Informatics for Educators And Professionals : Journal of Informatics
Vol. 10, No. 2, Desember 2025, 1 - 10
E-ISSN: 2548-3412
Terakreditasi **SINTA Peringkat 5**
Surat Keputusan Direktur Jenderal Pendidikan Tinggi, Riset, dan Teknologi, Nomor: 230/E/KPT/2022
masa berlaku mulai Volume 5 Nomor 1 Tahun 2020 sampai Volume 9 Nomor 2 Tahun 2024

Cross Project Defect Prediction Menggunakan Random Forest

Darusman[1],* Agus Subekti[2]

[1,2] Program Studi Magister Ilmu Komputer; Universitas Nusa Mandiri Jakarta; Jl. Raya Jatiwaringin Cipinang Melayu, Kec. Makasar, Kota Jakarta Timur, Daerah Khusus Ibukota Jakarta, No.telpon (021)28534471; e-mail: 14230029@nusamandiri.ac.id

* Korespondensi: e-mail: 14230029@nusamandiri.ac.id

Diterima: 16 Oktober 2025 ; Review: 12 Desember 2025; Disetujui: 16 Desember 2025

Cara sitasi: Darusman, Subekti A. 2025. Cross Project Defect Prediction Menggunakan Random Forest. Informatics for Educators and Professionals : Journal of Informatics. Vol.10 (2): 1-10.

Abstrak: Penelitian ini mengembangkan model prediksi cacat perangkat lunak menggunakan algoritma Random Forest dengan pendekatan Many-to-One Cross-Project Defect Prediction. Model diuji menggunakan dataset AEEEM sebagai data pelatihan dan dataset PROMISE sebagai target pengujian. Kedua dataset ini mencakup berbagai proyek perangkat lunak dengan fitur penting seperti ukuran kode, churn kode, dan kompleksitas yang berperan dalam memprediksi cacat perangkat lunak. Evaluasi kinerja model dilakukan dengan metrik Accuracy, AUC, Recall, Precision, dan F1-Score. Hasil penelitian model Random Forest memberikan performa yang sangat baik, dengan akurasi lebih dari 94% dan AUC lebih dari 0,92 pada sebagian besar dataset. Model ini juga berhasil menyeimbangkan Recall dan Precision, menghasilkan prediksi yang lebih akurat. Penelitian ini menggunakan penerapan teknik ensemble seperti stacking dan voting, yang meningkatkan kestabilan dan akurasi prediksi secara signifikan. Dengan pendekatan ini model Random Forest tidak hanya meningkatkan akurasi tetapi juga efisiensi dalam memprediksi cacat perangkat lunak lintas proyek dalam menghadapi data yang tidak seimbang dan beragam.

Kata kunci: Cross-Project Defect Prediction; Random Forest; Machine Learning;

Abstract: *This study developed a software defect prediction model using the Random Forest algorithm with the Many-to-One Cross-Project Defect Prediction approach. The model was tested using the AEEEM dataset as the training data and the PROMISE dataset as the test target. These two datasets cover a wide range of software projects with important features such as code size, code churn, and complexity that play a role in predicting software flaws. The evaluation of the model's performance was carried out with the Accuracy, AUC, Recall, Precision, and F1-Score metrics. The results of the Random Forest model provide excellent performance, with an accuracy of over 94% and an AUC of more than 0.92 on most datasets. The model also manages to balance Recall and Precision, resulting in more accurate predictions. This study uses the application of ensemble techniques such as stacking and voting, which significantly improves the stability and accuracy of the predictions. With this approach, the Random Forest model not only improves accuracy but also efficiency in predicting software defects across projects in the face of unbalanced and diverse data.*

Keywords*: Cross-Project Defect Prediction; Random Forest; Machine Learning;*

1. Pendahuluan

Dalam pengembangan perangkat lunak, cacat perangkat lunak menjadi salah satu masalah utama yang dapat mempengaruhi kualitas dan keberhasilan suatu proyek. Cacat yang tidak terdeteksi dapat menyebabkan kerusakan pada fungsionalitas sistem, memperburuk pengalaman pengguna, dan meningkatkan biaya pemeliharaan dalam jangka panjang [1][2][3]. Oleh karena itu, memiliki model prediksi yang akurat sangat penting untuk meminimalkan risiko dan meningkatkan efisiensi dalam pengembangan perangkat lunak. Identifikasi cacat secara manual sering kali memakan waktu dan tidak selalu efektif, sehingga penggunaan algoritma machine learning untuk prediksi cacat perangkat lunak menjadi alternatif yang menarik dan lebih efisien [4][5][6]. Penelitian ini menggunakan dataset AEEEM dan PROMISE, yang berisi berbagai informasi tentang proyek perangkat lunak dengan fitur-fitur seperti ukuran kode, churn kode, dan kompleksitas yang berperan penting dalam memprediksi cacat perangkat lunak. Dataset ini mencakup lima proyek perangkat lunak dengan jumlah sampel yang bervariasi dan tingkat cacat yang berbeda, sehingga memberikan data yang baik untuk pengembangan model prediksi.

Model random forest dipilih dalam penelitian ini karena kemampuannya yang sangat baik dalam menangani dataset yang besar, kompleks, dan tidak seimbang, serta kemampuannya untuk memberikan hasil yang stabil meskipun terdapat noise dalam data. Random Forest adalah algoritma berbasis pohon keputusan yang bekerja dengan cara membangun banyak pohon secara acak dan menggabungkan hasil prediksi dari pohon-pohon tersebut untuk menghasilkan keputusan akhir [7][8][9]. Dengan teknik ensemble seperti bagging, model ini mampu mengurangi overfitting dan meningkatkan kemampuan generalisasi pada data yang tidak terlihat. Evaluasi model dilakukan menggunakan metrik seperti Accuracy, AUC, Recall, Precision dan F1-Score yang memberikan gambaran menyeluruh mengenai kemampuan model dalam membedakan kelas cacat dan non-cacat [10][11][12]. Selain itu, untuk meningkatkan akurasi prediksi, teknik ensemble methods seperti stacking dan voting diterapkan untuk menggabungkan hasil dari beberapa model, yang terbukti memberikan prediksi yang lebih stabil dan akurat.

