models; for instance, Koru et al. [25] and Mende et al. [26] highlight that parameter selection significantly impacts model performance. Mende and Koschke [27] and Dyana et al. [28] both utilize default settings using software defect predictions utilizing Random Forest (RF). Jiang et al. [29] and Tosun et al. [11] similarly emphasize that the default parameter settings of research toolkits (e.g., R [30], WEKA [31], MATLAB [32] are suboptimal.

Moreover, research in SFP often relies on default settings provided by widely used toolkits such as R and Weka. For instance, Mende et al. [33] utilized an R package's default number of decision trees to train a Random Forest classifier, highlighting a tendency to adopt preset configurations. Despite these findings, previous studies rely on default settings, as underscored by Fu et al. [34] reported that 80% of the 50 highly-cited defect prediction studies used default parameters for time efficiency and simplicity. Beyond the hyperparameter, feature-important methods have become important in ML applications for interpreting model predictions. Notably, identifying key features can improve model interpretability in complex datasets, such as genetic data. The study provides an overview of five ML techniques, including Random Forest and Support Vector Machines, concerning metrics designed to uncover which features contribute most to accurate prediction [35].

These studies highlight the broad preference for default parameters in SFP studies and also motivate researchers into whether default settings can be at least as hyperparameter settings when enhanced by key features. Many studies indicate that hyperparameters are computationally expensive, especially for complex models and large datasets. Hyperparameter often requires advanced techniques like parallel processing or model simplification [1], [2], [36]. For instance, [37] analyzed the trade-offs between computational cost and hyperparameter accuracy, and they found that resource-efficient algorithms could obtain near-optimal results with fewer demands on computational resources. The authors do similar work [38] with the help of a bag-of-words model and efficient feature extraction techniques for saliency-based defect detection to demonstrate the potential of enhancing the accuracy of predictions while maintaining reduced computational costs.

Therefore, the computational cost of hyperparameter settings, encompassing time, memory, and processing power, is a crucial concern [39]. For instance, the authors in the study [2] found that optimization added less than 30 minutes for 75% of datasets in 12 out of 26 classification techniques. While some methods like C5.0 required minimal extra time, others like AdaBoost needed over 3 hours for larger datasets. This study concluded that optimization benefits outweigh the costs, especially with affordable cloud computing solutions.

Hence, the literature demonstrates that although performance accuracy can be considerably improved through hyperparameters, this improvement comes at a high computational cost. Thus, novel techniques are needed that achieve optimum performance benefits in terms of parameter utilization with the same computational efficiency as default settings.

III. STUDY DESIGN

A. DATASETS

This study used six datasets (regression-based) from the SFP domain [27], i.e., Ant 1.7, Xalan 2.6, Xerces 1.3, Lucene 2.0, Synapse 1.1, and Velocity 1.6. These datasets, which are publicly available and widely used in empirical SE studies, focus on bug detection as the target variable. These datasets were selected for their diversity, encompassing both large (e.g., xalan is a large dataset with 885 observations) and small/medium datasets (e.g., Lucene is a small dataset with 195 observations), ensuring robust and generalizable conclusions. Table 1 illustrates the characteristics of these

TABLE 1. Dataset information.

Datasets	Number of Features	Number of Observations	Widely Used Features
Ant	16	745	WMC (Weighted Methods per Class), CBO (Coupling Between Objects), LOC (Lines of Code), CAM (Cohesion Among Methods of Class), DIT (Depth of Inheritance Tree), NOC (Number of Children)
Xalan	21	885	
Xerces	21	588	
Lucene	22	195	
Synapse	22	222	
Velocity	22	229	

datasets, including the number of features, observations, and commonly used features [40].

B. HYPERPARAMETER OPTIMIZATION TECHNIQUE

Among numerous automated optimization techniques [41], [42], Random search is selected due to its scalability and efficiency.

1) RANDOM SEARCH

Random search [12] is an effective method for optimization tasks, including hyperparameter tuning in ML models and configuration settings in software systems. It is more straightforward yet effective than other optimization techniques, delivering competitive results requiring less computational effort. Its capability to explore the hyperparameter space without relying on local gradients enables it to be highly suitable for identifying a diverse set of configurations that may lead to better performance [12], [43], [44].

This technique evaluates a subset of hyperparameter settings that are randomly generated. This approach allows Random search to broaden the parameter space's exploration, often yielding more effective models within the same computational budget. This broader capability makes it comparable to advanced optimization techniques, such as Genetic algorithms, etc. [2]. The process of random search includes:

- Generating hyperparameter settings: randomly create parameter combinations for a specified number of iterations (e.g., five iterations produce five combinations).
- Assessing the performance: evaluating the performance of each randomly generated parameter setting using a defined metric.
- Identifying the best configurations: identify the parameter setting that performs best.

Therefore, a random search using the default search space was performed for 30 iterations. For optimizing parameters, we utilized random search with the default search space provided by the *mlr3* library. The default search space contains all the relevant hyperparameters for the learner (model) being optimized and their respective ranges predefined by the library. Since our aim was not to inspect single search spaces or determine the effect of a particular range of hyperparameters on performance (as done in previous study [1]), we took advantage of this standardized framework to promote reproducibility and align with widely accepted standards. A random search was performed for 30 iterations to guarantee diversity in exploring the hyperparameter space.

Random Search balances efficiency and accuracy by avoiding exhaustive evaluation and reducing computational effort. It is an excellent choice for hyperparameter tuning in this study. The working of Random search is represented by using Algorithm 1.

Algorithm 1 Random Search Algorithm

Input: Objective function $f(x)$, search space X, maximum iterations N
Output: Best solution x_{best} and its corresponding value $f(x_{\text{best}})$
Initialize x_{best} randomly from the search space X
$f_{\text{best}} \leftarrow f(x_{\text{best}})$
for $i = 1$ to N **do**
    Generate a random solution x_{new} from X
    Evaluate $f(x_{\text{new}})$
    if $f(x_{\text{new}})$ is better than f_{best} **then**
        $x_{\text{best}} \leftarrow x_{\text{new}}$
        $f_{\text{best}} \leftarrow f(x_{\text{new}})$
    end if
end for
return $x_{\text{best}}, f_{\text{best}}$

C. ML TECHNIQUES

Motivation: The study evaluated different ML models that differ in complexity, interpretability, and algorithmic principles. This selection aims to test the generalizability of our approach toward other types of models, ensuring that our findings are applicable across a broader set of algorithms employed in SFP. Therefore, the selection contains simpler, more interpretable models, such as Decision Tree, k-Nearest Neighbors, and ensemble-based models, such as Random Forest and Ranger, balancing traditional and moderately complex algorithms [45]. The employed models used in this study are computationally efficient and scalable, making them suitable for SFP tasks that often involve datasets with different sizes and complexities. Advanced models such as Ranger also serve as benchmarks to evaluate the robustness of our proposed approach.

The application of these ML models in the SFP domain has been established in prior studies, for instance:

- Study [46] incorporated Random Forest, K-nearest Neighbor, and Decision Tree
- Study [47] used K-Nearest Neighbors, Logistic Regression, Support Vector Machine, and Decision Tree

- Study [9] employed Decision Tree, Random Forest, k-nearest neighbours, and Support Vector Machine,
- Study [48] utilized a Support Vector Machine and a Random Forest in the same context.

The study employs a combination of models to achieve simplicity, interpretability, and performance. This study aligns with the existing literature and ensures the findings are practical and accessible for SQA practitioners who might not have the resources to access advanced models.

Therefore We have utilized Decision Tree, Ranger, Random Forest, Support Vector Machine, and k-nearest Neighbors.

1) SUPPORT VECTOR MACHINE (SVM)

SVM is a powerful tool for handling high-dimensional data and complex relationships, widely used in text classification and image recognition due to its effectiveness and simplicity [49].

2) DECISION TREES (DT)

DT, renowned for its simplicity, interpretability, and ability to handle numerical and categorical data, are widely used in healthcare, finance, and marketing to provide valuable insights into decision-making processes [50].

3) K-NEAREST NEIGHBORS (KNN)

KNN is a robust ML algorithm for classification and regression tasks, classifying data points according to the majority class of its "k" nearest neighbors. It assumes similar instances exist nearby and is non-parametric, making it easy to understand and implement [51].

