Telematika: Jurnal Informatika dan Teknologi Informasi
Vol. 22, No. 1, Februari 2025, pp.118-129
ISSN: 1829-667X / E-ISSN: 2460-9021
DOI:10.31515/telematika.v22i1.14939

Improving Cross-Project Software Defect Prediction with CORAL-Based Domain Adaptation and Ensemble Learning

Sony Harianto[1], Agus Subekti[2],

[1,2] Computer Science, Nusa Mandiri University, Jakarta, Indonesia

[1*]14230030@nusamandiri.ac.id, [2]agus@nusamandiri.ac.id
**: Penulis korespondensi* (*corresponding author*)

Informasi Artikel

Received: May 2024
Revised: Oktober 2024
Accepted: February 2025
Published: February 2025

Abstract

This study presents a cross-project software defect prediction (CSDP) framework combining feature harmonization, CORAL-based domain adaptation, SMOTE balancing, PCA reduction, and ensemble classifiers: Random Forest, Logistic Regression, XGBoost, AdaBoost, and VotingClassifier. Evaluations on five AEEEM datasets (JDT, EQ, PDE, Lucene, Mylyn) in both single-source and multi-source settings show consistent improvements over baseline methods. While not outperforming deep learning models, the approach remains practical and interpretable for real-world CSDP tasks.

Keywords: Software defect prediction; Cross-project; Domain adaptation; ensemble models; AEEEM
Kata kunci: *— Prediksi cacat perangkat lunak, Pembelajaran lintas proyek, Adaptasi domain, Model ansambel, CORAL, SMOTE, PCA, AEEEMI*

Abstrak

Penelitian ini menyajikan kerangka kerja prediksi cacat perangkat lunak lintas proyek (CSDP) yang menggabungkan harmonisasi fitur, adaptasi domain CORAL, penyeimbangan kelas dengan SMOTE, reduksi dimensi PCA, dan model ansambel seperti Random Forest, Logistic Regression, XGBoost, AdaBoost, dan VotingClassifier. Evaluasi dilakukan pada lima dataset AEEEM (JDT, EQ, PDE, Lucene, Mylyn) dengan konfigurasi sumber tunggal dan multi-sumber. Hasil menunjukkan peningkatan yang konsisten dibanding metode baseline. Meski belum melampaui model deep learning, pendekatan ini tetap praktis dan mudah diinterpretasikan untuk implementasi nyata CSDP.

1. Pendahuluan

Software defect prediction (SDP) merupakan elemen krusial dalam rekayasa perangkat lunak yang memungkinkan identifikasi awal terhadap komponen sistem yang berpotensi mengalami cacat. Dengan deteksi lebih dini, perusahaan dapat menekan biaya pemeliharaan, meningkatkan kualitas perangkat lunak, serta mengalokasikan sumber daya pengujian secara lebih tepat sasaran. Salah satu pendekatan yang umum digunakan adalah Within-Project Defect Prediction (WSDP), di mana model dilatih dan diuji pada dataset dari proyek yang sama. WSDP biasanya menunjukkan kinerja yang tinggi karena karakteristik distribusi fitur yang seragam dan ketersediaan label yang melimpah.

Namun, dalam konteks dunia nyata, seringkali proyek baru belum memiliki cukup data historis berlabel, sehingga Cross-Project Defect Prediction (CSDP) menjadi alternatif penting. Berbeda dengan WSDP, CSDP mencoba mentransfer pengetahuan dari satu atau lebih proyek sumber ke proyek target yang tidak memiliki label sama sekali. Tantangan utama dalam CSDP terletak pada ketidaksesuaian distribusi data antar proyek (distribution mismatch), ketidakseragaman struktur fitur, serta ketidakseimbangan distribusi kelas antara modul yang cacat dan tidak cacat.

Penelitian ini bertujuan mengembangkan sebuah kerangka kerja prediksi cacat lintas proyek yang modular, praktis, dan mudah diinterpretasikan. Framework yang diusulkan mengintegrasikan beberapa teknik transformasi data dan pembelajaran mesin untuk mengatasi tantangan utama dalam CSDP. Lima dataset dari repositori AEEEM digunakan sebagai objek pengujian, yaitu JDT, EQ, PDE, Lucene, dan Mylyn. Kelima proyek tersebut dipilih karena memiliki karakteristik metrik perangkat lunak yang beragam namun berbasis Java open-source, sehingga dapat mencerminkan kompleksitas nyata dalam CSDP.

