Journal of Autonomous Intelligence (2024) Volume 7 Issue 3
doi: 10.32629/jai.v7i3.1097

ORIGINAL RESEARCH ARTICLE

An empirical analysis of feature selection techniques for Software Defect Prediction

Tarunim Sharma[1,*], Aman Jatain[1], Shalini Bhaskar[1], Kavita Pabreja[2]

[1] *Amity School of Engineering and Technology, Gurgaon, Haryana 122413, India*

[2] *Maharaja Surajmal Institute, Janakpuri, New Delhi 110058, India*

*** Corresponding author:** Tarunim Sharma, tarunimsharma@msijanakpuri.com

ABSTRACT

Detecting software defects before they occur is crucial in software engineering as it impacts software system quality and reliability. Previous studies on predicting software defects have typically employed software features, such as code size, complexity, coupling, cohesion, inheritance, and other software metrics., to forecast whether a code file or commit is prone to defects in the future. However, it is advantageous to restrict the number of features employed in a defect prediction model to avoid the challenges associated with multicollinearity and the "curse of dimensionality" and to simplify the data analysis process. By using a reduced number of features, the defect prediction model can concentrate on the most significant variables and improve its accuracy. This research paper investigates the impact of eight feature selection methods on the accuracy and stability of six supervised learning models. This study is novel as it is based on exhaustive experimentation of each of the eight feature selection techniques with each of the six supervised learning models. Two notable findings have been obtained. First, we discovered that the association and coherence-based techniques have demonstrated the highest level of accuracy when it comes to defect prediction. The models that utilized these selected features outperformed those using the original features. Second, the feature selection techniques, namely Correlation feature selection, Recursive feature elimination, and Ridge feature selection when combined with the Support vector machine and Decision tree classifier, consistently selected low-variance features across multiple supervised defect prediction models. When combined with different classifiers, these techniques achieved exceptional performance on the publicly available NASA datasets CM1 and PC2. The findings revealed a remarkable accuracy rate of over 85% for CM1 and 95% for PC2, accompanied by precision, recall, and f-measure values exceeding 95%. These exceptional results indicate the achievement of the highest level of performance in the evaluation.

Keywords: software defect prediction; machine learning; feature selection; supervised models

ARTICLE INFO

Received: 7 August 2023
Accepted: 7 September 2023
Available online: 12 January 2024

1. Introduction

A recent report by The Standish Group revealed that only 16.2% of IT projects delivered all promised functionalities, and 31.1% of IT expenditures were classified as failures[1]. As stated in a study published in the Proceedings of the "International Conference on Machine Learning and Cybernetics", the effectiveness of software defect prediction models varied between 70% and 95% in terms of success rates. Given the time constraints faced by software developers, they must make the most of their testing time. Hence, developers face the need to prioritize their testing efforts effectively. Defect prediction models offer a valuable solution by guiding developers to allocate their limited testing resources toward the components that are more likely to contain defects. This strategic allocation optimizes their testing endeavours, leading to improved

efficiency in software development. Feature engineering serves as a critical aspect of machine learning, encompassing the extraction of pertinent attributes and the reduction of feature dimensions to mitigate overfitting. By meticulously selecting and refining relevant data patterns, this process aims to accentuate the significance of the underlying information, leading to more robust and accurate models[2]. It is used to identify the most important subset of characteristics from the initial feature set, offering benefits such as reducing the number of model parameters, shortening training time, improving generalization, and avoiding dimensionality issues.

This research aims to investigate the efficacy of diverse learning models by implementing a range of feature selection techniques. By analyzing their impact on defect prediction accuracy within a dataset, this study endeavours to shed light on the most effective approaches for enhancing the precision of defect prediction. With the judicious selection of feature selection techniques and learning models, the defects can be predicted in advance help to minimize the loss for software development organizations. The Cutting-edge technologies in computer science, such as Artificial Intelligence and machine learning, offer potential solutions for anticipating defect-prone modules before the software development process reaches the testing stage. Two main methods of feature selection are Supervised and Unsupervised feature selection as shown in **Figure 1**. In supervised feature selection, the selection of features is based on the output label class, and it identifies variables that can enhance the model's effectiveness by considering the target variables. On the other hand, unsupervised feature selection is based on selecting features and is used for unlabelled data.

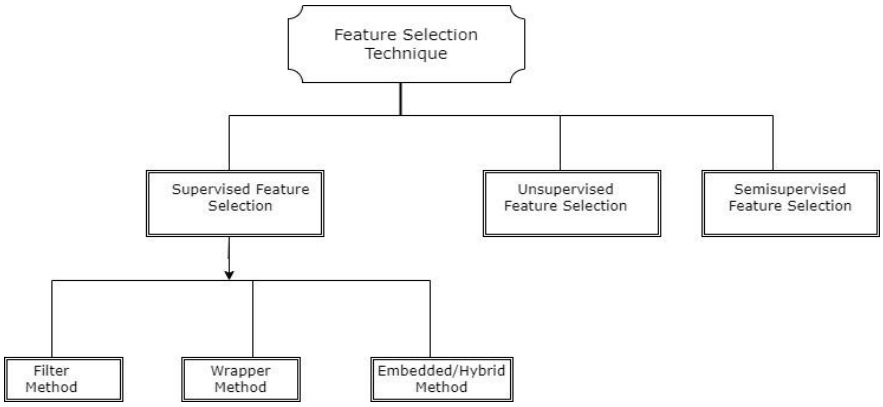

Figure 1. Feature Selection types.

The paper is structured into several sections. In the second section, a concise overview of the related background is presented. The third section delves into the methodology utilized to predict defects in software. Finally, the fourth section of the article includes a comparison and discussion of the outcomes obtained using various feature selection techniques and classifiers.

2. Related work

Software defect prediction is a critical task in software engineering aimed at identifying potential defects in software systems early in the development process. Various machine learning-based techniques have been proposed for software defect prediction, but the performance of these techniques largely depends on the selection of relevant features. The role of feature selection techniques is paramount in elevating the accuracy and efficiency of defect prediction models, marking a crucial factor in their continual improvement and effectiveness.

This literature review examines different feature selection methods employed in software defect prediction and evaluates their efficacy in enhancing the performance of the predictive models. This review aims to provide an overview of the latest research findings in this field.

According to Kondo et al.[3], for supervised models used in defect prediction, the use of either