4) RANDOM FOREST (RF)

RF is Known for its ability to handle high-dimensional data with complex relationships; RF captures nonlinearity and interactions, providing estimates of feature importance. Its versatility and effectiveness make it widely applied in domains like bio-informatics, finance, and image recognition [52].

5) RANGER

Ranger is a high-performance RF implementation in ML known for its efficiency and scalability. It builds on the principles of ensemble learning by constructing multiple DTs and aggregating their predictions during training. Ranger is highly valued for its robustness, ability to handle high-dimensional data, and ability to capture complex relationships. It has applications in bio-informatics, finance, and image recognition [53].

D. EVALUATION METRIC

The MAE is a widely used method for comparing the prediction accuracy of various ML models across multiple research studies [54], [55]. It was utilized to compare the prediction accuracy of models. It measures the average magnitude of the errors between predicted and actual values; it is a computationally efficient and interpretable metric. The formula is given below:

$$\text{MAE} = \frac{\sum_{i=0}^{n} |p_i - a_i|}{n} \tag{1}$$

where p_i is the predicted value, a_i is the actual value, and n is the number of data points.

where:

- p_i is the predicted value,
- a_i is the actual value, and
- n is the number of observations.

It is an excellent metric for measuring how a model performs when that model is used to minimize prediction errors. It can be explained in the following example. Suppose we have five software modules with actual defect counts: [5, 7, 10, 4, 8], and a machine learning model predicts: [6, 5, 9, 3, 7]. The MAE is calculated as follows:

$$\begin{aligned}\text{MAE} &= \frac{|6-5| + |5-7| + |9-10| + |3-4| + |7-8|}{5} \\ &= \frac{1+2+1+1+1}{5} = 1.2\end{aligned}$$

This means, on average, the model's predictions differ from actual values by 1.2 defect counts.

E. PERCENTAGE DIFFERENCE FORMULA

The percentage difference formula is employed to quantify accuracy and improvements in computational efficiency. It provides a relative measure of change between two values, making it widely used in comparative studies focusing on optimization techniques and resource utilization [1]. In addition, the formula is also used for improved accuracy when the difference in the accuracy after applying PFI becomes less (better) than before applying the PFI, indicating an improvement. This formula is used for computational efficiency to evaluate the relative reduction in computationally costly computations between the optimized and the default configurations. The formula is as below:

$$\frac{|V_1 - V_2|}{\left(\frac{V_1+V_2}{2}\right)} \times 100 \tag{2}$$

where:

- V_1 is the value Before PFI: (e.g., Accuracy before applying PFI or computational time with optimized settings)
- V_2 is the value After PFI: (e.g., Accuracy after applying the PFI (feature selection) or computational time with default settings)

Based on the discussions above, this study incorporates the percentage difference formula to quantify the improvement in accuracy and reduction in computational cost, aligning the studies [1], [2] with well-established methods for optimally resource-efficient performance.

F. TOOLS AND IMPLEMENTATION

This study utilized R, a widely recognized platform from data science and statistical computing, renowned for its extensive package ecosystem and versatility in analytical workflows [56]. The *mlr3* library was explicitly adopted in this study to streamline and implement ML capabilities.

G. MODEL-AGNOSTIC FEATURE SELECTION TECHNIQUES

Various model-agnostic methods, such as SHAP [16], [57], LIME [15] etc, have been used for feature selection. This study employed PFI [58], a widely recognized technique (model-agnostic), to evaluate feature consistency across different SFP datasets.

1) PFI

PFI [58] is a model-agnostic technique that measures the decrease in a model's performance when the value of a single feature is randomly shuffled. By observing the change in prediction accuracy after shuffling, PFI quantifies the importance of individual features. The process of PFI has been explained as follows:

1) Baseline Accuracy calculation: The baseline accuracy p is established for a pre-trained model with n features and a test set.
2) Feature shuffling: The values of each feature are iteratively shuffled, and the prediction scores of the model are recalculated on the modified datasets.
3) Score reduction: The importance of each n feature is determined based on the decrease in prediction accuracy compared to p.

PFI quantifies the contribution of various characteristics to the model's prediction performance through the extent to which their shuffling affects the accuracy at which the model performs. This process breaks the relationship between the feature and the target, indicating the feature's importance based on the resulting drop in the model's performance. It does not require feature normalization, can be generalized to nearly any machine learning model with handcrafted features, and is robust and efficient in implementation. PFI applies to non-linear or opaque models and can be repeated multiple times with different permutations for robust results [18], [59], [60]; therefore, previous studies have effectively utilized PFI [59], [61].

Algorithm 2, illustrates the procedural steps of PFI, where the impact of each feature is measured by comparing the change in model error when its values are permuted, thus quantifying its contribution to the predictive performance.

The PFI method has been applied to this study due to its simplicity, interpretability, and ability to generate actionable insights. PFI is straightforward to implement and interpret by permuting the feature values and measuring resulting changes in model performance, providing transparent and interpretable results. Moreover, the method PFI outperforms others in dealing with correlated features and hence tends to minimize the bias associated with importance scores. Given its widespread usage in SE literature [59], [62], [63], this study marks a baseline effort to refine and adapt the PFI method further. Future researchers in the domain can use the findings of this study to explore alternative techniques, validate results, and expand upon the presented insights.

Algorithm 2 Permutation Feature Importance

Input: Trained model f, dataset $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, number of features m, number of permutations P
Output: Importance scores for each feature
Train the model f on dataset D
Compute baseline accuracy acc_{baseline} on D
for each feature j from 1 to m **do**
  Set $score_j = 0$
  for each permutation p from 1 to P **do**
    Permute feature j in a copy of D to get D'
    Compute accuracy acc_{perm} on D'
    Update $score_j = score_j + (acc_{\text{baseline}} - acc_{\text{perm}})$
  end for
  Average score: $score_j = \frac{score_j}{P}$
end for
return Feature importance scores $\{score_1, \ldots, score_m\}$

H. EXPERIMENTAL SETUP

The study used six regression-based SFP datasets and a random search optimization technique to evaluate the accuracy of optimized ML models in optimized settings [12]. Performance accuracy was measured through MAE values, and the computational time required for both the optimized and default settings was recorded. To identify important features, the PFI model-agnostic technique was applied to extract the top ten features from the optimized settings. In addition, all the experiments ran on a system with specifications of an Intel Core i3-5010U CPU, 4 GB of RAM, and a 64-bit operating system running Windows 10 Professional.

I. EXPERIMENTAL SETTINGS

Sixty experiments were performed, 30 using default parameter settings and 30 using optimized parameters, across 5 ML models and 6 SFP datasets.

The following Metrics have been evaluated:

1) Predictive Accuracy: Measured using MAE and percentage difference before and after applying PFI-selected features. Cases of accuracy improvement and degradation were noted.
2) Computational cost: Processing times were observed for each experiment under the default and hyperparameter settings and percentage cost efficiency to quantify resource savings.

IV. RESULTS AND DISCUSSION

A. RQ: CAN DEFAULT SETTINGS ACHIEVE PERFORMANCE LEVELS COMPARABLE TO HYPERPARAMETER SETTINGS WHILE SIGNIFICANTLY REDUCING COMPUTATIONAL COSTS??

The study aims to evaluate how the PFI method can improve the performance of default settings and reduce computational costs by utilizing top (10) features derived from hyperparameter settings. To illustrate the results, we provided only the experiments for the RF/Ant, DT/Ant, Ranger/Ant, SVM/Ant, Ranger/Lucene, and Ranger/Xalan in the main text (see Table 2). In contrast, detailed results for all the datasets are provided in Table 4, Appendix A. To thoroughly analyse the effect of our approach, we perform an in-depth comparison of several aspects. This includes comparing performance differences between hyperparameter and default settings, the effectiveness of PFI-selected features, and the computational cost associated with both configurations. Such comparisons help us identify the trade-offs involved in hyperparameter and feature selection applied to SFP.

1) PERFORMANCE COMPARISON: HYPERPARAMETER VS. DEFAULT SETTINGS

The hyperparameter settings are better in 23 out of 30 (76.6%) experiments, while the default settings performed better only in 4 out of 30 experiments (13.3%). Consequently, no difference in prediction accuracy was observed in 3 out of 30 experiments (10%), as shown in Fig. 1. These findings aim to determine whether the performance of the hyperparameter settings with the default settings could significantly save computational cost, which is critical in real-world scenarios where computational resources are limited. This approach could enable software industries to utilize existing infrastructure efficiently.

