International Journal on Advanced Science Engineering Information Technology

Vol.15 (2025) No. 4
ISSN: 2088-5334

# Impact of Feature Set Size on the Performance of Machine Learning Models in Cross-Project Defect Prediction

Yahaya Zakariyau Bala [a], Pathiah Abdul Samat [a,*], Khaironi Yatim Sharif [b], Noridayu Manshor [c]

[a] *Software Engineering, Universiti Putra Malaysia Selangor, Malaysia*
[b] *Computer & Information Science, Universiti Teknologi PETRONAS, Perak, Malaysia*
[c] *Computer Science, Universiti Putra Malaysia Selangor, Malaysia*
*Corresponding author: *pathiah@upm.edu.my*

***Abstract*—Software defect prediction is a vital area in software engineering that helps developers detect potential faults before software is deployed. Cross-Project Defect Prediction (CPDP) is particularly valuable, as it enables the use of defect data from one project to predict errors in another, making it beneficial in cases where project-specific defect data is insufficient. However, the effectiveness of CPDP largely depends on how well the machine learning models are trained, and a key factor influencing their performance is the size of the feature set used. This study focuses on evaluating the impact of feature set size on the performance of two widely used machine learning models, Random Forest (RF) and Support Vector Machine (SVM), in the context of CPDP. We used defect datasets from the AEEEM repository, which consists of multiple real-world software projects. An outlier detection technique was applied to select the number of features in the training and testing data, ensuring a systematic analysis of their impact on model performance. The F1-score was used as the primary evaluation metric, as it provides a balance between precision and recall, making it a reliable measure of defect prediction accuracy. Our findings suggest that the size of the feature set plays a crucial role in determining the effectiveness of both RF and SVM models. Too many features introduce noise, reducing predictive accuracy, while too few cause underfitting, leading to the missed detection of defect patterns. Identifying an optimal feature set size improves model performance, providing practical insights for enhancing CPDP. Optimizing feature selection can lead to more accurate predictions, thereby aiding software maintenance and enhancing overall software quality.**

***Keywords*— Software; defect prediction; cross-project; features selection.**

*Manuscript received 12 Oct. 2024; revised 2 Feb. 2025; accepted 5 Jul. 2025. Date of publication 31 Aug. 2025.*


## I. INTRODUCTION

As technology advanced, so too has the use of software. Software with more benefits is typically more complex. The number of defects in software is closely correlated with its complexity [1], [2]. Debugging and testing must be conducted extensively to produce dependable software. Predicting defects in code is vital for maintaining software quality [3]-[5]. Software defect prediction (SDP) models can effectively identify software defects, saving money and effort on testing [6]. However, SDP models tend to perform optimally within the context of the project from which the data was collected, known as within-project defect prediction (WPDP). Applying these models to a new project with different characteristics and codebases often results in reduced prediction accuracy [7], [8].

Cross-project defect prediction (CPDP) models have emerged as an effective approach to predict defects by utilizing data from other projects, especially when project-specific data is scarce [9]-[11]. However, the performance of these models largely depends on the features used during training [12]. Feature selection, which involves identifying the most relevant features for prediction, can significantly influence the effectiveness of CPDP models [13], [14]. Balogun, et al. [15] Vescan, et al. [16], Manchala and Bisi [17], Bala, et al. [18], Rathore and Gupta [19], conducted a study on impact of feature methods on software defect prediction. They reported that feature selection methods improve the performance of SDP models.

However, feature selection methods focus on improving model performance by choosing the most relevant features. Therefore, investigating the impact of feature set size provides a broader understanding of model behavior, efficiency, and applicability in diverse and resource-

constrained environments typical of CPDP scenarios. Investigating the impact of feature set size on machine learning models in CPDP is crucial for optimizing model performance, improving generalization across projects, and ensuring model efficiency. It will also help balance the inclusion of diverse metrics, guiding feature engineering, which will lead to more accurate and reliable defect predictions.

This study explores how the number of features in a dataset affects the performance of machine learning models in CPDP settings. To measure this impact, we analyze changes in the commonly used evaluation metric F1-score. We test models trained on different feature set sizes ranging from small to large sets to determine whether increasing the number of features improves defect prediction or causes issues like overfitting and the curse of dimensionality. The novelty of this study lies in its systematic examination of feature set size variations across multiple real-world datasets using two distinct machine learning models, which provides a clearer understanding of their behavior in CPDP settings.

Accurate defect prediction is crucial for reducing software maintenance costs and improving reliability. In CPDP, selecting the right features plays a key role in ensuring models perform well across different projects. However, adding too many features can introduce noise, increase computational costs, and reduce accuracy. This study provides concrete evidence of how feature set sizes impacts defect prediction, helping software engineers optimize their models. Our findings offer practical guidance on balancing model complexity and accuracy, leading to more efficient defect detection in real-world software development

## II. Materials and Method

### A. Data Collection

We collected defect data from five open-source projects, ensuring a diverse representation of software systems. AEEEM datasets were introduced by [20]. The AEEEM datasets are a collection of software defect datasets that are commonly used in the research community for evaluating software defect prediction models. These datasets are derived from five different open-source software projects, providing a variety of data points for analysis. The AEEEM datasets are particularly useful for evaluating the impact of feature set size on machine learning performance in CPDP for two reasons: First, the datasets originate from diverse projects with different characteristics, which helps generalize the findings across various types of software systems. Second, they include a wide range of features (static code metrics, change metrics, and process metrics), which allows for the exploration of how different feature sets impact the performance of defect prediction models.

Table 1 provides a detail of the datasets used in the experimental analysis. The table lists five different projects, detailing the number of modules, the total number of features, and the count of defective modules within each project. This information is crucial for understanding the scale and characteristics of the datasets employed in the Cross-Project Defect Prediction (CPDP) experiments.