"Correlation-Based (CFS) or Consistency-Based (ConFS)" approaches for identifying feature subsets is recommended. These approaches have been demonstrated to be more effective than both the original models and feature reduction techniques. Employing these methodologies in conjunction with a supervised defect prediction algorithm can significantly boost the model's efficiency and streamline the selection process for a precise defect prediction model. In their study, Dao et al.[4] tackled the challenges posed by high-dimensional features, including issues such as computational inefficiency, noise, and over-fitting. To address these problems, the authors utilized a two-stage feature selection approach that enabled the reduction of feature dimensionality while ensuring optimal feature selection for maximum accuracy. In a study, Rostami et al.[5] thoroughly examined and classified various approaches for feature selection using a comparative approach. This article centers around the utilization of Swarm Intelligence techniques for feature selection. Specifically, it highlights the effectiveness of wrapper and filter approaches such as "Particle Swarm Optimization, Ant Colony Optimization, Artificial Bee Colony, Differential Evolution, Firefly Algorithm, Bat Algorithm, and Cat Swarm Optimization". These methods have demonstrated promising results in the field of feature selection. The investigation evaluates the benefits and drawbacks of intelligence-based methods for feature selection and scrutinizes the factors that impact their efficacy. In their research, Chen et al.[6] propose a novel approach for enhancing feature selection accuracy in defect prediction models. Their methodology leverages a customized deep learning algorithm, boosting algorithms, and the superposition criterion, all integrated within the Nested-Stacking framework. Through comprehensive testing encompassing feature engineering and baseline model selection, the framework successfully undergoes both within-project and cross-project validation, validating its effectiveness. The results demonstrate the model's exceptional capability in accurately forecasting software defects, providing robust decision-making support for software testing. Furthermore, the incorporation of a heterogeneous approach within the framework enhances its adaptability and generalization potential, establishing it as a superior alternative to existing mainstream software defect prediction techniques. Qu et. al.[7] found that multiple dimension-reduction techniques can improve performance in which Correlation Feature Selection was the most effective individual feature reduction method and had the least time complexity and Random Forest was the superior classifier. Umbarkar and Shukla[8] proposes a technique to improve intrusion detection system accuracy by reducing its feature set using Information Gain, Gain Ratio, and Correlation methods. The best features are selected using a heuristic approach. The technique is evaluated using the C 4.5 decision tree algorithm, and according to the findings, utilizing only 15 features resulted in a higher accuracy rate of 92.65% as compared to utilizing the entire dataset consisting of 41 features. The correlation-based feature reduction approach was found to be the most efficient method for achieving this outcome.

Zebari et.al.[9] concluded quoting high-dimensional data significantly impacts computational time, learning algorithms, model accuracy, and computer resources. As per their research, Support Vector Machine (SVM) and K Nearest Neighbor (KNN) are widely utilized analyzers, with SVM exhibiting the highest level of accuracy. In terms of feature extraction techniques, Convolutional Neural Networks (CNN) and Deep Neural Networks (DNN) are noteworthy methods. In a study conducted by Spencer et.al.[10] researchers revealed that the most precise model was produced by using the Chi-squared technique for selecting the best features in combination with the Bayes Net classification models. Pabreja et al.[11] have used the recursive feature elimination technique to extract the significant influential features that activate stress in the lives of working professionals. Another study[12] has extracted the five most influential determinants of the perceived Personal Wellbeing Index of college students during the COVID-19 time period have been identified by the application of feature importance functions of the machine learning algorithm. These findings highlight the potential benefits of utilizing advanced techniques in the development of predictive models for heart disease, which could ultimately lead to better diagnosis and treatment of this common ailment.

In their research, Marian et.al.[13] introduced a novel approach for predicting software defects using

"fuzzy decision trees". They assert that this method demonstrates superior performance over the standard decision tree, as indicated by higher AUC scores. While the concept of "fuzzy decision trees" is intriguing, the study lacks sufficient details regarding the implementation and evaluation processes, such as the criteria for selecting input features and the methodology for identifying the best model. Furthermore, the experiments were conducted on a limited dataset, encompassing only two software projects, which raises concerns about the generalizability of their proposed method. To achieve an efficient representation of the dataset and streamline the learning process of predictive models, using various statistical measures for selecting attributes is a crucial stage adopted by Elavarasan et.al.[14]. In the study, a hybrid feature extraction process is proposed for predicting crop yield using real-time data. The method combines Correlation-based Feature Selection and Random Forest -Recursive Feature Elimination to eliminate unnecessary features and Acquire a feature collection that exhibits exceptional predictive precision. The hybrid method outperforms other inbuilt feature selector methods with improved prediction accuracy and fewer error measures. The wrapper method for selecting attributes by eliminating features recursively using Random Forest is particularly noteworthy for its competitive results without requiring fine-tuning.

Two studies by Aggrawal and Pal[15] and Gárate-Escamila et al.[16] aimed to enhance the accuracy of predicting mortality and heart disease in patients. Aggrawal and Pal used a sequential feature selection technique with various learning models, achieving 86.67% accuracy when paired with the random forest classifier. Gárate-Escamila proposed a method combining chi-square and principal components analysis to reduce dimensions and improve accuracy. Their findings contribute to the advancement of machine learning methods for medical applications, offering valuable insights into the identification and prediction of heart disease and mortality occurrences and further emphasizing the importance of feature selection and dimensionality reduction in accurately predicting heart disease. Author research was conducted to evaluate different methods based on their ability to strike a balance between success index and classification accuracy by Bolón-Canedo et al.[17]. Their analysis revealed that ReliefF is the most suitable method regardless of the dataset, due to its minimal computational cost as a filter. The authors suggest the utilization of filter methods, particularly ReliefF, has robust generalization capabilities, and performs faster than embedded and wrapper techniques, which is not dependent on the induction algorithm. In their research, Liu et al.[18] introduced a novel supervised attribute selection technique known as Discriminative Low-Rank Preserving Projection (DLRPP). This approach leverages an adaptive graph to capture both regional and global design patterns in the data, thereby enhancing the accuracy of predictive models. By extracting relevant attributes and integrating them into machine learning algorithms, DLRPP significantly improves the performance and efficacy of various applications such as image recognition, speech processing, and natural language processing. Notably, this method incorporates a discriminative constraint term that further enhances its classification and recognition capabilities. Through studies conducted on public image databases, DLRPP has demonstrated its effectiveness as a feature selection algorithm, consistently outperforming other similar approaches. Yuan et.al.[19] investigated the effectiveness of integrated models for selecting stocks and found that Random Forest (RF) was the most effective approach for both selecting the subsets and stock price trend forecasting. The study also suggested that Random Forest is a powerful tool for making long-term investment decisions in the stock market.

After perusing several papers, it was discovered that numerous studies at a global level have employed diverse feature engineering and machine learning algorithms. However, none of these studies concentrated on a thorough examination and comparison of numerous prevalent feature selection techniques to determine the optimal one for deployment across a range of machine learning classifiers. This paper takes an analytical approach and applies eight feature selection techniques on two standardized software defect prediction datasets, namely CM1, and PC2. The top-performing technique under various categories are identified and subsequently tested with effective and widespread classifiers to arrive at the optimal solution for predicting

defects.

3. Experimental framework

To enhance the precision of software defect prediction models, a systematic investigation process has been executed using the algorithmic approach shown in **Figure 2**. The framework comprises several key components, such as the "dataset used, data pre-processing, feature selection methods, data splitting, model training using classifiers, and model evaluation". In the beginning, the dataset pre-processing step has been applied which is a crucial step to ensure an accurate representation of data quality. A comprehensive data refinement process was carried out to uphold data integrity, enhance feature quality, and eliminate any missing values. This meticulous preparation paved the way for a streamlined and effective data pre-processing phase, enabling insightful analysis and accurate results. Subsequently, various feature selection techniques such as Filter, Wrapper, and Embedded are employed to effectively identify significant features, eliminate less relevant features, and enhance the overall learning performance. The detailed experiment approach classification modelling to approximate the relationship between input and output variables is based on exhaustive techniques. For the purpose of feature selection techniques, the study utilizes six supervised learning algorithms, which include "Decision Trees, Random Forests, Support Vector Machines", etc. The purpose of this model is to effectively analyze the data and recognize trends present in the software defect prediction dataset. The following is an outline and explanation of the steps taken in the research methodology that was followed.