2) FEATURE RANKING: HYPERPARAMETER VS. DEFAULT SETTINGS

In the majority of the experiments, hyperparameter settings performed better than the default settings (see Fig. 1), and that is because both the default and hyperparameter settings produce different subsets of features (for example, the Fig. 2 & Fig. 3 illustrates the experiments of KNN/Synapse).

To illustrate these results further, Fig.2 and Fig.3 showed the feature ranking for the KNN/synapse experiments under default settings (Fig.2) and hyperparameter settings (Fig. 3). In the default settings (Fig. 2), for instance, *ce* has the highest importance, and then *cbo, dit* etc. Consequently, in Fig.3, feature *ce* has the highest importance, followed by *moa, mfa*, etc. This shows that *ce* is the most critical feature in both settings while other features' order (ranking) shifts significantly. For example, *dit* is ranked 3rd in the default setting while it drops in importance to 9th rank under the optimized settings.

This observation raises a critical question: What happens if we apply the important features applied with the optimized settings (using the PFI technique) to the default settings? We aim to explore i.e., (1) does it improves the performance of the default settings to a certain level or not and (2) how much computational cost it saves compared to the time required for the optimized settings.

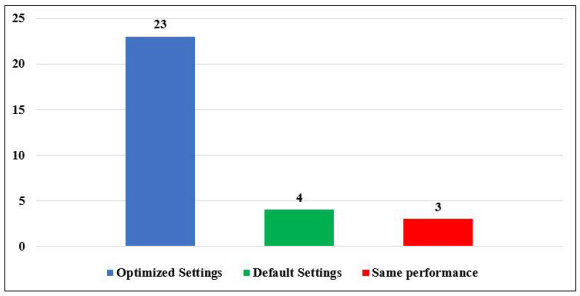

FIGURE 1. Performance comparison of the hyperparameter and default settings.

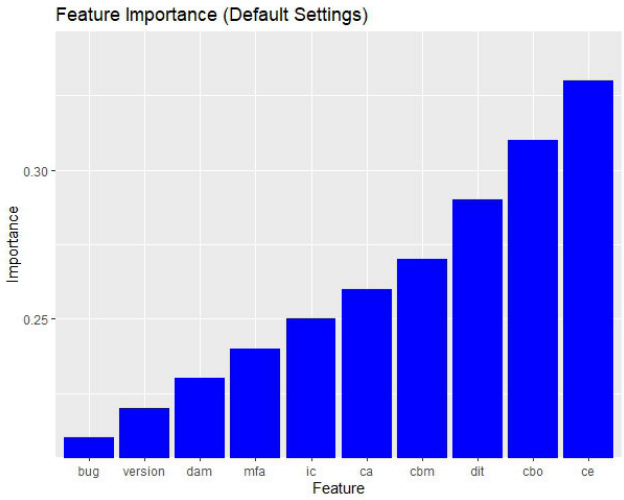

FIGURE 2. KNN/Synapse via default settings feature ranking.

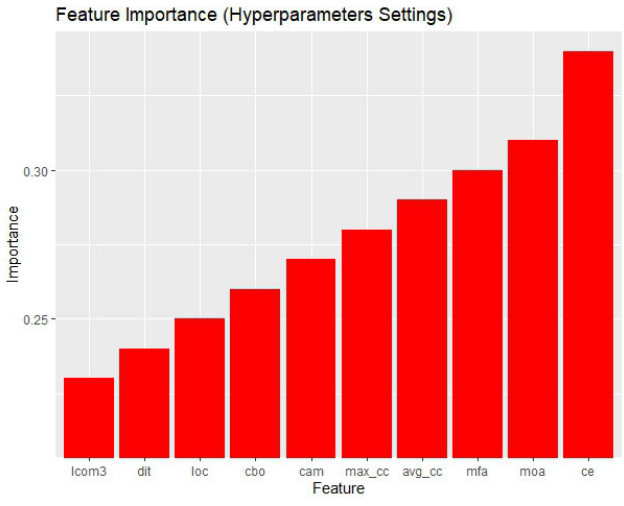

FIGURE 3. KNN/Synapse via hyperparameter settings feature ranking.

TABLE 2. Performance (Improved) and computational cost analysis of ML models on SFP Datasets with hyperparameter and default settings.

Experimental Settings	Tuned MAE (Hyperparameter)	Untuned MAE (Default)	% Diff. Before PFI	% Diff. After PFI	Absolute Changes in MAE	Relative Improvement in Accuracy (%)	% Diff. (Time Efficiency)
RF/Ant	0.489	0.512	4.440	0.230	+4.210	94.820	89.660%
DT/Ant	0.331	0.499	40.480	0.331	+40.180	99.250	93.750%
Ranger/Ant	0.501	0.511	1.980	0.400	+1.580	79.760	90.910%
SVM/Ant	0.531	0.581	8.990	2.230	+6.760	75.150	90.910%
Ranger/Lucene	2.221	2.901	26.550	2.220	+24.330	91.640	92.310%
Ranger/Xalan	0.410	0.419	2.170	0.240	+1.930	88.780	90.910%

3) PERFORMANCE IMPROVEMENT THROUGH PFI

Table 2 highlights the enhanced performance of default settings after incorporating features identified under hyperparameter settings to facilitate comparative analysis. The table highlights three primary metrics: *(Note: these are experiments where the hyperparameter settings performed better than the default settings, as shown in Fig. 1)*

1) *%Difference Before PFI (Hyperparameter vs Default):* This column represents, the initial accuracy gap between optimized and default settings, measured using MAE. For instance, the gap in the RF/Ant experiment is 4.440%.
2) *% Difference After PFI (Key Features Applied):* This column represents the percentage difference in MAE after applying PFI-selected features to default settings. For instance, the RF/Ant experiment improved to 0.230 from the original 4.440.
3) *Absolute changes in MAE:* This column represents the absolute difference between the MAE of the hyperparameter and the default model, highlighting the actual difference in accuracy. For instance, in the case of DT/Ant, the absolute change in MAE was +40.180, which shows that after applying key features, the default MAE was reduced from 0.499 to 0.331.
4) *Relative improvement in accuracy:* This column represents an improvement in model performance after applying PFI compared to before PFI. For instance, DT/Ant, the accuracy difference decreased from 40.480% (before PFI) to 0.300 % (after PFI), showing a 99.250% relative improvement.
5) *% Difference increase (Time Efficiency with Features):* The percentage reduction in computational cost between the hyperparameter and default settings. For instance, RF/Ant experiments achieved an 89.660% time efficiency increase.

These findings suggest that PFI-based feature selection improves the performance of models on multiple datasets. In most cases, applying the PFI-selected features to default settings resulted in accuracy levels comparable to or even better than hyperparameter settings, making it a worthwhile alternative to full hyperparameter tuning. In addition, the Absolute Changes in the MAE column indicate the practical improvement in the model's performance after optimization and feature selection. For instance, the DT/Ant model produced the most significant absolute reduction in MAE (40.180), which implies a dramatic improvement, whereas Ranger/Xalan has the smallest one (1.930). Interestingly, both Ranger/Lucene and Ranger/Xalan showed almost the same percentage (relevant improvements) of 91.640% vs. 88.780%, but absolute changes in MAE were drastically different: 24.330 vs. 1.930, respectively. The same was the case with SVM/Ant and RF/Ant, where both showed significant absolute reductions in MAE: 6.760 and 4.210, respectively, indicating the effectiveness of the optimization process. These absolute changes complement the percentage differences with a clearer view of the actual impact on model performance. Overall, the results present consistent improvements across all models in varying degrees due to data and algorithm-dependent factors. The graphical representation in Fig. 4 depicts the percentage differences in performance before and after applying PFI across a few selected experiments detailed in Table 2.

4) ACCURACY FINDINGS

Table 2 is used to summarize such results, and a few more selected examples are presented below (for detailed results, see Table 4 in appendix A):

- SVM/Lucence: In this experiment, the accuracy difference reduced from 27.790% to 1.500%, reflecting a remarkable 94.500% improvement.
- RF/Synapse: in this case, the accuracy difference decreased from 4.405% to 0.230%, resulting in a 94.900% improvement.
- DT/Velocity: Accuracy difference fell from 8.530% to 0.190%, marking an exceptional 97.70% improvement.

