J-INTECH (Journal of Information and Technology)
Accredited Sinta 4 Ministry of Higher Education, Science and Technology Republic of Indonesia SK No. 10/C/C3/DT.05.00/2025
E-ISSN: 2580-720X || P-ISSN: 2303-1425

Improving Software Defect Prediction Performance Using C4.5 Based Ensemble Learning with AdaBoost and Bagging Techniques

Dede Wintana[1*], Dinar Ismunandar[2], Eka Herdit Juningsih[3]

[1]Bina Sarana Informatika University, Faculty of Engineering and Informatics, Informatika Kampus Kota Sukabumi , Jl. Cemerlang No.8, Sukakarya, Warudoyong, Sukabumi, Jawa Barat, Indonesia.
[2]Bina Sarana Informatika University, Faculty of Engineering and Informatics,Teknologi Informasi, Jl.Keramat Raya 98, Senen, Jakarta Pusat, DKI Jakarta, Indonesia.
[3]Bina Sarana Informatika University, Faculty of Engineering and Informatics,Sistem Informasi, Jl.Keramat Raya 98, Senen, Jakarta Pusat, DKI Jakarta, Indonesia.

Keywords
AdaBoost; Bagging; C4.5; Ensemble; SDP.

Corresponding Author:
dede.dwe@bsi.ac.id

Abstract
Software defect prediction (SDP) plays a crucial role in improving software quality by enabling the early detection of faulty modules during the development phase. However, class imbalance within software defect datasets remains a significant challenge that adversely impacts prediction accuracy. This study aims to address this issue by implementing ensemble learning methods—specifically Bagging and AdaBoost—combined with the C4.5 decision tree algorithm to enhance classification performance. The research utilized five well-known datasets from the NASA MDP Repository (CM1, JM1, KC1, KC2, and PC1), each containing comprehensive software metrics and defect labels. The methodology involved several stages: data preprocessing (normalization and discretization), model training using 10-fold cross-validation, and performance evaluation through metrics such as accuracy and Area Under the Curve (AUC). Results indicate that both ensemble methods outperformed the standalone C4.5 algorithm across all datasets. Notably, the AdaBoost + C4.5 model yielded the highest accuracy in most scenarios, with the PC1 dataset reaching 97.20% accuracy. In comparison, C4.5 alone and C4.5 with Bagging recorded lower values, demonstrating the significant impact of adaptive weighting in AdaBoost. These findings affirm that ensemble learning, particularly AdaBoost, effectively mitigates the impact of class imbalance and improves prediction performance in SDP tasks.

1. Introduction

In software development, the occurrence of defects is inevitable, stemming from both logical and semantic errors during code implementation. These defects are predominantly attributed to human errors during the programming process. When such flawed code is executed, it may lead to system malfunctions or even total failure [1]. To reduce the cost and workload involved in the development and maintenance of software systems, it is essential to adopt proactive strategies that can anticipate potential defects at an early stage[2] One such

critical strategy is software defect prediction (SDP), which enables early detection of potentially faulty modules. This allows developers to allocate resources more effectively, ensuring the delivery of high-quality software that supports seamless business operations [3].

The primary objective of software defect prediction is to enhance software quality by minimizing the occurrence of defects during execution. By identifying high-risk modules, testing efforts can be more efficiently directed, thereby reducing testing costs and improving overall development efficiency [4].

However, one of the major challenges in implementing defect prediction is class imbalance in software metrics datasets [5] This issue arises when the number of defective modules (minority class) is significantly lower than non-defective modules (majority class), which negatively impacts model accuracy and reduces the model's ability to detect actual defective modules [6],[7] Imbalanced datasets often lead machine learning models to be biased toward the majority class, resulting in poor performance on the minority class [8]

To address this problem, two major approaches are commonly employed: data-level techniques, such as sampling, and algorithm-level techniques, such as ensemble learning [9]. This study focuses on the algorithm-level approach by enhancing learning algorithms or integrating multiple models through ensemble methods to improve classification reliability and accuracy[10] Ensemble techniques are known to outperform single classification models by aggregating multiple classifiers trained on the same data. In addition to improving accuracy, ensemble methods have proven more effective in handling imbalanced data than resampling techniques [11].

Among ensemble approaches, Bagging and Boosting are two widely used strategies. Bagging (Bootstrap Aggregating) reduces overfitting by generating multiple subsets of training data, training separate models on each subset, and combining their predictions—typically through averaging or voting—to reduce variance [12]. Bagging is particularly effective with unstable classifiers such as Decision Trees, which are sensitive to small changes in training data. Applying Bagging to the C4.5 algorithm has demonstrated improved classification accuracy compared to using a single model [13].

On the other hand, Boosting aims to minimize prediction error by combining multiple weak learners—models that perform only slightly better than random guessing—into a strong composite model[14],[15]. One of the most prominent Boosting algorithms is Adaptive Boosting (AdaBoost), a supervised learning technique commonly used in data mining to build classification models and address class imbalance problems [16],[17] AdaBoost assigns varying weights to training instances during each iteration, emphasizing those that are misclassified, thus guiding the learning process toward harder cases [18].