1) Prepare the dataset by addressing any missing or incorrect data, then divide it into separate sets for training and testing purposes. In the data splitting stage, the dataset for defect prediction is partitioned into a training set of 80% and a testing set of 20%. The models were trained using the training set and their performance was evaluated using the testing set.
2) A diverse array of feature selection techniques, such as "Chi-square, ANOVA, Correlation Feature Selection, Forward/Backward Feature Selection, Recursive Feature Elimination, Lasso, and Ridge regression" were employed to pinpoint the most relevant subset of features with the utmost significance.
3) Utilize an array of classifiers, including but not limited to "Random Forest, Decision Tree, Logistic Regression, SVM", and their counterparts, to train on the selected features within the training set.
4) Assess the proficiency of each model on the test set by examining crucial evaluation metrics like Precision, Recall, and F1 score.
5) Conduct a comprehensive analysis of the various models and their performance, ultimately identifying the optimal model with the most effective set of features.

3.1. Dataset description

The dataset used for analysis comprises two subsets: one with 498 records and 21 attributes obtained from the CM1 NASA machine repository, and another with 745 records and 36 attributes from the PC2 repository. These datasets contain instances that indicate both the absence and presence of software defects. To conduct experiments, the Jupyter Integrated Development Environment for Python was employed to implement various feature reduction algorithms on the CM1 and PC2 datasets. These datasets are widely recognized as benchmark datasets, providing a solid foundation for constructing an analytical machine-learning model. Both the CM1 and PC2 datasets provide researchers with standardized and publicly available datasets to develop, evaluate, and compare software defect prediction models. These datasets enable researchers to study the impact of different attributes and techniques on predicting the presence of defects in software modules, ultimately aiming to improve software quality and reliability. Both datasets contain various metrics and attributes associated with the software modules, such as Lines of code, Halstead complexity measures, McCabe cyclomatic complexity, and more. It is utilized for benchmarking and comparing different defect prediction algorithms and techniques.

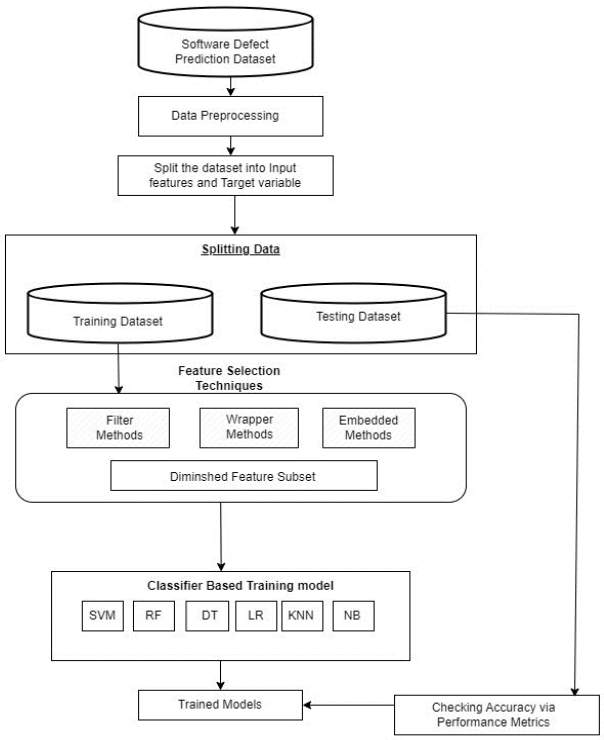

Figure 2. Algorithmic approach for software defect prediction.

3.2. Attribute filtering techniques

Reducing the number of features in a dataset is a crucial step in data engineering and analysis, as it helps overcome the problem of high dimensionality and enhances model efficiency. Feature reduction can significantly improve the accuracy of machine learning models, particularly in situations where class distributions are imbalanced, by selecting the most relevant attributes. The primary aim of feature selection techniques is to maximize the predictive power of models while minimizing errors, leading to precise and high-performing frameworks. Machine learning employs a variety of feature selection techniques, each designed to cater to the specific needs of the problem and dataset. These methods encompass numerous cutting-edge approaches that are widely utilized in the field of data analysis and machine learning. The efficacy of all three methods in handling a significant number of features is notable. However, their performance relies on selecting appropriate fitness functions, updating rules, and other associated parameters. Previous research has identified various effective techniques for feature selection, including filter, wrapper, and embedded methods, each with its own benefits and drawbacks discussed in **Table 1** below.

Table 1. Benefits and drawbacks associated with the three primary categories of feature selection techniques.

Feature selection technique	Pros	Cons	Examples
Filter	• Streamlined computationally • Fast processing • Lower overfitting risk • Improved ability to generalize new data • Simple to scale for use with large datasets	• May have lower precision than other methods, resulting in less accurate predictions • There is a risk that important features may be missed or not identified by the technique • This method lacks the capability to handle multicollinearity, a situation where strong correlations exist between features, posing a risk of yielding unreliable model outcomes.	"Pearson Correlation, Chi-Square, ANOVA, LDA", etc.

Table 1. (*Continued*).

Feature selection technique	Pros	Cons	Examples
Wrapper	• Attaining heightened precision yields greater accuracy in predictive outcomes. • Interacts with the classifier algorithm to select features, improving the model's effectiveness • Considers feature dependencies and how they interact with each other • Better generalization compared to the filter approach, leading to more reliable model results • Conducts a more comprehensive search of the feature set space, potentially identifying better combinations of features for the model.	• Computationally expensive due to the recursive process involved • Slow processing time, potentially taking longer to run than other feature selection techniques • There is a risk of overfitting, which occurs when the model becomes too complex and is not able to generalize well to new data.	"Forward Selection, Backward Elimination, Recursion Feature elimination" etc.
Embedded	• Incurs minimal additional expenses since it is integrated into the classification model training procedure. • Less computationally demanding than the wrapper technique, potentially requiring less processing time • Works with a classification model to select features, improving the effectiveness of the model • Lower risk of overfitting than the wrapper method. Improved generalization error, especially with larger datasets potentially outperforming the filter approach.	• Dependent on the learning algorithm used in the classification model training process • Identifying a small set of features may be challenging or problematic, potentially leading to less effective models.	"Lasso, Ridge, Elastic Net", etc.

From the above table discussion, it is concluded that the goal of feature selection is to reduce the number of features to a subset that is most informative for the problem at hand. This can help to improve model accuracy, reduce overfitting, and enhance the model's interpretability. Different techniques are discussed for feature selection, along with their own set of advantages and limitations. Among three techniques the first one is Filter techniques[20] which are computationally efficient and fast, making them suitable for large datasets as depicted in **Figure 3**. They work by evaluating each feature's relevance independently of the others, usually based on statistical tests, and selecting the top-ranked features. However, they may not capture the most relevant features of the problem, leading to less accurate predictions.

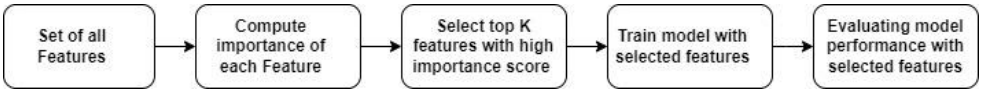

Figure 3. Filter method.

Another one is the Wrapper technique[21] which evaluates the feature subset's performance by interacting with the learning algorithm used in the model, which can result in better generalization and more reliable model results as shown in **Figure 4**. These methods search the entire feature space and can consider feature dependencies, leading to higher precision. However, they are computationally expensive, potentially taking longer to run than other feature selection techniques, and may overfit the data.