TABLE I
AEEEM DATASETS

| Project Name | Module | Features | Defective Modules |
|---|---|---|---|
| Equinox (EQ) | 325 | 71 | 129 |
| Eclipse JTD core (JTD) | 997 | 71 | 206 |
| Eclipse PDE UI (PDE) | 1862 | 71 | 245 |
| Mylyn (ML) | 1492 | 71 | 209 |
| Apache Lucene (LC) | 399 | 71 | 64 |

### B. Machine Learning Models

Machine learning is a branch of artificial intelligence that enables systems to learn and make decisions from data without explicit programming. It involves developing algorithms that can generalize from known examples, allowing predictions or decisions to be made about new, unseen data. This capability is particularly valuable in software engineering for tasks like defect prediction, where the goal is to identify potential faults in software based on historical data [21], [22].

In cross-project defect prediction (CPDP), models trained on data from one or more projects are used to predict defects in a different, unseen project. This approach is crucial because it allows organizations to leverage existing data to enhance the reliability of new projects. However, CPDP presents unique challenges due to differences in data distributions, feature spaces, and project characteristics across different software projects.

One critical factor that influences the performance of machine learning models in CPDP is the feature set size [23]. The number of features used to train a model can significantly impact its accuracy, generalizability, and computational efficiency. A larger feature set may provide more information, but can also lead to overfitting and increased computational complexity. Conversely, a smaller feature set might reduce model complexity but risk omitting important predictive information.

In this study, Random Forest (RF) and Support Vector Machine (SVM) are chosen because, first, RF is an ensemble learning method that builds multiple decision trees and combines their outputs to improve defect prediction accuracy [24]. It is robust to overfitting, especially when dealing with large feature sets, making it suitable for high-dimensional data typical in CPDP. Second, RF provides an inherent mechanism to assess the importance of features, which is beneficial in understanding the contribution of different features to model performance. This is particularly useful when evaluating the impact of feature set size. Third, CPDP often involves imbalanced datasets, where defective instances are much fewer than non-defective ones. RF handles this well by balancing errors across classes.

Similarly, SVM is known for its ability to perform well in high-dimensional spaces, making it an excellent choice when investigating different feature set sizes. It effectively finds the optimal hyperplane that separates classes, even when the number of features is large. In addition, SVMs' use of regularization parameters helps prevent overfitting, enhancing their generalization capability, which is critical in CPDP where data distributions vary between projects.

Furthermore, SVM can utilize different kernel functions (e.g., linear, polynomial, radial basis function) to map input features into higher-dimensional spaces, allowing for the capture of complex relationships in the data. This versatility is advantageous when exploring the impact of varying feature sets.

### C. Feature Selection Process

Feature selection is a fundamental process in machine learning that involves choosing a subset of relevant features for use in software defect model development [25]. The primary objective of feature selection is to enhance the efficiency and performance of models by reducing the dataset's dimensionality. This not only enhances the accuracy and interpretability of the models but also reduces the risk of overfitting and the computational resources required for training.

High-dimensional datasets often contain features that are redundant, irrelevant, or noisy. These extraneous features can obscure the patterns in the data, leading to models that are less accurate and more complex. Therefore, feature selection aims to identify and retain only those features that contribute the most to the prediction task.

One approach to feature selection leverages statistical principles to identify and remove outliers, which are data points that deviate significantly from other observations. Outliers can distort the learning process, leading to models that do not generalize well to new data. A commonly used statistical principle in this context involves the concept of standard deviation and the central mean.

In statistical terms, the standard deviation measures the amount of variation or dispersion in a set of data points relative to the mean (average) value [26]. When a data point's deviation from the mean exceeds a certain threshold, it is often considered an outlier. The general rule of thumb is that any feature whose standard deviation is significantly greater than the central mean is likely to contain outliers and may be considered for removal during feature selection.

As shown in Fig. 1, the feature selection process involves a series of steps to identify relevant features from a source dataset (S) and then map these features to a target dataset (T). First, the process starts with computing the central mean (mean_s) of the source dataset (S). This value serves as a reference point for evaluating the variability of individual features. Second, the standard deviation (std) of each feature (F) within the source dataset (S) is computed. This metric quantifies how much each feature deviates from the central mean. Third, the process compared each feature's standard deviation to the central mean. Specifically, if the standard deviation of a feature is more than three times the central mean (std > 3 (mean_s)), that feature is classified as an outlier [27], [28]. Outliers are typically excluded from further analysis as they may not provide meaningful information. Features with a standard deviation less than or equal to three times the central mean are considered relevant and retained for the next step. Fourth, among the relevant features identified, a subset of (N) features is selected. Finally, the corresponding features in the target dataset (T) are identified based on the selected features from the source dataset (S). These features are then used in subsequent cross-project defect prediction.

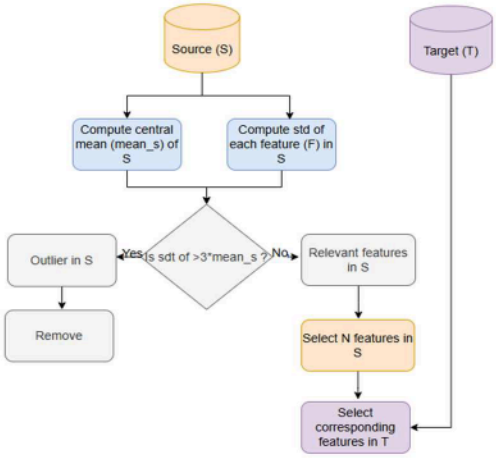


Fig. 1 Feature selection process

### D. Performance Measurement

To evaluate the prediction performance of our technique, we used F1_score which is one of the most used evaluation metrics in SDP [29], [30]. The F1-score is the harmonic mean of precision and recall, providing a single metric that balances the trade-off between these two important aspects of classification performance. We selected F1_score because of two reasons: First, software defect datasets are often imbalanced, with a smaller number of defective modules compared to non-defective ones. The F1-score is particularly suitable in such scenarios because it considers both false positives and false negatives, providing a more balanced evaluation compared to accuracy, which can be misleading in imbalanced datasets. Second, the F1-score provides a single value that summarizes both precision and recall. This simplifies the comparison between models and different feature sets, making it easier to interpret the results. It is calculated as follows.

