Building of Informatics, Technology and Science (BITS)
Volume 8, No 1, June 2026 Page: 615−628
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v8i1.10268

Evaluasi Komparatif Algoritma Decision Tree, Random Forest, dan XGBoost untuk Software Defect Prediction Menggunakan Dataset NASA Software Metrics

Okky Prasetia*, Syaeful Machfud

Fakultas Ilmu Komputer, Program Studi Teknik Informatika, Universitas Pamulang, Kota Tangerang Selatan, Indonesia
Email: 1,*dosen02837@unpam.ac.id , 2dosen02836@unpam.ac.id
Email Penulis Korespondensi: dosen02837@unpam.ac.id
Submitted: **11/06/2026**; Accepted: **30/06/2026**; Published**: 30/06/2026**

Abstrak−Software Defect Prediction (SDP) menjadi salah satu pendekatan penting untuk mendeteksi modul perangkat lunak yang berpotensi mengalami cacat sejak tahap awal pengembangan. Namun, performa algoritma prediksi masih dipengaruhi oleh karakteristik dataset sehingga belum terdapat algoritma yang secara konsisten memberikan hasil terbaik pada berbagai kasus. Selain itu, penggunaan dataset sintetis dalam penelitian SDP masih memerlukan validasi empiris untuk memastikan bahwa karakteristik data tersebut tetap representatif terhadap dataset referensi seperti NASA Metric Data Program (NASA MDP). Oleh karena itu, penelitian ini bertujuan membandingkan kinerja algoritma Decision Tree, Random Forest, dan XGBoost menggunakan dataset Playground Series Season 3 Episode 23, yaitu dataset sintetis yang dibangun berdasarkan karakteristik NASA MDP. Dataset diproses melalui tahapan preprocessing yang meliputi penanganan missing value, label encoding, dan standardisasi data, kemudian dievaluasi menggunakan 10-fold stratified cross-validation dengan metrik Accuracy, F1-Score, dan AUC-ROC. Hasil penelitian menunjukkan bahwa metode berbasis ensemble learning memberikan performa yang lebih baik dibandingkan single classifier. Random Forest memperoleh nilai Accuracy tertinggi sebesar 0,8144, sedangkan XGBoost menghasilkan nilai AUC-ROC tertinggi sebesar 0,7929, yang menunjukkan kemampuan diskriminasi terbaik dalam membedakan modul defective dan non-defective. Selain itu, analisis feature importance mengidentifikasi Lines of Code (LOC), Cyclomatic Complexity, Halstead Volume, IOCode, dan branchCount sebagai faktor yang paling berpengaruh terhadap prediksi cacat perangkat lunak. Berdasarkan hasil eksperimen pada dataset yang digunakan, pendekatan ensemble learning menunjukkan performa yang kompetitif sehingga dapat dipertimbangkan sebagai salah satu alternatif dalam pengembangan model Software Defect Prediction. Pemilihan algoritma tetap perlu disesuaikan dengan karakteristik dataset dan kebutuhan implementasi.

Kata Kunci: Software Defect Prediction; Decision Tree; Random Forest; XGBoost; Machine Learning.

Abstract−Software Defect Prediction (SDP) has become an important approach for identifying software modules that are likely to contain defects during the early stages of software development. However, the performance of prediction algorithms remains highly dependent on dataset characteristics, and no single algorithm has consistently demonstrated superior performance across different datasets. In addition, the use of synthetic datasets in SDP research still requires empirical validation to ensure that their characteristics remain representative of benchmark datasets such as the NASA Metric Data Program (NASA MDP). Therefore, this study aims to compare the performance of Decision Tree, Random Forest, and XGBoost using the Playground Series Season 3 Episode 23 dataset, a synthetic dataset developed based on the characteristics of the NASA MDP dataset. Prior to model training, the dataset underwent preprocessing, including missing value imputation, label encoding, and feature standardization. Model performance was evaluated using 10-fold stratified cross-validation with Accuracy, F1-Score, and AUC-ROC as the primary evaluation metrics. The experimental results indicate that ensemble learning methods achieved competitive performance compared with the single-classifier approach. Random Forest achieved the highest Accuracy of 0.8144, while XGBoost obtained the highest AUC-ROC score of 0.7929, indicating strong capability in distinguishing between defective and non-defective software modules on the evaluated dataset. Furthermore, feature importance analysis identified Lines of Code (LOC), Cyclomatic Complexity, Halstead Volume, IOCode, and branchCount as the most influential factors affecting software defect prediction. Based on the experimental results obtained from the selected dataset, the ensemble learning approach demonstrated competitive predictive performance and may be considered a promising alternative for developing Software Defect Prediction models. Nevertheless, the selection of the most appropriate algorithm should remain dependent on dataset characteristics and specific implementation requirements.

Keywords: Software Defect Prediction; Decision Tree; Random Forest; XGBoost; Machine Learning.

1. PENDAHULUAN

Software telah menjadi komponen fundamental dalam mendukung berbagai aktivitas di era digital, mulai dari sektor perbankan, layanan kesehatan, transportasi, industri manufaktur, hingga sistem pertahanan. Keandalan dan kualitas *software* menjadi faktor yang sangat penting karena kesalahan atau kegagalan sistem dapat menimbulkan kerugian finansial, gangguan operasional, bahkan membahayakan keselamatan pengguna. Oleh karena itu, pengembangan *software* yang berkualitas tinggi merupakan kebutuhan utama untuk menjamin aspek keandalan (*reliability*), keamanan (*security*), dan efisiensi operasional suatu sistem [1].

Meskipun berbagai metodologi pengembangan *software* terus mengalami perkembangan, kesalahan manusia (*human error*) selama proses analisis, perancangan, implementasi, maupun pemeliharaan masih menjadi penyebab utama munculnya cacat (*software defects*) pada kode program. Cacat yang tidak teridentifikasi sejak tahap awal pengembangan dapat terbawa hingga tahap pengujian maupun setelah sistem diimplementasikan, sehingga menyebabkan peningkatan biaya perbaikan, keterlambatan proses rilis, penurunan kualitas layanan, bahkan kegagalan sistem secara menyeluruh [2].

Building of Informatics, Technology and Science (BITS)
Volume 8, No 1, June 2026 Page: 615−628
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v8i1.10268

Berbagai penelitian menunjukkan bahwa biaya untuk memperbaiki cacat yang ditemukan pada fase pengujian atau setelah *software* dirilis dapat mencapai 10 hingga 100 kali lebih besar dibandingkan apabila cacat tersebut berhasil dideteksi pada tahap analisis kebutuhan (*requirement analysis*) atau perancangan sistem (*design phase*) [3]. Kondisi tersebut menunjukkan pentingnya penerapan mekanisme identifikasi dini terhadap modul-modul *software* yang berpotensi mengalami cacat. Dengan melakukan prediksi sejak awal, tim pengembang dapat memusatkan proses verifikasi dan pengujian pada modul yang memiliki tingkat risiko tinggi sehingga penggunaan sumber daya menjadi lebih efektif, kualitas *software* meningkat, dan biaya pengembangan dapat ditekan [4].

Dalam dua dekade terakhir, *Software Defect Prediction* (SDP) berkembang menjadi salah satu topik penelitian yang penting dalam bidang *Software Engineering*. SDP bertujuan membangun model prediksi yang mampu mengidentifikasi apakah suatu modul *software* termasuk kategori *defective* atau *non-defective* berdasarkan berbagai metrik yang diperoleh dari kode sumber, kompleksitas program, maupun riwayat pengembangan *software* [5]. Seiring berkembangnya teknologi kecerdasan buatan, pendekatan berbasis *Machine Learning* (ML) menjadi solusi yang banyak digunakan karena kemampuannya dalam mempelajari pola kompleks dari data berdimensi tinggi secara otomatis tanpa memerlukan aturan eksplisit [6].

Beragam algoritma *Machine Learning* telah diterapkan untuk menyelesaikan permasalahan SDP, di antaranya Naïve Bayes, k-Nearest Neighbor, Support Vector Machine, Decision Tree, Random Forest, Artificial Neural Network, Extreme Learning Machine, hingga Extreme Gradient Boosting (XGBoost).[7]Meskipun demikian, belum terdapat satu algoritma yang mampu memberikan performa terbaik secara konsisten pada seluruh dataset. Perbedaan karakteristik proyek *software*, seperti ukuran sistem, bahasa pemrograman, kompleksitas kode, dan domain aplikasi, menyebabkan performa setiap algoritma sangat bergantung pada karakteristik data yang digunakan. Fenomena ini sejalan dengan konsep *No Free Lunch Theorem*, yang menyatakan bahwa tidak ada algoritma pembelajaran yang unggul untuk semua jenis permasalahan [8].

Penelitian-penelitian terbaru menunjukkan bahwa pendekatan *ensemble learning*, khususnya Random Forest dan XGBoost, memiliki kemampuan prediksi yang lebih baik dibandingkan algoritma klasifikasi tunggal karena mampu mengurangi variansi dan meningkatkan kemampuan generalisasi model [9]. Namun demikian, kajian komparatif yang mengevaluasi performa Decision Tree, Random Forest, dan XGBoost pada dataset standar prediksi cacat *software* masih relatif terbatas. Penelitian semacam ini diperlukan untuk memberikan rekomendasi empiris mengenai algoritma yang paling sesuai digunakan dalam proses prediksi cacat *software* pada berbagai karakteristik data [10]. Selain pemilihan algoritma, tahap *data preprocessing* juga memiliki peran yang sangat penting dalam meningkatkan kualitas model prediksi. Permasalahan yang umum dijumpai pada dataset SDP meliputi keberadaan *missing values*, fitur kategorikal yang memerlukan proses transformasi ke bentuk numerik, serta ketidakseimbangan kelas (*class imbalance*) akibat jumlah modul *non-defective* yang jauh lebih banyak dibandingkan modul *defective* . Penanganan yang tepat terhadap permasalahan tersebut dapat meningkatkan kemampuan model dalam mengenali pola cacat dan menghasilkan prediksi yang lebih akurat.