PFI improved accuracy in almost all cases, but there were a few cases where accuracy slightly degraded by some margin; this shows the need to evaluate feature selection techniques in given contexts, as shown in Table 3.

For instance

- SVM/Synapse: The accuracy difference increased from 0.196% to 2.035% after applying PFI.
- RF/Xerces: This experiment increased the accuracy difference from 0.622% to 6.913%.
- RF/Xalan: Accuracy difference increased from 0.521% to 5.216%.

In addition, the MAE column in the table exhibits a degradation trend after applying the importance of the feature.

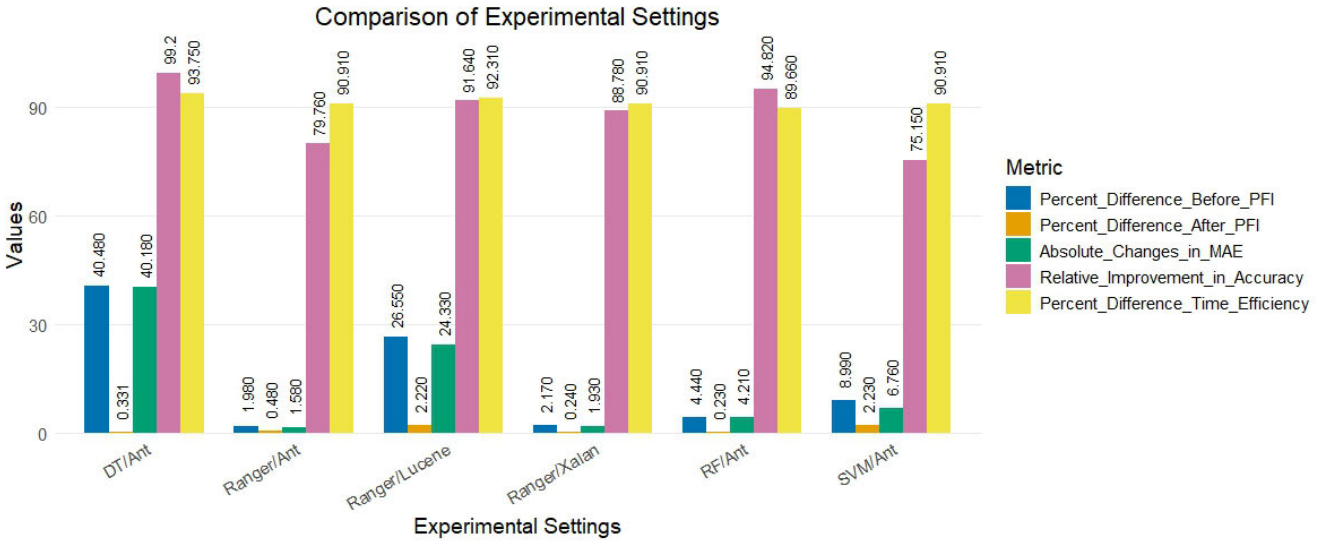

FIGURE 4. Percentage differences in performance before and after applying PFI across selected experimental settings.

TABLE 3. Performance and computational cost analysis of ML models on various datasets with hyperparameter and default settings (where accuracy degraded).

Experimental Settings	Tuned MAE (Hyperparameter)	Untuned MAE (Default)	% Diff. in Accuracy (Before PFI)	MAE After PFI Features	% Diff. in Accuracy (After PFI)	Absolute Changes in MAE
RF/xalan	7.849	7.890	0.521	7.450	5.216	-4.952
DT/xalan	0.335	0.335	0.111	0.334	0.332	-0.222
RF/Xerces	7.849	7.898	0.622	8.411	6.913	-6.290
DT/Xerces	0.327	0.338	3.367	0.315	3.461	0.094
SVM/Xerces	1.881	1.904	1.215	1.915	1.791	-0.576
RF/Lucene	1.170	1.180	0.851	1.160	0.858	-0.007
SVM/Synapse	2.035	2.039	0.196	1.994	2.035	-1.839

For instance, the RF/xalan experiment model resulted in a considerable hike in MAE (−4.952), which signifies a drop in performance. Similarly, in the case of the RF/Xerces model, degradation has been observed in MAE (−6.290). Among the smaller-sized models, degradation is still quite noteworthy in some cases, like that of DT/xalan (−0.222) and SVM/Xerces (−0.576), indicating the negative effect of feature selection on such models' performance. Even this is not very small, like that of RF/Lucene (−0.007), showing a deterioration in accuracy. These declining accuracy indicate that PFI might have effects that depend on dataset features, feature interaction, or specific model dependency. Another reason for degradation could be that the features are ranked based on particular contributions, which does not consider inter-feature dependency, thus leading to possibly suboptimal feature selection in different datasets. In contrast, despite these exceptions, the overall trend shows that PFI-selected features can efficiently transform default model settings into highly viable configurations with minimal computational overhead. Table 4 of Appendix A contains a detailed breakdown of all experiments.

5) COMPUTATIONAL COST RESULTS

The findings show significant reductions in computational costs when utilizing PFI-selected features, for instance:

- SVM/Lucene: In this case, the processing time was reduced from 46 seconds to 17 seconds, illustrating a 92.060% reduction.
- DT/Velocity: In this experiment, processing time was reduced from 45 seconds to 16 seconds, achieving a remarkable reduction of 95.080%.
- kKNN/Xalan: The computational cost dropped by 96.770%.
- RF/Ant: Dropped computational cost by 89.660%.
- Ranger/Ant and SVM/Ant: Both achieved a 90.910% reduction in computational cost.

Fig. 5 shows the computational cost efficiency across various models and datasets.

6) VARIATIONS IN EFFICIENCY

For instance, in the RF/Xerces experiment, the computational cost narrowed down from 39 seconds to 15 seconds, representing an 88.890% decrease. While PFI generally enhances

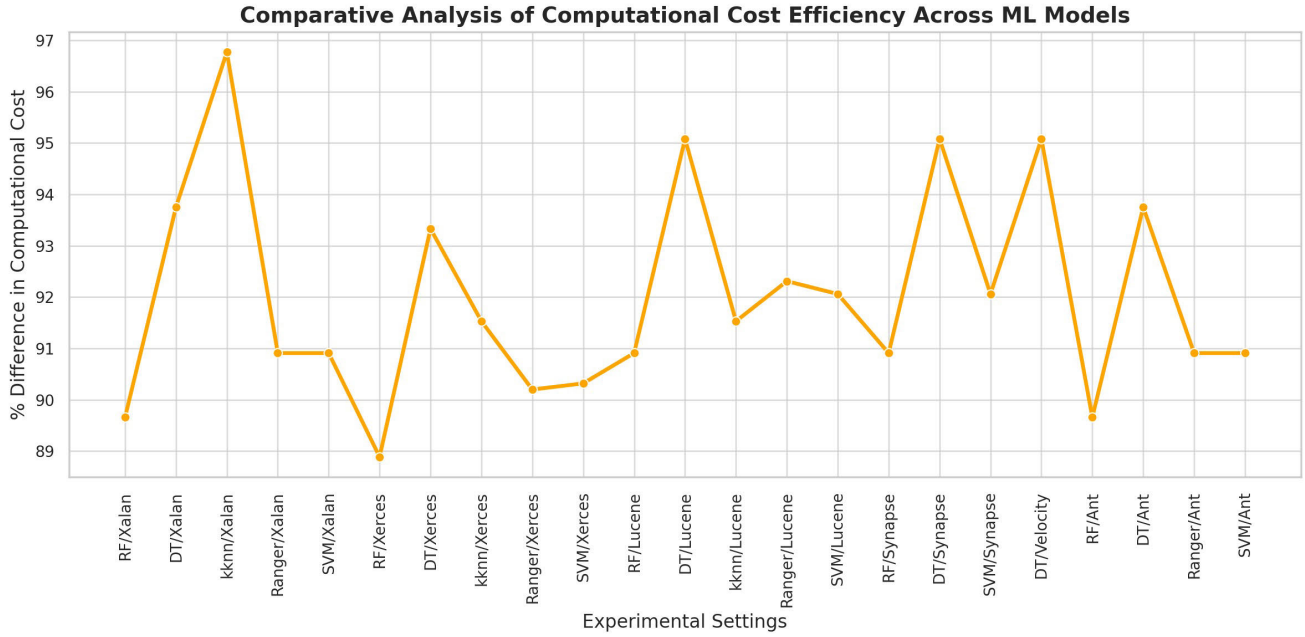

FIGURE 5. Comparative analysis of computation cost efficiency across ML models for different datasets.

computational efficiency, it could vary according to dataset and model configuration. By utilizing PFI-selected features, this study demonstrates that the default settings can yield their optimum without overheads, representing a promising alternative for SFP applications. In addition, Fig. 5 depicts the computational cost difference between default settings with PFI-selected features and hyperparameter settings across different experimental setups (for all experiments). The X-axis shows the combination of ML models and datasets, e.g., RF/Xalan, RF/Xerces, etc, illustrating the comparison across different experimental settings (for details, see Table 4 in Appendix A). while the Y-axis displays the percentage difference in computational cost, highlighting the increase and decrease in efficiency for optimized settings compared to default settings. Each point in the figure corresponds to a specific experimental setting, with its position indicating the percentage difference in computational cost for that particular model/dataset combination. The trend lines connecting these points show significant differences across the experiments, showcasing how the computational cost differences vary. For instance, kKNN/Xalan show relatively higher percentage differences, underscoring computational efficiency variability depending on the chosen model and dataset.