*1) Precision*: Defined the ratio of the number of software module correctly predicted as non-defective to the total number of modules predicted as non-defective.

$$Precision\ TP/(TP + FP) \quad (1)$$

*2) Recall*: Defined the ratio of the number of modules correctly predicted as non-defective to the number of all non-defective modules.

$$Recall\ TP/(TP + FN) \quad (2)$$

*3) F1-score*: Defined the harmonic representation of precision and recall. The higher F-measure means better performance.

$$F1 - score\ (2 * Precision * Recall)/(Precision + Recall) \quad (3)$$

### E. Statistical Evaluation

To evaluate whether the difference in performance among different feature set size is statistically significant, we performed a pairwise Wilcoxon signed-rank test on the RF and SVM results across various feature set sizes. For instance, we compared the performance across all pairs of feature set sizes (N=9 vs N=12, N=9 vs N=15, N=9 vs N=18, etc.). The test is non-parametric. The following hypothesis were

established. If the p-value is less than 0.05, we reject the null hypothesis and conclude that the difference between the two features set sizes is statistically significant. Otherwise, we accept the null hypothesis.

- Null Hypothesis (H0): There is no significant difference between the two features set sizes
- Alternative Hypothesis (H1): There is a significant difference between the two features set sizes.

### *F. Experimentation Setup*

Fig. 2 provides a step-by-step illustration of the experimental process aimed at predicting defects in a target project using machine learning models. First, the experiment begins with data collection, where the AEEEM dataset is gathered. This dataset serves as the foundation for the subsequent analysis and modeling tasks. Second, the dataset is divided into two groups: the source dataset and the target dataset. The source dataset is used to train machine learning models, while the target dataset is used for testing. Third, relevant features are selected from both the source and target datasets. The selection is performed with varying feature set sizes: 9, 12, 15, 18, 21, and 71. These different sizes enable testing of how the number of features affects the performance of the machine learning models.

Fourth, two machine learning models, RF and SVM, are trained using the selected features from the source dataset. These models are trained to predict defects based on the features provided. The trained RF and SVM models are then applied to the target dataset to predict defects. The models use the relevant features from the target dataset to make these predictions. Fifth, the performance of the models is evaluated using the F1-score, a metric that balances precision and recall, providing a single measure of model accuracy. Results are recorded for each feature set size, allowing for a comparison of model performance across different feature selections.

Programming language: Python was used for data processing, feature selection, and model training. Libraries such as scikit-learn, pandas, numpy, and matplotlib were employed for model building, data handling, and visualization. The experiments were performed on a system with 8 GB of RAM and an ntel(R) Core(TM) i7-4600U processor, running the Windows operating system.

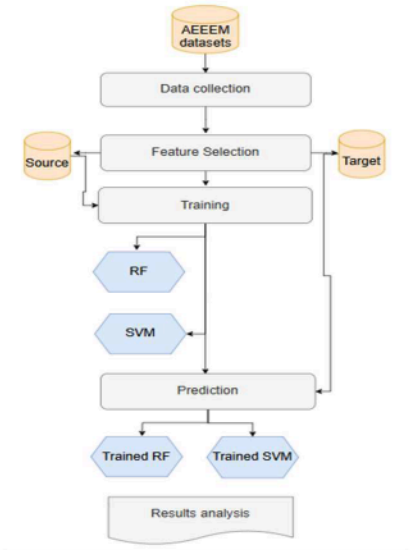


Fig. 2 Investigation process

## III. Results and Discussion

In the context of this investigation, the performance of two machine learning models, Random Forest (RF) and Support Vector Machine (SVM), was analyzed with respect to different feature set sizes. The goal was to understand how varying the number of features impacts the models' ability to predict software defects in a cross-project setting.

For this experiment, feature sets of varying sizes were selected, ranging from (N = 9) to (N = 21), as well as the entire feature set consisting of 71 features. The performance of the models was evaluated using the F1-score, a harmonic mean of precision and recall, which is particularly useful for assessing models on imbalanced datasets, such as those commonly encountered in defect prediction.

The results of RF and SVM models are documented in Tables II and III, respectively, where each table presents the F1-scores achieved by the models across different feature set sizes. These results provide insights into how the inclusion of additional features influences the predictive power of the models and helps identify the optimal feature set size that balances model complexity with performance.

### *A. Random Forest (RF) Performance*

Table II summarizes the F1-scores achieved by the Random Forest model across different feature set sizes. The table illustrates how the model's performance evolves as more features are incorporated into the analysis. The table compares the performance across different feature sizes to assess the impact of feature set size on the model's predictive capability. Each column corresponds to a distinct feature size, with the final row summarizing the mean F1-scores (F1_mean) across all tested feature sizes. This summary provides an overall perspective on the RF model's performance, highlighting the optimal feature size for achieving the best prediction performance.

AS shown in the table, with 9 features, the RF model achieved an F1 mean score of 0.83, indicating a good balance between precision and recall at this feature set size. With 12 features, RF reached its highest F1 mean score of 0.85. This suggests that adding three more features helped improve the model's performance by capturing additional relevant information. With 15 features, the F1 mean score dropped slightly to 0.82, indicating that increasing the feature set beyond 12 may introduce noise or redundant information that negatively impacts performance. With 18 features, the F1 mean score further decreased to 0.81, supporting the observation that including too many features can degrade the model's effectiveness. With 21 features, the F1 mean score returned to 0.83, suggesting some variability in the model's response to additional features, but generally confirming that an optimal feature set size is around 12. With 71 features (all features), the F1 mean score of RF further decreased to 0.77.