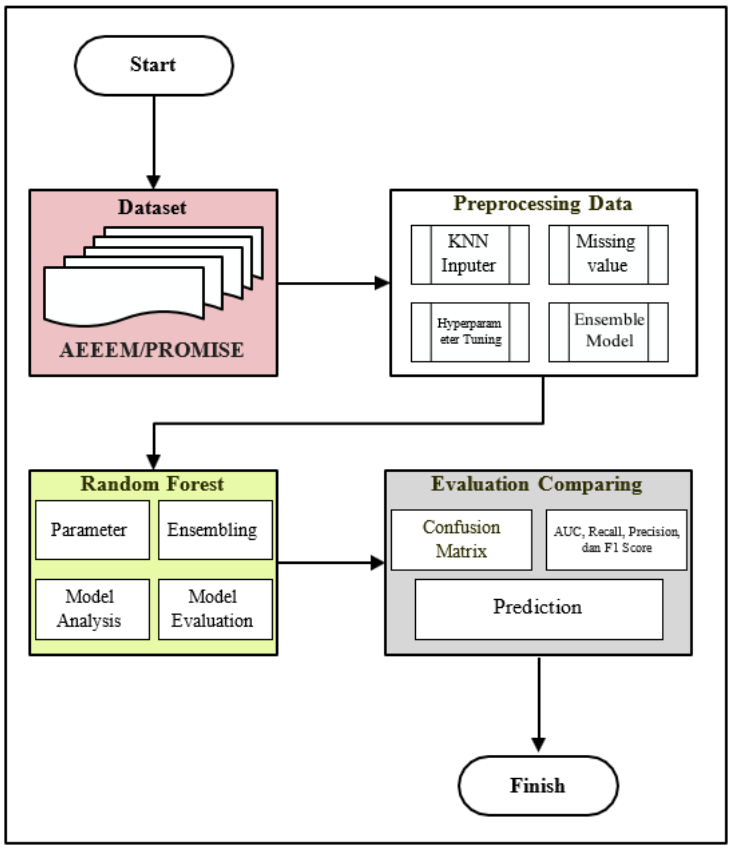
Dalam penelitian ini, dilakukan hyperparameter tuning menggunakan grid search untuk mengoptimalkan parameter penting dari random forest, seperti max_depth, n_estimators, dan min_samples_split [13][14][15]. Proses ini bertujuan untuk menemukan kombinasi parameter yang dapat meningkatkan kinerja model dan menghindari overfitting. Meskipun random forest menunjukkan kinerja yang sangat baik, penelitian ini juga mengidentifikasi tantangan terkait outliers pada beberapa dataset, yang dapat mempengaruhi stabilitas dan akurasi model. Oleh karena itu, teknik deteksi outliers yang lebih baik sangat diperlukan untuk meningkatkan kualitas prediksi. Penggunaan teknik ensemble dan penerapan metode deep learning dapat lebih meningkatkan akurasi model dalam prediksi cacat perangkat lunak. Penelitian ini diharapkan dapat memberikan kontribusi penting dalam pengembangan perangkat lunak dengan membantu para profesional perangkat lunak dalam membuat keputusan yang lebih tepat, efisien, dan efektif dalam mengelola proyek perangkat lunak, serta mengurangi ketergantungan pada pengujian manual yang memakan waktu dan biaya.

Meskipun banyak penelitian yang menggunakan algoritma machine learning untuk prediksi cacat perangkat lunak, sebagian besar berfokus pada satu proyek, yang membatasi kemampuan model untuk digeneralisasi pada proyek lain. Dalam konteks Cross-Project Defect Prediction (CPDP), tantangan utama masih terletak pada ketidakseimbangan data dan variasi antar proyek, serta pengaruh outliers yang dapat menurunkan akurasi model. Selain itu, meskipun teknik ensemble seperti stacking dan voting memiliki potensi untuk meningkatkan akurasi, penerapannya dalam CPDP belum banyak dieksplorasi. Penelitian ini bertujuan untuk menerapkan Random Forest dalam CPDP, mengoptimalkan model melalui hyperparameter tuning, dan menggunakan teknik ensemble lebih lanjut. Dengan demikian, penelitian ini tidak hanya berfokus pada peningkatan akurasi, tetapi juga pada metodologi yang lebih efisien dalam menangani ketidakseimbangan data dan kompleksitas proyek perangkat lunak.

2. Metode Penelitian

Penelitian yang digunakan untuk prediksi cacat perangkat lunak dengan pendekatan Cross-Project Defect Prediction (CPDP) menggunakan algoritma Random Forest. Dimulai dengan pemilihan dataset AEEEM dan PROMISE, data tersebut kemudian diproses melalui tahap pra-pemrosesan yang mencakup penanganan nilai hilang menggunakan teknik KNN

Imputer dan tuning hyperparameter. Selanjutnya, model Random Forest diterapkan untuk menganalisis dan mengevaluasi hasilnya dengan teknik ensemble. Hasil evaluasi dibandingkan menggunakan metrik seperti AUC, Recall, Precision, dan F1-Score, yang pada akhirnya menghasilkan prediksi.



