Repeater : Publikasi Teknik Informatika dan Jaringan
Volume. 4 Nomor. 3 Juli 2026
e-ISSN : 3046-7276, p-ISSN : 3046-7284, Hal 63-74
DOI: https://doi.org/10.62951/repeater.v4i3.953
Tersedia: https://journal.aptii.or.id/index.php/Repeater

Prediksi dan Deteksi Bug pada *Visual Studio Code* menggunakan Algoritma *Naive Bayes*

Siti Hardianti[1*], Roberto Kaban[2]
[1]Program Studi Rekayasa Perangkat Lunak, Institut Teknologi dan Bisnis Indonesia, Indonesia
[2]Program Studi Teknik Informatika, Institut Teknologi dan Bisnis Indonesia, Indonesia
Email: sitihardianti71203@gmail.com[1], roberto.kaban@yahoo.com[2]
**Penulis Korespondensi: sitihardianti71203@gmail.com*

Abstract. *Software quality is a critical aspect of modern software engineering. One of the primary challenges developers face is the early detection of bugs before software is released into the production environment. This study develops a bug prediction model using the Naive Bayes algorithm applied to the JM1 dataset from NASA's Metrics Data Program, sourced from Kaggle. The JM1 dataset consists of source code metrics from a NASA project, comprising 10,885 modules with 21 numerical features that include Halstead and McCabe metrics. All experimental stages were conducted using Python with the Pandas, Scikit-learn, NumPy, and Matplotlib libraries. Experimental results show that the Naive Bayes model achieved an accuracy of 79.93%, an ROC-AUC value of 0.6761, precision of 46.26%, recall of 23.52%, and an F1-Score of 31.18%. These findings indicate that while Naive Bayes effectively identifies non-defective modules, it faces challenges in detecting defective modules due to significant class imbalance (80.65% vs. 19.35%). The contributions of this study include an in-depth analysis of the impact of class imbalance on bug prediction performance, as well as recommendations for handling techniques such as SMOTE and ensemble learning to improve future performance.*

Keywords: *Bug Prediction; Machine Learning; Naive Bayes; NASA JM1; Software Defect Prediction.*

Abstrak. Kualitas perangkat lunak merupakan aspek kritis dalam rekayasa perangkat lunak modern. Salah satu tantangan utama yang dihadapi pengembang adalah deteksi bug secara dini sebelum perangkat lunak dirilis ke lingkungan produksi. Penelitian ini mengembangkan model prediksi bug menggunakan algoritma *Naive Bayes* yang diterapkan pada dataset JM1 dari NASA Metrics Data Program yang bersumber dari Kaggle. Dataset JM1 merupakan data metrik kode sumber dari proyek NASA yang terdiri dari 10.885 modul dengan 21 fitur numerik mencakup metrik Halstead dan McCabe. Penelitian ini menggunakan pendekatan kuantitatif dengan pembagian data 80:20 untuk pelatihan dan pengujian. Hasil eksperimen menunjukkan bahwa model *Naive Bayes* mencapai akurasi sebesar 79,93%, nilai ROC-AUC sebesar 0,6761, precision 46,26%, recall 23,52%, dan F1-Score 31,18%. Temuan ini mengindikasikan bahwa *Naive Bayes* mampu mengidentifikasi modul non-cacat dengan baik, namun menghadapi tantangan dalam mendeteksi modul cacat akibat ketidakseimbangan kelas yang signifikan (80,65% vs 19,35%). Kontribusi penelitian ini mencakup analisis mendalam terhadap pengaruh ketidakseimbangan kelas pada performa prediksi bug serta rekomendasi teknik penanganan seperti SMOTE dan ensemble learning untuk peningkatan performa di masa mendatang.

Kata Kunci: *Machine Learning*; *Naive Bayes*; NASA JM1; Prediksi Bug; *Software Defect Prediction*.

1. LATAR BELAKANG

Dalam era transformasi digital yang semakin pesat, kualitas perangkat lunak menjadi penentu utama keberhasilan suatu produk teknologi Krasner (2022) . Bug atau cacat perangkat lunak bukan hanya permasalahan teknis semata, tetapi juga berimplikasi langsung terhadap kerugian ekonomi, reputasi organisasi, dan bahkan keselamatan pengguna. Laporan dari *Consortium for Information and Software Quality* (CISQ) pada tahun 2022 memperkirakan kerugian akibat kualitas perangkat lunak yang buruk di Amerika Serikat mencapai 2,41 triliun dolar per tahun, di mana kegagalan operasional menjadi penyumbang terbesar.

Naskah Masuk: 13 Mei 2026; Revisi: 15 Juni 2026; Diterima: 16 Juli 2026; Terbit: 23 Juli 2026

Visual Studio Code (VS Code) merupakan salah satu *Integrated Development Environment* (IDE) paling populer di dunia saat ini Stack Overflow Developer Survey 2023., digunakan oleh lebih dari 73% pengembang menurut survei *Stack Overflow Developer Survey* 2023. Sebagai proyek *open-source* yang aktif dikembangkan oleh Microsoft dengan ribuan kontributor, VS Code menghadapi tantangan manajemen kualitas kode yang kompleks. Pada repositori GitHub-nya, VS Code secara konsisten memiliki ratusan isu terbuka yang berkaitan dengan bug dan defect, yang memerlukan proses triase dan prioritisasi yang efisien.