V. DISCUSSION

In this study, we observed that in 76.6% of the cases, the hyperparameter settings outperformed the default settings. In contrast, in 13.3% of the experiments, the default settings delivered better performance than the optimized configurations. In 10% of the experiments, no difference in prediction accuracy was detected (for instance, RF/Xerces and RF/Xalan showed marginal differences). Prior studies [2], [64] also suggest that hyperparameter settings perform better than the default settings but with high computational costs. However, unlike traditional studies, this research addresses the problem of computational load by integrating the PFI technique to balance performance and cost. This study employed the PFI (an XAI technique) to address this challenge and extract the most important features in the hyperparameter settings. These features were re-applied to models under default settings, eliminating the need for further parameter tuning. The results demonstrate that this approach achieves almost the same performance levels comparable to hyperparameters while significantly reducing computational cost. For instance, DT/Ant showed accuracy improvements of over 99.250% (see Fig. 6), making them highly efficient in resource utilization. In addition, SVM/Lucene showed a dramatic improvement, narrowing the performance gap by 94.500%. In contrast, RF/Synapse reduced the accuracy difference from 4.454% to 0.227% after applying PFI features, Which is 94.900%. These results make PFI a promising tool for enabling efficient predictive modeling in real-world scenarios.

The results highlight the potential of PFI in narrowing the performance gap between different methods without incurring high computational overheads. This approach provides SE practitioners with a practical solution to utilize existing computational infrastructure efficiently toward near-optimal performance. This approach helps resource-constrained environments, such as small-scale software development projects or restricted access to high-performance computing systems in an organization. Another valuable characteristic of PFI

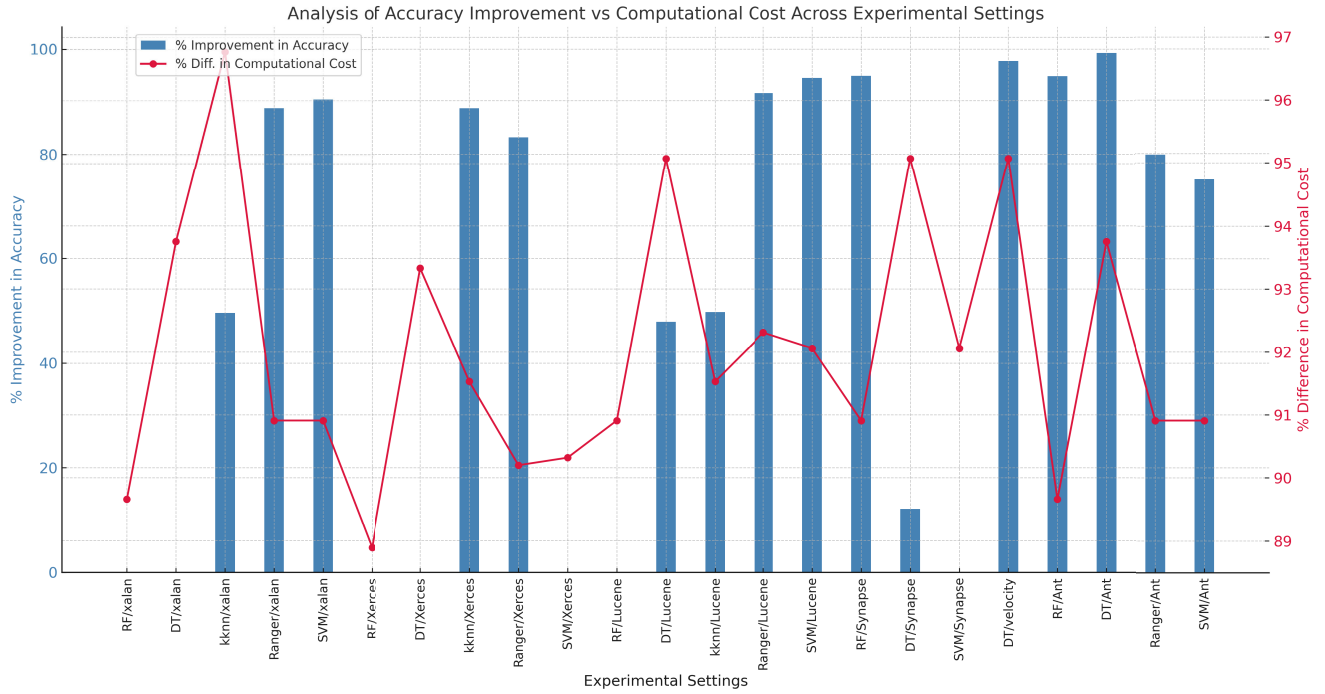

FIGURE 6. Trade-off between Accuracy and computational cost(percentages) across different experiments.

for software fault prediction is its ability to prioritize critical features. This would allow for better resource usage and better-quality software to be built. For instance, in the experiment, KKNN/Lucene exhibited 49.780% improvement, accompanied by computational time narrowed down by 91.530%. In addition, DT/Lucene accuracy improved by 47.910%, and computational cost was reduced by 95.080%. These are aligned with the larger goals of cost-effective ML applications in the domain of SE.

However, with the improved results, it should be noted that, in a few cases, there is limited or no improvement using PFI-selected features. The experiments include RF/Xalan, DT/Xalan, RF/Xerces, DT/Xerces, SVM/Xerces, RF/Lucene, and SVM/Synapse. This lack of improvement may be attributed to dataset-specific characteristics or model-specific limitations, such as imbalanced distributions, noisy data, or the inherent complexity of specific feature interactions. Similarly, the ML model's nature might contribute to this result because some models are less sensitive to feature importance rankings provided by PFI.

For instance, in RF/Xalan, time saved in processing was minimal, and improvements in accuracy were 5.216%. Similarly, SVM/Xerces showed negligible changes in accuracy from 1.215% (before applying PFI) to 1.791% (after applying PFI). This means the characteristics of the dataset might determine the effectiveness of PFI. Further investigation is needed to understand the dataset's properties and the model's behavior while addressing these challenges. Further study into complementary feature selection techniques should also be undertaken. This study highlights the utility of techniques like PFI in XAI to address this trade-off in performance and efficiency. Beyond predicting software faults, the insights can be expanded to other domains, such as effort estimation and resource allocation, where high computational cost remains a critical determinant. These findings set the course for further improving cost-effective predictive modeling in the domain of SE by coming up with a scalable and easy-to-access approach (see Fig. 6).

A. COMPARISON OF THE PROPOSED METHOD WITH PRIOR STUDIES

We compared the performance of our proposed method with the previous studies referred to in the Background and Related Work section.

- Tantithamthavorn et al. [2] demonstrated that optimization of parameters improved performance up to 40% AUC but had high computational costs. At the same time, our approach yields similar improvements in performance with default settings using PFI-selected features and significantly reduces the computational load.
- · Jiang et al. [29], Tosun et al. [11] highlight the inefficiency of default parameters. However, by incorporating feature importance, our method achieves a balance and shows how default settings are practical when the key features are added; for instance, RF/Synapse reduced accuracy gaps from 4.450% to 0.230%, reducing the computational cost by 10.55
- Fu et al. [34] have also reported that 80% of the 50 highly-cited defect prediction studies used default parameters for time efficiency and simplicity. This trend

underscores a great need for methods, such as the one proposed in this study, that balance interpretability with simplicity and computational efficiency but maintain high predictive accuracy.