Sumber : Hasil Penelitian (2025)

Gambar 1 Metode Eksperimen

Gambar 1 Metode eksperimen dengan analisis data untuk memprediksi cacat perangkat lunak menggunakan algoritma random forest pada dataset AEEEM dan PROMISE. Dataset AEEEM mencakup berbagai informasi tentang proyek perangkat lunak, termasuk fitur-fitur seperti ukuran kode, *churn* kode, dan kompleksitas, yang dapat mempengaruhi prediksi cacat perangkat lunak. Dalam penelitian ini, pendekatan *Many to One Cross-Project Defect Prediction* diterapkan, di mana dataset AEEEM digunakan sebagai source projects untuk pelatihan model *(training),* sementara dataset PROMISE digunakan sebagai target project untuk pengujian *(testing)* [16][17][18]. Penelitian ini bertujuan untuk mengevaluasi kinerja model random forest dalam memprediksi cacat perangkat lunak dengan menggunakan metrik evaluasi seperti *Accuracy, AUC, Recall, Precision* dan *F1-Score* untuk menghasilkan prediksi yang akurat dan andal pada berbagai proyek perangkat lunak.

1. Dataset

Dataset yang digunakan dalam penelitian ini adalah dataset AEEEM dan PROMISE yang berisi informasi terkait proyek perangkat lunak dengan berbagai atribut yang mencakup ukuran kode, *churn* kode, kompleksitas, dan fitur lainnya yang mempengaruhi prediksi cacat perangkat lunak. Dataset AEEEM mencakup lima proyek perangkat lunak, yaitu EQ, JDT, LC, ML, dan PDE, dengan jumlah *instance* yang bervariasi antara 325 hingga 1862, serta tingkat cacat yang berbeda pada setiap proyek. Sementara itu, dataset PROMISE terdiri dari

beberapa proyek seperti ivy2.0, poi3.0, synapse1.2, xalan2.6, dan xerces1.4, dengan jumlah *instance* antara 241 hingga 885, serta tingkat cacat yang bervariasi, mulai dari 15% hingga 52%. Dataset ini memiliki banyak fitur yang telah dibersihkan dan diproses sebelum dibagi menjadi data latih dan data uji, dengan tujuan untuk memprediksi tingkat cacat perangkat lunak berdasarkan atribut-atribut yang ada.

Tabel 1 Dataset

Dataset	Proyek	Jumlah Instance	Tingkat cacat (%)
PROMISE	ivy2.0	241	15
	poi3.0	442	201
	synapse1.2	256	52
	xalan2.6	885	271
	xerces1.4	588	181

Sumber : Hasil Penelitian (2025)

2. Pra-pemrosesan Data

Pra-pemrosesan data mencakup untuk memastikan kualitas data yang digunakan dalam model prediksi. Langkah pertama adalah penanganan nilai yang hilang, di mana nilai NaN atau kosong diisi menggunakan imputasi sederhana [19]. Untuk data numerik, imputasi dilakukan dengan menggunakan KNNImputer, yang mengisi nilai hilang berdasarkan nilai tetangga terdekat, sedangkan untuk data kategorikal, imputasi dilakukan dengan menggunakan nilai mode atau nilai yang paling sering muncul [20][21]. Setelah penanganan nilai hilang, dilakukan seleksi fitur untuk mengurangi dimensi data yang tidak relevan atau berlebihan yang dapat mempengaruhi kinerja model [22]. Data kemudian dibagi menjadi dua set *Training Set* yang digunakan untuk melatih model, dan *Test Set* untuk menguji kinerja model. Pembagian data ini dilakukan dengan proporsi 80% untuk data latih dan 20% untuk data uji, yang memungkinkan evaluasi yang lebih akurat terhadap performa model.

3. Hyperparameters Tuning

Hyperparameters tuning dilakukan dengan menggunakan *grid search* untuk menemukan kombinasi parameter terbaik, seperti *max_depth, min_samples_split, n_estimators,* dan *max_features* [23][24]. Proses ini bertujuan untuk mengoptimalkan kinerja model dengan memilih nilai-nilai parameter yang dapat meningkatkan akurasi prediksi. Penyesuaian parameter bertujuan untuk menghindari *overfitting* dengan memastikan bahwa model tidak terlalu menyesuaikan diri dengan data latih, sehingga model dapat memiliki kemampuan generalisasi yang lebih baik ketika diuji dengan data yang belum terlihat. Dengan demikian, *hyperparameters tuning* sangat penting untuk meningkatkan performa model, mengurangi kesalahan prediksi, dan meningkatkan kemampuan model dalam menangani berbagai skenario yang ada pada data uji.

4. Random Forest

Random Forest merupakan algoritma *ensemble* berbasis pohon keputusan. Random Forest bekerja dengan membangun beberapa pohon keputusan secara acak dan menggabungkan hasil prediksi dari pohon-pohon tersebut untuk menghasilkan keputusan akhir [25][26]. Setiap pohon keputusan dalam random forest dibangun dengan memilih subset acak dari fitur dan sampel data, yang membantu meningkatkan keragaman antar pohon dan mengurangi *overfitting.* Algoritma ini menggunakan teknik *bagging (bootstrap aggregating)* untuk melatih model secara paralel, dimana setiap pohon dilatih dengan subset data yang berbeda. Keunggulan random forest terletak pada kemampuannya menangani dataset besar, data yang tidak seimbang, dan data dengan banyak fitur, serta memberikan hasil prediksi yang sangat akurat dan stabil [27][28]. Selain itu, random forest memberikan fitur penting seperti estimasi variabel yang berkontribusi besar pada prediksi, yang memudahkan interpretasi model.