In addition to Bagging and AdaBoost, this study also utilizes the C4.5 decision tree algorithm as a base learner. C4.5 is a well-established classification algorithm that constructs decision trees based on entropy and information gain, offering high interpretability and support for both discrete and continuous attributes [19]. Despite its advantages, C4.5 exhibits instability under varying data conditions, which can affect its accuracy [20]. Therefore, integrating it with ensemble methods such as AdaBoost and Bagging can significantly improve its performance. Previous studies have reported that such integration leads to notable improvements in accuracy, precision, recall, and F1-score compared to using C4.5 alone [21],[22].

The dataset used in this research is sourced from the NASA MDP Repository, specifically the PROMISE datasets (CM1, JM1, KC1, KC2, and PC1), which are widely adopted in software defect prediction research due to their comprehensive software metrics and defect labels. This study aims to produce models with significantly improved accuracy and AUC (Area Under the ROC Curve) values, providing an effective solution to the class imbalance issue in the context of software defect prediction.

2. Research Method

This study on software defect prediction is quantitative in nature, as it utilizes numerical data for analysis. The research involves several stages, starting with data collection, preprocessing, data splitting, and model training

and testing. An ensemble learning approach is then applied by combining the AdaBoost and C4.5 algorithms. The final stage includes model validation and evaluation, using accuracy and AUC (Area Under the Curve) as performance metrics. The overall research process is illustrated in Figure 1 below

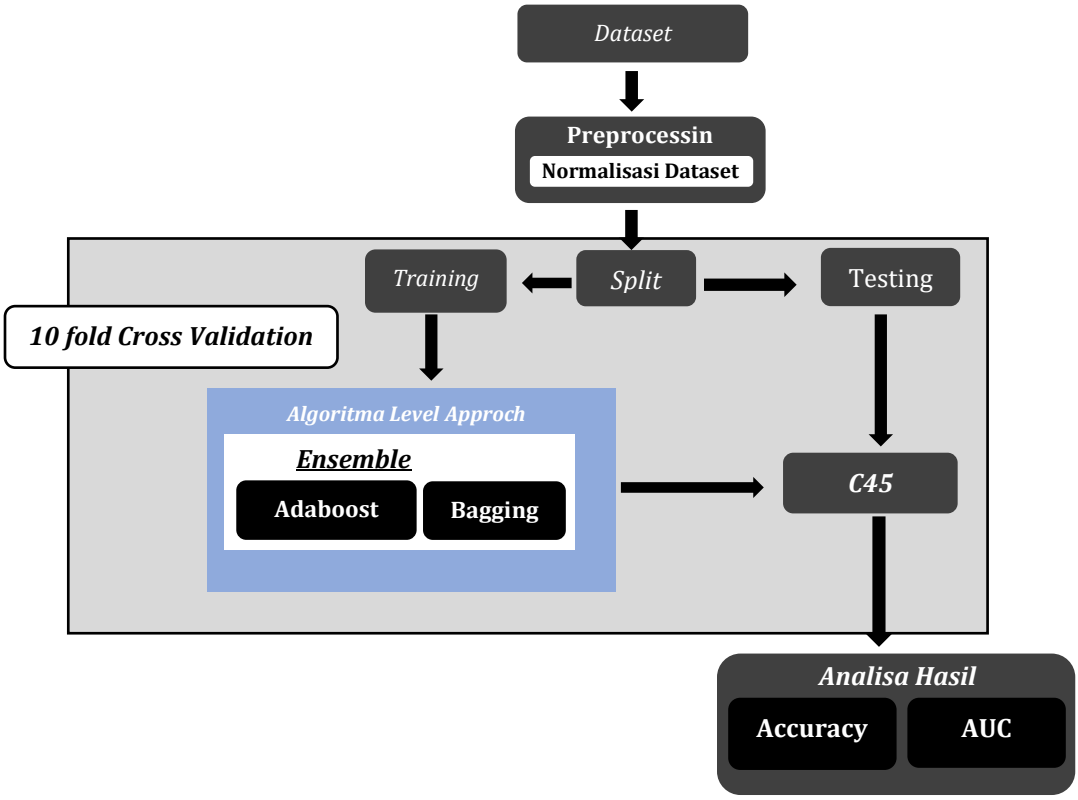


Figure 1. Step Of the art

2.1 Software Matric

In the context of software defect prediction, this study adopts a quantitative research approach, as it utilizes numerically structured datasets. The research process consists of several stages, including data collection, preprocessing, data splitting (training and testing), followed by the implementation of ensemble learning approaches using AdaBoost + C4.5 and Bagging + C4.5. The final stage involves evaluating and analyzing the performance of the resulting models by measuring their accuracy and AUC (Area Under the ROC Curve). The overall research workflow is illustrated in Figure 1 as follows.