Berdasarkan penelitian-penelitian tersebut, masih terdapat beberapa keterbatasan. Sebagian besar penelitian hanya berfokus pada satu atau dua algoritma klasifikasi sehingga belum memberikan perbandingan yang komprehensif antara metode *single classifier* dan *ensemble learning*. Selain itu, penggunaan dataset Playground Series Season 3 Episode 23 yang merepresentasikan karakteristik NASA MDP masih relatif terbatas. Penelitian terdahulu juga umumnya menggunakan metrik evaluasi yang terbatas sehingga belum memberikan analisis performa model secara menyeluruh.

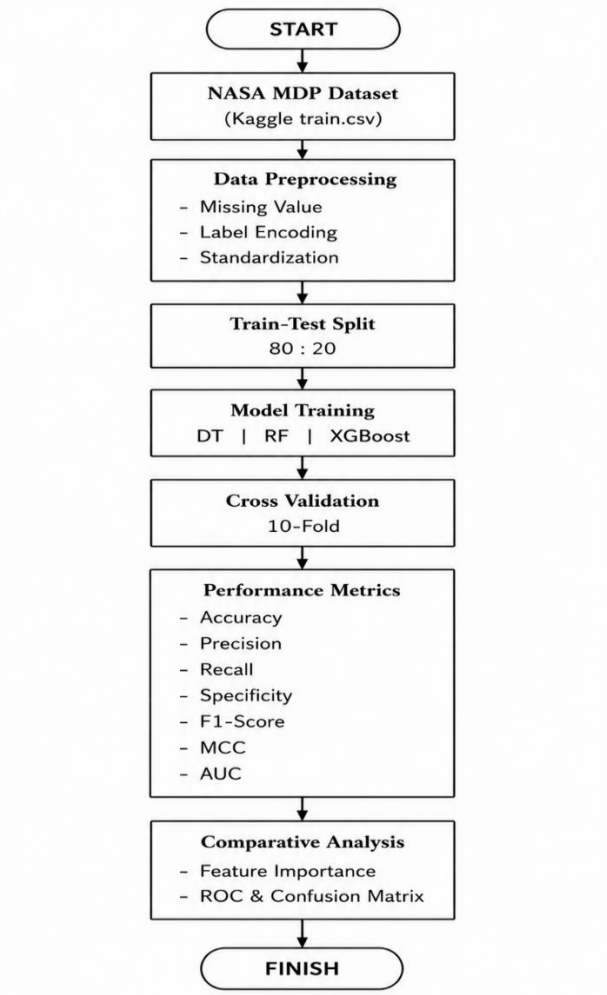
Oleh karena itu, penelitian ini membandingkan Decision Tree, Random Forest, dan XGBoost menggunakan preprocessing yang seragam, validasi 10-fold stratified cross-validation, serta tujuh metrik evaluasi sehingga diharapkan menghasilkan rekomendasi algoritma yang lebih komprehensif untuk Software Defect Prediction. Dataset yang digunakan merupakan Playground Series Season 3 Episode 23 (Binary Prediction of Software Defects) yang tersedia secara publik melalui platform Kaggle. Evaluasi performa model dilakukan menggunakan validasi silang (*cross validation*) serta berbagai metrik evaluasi, meliputi Accuracy, Precision, Recall, Specificity, F1-Score, Matthews Correlation Coefficient (MCC), dan Area Under the Receiver Operating Characteristic Curve (AUC-ROC), sehingga diperoleh gambaran yang komprehensif mengenai kemampuan masing-masing algoritma dalam mengidentifikasi modul *software* yang berpotensi mengalami cacat

Hasil penelitian ini diharapkan dapat memberikan kontribusi teoritis dalam memperkaya kajian mengenai penerapan *Machine Learning* pada *Software Defect Prediction*, sekaligus memberikan kontribusi praktis berupa rekomendasi algoritma yang memiliki performa terbaik untuk mendukung proses *Software Quality Assurance* (SQA). Selain itu, penelitian ini diharapkan dapat menjadi referensi bagi penelitian selanjutnya dalam mengembangkan model prediksi cacat *software* yang lebih akurat, robust, dan mampu diimplementasikan pada lingkungan pengembangan *software* modern.

2. METODOLOGI PENELITIAN

Penelitian ini mengikuti alur sistematis yang terdiri dari enam tahap utama, seperti yang diilustrasikan pada Gambar 1. Secara ringkas, tahapan tersebut meliputi: (1) pengumpulan dataset, (2) preprocessing data, (3) pembagian dataset, (4) pelatihan model, (5) evaluasi model, dan (6) komparasi kinerja.

Building of Informatics, Technology and Science (BITS)
Volume 8, No 1, June 2026 Page: 615−628
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v8i1.10268



Gambar 1. Diagram alir metode penelitian yang diusulkan

Berdasarkan Gambar 1 Penelitian ini dimulai dengan mengunduh dataset NASA MDP dari Kaggle (train.csv). Data kemudian melalui tahap preprocessing yang mencakup imputasi missing values dengan median, label encoding untuk fitur kategorikal, dan standardisasi menggunakan StandardScaler. Setelah data bersih, dilakukan stratified split dengan rasio 80:20 sehingga diperoleh 78.720 sampel untuk training dan 19.680 sampel untuk validasi. Selanjutnya, tiga model yaitu Decision Tree, Random Forest, dan XGBoost dilatih pada training set. Untuk mendapatkan estimasi performa yang robust, diterapkan 10-fold stratified cross validation. Performa setiap model diukur menggunakan tujuh metrik: Accuracy, Precision, Recall, Specificity, F1-Score, MCC, dan AUC-ROC. Terakhir, dilakukan analisis komparatif yang mencakup feature importance serta visualisasi ROC curve dan confusion matrix untuk menentukan algoritma terbaik dalam prediksi cacat perangkat lunak.

2.1. Dataset

Dataset yang digunakan adalah Playground Series Season 3 Episode 23 yang disediakan melalui platform Kaggle. Dataset ini merupakan data sintetis yang dibangun berdasarkan karakteristik dataset NASA Metric Data Program (NASA MDP) sehingga tetap merepresentasikan permasalahan prediksi cacat *software*.

Berdasarkan hasil eksplorasi data awal, dataset train.csv memiliki total 101.763 sampel dengan 25 fitur numerik dan kategorikal, serta satu fitur target defects yang bersifat biner (0 = *non-defective*, 1 = *defective*). Dataset test.csv yang disediakan untuk keperluan prediksi akhir memiliki **54.400 sampel** dengan fitur yang sama (tanpa kolom target). Spesifikasi lengkap dataset disajikan pada Tabel 1.

Tabel 1. Karakteristik Dataset

Aspek	Keterangan
Sumber	Kaggle Playground Series S3E23
Jumlah sampel training	101.763
Jumlah sampel test	54.400
Jumlah fitur	25
Fitur target	defects (binary: 0 = non-defective, 1 = defective)
Tipe fitur	Numerik dan kategorikal

Building of Informatics, Technology and Science (BITS)
Volume 8, No 1, June 2026 Page: 615−628
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v8i1.10268

Berdasarkan Tabel 1, dataset yang digunakan dalam penelitian ini merupakan Playground Series Season 3 Episode 23 yang diperoleh dari platform Kaggle dan dibangun berdasarkan karakteristik NASA Metric Data Program (NASA MDP). Dataset tersebut terdiri atas 101.763 sampel data pelatihan (training) dan 54.400 sampel data pengujian (test) dengan total 25 fitur yang merepresentasikan berbagai metrik perangkat lunak. Variabel target yang digunakan adalah defects, yaitu variabel biner yang mengelompokkan setiap modul perangkat lunak ke dalam kelas non-defective (0) dan defective (1). Selain itu, dataset memiliki kombinasi fitur numerik dan kategorikal, sehingga diperlukan tahapan *preprocessing*, seperti penanganan *missing value*, transformasi fitur kategorikal, dan standardisasi data sebelum dilakukan proses pelatihan model. Karakteristik dataset tersebut menunjukkan bahwa data yang digunakan telah memenuhi kebutuhan penelitian untuk melakukan komparasi algoritma Decision Tree, Random Forest, dan XGBoost dalam prediksi cacat perangkat lunak. Setelah proses *preprocessing* dan sebelum dilakukan pembagian data (*train-test split*), dilakukan analisis distribusi kelas pada dataset *training* untuk mengidentifikasi potensi ketidakseimbangan kelas (*class imbalance*). Distribusi kelas disajikan pada Tabel 2.

Tabel 2. Distribusi Kelas pada Dataset Training

Kelas	Jumlah Sampel	Persentase
Non-Defective (0)	78.699	77,34%
Defective (1)	23.064	22,66%
Total	101.763	100%

Berdasarkan Tabel 2, terlihat bahwa terdapat ketidakseimbangan kelas (*class imbalance*) di mana jumlah sampel *non-defective* (77,34%) lebih besar dibandingkan sampel *defective* (22,66%). Kondisi ini perlu mendapatkan penanganan khusus pada tahap validasi model, seperti penggunaan *stratified cross validation* untuk memastikan proporsi kelas tetap terjaga di setiap *fold*

2.2. Preprocessing Data

Berdasarkan rekomendasi dari literatur [17]-[19], tahap preprocessing yang dilakukan meliputi:

2.2.1. Penanganan Missing Values

Missing values merupakan masalah umum dalam dataset metrik perangkat lunak [20]. Pada penelitian ini, missing values ditangani dengan strategi imputasi menggunakan median, seperti yang direkomendasikan oleh [21], karena median lebih robust terhadap outlier dibandingkan mean.

$$x_{imputed} = \text{median}(x_1, x_2, \ldots, x_n) \quad (1)$$

2.2.2. Encoding Fitur Kategorikal

Fitur kategorikal diubah ke bentuk numerik menggunakan *Label Encoding* [22]:

$$\text{encoded}(c) = \text{index}(c) \text{where} c \in \text{categories} \quad (2)$$

2.2.3. Normalisasi (Opsional)

Fitur numerik distandarisasi menggunakan *StandardScaler* [23]:

$$z = \frac{x-\mu}{\sigma} \quad (3)$$

dimana μ adalah rata-rata fitur dan σ adalah standar deviasi fitur.