5. Model Ansambel

Model ansambel digunakan untuk menggabungkan hasil dari berbagai model guna meningkatkan akurasi prediksi. Teknik stacking dan *voting* diterapkan untuk mengombinasikan output dari model random forest. Pada teknik *stacking,* prediksi yang dihasilkan oleh model-model individu digunakan sebagai input untuk model meta, yang kemudian menghasilkan prediksi akhir berdasarkan kombinasi *output* tersebut [29][30]. Sedangkan pada teknik *voting,* hasil prediksi dari berbagai model digabungkan dengan cara memberikan suara mayoritas, di mana prediksi yang paling sering muncul menjadi prediksi akhir [31][32]. Pendekatan *ansambel* ini bertujuan untuk memanfaatkan kekuatan masing-masing model dan menghasilkan prediksi yang lebih stabil, akurat, dan dapat diandalkan dalam menghadapi berbagai variasi data.

6. Parameter

Parameter model yang digunakan dalam penelitian ini adalah random forest, yang melibatkan beberapa parameter penting seperti *n_estimators, max_depth, min_samples_split,* dan *max_features.* Parameter *n_estimators* menentukan jumlah pohon yang akan dibangun dalam hutan acak, sementara *max_depth* mengatur kedalaman maksimal dari setiap pohon keputusan untuk mencegah *overfitting.* Min_samples_split mengontrol jumlah sampel minimum yang diperlukan untuk membagi simpul, dan *max_features* menentukan jumlah fitur yang dipilih secara acak untuk setiap pohon [33][34]. Pengaturan parameter-parameter ini bertujuan untuk mengoptimalkan kinerja model dalam memprediksi cacat perangkat lunak dan menghindari *overfitting,* sekaligus memastikan bahwa model dapat menggeneralisasi dengan baik pada data yang belum terlihat.

7. Ensembling

Ensembling digunakan untuk menggabungkan prediksi dari beberapa model dengan tujuan untuk menghasilkan prediksi yang lebih akurat dan stabil. Dalam penelitian ini, teknik voting dan *stacking* diterapkan untuk mengombinasikan hasil dari model random forest dengan model lainnya. Teknik voting menggabungkan hasil prediksi dari model-model individu dengan memberi suara mayoritas, di mana hasil prediksi yang paling banyak dipilih menjadi prediksi akhir [35]. Sementara itu, *stacking* menggunakan output prediksi dari berbagai model sebagai input untuk model meta, yang kemudian menghasilkan prediksi akhir berdasarkan kombinasi output tersebut. Kedua teknik ini terbukti meningkatkan akurasi prediksi secara keseluruhan dengan mengurangi kelemahan yang ada pada model individu, serta memberikan prediksi yang lebih stabil, akurat, dan dapat diandalkan, sehingga memperkuat kemampuan model dalam menghadapi variasi data.

8. Analisis Model

Analisis model dalam penelitian ini menunjukkan bahwa random forest memberikan kinerja sangat baik dalam CPDP pada dataset AEEEM dan PROMISE, dengan akurasi pada semua dataset, serta hasil evaluasi menggunakan *ROC Curve, Confusion Matrix*, dan *validation curve* yang menunjukkan keseimbangan antara *recall* dan *precision*, kemampuan model dalam membedakan antara kelas cacat dan non-cacat, serta membantu memilih parameter *max_depth* yang optimal untuk meningkatkan akurasi, yang secara keseluruhan mengindikasikan model ini mampu memberikan prediksi yang stabil, akurat, dan dapat diandalkan dalam memprediksi cacat perangkat lunak.

9. Evaluasi Model

Evaluasi model dilakukan pada data uji menggunakan metrik seperti *Akurasi, AUC, Recall, Precision* dan *F1-Score* untuk menilai kinerja prediksi model dalam membedakan antara kelas cacat dan non-cacat. Metrik ini memberikan gambaran menyeluruh mengenai kemampuan model dalam mengenali pola cacat perangkat lunak serta keefektifan model dalam mengklasifikasikan data yang belum pernah dilihat [36][37]. Evaluasi ini menunjukkan kekuatan dan kelemahan model dalam hal stabilitas dan akurasi prediksi. Hasil evaluasi dari model random forest yang diterapkan pada berbagai dataset menunjukkan kinerja yang sangat baik, dengan model mampu menyeimbangkan *recall* dan *precision* secara efektif. Pendekatan *ensemble* yang digunakan dalam penelitian ini memberikan hasil yang lebih stabil dan akurat,

memperkuat kemampuan model dalam memprediksi cacat perangkat lunak dan meningkatkan generalisasi pada dataset yang berbeda.