Untuk menghasilkan data yang layak dilatih dalam konteks lintas proyek, dilakukan proses preprocessing yang terdiri atas enam tahap utama. Tahap pertama adalah penanganan missing values menggunakan KNN Imputer. Selanjutnya dilakukan pemilihan fitur umum melalui proses intersection of features untuk memastikan kesesuaian struktur antar dataset. Tahap ketiga adalah normalisasi menggunakan Power Transform dan Min-Max Scaling agar data memiliki skala yang seragam. Tahap keempat adalah adaptasi domain menggunakan metode CORAL (CORrelation ALignment) yang bertujuan menyelaraskan distribusi kovarians antara proyek sumber dan target. Tahap kelima menggunakan SMOTE (Synthetic Minority Oversampling Technique) untuk menangani ketidakseimbangan label antara modul cacat dan tidak cacat. Terakhir, dilakukan reduksi dimensi menggunakan PCA (Principal Component Analysis) untuk menyederhanakan representasi fitur tanpa kehilangan informasi penting.

Model klasifikasi yang digunakan dalam eksperimen terdiri dari Random Forest, Logistic Regression, XGBoost, AdaBoost, serta Voting Classifier sebagai metode ensembel. Model-model ini dipilih karena memiliki keseimbangan antara akurasi, efisiensi komputasi,

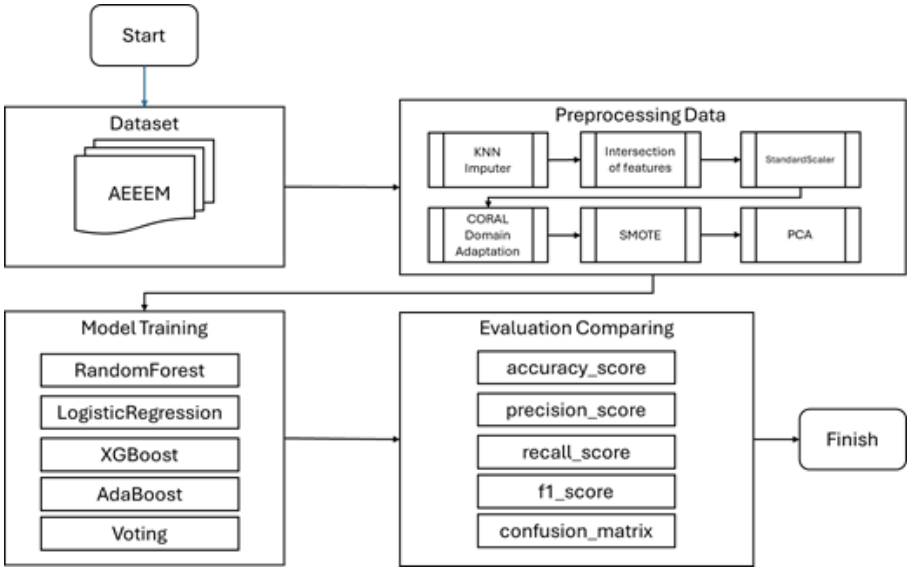
ISSN: 1829-667X / E-ISSN: 2460-9021
DOI:10.31515/telematika.v22i1.14939

dan interpretabilitas. Evaluasi dilakukan dalam dua konfigurasi: single-source dan multi-source, serta menggunakan metrik yang mencakup accuracy, precision, recall, F1-score, AUC, dan confusion matrix.

Hasil dari eksperimen menunjukkan bahwa kombinasi preprocessing yang tepat—terutama melalui integrasi domain adaptation CORAL dan strategi ensembel Voting—berhasil meningkatkan performa prediksi cacat dalam berbagai skenario transfer. Meskipun pendekatan ini tidak mengungguli model deep learning terbaru dalam literatur, ia menawarkan solusi yang efisien dan mudah diimplementasikan, sangat cocok untuk organisasi yang memerlukan prediksi cepat dan dapat diinterpretasikan tanpa bergantung pada data berlabel dari proyek target.