2.3. Pembagian Dataset

Dataset training dibagi menjadi dua subset menggunakan metode *stratified splitting* dengan rasio 80:20:
Training set (80%) : digunakan untuk melatih model
Validation set (20%) : digunakan untuk evaluasi awal

Pembagian dilakukan secara *stratified* untuk memastikan proporsi kelas defective dan non-defective tetap seimbang di kedua subset.

2.4. Algoritma yang Dikomparasi

Penelitian ini membandingkan tiga algoritma klasifikasi, yaitu Decision Tree, Random Forest, dan XGBoost. Decision Tree dipilih sebagai model baseline karena memiliki struktur sederhana dan mudah diinterpretasikan. Random Forest digunakan sebagai representasi metode ensemble berbasis bagging yang mampu meningkatkan generalisasi model melalui kombinasi banyak pohon keputusan. Sementara itu, XGBoost merupakan algoritma boosting yang mampu memperbaiki kesalahan prediksi secara bertahap sehingga dikenal memiliki performa tinggi pada berbagai kasus klasifikasi. [11] [12] [13].

Building of Informatics, Technology and Science (BITS)
Volume 8, No 1, June 2026 Page: 615−628
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v8i1.10268

2.5. Validasi Model

Untuk mendapatkan estimasi performa yang lebih robust, penelitian ini menggunakan *K-Fold Cross Validation* dengan k = 10 di mana dataset dibagi menjadi sepuluh subset dengan ukuran yang relatif sama. Pada metode ini, data dibagi menjadi K subset (fold) dengan ukuran yang sama. Secara bergantian, satu fold digunakan sebagai *validation set* dan $K - 1$ fold lainnya sebagai *training set*. Proses ini diulang sebanyak K kali sehingga setiap fold berkesempatan menjadi data validasi [14]. Mengingat potensi ketidakseimbangan kelas (class imbalance) pada SDP, penelitian ini menggunakan *stratified cross validation* yang mempertahankan proporsi kelas yang sama di setiap fold [15].

2.6. Metrik Evaluasi

Evaluasi model dilakukan menggunakan Accuracy, Precision, Recall, Specificity, F1-Score, MCC, dan AUC-ROC.

a. Accuracy
Accuracy digunakan untuk mengukur proporsi seluruh prediksi yang berhasil diklasifikasikan dengan benar oleh model terhadap keseluruhan data pengujian. Semakin tinggi nilai accuracy, semakin baik kemampuan model dalam melakukan klasifikasi secara umum. Nilai accuracy dihitung menggunakan Persamaan (4).

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (4)$$

Dalam evaluasi model klasifikasi, TP (*True Positive*) menunjukkan jumlah modul *defective* yang berhasil diprediksi sebagai *defective*. TN (*True Negative*) merepresentasikan jumlah modul *non-defective* yang diprediksi dengan benar sebagai *non-defective*. FP (*False Positive*) adalah jumlah modul *non-defective* yang keliru diprediksi sebagai *defective*, sedangkan FN (*False Negative*) menunjukkan jumlah modul *defective* yang salah diprediksi sebagai *non-defective*.

b. Precision
Precision digunakan untuk mengukur tingkat ketepatan model dalam memberikan prediksi positif. Nilai precision menunjukkan seberapa banyak data yang diprediksi sebagai *defective* benar-benar merupakan modul yang mengalami cacat. Semakin tinggi nilai precision, semakin kecil kemungkinan model menghasilkan *false positive*. Perhitungan precision menggunakan Persamaan (5).

$$Precision = \frac{TP}{TP+FP} \quad (5)$$

Dalam evaluasi model, TP (*True Positive*) menunjukkan jumlah prediksi positif yang benar, sedangkan FP (*False Positive*) menunjukkan jumlah prediksi positif yang sebenarnya berasal dari kelas negatif.

c. Sensitivity (*Recall*)
Recall atau sensitivity digunakan untuk mengukur kemampuan model dalam mendeteksi seluruh modul perangkat lunak yang benar-benar mengalami cacat. Metrik ini sangat penting pada kasus Software Defect Prediction karena menunjukkan seberapa banyak modul defective yang berhasil dikenali oleh model. Nilai recall dihitung menggunakan Persamaan (6).

$$Recall = \frac{TP}{TP+FN} \quad (6)$$

Dalam evaluasi model, TP (*True Positive*) menunjukkan jumlah modul *defective* yang berhasil diprediksi dengan benar, sedangkan FN (*False Negative*) merupakan jumlah modul *defective* yang tidak berhasil dikenali atau terlewat oleh model.

d. Specificity
Specificity digunakan untuk mengukur kemampuan model dalam mengidentifikasi secara benar modul perangkat lunak yang tidak mengalami cacat (non-defective). Metrik ini menunjukkan proporsi data negatif yang berhasil diprediksi dengan benar dibandingkan dengan seluruh data yang sebenarnya termasuk dalam kelas negatif. Nilai specificity yang tinggi menunjukkan bahwa model mampu meminimalkan kesalahan False Positive (FP), yaitu kondisi ketika modul yang sebenarnya tidak mengalami cacat diprediksi sebagai mengalami cacat. Dalam konteks Software Defect Prediction, specificity penting untuk mengurangi kesalahan identifikasi yang dapat menyebabkan proses inspeksi atau pengujian dilakukan pada modul yang sebenarnya tidak bermasalah, sehingga penggunaan waktu dan sumber daya menjadi lebih efisien. Nilai specificity dihitung menggunakan Persamaan (7).

$$Specificity = \frac{TN}{TN+FP} \quad (7)$$

TN (*True Negative*) merupakan jumlah modul *non-defective* yang berhasil diprediksi secara benar sebagai *non-defective*. Sementara itu, FP (*False Positive*) adalah jumlah modul *non-defective* yang keliru diprediksi sebagai *defective*.

Building of Informatics, Technology and Science (BITS)
Volume 8, No 1, June 2026 Page: 615−628
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v8i1.10268

3. HASIL DAN PEMBAHASAN

3.1. Desain Eksperimen

Eksperimen pada penelitian ini dirancang untuk membandingkan kinerja tiga algoritma klasifikasi, yaitu Decision Tree, Random Forest, dan XGBoost, dalam melakukan prediksi cacat *software* menggunakan dataset Playground Series Season 3 Episode 23 yang merepresentasikan karakteristik NASA Metric Data Program (NASA MDP). Sebelum proses pelatihan model, dataset terlebih dahulu melalui tahap *preprocessing* yang meliputi penanganan *missing value*, transformasi fitur kategorikal menggunakan *Label Encoding*, serta standardisasi fitur numerik menggunakan *StandardScaler*.

Selanjutnya, dataset dibagi menjadi data pelatihan dan data validasi dengan rasio 80:20 menggunakan metode stratified train-test split untuk mempertahankan proporsi kelas pada kedua subset. Proses pelatihan dilakukan menggunakan parameter yang telah ditetapkan untuk masing-masing algoritma, sedangkan evaluasi kinerja model dilakukan melalui 10-fold stratified cross validation guna memperoleh estimasi performa yang lebih stabil dan mengurangi bias akibat pembagian data. Kinerja model dianalisis menggunakan metrik Accuracy, Precision, Recall, Specificity, F1-Score, Matthews Correlation Coefficient (MCC), dan AUC-ROC, kemudian dilengkapi dengan analisis confusion matrix, ROC curve, feature importance, dan learning curve untuk memberikan gambaran yang komprehensif mengenai kemampuan setiap algoritma dalam memprediksi cacat *software*.