As the feature set size increases from (N = 9 ) to ( N = 21), and ultimately to the complete set of 71 features, the F1-score exhibits variations. Initially, an increase in the number of features often leads to improved model performance, as the model gains access to more information, which enables it to make better predictions. However, beyond a certain point, the addition of more features can lead to diminishing returns or

even a decline in performance. This is due to the introduction of irrelevant or redundant features that add noise to the model, potentially causing overfitting. The results of the Random Forest model underscore the importance of determining the optimal number of features. This knowledge enables the development of more robust, accurate, and efficient models, ultimately leading to improved outcomes in cross-project defect prediction tasks.

TABLE II
F1_SCORES OF RF

| Source | Target | N=9 | N=12 | N=15 | N=18 | N=21 | N=71 |
|---|---|---|---|---|---|---|---|
| EQ | JDT | 0.68 | 0.81 | 0.77 | 0.83 | 0.83 | 0.70 |
| EQ | ML | 0.67 | 0.60 | 0.66 | 0.64 | 0.64 | 0.40 |
| EQ | PDE | 0.69 | 0.60 | 0.59 | 0.59 | 0.59 | 0.56 |
| EQ | LC | 0.72 | 0.79 | 0.69 | 0.68 | 0.68 | 0.68 |
| JDT | EQ | 0.63 | 0.73 | 0.71 | 0.72 | 0.72 | 0.73 |
| JDT | ML | 0.82 | 0.78 | 0.82 | 0.82 | 0.82 | 0.80 |
| JDT | LC | 0.70 | 0.86 | 0.88 | 0.81 | 0.81 | 0.78 |
| JDT | PDE | 0.82 | 0.85 | 0.98 | 0.98 | 0.85 | 0.77 |
| ML | EQ | 0.73 | 0.72 | 0.74 | 0.73 | 0.73 | 0.72 |
| ML | JDT | 0.86 | 0.86 | 0.85 | 0.81 | 0.81 | 0.86 |
| ML | PDE | 0.87 | 0.88 | 0.88 | 0.87 | 0.87 | 0.86 |
| ML | LC | 0.92 | 0.93 | 0.90 | 0.90 | 0.90 | 0.81 |
| PDE | EQ | 0.73 | 0.97 | 0.74 | 0.74 | 0.74 | 0.72 |
| PDE | JDT | 0.84 | 0.85 | 0.85 | 0.85 | 0.85 | 0.80 |
| PDE | ML | 0.90 | 0.89 | 0.90 | 0.88 | 0.88 | 0.81 |
| PDE | LC | 0.89 | 0.90 | 0.91 | 0.93 | 0.93 | 0.80 |
| LC | EQ | 0.74 | 0.73 | 0.73 | 0.75 | 0.75 | 0.73 |
| LC | JDT | 0.86 | 0.86 | 0.86 | 0.85 | 0.85 | 0.80 |
| LC | ML | 0.86 | 0.85 | 0.87 | 0.85 | 0.85 | 0.77 |
| LC | PDE | 0.89 | 0.88 | 0.85 | 0.87 | 0.87 | 0.79 |
| MEAN | | 0.79 | 0.82 | 0.81 | 0.81 | 0.80 | 0.74 |

### *B. Support Vector Machine (SVM) Performance*

Table 3 presents the F1-scores obtained by the Support Vector Machine (SVM) classifier when applied to Cross-Project Defect Prediction (CPDP) tasks. The table compares the performance across different feature sizes to assess the impact of feature set size on the model's predictive capability. Each column corresponds to a distinct feature size, with the final row summarizing the mean F1-scores (F1_mean) across all tested feature sizes. This summary provides an overall perspective on the SVM model's performance, highlighting the optimal feature size for achieving the best prediction performance.

TABLE III
F1_SCORES OF SVM

| Source | Target | N=9 | N=12 | N=15 | N=18 | N=21 | N=71 |
|---|---|---|---|---|---|---|---|
| EQ | JDT | 0.86 | 0.86 | 0.86 | 0.87 | 0.87 | 0.80 |
| EQ | ML | 0.87 | 0.87 | 0.85 | 0.82 | 0.82 | 0.80 |
| EQ | PDE | 0.59 | 0.68 | 0.64 | 0.63 | 0.63 | 0.71 |
| EQ | LC | 0.88 | 0.87 | 0.87 | 0.87 | 0.87 | 0.78 |
| JDT | EQ | 0.71 | 0.72 | 0.72 | 0.71 | 0.71 | 0.70 |
| JDT | ML | 0.81 | 0.83 | 0.84 | 0.84 | 0.84 | 0.87 |
| JDT | LC | 0.79 | 0.71 | 0.71 | 0.73 | 0.73 | 0.75 |
| JDT | PDE | 0.79 | 0.76 | 0.83 | 0.84 | 0.86 | 0.77 |
| ML | EQ | 0.66 | 0.66 | 0.65 | 0.68 | 0.68 | 0.68 |
| ML | JDT | 0.84 | 0.84 | 0.84 | 0.84 | 0.84 | 0.82 |
| ML | PDE | 0.72 | 0.73 | 0.73 | 0.74 | 0.74 | 0.68 |
| ML | LC | 0.67 | 0.66 | 0.64 | 0.62 | 0.62 | 0.62 |
| PDE | EQ | 0.71 | 0.78 | 0.70 | 0.70 | 0.70 | 0.71 |
| PDE | JDT | 0.86 | 0.86 | 0.86 | 0.86 | 0.86 | 0.80 |
| PDE | ML | 0.89 | 0.89 | 0.90 | 0.89 | 0.89 | 0.80 |
| PDE | LC | 0.83 | 0.80 | 0.82 | 0.84 | 0.84 | 0.78 |
| LC | EQ | 0.71 | 0.73 | 0.72 | 0.71 | 0.71 | 0.69 |
| LC | JDT | 0.86 | 0.86 | 0.85 | 0.85 | 0.85 | 0.74 |
| LC | ML | 0.83 | 0.84 | 0.86 | 0.76 | 0.76 | 0.71 |
| LC | PDE | 0.79 | 0.79 | 0.80 | 0.78 | 0.78 | 0.75 |
| MEAN | | 0.78 | 0.79 | 0.78 | 0.78 | 0.78 | 0.75 |