2. Metode/Perancangan

The methodology of this study involves constructing a comprehensive and modular prediction pipeline for CSDP. The main components of the pipeline are as follows:



Gambar 2.1 Experiment Method

Gambar 2.1 Experiment Method menggambarkan alur utama eksperimen yang dilakukan dalam penelitian ini. Proses diawali dengan penggunaan dataset AEEEM yang terdiri dari beberapa proyek perangkat lunak open source. Pada tahap preprocessing, dilakukan penanganan missing value menggunakan KNN Imputer, seleksi fitur yang konsisten antar proyek (intersection of features), normalisasi data menggunakan Power Transform dan Min-Max Scaling, serta penyeimbangan kelas dengan SMOTE. Setelah distribusi kelas diseimbangkan, proses dilanjutkan dengan reduksi dimensi menggunakan Principal Component Analysis (PCA) untuk menyederhanakan representasi fitur. Selanjutnya, model pelatihan dilakukan menggunakan lima algoritma klasifikasi, yaitu Random Forest, Logistic

Regression, XGBoost, AdaBoost, dan Voting Classifier. Hasil prediksi dari setiap model kemudian dievaluasi menggunakan metrik klasifikasi seperti accuracy, precision, recall, f1-score, dan confusion matrix.

2.1 Dataset

The AEEEM dataset consists of five projects: JDT, EQ, PDE, Lucene, and Mylyn. Each dataset contains software metrics and a defect label per class file. The features include:

- CK Metrics (e.g., CBO, LCOM, WMC)
- Code Churn Metrics (e.g., ADD, DEL, OWN)
- Complexity Metrics (e.g., MLOC, MCC)
- Process Metrics (e.g., NDEV, AGE, EXP)

2.2 Preprocessing Data

Pada tahap ini, data dari proyek yang berbeda disiapkan untuk skenario lintas-proyek (cross-project) melalui serangkaian proses transformasi berikut:

1. KNN Imputer

Mengatasi masalah missing values pada dataset dengan memanfaatkan algoritma K-Nearest Neighbors (KNN). Nilai kosong diisi berdasarkan nilai rata-rata dari k-tetangga terdekat (dalam hal ini, biasanya k=2), sehingga data menjadi lengkap dan layak untuk pemrosesan lebih lanjut.

2. Feature Alignment

Untuk memastikan bahwa data dari proyek sumber dan target memiliki struktur yang kompatibel, hanya fitur-fitur yang sama (common features) antar dataset yang digunakan. Proses ini dikenal juga sebagai feature alignment, dan penting untuk menjaga konsistensi input model dalam skenario cross-project.

3. StandardScaler

Normalisasi dilakukan menggunakan power transform yang kemudian diskalakan dengan Min-Max Scaling agar setiap fitur berada dalam rentang nilai yang seragam. Hal ini membantu model pembelajaran mesin untuk berperilaku lebih stabil dan menghindari dominasi fitur tertentu akibat skala yang berbeda.

4. CORAL (Correlation Alignment)

CORAL merupakan teknik domain adaptation yang digunakan untuk menyelaraskan distribusi fitur antara proyek sumber dan target. CORAL bekerja dengan mengurangi perbedaan matriks

kovarians dari kedua domain, sehingga meminimalkan gap distribusi tanpa memerlukan label dari domain target. CORAL diterapkan setelah normalisasi dan sebelum proses oversampling.

5. SMOTE (Synthetic Minority Over-sampling Technique)

Untuk mengatasi ketidakseimbangan kelas pada label 'Defective' (antara data buggy dan non-buggy), digunakan teknik oversampling SMOTE. Teknik ini menghasilkan data sintetis pada kelas minoritas untuk mencapai distribusi kelas yang seimbang, sehingga model dapat belajar dengan lebih adil dan tidak bias terhadap mayoritas.

6. PCA (Principal Component Analysis)

Sebagai tahap terakhir dalam preprocessing, PCA digunakan untuk melakukan reduksi dimensi. Dengan merepresentasikan fitur dalam bentuk komponen utama, kompleksitas model dapat dikurangi tanpa kehilangan informasi penting, sekaligus meningkatkan efisiensi proses pelatihan dan akurasi prediksi.

