Journal of Computer Science

Research Article

Uncertainty-Aware Ensemble Models for Improved Defect Detection in Noisy Data

Madhavi Perla[1], Gadi Lava Raju[2], A Radha Krishna[3], E. Sree Devi[4], Bechoo Lal[4], Aruna Bhaskar K[4] and Solleti Phani Kumar[4]

[1]*Department of Computer Science and Engineering, AI&ML, GMR Institute of Technology (GMRIT), Rajam, Andhra Pradesh, India*
[2]*Department of Computer Science and Engineering, Aditya University, Aditya Nagar, Surampalem, Andhra Pradesh, India*
[3]*CSE (AI and ML) Department, Pragati Engineering College, Surampalem, Andhra Pradesh, India*
[4]*Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation (KLEF), K L University, Guntur, Andhra Pradesh, India*

Article history
Received: 31-03-2025
Revised: 04-06-2025
Accepted: 27-06-2025

Corresponding Author:
Bechoo Lal
Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation (KLEF), K L University, Guntur, Andhra Pradesh, India
Email: drblalpersonal@gmail.com

Abstract: Software defect prediction plays a crucial role in ensuring software quality and reliability, especially as modern systems become more complex and data rich. This study introduces an uncertainty-aware ensemble learning framework aimed at improving defect classification performance in noisy and imbalanced datasets, particularly those from the PROMISE and NASA KC1 repositories. The proposed model integrates multiple classifiers in a multi-learner ensemble structure to enhance generalization, improve true positive rates, and address the limitations of conventional single-model approaches. Key techniques include chi-square-based feature selection, ensemble pruning to avoid overfitting, and neural network-based classification through Extreme Learning Machines (ELMs). The methodology emphasizes the use of both homogeneous and heterogeneous ensembles, with training and prediction phases structured to handle data sparsity, high dimensionality, and class imbalance. Runtime experiments using decision trees, Naïve Bayes, and cost-sensitive learning demonstrated superior results for the ensemble model compared to traditional classifiers. Evaluation metrics such as accuracy, F-measure (0.9729), recall (0.7143), true positive rate (0.9857), and ROC AUC further validated the ensemble's predictive robustness. Experimental results on the KC1 dataset showed that the proposed model outperformed baseline models in both accuracy and area under the ROC curve. Advanced data balancing techniques, including under-sampling, over-sampling, and active learning, were employed to improve the model's ability to identify minority class instances. These findings suggest that uncertainty-aware ensemble approaches are effective tools for improving defect detection, particularly in noisy and imbalanced environments.

Keywords: Software Defect Prediction, Ensemble Learning, Promise Dataset, Neural Networks Class Imbalance, ROC Curve, Extreme Learning Machine (ELM), Naïve Bayes, Feature Selection, Software Quality

Introduction

As software defect size increases, accurately predicting various defect types with a high true positive rate becomes more challenging (Raj and Singh, 2017). The primary goal of this study is to enhance defect prediction and classification performance on the PROMISE defect dataset, which presents considerable variability in defect types, dimensionality, data volume, and sparsity (Ali et al., 2024; Zhang et al., 2022). Accurate evaluation of software metrics and defect prediction are crucial quality determinants that directly impact the success of a software product.

Traditional classification models typically rely on metric-based relationships and probabilistic estimators for defect prediction (Balogun et al., 2019). Prior approaches

have also explored the comparative effectiveness of McCabe's complexity metrics versus Halstead's lines of code metrics. In contrast, this study introduces a novel multi-learner ensemble model that integrates predictions from multiple trained classifiers to address these challenges (Balogun et al., 2021). Ensemble learning combines outputs from diverse models to improve overall predictive accuracy and mitigate issues such as data imbalance (Balogun et al., 2020).

By leveraging ensemble learning, our approach enhances the true positive rate in defect prediction more effectively than conventional single-model techniques (Kamrun et al., 2017; Dyana Rashid et al., 2017). Furthermore, it offers a robust solution to the problem of class imbalance, a common issue in defect datasets (Lear, 2021). Finally, we recognize that uncertainty a core challenge in machine learning stems from working with incomplete or imperfect information. This aspect often confounds beginners, particularly software engineers accustomed to deterministic systems (Iqbal et al., 2019). In contrast, machine learning embraces probabilistic modelling to map inputs to outputs, whether for regression or classification tasks.

A software defect refers to a flaw that prevents a program from performing its intended functions correctly. Defect prediction plays a critical role in identifying such faults, which may arise from both manual errors and automated processes during various stages of the Software Development Life Cycle (SDLC). As reliance on software systems continues to grow, ensuring high performance and reliability has become increasingly vital. Failures and defects not only compromise user satisfaction but also reflect poorly on software quality (Shepperd et al., 2013). With rising system complexity and tighter software constraints, delivering a high-quality final product poses significant challenges. Undetected defects can lead to financial losses and wasted development time, emphasizing the need for early and accurate prediction methods.

Bug tracking systems typically maintain detailed records of identified defects, including their severity levels across various platforms. These defects represent operational inconsistencies that produce erroneous results, often tied to specific code attributes. Identifying and utilizing these attributes effectively is essential for improving the performance of defect prediction models. Various machine learning techniques such as decision trees, logistic regression, neural networks, Support Vector Machines (SVM), and naïve Bayes have been employed for defect detection. However, many traditional models struggle to accurately select the most relevant defect features for precise classification, limiting their overall effectiveness.