As shown in the table, with 9 features, the SVM model achieved an F1 mean score of 0.78, indicating slightly lower performance compared to RF at this feature set size. With 12 features, the SVM's F1 mean score increased to 0.79, showing a modest improvement with the addition of features. With 15 features, the SVM performance remained steady with an F1 mean score of 0.79, suggesting that adding more features beyond this point does not significantly impact its performance. With 18 features, the F1 mean score decreased slightly to 0.78, indicating potential overfitting or the inclusion of irrelevant features. With 21 features, the F1 mean score remained at 0.78, reinforcing that increasing the feature set beyond a certain point does not benefit the SVM model. With 71 features (all features), the F1 mean score of SVM further decreased to 0.75, confirming the importance of having feature selection.

### *C. Comparative Analysis*

Fig. 3 illustrates the mean F1-scores (F1_mean) achieved by the Random Forest (RF) and Support Vector Machine (SVM) classifiers across various feature sizes in Cross-Project Defect Prediction (CPDP) settings. The Fig provides a comparative analysis of the two classifiers, showing how their performance varies as the number of features changes. This visualization helps identify the feature size that maximizes predictive performance for both models.

As shown in Fig. 3, for both RF and SVM, the optimal feature set size appears to be 12 features. This size provides the highest F1 mean score for RF and a slight improvement for SVM. RF consistently outperforms SVM across all feature set sizes, with the most notable difference observed at the 12-feature set, where RF achieves an F1 mean score of 0.82 compared to SVM's F1 mean score of 0.79. Both algorithms show a trend where performance improves with initial increases in the number of features, but then either increases or declines, indicating the importance of careful feature selection to avoid overfitting and noise.

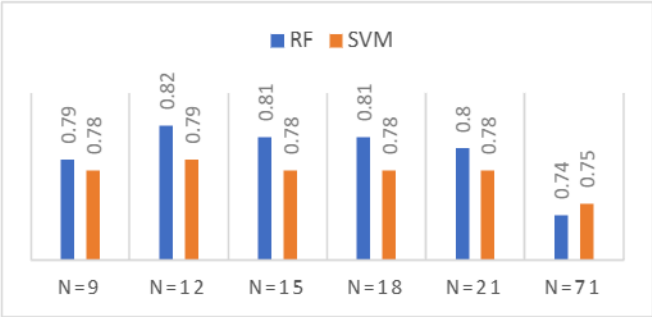


Fig. 3 F1 mean scores of RF and SVM

### *D. Selected Features Analysis*

Fig 4 depicts the frequency with which each metric was selected across five different iterations in the feature selection process. This Fig highlights the consistency and importance of specific metrics in contributing to the model's performance during Cross-Project Defect Prediction (CPDP) settings. Similarly, Table IV provides a detailed account of the metrics selected in each feature size experiment. By examining this table, one can identify which metrics were consistently chosen across different feature sizes and iterations.

As seen in both the Figure and the table, Coupling between Object Classes (CBO), fanIn, fanOut, number of Attributes (NOA), number of Public Methods (NOPM), CvsEntropy, number of Bugs Found (NOBF), number of Non Trivial Bugs Found (NONTBF) were consistently selected across all experiments, indicating their high relevance to defect prediction. This suggests that features related to code complexity and external dependencies are crucial in predicting defects across projects. Similarly, the number of Methods (NOM), the number of Private Attributes (NOPA), the WCHU Coupling between Object Classes (WCHU_CBO), and the WCHU number of Lines of Code (WCHU_LOC) were selected in four out of five experiments, making them moderately relevant. They likely capture aspects of the code structure that are generally predictive but not universally applicable across all projects. The results suggest that focusing on a smaller, highly relevant subset of features (like CBO and Fan-out) can lead to more robust and generalizable models.

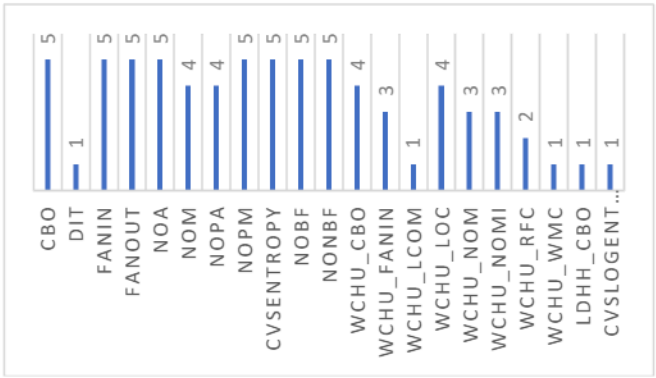


Fig. 4 Frequency of feature selection

TABLE IV
METRICS IN EACH FEATURE SET

| N=9 | N=12 | N=15 | N=18 | N=21 |
|---|---|---|---|---|
| FANIN | FANIN | FANIN | FANIN | FANIN |
| NOM | NOM | CVSENTROPY | NOM | NOM |
| NOA | NOA | NOA | CVSENTROPY | CVSENTROPY |
| CVSENTROPY | CVSENTROPY | NOPM | NOA | NOA |
| NOPM | NOPM | NOM | NOPM | NOPM |
| NOTBF: | FANOUT | FANOUT | FANOUT | FANOUT |
| NOBF | NONTBF | NONTBF | NONTBF | NONTBF |
| FANOUT | NOBF | NOBF | NOBF | NOBF |
| CBO | CBO | NOPA | CBO | CBO |
| | NOPA | WCHU_CBO | NOPA | NOPA |
| | WCHU_CBO | CBO | WCHU_CBO | WCHU_CBO |
| | WCHU_LOC | WCHU_LOC | WCHU_LOC | WCHU_LOC |
| | | WCHU_NOMI | WCHU_NOMI | WCHU_NOMI |
| | | WCHU_FANIN | WCHU_LCOM | WCHU_LCOM |
| | | WCHU_RFC | WCHU_RFC | WCHU_RFC |
| | | | WCHU_FANIN | WCHU_FANIN |
| | | | WCHU_WMC | WCHU_WMC |
| | | | DIT | DIT |
| | | | | NOAI |
| | | | | NOPA |
| | | | | NOC |