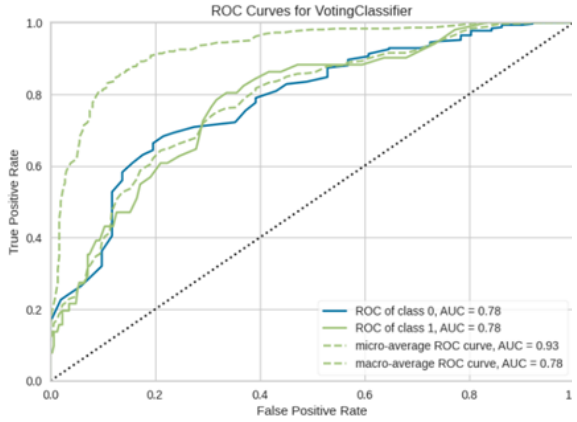
3. Hasil dan Pembahasan

Penelitian ini untuk mengevaluasi dan menganalisis model Random Forest dalam Cross-Project Defect Prediction menggunakan dataset AEEEM dan PROMISE, yang masing-masing terdiri dari berbagai proyek perangkat lunak dengan tingkat cacat yang bervariasi. Model Random Forest dipilih karena kemampuannya dalam menangani masalah prediksi dengan banyak fitur serta kemampuan stabilitasnya dalam mengatasi overfitting berkat teknik ensemble. Selama model diuji menggunakan metrik evaluasi seperti Accuracy, AUC, Recall, Precision, dan F1-Score. Hasil model Random Forest memberikan performa yang sangat baik, dengan akurasi lebih dari 94% pada seluruh dataset yang digunakan. Dataset poi3.0 performa terbaik dengan akurasi 94,52%, diikuti oleh dataset xalan2.6 dan synapse1.2, yang juga AUC lebih dari 0,94, bahwa model mampu membedakan dengan baik antara kelas cacat dan non-cacat. Metrik AUC yang tinggi pada dataset poi3.0 0,9286 mengindikasikan bahwa model memiliki kemampuan yang sangat baik dalam membedakan antara dua kelas yang tidak seimbang, yaitu cacat dan non-cacat.

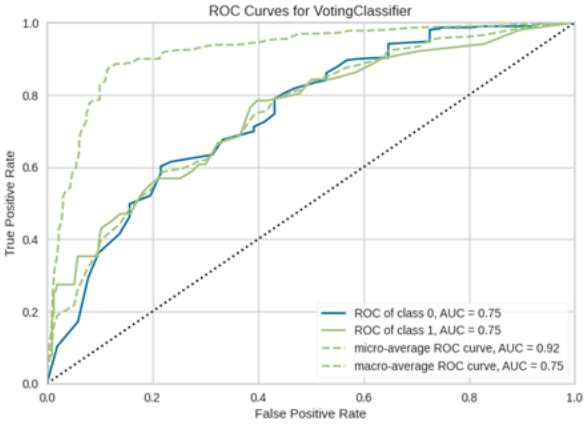
Metrik lainnya, seperti recall, kemampuan model dalam mendeteksi cacat perangkat lunak dengan baik. Pada dataset ivy2.0, model memperoleh recall sebesar 63,64%, yang berarti model dapat menangkap sebagian besar instansi cacat yang ada. Namun, meskipun nilai recall cukup baik, pada beberapa dataset lainnya, Precision dan F1-Score menunjukkan bahwa model berhasil menjaga keseimbangan antara false positive dan true positive, mengurangi kesalahan dalam prediksi cacat. Dataset poi3.0 memperoleh Precision yang sangat baik, yaitu 93,20%, yang menunjukkan bahwa sebagian besar prediksi cacat yang dibuat oleh model adalah benar. F1-Score, yang menggabungkan precision dan recall, menunjukkan hasil yang seimbang, dengan dataset JDT menghasilkan F1-Score sebesar 0,84, mencerminkan keseimbangan antara kemampuan model dalam mendeteksi cacat dan menghindari kesalahan prediksi positif palsu. Meski demikian, dataset LC memiliki F1-Score yang lebih rendah 0,73, yang menunjukkan bahwa model sedikit kesulitan dalam menangani data yang lebih seimbang atau lebih bervariasi, meskipun secara keseluruhan model tetap menunjukkan kinerja yang baik.

Evaluasi kinerja model dilakukan menggunakan confusion matrix dan ROC Curve. Confusion Matrix yang ditampilkan memberikan informasi mendalam mengenai True Positive, True Negative, False Positive, dan False Negative, yang membantu menganalisis performa model dalam memprediksi kelas yang benar dibandingkan dengan kelas yang salah. Semakin tinggi nilai TP dan semakin rendah nilai FP, semakin baik performa model dalam memprediksi cacat perangkat lunak. Di sisi lain, ROC Curve menunjukkan seberapa baik model dalam membedakan antara kelas cacat dan non-cacat, dengan AUC yang lebih tinggi mengindikasikan model yang lebih efektif dalam klasifikasi. Namun, penelitian ini juga mengidentifikasi tantangan terkait dengan outliers pada beberapa dataset, seperti pada LC, yang menunjukkan data yang mungkin tidak terwakili dengan baik oleh fitur yang ada. Outliers ini perlu dianalisis lebih lanjut karena dapat mempengaruhi stabilitas dan akurasi model. Untuk mengatasi masalah ini, penerapan teknik deteksi outliers yang lebih baik sangat diperlukan untuk meningkatkan kualitas prediksi dan stabilitas model.

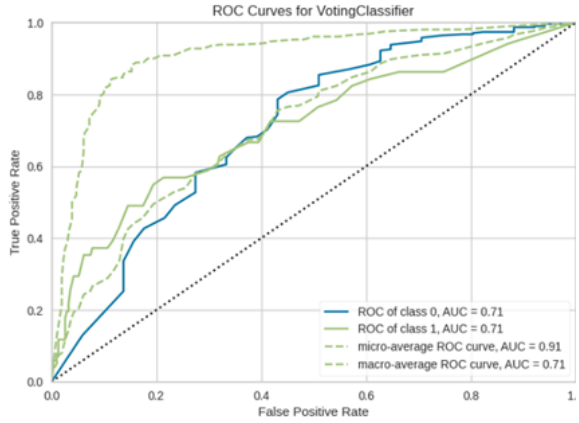
Secara keseluruhan, penelitian ini menunjukkan bahwa model Random Forest sangat efektif dalam memprediksi cacat perangkat lunak lintas proyek, dengan kinerja yang sangat baik pada berbagai dataset. Meskipun ada tantangan terkait dengan distribusi data yang tidak seimbang dan outliers, teknik ensemble seperti stacking dan voting yang digunakan dalam penelitian ini terbukti meningkatkan akurasi dan kestabilan prediksi, serta memberikan kontribusi penting dalam pengembangan model CPDP yang lebih efisien.