Naïve Bayes is an effective classification technique widely used for software defect prediction, leveraging historical data to distinguish between defective and non-defective cases. It treats bug prediction as a binary classification problem by analysing past software metrics to train a predictive model. However, when the dataset includes mixed attribute types, missing values, or uncertain data, the accuracy of predictions can be compromised (Tantithamthavorn et al., 2020).

Dynamic analysis for defect detection typically operates across three conceptual layers. The first is the systematic testing layer, which involves predefined strategies to execute the target program and expose potential error states. The second is the data collection layer, where information about the internal behaviour of the software during execution is gathered to verify its correctness. The final layer is the model abstraction layer, where the collected behavioural data is used to construct a program of an abstract model, which is then analysed to detect possible faults (Soe et al., 2018).

Despite its advantages, dynamic analysis has inherent limitations. It only examines parts of the software that are executed during testing, making full program analysis impossible. Moreover, it requires executable environments and representative input data, which can be challenging to create. Research such as that cited in (Omri and Sinz, 2020) emphasizes the importance of various software metrics in prediction models. These models often apply different algorithms to explore metric correlations, calculate bug frequencies, and enhance forecast accuracy. Object-oriented metrics are considered valuable for assessing the quality of object-oriented systems.

Additionally, a defect prediction framework was proposed to classify defects based on severity high, medium, or low. The outcomes yield that this model fulfilled vantage in identifying severe defects compared to traditional approaches (Li et al., 2018). In general, regression techniques are used to estimate the quantity and density of software defects, while classification methods aim to determine whether a specific software component (such as a module, class, or file) is likely to be defective.

Data Classification Using Neural Networks

Extreme Learning Machine (ELM) is recognized for its rapid training capabilities equated to algorithms of iterative for gradient-based ancestral such as the Backpropagation method. In addition to its speed, ELM demonstrates robust generalization performance and eliminates the need for tedious parameter tuning. A refined version of the ELM algorithm introduces several improvements: Exceptionally fast learning, enhanced generalization, and a parameter-free configuration (Wei et al., 2025). One innovative method to tackle the overfitting problem, which is common in traditional neural network models, involves a voting-based ELM ensemble. In this trail, ELM models in multiple are trained independently on randomly selected data subsets, and their outputs are combined using a majority

voting mechanism. This technique builds classifiers by selecting a subset of random input data and a corresponding feature sub-set, then training the ELM model using this reduced dataset. It effectively addresses challenges like slow learning speeds and suboptimal generalization associated with conventional Single Layer Feedforward Networks (SLFNs). ELM is widely applied in both classification and regression tasks due to its consistent performance (Dam, 2018).

A study cited introduced a novel ensemble machine learning technique aimed at improving defect classification and addressing limitations of traditional classification models. Most conventional approaches rely heavily on analysing the morphological characteristics of datasets, such as those in NASA's defect repositories. This study incorporated three key classification strategies single C4.5 decision trees, Bagging, and AdaBoost demonstrating their effectiveness in enhancing classification accuracy. The proposed defect prediction method aims to assist healthcare professionals by enabling early and reliable identification of software defects (Zhang et al., 2016).

In related fields like text categorization, class imbalance is a frequent challenge. These datasets often include many irrelevant documents and relatively few items of actual interest. Bayesian Network (BN) classifiers are frequently used in such scenarios for their accuracy and their ability to model variable relationships (Bahaweres et al., 2020). However, BN classifiers struggle with imbalanced datasets. To mitigate this, several advanced classification techniques have been developed to improve performance in skewed data environments. These include sampling-based methods, cost-sensitive learning, recognition-driven models, and active learning techniques:

- Under-sampling addresses imbalance by reducing the number of majority class instances
- Over-sampling involves augmenting the minority class with synthetic samples
- Cost-sensitive learning introduces error cost matrices to penalize misclassifications more effectively and guides learning even when data is unevenly distributed

By using both homogeneous and heterogeneous voting ensembles to evaluate several machine learning classifiers, the approach shows that active learning dramatically lowers annotation effort while preserving or enhancing prediction accuracy. When compared to conventional single-classifier methods, extensive testing on benchmark PROMISE datasets demonstrates gains in accuracy, F-measure, and ROC-AUC (Liapis et al., 2024).

Use preprocessing methods like cost-sensitive learning, data standardization, and feature selection in this framework. The ensemble model improves the detection of faulty modules by dynamically choosing the best classifiers depending on performance measures. When compared to standalone deep learning and conventional machine learning models, experimental evaluations on benchmark software defect datasets show higher accuracy, recall, and ROC-AUC (Srinivas et al., 2025).

Recognition-based learning focuses on extracting classification rules primarily from minority class data, sometimes disregarding the majority class entirely. Meanwhile, active learning strategies aim to address the limitations of unlabelled data by selectively querying the most informative samples to improve model training.

In the realm of large-scale data processing, Distributed Data Mining (DDM) has emerged as a powerful approach for developing knowledge-based decision models. As distributed computing systems expand, vast datasets are generated across numerous data centres. Extracting meaningful patterns from these large, heterogeneous, and privacy-sensitive repositories is often infeasible using conventional data mining methods. DDM enables pattern discovery across decentralized databases, functioning similarly to traditional mining methods but optimized for distributed environments.