## *E. How Dataset Characteristics Affect Model Performance*

In this subsection, we examine how specific dataset characteristics, including project size, application domain, and programming language, significantly influence model performance. The results from different feature set sizes (N=9, 12, 15, 18, 21, 71) highlight how these factors influence prediction accuracy.

*1) Project size:* The size of a software project, measured by the number of modules and the distribution of defects, directly affects how well a model trained on one source project performs when applied to a target project. It can be observed that larger projects, such as PDE with 1,862 modules and ML with 1,492 modules, contain more diverse defect patterns, making them better sources for CPDP when applied to similarly large projects. For instance, when PDE was used as a source project to predict defects in ML, the model performed well across different feature set sizes, with an F1 Score of 0.90 at N=9, 0.89 at N=12, and 0.90 at N=15. Likewise, when ML was used to predict PDE, performance remained high. However, when transferring knowledge between projects of significantly different sizes, prediction accuracy dropped. For instance, EQ with 325 modules did not generalize well when predicting ML with 1,492 modules, with the F1 Score declining from 0.67 (N = 9) to 0.40 (N = 71). This suggests that models trained on small projects struggle to capture the complexity of larger projects, leading to lower prediction performance.

*2) Application Domain:* Differences in application domains also impact CPDP performance. Models tend to perform better when trained and tested on projects within the same domain. For example, Apache Lucene (LC), a text-search engine library, demonstrated consistent prediction performance across various projects. When it was used to predict PDE, accuracy remained stable between 0.89 and 0.87 for feature set sizes N = 9 to N = 21. Similarly, projects from the Eclipse ecosystem, such as ML and PDE, performed well when used interchangeably as source and target projects, with F1_score ranging from 0.90 to 0.88. However, it can be observed that cross-domain predictions perform poorly as well. For instance, when EQ (Eclipse Equinox) was used to predict defects in ML, the accuracy consistently ranged from 0.67 (N = 9) to 0.40 (N = 71). This suggests that projects with domain-specific characteristics do not generalize well to other domains, making it difficult for models to transfer defect patterns effectively.

*3) Programming Language*: Although all the projects in this study were developed in Java, differences in coding practices and software architecture still affected CPDP performance. For example, JDT (Eclipse JDT Core) demonstrated strong performance when predicting PDE, with an F1 Score of 0.82-0.98 at N=9 to N=15, likely due to shared development environments and similar coding structures. However, as more features were added, performance dropped slightly at N=21 and N=71, suggesting that too many features introduce noise rather than improving model performance. Similarly, LC, which has a different functional purpose compared to Eclipse-based projects, showed mixed results when used as a source dataset. It performed well when predicting ML (0.92 at N=9) but saw its accuracy decline as the feature set increased (dropping to 0.81 at N=71). This indicates that even when projects share the same programming language, differences in their architecture and design can still impact prediction performance.

This study examines the effect of feature set size on machine learning performance in cross-project defect prediction (CPDP) using Random Forest (RF) and Support Vector Machine (SVM) algorithms. The findings reveal how

varying the number of selected features affects model performance, with statistical tests highlighting significant differences across feature set sizes. Furthermore, the practical implications of these findings are noteworthy. In real-world software engineering practices, where projects often differ in structure and available metrics, the ability to fine-tune feature set size provides a cost-effective method for improving defect prediction models. Organizations can enhance early fault detection by optimizing the number of input features, even in the absence of historical defect data specific to their project.

### *F. Impact of Feature Set Size on machine learning Performance*

A key observation from the results is that the number of selected features plays a crucial role in predictive performance. As shown in Table II, for RF, we observe that smaller feature sets (N = 9) tend to yield lower predictive scores across different projects. For instance, in EQ → ML RF scores, the F1_score was 0.67, and in EQ → PDE scores, the F1_score was 0.69; the prediction scores at N=9 were significantly lower than those at N=12 to N=18. However, performance consistently rises as the feature set increases up to N=12, N=15, and N=18, with mean F1 scores stabilizing around 0.80-0.82. The highest scores were observed at N=12 and N=15, suggesting that a moderate number of selected features improves the model's ability to generalize across projects.

For SVM, a similar pattern was observed. Smaller feature sets (N = 9, N = 12) generally yield good results, but the improvement is not as pronounced as in RF. Interestingly, SVM tends to perform more consistently across different feature set sizes, with F1 Scores remaining relatively stable between N = 9 and N = 21. The mean F1_scores stabilize around 0.78-0.79, with slight variation.

### *G. Impact of Large Feature Sets (N=71)*

The results indicate that using all available features (N = 71) does not necessarily yield better results. For RF, performance drops significantly, suggesting that a large number of features introduces noise and redundancy, negatively impacting model generalization. This can also be observed in specific cases, such as EQ → ML, JDT → PDE, and LC → ML, where RF scores an F1_score of 0.40, 0.77, and 0.77, respectively. Notably, N=71 yields some of the lowest scores.