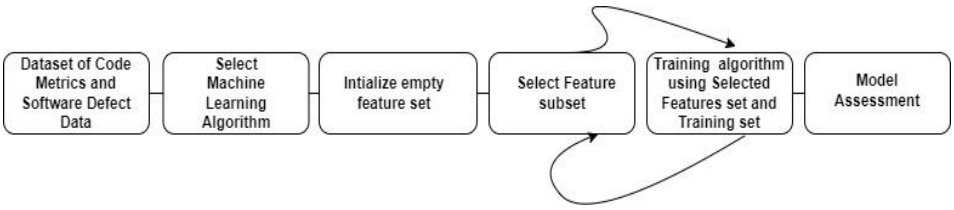

Figure 4. Wrapper method.

The next one is Embedded methods that combine the benefits of both filter and wrapper techniques. Embedded methods integrate feature selection algorithms into machine learning training, reducing computational expenses and improving model accuracy[22] as presented in **Figure 5**. Regularization techniques, including L1 and L2 regularization as well as Elastic Nets, belong to the category of embedded methods in machine learning. These methods introduce penalty terms to the coefficients during model training, aiming to prevent overfitting and potentially eliminate less important features. By adding these penalties, the models are encouraged to find a balance between accuracy and complexity, leading to more generalized and robust results. Another embedded technique, Random Forest Importance, assesses the importance of features based on their usage within a random forest algorithm. By analysing the contribution of each feature in the ensemble of decision trees, Random Forest Importance can provide valuable insights into the factors that play a significant role in the prediction task. This technique not only aids in reducing dimensionality but also improves model accuracy and interpretability, facilitating a better understanding of the factors influencing software defects or any other target variable of interest.

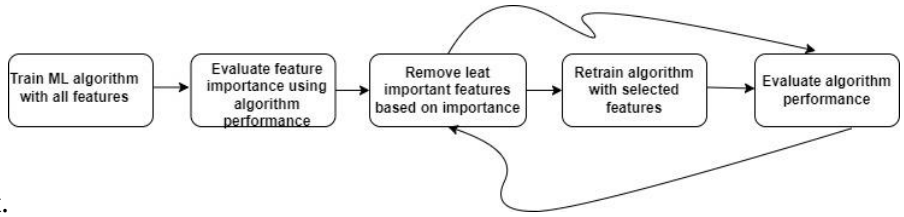

Figure 5. Embedded method.

Hybrid methods combine two or more feature selection techniques, leveraging their strengths to improve accuracy and efficiency as illustrated in **Figure 6**. Popular approaches include combining a filter method with a wrapper method or an embedded method with a filter method. Ensemble methods apply multiple feature selection methods and combine their outputs to create a final set of selected features. These methods identify important features and eliminate redundancies, but may be computationally expensive. Examples include" Genetic algorithms, Ant Colony Optimization, and Particle Swarm Optimization". These methods integrate feature selection into the classification model training process, minimizing additional expenses and reducing the risk of overfitting. They can identify a small subset of features that can result in improved generalization error performance, potentially outperforming the filter approach with larger datasets. However, identifying this subset of features can be challenging, and selecting a small set of features may result in less effective models.

Figure 6. Hybrid method.

Choosing the right feature selection method requires careful consideration of various factors such as the dataset's characteristics and the specific problem being tackled. Different methods, including filter, wrapper, and hybrid/embedded approaches, have their advantages and limitations. For instance, filter methods are computationally efficient and can be applied to large datasets, whereas wrapper methods consider feature

dependencies and can lead to more reliable model results. Embedded techniques, on the other hand, can strike a balance between reducing additional expenses and minimizing overfitting risks. To choose the most appropriate feature selection method, it's crucial to weigh the pros and cons of each technique and how they fit the problem at hand. Factors such as the number of features, their relationships, and their relevance to the target variable should be considered. By selecting an appropriate technique, one can ensure a reduction in the number of features while enhancing both model accuracy and interpretability. Ultimately, choosing the right feature selection method can help unlock the full potential of machine learning algorithms and pave the way for more effective and efficient data-driven decision-making.

4. Design of the classifier model

Defect prediction is crucial to software quality assurance and testing, particularly in software development. One effective technique for this is classification-based predictive modeling, which involves creating a model using a labeled dataset of past code changes and input features such as code complexity and developer experience. This paper explores the utilization of diverse algorithms for this purpose, encompassing "Decision Trees, Random Forest, Support Vector Machines (SVM), Gaussian Naive Bayes (GNB), Logistic Regression, and K-nearest neighbors (KNN)". SVM can identify a hyperplane that separates defective and non-defective modules in the software and can handle high-dimensional and imbalanced data. However, its performance may depend on the chosen hyperparameters, and it can be computationally expensive for large datasets. Decision Tree Classifier is easy to interpret and can handle both categorical and continuous data. Random Forest creates multiple decision trees and can handle imbalanced data and automatically handle missing data and outliers, but it requires more computational resources and may be less interpretable than a single decision tree. Gaussian Naïve Bayes is a fast and efficient probabilistic algorithm that can handle missing data and outliers and is less prone to overfitting. Logistic Regression can handle imbalanced data and provide an interpretation of model coefficients, but it may not perform well for nonlinear or complex relationships between input features and output labels. Both Gaussian Naïve Bayes and Logistic regressions are effective tools for handling high-dimensional data and numerous features in software defect prediction. KNN is a simple and interpretable algorithm that calculates the distance between a new instance and all instances in the training set, selects the k-nearest instances, and classifies the new instance based on the majority class of those k neighbors. It can be trained on a dataset of software modules with known defect status and used to predict the status of new modules based on features such as code complexity, code churn, and developer experience. However, it can be computationally expensive for large datasets and may require careful tuning of hyperparameters, and the quality of the predictions can be affected by the choice of features and the quality of the training data[23].

5. Results and discussion

Various feature selection techniques along with numerous classification models have been experimented with the CM1 and PC2 datasets and quite convincing results have been obtained. **Table 2** shows the defect prediction accuracy achieved by 8 different feature selection techniques in combination with 6 machine learning classifiers that have been experimented on CM1 and PC2 respectively. We have used three filter methods viz "Anova, Chi2, Correlation feature selection, three wrapper-based methods used were "Forward Feature Selection, Backward Feature Selection, and Recursive Feature Elimination" techniques and two Embedded feature selectors "Lasso and Ridge" with different learning algorithms.

In the study, the effectiveness of the feature selection techniques was assessed using various evaluation metrics, including Precision, F1 score, Accuracy, and Recall. These metrics comprehensively analyzed how well the feature selection techniques performed in addressing classification problems. By considering these assessment metrics, researchers were able to gauge the efficiency of the feature selection techniques in the

study.

In this Accuracy is a metric that measures the overall correctness of the model's predictions. It determines the proportion of correctly classified instances, both defective and non-defective, out of the total instances in the dataset.

$$\text{Accuracy} = (TP+TN)/(TP+TN+FP+FN)$$

Precision is a metric that measures the proportion of correctly identified defective instances out of all instances predicted as defective by the model. It focuses on the accuracy of positive predictions made by the model. Precision is given by

$$\text{Precision} = TP/(TP+FP)$$

In software defect prediction, Recall refers to the model's ability to accurately detect actual defective instances. It is a crucial metric to ensure that software defects are identified effectively, reducing the risk of releasing faulty software to users and enhancing overall software quality. Achieving a high recall is essential for proactive defect resolution, but it should be balanced with other performance metrics depending on specific project requirements.

$$\text{Recall} = TP/(TP+FN)$$

The F-score, a comprehensive evaluation metric, combines precision and recall to assess the test's overall accuracy. By utilizing these metrics and their corresponding formulas, researchers can evaluate the model's effectiveness.

$$\text{F-measure} = (2 \times \text{Precision} \times \text{Recall})/(\text{Precision} + \text{Recall})$$

Overall, the results of the experiment provide insights into the effectiveness of different feature selection methods and their impact on defect prediction accuracy. According to the analysis, the models built using the chosen feature subsets outperform the original dataset features and their implementation substantially enhanced the precision of the classifiers.