Our study highlights the utility of techniques like PFI in XAI to address this trade-off in performance and efficiency, as shown in Fig.6. Beyond predicting software faults, the insights can be expanded to other domains, such as effort estimation and resource allocation, where high computational cost remains a critical determinant. These findings set the course for further improving cost-effective predictive modeling in the domain of SE by coming up with a scalable and easy-to-access approach.

VI. THREATS TO VALIDITY

This study is based on six datasets from the Tera Promise database [27], which includes large and small datasets. However, it is crucial to note that different results may be obtained using various SFP datasets from other repositories. Furthermore, the study focuses on regression-based SFP datasets, leaving classification-based SFP datasets that can produce different results. Moreover, five different types of ML models were used, and it is left to question whether these results can be generalized to other models or SE tasks. Even though PFI was shown to be helpful in this work, the fact that it is susceptible to dataset characteristics and quality variations necessitates further exploration to understand its potential and limitations fully. Despite these limitations, the study provides detailed experimental descriptions, allowing researchers to replicate and build upon findings in new contexts. We acknowledge the following as potential threats to validity that may impact the results of this study:

A. INTERNAL VALIDITY

While the random search method was used in this study, the choice of parameter ranges could introduce some bias into the results. Further studies should consider the alternative techniques and extended parameter space.

B. EXTERNAL VALIDITY

The results are based on particular datasets (Ant, Xalan, Xerces, Lucene, Ant, Synapse) and SFP tasks. Although these datasets are widely used in SE research, further work is needed to generalize the findings to other domains, such as effort estimation or resource allocation. In addition, this study considered five ML models, although these are already well-established in SFP. However, their performance and feature importance might not translate to more complex or newly developed models, such as deep learning approaches.

C. CONSTRUCT VALIDITY

The PFI technique assumes that important features directly influence model predictions but may be sensitive to collinearity. This could lead to misinterpretation of rankings, so using other feature selection techniques would be beneficial.

D. CONCLUSION VALIDITY

The study primarily used accuracy and computational cost as the evaluation metrics for SFP, but more metrics such as F1-score, precision, and recall might help better understand model performance, especially when dealing with imbalanced datasets.

VII. CONCLUSION

This study presents a novel methodology that leverages PFI to achieve the predictive performance of optimized parameter settings while significantly reducing the computational cost of default settings in SFP datasets. We conducted extensive experiments using the PFI (XAI method) across five ML models (DT, KNN, Ranger, RF, and SVM) and six diverse SFP datasets. The results have shown that the proposed approach considerably improves model performance without the high computational costs typically associated with parameter optimization. The results confirm that the proposed approach overcomes the trade-off between predictive performance and computational efficiency. In general, it showed comparable accuracies to optimized settings while realizing computational cost savings of as much as 99.250% in certain instances. For instance, in the DT/Ant experiment, the accuracy difference between default and optimized settings was reduced from 40.480% to 0.331%, alongside a drop in computational cost of 99.250%. The results demonstrate promising applicability in real-world situations within resource constraints and a cost-effective solution to the tasks in SE. To the best of our knowledge, this study is the first to leverage PFI to bridge the gap between high accuracy from optimized settings and the efficiency of default settings by providing an adaptive framework that can be applied to a wide range of SE tasks. The experiments validate the effectiveness of this approach in SFP, and further avenues of research are opened in terms of its adaptability in other domains.

VIII. FUTURE WORK

For future work, we aim to extend this methodology to other domains, such as code smell detection and mutation testing, to see if the findings persist. We also plan to expand it to further ML models and PFI methods like Incremental Permutation Feature Importance (iPFI) [61], DEPICT: Diffusion-Enabled Permutation Importance [65], Conditional permutation importance (CPI) [66]. etc. Such extensions would demonstrate how scalable and adaptable the method is within various contexts and techniques.

APPENDIX A

(The table only includes those experiments where optimized settings improved performance, excluding cases where default settings outperformed or both default and hyperparameter settings showed the same performance.)

TABLE 4. Comparison of experimental settings, MAE, Accuracy, and computational cost.

Experimental Settings	% Diff. in Accuracy (Before PFI)	% Diff. in Accuracy (After PFI)	Absolute Changes in MAE	Relative Improvement (%)	Improvement/ No improvement	Processing Time (Default with PFI Features)	Processing Time (Optimized, 50 Iterations)	% Diff. in Computational Cost
RF/xalan	0.521	5.216	4.952	901.340	No Improvement	16 sec	42 sec	89.660%
DT/xalan	0.111	0.332	0.222	200.380	No Improvement	17 sec	47 sec	93.750%
kknn/xalan	3.174	1.600	1.574	49.590	Improvement	16 sec	46 sec	96.770%
Ranger/xalan	2.171	0.240	1.930	88.780	Improvement	15 sec	40 sec	90.910%
SVM/xalan	10.328	0.985	9.343	90.460	Improvement	18 sec	48 sec	90.910%
RF/Xerces	0.622	6.913	6.290	1010.320	No Improvement	15 sec	39 sec	88.890%
DT/Xerces	3.367	3.461	0.094	2.800	No Improvement	16 sec	44 sec	93.330%
kknn/Xerces	3.005	0.338	2.667	88.750	Improvement	16 sec	43 sec	91.530%
Ranger/Xerces	1.503	0.252	1.251	83.230	Improvement	14 sec	37 sec	90.200%
SVM/Xerces	1.215	1.791	0.576	47.370	No Improvement	17 sec	45 sec	90.320%
RF/Lucene	0.851	0.858	0.007	0.860	No Improvement	15 sec	40 sec	90.910%
DT/Lucene	1.680	0.875	0.804	47.910	Improvement	16 sec	45 sec	95.080%
kknn/Lucene	1.785	0.896	0.888	49.780	Improvement	16 sec	43 sec	91.530%
Ranger/Lucene	26.550	2.220	24.330	91.640	Improvement	14 sec	38 sec	92.31%
SVM/Lucene	27.792	1.500	26.291	94.500	Improvement	17 sec	46 sec	92.061%
RF/Synapse	4.454	0.227	4.227	94.900	Improvement	15 sec	40 sec	90.910%
DT/Synapse	0.225	0.197	0.027	12.130	Improvement	16 sec	45 sec	95.080%
SVM/Synapse	0.196	2.035	1.839	934.510	No Improvement	17 sec	46 sec	92.061%
DT/velocity	8.532	0.192	8.340	97.740	Improvement	16 sec	45 sec	95.080%
RF/Ant	4.440	0.230	4.210	94.820	Improvement	16 sec	42 sec	89.660%
DT/Ant	40.480	0.331	40.180	99.250	Improvement	17 sec	47 sec	93.750%
Ranger/Ant	1.980	0.400	1.580	79.760	Improvement	15 sec	40 sec	90.910%
SVM/Ant	8.990	2.230	6.760	75.150	Improvement	18 sec	48 sec	90.910%

REFERENCES

[1] A. Ali and C. Gravino, "The impact of parameters optimization in software prediction models," in *Proc. 48th Euromicro Conf. Softw. Eng. Adv. Appl. (SEAA)*, Aug. 2022, pp. 217–224.

[2] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. Matsumoto, "The impact of automated parameter optimization on defect prediction models," *IEEE Trans. Softw. Eng.*, vol. 45, no. 7, pp. 683–711, Jul. 2019.

[3] S. Pandey and K. Kumar, "Software fault prediction for imbalanced data: A survey on recent developments," *Proc. Comput. Sci.*, vol. 218, pp. 1815–1824, Aug. 2023.

[4] S. Kaliraj, V. G. P. Sahasranth, and V. Sivakumar, "A holistic approach to software fault prediction with dynamic classification," *Automated Softw. Eng.*, vol. 31, no. 2, p. 70, Nov. 2024.

[5] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, vol. 415, pp. 295–316, Nov. 2020.

[6] G. Menghani, "Efficient deep learning: A survey on making deep learning models smaller, faster, and better," *ACM Comput. Surv.*, vol. 55, no. 12, pp. 1–37, Dec. 2023.

[7] N. S. Thomas and S. Kaliraj, "An improved and optimized random forest based approach to predict the software faults," *Social Netw. Comput. Sci.*, vol. 5, no. 5, p. 530, May 2024.