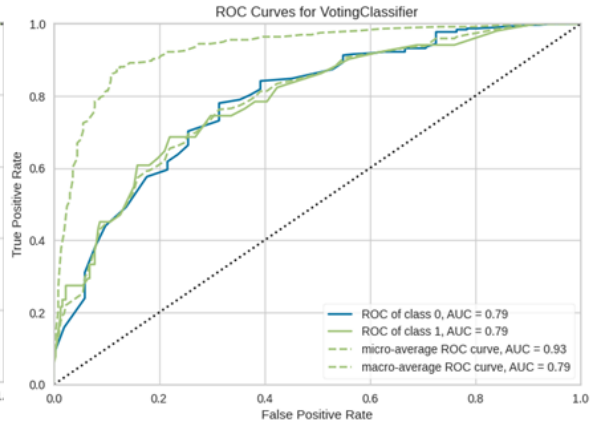
(a) Kurva ROC (ivy2.0)



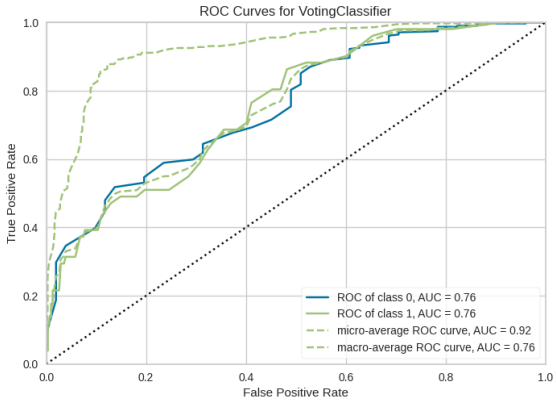
(b) Kurva ROC (poi3.0)



(c) Kurva ROC (synapse1.2)



(d) Kurva ROC (xalan2.6)



(e) Kurva ROC (xerces1.4)

Sumber : Hasil Penelitian (2025)

Gambar 2 ROC Curve Comparison

Gambar 2 kurva ROC dari model voting classifier yang diterapkan pada lima dataset berbeda, yaitu ivy2.0, poi3.0, synapse1.2, xalan2.6, dan xerces1.4. Kurva ROC digunakan untuk mengukur kinerja model klasifikasi, khususnya dalam hal trade-off antara True Positive Rate (TPR) dan False Positive Rate (FPR). Gambar menunjukkan AUC untuk setiap kelas dan rata-rata AUC untuk model yang digunakan. Setiap sub-gambar (a-d) menunjukkan kurva ROC untuk dataset yang berbeda, sedangkan sub-gambar (e) menunjukkan perbandingan ROC secara keseluruhan. Nilai AUC yang lebih tinggi menunjukkan model yang lebih baik dalam klasifikasi, di mana AUC yang lebih besar mengindikasikan kemampuan model yang lebih baik dalam membedakan antara kelas positif dan negatif. Seperti yang terlihat pada gambar, dataset poi3.0 dan xalan2.6 menunjukkan AUC yang lebih tinggi, menandakan performa klasifikasi yang lebih baik dibandingkan dengan dataset lainnya.

Tabel 2 Hasil Prediksi

Dataset	Accuracy	AUC	Recall	Precision	F1
ivy2.0	0.94	**0.94**	0.63	0.93	0.75
poi3.0	0.94	0.92	**0.65**	0.93	**0.76**
synapse1.2	0.94	0.92	0.63	0.91	0.74
xalan2.6	**0.94**	0.94	0.64	**0.95**	0.76
xerces1.4	**0.94**	0.93	0.65	0.93	0.76

Sumber : Hasil Penelitian (2025)

Tabel 2 Hasil evaluasi model random forest pada berbagai dataset dengan metrik *Accuracy, AUC, Recall, Precision* dan *F1 Score.* Model ini menunjukkan kinerja yang sangat baik dengan akurasi lebih dari 94% pada semua dataset. Dataset poi3.0 memiliki akurasi tertinggi (94,52%) dan *AUC* 0,9286, sementara synapse1.2 dan xalan2.6 juga menunjukkan hasil yang kuat dengan *AUC* lebih dari 0,94. Secara keseluruhan, model random forest berhasil menyeimbangkan *recall* dan *precision* dengan baik, serta menunjukkan kemampuan yang kuat dalam memprediksi cacat pada berbagai dataset.

4. Kesimpulan

Penelitian ini mengevaluasi kinerja model random forest dalam CPDP menggunakan dataset AEEEM dan PROMISE, yang terdiri dari berbagai proyek perangkat lunak dengan tingkat cacat yang bervariasi. Hasil evaluasi menunjukkan bahwa model random forest memberikan performa yang sangat baik, dengan akurasi lebih dari 94% pada seluruh dataset yang digunakan. Dataset poi3.0 menunjukkan akurasi tertinggi (94,52%) dan AUC 0,9286, yang menandakan kemampuan model dalam membedakan kelas cacat dan non-cacat dengan sangat baik. Metrik Recall, Precision, dan F1- Score juga menunjukkan bahwa model sangat efektif dalam mendeteksi cacat perangkat lunak, dengan F1-Score tertinggi pada dataset poi3.0 (0,7697). Evaluasi menggunakan confusion matrix dan ROC Curve mengonfirmasi bahwa model mampu meminimalkan false positive dan false negative, serta memberikan hasil yang sangat baik dalam klasifikasi cacat. Meskipun demikian, penelitian ini mengidentifikasi adanya tantangan terkait outliers pada beberapa dataset, seperti pada LC, yang perlu dianalisis lebih lanjut dengan menggunakan teknik deteksi outlier untuk meningkatkan akurasi model. Selain itu, penerapan teknik ensemble methods seperti stacking dan voting terbukti meningkatkan akurasi dan kestabilan prediksi, dan penggunaan teknik lebih canggih seperti deep learning atau ensemble methods yang lebih kompleks dapat lebih mengoptimalkan kinerja model. Hyperparameter tuning yang dilakukan dengan grid search untuk menyesuaikan parameter penting seperti max_depth, n_estimators, dan min_samples_split terbukti vital untuk mengoptimalkan kinerja model. Penelitian ini untuk kedepannya penggunaan dataset yang lebih besar dan lebih beragam untuk pengembangan model lebih lanjut, guna meningkatkan akurasi dan kemampuan generalisasi dalam memprediksi cacat perangkat lunak, yang pada akhirnya dapat mempercepat proses pengambilan keputusan dalam industri perangkat lunak dan mengurangi ketergantungan pada pengujian manual yang memakan waktu dan biaya.