When dealing with severely imbalanced datasets, under-sampling remains a commonly adopted solution. However, evolutionary algorithms often struggle with high-dimensional data due to computational and memory limitations. To overcome this, divide-and-conquer strategies are employed, where the dataset is partitioned into smaller, manageable subsets for parallel processing. This strategy effectively addresses class imbalance issues even in high-dimensional spaces. The proposed study emphasizes maintaining independence between majority and minority class instances to mitigate the impact of limited sample sizes. Experiments were conducted on datasets containing up to a million records, using domain-agnostic evaluation techniques for better generalizability.

Advanced evolutionary techniques such as “Genetic Algorithms, Rough Set Theory”, and “Support Vector Machines (SVMs)” are frequently utilized for documenting classification in large corpora. Genetic algorithms are effective at identifying complex patterns and deriving classification rules. The study also explores hybrid models, integrating genetic algorithms with decision tree techniques to produce optimized classification structures.

An ensemble-based method for predicting software defects that aims to outperform conventional single-model methods in classification. To improve prediction accuracy and resilience, the suggested system uses ensemble techniques to integrate several machine learning classifiers. In order to manage high-dimensional software metrics and minimize noise, the study places a strong emphasis on efficient feature selection and data pretreatment. The ensemble model outperforms individual base classifiers in terms of accuracy, precision, recall, and F-measure, according to experiments done on

benchmark PROMISE datasets (Harikiran et al., 2023).

To increase predicted accuracy and resilience, the system combines different classifiers at the decision level and integrates several software metrics through data-level fusion. The authors use feature engineering, normalization, and ensemble voting techniques to solve issues such data heterogeneity, noise, and class imbalance. The fusion-based ensemble technique outperforms individual classifiers and traditional models in terms of accuracy, precision, recall, and F-measure, according to experimental results on benchmark software fault datasets (Abbas et al., 2023).

Improving software defect prediction performance in noisy and highly imbalanced datasets common problems in real-world software repositories—is the main goal of this research. To improve defect classification accuracy, the authors suggest a data-centric framework that incorporates ensemble learning, data balancing methods, and noise filtering. In order to reduce class imbalance, their method combines under-sampling and oversampling procedures, and noise reduction techniques are used to enhance data quality. When compared to traditional machine learning models, extensive trials on benchmark PROMISE datasets show notable gains in recall, F-measure, and ROC-AUC (Shi et al., 2023).

Combine several versions of the same basic classifier that have been trained on various data subsets to create homogeneous ensemble models for software fault prediction. By lowering variance and increasing prediction stability, their method outperforms single classifiers on benchmark defect datasets in terms of accuracy, recall, and F-measure (Mamatha and Kumari, 2023) In order to lower labelling costs while enhancing performance, Liapis et al. present a voting ensemble-based defect prediction model combined with active learning. By merging several classifiers and selectively training on the most informative examples, the method improves accuracy and generalization, especially in unbalanced datasets (Liapis et al., 2024).

Lastly, Artificial Neural Networks (ANNs) are modelled as interconnected neurons that simulate brain-like structures for learning. They are capable of approximating inputs and outputs in pattern recognition and statistical learning scenarios. ANNs adjust to unknown inputs through network training and are widely used in classification tasks. K-Nearest Neighbours (KNN), another classification method, operates on a lazy learning principle where training occurs at classification time. It evaluates proximity-based similarity metrics and uses kernel functions to optimize feature estimation in biological defect datasets, making it effective for document classification within bioinformatics repositories (Ghotra et al., 2017).

Problem Statement and Literature Survey

Predicting software defects is crucial for ensuring software reliability, yet conventional machine learning models struggle with imbalanced datasets, mixed attribute types (both categorical and continuous), and optimal feature selection. Algorithms like Naïve Bayes, decision trees, and SVMs often suffer from overfitting, data uncertainty, and ineffective handling of class imbalance. Although ensemble methods and dynamic analysis improve predictive capabilities, they face challenges such as computational inefficiency, incomplete code analysis, and false alarms. This study introduces an ensemble-based framework that incorporates chi-square feature selection with probabilistic classifiers to enhance accuracy, reduce misclassification, and improve generalization across diverse software repositories, including NASA's KC1 dataset.

Gaps in Research

Feature Selection: Conventional models struggle to pinpoint relevant attributes for defect prediction, leading to suboptimal classification.

Class Imbalance: Standard classifiers perform poorly on imbalanced datasets, while oversampling and under sampling can cause overfitting or data loss.

Mixed Data Types: Few approaches effectively handle datasets containing both categorical and continuous attributes, especially when faced with missing or uncertain values.

Dynamic Analysis Limitations: Dependency on executable environments and incomplete code coverage impacts defect detection accuracy.

Overfitting: Decision trees and neural networks require careful tuning to prevent excessive model complexity.

Uncertainty Handling: Existing models do not comprehensively address uncertainty in defect prediction within high-dimensional datasets.

Literature Survey

The prediction of defect in the software field has progressed significantly with the integration of Machine Learning (ML) techniques, yet achieving high accuracy and generalizability remains a challenge. Early research explored software metrics such as McCabe's cyclomatic complexity and Halstead's measures, which were commonly applied in datasets like NASA's KC1 for training predictive models. Naïve Bayes emerged as a favoured probabilistic method for binary classification, relying on historical defect data. However, its efficiency is hindered when handling mixed data types and missing values, underscoring the necessity for robust preprocessing strategies.