3.2. Evaluasi Kinerja Model

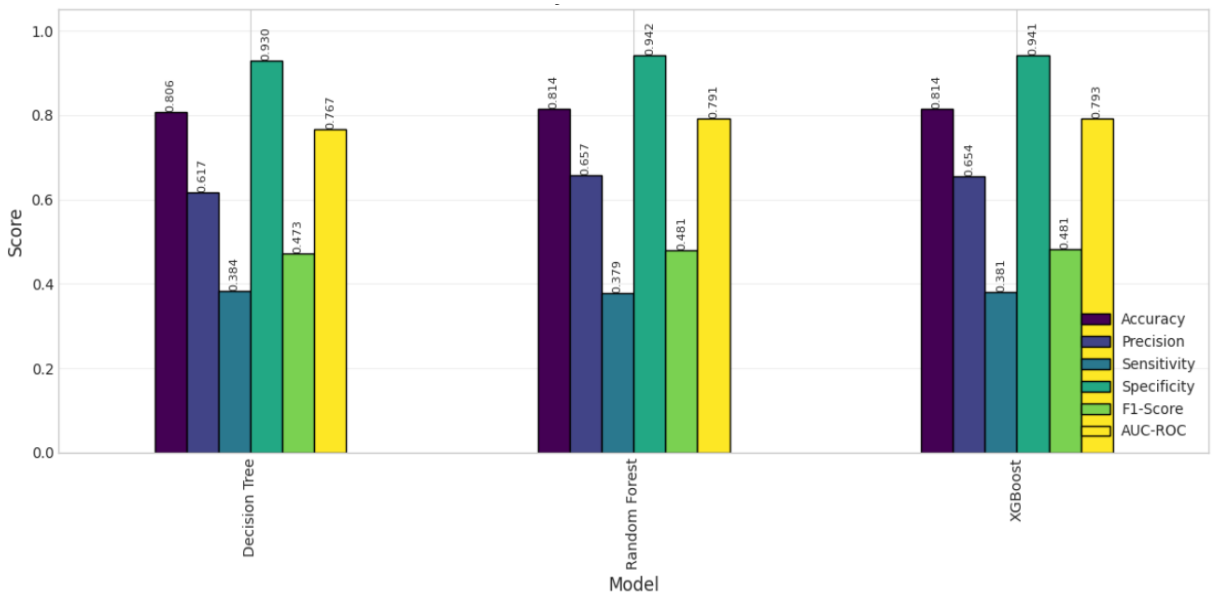
Tahap pertama evaluasi dilakukan tanpa menggunakan teknik reduksi fitur. Tujuan dari tahap ini adalah untuk mengukur performa baseline dari setiap algoritma sebelum dilakukan optimasi lebih lanjut. Hasil evaluasi disajikan pada Tabel 3.

Tabel 3. Hasil evaluasi kinerja model

Model	Accuracy	Precision	Sensitivity	Specificity	F1-Score	MCC	AUC-ROC CV
Decision Tree	0,8064	0,6172	0,3837	0,9302	0,4732	0,3778	0,7668
Random Forest	0,8144	0,6568	0,3791	0,9419	0,4808	0,3986	0,7915
XGBoost	0,814	0,6541	0,3809	0,941	0,4814	0,3981	0,7929

Berdasarkan Tabel 3, Random Forest memperoleh nilai Accuracy tertinggi sebesar 0,8142, sedangkan XGBoost menghasilkan nilai AUC-ROC tertinggi sebesar 0,7919 pada dataset yang digunakan. Hasil pengujian statistik menggunakan Wilcoxon Signed-Rank Test menunjukkan bahwa perbedaan performa antar algoritma signifikan secara statistik ($p < 0,05$). Dengan demikian, peningkatan performa metode ensemble learning tidak hanya terlihat dari nilai metrik evaluasi, tetapi juga didukung oleh hasil pengujian statistik. Namun demikian, pemilihan algoritma tetap perlu mempertimbangkan aspek efisiensi komputasi karena setiap metode memiliki karakteristik waktu pelatihan dan prediksi yang berbeda.

Untuk memperjelas hasil evaluasi pada Tabel 3, dilakukan visualisasi dalam bentuk diagram batang sehingga perbedaan performa setiap algoritma pada masing-masing metrik evaluasi dapat diamati secara lebih mudah. Visualisasi tersebut ditampilkan pada Gambar 2.



Gambar 2. Perbandingan akurasi prediksi cacat perangkat lunak tanpa reduksi fitur

Building of Informatics, Technology and Science (BITS)
Volume 8, No 1, June 2026 Page: 615−628
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v8i1.10268

Gambar 2 menyajikan perbandingan performa komparatif antara tiga algoritma klasifikasi—Decision Tree, Random Forest, dan XGBoost—pada skenario baseline tanpa dilakukan reduksi fitur. Secara umum, Berdasarkan Gambar 2, algoritma Random Forest dan XGBoost menunjukkan performa yang lebih baik pada sebagian besar metrik evaluasi dibandingkan Decision Tree untuk dataset yang digunakan dalam penelitian ini. Namun, hasil tersebut merefleksikan performa pada dataset Playground Series Season 3 Episode 23 dan tidak secara langsung menunjukkan bahwa kedua algoritma tersebut akan memberikan hasil yang sama pada dataset Software Defect Prediction lainnya.

Namun, perhatian khusus perlu diberikan pada kesenjangan yang signifikan antara nilai Specificity yang mencapai kisaran 0.930 - 0.942 dengan nilai Sensitivity yang tertahan di bawah 0.390 untuk ketiga model. Fenomena ini mengindikasikan adanya efek ketidakseimbangan kelas (class imbalance) yang masif pada dataset target. Akibatnya, model memiliki bias yang tinggi dalam mengenali modul non-defect (kelas mayoritas), tetapi kurang sensitif dalam mendeteksi modul defect (kelas minoritas). Hal ini terkonfirmasi dari nilai F1-Score yang moderat, di mana Random Forest dan XGBoost berimbang di angka 0.481, mengungguli Decision Tree yang hanya meraih 0.473

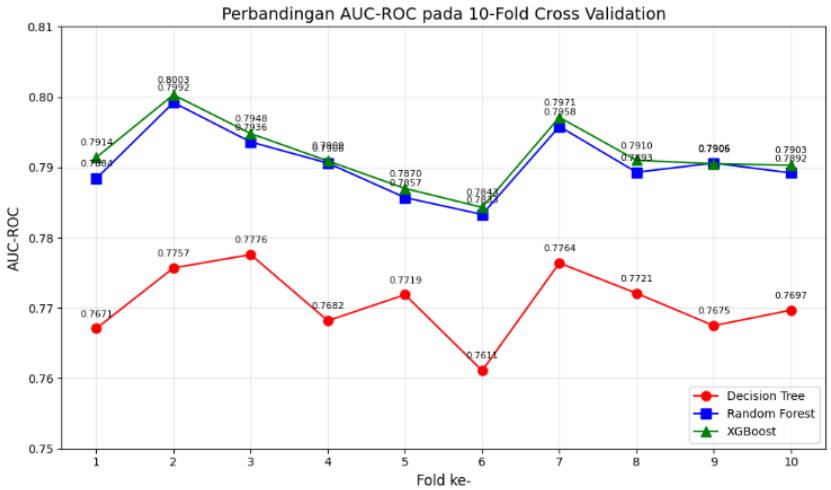
3.4. Evaluasi Kinerja dengan Cross-Validation

Untuk mendapatkan estimasi performa yang lebih robust, dilakukan validasi silang 10-fold. Hasil validasi silang disajikan pada Tabel 4 dan Gambar 3.

Tabel 4. Hasil 10-Fold Cross Validation (Nilai AUC-ROC) untuk Setiap Model

Fold ke-	Decision Tree	Random Forest	XGBoost
1	0,7671	0,7884	0,7914
2	0,7757	0,7992	0,8003
3	0,7776	0,7936	0,7948
4	0,7682	0,7906	0,7909
5	0,7719	0,7857	0,7870
6	0,7611	0,7833	0,7843
7	0,7764	0,7958	0,7971
8	0,7721	0,7893	0,7910
9	0,7675	0,7906	0,7905
10	0,7697	0,7892	0,7903
Mean	0,7707	0,7906	0,7918
Std	0,0048	0,0044	0,0044

Berdasarkan Tabel 4, hasil 10-fold cross validation menunjukkan bahwa XGBoost memberikan rata-rata AUC tertinggi sebesar 0,7918, diikuti oleh Random Forest (0,7906) dan Decision Tree (0,7707). Dari sisi stabilitas, Random Forest dan XGBoost memiliki standar deviasi yang lebih rendah (0,0044) dibandingkan Decision Tree (0,0048), mengindikasikan bahwa kedua metode *ensemble* memiliki konsistensi performa yang lebih baik di berbagai *fold*.

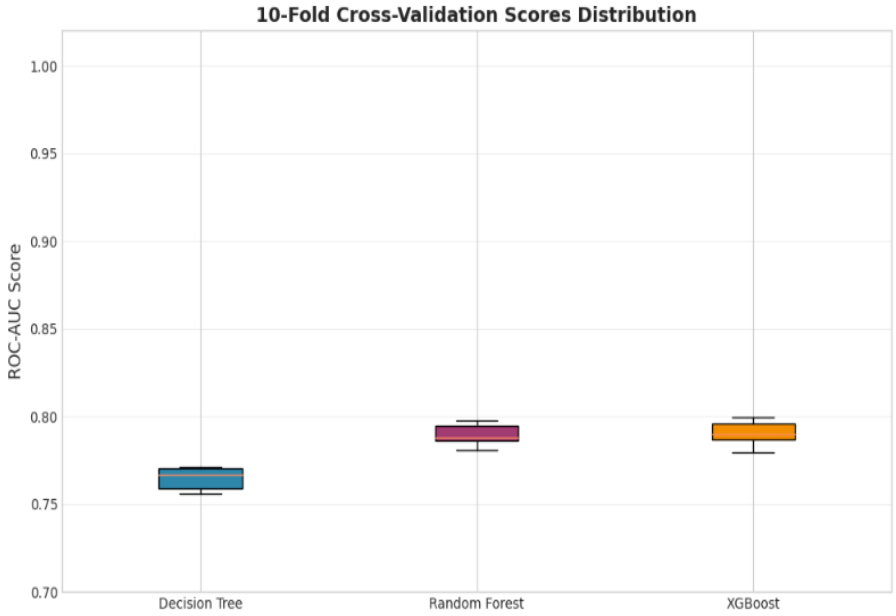


Gambar 3. Perbandingan kinerja algoritma Decision Tree, Random Forest, dan XGBoost pada prediksi cacat *software*.