Table 2. Classification accuracies of reduced feature subsets using Filter-based, Wrapper Based, and Embedded methods with LR, RF, DT, SVM, GNB, and KNN (CHI: Chi-square, ANOVA: ANOVA method, CFS: Correlation Feature Selection, FFS: Forward Feature Selection, BFS: Backward Feature Selection, RFE: Recursive Feature Elimination, Lasso, Ridge) on CM1 and PC2 dataset.

	Feature Selection Techniques															
	Filter method						**Wrapper method**						**Embedded method**			
	CHI 2		**Anova**		**CFS**		**FFS**		**BFS**		**RFE**		**Lasso**		**Ridge**	
6 Classification Techniques	**CM1**	**PC2**	**CM1**	**PC2**	**CM1**	**PC2**	**CM1**	**PC2**	**CM1**	**PC2**	**CM1**	**PC2**	**CM1**	**PC2**	**CM1**	**PC2**
LR	88	97	88	97	88	97	88	97	87	96	87	95	88	96	88	97
RF	88	97	87	97	87	97	87	97	87	97	85	97	87	97	88	97
DT	80	94	84	96	83	96	87	97	86	97	83	95	81	97	85	95
SVM	88	97	88	97	88	97	88	97	88	98	87	97	88	97	88	97
GNB	87	96	87	95	87	95	86	97	86	95	87	94	88	93	86	95
KNN	88	97	87	97	87	97	88	97	86	97	85	97	88	97	88	97

From **Table 2** it is evident that Wrapper methods when used with the SVM machine learning model have resulted in 88% and 97% accuracy for the prediction of software defects on CM1 and PC2 datasets respectively. A graphical comparison of the results of each prediction model accuracy on the CM1 and PC2 datasets is presented in **Figure 7a** and **Figure 7b** visualizing the best accuracy performance of each feature selection technique. The superiority of specific feature selection techniques became apparent when compared to other combinations with machine learning algorithms. Notably, the Correlation feature selection method from the Filter category, Recursive Feature elimination from the Wrapper category, and Ridge feature

selection techniques categorized under the Embedded method demonstrated exceptional performance. These techniques surpassed alternative combinations in terms of their ability to select relevant features for the machine learning models.

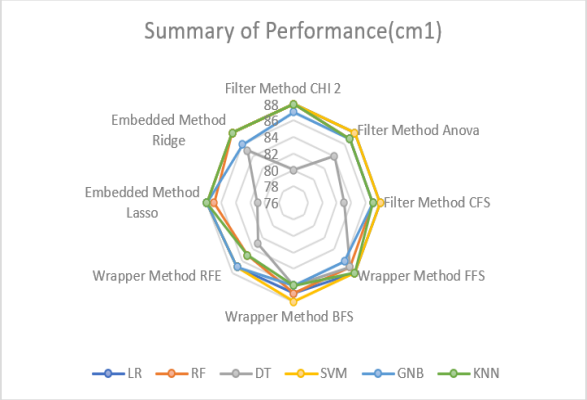

Figure 7 (a). Comparison of accuracy on CM1 dataset.

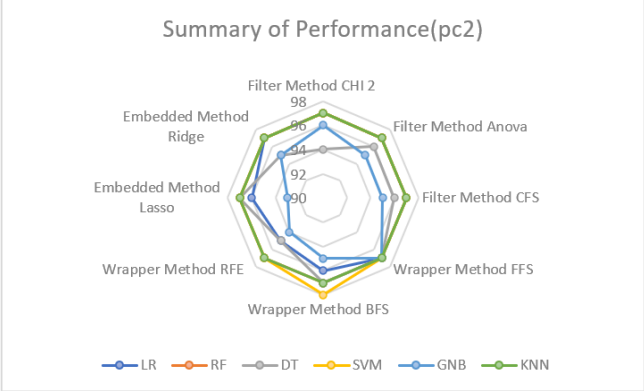

Figure 7 (b). Comparison of accuracy on PC2 dataset.

To measure the effectiveness of classification techniques, a variety of other performance metrics such as "Precision, Recall, and F1 score" are also utilized. The study focuses on six different classification techniques and in addition employed eight distinct feature selection techniques on two datasets shown in **Tables 3–5**. The outcomes of the classification approach are depicted in **Figure 8(a–c)**, showcasing the performance metrics—Precision, Recall, and F1-Score—of each prediction model applied to the CM1 dataset. These models were evaluated using the Filter method, Wrapper method and Embedded method and the results of various prediction models is shown in **Figure 8 (a–c)** respectively. Figure 9 presents an analysis of the classification approach, emphasizing the Precision, Recall, and F1-Score metrics for individual prediction models applied to the PC2 dataset. This assessment utilized techniques such as the Filter Method depicted in Fig 9(a), Wrapper Method illustrated in Fig 9(b), and Embedded Method shown in Fig 9(c).

Table 3. Performance measure of best-reduced feature subsets using the Filter, Wrapper, and Embedded methods using a variety of performance metrics such as Precision, Recall, and F1 score on the CM1 dataset.

CM1																		
Feature Selection Techniques	**Chi2**			**ANOVA**			**CFS**			**FFS**			**BFS**			**RFE**		
Classifier	**P**	**R**	**F1-score**	**P**	**R**	**F1-score**	**P**	**R**	**F1-score**	**P**	**R**	**F1-score**	**P**	**R**	**F1-score**	**P**	**R**	**F1-score**
LR	0.77	0.88	0.82	0.77	0.88	0.82	0.95	0.97	0.96	0.77	0.88	0.82	0.77	0.87	0.82	0.77	0.87	0.82
RF	0.85	0.88	0.85	0.77	0.87	0.82	0.95	0.97	0.96	0.82	0.87	0.83	0.77	0.87	0.82	0.77	0.85	0.81
DT	0.83	0.80	0.81	0.81	0.84	0.83	0.95	0.96	0.95	0.77	0.87	0.82	0.81	0.86	0.83	0.81	0.83	0.82
SVM	0.77	0.88	0.82	0.77	0.88	0.82	0.95	0.97	0.96	0.77	0.87	0.82	0.77	0.88	0.82	0.84	0.87	0.85
NB	0.84	0.87	0.85	0.84	0.87	0.85	0.96	0.95	0.96	0.77	0.86	0.81	0.77	0.86	0.81	0.84	0.87	0.85
KNN	0.77	0.88	0.82	0.82	0.87	0.83	0.95	0.97	0.96	0.77	0.88	0.82	0.77	0.86	0.81	0.77	0.85	0.81

Feature Selection Techniques	**LASSO**			**RIDGE**		
Classifier	**P**	**R**	**F1-score**	**P**	**R**	**F1-score**
LR	0.77	0.88	0.82	0.77	0.88	0.82
RF	0.77	0.87	0.82	0.84	0.88	0.84
DT	0.79	0.81	0.80	0.83	0.85	0.84
SVM	0.77	0.88	0.82	0.77	0.88	0.82
NB	0.85	0.88	0.85	0.83	0.86	0.84
KNN	0.77	0.88	0.82	0.77	0.88	0.82

Table 4. Performance measure of best-reduced feature subsets using the Filter, Wrapper, and Embedded methods using a variety of performance metrics such as Precision, Recall, and F1 score on the PC2 dataset.