Table 1. Atribut Dataset

No	Fitur	Code	CM1	JM1	KC1	KC2	PC1
1	Line Of Code	(loc)	v	v	v	v	v
2	cyclomatic compelexity	(v(g))	v	v	v	v	v
3	essensial complexity	(ev(g))	v	v	v	v	v
4	Design Complexity	(iv(g))	v	v	v	v	v
5	Unique Operators	(uniq_Op)	v	v	v	v	v
6	Unique Operands	(uniq_Opnd)	v	v	v	v	v
7	Total Operators	(total_Op)	v	v	v	v	v
8	Total Operands	(TotalOpnd)	v	v	v	v	v

No	Fitur	Code	CM1	JM1	KC1	KC2	PC1
9	Total Operators dan Operands	(n)	v	v	v	v	v
10	volume	(v)	v	v	v	v	v
11	program legth	(l)	v	v	v	v	v
12	Diviculity	(d)	v	v	v	v	v
13	intelegence	(I)	v	v	v	v	v
14	Time to write program	(t)	v	v	v	v	v
15	Effort to Write Program	(e)	v	v	v	v	v
16	Errort Estimate	(b)	v	v	v	v	v
17	Count of Statement Lines	(lOCode)	v	v	v	v	v
18	Count of Code and Comments Lines	(locCodeAndComment)	v	v	v	v	v
19	Count of Blank Lines	(lOBlank)	v	v	v	v	v
20	Count of Lines of Comments	(lOComment)	v	v	v	v	v
21	Metrik Branch Count	(branchCount)	v	v	v	v	v

2.2 Preprocessing

At this stage, normalization is applied to the dataset. The method used transforms numerical data into categorical data by dividing the value range of an attribute into several intervals. This process reduces the number of attribute values, making the data easier to analyze [22]

2.3 Split Training and Testing

Data splitting is a technique used to divide a dataset into several parts, such as training data and testing data, and is one of the key factors that influence the performance of classification models in machine learning algorithms, The proportion of the split between training and testing data may vary depending on the characteristics of each dataset used [23]. Common techniques for dividing datasets include *holdout validation* and *k-fold cross validation*. The validation process is essential to ensure that each data instance has a fair chance of being used for both training and testing purposes, resulting in a more objective and comprehensive model evaluation.

2.4 Algoritm c45

The C4.5 algorithm is one of the classification methods used to convert large datasets into a decision tree that represents a set of classification rules. This algorithm is an extension of the ID3 algorithm and was introduced by J. Ross Quinlan [24]. C4.5 offers several advantages, including its ability to handle missing values, process continuous (numeric) data, and perform pruning to simplify the decision tree. Additional strengths of C4.5 include its capability to manage both numerical and categorical attributes, the interpretability of the resulting model, and its relatively fast tree construction process compared to other algorithms [25][26]

The decision tree construction using the C4.5 algorithm follows a systematic series of steps, including selecting the best attribute based on the gain ratio, splitting the data into tree branches according to the selected attribute, and recursively building the tree structure until the stopping condition is met.

1. The attributes in the dataset are classified according to their respective target classes to be analyzed during the root selection process of the decision tree.
2. The attribute with the highest gain value or the lowest entropy, as calculated using Equations (1) and (2), is selected as the root node, as it is considered the most informative for the data splitting process.

$$Entropy\ (s) = \sum_{i=1}^{n} - pi\ log^2 pi$$

$$Gain\ (S, A) = entropy(S) - \sum_{i=1}^{n} \frac{|si|}{s}\ x\ Entropy\ (s)$$

3. The procedure of selecting the attribute with the highest gain value or the lowest entropy is repeated recursively for each branch of the tree until all nodes have an appropriate splitting attribute.
4. The decision tree construction process is terminated when all leaf nodes contain data belonging to a homogeneous class, or when no remaining attributes are available for further splitting.

2.5 Adaboost + C45 Model Evaluation

In the first experimental scenario, the C4.5 algorithm was implemented in combination with the AdaBoost ensemble method. The model training was carried out using the Weka application. AdaBoost plays a key role in improving classification performance by assigning different weights to each training instance. These weights are adjusted iteratively based on the classification error rate, following the equations presented below.

Input

$Training\ data\ D = \{(x1, y1) \ldots\ldots (xm, ym)$
$Base\ Classifier$= L
$Number\ of\ iteration = T$

Proces

1. Initialize sample wights
   $D_t(i) = 1/m$
   $For\ i\ =\ 1,2,3 \ldots\ldots\ldots, m$
   $For\ \ t = 1,2,3, \ldots\ldots\ldots T$
2. train the base classifier (L), ht using the weighted training data distribution
   $D_t H_t\ =\ L(D_t)$
3. Compute the error of the classifier :
   $\in_t = \sum D_t\ \ (i)$
4. Where is the indicator function (1 if true, 0 otherwise)compute the classifier weight :
   $\alpha t = \frac{1}{2}\ In(\frac{1-\in t}{\in t})$
5. Update the sample weights :
   $D_{t+1}(i) = \frac{Dt(i)}{Zt}\ X\ \{-\alpha t\ Atau^{\alpha t}\}$