Dynamic analysis frameworks have been employed to identify runtime defects by operating across testing, data collection, and abstraction layers. Although effective in validating software behaviour, these methods are

constrained by their reliance on representative input data and their inability to assess untested code paths. Research has highlighted the significance of metric correlations in prediction models, advocating for object-oriented metrics to assess software quality. Additionally, severity-based defect classification frameworks have demonstrated superior performance compared to conventional methods.

To tackle imbalance of class, sampling strategies such as oversampling (minority synthetic generation) and under sampler (random majority reduction) have been explored alongside cost-sensitive learning approaches. However, oversampling increases the risk of overfitting, whereas under sampling might remove crucial majority-class instances. Alternative strategies like recognition-based and active learning methods have been proposed to enhance minority-class identification, but their scalability remains a concern for large datasets.

Ensemble learning methods, including Bagging and AdaBoost, have improved predictive stability by integrating multiple classifiers. Research incorporating C4.5 decision trees within ensemble techniques has demonstrated enhanced accuracy in defect detection within healthcare software. Extreme Learning Machines (ELMs) have also exhibited rapid training benefits and generalization capabilities over gradient-based neural networks. Further, voting-based ELM ensembles have mitigated overfitting by aggregating predictions from diverse data subsets, though computational challenges persist in high-dimensional scenarios.

Distributed Data Mining (DDM) has been introduced to handle large-scale datasets via divide-and-conquer strategies. Evolutionary algorithms such as Genetic Algorithms (GAs) and Rough Set Theory have been combined with decision trees to extract relevant rules. For instance, GA-driven feature selection has improved classification accuracy in document repositories, while Rough Sets have enhanced attribute reduction in bioinformatics applications. Despite their advantages, evolutionary models often struggle with computational demands in extensive datasets.

Artificial Neural Networks (ANNs) and k-Nearest Neighbours (KNN)" have also been utilized for defect prediction. While ANNs excel in pattern recognition, they require extensive parameter tuning, and KNN's similarity-based approach increases classification-time costs. Hybrid models that merge Bayesian Networks with ensemble trees have proven effective in uncertainty management but lack scalability. Decision tree pruning techniques aim to balance complexity and accuracy while minimizing false alarms.

In conclusion, although ML and ensemble frameworks have significantly advanced defect prediction, key limitations persist:

- Ineffective handling of mixed data types and missing values
- Scalability challenges in high-dimensional datasets
- Dependence on balanced datasets or suboptimal sampling methods
- Absence of comprehensive models addressing uncertainty and dynamic analysis constraints
- This study proposes an ensemble-based approach integrating chi-square feature selection and probabilistic classifiers to bridge these gaps and improve defect prediction accuracy

Methods

This study introduces a classification model based of ensemble designed to tackle the challenges of classes of imbalance and improving prediction defects in software. The proposed framework integrates feature selection and probability estimation techniques to enhance prediction accuracy. Ensemble learning, which involves combining the outputs of multiple trained classifiers, is utilized to boost the true positive rate and mitigate the negative effects of imbalanced data (Fig. 1).

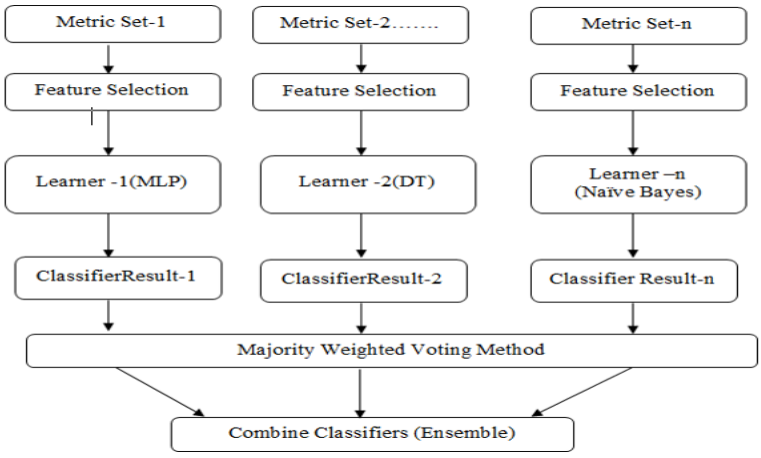

Fig. 1: Ensemble feature selection model for the proposed method

There are two primary types of ensemble learners: Homogeneous ensembles, which use the same algorithm across all base models, and heterogeneous ensembles, which incorporate diverse classification algorithms. Each base learner in the proposed model is constructed using a distinct classification algorithm, contributing to the robustness of the overall system:

$$H * = F(h1, h2, \ldots, hs)$$

(or)

$$H(xi) = \text{sign}(j = 1 \sum N \, wj \, hj \, (xi))$$

Where $w_j$is the weight of the selected classifier h_j.

The multi-learner ensemble model is structured into two main stages: Model training and model deployment. During the training phase, an ensemble denoted as E is formed by creating multiple base classifiers. The data of training is divided into sub-sets, with each sub-set utilized to retinue base classifier of an individual. In the deployment phase, predictions from all base classifiers are combined using an integration function, H, to produce the final output model of ensemble.