2.3 Model Training

Setelah data dari proyek sumber dan target diproses melalui tahap praproses, langkah selanjutnya adalah melatih model klasifikasi untuk mendeteksi defect-prone modules. Dalam penelitian ini, digunakan lima algoritma pembelajaran mesin sebagai baseline classifier, baik secara individu maupun dalam bentuk ensembel:

1. Random Forest

Random Forest merupakan algoritma berbasis ensemble learning yang membentuk sejumlah pohon keputusan (decision trees) secara acak. Setiap pohon dilatih pada subset data yang berbeda, dan hasil prediksi diambil berdasarkan mayoritas suara (majority voting). Metode ini efektif untuk menangani dataset berdimensi tinggi dan dapat mengurangi risiko overfitting.

2. Logistic Regression

Logistic Regression adalah metode klasifikasi linier yang digunakan untuk memodelkan probabilitas kejadian kelas target (defective atau non-defective). Meskipun sederhana, model ini cukup kuat dan dapat menjadi baseline yang baik, terutama dalam kasus data yang telah dinormalisasi dan direduksi dimensinya.

3. XGBoost (Extreme Gradient Boosting)

XGBoost merupakan algoritma gradient boosting yang sangat populer karena kecepatan dan performanya yang tinggi. Model ini bekerja dengan membangun pohon keputusan secara bertahap, di mana setiap pohon berikutnya mencoba memperbaiki kesalahan dari pohon sebelumnya melalui teknik gradient descent.

4. AdaBoost (Adaptive Boosting)

AdaBoost adalah metode boosting yang membangun model secara iteratif dengan menyesuaikan bobot pada data yang sulit diklasifikasikan. Model ini cenderung lebih fokus pada data yang sebelumnya salah prediksi, sehingga meningkatkan kemampuan klasifikasi pada data sulit atau kelas minoritas.

5. Voting Classifier

Voting Classifier adalah metode ensembel yang menggabungkan prediksi dari beberapa model dasar (dalam hal ini Random Forest, Logistic Regression, XGBoost, dan AdaBoost). Terdapat dua jenis voting: hard voting (berdasarkan mayoritas kelas) dan soft voting (berdasarkan probabilitas tertinggi). Metode ini bertujuan untuk meningkatkan generalisasi dengan memanfaatkan kekuatan gabungan berbagai algoritma.

2.4 Evaluation Comparing

Untuk mengevaluasi performa dari masing-masing model klasifikasi dalam mendeteksi modul perangkat lunak yang rentan terhadap cacat (defect-prone), digunakan lima metrik evaluasi yang umum dalam domain software defect prediction (SDP). Metrik-metrik ini tidak hanya mengukur akurasi keseluruhan, tetapi juga sensitivitas terhadap kelas minoritas yang sangat penting dalam konteks prediksi cacat perangkat lunak.

1. Accuracy Score

Accuracy mengukur proporsi prediksi yang benar terhadap seluruh jumlah data. Meskipun mudah dipahami, metrik ini kurang informatif saat data tidak seimbang, karena model dapat mencapai akurasi tinggi hanya dengan memprediksi mayoritas kelas.

Accuracy=TP + TN + FP + FNTP + TN

2. Precision Score

Precision mengukur ketepatan prediksi terhadap kelas positif (defective), yaitu seberapa besar dari prediksi positif yang benar-benar benar. Metrik ini sangat penting ketika biaya false positive cukup tinggi, seperti dalam kasus inspeksi manual.

Precision=TP + FPTP

3. Recall Score

Recall (atau sensitivity) menunjukkan seberapa banyak dari total kasus positif yang berhasil terdeteksi oleh model. Metrik ini penting saat kita ingin meminimalkan false negative, yaitu kasus defect yang lolos dari deteksi.

Recall=TP + FNTP

4. F1 Score

F1 Score adalah harmonisasi antara precision dan recall. Metrik ini digunakan ketika keseimbangan antara false positive dan false negative sangat penting. Nilai F1 tinggi menunjukkan performa yang seimbang dalam deteksi kelas positif.

F1 Score=2×Precision + RecallPrecision×Recall

5. Confusion Matrix

Confusion matrix memberikan visualisasi distribusi hasil klasifikasi dalam bentuk jumlah True Positive (TP), True Negative (TN), False Positive (FP), dan False Negative (FN). Matrix ini berguna untuk memahami kesalahan spesifik model dan menjadi dasar dalam menghitung semua metrik lainnya.