PC2																		
Feature Selection Techniques	**Chi2**			**ANOVA**			**CFS**			**FFS**			**BFS**			**RFE**		
Classifier	**P**	**R**	**F1-score**	**P**	**R**	**F1-score**	**P**	**R**	**F1-score**	**P**	**R**	**F1-score**	**P**	**R**	**F1-score**	**P**	**R**	**F1-score**
LR	0.95	0.97	0.96	0.95	0.97	0.96	0.95	0.97	0.96	0.95	0.97	0.96	0.95	0.96	0.95	0.98	0.98	0.97
RF	0.95	0.97	0.96	0.95	0.97	0.96	0.95	0.97	0.96	0.95	0.97	0.96	0.95	0.97	0.96	0.95	0.97	0.96
DT	0.95	0.94	0.94	0.95	0.96	0.95	0.95	0.96	0.95	0.95	0.97	0.96	0.95	0.97	0.96	0.95	0.95	0.95
SVM	0.95	0.97	0.96	0.95	0.97	0.96	0.95	0.97	0.96	0.97	0.97	0.96	0.95	0.97	0.96	0.95	0.97	0.96
NB	0.96	0.96	0.96	0.96	0.95	0.96	0.96	0.95	0.96	0.95	0.97	0.96	0.95	0.97	0.96	0.96	0.94	0.95
KNN	0.95	0.97	0.96	0.95	0.97	0.96	0.95	0.97	0.96	0.95	0.97	0.96	0.95	0.97	0.96	0.95	0.97	0.96

Table 5. Performance measure of best-reduced feature subsets using the Filter, Wrapper, and Embedded methods using a variety of performance metrics such as Precision, Recall, and F1 score on the PC2 dataset.

Feature Selection Techniques	**LASSO**			**RIDGE**		
Classifier	**P**	**R**	**F1-score**	**P**	**R**	**F1-score**
LR	0.95	0.96	0.95	0.95	0.97	0.96
RF	0.95	0.97	0.96	0.95	0.97	0.96
DT	0.96	0.97	0.96	0.95	0.95	0.95
SVM	0.95	0.97	0.96	0.95	0.97	0.96
NB	0.96	0.93	0.94	0.96	0.95	0.95
KNN	0.95	0.97	0.96	0.95	0.97	0.96

The findings demonstrate that all models created using diverse feature selection techniques outperform the original feature subsets of the dataset. The initial feature set attained a maximum accuracy of 88%, a precision of 77%, a recall of 88%, and an F-measure of 82% when utilized with the KNN classifier on the CM1 dataset. However, with the application of various feature selection approaches, the accuracy of the classifiers noticeably improved. The highest accuracy achieved by each feature selection technique is consolidated and presented in **Tables 3–5**. Upon applying feature selection techniques to the CM1 dataset, a distinct set of significant features was selected for analysis. These chosen features offer valuable insights into the codebase's characteristics and complexity. In the CM1 dataset, the selected features include Lines of Code, Variables, Branches, Lines of Comment, Unique Operands, Total Operands, Comments, and Comments-to-Code ratio. Similarly, in the PC2 dataset, the selected features encompass Lines of Code Total, Number of Operands, Number of Lines, Lines Blank, Condition Count, Call Pairs, Lines of Code Executable, Maintenance Severity, Node count, and Normalized Cyclomatic Complexity. These meticulously chosen features aim to capture crucial aspects of the codebase, such as its size, complexity, documentation, control flow, and potential maintenance requirements. Notably, the feature subset obtained through the Correlation Feature selection, Recursive feature elimination, Lasso and Ridge Feature selection techniques yielded the highest classification accuracy of 88%, a precision of 95%, a sensitivity of 97%, and an f-measure of 96% when utilized with SVM classifier on CM1 dataset and with PC2 highest classification accuracy of 97%, a precision of 98%, a sensitivity of 98%, and an f-measure of 97% when utilized with the DT classifier. According to **Tables 3–5**, all models created by combining feature selection techniques achieved more than 85% accuracy.

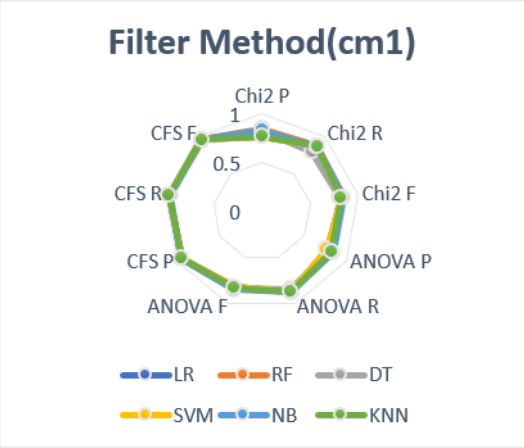

Figure 8(a). Results of each prediction model using performance metrics Precision, Recall, and F1-Score on the CM1 dataset using Filter Method

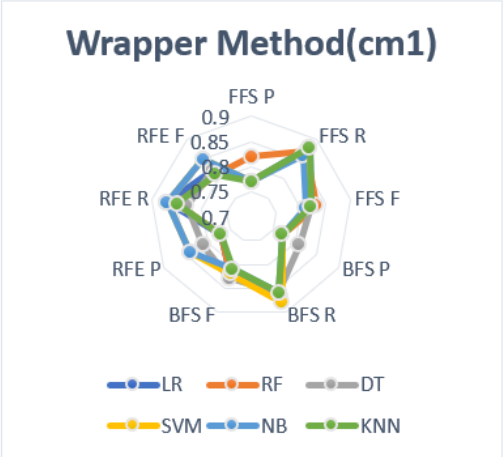

Figure 8(b). Results of each prediction model using performance metrics Precision, Recall, and F1-Score on the CM1 dataset using Wrapper Method

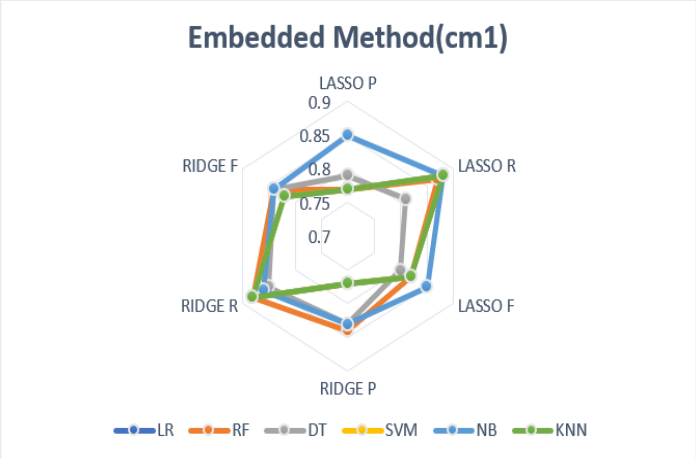

Figure 8(c). Results of each prediction model using performance metrics Precision, Recall, and F1-Score on the CM1 dataset using Embedded Method.