Overall, ensemble models tend to outperform individual base classifiers in defect prediction tasks, offering higher reliability and accuracy in identifying software defects.

Function for Selection Feature

Chi-Square-Based Defect Selection

The chi-square method can be utilized to evaluate defects by computing chi-square statistics in relation to the defect class distribution. As a non-parametric statistical technique, it helps identify discrepancies between the observed defect distribution and the expected defect occurrence. By applying this approach, variations in defect patterns can be analysed, leading to better classification and decision-making in quality assessment and other analytical domains.

Algorithm for Method Filtering

Input: Defect datasets as DList, A is attribute, I(A) is instance of A, S.D: Standard deviation, **Prob:** Probability, v : feature value, C_m : mth class.
Output: Classified defect results.
Procedure:
// Ensemble Selection
For each metric set Data[i] in DList Do
For each attribute A_j in Data[i] Do
For each instance $I(A_j)$ in A_j do
If (Typeof($I(A_j)$) == Numerical && A_j == Null) Then
$I(A_j) = (\text{Max}(D1(A_j), D2(A_j)) / (\mu - \text{S.D}(D1, D2)))$; ---------- (3.2)
End if End for
End for
// Nominal attributes
If (Typeof($I(A_j)$) == Nominal && Then A_j = Null)
Find conditional posterior probability estimation as
$P_1(v_i) = \log(\text{Prob}(v_i / C_m)) + \log(\text{Prob}(C_m))$.
$P_2(v_j) = \log(\text{Prob}(v_j / C_m)) + \log(\text{Prob}(C_m))$.
$P_1(v_i)$ is associated with classifier-1 and $P_2(v_j)$ is associated with classifier-2.
$A_j = \text{maxprob}(\{P_1(v_i)\}, \{P_2(v_j)\})$;

Algorithm for Model Detection for Ensemble.

Input: Filtered data FA(D), O_e is the output and x_i is the input vector with M neural networks.
For each filtered subset in filtered datasets
Do
For each attribute $FA(D_i)$ do
Divide the data instances of $FA(D_i)$ into „k" independent subsets.
Select classifier C_i
While i<k
Do
If (C_i is MLE) Then

$$O_{ens} = (1/M) * \sum_{m,i=1}^{M} O_m(x_i); \quad \text{---------- (3.4)}$$

Where O_m is the output and x_i is the input vector with M neural networks.
Else if (C_i is BN)
Then
Let $V = \{V_1, V_2 \ldots V_n\}$ be the discrete or continuous random variables used in the Bayesian computation values for defect prediction model. The probability Computation of V_i is shown as $P(V_i / a_x)$ where a_x represents the parent nodes of V_i
Then the joint probability distribution of X can be given as

$$Prob(V) = Prob(V_1, V_2, \ldots V_n)$$

$$\text{Prob}(V) = \text{Prob}(V_1, V_2 \ldots V_n)$$
$$\text{Prob}(V_1 / V_2, \ldots V_n).\text{Prob}(V_2 / V_3, \ldots V_n) \ldots \text{Prob}(V_{n-1} / V_n)\,\text{Prob}(V_n)$$

$$= \prod_{i=1}^{n} \text{Prob}(V_i / V_{i+1}, \ldots V_n)$$

Else DT(i);//decision
tree End if
End while
Calculate misclassified error rate and statistical true positive rate;
Done
Done

In this framework, the output from Algorithm 1 where filtered data is used to measure the proposed is provided as input to Algorithm 2, which applies an ensemble learning model to predict defect accuracy. Misclassification error refers to instances where data points known to belong to a specific category are incorrectly classified into another. Rather than modelling misclassification directly, the focus is on minimizing it, and the most suitable method for doing so depends on the problem nature and the attributions of the dataset. The most effective approach is the one that results in the lowest error of misclassification. The Error Rate (ERR) is defined as the ratio of predictions of incorrect to the number of total instances in the dataset.

The system uses standard datasets as input to handle values of m is laid in both continual and categorical attributes. Once filtered, these datasets are utilized to derive phase-specific patterns of decision, which support classification through an ensemble model. Naïve Bayes, a robust probabilistic classifier, is used to predict defect presence by learning from historical metric data. It treats defect prediction as a binary classification task. However, when the dataset includes a mix of attribute types, defect prediction becomes more challenging, especially when data is incomplete or uncertain.

To avoid overfitting in decision trees, pruning techniques are essential. Two primary pruning strategies

exist: pre-pruning and post-pruning. Pre-pruning introduces a threshold parameter α during tree construction. If the impurity reduction caused by a node's split falls below α, the node is treated as a leaf, halting further tree growth. Choosing the right α value is critical setting it too high may prematurely halt useful splits, reducing model accuracy, while a very small α may result in a complex tree with little benefit over the unpruned version.

Results

KC1 is a publicly available dataset from NASA's software metrics repository, designed to support the estimation and enhancement of models of predictive in software engineering. It originates from a C++ program responsible for managing ground-based data storage used in signal reception and processing. The dataset includes metrics derived from McCabe's cyclomatic complexity and Halstead's software science measures, with a primary focus on individual software modules.

In this context, the probability of detecting software defects is directly related to the amount of effort invested higher detection accuracy typically demands greater effort. However, as detection effectiveness improves, the chance of generating false alarms tends to decrease. This inverse relationship is illustrated by the Receiver Operating Characteristic (ROC) curve, which plots the trade-off between true positive and false positive rates.