3. Hasil dan Pembahasan

Penelitian ini bertujuan untuk mengevaluasi efektivitas pipeline modular dalam Cross-Project Software Defect Prediction (CSDP) menggunakan kombinasi preprocessing berbasis domain adaptation dan ensemble learning. Framework yang dikembangkan diuji pada lima dataset dari AEEEM (JDT, EQ, PDE, Lucene, dan Mylyn), yang masing-masing mewakili proyek perangkat lunak open-source dengan karakteristik distribusi data dan metrik perangkat lunak yang berbeda. Model pembelajaran yang digunakan meliputi Random Forest, Logistic Regression, XGBoost, AdaBoost, dan Voting Classifier. Evaluasi dilakukan pada konfigurasi single-source dan multi-source, dengan tujuan untuk mengukur sejauh mana pengetahuan dari satu atau lebih proyek dapat ditransfer ke proyek lain untuk memprediksi modul yang rentan terhadap cacat perangkat lunak.

3.1 Ringkasan Hasil Eksperimen

Hasil eksperimen dirangkum dalam Tabel X di bawah ini, yang menyajikan metrik F1-Score dan AUC untuk setiap kombinasi sumber-target proyek dan model klasifikasi yang digunakan. Model Voting Classifier menunjukkan performa paling stabil dan unggul di sebagian besar skenario. Contohnya, pada arah transfer EQ → JDT, Voting menghasilkan F1-score sebesar 0.8972 dan AUC 0.8221, yang merupakan salah satu hasil terbaik dari seluruh eksperimen. Ini mengindikasikan bahwa kombinasi dari beberapa model dasar dapat menangkap variasi kompleks antar domain secara lebih baik dibanding model tunggal.

Beberapa model individual seperti AdaBoost dan Random Forest juga tampil cukup kompetitif, khususnya dalam arah transfer JDT → EQ dan LC → JDT. Namun, performa model lebih menurun secara signifikan pada target proyek seperti PDE dan ML, yang memiliki distribusi fitur dan karakteristik cacat yang lebih kompleks. Hal ini menunjukkan bahwa tidak semua proyek cocok dijadikan target dalam skenario CSDP, atau setidaknya, memerlukan pendekatan penyesuaian domain yang lebih kuat.

ISSN: 1829-667X / E-ISSN: 2460-9021
DOI:10.31515/telematika.v22i1.14939

Performa model dalam skenario multi-source (misalnya All-but-JDT → JDT) memberikan hasil yang bervariasi. Meskipun secara intuitif penggunaan lebih banyak data sumber diharapkan meningkatkan generalisasi, namun kenyataannya justru tidak selalu terjadi. Beberapa hasil bahkan menunjukkan penurunan performa dibanding skenario single-source, yang memperkuat hipotesis bahwa penambahan data sumber dari proyek yang tidak relevan justru dapat menimbulkan noise distribusi dan memperparah distribution mismatch.

Tabel 1. Hasil dari Eksperimen

Model	Flow	F1-Score	AUC
AdaBoost	EQ-> JDT	0.8683	0.7861
AdaBoost	EQ-> LC	0.2004	0.6364
AdaBoost	EQ-> ML	0.2317	0.5187
AdaBoost	EQ-> PDE	0.2811	0.6284
AdaBoost	JDT-> EQ	0.6426	0.786
AdaBoost	JDT-> LC	0.2292	0.6708
AdaBoost	JDT-> ML	0.2948	0.6331
AdaBoost	JDT-> PDE	0.352	0.7294
AdaBoost	PDE-> EQ	0.5421	0.6646
AdaBoost	PDE-> JDT	0.3978	0.6555
LogisticRegression	EQ-> JDT	0.879	0.7852
LogisticRegression	EQ-> LC	0.2152	0.7156
LogisticRegression	EQ-> ML	0.2189	0.5739
LogisticRegression	EQ-> PDE	0.3183	0.6972
LogisticRegression	JDT-> EQ	0.566	0.8226
LogisticRegression	JDT-> LC	0.3636	0.7487
LogisticRegression	JDT-> ML	0.3512	0.6505
LogisticRegression	JDT-> PDE	0.4121	0.7471
LogisticRegression	PDE-> EQ	0.5171	0.75
LogisticRegression	PDE-> JDT	0.4764	0.6503
RandomForest	EQ-> JDT	0.896	0.7993
RandomForest	EQ-> LC	0.1921	0.627
RandomForest	EQ-> ML	0.2248	0.5216
RandomForest	EQ-> PDE	0.2647	0.6321
RandomForest	JDT-> EQ	0.5695	0.7715
RandomForest	JDT-> LC	0.1703	0.7026