Referensi

[1] K. Javed and and M. A. W. , Ren Shengbing , Muhammad Asim , “Cross-Project Defect Prediction Based on Domain Adaptation and LSTM Optimization,” https://www.mdpi.com/journal/algorithms, 2024, doi: https://doi.org/10.3390/a17050175.

[2] S. Pal and A. Sillitti, “Cross-Project Defect Prediction: A Literature Review,” 2022. doi: 10.1109/ACCESS.2022.3221184.

[3] Y. Zhao, Y. Zhu, Q. Yu, and X. Chen, “Cross-Project Defect Prediction Considering Multiple Data Distribution Simultaneously,” Symmetry (Basel)., vol. 14, no. 2, 2022, doi: 10.3390/sym14020401.

[4] T. Lei, J. Xue, Y. Wang, Z. Niu, Z. Shi, and Y. Zhang, “WCM-WTrA: A Cross-Project Defect Prediction Method Based on Feature Selection and Distance-Weight Transfer Learning,” Chinese J. Electron., vol. 31, no. 2, 2022, doi: 10.1049/cje.2021.00.119.

[5] J. Zou, Z. Li, X. Liu, and H. Tong, “MSCPDPLab: A MATLAB toolbox for transfer learning based multi-source cross-project defect prediction,” SoftwareX, vol. 21, 2023, doi: 10.1016/j.softx.2022.101286.

[6] Y. Zhao, Y. Zhu, Q. Yu, and X. Chen, “Cross-project defect prediction method based on manifold feature transformation,” Futur. Internet, vol. 13, no. 8, 2021, doi: 10.3390/fi13080216.

[7] Y. Khatri and S. K. Singh, “Cross project defect prediction: a comprehensive survey with its SWOT analysis,” Innov. Syst. Softw. Eng., vol. 18, no. 2, 2022, doi: 10.1007/s11334-020- 00380-5.

[8] K. K. Bejjanki, S. P. Kanchanapally, and M. K. Thota, “Class Imbalance Reduction and Centroid based Relevant Project Selection for Cross Project Defect Prediction,” Int. J. Recent Innov. Trends Comput. Commun., vol. 11, no. 6 s, 2023, doi: 10.17762/ijritcc.v11i6s.6933.

[9] A. Agrawal and R. Malhotra, “Cross project defect prediction for open source software,” Int. J. Inf. Technol., vol. 14, no. 1, 2022, doi: 10.1007/s41870-019-00299-6.

[10] Y. Sun et al., “Unsupervised domain adaptation based on discriminative subspace learning for cross-project defect prediction,” Comput. Mater. Contin., vol. 68, no. 3, 2021, doi: 10.32604/cmc.2021.016539.

[11] B. L. Sinaga, S. Ahmad, Z. A. Abas, and I. E. A. Jalil, “A recommendation system of training data selection method for cross-project defect prediction,” Indones. J. Electr. Eng. Comput. Sci., vol. 27, no. 2, 2022, doi: 10.11591/ijeecs.v27.i2.pp990-1006.

[12] Z. Li, H. Zhang, X. Y. Jing, J. Xie, M. Guo, and J. Ren, “DSSDPP: Data Selection and Sampling Based Domain Programming Predictor for Cross-Project Defect Prediction,” IEEE Trans. Softw. Eng., vol. 49, no. 4, 2023, doi: 10.1109/TSE.2022.3204589.

[13] B. Sotto-Mayor and M. Kalech, “Cross-project smell-based defect prediction,” Soft Comput., vol. 25, no. 22, 2021, doi: 10.1007/s00500-021-06254-7.

[14] F. Wu, X. Zheng, Y. Sun, Y. Gao, and X. Y. Jing, “Joint Domain Adaption and Pseudo-Labeling for Cross-Project Defect Prediction,” IEICE Trans. Inf. Syst., vol. E105D, no. 2, 2022, doi: 10.1587/transinf.2021EDL8061.

[15] B. Umamaheswara Sharma and R. Sadam, “How far does the predictive decision impact the software project? The cost, service time, and failure analysis from a cross-project defect prediction model,” J. Syst. Softw., vol. 195, 2023, doi: 10.1016/j.jss.2022.111522.

[16] J. Bai, J. Jia, and L. F. Capretz, “A three-stage transfer learning framework for multi-source cross-project software defect prediction,” Inf. Softw. Technol., vol. 150, 2022, doi: 10.1016/j.infsof.2022.106985.

[17] S. Noreen, R. Bin Faiz, S. Alyahya, and M. Maddeh, “Performance Evaluation of Convolutional Neural Network for Multi-Class in Cross Project Defect Prediction,” Appl. Sci., vol. 12, no. 23, 2022, doi: 10.3390/app122312269.