Dari Gambar 3 tersebut dapat disimpulkan bahwa Random Forest memperoleh nilai terbaik pada metrik Accuracy (0,8144), Precision (0,6568), Specificity (0,9419), dan MCC (0,3986). Sementara itu, XGBoost menunjukkan performa terbaik pada F1-Score (0,4814) dan AUC-ROC (0,7918), yang mengindikasikan kemampuan yang lebih baik dalam membedakan modul *defective* dan *non-defective*. Di sisi lain, Decision Tree memiliki nilai Recall (0,3837) tertinggi, namun secara keseluruhan performanya masih berada di bawah kedua metode berbasis *ensemble learning*. Hasil ini menunjukkan bahwa pendekatan *ensemble learning*, khususnya Random Forest dan

Building of Informatics, Technology and Science (BITS)
Volume 8, No 1, June 2026 Page: 615−628
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v8i1.10268

XGBoost, memberikan kinerja yang lebih unggul dan stabil dalam prediksi cacat *software* dibandingkan penggunaan Decision Tree sebagai *single classifier*.



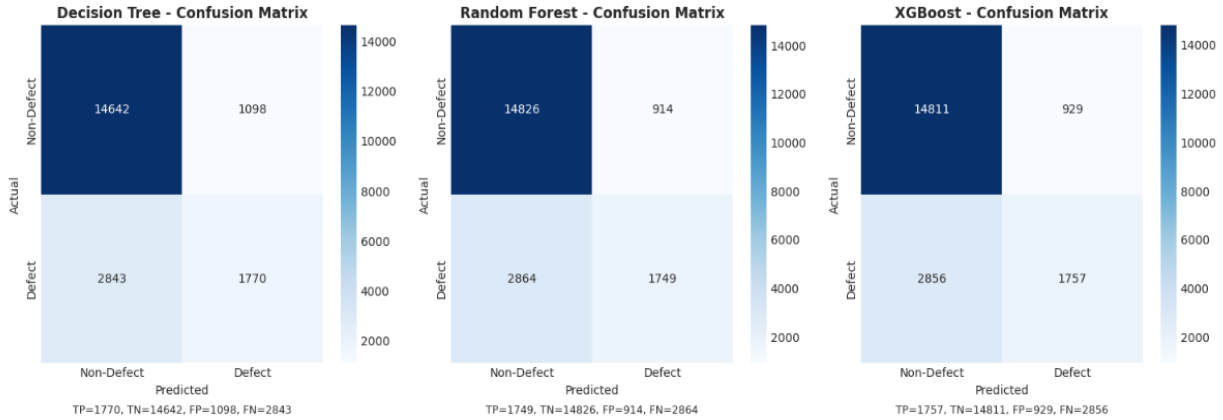
Gambar 4. 10-fold cross validation untuk setiap model

Gambar 4 menunjukkan distribusi nilai ROC-AUC hasil 10-fold cross validation untuk algoritma Decision Tree, Random Forest, dan XGBoost. Berdasarkan boxplot yang dihasilkan, Random Forest dan XGBoost memiliki nilai median ROC-AUC yang hampir sama, yaitu sekitar 0,79, sedangkan Decision Tree memiliki median yang lebih rendah, yaitu sekitar 0,76. Hasil ini menunjukkan bahwa kedua metode berbasis *ensemble learning* memiliki kemampuan klasifikasi yang lebih baik dalam membedakan modul *software* yang mengalami cacat dan tidak mengalami cacat dibandingkan dengan Decision Tree.

Selain memiliki performa yang tinggi, Random Forest juga menunjukkan tingkat kestabilan yang paling baik, ditunjukkan oleh ukuran boxplot yang lebih sempit sehingga variasi hasil antar lipatan validasi relatif kecil. Sementara itu, XGBoost memiliki performa yang sangat kompetitif dengan variasi yang sedikit lebih besar, sedangkan Decision Tree menghasilkan performa yang paling rendah. Secara keseluruhan, hasil 10-fold cross validation memperkuat bahwa metode *ensemble learning*, khususnya Random Forest dan XGBoost, memberikan prediksi yang lebih akurat dan konsisten dibandingkan metode Decision Tree tunggal.

3.5. Confusion Matrix

Analisis confusion matrix penting untuk memahami jenis kesalahan yang dibuat oleh setiap model. Gambar 5 menyajikan confusion matrix untuk ketiga model.



Gambar 5. Confusion matrix untuk (a) Decision Tree, (b) Random Forest, (c) XGBoost

Dari Gambar 5, dapat diamati detail performa klasifikasi masing-masing model sebagai berikut:

a) Decision Tree: Menghasilkan nilai False Positives (FP = 1098) yang paling tinggi dibandingkan model lainnya. Hal ini mengindikasikan bahwa model tunggal ini memiliki kecenderungan paling besar dalam salah memprediksi

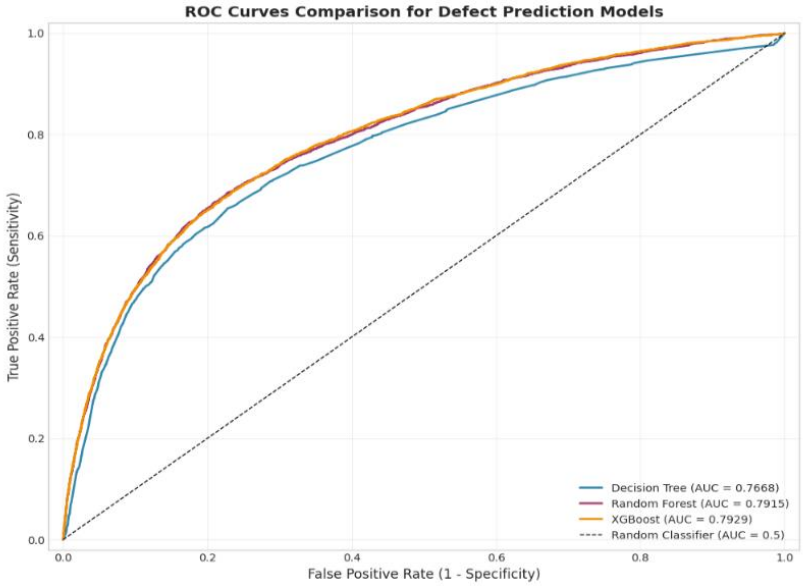
Building of Informatics, Technology and Science (BITS)
Volume 8, No 1, June 2026 Page: 615−628
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v8i1.10268

modul normal (*non-defect*) sebagai modul yang cacat (*defective*). Meskipun nilai *True Positive* (TP = 1770) sedikit lebih tinggi, tingginya angka *error* pada FP menurunkan presisi model secara keseluruhan.

b) Random Forest: Berhasil mengoptimalkan prediksi pada kelas *non-defect* dengan menghasilkan True Negatives (TN = 14826) tertinggi dan menekan tingkat kekeliruan False Positives (FP = 914) hingga ke titik paling rendah di antara ketiga model. Namun, model ini menunjukkan performa yang sedikit lebih konservatif dalam mendeteksi modul cacat yang sebenarnya (*False Negatives* cenderung naik menjadi 2864).

c) XGBoost: Menunjukkan performa yang paling seimbang dan optimal dalam menangani *trade-off* antara kesalahan prediksi. XGBoost mampu memperbaiki kelemahan *False Negatives* (FN) dari Random Forest (turun menjadi 2856) sekaligus mempertahankan tingkat *False Positives* (FP = 929) yang tetap jauh lebih rendah daripada Decision Tree. Karakteristik ini membuat XGBoost menjadi model dengan kemampuan generalisasi yang paling stabil untuk kedua kelas.

3.6. Kurva ROC



Gambar 6. Kurva ROC untuk perbandingan ketiga model

Berdasarkan Gambar 6, kurva ROC (*Receiver Operating Characteristic*) digunakan untuk mengevaluasi kemampuan model dalam membedakan modul *software* yang mengalami cacat (*defective*) dan yang tidak mengalami cacat (*non-defective*). Semakin dekat kurva menuju sudut kiri atas, maka semakin baik kemampuan model dalam melakukan klasifikasi karena menghasilkan nilai *True Positive Rate* yang tinggi dengan *False Positive Rate* yang rendah. Sementara itu, garis diagonal putus-putus merepresentasikan klasifikasi acak (*random classifier*) dengan nilai AUC sebesar 0,5, sehingga model yang memiliki kurva jauh di atas garis tersebut dapat dikatakan memiliki kemampuan prediksi yang baik.

Hasil penelitian menunjukkan bahwa XGBoost memperoleh nilai AUC-ROC tertinggi sebesar 0,7929, diikuti oleh Random Forest sebesar 0,7915, dan Decision Tree sebesar 0,7668. Nilai tersebut menunjukkan bahwa XGBoost memiliki kemampuan terbaik dalam membedakan modul yang berpotensi mengalami cacat dengan modul yang tidak mengalami cacat. Meskipun selisih nilai AUC antara XGBoost dan Random Forest hanya sekitar 0,0014, keduanya tetap menunjukkan performa yang lebih baik dibandingkan Decision Tree sebagai model tunggal.