For SVM, the decline in performance at N=71 is not as drastic, but it remains noticeable, with a mean F1 Score of 0.75. Unlike RF, which suffers significantly when too many features are used, SVM is observed to handle larger feature sets with slightly more robustness. However, in specific cases, such as EQ → LC and PDE → LC, the SVM scores a mean F1 Score of 0.78 and 0.78, respectively. Nevertheless, performance still decreases compared to moderate feature sizes.

### *H. Statistical Significance of Performance Differences between Feature Set Sizes*

To better understand the impact of feature set size, we conducted a Wilcoxon signed-rank test to evaluate the significance of differences in performance between feature sizes. The Wilcoxon signed-rank test confirms that N=9 is significantly different from N=12, N=15, and N=18, indicating that minimal feature sets have a negative impact on model performance. N=71 is significantly different from all other feature sizes, indicating that the large features reduce predictive accuracy. It was also observed that no significant differences were found between N=12, N=15, and N=18, suggesting that these feature sets provide similar predictive performance. These findings suggest that moderate feature sets (N = 12–N = 18) are optimal, as they balance predictive power with computational efficiency. The results also highlight a key difference between RF and SVM.

### *I. Practical Implications for Cross-Project Defect Prediction*

These findings provide valuable insights for researchers and practitioners working on CPDP. First, feature selection is crucial, as using all available features (N = 71) reduces performance, emphasizing the need for effective feature selection techniques. Additionally, using minimal feature sets (N=9) results in weaker model performance, likely due to insufficient information for generalization. Secondly, moderate feature sets provide the best balance between predictive power and computational cost. Thirdly, RF exhibits greater performance fluctuations across different feature sizes, indicating the need for careful tuning. While SVM demonstrates a more consistent performance across feature set sizes, making it a suitable choice when feature selection is unclear.

## IV. CONCLUSION

The investigation into the impact of feature set size on the performance of machine learning models in cross-project defect prediction (CPDP) has highlighted the critical role that feature selection plays in developing effective predictive models. This study demonstrates that the size of the feature set significantly impacts machine learning performance in CPDP, with moderate feature sets (N = 12–N = 18) yielding the best results. While RF benefits greatly from proper feature selection, it is more sensitive to large feature sets compared to SVM, which remains more stable but still benefits from optimal feature selection. Despite these differences, both models demonstrate that moderate feature set sizes (N = 12–N = 18) are generally optimal, while minimal (N = 9) and extensive (N = 71) feature sets should be avoided.

The study also confirms that project size, application domain, and programming language have a significant impact on CPDP performance. Models trained on projects of similar sizes and within the same domain achieve higher accuracy, whereas cross-domain and cross-size predictions exhibit greater variability. This work contributes to the CPDP literature by empirically demonstrating that balancing feature set size is critical to achieving accurate and efficient defect prediction across diverse software projects, which have been underexplored in prior studies.

Future research should focus on improving CPDP models by incorporating adaptive feature selection methods, domain adaptation techniques, and transfer learning approaches. These strategies could help enhance prediction accuracy across diverse projects, making CPDP more reliable for real-world applications.

## ACKNOWLEDGMENT

Universiti Putra Malaysia and TetFund Nigeria have supported this work.

## REFERENCES

[1] R. Kumar, "An experimental analysis of the software bug prediction and identifications approaches with different levels of inheritance," in *Proc. 1st Int. Conf. Adv. Electr., Electron. Comput. Intell. (ICAEECI)*, 2023, pp. 1–5, doi: 10.1109/icaeeci58247.2023.10370782.

[2] M. Kakkar, S. Jain, A. Bansal, and P. S. Grover, "Combining data preprocessing methods with imputation techniques for software defect prediction," in *Research Anthology on Recent Trends, Tools, and Implications of Computer Programming*, 2021, pp. 1792–1811, doi:10.4018/978-1-7998-3016-0.ch081.

[3] J. Pachouly, S. Ahirrao, K. Kotecha, G. Selvachandran, and A. Abraham, "A systematic literature review on software defect prediction using artificial intelligence: Datasets, data validation methods, approaches, and tools," *Eng. Appl. Artif. Intell.*, vol. 111, p. 104773, 2022, doi: 10.1016/j.engappai.2022.104773.

[4] Z. M. Zain, S. Sakri, N. H. A. Ismail, and R. M. Parizi, "Software defect prediction harnessing on multi 1-dimensional convolutional neural network structure," *Comput., Mater. Continua*, vol. 71, no. 1, pp. 1522–1546, 2022, doi: 10.32604/cmc.2022.022085.

[5] F. Matloob, T. M. Ghazal, N. Taleb, S. Aftab, M. Ahmad, M. A. Khan, and T. R. Soomro, "Software defect prediction using ensemble learning: A systematic literature review," *IEEE Access*, vol. 9, pp. 98754–98771, 2021, doi: 10.1109/access.2021.3095559.

[6] A. Jalil, R. B. Faiz, S. Alyahya, and M. Maddeh, "Impact of optimal feature selection using hybrid method for a multiclass problem in cross project defect prediction," *Appl. Sci.*, vol. 12, no. 23, p. 12167, 2022, doi: 10.3390/app122312167.

[7] Z. Li, H. Zhang, X. Y. Jing, J. Xie, M. Guo, and J. Ren, "Dssdpp: Data selection and sampling based domain programming predictor for cross-project defect prediction," *IEEE Trans. Softw. Eng.*, vol. 49, no. 4, pp. 1941–1963, 2022, doi: 10.1109/TSE.2022.3204589.

[8] H. Luo, H. Dai, W. Peng, W. Hu, and F. Li, "An empirical study of training data selection methods for ranking-oriented cross-project defect prediction," *Sensors*, vol. 21, no. 22, p. 7535, 2021, doi:10.3390/s21227535.