[18] S. Tang, S. Huang, C. Zheng, E. Liu, C. Zong, and Y. Ding, “A novel cross-project software defect prediction algorithm based on transfer learning,” Tsinghua Sci. Technol., vol. 27, no. 1, 2022, doi: 10.26599/TST.2020.9010040.

[19] A. Jalil, R. Bin Faiz, S. Alyahya, and M. Maddeh, “Impact of Optimal Feature Selection Using Hybrid Method for a Multiclass Problem in Cross Project Defect Prediction,” Appl. Sci., vol. 12, no. 23, 2022, doi: 10.3390/app122312167.

[20] Y. Li, M. Wen, Z. Liu, and H. Zhang, “Using Cost-cognitive Bagging Ensemble to Improve Cross-project Defects Prediction,” J. Internet Technol., vol. 23, no. 4, 2022, doi: 10.53106/160792642022072304013.

[21] Y. Xing, X. Qian, Y. Guan, B. Yang, and Y. Zhang, “Cross-project defect prediction based on G-LSTM model,” Pattern Recognit. Lett., vol. 160, 2022, doi: 10.1016/j.patrec.2022.04.039.

[22] H. Tong, W. Lu, W. Xing, and S. Wang, “ARRAY: Adaptive triple feature-weighted transfer Naive Bayes for cross-project defect prediction,” J. Syst. Softw., vol. 202, 2023, doi: 10.1016/j.jss.2023.111721.

[23] M. M. Ozturk, “Complexfuzzy: Novel Clustering Method for Selecting Training Instances of Cross-Project Defect Prediction,” Comput. Sci., vol. 22, no. 1, 2021, doi: 10.7494/csci.2021.22.1.3743.

[24] X. Zong, G. Li, S. Zheng, H. Zou, H. Yu, and S. Gao, “Heterogeneous Cross-Project Defect Prediction via Optimal Transport,” IEEE Access, vol. 11, 2023, doi: 10.1109/ACCESS.2023.3241924.
[25] R. Haque, A. Ali, S. Mcclean, I. Cleland, and J. Noppen, “Heterogeneous Cross-Project Defect Prediction Using Encoder Networks and Transfer Learning,” IEEE Access, vol. 12, 2024, doi: 10.1109/ACCESS.2023.3343329.
[26] Y. Z. Bala, P. Abdul Samat, K. Y. Sharif, and N. Manshor, “Improving Cross-Project Software Defect Prediction Method Through Transformation and Feature Selection Approach,” IEEE Access, vol. 11, 2023, doi: 10.1109/ACCESS.2022.3231456.
[27] S. Amasaki, H. Aman, and T. Yokogawa, “An extended study on applicability and performance of homogeneous cross-project defect prediction approaches under homogeneous cross-company effort estimation situation,” Empir. Softw. Eng., vol. 27, no. 2, 2022, doi: 10.1007/s10664-021-10103-4.
[28] J. Wu, Y. Wu, N. Niu, and M. Zhou, “MHCPDP: multi-source heterogeneous cross-project defect prediction via multi-source transfer learning and autoencoder,” Softw. Qual. J., vol. 29, no. 2, 2021, doi: 10.1007/s11219-021-09553-2.
[29] Q. Zou, L. Lu, Z. Yang, X. Gu, and S. Qiu, “Joint feature representation learning and progressive distribution matching for cross-project defect prediction,” Inf. Softw. Technol., vol. 137, 2021, doi: 10.1016/j.infsof.2021.106588.
[30] O. P. Omondiagbe, S. A. Licorish, and S. G. MacDonell, “Improving transfer learning for software cross-project defect prediction,” Appl. Intell., vol. 54, no. 7, 2024, doi: 10.1007/s10489-024-05459-1.
[31] F. Zeng, W. Lin, Y. Xing, L. Sun, and B. Yang, “A Cross-project Defect Prediction Model Using Feature Transfer and Ensemble Learning,” Teh. Vjesn., vol. 29, no. 4, 2022, doi: 10.17559/TV-20220421110027.
[32] W. Wen et al., “A Cross-Project Defect Prediction Model Based on Deep Learning With Self-Attention,” IEEE Access, vol. 10, 2022, doi: 10.1109/ACCESS.2022.3214536.
[33] H. Song, G. Wu, L. Ma, Y. Pan, Q. Huang, and S. Jiang, “Adversarial domain adaptation for cross-project defect prediction,” Empir. Softw. Eng., vol. 28, no. 5, 2023, doi: 10.1007/s10664-023-10371-2.
[34] C. Cui, B. Liu, and S. Wang, “WIFLF: An approach independent of the target project for cross-project defect prediction,” J. Softw. Evol. Process, vol. 34, no. 12, 2022, doi: 10.1002/smr.2497.
[35] C. Ni, X. Xia, D. Lo, X. Chen, and Q. Gu, “Revisiting Supervised and Unsupervised Methods for Effort-Aware Cross-Project Defect Prediction,” IEEE Trans. Softw. Eng., vol. 48, no. 3, 2022, doi: 10.1109/TSE.2020.3001739.
[36] L. Goel, M. Sharma, S. K. Khatri, and D. Damodaran, “An empirical analysis of the statistical learning models for different categories of cross-project defect prediction,” Int. J. Comput. Aided Eng. Technol., vol. 14, no. 2, 2021, doi: 10.1504/IJCAET.2021.113549.
[37] R. Malhotra and S. Meena, “Empirical validation of feature selection techniques for cross- project defect prediction,” Int. J. Syst. Assur. Eng. Manag., vol. 15, no. 5, 2024, doi: 10.1007/s13198-023-02051-7.