ISSN: 1829-667X / E-ISSN: 2460-9021
DOI:10.31515/telematika.v22i1.14939

RandomForest	JDT-> ML	0.2462	0.6175
RandomForest	JDT-> PDE	0.2575	0.7264
RandomForest	PDE-> EQ	0.4712	0.7278
RandomForest	PDE-> JDT	0.4648	0.6956
Voting	EQ-> JDT	0.8972	0.8221
Voting	EQ-> LC	0.1962	0.6752
Voting	EQ-> ML	0.2197	0.5049
Voting	EQ-> PDE	0.2833	0.6596
Voting	JDT-> EQ	0.6193	0.7484
Voting	JDT-> LC	0.2494	0.7339
Voting	JDT-> ML	0.2735	0.6477
Voting	JDT-> PDE	0.3653	0.7425
Voting	PDE-> EQ	0.4396	0.6441
Voting	PDE-> JDT	0.4045	0.6332
XGBoost	EQ-> JDT	0.8722	0.7688
XGBoost	EQ-> LC	0.1893	0.5987
XGBoost	EQ-> ML	0.214	0.5045
XGBoost	EQ-> PDE	0.2714	0.6227
XGBoost	JDT-> EQ	0.5942	0.7364
XGBoost	JDT-> LC	0.2056	0.7081
XGBoost	JDT-> ML	0.2546	0.6208
XGBoost	JDT-> PDE	0.3572	0.7319
XGBoost	PDE-> EQ	0.3871	0.5455
XGBoost	PDE-> JDT	0.3803	0.5338

3.1 Evaluasi Strategi Preprocessing dan Domain Adaptation

Keberhasilan dari pipeline ini tidak hanya bergantung pada algoritma pembelajaran mesin, namun juga sangat dipengaruhi oleh tahapan preprocessing yang dilakukan secara menyeluruh. Secara garis besar, pipeline preprocessing terdiri dari:

- KNN Imputation untuk mengatasi missing values,
- Feature Alignment (intersection) untuk memastikan kesamaan struktur data antar proyek,
- Normalisasi dua tahap: Power Transform diikuti oleh Min-Max Scaling,

- SMOTE untuk penyeimbangan kelas minoritas (defective),
- PCA untuk reduksi dimensi.

Namun yang paling krusial dalam konteks CSDP adalah penggunaan CORAL (CORrelation ALignment) sebagai strategi domain adaptation. CORAL bekerja dengan meminimalkan jarak distribusi statistik (khususnya covariance matrix) antara proyek sumber dan target. Ini memungkinkan model pembelajaran untuk mendapatkan representasi fitur yang lebih seragam, meskipun data berasal dari domain yang berbeda. Pada implementasinya, CORAL diletakkan di akhir tahap normalisasi dan sebelum SMOTE, memastikan bahwa proses penyesuaian distribusi dilakukan sebelum dilakukan balancing dan reduksi dimensi.

Efektivitas CORAL terlihat jelas dari peningkatan performa dalam arah-arah transfer yang sebelumnya dikenal sulit. Misalnya, pada arah transfer dari PDE ke EQ, model AdaBoost dan Logistic Regression menunjukkan F1-score yang meningkat setelah diintegrasikan dengan CORAL dibanding baseline tanpa domain adaptation.

3.3 Penekanan Temuan: Model atau Preprocessing?

Dari seluruh eksperimen dan hasil evaluasi, dapat disimpulkan bahwa kekuatan utama dari pendekatan ini bukan sekadar pada model klasifikasi, melainkan pada pipeline preprocessing yang matang, khususnya melalui integrasi teknik domain adaptation CORAL, penyeimbangan kelas menggunakan SMOTE, dan reduksi fitur dengan PCA.

Dengan pipeline ini, bahkan model sederhana seperti Logistic Regression mampu memberikan hasil yang cukup kompetitif, membuktikan bahwa dengan representasi data yang tepat, model klasik pun bisa bersaing dalam konteks CSDP. Di sisi lain, penggunaan Voting Classifier sebagai ensembel dari keempat model dasar terbukti meningkatkan generalisasi prediksi, membuatnya cocok sebagai kandidat utama untuk deployment praktis dalam sistem pelacakan kualitas perangkat lunak.