[8] T. Zhou, X. Sun, X. Xia, B. Li, and X. Chen, "Improving defect prediction with deep forest," *Inf. Softw. Technol.*, vol. 114, pp. 204–216, Oct. 2019.

[9] N. Nikravesh and M. R. Keyvanpour, "Parameter tuning for software fault prediction with different variants of differential evolution," *Expert Syst. Appl.*, vol. 237, Mar. 2024, Art. no. 121251.

[10] R. Kaur and R. Vaithiyanathan, "Hybrid YSGOA and neural networks based software failure prediction in cloud systems," *Sci. Rep.*, vol. 14, no. 1, p. 16035, Jul. 2024.

[11] A. Tosun and A. Bener, "Reducing false alarms in software defect prediction by decision threshold optimization," in *Proc. 3rd Int. Symp. Empirical Softw. Eng. Meas.*, Oct. 2009, pp. 477–480.

[12] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, no. 1, pp. 281–305, Mar. 2012.

[13] M. Du, N. Liu, and X. Hu, "Techniques for interpretable machine learning," *Commun. ACM*, vol. 63, no. 1, pp. 68–77, Dec. 2019.

[14] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera, "Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI," *Inf. Fusion*, vol. 58, pp. 82–115, Jun. 2020.

[15] Z. C. Lipton, "The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery," *Queue*, vol. 16, no. 3, pp. 31–57, Jun. 2018.

[16] M. Ribeiro, S. Singh, and C. Guestrin, "'Why should I trust you?': Explaining the predictions of any classifier," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Demonstrations*, 2016, pp. 1135–1144.

[17] M. R. Santos, A. Guedes, and I. Sanchez-Gendriz, "SHapley additive exPlanations (SHAP) for efficient feature selection in rolling bearing fault diagnosis," *Mach. Learn. Knowl. Extraction*, vol. 6, no. 1, pp. 316–341, Feb. 2024.

[18] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, pp. 5–32, Jul. 2001.

[19] C. Strobl, A.-L. Boulesteix, A. Zeileis, and T. Hothorn, "Bias in random forest variable importance measures: Illustrations, sources and a solution," *BMC Bioinf.*, vol. 8, no. 1, pp. 1–21, Dec. 2007.

[20] R. Zhu, D. Zeng, and M. R. Kosorok, "Reinforcement learning trees," *J. Amer. Stat. Assoc.*, vol. 110, no. 512, pp. 1770–1784, Apr. 2015.

[21] B. Gregorutti, B. Michel, and P. Saint-Pierre, "Correlation and variable importance in random forests," *Statist. Comput.*, vol. 27, no. 3, pp. 659–678, May 2017.

[22] A. Fisher, C. Rudin, and F. Dominici, "All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously," *J. Mach. Learn. Res.*, vol. 20, no. 177, pp. 1–81, 2019.

[23] G. König, C. Molnar, B. Bischl, and M. Grosse-Wentrup, "Relative feature importance," in *Proc. 25th Int. Conf. Pattern Recognit. (ICPR)*, Jan. 2021, pp. 9318–9325.

[24] J. Kang, S. Kwon, D. Ryu, and J. Baik, "HASPO: Harmony search-based parameter optimization for just-in-time software defect prediction in maritime software," *Appl. Sci.*, vol. 11, no. 5, p. 2002, Feb. 2021.

[25] A. G. Koru and H. Liu, "An investigation of the effect of module size on defect prediction using static measures," in *Proc. Workshop Predictor Models Softw. Eng.*, May 2005, pp. 1–5.

[26] T. Mende, "Replication of defect prediction studies: Problems, pitfalls and recommendations," in *Proc. 6th Int. Conf. Predictive Models Softw. Eng.*, Sep. 2010, pp. 1–10.

[27] T. Mende and R. Koschke, "Revisiting the evaluation of defect prediction models," in *Proc. 5th Int. Conf. Predictor Models Softw. Eng.*, May 2009, pp. 1–10.

[28] D. R. Ibrahim, R. Ghnemat, and A. Hudaib, "Software defect prediction using feature selection and random forest algorithm," in *Proc. Int. Conf. New Trends Comput. Sci. (ICTCS)*, Oct. 2017, pp. 252–257.

[29] Y. Jiang, B. Cukic, and T. Menzies, "Can data transformation help in the detection of fault-prone modules?" in *Proc. Workshop Defects Large Softw. Syst.*, Jul. 2008, pp. 16–20.

[30] R Core Team, *R: A Language and Environment for Statistical Computing*, Vienna, Austria: R Foundation for Statistical Computing, 2013.

[31] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: An update," *ACM SIGKDD Explor. Newslett.*, vol. 11, no. 1, pp. 10–18, Jun. 2009.

[32] MathWorks Inc., *MATLAB 8.5.0 (R2015a)*. Natick, MA, USA: MathWorks, 2015.

[33] T. Mende, R. Koschke, and M. Leszak, "Evaluating defect prediction models for a large evolving software system," in *Proc. 13th Eur. Conf. Softw. Maintenance Reeng.*, 2009, pp. 247–250.

[34] W. Fu, T. Menzies, and X. Shen, "Tuning for software analytics: Is it really necessary?" *Inf. Softw. Technol.*, vol. 76, pp. 135–146, Aug. 2016.

[35] A. M. Musolf, E. R. Holzinger, J. D. Malley, and J. E. Bailey-Wilson, "What makes a good prediction? Feature importance and beginning to open the black box of machine learning in genetics," *Hum. Genet.*, vol. 141, no. 9, pp. 1515–1528, Sep. 2022.

[36] D. Justus, J. Brennan, S. Bonner, and A. S. McGough, "Predicting the computational cost of deep learning models," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2018, pp. 3873–3882.

[37] C. Gianoglio, E. Ragusa, P. Gastaldo, and M. Valle, "Trade-off between accuracy and computational cost with neural architecture search: A novel strategy for tactile sensing design," *IEEE Sensors Lett.*, vol. 7, no. 5, pp. 1–4, May 2023.

[38] M. Kanwal, M. M. Riaz, S. S. Ali, and A. Ghafoor, "Saliency-based fabric defect detection via bag-of-words model," *Signal, Image Video Process.*, vol. 17, no. 4, pp. 1687–1693, Jun. 2023.

[39] K. Li, Z. Xiang, T. Chen, S. Wang, and K. C. Tan, "Understanding the automated parameter optimization on transfer learning for cross-project defect prediction: An empirical study," in *Proc. IEEE/ACM 42nd Int. Conf. Softw. Eng. (ICSE)*, Oct. 2020, pp. 566–577.

[40] A. Ali and C. Gravino, "Evaluating the impact of feature selection consistency in software prediction," *Sci. Comput. Program.*, vol. 213, Jan. 2022, Art. no. 102715.

[41] J. C. Bean, "Genetic algorithms and random keys for sequencing and optimization," *ORSA J. Comput.*, vol. 6, no. 2, pp. 154–160, May 1994.

[42] L. I. Kuncheva and J. C. Bezdek, "Nearest prototype classification: Clustering, genetic algorithms, or random search?" *IEEE Trans. Syst., Man Cybern., C, Appl. Rev.*, vol. 28, no. 1, pp. 160–164, Jan. 1998.

[43] P. Probst, M. N. Wright, and A. Boulesteix, "Hyperparameters and tuning strategies for random forest," *WIREs Data Mining Knowl. Discovery*, vol. 9, no. 3, p. 1301, May 2019.

[44] L. Li and A. Talwalkar, "Random search and reproducibility for neural architecture search," in *Proc. Uncertainty Artif. Intell.*, Jan. 2019, pp. 367–377.

[45] A. A. Khan, O. Chaudhari, and R. Chandra, "A review of ensemble learning and data augmentation models for class imbalanced problems: Combination, implementation and evaluation," *Expert Syst. Appl.*, vol. 244, Jun. 2024, Art. no. 122778.

[46] G. R. Choudhary, S. Kumar, K. Kumar, A. Mishra, and C. Catal, "Empirical analysis of change metrics for software fault prediction," *Comput. Electr. Eng.*, vol. 67, pp. 15–24, Apr. 2018.

[47] P. Manchala and M. Bisi, "Diversity based imbalance learning approach for software fault prediction using machine learning models," *Appl. Soft Comput.*, vol. 124, Jul. 2022, Art. no. 109069.