Results for Decision Patterns

Decision Tree Evaluation Summary: Dataset 1

Key Classification Rules
1. Initial Split: numberOfMethodsInherited< 34.5
If numberOfPublicAttributes< 3.5 **and** numberOfPublicMethods< 0.5 → **Defect (Y)**
Else, if wmc< 2.5 → **Defect (Y)**
Otherwise, if numberOfAttributes< 5.5 and numberOfLinesofCode< 122.5:
If dit< 2.5 and wmc< 16 and noc< 1.5:
If fanIn< 2.5 → **Defect (Y)**
Else, evaluate based on numberOfLinesofCode< 73.5
2. Alternative Path: numberOfMethodsInherited ≥ 58.5
If cbo< 6.5 → **Defect (Y)**
Else, if dit< 6.5:
If fanIn< 0.5 → **Defect (Y)**
Else, if numberOfPublicMethods< 7.5 → **Defect (Y)**
3. Lines of Code Path: numberOfLinesofCode ≥ 125.5
If numberOfPrivateAttributes< 3.5:
Follow nested checks involving thresholds on wmc (e.g., 68.5) and lcom (e.g., 144.5)
Else, if rfc< 85.5 → **No Defect (N)**

Decision Tree Evaluation Summary: Dataset 2
Key Classification Rules
1. Main Split: numberOfPrivateMethods< 0.5
If fanOut< 3.5:
Further split on numberOfPublicAttributes< 29 and numberOfMethodsInherited< 9.5
Else:
If numberOfAttributesInherited< 14.5:
Check numberOfLinesofCode< 55.5 and related features
2. Alternate Branch: numberOfPrivateMethods ≥ 0.5
If fanOut< 20.5:
Evaluate based on thresholds for wmc (< 2.5) and numberOfLinesofCode (< 116.5)
Else:
If numberOfPublicMethods< 3.5 → **Defect (Y)**
Otherwise, check if numberOfAttributes< 27.5 for further classification

Notations & Definitions
Y: Indicates a module is predicted to contain a defect
N: Indicates no defect is predicted
Thresholds: Split points (e.g., < 34.5) used in the decision rules
Feature Definitions:
wmc: Weighted Methods per Class
dit: Depth of Inheritance Tree
cbo: Coupling Between Object Classes
lcom: Lack of Cohesion in Methods

Insights & Patterns
Strong Precision: High F-measure (~0.97) reflects precise defect identification.
Moderate Recall: Recall of ~0.71 suggests some missed defect cases.
Influential Metrics:
NumberOfMethods Inherited (key thresholds: 34.5, 58.5)
NumberOfLinesofCode (notable at 125.5 in Dataset 1)
fanOut (splits at 3.5 and 20.5 in Dataset 2)
Deep Tree Complexity: Complex conditions emerge in paths with:
High coupling (cbo ≥ 6.5)
Deeper inheritance hierarchies (dit ≥ 6.5)

Elapsed time:181.074s
Classification true positive rate:93.545% Incorrectly instances: 6.7%

Table 1: Performance Metrics

Metric	Value
Elapsed Time	64.276s
Number of Iterations	26
F-Measure	0.97291
Recall	0.71432
True Positive Rate	0.98565
False Positive Rate	0.01435
Classification Accuracy	0.97056

Table 1 presents the performance metrics and their corresponding values.

Accuracy

Classification Accuracy refers to the accuracy of a model in predicting the correct outcomes. It is calculated as the symmetry of predictions correctly made by the compared model to the total number of intake sampling.

Accuracy = Number of corrected effect predictions/Total number of defect.

Table 2 presents a comparative analysis between the proposed model and conventional classification methods, focusing on accuracy as the primary evaluation metric. The outcomes certainly demonstrate that the proposed model outcompetes traditional approaches in terms of classification accuracy. Area Under the Curve (AUC) is a widely recognized metric for evaluating the effectiveness of classifiers of binary. It represents the probability that the model will entrust a score of higher to a selected positive instance at random selected negative one.

Figure 2 illustrates the AUC comparison between the ensemble-based model and standard classification techniques, highlighting the true positive rate. The findings indicate that the ensemble model achieves a notably higher accuracy than the traditional methods.

Table 2: Classification models of classification comparison in terms of accuracy

Algorithm	KC1	CM1	JM1	MC2	PC4	MW1	PC1	PC3
Ensemble Modle	0.97	0.94	0.96	0.972	0.968	0.965	0.971	0.968
SVM	0.92	0.919	0.945	0.967	0.925	0.931	0.96	0.59
DT	0.87	0.8167	0.896	0.91	0.932	0.932	0.856	0.887
NaiveBayes	0.911	0.897	0.885	0.8545	0.832	0.942	0.9109	0.91