Remarkably, the utilization of feature selection techniques based on the concept of iteration, involving progressive feature addition, elimination, and recursive refinement, displayed remarkable classification performance. These iterative approaches exhibited an accuracy surpassing 85% in the case of CM1 and 95% in the case of PC2, as evidenced by the data presented in **Figure 8** and **Figure 9**.

Though Wrapper-based, and Embedded feature selection techniques are performing well for software defect prediction because they use a learning algorithm to iteratively evaluate feature subsets, resulting in higher accuracy by identifying the most relevant features. Their exceptional search capabilities capture feature interaction and specificity of prediction, allowing them to outperform in identifying the optimal feature subset. This flexibility in evaluating learning algorithms used makes them the preferred methods for software defect prediction. The selection of an appropriate feature selection method relies heavily on the unique characteristics of the dataset under analysis and the specific goals of the researcher. For instance, when the objective is to identify the most vital features within a dataset, Filter methods prove advantageous as they offer a ranked list of features and require fewer computational resources. Conversely, in scenarios where the dataset contains a limited number of features, typically ranging from tens to hundreds, Wrapper methods tend to yield the best predictive performance. In such cases, employing model selection algorithms

can help identify the optimal wrapper algorithm for the given task. However, when dealing with large-scale datasets comprising millions of features, both wrapper and embedded methods may not be practically feasible due to computational constraints and challenges in handling class imbalances. Despite their capability to model feature dependencies and achieve superior classifier accuracy compared to filter methods, their efficiency diminishes in such scenarios.

Moreover, the author suggests the exploration of a hybrid feature selection method that combines the strengths of filter, wrapper, and embedded methods for future research. In recent times also, novel feature selection strategies have emerged, including hybrid methods that combine various feature selection techniques in a two-step approach, as well as ensemble methods that leverage the outputs of multiple feature selection methods. This approach involves a multi-step procedure where filter methods are initially employed to eliminate irrelevant or redundant features based on their individual characteristics. Subsequently, wrapper methods are utilized to assess various feature subsets using a specific learning algorithm to identify the optimal subset. Finally, embedded methods can be employed to further refine the selected features during the model training phase.

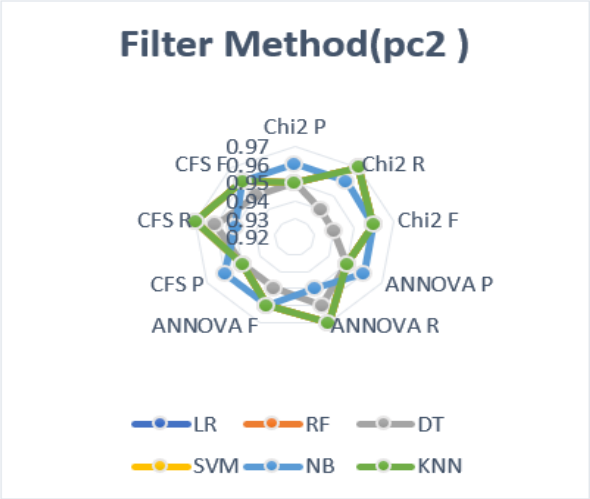

Figure 9(a). Results of each prediction model using performance metrics Precision, Recall, and F1-Score on the PC2 dataset using Filter Method.

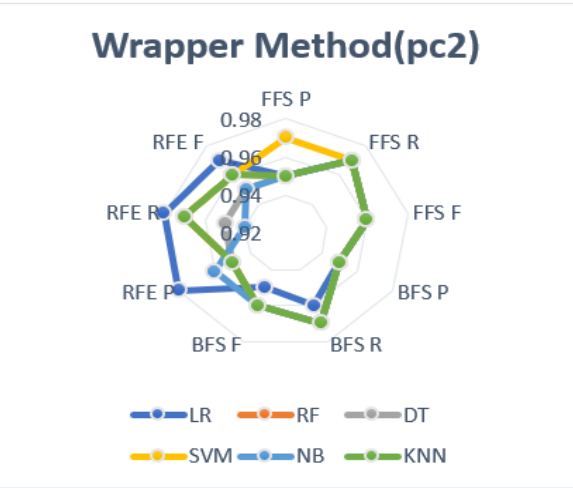

Figure 9(b). Results of each prediction model using performance metrics Precision, Recall, and F1-Score on the PC2 dataset using Wrapper Method.

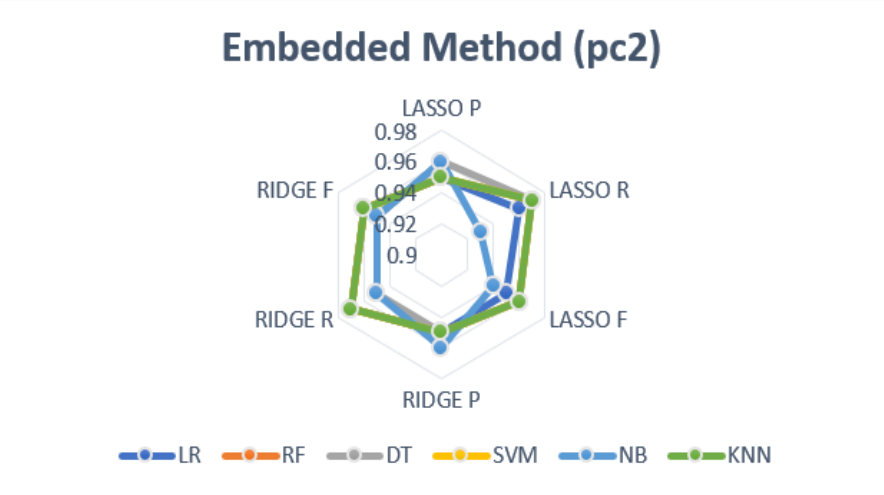

Figure 9(c). Results of each prediction model using performance metrics Precision, Recall, and F1-Score on the PC2 dataset using Embedded Method.

The hybrid method aims to achieve a balance between computational efficiency, predictive performance, and generalizability. By leveraging the advantages of each method while mitigating their respective limitations, this approach leads to a more robust and effective feature selection process. Consequently, it enhances model performance and interpretability, offering improved outcomes in terms of both accuracy and comprehensibility. These approaches harness the strengths of different feature selection methods they incorporate, leading to enhanced outcomes and improved results.

6. Conclusion

In the realm of software engineering, a multitude of substantial datasets has been generated, many of which contain redundant or extraneous features. As such, the process of selecting the best attributes plays a significant role in the initial stages of machine learning operations. Its primary objective is to identify and eliminate unnecessary attributes, resulting in a reduced feature set and improved classification accuracy. This study is based on eight different feature selection techniques to identify effective features from the datasets. By extensively experimenting with six different machine learning algorithms, the study achieved notable classifier accuracies. Specifically, for datasets CM1 and PC2, the achieved classifier accuracies were 88% and 97% respectively. After executing an extensive comparison of feature selection techniques based on accuracy and performance metrics, it became apparent that Correlation feature selection, Recursive feature elimination, and Ridge each possess their own strengths and weaknesses in terms of performance metrics. Correlation feature selection takes feature correlation into account but might miss other predictive factors, while Recursive feature elimination excels at capturing feature interactions but may overlook individual feature characteristics. On the other hand, Lasso regularization effectively reduces dimensionality but struggles with complex relationships, while Ridge feature selection addresses multicollinearity issues and provides stable coefficient estimates. After conducting rigorous experimentation, we achieved higher classifier accuracies with datasets CM1 and PC2 using three distinct feature selection techniques: Correlation feature selection, Recursive feature elimination, and Ridge feature selection. These techniques resulted in a reduction of the number of selected features, showcasing their ability to identify the most relevant features for classification tasks.