[48] K. Phung, E. Ogunshile, and M. E. Aydin, "Domain-specific implications of error-type metrics in risk-based software fault prediction," *Softw. Quality J.*, vol. 33, no. 1, pp. 1–41, Mar. 2025.

[49] N. Deng, Y. Tian, and C. Zhang, *Support Vector Machines: Optimization Based Theory, Algorithms, and Extensions*. Boca Raton, FL, USA: CRC Press, 2012.

[50] A. Navada, A. N. Ansari, S. Patil, and B. A. Sonkamble, "Overview of use of decision tree algorithms in machine learning," in *Proc. IEEE Control Syst. Graduate Res. Colloq.*, Jun. 2011, pp. 37–42.

[51] K. Taunk, S. De, S. Verma, and A. Swetapadma, "A brief review of nearest neighbor algorithm for learning and classification," in *Proc. Int. Conf. Intell. Comput. Control Syst. (ICCS)*, May 2019, pp. 1255–1260.

[52] Y. Liu, Y. Wang, and J. Zhang, "New machine learning algorithm: Random forest," in *Proc. 3rd Int. Conf. Inf. Comput. Appl. (ICICA)*, Chengde, China. Cham, Switzerland: Springer, Jan. 2012, pp. 246–252.

[53] M. N. Wright and A. Ziegler, "Ranger: A fast implementation of random forests for high dimensional data in C++ and R," 2015, *arXiv:1508.04409*.

[54] M. Shepperd and S. MacDonell, "Evaluating prediction systems in software project estimation," *Inf. Softw. Technol.*, vol. 54, no. 8, pp. 820–827, Aug. 2012.

[55] W. B. Langdon, J. Dolado, F. Sarro, and M. Harman, "Exact mean absolute error of baseline predictor, MARP0," *Inf. Softw. Technol.*, vol. 73, pp. 16–18, May 2016.

[56] A. Vargha and H. D. Delaney, "A critique and improvement of the CL common language effect size statistics of McGraw and wong," *J. Educ. Behav. Statist.*, vol. 25, no. 2, pp. 101–132, Jun. 2000.

[57] H. K. Dam, T. Pham, S. Wee Ng, T. Tran, J. Grundy, A. Ghose, T. Kim, and C.-J. Kim, "A deep tree-based model for software defect prediction," 2018, *arXiv:1802.00921*.

[58] C. Molnar, G. König, B. Bischl, and G. Casalicchio, "Model-agnostic feature importance and effects with dependent features: A conditional subgroup approach," *Data Mining Knowl. Discovery*, vol. 38, no. 5, pp. 2903–2941, Sep. 2024.

[59] M. Saarela and S. Jauhiainen, "Comparison of feature importance measures as explanations for classification models," *Social Netw. Appl. Sci.*, vol. 3, no. 2, p. 272, Feb. 2021.

[60] A. Altmann, L. Toloşi, O. Sander, and T. Lengauer, "Permutation importance: A corrected feature importance measure," *Bioinformatics*, vol. 26, no. 10, pp. 1340–1347, May 2010.

[61] F. Fumagalli, M. Muschalik, E. Hüllermeier, and B. Hammer, "Incremental permutation feature importance (iPFI): Towards online explanations on data streams," *Mach. Learn.*, vol. 112, no. 12, pp. 4863–4903, Dec. 2023.

[62] G. K. Rajbahadur, S. Wang, G. A. Oliva, Y. Kamei, and A. E. Hassan, "The impact of feature importance methods on the interpretation of defect classifiers," *IEEE Trans. Softw. Eng.*, vol. 48, no. 7, pp. 2245–2261, Jul. 2022.

[63] Y. Zhao, Z. Huang, L. Gong, Y. Zhu, Q. Yu, and Y. Gao, "Evaluating the impact of data transformation techniques on the performance and interpretability of software defect prediction models," *IET Softw.*, vol. 2023, no. 1, Jan. 2023, Art. no. 6293074.

[64] G. Canfora, A. De Lucia, M. Di Penta, R. Oliveto, A. Panichella, and S. Panichella, "Multi-objective cross-project defect prediction," in *Proc. IEEE 6th Int. Conf. Softw. Test., Verification Validation*, Mar. 2013, pp. 252–261.

[65] S. Jabbour, G. Kondas, E. Kazerooni, M. Sjoding, D. Fouhey, and J. Wiens, "DEPICT: Diffusion-enabled permutation importance for image classification tasks," 2024, *arXiv:2407.14509*.

[66] D. Debeer and C. Strobl, "Conditional permutation importance revisited," *BMC Bioinf.*, vol. 21, no. 1, pp. 1–30, Dec. 2020.

ADAM KHAN received the B.S. degree in information technology from Gomal University, Dera Ismail Khan, Pakistan, in 2006, and the M.S. degree in computer science from Abasyn University, Peshawar, Pakistan, in 2013. He is currently a Ph.D. Scholar in computer science with the Department of Computer Science and IT, Sarhad University of Science & Information Technology, Peshawar. His research interests include explainable AI, machine learning, and empirical software engineering.

ASAD ALI received the M.Sc. degree in computer engineering from Near East University, Cyprus, in 2014, and the Ph.D. degree in computer science and information engineering from the University of Salerno, Italy, in 2021. He is currently an Assistant Professor in computer science with the Sarhad University of Science & Information Technology, Peshawar, Pakistan. His research interests include empirical software engineering, machine learning, and explainable artificial intelligence.

JAHANGIR KHAN received the M.S. degree from the University of Sindh, Jamshoro, Sindh, Pakistan, in 2005, the Ph.D. degree in information technology from China Agricultural University, Beijing, in 2018, and the Postdoctoral degree from the Engineering Academy of Serbia, Serbia. He is currently a Professor and the Head of the Department of Computer Science and IT, Sarhad University of Science & Information Technology, Peshawar, Pakistan. His research interests include the application of IT using satellite remote sensing and AI, climate change, and natural hazards and explainable AI.

FASEE ULLAH received the Graduate degree from the Faculty of Computing, Universiti Teknologi Malaysia (UTM), Malaysia, in 2017. He has also completed the postdoctoral fellowship from the University of Macau (2019–2021), which is the academic talented program of the Government of Macau. Currently, he is an Associate Professor with the Department of Computer and Information Sciences, Universiti Teknologi PETRONAS, Seri Iskandar, Perak Darul Ridzuan, Malaysia. He has published many research papers in reputed impact factor journals and conferences. He was a recipient of the Chancellor Award and the Best Student Award at UTM during the Ph.D. degree, for his excellent research contributions to wireless communication and health monitoring. His research interests include wireless body area network, wireless sensor networks, cloud security, smart hash security designing, smart cities, big data analytics, machine learning, deep learning, and the Internet of Things. He is currently providing reviewing services to IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING, IEEE TRANSACTIONS ON CLOUD COMPUTING, IEEE ACCESS, IEEE SENSOR JOURNAL, ACM, and *International Journal of Distributed Sensor Networks*.

MUHAMMAD FAHEEM (Member, IEEE) is a Senior Scientist at VTT Technical Research Centre of Finland. He has held various academic positions at Abdullah Gül University, Turkey, COMSATS University, Pakistan, and Universiti Teknologi Malaysia. He served as an Assistant Professor, a Program Manager for the Master's in Robotics, and Academy of Finland Researcher at the University of Vaasa. His research expertise spans cybersecurity, AI, ML, BC, SGs, and the IoT. He specializes in designing, modeling, developing, and piloting ML and BCbased solutions for smart grid applications. He has received several prestigious awards, including the Top 2% Scientist Award from Stanford University, the Highly Cited Scientist Award from Elsevier, and the Top 1% Turkish Scientist in the World Award. He is actively involved in editorial boards of reputed journals such as *Sustainable Futures* (Elsevier), *PLOS ONE*, *Frontiers in the Internet of Things*, *Frontiers in Artificial Intelligence, and Computers*, *Materials and Continua*, among others. He has also served as a Lead Guest Editor for special issues on AI, ML, and BC technologies in leading journals, including IET, Wiley, and MDPI. He is a reviewer for prestigious publications by ACM, IEEE, Springer, Wiley, and Elsevier. He has chaired sessions at several IEEE conferences, including AIE2024 (Finland) and CAC2023 (Malaysia). Additionally, he is an active member of ERIGrid 2.0, the JUFO panel (evaluating journal quality and standards), and AI-Doc (evaluating doctoral students) in Finland.