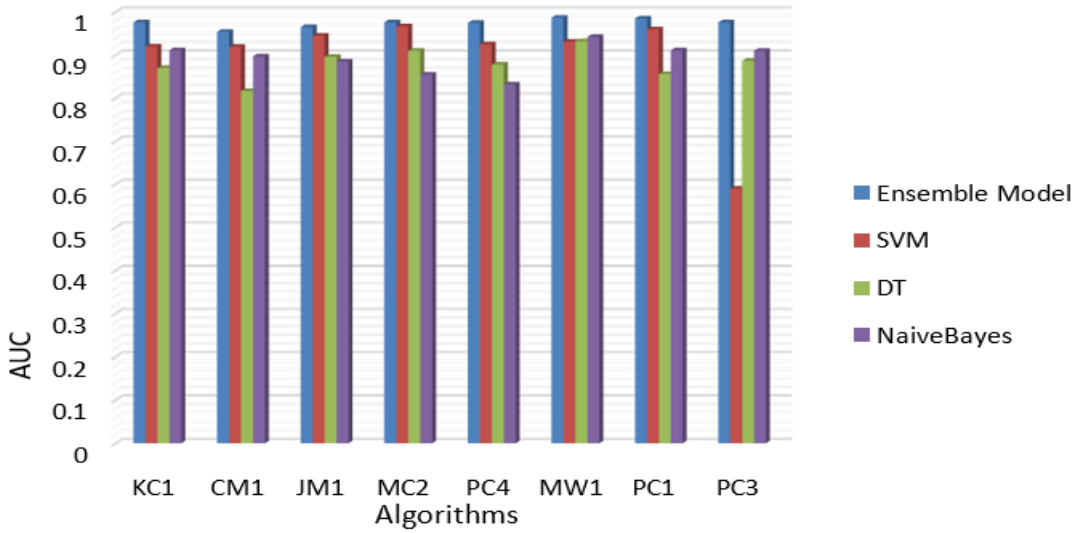

Fig. 2: Comparison of AUC model of ensemble with models of classification

Conclusion

This study presents a comprehensive ensemble-based methodology to enhance software defect prediction, especially in datasets characterized by noise, missing values, and class imbalance. By integrating diverse classifiers through a multi-learner ensemble framework, the proposed model demonstrates significant improvements in prediction accuracy, true positive rates, and robustness against uncertain data distributions. The use of chi-square-based feature selection and pruning strategies in decision trees ensures that only the most relevant features are retained, thereby reducing overfitting and enhancing generalization.

The Extreme Learning Machine (ELM) model, particularly when combined with ensemble voting, showed remarkable speed and performance, outperforming traditional neural networks and single-layer feedforward networks. Experimental validation on the KC1 dataset confirmed that the ensemble model surpasses conventional classification methods in both accuracy (97.05%) and area under the ROC curve, effectively identifying defective software components with high reliability. Moreover, the ensemble model maintained strong performance despite the presence of imbalanced class distributions, thanks to the integration of cost-sensitive and sampling-based learning strategies.

Further, runtime results from datasets 1 and 2 reinforced the model's ability to deliver high precision (F-measure: 0.9729) and maintain a low false positive rate (1.43%). The true positive rate of over 93% indicates that the ensemble model is well-suited for real-world applications where early and accurate defect detection is critical. In summary, the proposed uncertainty-aware ensemble model offers a

powerful and scalable solution for improving software quality assurance, particularly in environments with complex data and evolving system requirements.

Acknowledgment

Thank you to the publisher for their support in the publication of this research article. We are grateful for the resources and platform provided by the publisher, which have enabled us to share our findings with a wider audience. We appreciate the efforts of the editorial team in reviewing and editing our work, and we are thankful for the opportunity to contribute to the field of research through this publication.

Funding Information

The authors have not received any financial support or funding to report.

Author's Contributions

All authors contributed equally to this study.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and no ethical issues involved.

References

Abbas, S., Aftab, S., Adnan Khan, M., M. Ghazal, T., Al Hamadi, H., & Yeob Yeun, C. (2023). Data and Ensemble Machine Learning Fusion Based Intelligent Software Defect Prediction System. *Computers, Materials & Continua*, *75*(3), 6083–6100. https://doi.org/10.32604/cmc.2023.037933

Ali, M., Mazhar, T., Al-Rasheed, A., Shahzad, T., Yasin Ghadi, Y., & Amir Khan, M. (2024). Enhancing software defect prediction: a framework with improved feature selection and ensemble machine learning. *PeerJ Computer Science*, *10*, e1860. https://doi.org/10.7717/peerj-cs.1860

Bahaweres, R. B., Imam Suroso, A., Wahyu Hutomo, A., Permana Solihin, I., Hermadi, I., & Arkeman, Y. (2020). Tackling Feature Selection Problems with Genetic Algorithms in Software Defect Prediction for Optimization. *Proceedings of the 2020 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS)*, 64–69. https://doi.org/10.1109/icimcis51567.2020.9354282

Balogun, A. O., Basri, S., Abdulkadir, S. J., & Hashim, A. S. (2019). Performance Analysis of Feature Selection Methods in Software Defect Prediction: A Search Method Approach. *Applied Sciences*, *9*(13), 2764. https://doi.org/10.3390/app9132764

Balogun, A. O., Basri, S., Capretz, L. F., Mahamad, S., Imam, A. A., Almomani, M. A., Adeyemo, V. E., Alazzawi, A. K., Bajeh, A. O., & Kumar, G. (2021). Software Defect Prediction Using Wrapper Feature Selection Based on Dynamic Re-Ranking Strategy. *Symmetry*, *13*(11), 2166. https://doi.org/10.3390/sym13112166

Balogun, O., Selamat, A., & Ibrahim, R. (2020). SMOTE based homogeneous ensemble methods for software defect prediction". *Lecture Notes in Computer Science*, 615–631.