[9] A. Abdu, Z. Zhai, H. A. Abdo, R. Algabri, and S. Lee, "Graph-based feature learning for cross-project software defect prediction," *Comput., Mater. Continua*, vol. 77, no. 1, pp. 161–180, 2023, doi:10.32604/cmc.2023.043680.

[10] J. Xian, J. Li, Q. Zou, and Y. Xian, "LPDA: Cross-project software defect prediction approach via locality preserving and distribution alignment," *Int. J. Adv. Comput. Sci. Appl.*, vol. 14, no. 12, pp. 892–897, 2023, doi: 10.14569/IJACSA.2023.0141290.

[11] T. Zhang, Y. Yu, X. Mao, Y. Lu, Z. Li, and H. Wang, "FENSE: A feature-based ensemble modeling approach to cross-project just-in-time defect prediction," *Empir. Softw. Eng.*, vol. 27, no. 7, p. 162, 2022, doi: 10.1007/s10664-022-10185-8.

[12] T. Tahir et al., "Early software defects density prediction: Training the international software benchmarking cross projects data using supervised learning," *IEEE Access*, vol. 11, pp. 141965–141986, 2023, doi: 10.1109/access.2023.3339994.

[13] R. Vashisht and S. A. M. Rizvi, "Feature engineering to heterogeneous cross software projects defect prediction: A novel framework," *Arab. J. Sci. Eng.*, vol. 48, no. 2, pp. 2539–2560, 2023, doi: 10.1007/s13369-022-07337-9.

[14] Y. Khatri and S. K. Singh, "An effective feature selection based cross-project defect prediction model for software quality improvement," *Int. J. Syst. Assurance Eng. Manage.*, vol. 14, no. 1, pp. 154–172, 2023, doi: 10.1007/s13198-022-01831-x.

[15] F. Li et al., "The impact of feature selection techniques on effort-aware defect prediction: An empirical study," *IET Softw.*, vol. 17, no. 2, pp. 168–193, 2023, doi: 10.1049/sfw2.12099.

[16] L. P. Manik, "Feature selection in machine learning for cross-project software defect number prediction," in *Proc. 14th Int. Conf. Inf. Commun. Technol. Syst. (ICTS)*, 2023, pp. 18–23, doi:10.1109/icts58770.2023.10330829.

[17] P. Manchala and M. Bisi, "A study on cross-project fault prediction through resampling and feature reduction along with source projects selection," *Autom. Softw. Eng.*, vol. 31, no. 2, p. 67, 2024, doi:10.1007/s10515-024-00465-6.

[18] Y. Z. Bala, P. A. Samat, K. Y. Sharif, and N. Manshor, "Cross-project software defect prediction through multiple learning," *Bull. Electr. Eng. Inform.*, vol. 13, no. 3, pp. 2027–2035, 2024, doi:10.11591/eei.v13i3.5258.

[19] A. Wang et al., "Cross-project software defect prediction using differential perception combined with inheritance federated learning," *Electronics*, vol. 13, no. 24, p. 4893, 2024, doi:10.3390/electronics13244893.

[20] M. D'Ambros, M. Lanza, and R. Robbes, "Evaluating defect prediction approaches: A benchmark and an extensive comparison," *Empir. Softw. Eng.*, vol. 17, pp. 531–577, 2012, doi: 10.1007/s10664-011-9173-9.

[21] S. Stradowski and L. Madeyski, "Industrial applications of software defect prediction using machine learning: A business-driven systematic literature review," *Inf. Softw. Technol.*, vol. 159, p. 107192, 2023, doi: 10.1016/j.infsof.2023.107192.

[22] M. Begum et al., "Software defects identification: Results using machine learning and explainable artificial intelligence techniques," *IEEE Access*, vol. 11, pp. 132750–132765, 2023, doi:10.1109/access.2023.3329051.

[23] Q. Zou, L. Lu, S. Qiu, X. Gu, and Z. Cai, "Correlation feature and instance weights transfer learning for cross project software defect prediction," *IET Softw.*, vol. 15, no. 1, pp. 55–74, 2021, doi:10.1049/sfw2.12012.

[24] M. Mafarja et al., "Classification framework for faulty software using enhanced exploratory whale optimizer-based feature selection scheme and random forest ensemble learning," *Appl. Intell.*, vol. 53, no. 15, pp. 18715–18757, 2023, doi: 10.1007/s10489-022-04427-x.

[25] Y. Z. Bala, P. A. Samat, K. Y. Sharif, and N. Manshor, "Improving cross-project software defect prediction method through transformation and feature selection approach," *IEEE Access*, vol. 11, pp. 2318–2326, 2022, doi: 10.1109/access.2022.3231456.

[26] Z. Xiao et al., "Real-time milling tool breakage monitoring based on multiscale standard deviation diversity entropy," *Int. J. Mech. Sci.*, vol. 240, p. 107929, 2023, doi: 10.1016/j.ijmecsci.2022.107929.

[27] C. C. Aggarwal, "Outlier ensembles," in *Outlier Analysis*, Cham, Switzerland: Springer, 2017, pp. 185–218, doi: 10.1007/978-3-319-47578-3_6.

[28] A. Boukerche, L. Zheng, and O. Alfandi, "Outlier detection: Methods, models, and classification," *ACM Comput. Surv.*, vol. 53, no. 3, pp. 1–37, 2020, doi: 10.1145/3381028.

[29] M. Massoudi, N. K. Jain, and P. Bansal, "Software defect prediction using dimensionality reduction and deep learning," in *Proc. 3rd Int. Conf. Intell. Commun. Technol. Virtual Mobile Netw. (ICICV)*, 2021, pp. 884–893, doi: 10.1109/ICICV50876.2021.9388622.

[30] L. Lavazza and S. Morasca, "Comparing ϕ and the F-measure as performance metrics for software-related classifications," *Empir. Softw. Eng.*, vol. 27, no. 7, p. 185, 2022, doi: 10.1007/s10664-022-10199-2.