7. Future scope

In the concluding statements, it was observed that Correlation feature selection, Recursive feature elimination, and Ridge feature selection techniques exhibit distinctive merits and limitations. Each of these approaches has its own set of advantages and drawbacks that must be considered when selecting the most appropriate method for a given task. To overcome these limitations in the future, it is recommended to adopt “Hybrid feature selection techniques” that integrate multiple methods, resulting in a more comprehensive and robust process. These blended strategies will not only enhance prediction accuracy and model performance but will also provide deeper insights into the causes of software defects. Their flexibility and adaptability would allow for customization to specific requirements, making them valuable assets in software defect prediction. Additionally, the next identified and important task is to address class imbalance issues within the datasets that can introduce biased model training, favoring the majority class and causing challenges in accurately identifying and predicting instances of the minority class. As a result, the overall performance of the defect prediction is adversely affected, compromising its effectiveness in detecting defects across the entire dataset. By leveraging these synthetic methods and effectively handling class imbalance, the authors anticipate a significant improvement in the overall performance and accuracy of the predictive system. Furthermore, the authors aim to explore alternative software defect prediction datasets and integrate discovered models into the system, harnessing the advanced features of these models to make notable contributions to the software development industry.

Author contributions

Conceptualization, KP and TS; methodology, TS and KP; validation, KP, AJ and SB; formal Analysis, TS and KP; data Curation, TS; writing—original draft preparation, TS; writing—review and editing, AJ and SB; visualization, TS. All authors have read and agreed to the published version of the manuscript.

Conflict of interest

The authors declare no conflict of interest.

References

1. The Standish Group report 83.9% of IT projects partially or completely fail. Available online: https://www.linkedin.com/pulse/standish-group-report-839-projects-partially-fail-akbar-shahamati (accessed on 1 June 2020).
2. Anand K, Jena A, Choudhary T, Performance Analysis of Feature Selection Techniques in Software Defect Prediction using Machine Learning," 2022 International Conference on Advancements in Smart, Secure and Intelligent Computing (ASSIC), Bhubaneswar, India, 2022, pp. 1-7, doi:10.1109/ASSIC55218.2022.10088364.
3. Kondo M, Bezemer CP, Kamei Y, Hassan AE, Mizuno O. The impact of feature reduction techniques on defect prediction models. Empirical Software Engineering 2019;24(4):1925-1963. doi:10.1007/s10664-018-9679-5
4. Dao FY, Lv H, Wang F, et al. Identify origin of replication in Saccharomyces cerevisiae using two-step feature selection technique. Bioinformatics 2018; 35(12): 2075–2083. doi: 10.1093/bioinformatics/bty943
5. Rostami M, Berahmand K, Nasiri E, Forouzandeh S. Review of swarm intelligence-based feature selection methods. Engineering Applications of Artificial Intelligence 2021; 100: 104210. doi: 10.1016/j.engappai.2021.104210
6. Chen L, Wang C, Song S. Software defect prediction based on nested-stacking and heterogeneous feature selection. Complex & Intelligent Systems 2022; 8(4): 3333–3348. doi: 10.1007/s40747-022-00676-y
7. Qu K, Gao F, Guo F, Zou Q. Taxonomy dimension reduction for colorectal cancer prediction. Computational Biology and Chemistry 2019; 83: 107160. doi: 10.1016/j.compbiolchem.2019.107160
8. Umbarkar S, Shukla S. Analysis of heuristic based feature reduction method in intrusion detection system. In: Proceedings of the 2018 5th International Conference on Signal Processing and Integrated Networks (SPIN); 22-23 February 2018; Noida, India. pp. 717–720
9. Zebari R, Abdulazeez A, Zeebaree D, et al. A comprehensive review of dimensionality reduction techniques for feature selection and feature extraction. Journal of Applied Science and Technology Trends 2020; 1(2): 56–70. doi: 10.38094/jastt1224
10. Spencer R, Thabtah F, Abdelhamid N, Thompson M. Exploring feature selection and classification methods for predicting heart disease. Digital Health 2020; 6: 205520762091477. doi: 10.1177/205207620914777
11. Pabreja K, Singh A, Singh R, et al. Prediction of stress level on indian working professionals using machine learning. International Journal of Human Capital and Information Technology Professionals 2022; 13(1): 1–26. doi: 10.4018/ijhcitp.297077
12. Pabreja K, Arya S, Madnani P. An ensemble machine learning model for automatic prediction of perceived personal well-being of Indian university students during COVID-19 lockdown. MIER Journal of Educational Studies Trends and Practices 2022; 12(2): 301–319. doi: 10.52634/mier/2022/v12/i2/2280
13. Marian Z, Mircea IG, Czibula IG, Czibula G. A novel approach for software defect prediction using fuzzy decision trees. In: Proceedings of the 2016 18th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC); 24–27 September 2016; Timisoara, Romania. pp. 240–247.
14. Elavarasan D, Vincent P M DR, Srinivasan K, Chang CY. A hybrid CFS filter and RF-RFE wrapper-based feature extraction for enhanced agricultural crop yield prediction modeling. Agriculture 2020; 10(9): 400. doi: 10.3390/agriculture10090400
15. Aggrawal R, Pal S. Sequential feature selection and machine learning algorithm-based patient's death events prediction and diagnosis in heart disease. SN Computer Science 2020; 1(6): 344. doi: 10.1007/s42979-020-00370-1
16. Gárate-Escamila AK, Hajjam El Hassani A, Andrès E. Classification models for heart disease prediction using feature selection and PCA. Informatics in Medicine Unlocked 2020; 19: 100330. doi: 10.1016/j.imu.2020.100330
17. Bolón-Canedo V, Sánchez-Maroño N, Alonso-Betanzos A. A review of feature selection methods on synthetic data. Knowledge and Information Systems 2012; 34(3): 483–519. doi: 10.1007/s10115-012-0487-8
18. Liu Z, Wang J, Liu G, Zhang L. Discriminative low-rank preserving projection for dimensionality reduction. Applied Soft Computing 2019; 85: 105768. doi: 10.1016/j.asoc.2019.105768
19. Yuan X, Yuan J, Jiang T, Ain QU. Integrated long-term stock selection models based on feature selection and machine learning algorithms for China stock market. IEEE Access 2020; 8: 22672–22685. doi: 10.1109/access.2020.2969293

20. Khandelwal R. Feature selection in Python using the Filter method. Available online: https://towardsdatascience.com/feature-selection-in-python-using-filter-method-7ae5cbc4ee05 (accessed on 20 October 2023).
21. Tan J. Feature selection for machine learning in Python—Wrapper methods. Available online: https://towardsdatascience.com/feature-selection-for-machine-learning-in-python-wrapper-methods-2b5e27d2db31 (accessed on 14 October 2020).
22. Gupta A. A comprehensive guide to Feature Selection using Wrapper Methods in Python. Available online: https://www.analyticsvidhya.com/blog/2020/10/feature-selection-techniques-in-machine-learning/ (accessed on 20 October 2023).
23. Sharma T, Jatain A, Bhaskar S, Pabreja K. Ensemble machine learning paradigms in software defect prediction. Procedia Computer Science 2023; 218: 199–209. doi: 10.1016/j.procs.2023.01.002