Output

The result of the final classifier is calculated using the following equation:

$$H(x)\ =\ sign\ (h \sum_{t\ =\ 1}^{T}\ \alpha t_{ht}(X))$$

2.6 Bagging + C45 Model Evaluation

The classification model training in this study was conducted by combining the C4.5 algorithm with the Bagging ensemble method. This process involves several systematic steps, in which Bagging generates multiple learning models using bootstrap sampling to create diverse subsets of the training data. Each model is trained independently, and the final classification result is obtained by aggregating the predictions of all models using a majority voting approach. The steps of the Bagging algorithm in the classification process are described by the following equations:

Input

$Training\ data\ D = \{(x1, y1) \ldots\ldots (xm, ym)$

$Base\ Classifier$= L
$Number\ of\ iteration = T$

Proses

1. Randomly generate training samples from the original dataset using the bootstrap sampling technique, where each subset may contain duplicated instances.
2. Train a base classifier $D_t H_t = L(D_t)$, where LLL is the learning function applied to the bootstrapped training dataset D_t
3. Combine the predictions from all trained models by applying a majority voting function to determine the final output class.

Output

The final classifier result is calculated using the following equation:

$$H(X) = arg\ max_y \sum_{t-1}^{T} 1\,(y = h_t(X))$$

2.7 Model Evaluation

The performance evaluation of the classification model in this study was conducted using a Confusion Matrix, which consists of four key components: true positive (TP), true negative (TN), false positive (FP), and false negative (FN). Based on this matrix, several evaluation metrics such as precision, sensitivity (recall), specificity, and F-score can be calculated to assess the effectiveness of the model.

In addition, to obtain a more reliable accuracy measurement and to avoid training data bias, the k-fold cross-validation technique was employed with k=10k = 10k=10. In this approach, the dataset is randomly divided into 10 equal subsets, and the training and testing processes are repeated 10 times, each with a different combination of training and test data. The final accuracy is calculated by averaging the accuracy values across all iterations, resulting in a more stable and representative estimate of the model's performance.

2.8 Result Analysis

The final stage of this study involves the analysis phase, which aims to compare the results obtained from each experimental scenario described in the model training section. The objective is to identify the scenario that yields the best classification performance. The comparison is based on several key evaluation metrics, namely precision, specificity, sensitivity (recall), F1-score, and accuracy, all of which are derived from the classification results of each model scenario.

3. Result and Discussions

3.1 Dataset

This study utilizes datasets obtained from the NASA Metrics Data Program (MDP), comprising five datasets: CM1, JM1, KC1, KC2, and PC1. These datasets were developed by the United States National Aeronautics and Space Administration (NASA) and are widely used in research related to defects in both hardware and software systems. The NASA datasets are publicly available through the PROMISE repository and the official MDP website.

In this study, the C4.5 algorithm is implemented as the base classifier to evaluate the effectiveness of ensemble learning approaches, specifically using the AdaBoost and Bagging methods. The evaluation of the model focuses on measuring classification accuracy, with the WEKA application used as a tool for data processing and analysis.

The detailed characteristics of each NASA MDP dataset used in this study are presented in the following table.

Table 2. Dataset

loc	v(g)	ev(g)	iv(g)	n	v	l	d	i	e	…	Defect
tr1.1	1.4	1.4	1.4	1.3	1.3	1.3	1.3	1.3	1.3	…	False
1	1.0	1	1	1	1	1	1	1	1	…	true
415	59.0	50	51	1159	8411.31	0.01	103.53	81.24	870848.58	…	true
230	33.0	10	16	575	3732.82	0.03	39.82	93.74	148644.06	…	true
175	26.0	12	13	500	3123.96	0.03	29.48	105.96	92103.07	…	true
163	16.0	13	11	440	2714.77	0.03	32.25	84.14	87589.65	…	true
152	11.0	6	11	432	2629.78	0.03	31.68	83.01	83311.56	…	true
3	1.0	1	1	1	0	0	0	0	0	…	False
14	2.0	1	2	22	88	0.17	5.79	15.21	509.14	…	False
…	…	…	…	…	…	…	…	…	…	…	…

3.2 Pengujian Model

The model evaluation in this study was conducted using the K-Fold Cross-Validation approach, with performance measured through the Area Under the Curve (AUC) metric. No explicit split between training and testing data was performed at the outset, as the entire dataset was analyzed using a 10-fold cross-validation scheme. In this process, the data is evenly divided into ten subsets. The model is trained and tested over 10 iterations, where in each iteration, nine subsets are used for training and one subset is used for testing. This procedure continues until every subset has been used exactly once as a test set. The final performance result is obtained by averaging the evaluation metrics across all iterations. The AUC metric is used to assess the model's ability to distinguish between positive and negative classes comprehensively.

Prior to the training phase, the dataset was imported into the WEKA application for preprocessing. During this stage, a filtering process was carried out using the resample method to eliminate instances considered as noise. The objective of this step is to enhance the quality of the training data, thereby allowing the classification model to achieve better performance and generalizability.