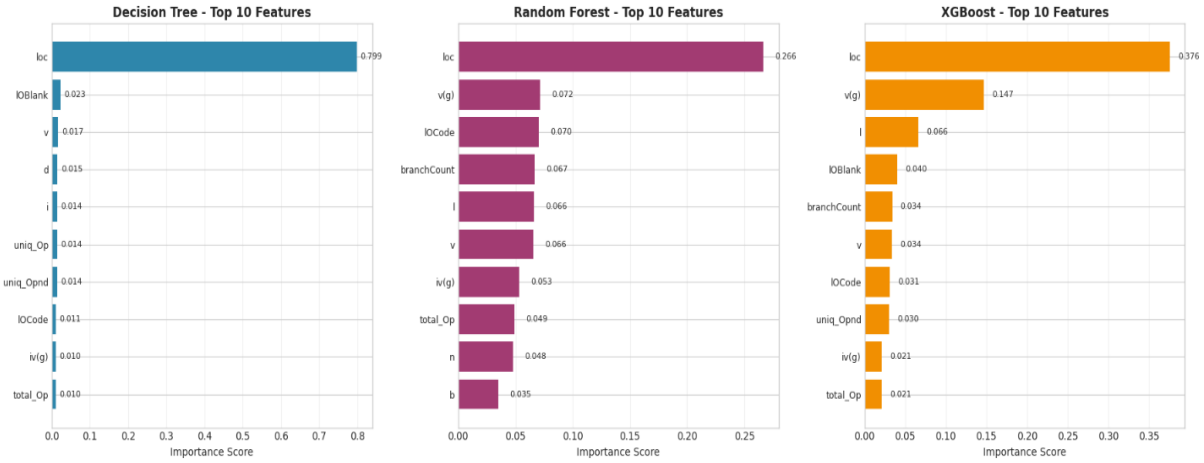
Kurva ROC juga memperlihatkan bahwa Random Forest dan XGBoost memiliki bentuk kurva yang hampir berhimpitan dan lebih mendekati sudut kiri atas dibandingkan Decision Tree. Hal ini menunjukkan bahwa kedua metode berbasis *ensemble learning* mampu menghasilkan proses klasifikasi yang lebih akurat dan stabil. Sebaliknya, kurva Decision Tree berada di bawah kedua model tersebut, yang mengindikasikan bahwa penggunaan satu pohon keputusan lebih rentan terhadap variasi data sehingga kemampuan generalisasinya lebih rendah.

Keunggulan Random Forest dan XGBoost disebabkan oleh mekanisme *ensemble learning* yang menggabungkan banyak pohon keputusan untuk menghasilkan prediksi akhir. Random Forest menggunakan pendekatan *bagging* untuk mengurangi variasi (*variance*), sedangkan XGBoost menggunakan pendekatan *boosting* yang membangun model secara bertahap dengan memperbaiki kesalahan pada iterasi sebelumnya. Oleh karena itu, kedua algoritma mampu membentuk batas keputusan (*decision boundary*) yang lebih optimal dalam memisahkan kelas *defective* dan *non-defective*, sehingga menghasilkan performa prediksi yang lebih baik dibandingkan Decision Tree.

3.7. Feature Importance Analysis

Untuk memahami faktor-faktor apa saja yang paling berkontribusi dalam prediksi cacat perangkat lunak, dilakukan analisis feature importance.

Building of Informatics, Technology and Science (BITS)
Volume 8, No 1, June 2026 Page: 615−628
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v8i1.10268



Gambar 7. Feature importance untuk (a) Decision Tree, (b) Random Forest, (c) XGBoost (seperti Fig. 18 di artikel referensi)

Berdasarkan Gambar 7, fitur loc merupakan fitur yang paling dominan pada ketiga model, sedangkan fitur v(g), v, IOCode, dan l/branchCount juga memberikan kontribusi yang tinggi meskipun urutan tingkat kepentingannya berbeda pada setiap algoritma. Berdasarkan hasil analisis *feature importance*, lima fitur yang paling dominan dalam proses prediksi cacat *software* pada ketiga algoritma yang diuji adalah loc (Lines of Code), v(g) (Cyclomatic Complexity), v (Halstead Volume), IOCode (Operator Instruction Lines), serta l (Halstead Program Length) atau branchCount (jumlah percabangan). Fitur loc menjadi variabel dengan tingkat kepentingan tertinggi pada seluruh model, menunjukkan bahwa ukuran kode memiliki hubungan yang kuat dengan kemungkinan munculnya cacat pada *software*. Selanjutnya, v(g) dan branchCount merepresentasikan kompleksitas alur logika program, di mana semakin kompleks struktur percabangan suatu modul, semakin besar potensi terjadinya kesalahan implementasi. Sementara itu, v (Halstead Volume) dan l menggambarkan kompleksitas dan ukuran program berdasarkan jumlah operator dan operand yang digunakan, sedangkan IOCode mencerminkan banyaknya instruksi operasi dalam kode. Dominasi fitur-fitur tersebut menunjukkan bahwa ukuran dan kompleksitas kode merupakan faktor utama yang memengaruhi terjadinya cacat *software*. Temuan ini mengindikasikan bahwa modul dengan kode yang lebih panjang, lebih kompleks, dan memiliki banyak percabangan cenderung memiliki risiko cacat yang lebih tinggi dibandingkan modul dengan struktur yang lebih sederhana, sehingga metrik-metrik tersebut dapat dijadikan indikator penting dalam proses *Software Defect Prediction* dan pengambilan keputusan pada kegiatan *Software Quality Assurance*.

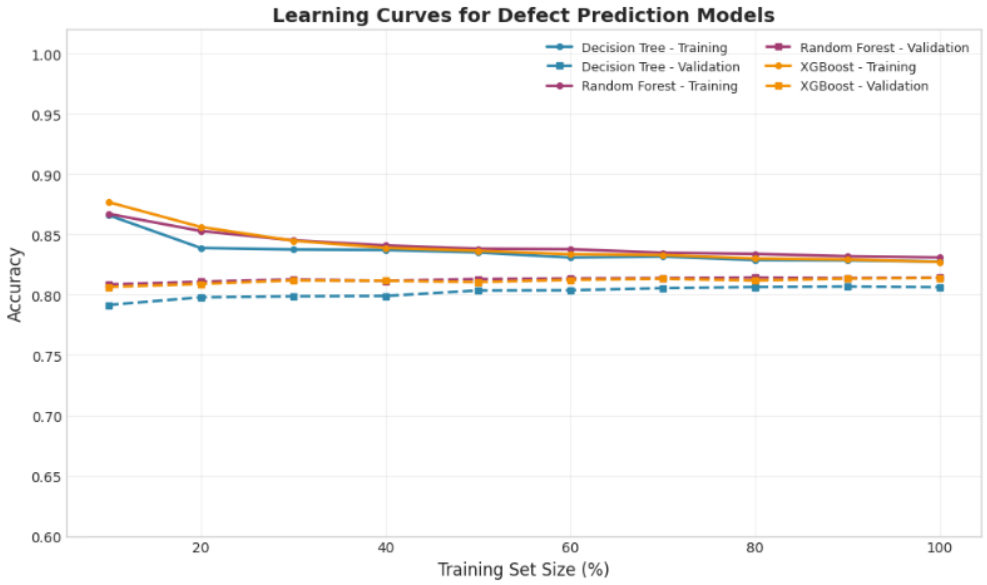
Berdasarkan hasil analisis feature importance pada Gambar 7, dapat diidentifikasi beberapa fitur yang memberikan kontribusi terbesar dalam proses prediksi cacat *software*. Informasi ini memiliki peranan penting karena dapat membantu praktisi maupun pengembang *software* dalam memfokuskan perhatian pada metrik-metrik yang paling berpengaruh terhadap kualitas perangkat lunak. Selain itu, pemilihan fitur yang dominan juga dapat mengurangi biaya pengumpulan dan pengolahan data dengan hanya menggunakan atribut yang relevan, sekaligus meningkatkan interpretabilitas model sehingga hasil prediksi lebih mudah dipahami dan dijelaskan [16].

Hasil penelitian menunjukkan bahwa fitur loc (Lines of Code), v(g) (Cyclomatic Complexity), ev(g) (Essential Complexity), iv(g) (Design Complexity), dan branchCount (Jumlah Percabangan) merupakan fitur yang paling dominan dalam memengaruhi prediksi cacat *software*. Dominasi fitur-fitur tersebut mengindikasikan bahwa ukuran kode dan tingkat kompleksitas struktur program merupakan faktor utama yang berkaitan dengan munculnya cacat pada perangkat lunak. Semakin besar ukuran kode dan semakin kompleks alur logika yang dimiliki suatu modul, maka semakin tinggi pula risiko terjadinya kesalahan atau cacat. Temuan ini memberikan implikasi praktis bahwa proses pengembangan dan pengujian *software* sebaiknya lebih difokuskan pada modul-modul yang memiliki karakteristik tersebut, sehingga kualitas perangkat lunak dapat ditingkatkan secara lebih efektif dan efisien.

3.8. Learning Curve Analysis

Learning curve digunakan untuk mendiagnosis apakah model mengalami overfitting atau underfitting. Gambar 8 menunjukkan *learning curve* dari algoritma Decision Tree, Random Forest, dan XGBoost pada berbagai ukuran data pelatihan. Terlihat bahwa nilai *training accuracy* cenderung menurun, sedangkan *validation accuracy* meningkat dan kemudian stabil seiring bertambahnya jumlah data pelatihan. Hal ini menunjukkan bahwa model semakin mampu melakukan generalisasi terhadap data baru. Random Forest dan XGBoost menghasilkan *validation accuracy* yang sedikit lebih tinggi dibandingkan Decision Tree, sehingga memiliki kemampuan prediksi yang lebih baik. Selain itu, selisih antara kurva pelatihan dan validasi relatif kecil, mengindikasikan bahwa ketiga model hanya mengalami *overfitting* ringan. Kurva yang mulai konvergen pada ukuran data sekitar 50%–60% juga menunjukkan bahwa penambahan data setelah titik tersebut tidak memberikan peningkatan performa yang signifikan.