Dam, H. K. (2018). A deep tree based model for software defect prediction". *ArXiv (Computer Science > Software Engineering*. https://doi.org/10.48550/arXiv.1802.00921

Dyana Rashid, I., Ghnemat, R., & Hudaib, A. (2017). Software Defect Prediction using Feature Selection and Random Forest Algorithm. *Proceeding of the 2017 International Conference on New Trends in Computing Sciences (ICTCS)*, 252–257. https://doi.org/10.1109/ictcs.2017.39

Ghotra, B., McIntosh, S., & Hassan, A. E. (2017). A Large-Scale Study of the Impact of Feature Selection Techniques on Defect Classification Models. *Proceeding of the 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, 146–157. https://doi.org/10.1109/msr.2017.18

Harikiran, J., Sai Chandana, B., Srinivasarao, B., Raviteja, B., & Subba Reddy, T. (2023). Software Defect Prediction Based Ensemble Approach. *Computer Systems Science and Engineering*, *45*(3), 2313–2331. https://doi.org/10.32604/csse.2023.029689

Iqbal, A., Aftab, S., Ullah, I., Salman Bashir, M., & Anwaar Saeed, M. (2019). A Feature Selection based Ensemble Classification Framework for Software Defect Prediction. *International Journal of Modern Education and Computer Science*, *11*(9), 54–64. https://doi.org/10.5815/ijmecs.2019.09.06

Kamrun, N. N., Ali Asif, S., Seal Ami, A., & Ul Gias, A. (2017). Modeling Software Defects as Anomalies: A Case Study on Promise Repository. *Journal of Software*, *12*(10), 759–772. https://doi.org/10.17706/jsw.12.10.759-772

Lear, A. (2021). Ensemble machine learning model for software defect prediction. *Advances in Computing and Data Sciences*, 317–326.

Li, Z., Jing, X.-Y., & Zhu, X. (2018). Progress on approaches to software defect prediction. *IET Software*, *12*(3), 161–175. https://doi.org/10.1049/iet-sen.2017.0148

Liapis, C. M., Karanikola, A., & Kotsiantis, S. (2024). Data-efficient software defect prediction: A comparative analysis of active learning-enhanced models and voting ensembles. *Information Sciences*, *676*, 120786. https://doi.org/10.1016/j.ins.2024.120786

Mamatha, R., & Kumari, P. L. S. (2023). & Sharada, A.,"Enhanced software defect prediction through homogeneous ensemble models." *International Journal of Intelligent Systems and Applications in Engineering*, *11*(3), 42–51.

Omri, S., & Sinz, C. (2020). Deep Learning for Software Defect Prediction: A Survey. *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops (ICSEW 2020)*, 209–214.

Raj, K., & Singh, K. P. (2017). SVM with Feature Selection and Extraction Techniques for Defect-Prone Software Module Prediction. *Proceedings of Sixth International Conference on Soft Computing for Problem Solving*, *547*, 279–289. https://doi.org/10.1007/978-981-10-3325-4_28

Shepperd, M., Song, Q., Sun, Z., & Mair, C. (2013). Data Quality: Some Comments on the NASA Software Defect Datasets. *IEEE Transactions on Software Engineering*, *39*(9), 1208–1215. https://doi.org/10.1109/tse.2013.11

Shi, H., Ai, J., Liu, J., & Xu, J. (2023). Improving Software Defect Prediction in Noisy Imbalanced Datasets. *Applied Sciences*, *13*(18), 10466. https://doi.org/10.3390/app131810466

Soe, Y. N., Santosa, P. I., & Hartanto, R. (2018). Software Defect Prediction Using Random Forest Algorithm. *Proceeding of the 2018 12th South East Asian Technical University Consortium (SEATUC)*, 1–5. https://doi.org/10.1109/seatuc.2018.8788881

Srinivas, K., Janapati, K. C., Sai Kiran, M., Sudheer Reddy, D., Vinay, T., Vamshi, & Yashwanth, C. (2025). Deep ensemble optimal classifier-based software defect prediction for early detection and quality assurance. *Journal of Umm Al-Qura University for Engineering and Architecture*, *16*(4), 1057–1068. https://doi.org/10.1007/s43995-025-00138-9

Tantithamthavorn, C., Hassan, A. E., & Matsumoto, K. (2020). The Impact of Class Rebalancing Techniques on the Performance and Interpretation of Defect Prediction Models. *IEEE Transactions on Software Engineering*, *46*(11), 1200–1219. https://doi.org/10.1109/tse.2018.2876537

Wei, X., Yang, F., Zhong, F., & Zeng, G. (2025). A Fine-Grained Defect Prediction Method Based on Drift-Immune Graph Neural Networks. *Computers, Materials & Continua*, *82*(2), 3563–3590. https://doi.org/10.32604/cmc.2024.057697

Zhang, F., Zheng, Q., Zou, Y., & Hassan, A. E. (2016). Cross-project defect prediction using a connectivity-based unsupervised classifier. *Proceedings of the 38th International Conference on Software Engineering*, 309–320. https://doi.org/10.1145/2884781.2884839

Zhang, S., Jiang, S., & Yan, Y. (2022). A Software Defect Prediction Approach Based on BiGAN Anomaly Detection"Journal of Systems and Software. *Scientific Programming*, *5024399*, 1–13. /https://doi.org/10.1155/2022/5024399