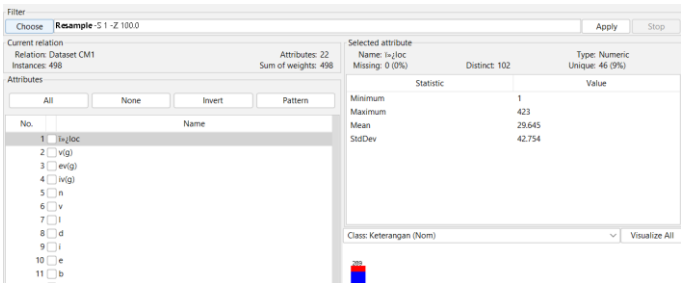


Figure 2 Praprocesing (resempling data)

The next step involves the discretization process, which is one of the techniques used during the preprocessing stage. Discretization aims to simplify numerical attributes by converting them into categorical attributes. This process is performed by dividing the range of numeric values into several discrete intervals, allowing each numeric value to be classified into a predefined category. This approach not only helps reduce data complexity

but also enhances the effectiveness of classification algorithms, which tend to perform more optimally when working with categorical data.

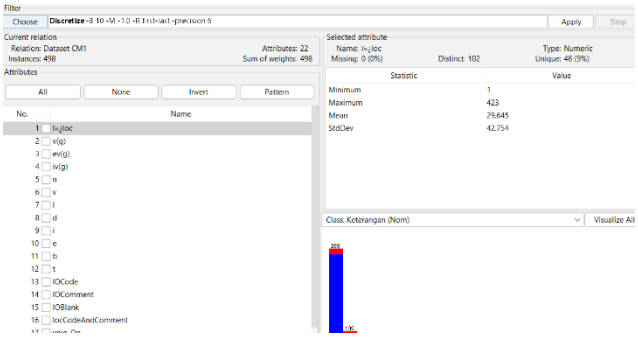


Figure 3. PraProcesing data (Diskrit Datset)

After completing the discretization stage, classification model testing was conducted using several approaches. First, the model was evaluated using the standalone C4.5 algorithm as a baseline. Subsequently, testing was carried out using the Bagging method with C4.5 as the base classifier to assess performance improvements through ensemble techniques. Finally, the model was also tested using the AdaBoost approach combined with the C4.5 algorithm to evaluate the effectiveness of adaptive weighting in improving classification outcomes.

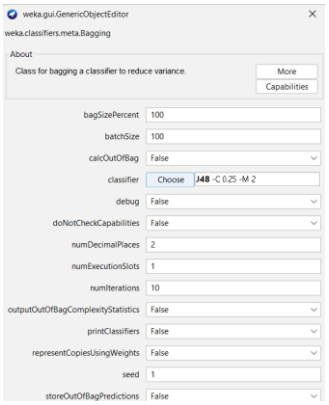


Figure 5. Evaluation Model Bagging + C45

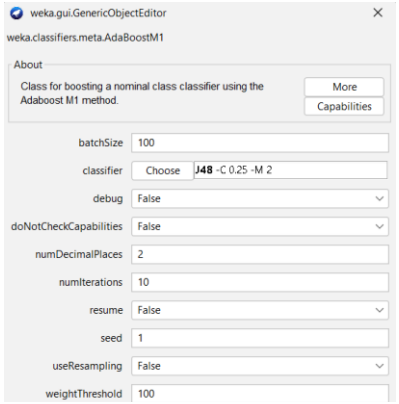
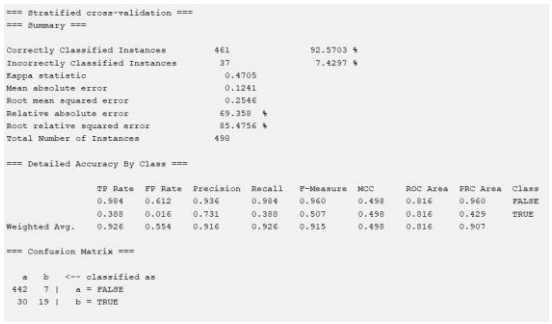


Figure 4. Evaluation Model AdaBoost + C45

Using the same approach, namely K-Fold Cross Validation, the evaluation results of the models were obtained based on the accuracy values from each testing scenario. The detailed accuracy results obtained from these experiments are presented as follows:

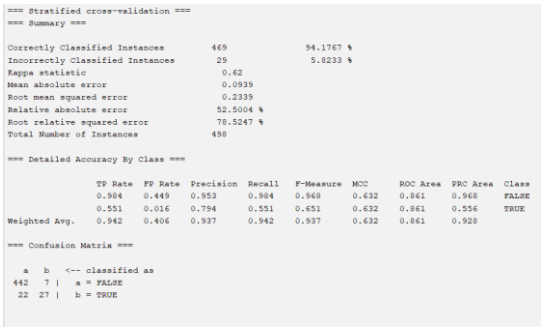
Table 3. Accuracy Results of Model Evaluation