Building of Informatics, Technology and Science (BITS)
Volume 8, No 1, June 2026 Page: 615−628
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v8i1.10268



Gambar 8. Learning curve untuk ketiga model

3.9. Analisis Kesalahan (Error Analysis)

Berdasarkan confusion matrix pada Gambar 5, terdapat dua jenis kesalahan klasifikasi yang menjadi perhatian utama, yaitu False Positive (FP) dan False Negative (FN). *False Positive* terjadi ketika modul *non-defective* diprediksi sebagai *defective*, sehingga dapat menyebabkan pemborosan waktu dan sumber daya pada proses *Software Quality Assurance (SQA)* karena modul yang sebenarnya tidak bermasalah tetap menjalani proses inspeksi dan pengujian lebih lanjut. Sebaliknya, *False Negative* terjadi ketika modul yang sebenarnya mengalami cacat diprediksi sebagai *non-defective*. Kesalahan ini memiliki dampak yang lebih serius karena cacat tidak terdeteksi pada tahap pengujian dan berpotensi terbawa hingga tahap implementasi, yang pada akhirnya dapat menyebabkan penurunan kualitas *software*, peningkatan biaya pemeliharaan, bahkan kegagalan sistem [17].

Hasil penelitian menunjukkan bahwa XGBoost mampu menghasilkan jumlah *False Negative* yang lebih rendah dibandingkan Decision Tree dan Random Forest, sehingga memiliki kemampuan yang lebih baik dalam mendeteksi modul yang berpotensi mengalami cacat. Sementara itu, Decision Tree menghasilkan jumlah *False Positive* yang relatif lebih tinggi, yang mengindikasikan kecenderungan model tersebut memberikan prediksi positif secara berlebihan. Temuan ini menunjukkan bahwa pendekatan *ensemble learning*, khususnya XGBoost, lebih efektif dalam meminimalkan kesalahan klasifikasi yang kritis pada prediksi cacat *software*. Kemampuan tersebut sangat penting karena kegagalan mendeteksi modul yang cacat dapat menimbulkan risiko yang jauh lebih besar dibandingkan biaya tambahan akibat pemeriksaan modul yang sebenarnya tidak bermasalah. Oleh karena itu, XGBoost menjadi alternatif yang lebih tepat untuk mendukung proses Software Defect Prediction pada tahap awal pengembangan perangkat lunak.

3.10. Perbandingan dengan Penelitian Terdahulu

Untuk memberikan gambaran mengenai kontribusi penelitian yang diusulkan, dilakukan analisis komparatif terhadap beberapa penelitian terdahulu yang membahas *Software Defect Prediction*. Perbandingan difokuskan pada metode yang digunakan serta performa model yang dilaporkan, sehingga dapat menunjukkan posisi dan kontribusi penelitian ini dalam konteks perkembangan metode prediksi cacat perangkat lunak. Hasil perbandingan tersebut disajikan pada Tabel 5.

Tabel 5. Perbandingan dengan Penelitian Terdahulu

Referensi	Metodologi	Akurasi Terbaik
Mehta et al.[6]	XGBoost + Stacking	Accuracy > 0.9000
Mustaqeem et al.[7]	SVM + PCA	Accuracy 0.9520
Pradhan et al.[18]	PCA-LDA + IMPSO-ELM	Accuracy 0.9836
Camelia-Petrina et al.[19]	Random Forest + HFS	Random Forest terbaik
Ustyannie et al.[20]	Oversampling + SVM	Accuracy 0.8402
Penelitian ini	Decision Tree	Accuracy 0.8064
Penelitian ini	Random Forest	Accuracy 0.8144
Penelitian ini	XGBoost	Accuracy 0.8140; AUC-ROC 0.7929

Berdasarkan Tabel 5, penelitian terdahulu umumnya memanfaatkan pendekatan hibrida, optimasi, atau teknik feature selection sehingga mampu menghasilkan akurasi yang lebih tinggi. Sementara itu, penelitian ini berfokus pada

Building of Informatics, Technology and Science (BITS)
Volume 8, No 1, June 2026 Page: 615−628
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v8i1.10268

komparasi tiga algoritma klasifikasi dasar, yaitu Decision Tree, Random Forest, dan XGBoost, tanpa menerapkan teknik optimasi maupun feature engineering lanjutan. Meskipun menghasilkan nilai akurasi dan AUC yang lebih rendah dibandingkan beberapa penelitian terdahulu, hasil penelitian ini menunjukkan bahwa metode ensemble learning, khususnya Random Forest dan XGBoost, tetap memberikan performa yang lebih baik dibandingkan Decision Tree. Temuan ini dapat menjadi baseline bagi penelitian selanjutnya yang mengintegrasikan teknik feature selection, penanganan class imbalance, atau optimasi hiperparameter untuk meningkatkan performa prediksi cacat perangkat lunak.

3.11. Statistical Significance Analysis

Untuk melengkapi analisis kinerja model, penelitian ini tidak hanya membandingkan nilai metrik evaluasi, tetapi juga melakukan pengujian signifikansi statistik serta analisis efisiensi komputasi. Pengujian signifikansi dilakukan untuk mengetahui apakah perbedaan performa antaralgoritma benar-benar signifikan secara statistik atau hanya dipengaruhi oleh variasi selama proses validasi silang. Selain itu, analisis waktu komputasi dilakukan untuk mengevaluasi efisiensi masing-masing algoritma dari aspek waktu pelatihan (*training time*) dan waktu prediksi (*prediction time*), sehingga memberikan gambaran yang lebih komprehensif mengenai keseimbangan antara performa klasifikasi dan kebutuhan komputasi pada implementasi nyata. Selain melakukan pengujian signifikansi statistik, penelitian ini juga mengevaluasi efisiensi komputasi setiap algoritma melalui pengukuran waktu pelatihan (*training time*) dan waktu prediksi (*prediction time*). Analisis ini bertujuan untuk memberikan pertimbangan praktis dalam pemilihan algoritma, karena performa klasifikasi yang tinggi tidak selalu diikuti oleh efisiensi komputasi yang baik. Hasil pengukuran waktu komputasi masing-masing algoritma disajikan pada Tabel 7.

Tabel 6. Hasil Wilcoxon Signed-Rank Test

Comparison	Statistic	p-value	Significant
Decision Tree vs Random Forest	0.0	0.0020	Yes
Decision Tree vs XGBoost	0.0	0.0020	Yes
Random Forest vs XGBoost	8.0	0.0488	Yes

Berdasarkan Tabel 6, dilakukan pengujian signifikansi statistik menggunakan Wilcoxon Signed-Rank Test terhadap nilai AUC-ROC yang diperoleh dari proses 10-fold stratified cross-validation. Hasil pengujian menunjukkan bahwa seluruh pasangan algoritma memiliki nilai p-value < 0,05, yaitu 0,0020 untuk perbandingan Decision Tree–Random Forest, 0,0020 untuk Decision Tree–XGBoost, dan 0,0488 untuk Random Forest–XGBoost. Hasil tersebut menunjukkan bahwa perbedaan performa antar algoritma bersifat signifikan secara statistik, sehingga peningkatan performa yang diperoleh tidak hanya disebabkan oleh variasi acak selama proses validasi silang. Dengan demikian, penggunaan metode ensemble learning memberikan peningkatan performa yang dapat dibuktikan secara statistik dibandingkan pendekatan single classifier pada dataset yang digunakan.

Tabel 7. Training and Prediction Time

Model	Training Time (s)	Prediction Time (s)
Decision Tree	0.7890	0.0054
Random Forest	4.6690	0.1158
XGBoost	5.9217	0.0287

Berdasarkan Tabel 7, Decision Tree memiliki waktu pelatihan tercepat, yaitu 0,7890 detik, serta waktu prediksi sebesar 0,0054 detik, sehingga menjadi model yang paling efisien dari sisi komputasi. Random Forest memerlukan waktu pelatihan sebesar 4,6690 detik dengan waktu prediksi 0,1158 detik, sedangkan XGBoost membutuhkan waktu pelatihan paling lama, yaitu 5,9217 detik, namun memiliki waktu prediksi yang lebih cepat dibandingkan Random Forest, yaitu 0,0287 detik. Temuan ini menunjukkan adanya trade-off antara performa klasifikasi dan efisiensi komputasi. Meskipun XGBoost memperoleh nilai AUC-ROC tertinggi, model tersebut memerlukan waktu pelatihan yang lebih besar dibandingkan Decision Tree dan Random Forest. Oleh karena itu, pemilihan algoritma tidak hanya mempertimbangkan akurasi prediksi, tetapi juga kebutuhan komputasi dan karakteristik implementasi pada lingkungan nyata.

3.12 Pembahasan

Hasil eksperimen menunjukkan bahwa algoritma Random Forest dan XGBoost memiliki performa yang lebih baik dibandingkan Decision Tree dalam prediksi cacat *software*. Berdasarkan hasil evaluasi, XGBoost memperoleh nilai AUC-ROC tertinggi sebesar 0,7929, yang menunjukkan kemampuan terbaik dalam membedakan modul *defective* dan *non-defective*, sedangkan Random Forest menghasilkan nilai accuracy tertinggi sebesar 0,8144. Sementara itu, Decision Tree memberikan performa yang lebih rendah dibandingkan kedua metode *ensemble learning*. Temuan ini sejalan dengan penelitian Mehta et al. [1] yang menyatakan bahwa metode berbasis *ensemble* mampu menghasilkan performa prediksi yang lebih baik dibandingkan model tunggal karena memiliki kemampuan yang lebih baik dalam mempelajari pola data yang kompleks.

Building of Informatics, Technology and Science (BITS)
Volume 8, No 1, June 2026 Page: 615−628
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v8i1.10268

Keunggulan Random Forest dan XGBoost tidak terlepas dari mekanisme *ensemble learning* yang digunakan. Random Forest menerapkan pendekatan *bagging* dengan menggabungkan banyak pohon keputusan sehingga mampu mengurangi variasi (*variance*) model dan meningkatkan stabilitas prediksi. Sementara itu, XGBoost menggunakan pendekatan *boosting* yang membangun model secara bertahap dengan memperbaiki kesalahan pada iterasi sebelumnya, sehingga menghasilkan model yang lebih optimal dalam memisahkan kelas *defective* dan *non-defective*. Sebaliknya, Decision Tree sebagai *single classifier* lebih rentan mengalami *overfitting* karena sangat sensitif terhadap perubahan data pelatihan, sehingga kemampuan generalisasinya menjadi lebih rendah dibandingkan kedua metode *ensemble* tersebut.