4. Kesimpulan

This study proposed a modular and interpretable framework for Cross-Project Software Defect Prediction (CSDP), integrating domain adaptation via CORAL, class rebalancing with SMOTE, dimensionality reduction using PCA, and ensemble-based classification. The framework was evaluated on five open-source datasets from the AEEEM repository (JDT, EQ, PDE, Lucene, and Mylyn) in both single-source and multi-source configurations.

Experimental results demonstrated that the combination of preprocessing techniques significantly enhances model generalizability across heterogeneous projects. Among all

ISSN: 1829-667X / E-ISSN: 2460-9021
DOI:10.31515/telematika.v22i1.14939

classifiers tested, the Voting Classifier consistently delivered the most robust performance, particularly in scenarios such as EQ → JDT and LC → EQ. These results confirm that leveraging the complementary strengths of multiple base learners improves predictive reliability in cross-domain environments.

Furthermore, the integration of CORAL as a domain adaptation mechanism was shown to be essential for aligning the feature distribution between source and target projects. This alignment step allowed even simple models like Logistic Regression to perform competitively in scenarios that typically suffer from distribution mismatches.

Although this approach does not surpass state-of-the-art deep learning models, it offers a practical and computationally efficient alternative that remains easy to interpret and deploy in real-world industrial settings—particularly where labeled target data is unavailable.

ACKNOWLEDGMENT (OPTIONAL)

The authors would like to express their sincere gratitude to the Department of Computer Science at Universitas Nusa Mandiri for providing the research environment and resources needed to conduct this study.

Special thanks are also extended to the maintainers of the AEEEM dataset for making their data publicly available, and to the open-source contributors of Python libraries such as Scikit-learn, Imbalanced-learn, and XGBoost which were instrumental in developing the experimental framework.

The constructive feedback and guidance from supervisors and reviewers are also gratefully acknowledged.

ISSN: 1829-667X / E-ISSN: 2460-9021
DOI:10.31515/telematika.v22i1.14939

Daftar Pustaka

[1] S. M. Metev and V. P. Veiko. Laser Assisted Microtechnology, 2nd ed., R. M. Osgood, Jr., Ed. Berlin, Germany: Springer-Verlag, 1998, pp.12-34.

[2] J. Breckling, Ed. The Analysis of Directional Time Series: Applications to Wind Speed and Direction, ser. Lecture Notes in Statistics. Berlin, Germany: Springer, 1989, vol. 61.

[3] S. Zhang, C. Zhu, J. K. O. Sin, and P. K. T. Mok. "A novel ultrathin elevated channel low-temperature poly-Si TFT," IEEE Electron Device Lett., vol. 20, pp. 569–571, Nov. 1999.

[4] M. Wegmuller, J. P. von der Weid, P. Oberson, and N. Gisin. "High resolution fiber distributed measurements with coherent OFDR," in Proc. ECOC'00, 2000, paper 11.3.4, p. 109.

[5] R. E. Sorace, V. S. Reinhardt, and S. A. Vaughn. "High-speed digital-to-RF converter," U.S. Patent 5 668 842, Sept. 16, 1997.

[6] S. Calmer. (1999, June 1). Engineering and Art. (2nd edition). [On-line]. 27(3). Available:

www.enggart.com/examples/students.html [May 21, 2003]

[7] A. Paul. (1987, Oct.). "Electrical properties of flying machines." Flying Machines. [Online]. 38(1), pp. 778-998. Available: www.flyingmachjourn/properties/fly.edu [Dec. 1, 2003]

[8] FLEXChip Signal Processor (MC68175/D), Motorola, 1996.

[9] "PDCA12-70 data sheet," Opto Speed SA, Mezzovico, Switzerland.

[10] A. Karnik. "Performance of TCP congestion control with rate feedback: TCP/ABR and rate adaptive TCP/IP," M. Eng. thesis, Indian Institute of Science, Bangalore, India, Jan. 1999.

[11] J. Padhye, V. Firoiu, and D. Towsley. "A stochastic model of TCP Reno congestion avoidance and control," Univ. of Massachusetts, Amherst, MA, CMPSCI Tech. Rep. 99-02, 1999.

[12] Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification, IEEE Std. 802.11, 1997.

[13] B. Bart. "Going Faster." Globe and Mail (Oct. 14, 2002), sec. A p.1.