Dataset	Test method Comparasion		
	C45	C45 + Bagging	C45+AdaBoost
CM1	91,7671	92,5703	94,1767
JM1	83,079	87,1073	88,7006
KC1	85,5092	87,0555	87,6719
KC2	90,8046	90,9962	91,76265
PC1	94,6799	95,8521	97,2047



```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         461               92.5703 %
Incorrectly Classified Instances        37                7.4297 %
Kappa statistic                          0.4705
Mean absolute error                      0.1241
Root mean squared error                  0.2546
Relative absolute error                 69.358  %
Root relative squared error             85.4756 %
Total Number of Instances              498

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   MCC      ROC Area  PRC Area  Class
                 0.984     0.612     0.936      0.984     0.960      0.498     0.816     0.960     FALSE
                 0.388     0.016     0.731      0.388     0.507      0.498     0.816     0.429     TRUE
Weighted Avg.    0.926     0.554     0.916      0.926     0.915      0.498     0.816     0.907

=== Confusion Matrix ===

   a   b   <-- classified as
 442   7 |   a = FALSE
  30  19 |   b = TRUE
```

Figure 6. Model Accuracy Results for C4.5 with Bagging



```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         469               94.1767 %
Incorrectly Classified Instances        29                5.8233 %
Kappa statistic                          0.62
Mean absolute error                      0.0939
Root mean squared error                  0.2339
Relative absolute error                 52.5004 %
Root relative squared error             78.5247 %
Total Number of Instances              498

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   MCC      ROC Area  PRC Area  Class
                 0.984     0.449     0.953      0.984     0.968      0.632     0.861     0.968     FALSE
                 0.551     0.016     0.794      0.551     0.651      0.632     0.861     0.556     TRUE
Weighted Avg.    0.942     0.406     0.937      0.942     0.937      0.632     0.861     0.928

=== Confusion Matrix ===

   a   b   <-- classified as
 442   7 |   a = FALSE
  22  27 |   b = TRUE
```

Figure 7. Model Accuracy Results for C4.5 with AdaBoost

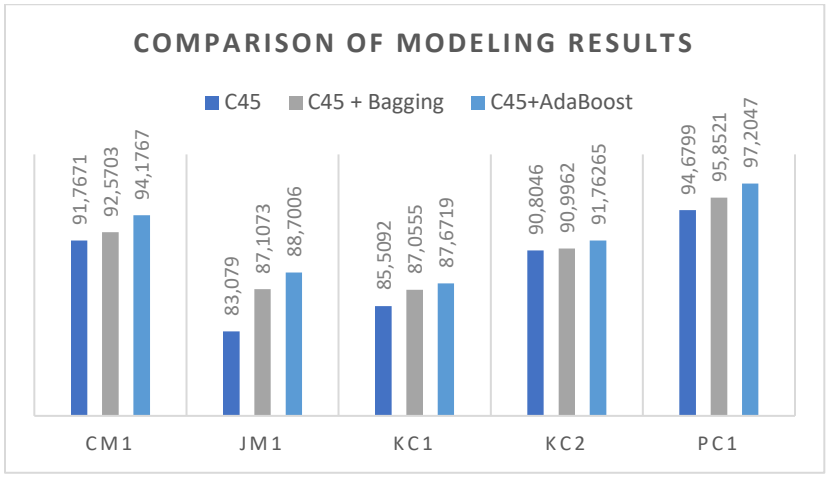


Figure 8. Comparison of modeling results

The experimental results across five datasets demonstrate that the implementation of *Ensemble Learning* methods significantly improves classification accuracy compared to the standalone C4.5 algorithm. On the CM1 dataset, both C4.5 and C4.5 + Bagging achieved an accuracy of 92,5703%, while C4.5 + AdaBoost increased the accuracy to 94,1767%. For the JM1 dataset, accuracy improved from 83.08% (C4.5) to 87.11% (Bagging) and further to 88.70% (AdaBoost). On the KC1 dataset, the accuracy rose from 85.51% (C4.5) to 87.06% (Bagging) and reached 87.67% (AdaBoost). In the KC2 dataset, the accuracy slightly increased from 90.80% (C4.5) to 91.00% (Bagging) and to 91.76% (AdaBoost). Finally, on the PC1 dataset, C4.5 achieved 94.68%, which improved to 95.85% with Bagging and reached the highest value of 97.20% with AdaBoost. Overall, the combination of C4.5 with AdaBoost consistently yielded the highest accuracy across most datasets evaluated.

4. Conclusions and Future Works

Based on the experimental results, it can be concluded that the application of Ensemble Learning methods has a significant impact on improving classification accuracy. For the CM1 dataset, the C4.5 algorithm achieved an accuracy of 94,17%. When combined with the *Bagging* technique, the accuracy remained at 92,57%, while the use of *AdaBoost* increased the accuracy to 93.97%. In the case of the JM1 dataset, C4.5 alone achieved an accuracy of 83.08%. This improved to 87.10% with *C4.5 + Bagging*, and further increased to 88.70% using *C4.5 + AdaBoost*. For the KC1 dataset, C4.5 produced an accuracy of 85.5%, which rose to 87.05% when combined with *Bagging*, and reached 87.67% with *AdaBoost*. On the KC2 dataset, the C4.5 algorithm alone achieved 90.8%, improved slightly to 90.99% with *Bagging*, and further to 91.76% with *AdaBoost*. Finally, for the PC1 dataset, the C4.5 method yielded an accuracy of 91.67%, increased to 95.85% with *Bagging*, and reached the highest accuracy of 97.2% when using *C4.5 + AdaBoost*.