Hasil 10-fold cross validation juga menunjukkan bahwa ketiga model memiliki tingkat stabilitas yang baik, ditunjukkan oleh nilai standar deviasi yang relatif kecil pada setiap pengujian. Di antara ketiganya, XGBoost memiliki nilai standar deviasi paling rendah, yang mengindikasikan bahwa model tersebut mampu menghasilkan performa yang lebih konsisten pada berbagai subset data. Kondisi ini menunjukkan bahwa XGBoost tidak hanya memiliki kemampuan klasifikasi yang baik, tetapi juga memiliki tingkat generalisasi yang lebih tinggi ketika dihadapkan pada data yang belum pernah digunakan selama proses pelatihan.

Meskipun demikian, hasil penelitian ini tidak dapat dijadikan dasar bahwa XGBoost merupakan algoritma terbaik untuk seluruh permasalahan prediksi cacat *software*. Sesuai dengan konsep No Free Lunch Theorem, performa suatu algoritma sangat dipengaruhi oleh karakteristik dataset, distribusi data, dan domain aplikasi yang digunakan. Oleh karena itu, pemilihan algoritma sebaiknya mempertimbangkan karakteristik data dan kebutuhan implementasi sehingga model yang dipilih mampu memberikan performa yang optimal pada kasus yang dihadapi.

4. KESIMPULAN

Penelitian ini menunjukkan bahwa pendekatan ensemble learning, khususnya Random Forest dan XGBoost, memberikan performa yang kompetitif dalam prediksi cacat perangkat lunak pada dataset Playground Series Season 3 Episode 23 yang dibangun berdasarkan karakteristik NASA Metric Data Program (NASA MDP). Hasil penelitian juga menunjukkan bahwa metrik yang berkaitan dengan ukuran dan kompleksitas kode memiliki pengaruh penting terhadap proses prediksi cacat perangkat lunak, sehingga dapat menjadi informasi yang bermanfaat dalam mendukung proses Software Quality Assurance (SQA). Meskipun demikian, penelitian ini memiliki keterbatasan karena menggunakan dataset sintetis yang dirancang untuk merepresentasikan karakteristik NASA MDP. Meskipun dataset tersebut menyediakan lingkungan eksperimen yang konsisten, karakteristiknya belum tentu sepenuhnya mencerminkan kompleksitas, variasi, dan tingkat *noise* yang terdapat pada data perangkat lunak nyata (*real-world software projects*). Oleh karena itu, hasil penelitian ini perlu diinterpretasikan sesuai dengan ruang lingkup dataset yang digunakan dan belum dapat digeneralisasi untuk seluruh kasus *Software Defect Prediction*. Penelitian selanjutnya disarankan untuk melakukan validasi menggunakan dataset nyata dari berbagai proyek perangkat lunak, membandingkan hasil pada beberapa repositori publik, serta mengombinasikan evaluasi performa dengan analisis efisiensi komputasi dan uji signifikansi statistik agar diperoleh kesimpulan yang lebih komprehensif dan memiliki validitas eksternal yang lebih kuat.

REFERENCES

[1] S. R. Goyal, “Effective software defect prediction with deep neural networks,” *Results in Engineering*, vol. 29, no. November 2025, p. 108378, 2026, doi: 10.1016/j.rineng.2025.108378.
[2] Y. Ding *et al.*, “Metric information mining with metric attention to boost software defect prediction performance,” *Science of Computer Programming*, vol. 248, no. August 2025, 2026, doi: 10.1016/j.scico.2025.103381.
[3] L. Shen and H. Bai, “Engineering software-based evaluation of pelvic floor muscle defects in anterior vaginal wall prolapse,” *Journal of Radiation Research and Applied Sciences*, vol. 19, no. 1, p. 102119, 2026, doi: 10.1016/j.jrras.2025.102119.
[4] X. Wei, “Research on Preprocessing Techniques for Software Defect Prediction Dataset Based on Hybrid Category Balance and Synthetic Sampling Algorithm,” *Procedia Computer Science*, vol. 262, pp. 840–848, 2025, doi: 10.1016/j.procs.2025.05.117.
[5] L. Lavazza, S. Morasca, and G. Rotoloni, “Software Defect Prediction evaluation: New metrics based on the ROC curve,” *Information and Software Technology*, vol. 187, no. August, p. 107865, 2025, doi: 10.1016/j.infsof.2025.107865.
[6] T. Bayramova, “Software Defect Prediction Using the Machine Learning Methods,” *Problems of Information Technology*, vol. 14, no. 2, pp. 23–31, 2023, doi: 10.25045/jpit.v14.i2.03.
[7] Akhlas Tariq Hasan and Shayma Mustafa Mohi-Aldeen, “Software Defect Prediction Based On Deep Learning Algorithms : A Systematic Literature Review,” *AL-Rafidain Journal of Computer Sciences and Mathematics*, vol. 19, no. 1, pp. 67–79, 2025, doi: 10.33899/csmj.2025.156086.1160.
[8] E. A. Kusnanti *et al.*, “Prediksi cacat perangkat lunak menggunakan recurrent neural network berbasis pca,” *Jurnal Ilmiah Teknologi Informasi*, pp. 23–31, 2024.
[9] Emma Andini, M. R. Faisal, Rudy Herteno, R. A. Nugroho, Friska Abadi, and Muliadi, “Peningkatan Kinerja Prediksi Cacat Software Dengan Hyperparameter Tuning Pada Algoritma Klasifikasi Deep Forest,” *Jurnal Mnemonic*, vol. 5, no. 2, pp. 119–127, 2022, doi: 10.36040/mnemonic.v5i2.4793.
[10] Emma Andini, M. R. Faisal, Rudy Herteno, R. A. Nugroho, Friska Abadi, and Muliadi, “Peningkatan Kinerja Prediksi Cacat Software Dengan Hyperparameter Tuning Pada Algoritma Klasifikasi Deep Forest,” *Jurnal Mnemonic*, vol. 5, no. 2, pp. 119–127, 2022, doi: 10.36040/mnemonic.v5i2.4793.

Building of Informatics, Technology and Science (BITS)
Volume 8, No 1, June 2026 Page: 615−628
ISSN 2684-8910 (media cetak)
ISSN 2685-3310 (media online)
DOI 10.47065/bits.v8i1.10268

[11] Y. B. L. Kintomonho, M. N. Atchadé, and D. Daddah, "Decision tree-based statistical learning and quantile regression adjustment: Insights from pregnant women in Benin," *Scientific African*, vol. 29, pp. 1–10, Sep. 2025, doi: 10.1016/j.sciaf.2025.e02832.
[12] P. Rosyani, A. M. Lutfi, E. Purwadi, Kamaluddin, Y. A. Hanaan, and I. H. Ikasari, "Application of Random Forest for Rice Plant Disease Classification," *International Journal of Integrative Sciences*, vol. 4, no. 1, pp. 141–150, Feb. 2025, doi: 10.55927/ijis.v4i1.13477.
[13] M. Abdullahi *et al.*, "Detecting Cybersecurity Attacks in Internet of Things Using Artificial Intelligence Methods: A Systematic Literature Review," 2022. doi: 10.3390/electronics11020198.
[14] N. * Risky, D. Setiyawan, D. Hermawan, and O. Herdiyanto, "Prediksi Kelulusan Mahasiswa Menggunakan Algoritma Decision Tree C4.5 Berbasis Data Akademik dengan Validasi 10-Fold," *TIN: Terapan Informatika Nusantara*, vol. 6, no. 6, pp. 670–678, Nov. 2025, doi: 10.47065/tin.v6i6.8662.
[15] N. Istiqomah and M. Murinto, "Klasifikasi Penyakit Tanaman Padi Berbasis Citra Daun Menggunakan Convolutional Neural Network (CNN)," *JSTIE (Jurnal Sarjana Teknik Informatika) (E-Journal)*, vol. 12, no. 1, p. 18, Feb. 2024, doi: 10.12928/jstie.v12i1.27314.
[16] N. Paul, G. C. Sunil, D. Horvath, and X. Sun, "Deep learning for plant stress detection: A comprehensive review of technologies, challenges, and future directions," Feb. 01, 2025, *Elsevier B.V.* doi: 10.1016/j.compag.2024.109734.
[17] R. Rashkovits and I. Lavy, "Mapping Common Errors in Entity Relationship Diagram Design of Novice Designers," *International Journal of Database Management Systems*, vol. 13, no. 1, pp. 1–19, 2021, doi: 10.5121/ijdms.2021.13101.
[18] D. Pradhan and D. Muduli, "Improved prediction of software defect: A particle swarm optimization based ELM approach," *e-Prime - Advances in Electrical Engineering, Electronics and Energy*, vol. 13, no. July, p. 101081, 2025, doi: 10.1016/j.prime.2025.101081.
[19] N. Camelia-Petrina and V. Andreea, "Software Defect Prediction Models. A Replication and Extension Study," *Procedia Computer Science*, vol. 270, pp. 1936–1945, 2025, doi: 10.1016/j.procs.2025.09.314.
[20] W. Ustyannie, E. Setyaningsih, and C. Iswahyudi, "Optimization of software defects prediction in imbalanced class using a combination of resampling methods with support vector machine and logistic regression," *Jurnal Infotel*, vol. 13, no. 4, pp. 176–184, 2021, doi: 10.20895/infotel.v13i4.726.