Overall, these findings indicate that the use of Ensemble Learning techniques, particularly *Bagging* and *AdaBoost*, significantly enhances model performance compared to the standalone C4.5 algorithm. For future research, it is recommended to explore other classification methods such as Random Forest, Naïve Bayes, Support Vector Machine (SVM), or other suitable algorithms. Furthermore, combining techniques at the data level, such as resampling, with algorithm-level approaches may offer promising directions to further enhance model performance.

5. References

[1] A. Hardoni, D. P. Rini, and S. Sukemi, "Integrasi SMOTE pada Naive Bayes dan Logistic Regression Berbasis Particle Swarm Optimization untuk Prediksi Cacat Perangkat Lunak," *J. Media Inform. Budidarma*, vol. 5, no. 1, p. 233, 2021, doi: 10.30865/mib.v5i1.2616.

[2] E. Hari Agus Prastyo, S. Suhartono, M. Faisal, M. A. Yaqin, and R. A. J. Firdaus, "Naive Bayes Classification Untuk Prediksi Cacat Perangkat Lunak," *JIPI (Jurnal Ilm. Penelit. dan Pembelajaran Inform.*, vol. 9, no. 2, pp. 782–791, 2024, doi: 10.29100/jipi.v9i2.5508.

[3] D. Wintana, G. Gunawan, H. Sulaeman, and S. Bahri, "Penerapan Multi-Layer Perceptron dan Diskrit pada Prediksi Cacat Software," *J-Intech*, vol. 12, no. 02, pp. 321–329, 2024, doi: 10.32664/j-intech.v12i02.1422.

[4] E. A. Kusnanti, L. D. F. Vantie, and U. L. Yuhana, "Software Defect Prediction Using Pca Based Recurrent Neural Network," *JUTI J. Ilm. Teknol. Inf.*, pp. 23–31, 2024, doi: 10.12962/j24068535.v22i1.a1199.

[5] C. G. ; Amanda, "Optimasi Multi-Objektif Prediksi Cacat Perangkat Lunak Melalui Integrasi Nsga-Ii Berbasis Pymoo," UIN Syarif Hidayatullah Jakarta, 2025.

[6] M. Salsabila, "Pendekatan visual analytics dalam pemodelan prediksi cacat perangkat lunak menggunakan kombinasi pca dan smote," UIN Syarif Hidayatullah Jakarta, 2022.

[7] N. Ichsan *et al.*, "Prediksi Cacat Software Menggunakan Class Balancer Bagging C4 . 5 dan Analisis Statistik SPSS dalam Konteks Akuntansi," vol. 5, no. 1, pp. 47–54, 2025.

[8] D. Pramadhana, "Klasifikasi Penyakit Diabetes Menggunakan Metode CFS dan ROS dengan Algoritma J48 Berbasis Adaboost," *Edumatic J. Pendidik. Inform.*, vol. 5, no. 1, pp. 89–98, 2021, doi: 10.29408/edumatic.v5i1.3336.

[9] N. Ichsan, R. Sopandi, H. Priyandaru, and M. Tabrani, "Pendekatan Level Data Smote Pada Algoritma Bagging C4.5 Untuk Prediksi Cacat Software Smote Data Level Approach of C4.5 Bagging Algorithm for Software Defect Prediction," *CerminJurnal Penelit.*, vol. 7, pp. 402–416, 2023.

[10] R. Sabaruddin, S. Murni, and W. Nugraha, "Pemanfaatan Resampling Untuk Penanganan Ketidakseimbangan Kelas Pada Prediksi Cacat Software Berbasis C5.0," *J. Teknol. Inf. Mura*, vol. 15, no. 1, pp. 14–23, 2023, doi: 10.32767/jti.v15i1.1956.

[11] N. Wulandari and B. Badieah, "Implementasi Teknik Resampling Untuk Mengatasi Ketidakseimbangan Data Terhadap Klasifikasi Anemia Menggunakan Support Vector Machine," *J. Rekayasa Sist. Inf. dan Teknol.*, vol. 2, no. 3, pp. 942–951, 2025, doi: 10.70248/jrsit.v2i3.1856.

[12] E. R. Putri and D. B. Arianto, "Perbandingan Performa Algoritma Metode Bagging dan Boosting pada Prediksi Konsentrasi PM10 di Jakarta Utara," *J. Nas. Teknol. dan Sist. Inf.*, vol. 10, no. 1, pp. 72–81, 2024, doi: 10.25077/teknosi.v10i1.2024.72-81.

[13] N. D. Saputri, K. Khalid, and D. Rolliawati, "Komparasi penerapan metode Bagging dan Adaboost pada Algoritma C4. 5 untuk prediksi Penyakit Stroke," *Sist. J. Sist. Inf.*, vol. 11, no. 3, pp. 567–577, 2022, [Online]. Available: http://sistemasi.ftik.unisi.ac.id.

[14] A. H. Marsuhandi, A. M. Soleh, H. Wijayanto, and D. D. Domiri, "Pemanfaatan Ensemble Learning Dan Penginderaan Jauh Untuk Pengklasifikasian Jenis Lahan Padi," *Semin. Nas. Off. Stat.*, vol. 2019, no. 1, pp.

188–195, 2020, doi: 10.34123/semnasoffstat.v2019i1.247.

[15] S. Sidiq, P. Korespondensi, and N. Shobi Mabrur, "Pengembangan Model Prediksi Risiko Diabetes Menggunakan Pendekatan AdaBoost dan Teknik Oversampling SMOTE," *J. Ilm. Inform. Dan Ilmu Komput.*, vol. 4, pp. 13–23, 2025, [Online]. Available: https://doi.org/10.58602/jima-ilkom.v4i1.41.

[16] P. Setiyadi, M. N. Prayogi, and A. Solichin, "Optimalisasi Prediksi Kehilangan Karyawan Menggunakan Teknik Rfe, Smote, Dan Adaboost," *JIPI (Jurnal Ilm. Penelit. dan Pembelajaran Inform.*, vol. 9, no. 4, pp. 2131–2145, 2024, doi: 10.29100/jipi.v9i4.5642.

[17] N. Purnama and N. W. Utami, "Penerapan Algoritma Adaboost Untuk Optimasi Prediksi Kunjungan Wisatawan Ke Bali Dengan Metode Decission Tree," *JUSIM (Jurnal Sist. Inf. Musirawas)*, vol. 8, no. 2, pp. 119–126, 2023, doi: 10.32767/jusim.v8i2.2197.

[18] Y. Crismayella, N. Satyahadewi, and H. Perdana, "Algoritma Adaboost pada Metode Decision Tree untuk Klasifikasi Kelulusan Mahasiswa," *Jambura J. Math.*, vol. 5, no. 2, pp. 278–288, 2023, doi: 10.34312/jjom.v5i2.18790.

[19] Y. I. Lestari, S. Defit, and Y. Yuhandri, "Prediksi Tingkat Kepuasan Pelayanan Online Menggunakan Metode Algoritma C.45," *J. Inform. Ekon. Bisnis*, vol. 3, pp. 148–154, 2021, doi: 10.37034/infeb.v3i4.104.

[20] M. A. U. Yosef Mulyanto Dawa, Abdul Aziz, "OPTIMASI ALGORITMA C4.5 BERBASIS PARTICLE SWARM OPTIMIZATION (PSO)UNTUK MENENTUKAN WHOLESALES PENJUALAN Yosef," *J. Ris. Mhs. Bid. Teknol. Inf.*, vol. 6, pp. 21–26, 2023, [Online]. Available: https://ejournal.unikama.ac.id/index.php/JFTI.

[21] M. Agusviyanda; Novita, Rita; Saleh, Alfa; Jamaris, "Peningkatan Algoritma C4.5 Menggunakan Ensemble Learning Untuk Mendeteksi Penyakit Ginjal," *INFORMATIKA*, vol. 13, no. 1, pp. 670–680, 2025, doi: https://doi.org/10.36987/informatika.v12i3.7542.

[22] M. M. Kholil, F. Alzami, and M. A. Soeleman, "AdaBoost Based C4.5 Accuracy Improvement on Credit Customer Classification," in *2022 International Seminar on Application for Technology of Information and Communication (iSemantic)*, Sep. 2022, pp. 351–356, doi: 10.1109/iSemantic55962.2022.9920463.

[23] I. K. Nuraini, "Menggunakan Cost Sensitive Learning dan Neural Network Optimasi Imbalance Class Pada Prediksi Cacat Perangkat Lunak," *Repository.Uinjkt.Ac.Id*, 2023, [Online]. Available: https://repository.uinjkt.ac.id/dspace/handle/123456789/70877%0Ahttps://repository.uinjkt.ac.id/dspace/bitstream/123456789/70877/1/ISLAH KHOFIFAH NURAINI-FST.pdf.

[24] M. Adriansa, L. Yulianti, and L. Elfianty, "Analisis Kepuasan Pelanggan Menggunakan Algoritma C4.5," *J. Tek. Inform. UNIKA St. Thomas*, vol. 07, no. 21, pp. 115–121, 2022, doi: 10.54367/jtiust.v7i1.1983.

[25] S. Umam and F. W. Christanto, "Algoritma C4.5 Pada Sistem Analisis Data Untuk Klasifikasi Nasabah Sebagai Dasar Promosi Penjualan Produk Asuransi," *J. Tek. Inform. dan Sist. Inf.*, vol. 10, no. 1, pp. 875–884, 2023, [Online]. Available: http://jurnal.mdp.ac.id.

[26] B. Basiroh and H. Irjananto, "C4.5 Algorithm As a Decision Support System for Social Welfare Aid Recipients," *J. Teknol. Inf. dan Komun.*, vol. 12, no. 1, p. 9, 2024, doi: 10.30646/tikomsin.v12i1.816.