Pendekatan tradisional dalam pengujian perangkat lunak bersifat reaktif Yuwono (2025), yakni bug diidentifikasi setelah ditemukan oleh pengguna atau melalui proses pengujian manual yang memakan waktu dan sumber daya signifikan (Abdu et al., 2024a). Paradigma *Software Defect Prediction* (SDP) hadir sebagai solusi proaktif Li et al. (2018)., memungkinkan tim pengembang untuk mengidentifikasi modul kode yang berpotensi mengandung bug bahkan sebelum pengujian intensif dilakukan (Arasteh et al., 2024)

Algoritma *Naive Bayes* telah lama menjadi pilihan populer Hernández-Molinos et al. (2023a), dalam domain klasifikasi teks dan data kategorikal, namun aplikasinya dalam prediksi bug perangkat lunak masih relatif terbatas dibandingkan dengan metode ensemble seperti Random Forest atau Gradient Boosting. Kesederhanaan komputasi, kemampuan interpretasi yang baik Prastyo et al. (2024), dan performa yang kompetitif menjadikan *Naive Bayes* relevan untuk dikaji lebih mendalam, terutama dalam konteks dataset berukuran besar dengan ketidakseimbangan kelas (Hernández-Molinos et al., 2023b).

Visual Studio Code merupakan salah satu lingkungan pengembangan (*Integrated Development Environment*/IDE) yang banyak digunakan oleh pengembang perangkat lunak karena bersifat ringan, mendukung berbagai bahasa pemrograman, dan memiliki ekosistem ekstensi yang luas. Meskipun telah menyediakan berbagai fitur seperti debugging dan code linting, *Visual Studio Code* belum secara bawaan menyediakan fitur prediksi bug berbasis machine learning. Oleh karena itu, penelitian ini mengembangkan model prediksi bug menggunakan algoritma Gaussian *Naive Bayes* yang dapat diimplementasikan dan dijalankan melalui lingkungan *Visual Studio Code* sebagai alat bantu dalam proses pengembangan perangkat lunak.

2. KAJIAN TEORITIS

Software Defect Prediction

Software Defect Prediction (SDP) merupakan disiplin ilmu yang memanfaatkan teknik machine learning Rumbutis et al. (2024).dan data mining untuk memperkirakan kemungkinan keberadaan bug pada komponen perangkat lunak berdasarkan atribut yang dapat diukur dari kode sumber. Konsep ini pertama kali diperkenalkan secara sistematis oleh (Taskeen et al., 2023). yang menunjukkan bahwa distribusi bug dalam perangkat lunak bersifat tidak merata Bowes et al. (2018) di mana sebagian kecil modul cenderung mengandung sebagian besar bug yang ditemukan.

Metrik kode sumber yang umum digunakan sebagai fitur dalam SDP mencakup dua kategori utama: metrik Halstead dan metrik McCabe (Rebro et al., 2023). mengukur kompleksitas perangkat lunak berdasarkan jumlah operator dan operand unik maupun total dalam suatu program. Metrik ini meliputi volume (V), kesulitan (D), upaya (E), dan estimasi jumlah bug (B). Sementara itu, metrik McCabe atau Cyclomatic Complexity yang diperkenalkan Thomas McCabe (1976) mengukur kompleksitas logika suatu program melalui analisis graf alur kontrol.

Algoritma *Naive Bayes*

Naive Bayes merupakan algoritma klasifikasi probabilistik yang didasarkan pada Teorema Bayes dengan asumsi independensi kondisional antar fitur (Hernández-Molinos et al., 2023a). Secara formal, Teorema Bayes dinyatakan sebagai:

$$P(C \mid X) = \frac{P(X|C) \times P(C)}{P(X)} \quad \text{..... (i)}$$

di mana C adalah kelas (defect/non-defect) dan X adalah vektor fitur metrik kode sumber.

Asumsi "*naive*" bahwa setiap fitur bersifat independen satu sama lain, meskipun tidak selalu valid dalam praktik, justru menjadi kekuatan algoritma ini dalam hal efisiensi komputasi dan ketahanan terhadap overfitting (Prastyo et al., 2024). Dalam konteks prediksi bug, Gaussian *Naive Bayes* digunakan karena fitur-fitur metrik kode sumber bersifat kontinu dan dapat dimodelkan dengan distribusi Gaussian.

Penelitian Terdahulu

Melakukan *tertiary study* terhadap berbagai penelitian *Software Defect Prediction* yang dipublikasikan selama beberapa tahun terakhir. Hasil penelitian menunjukkan bahwa hingga saat ini belum terdapat satu algoritma machine learning yang secara konsisten memberikan performa terbaik pada seluruh dataset. Performa model sangat dipengaruhi oleh karakteristik

dataset, teknik prapemrosesan data, serta metrik evaluasi yang digunakan. Temuan tersebut menunjukkan bahwa pemilihan algoritma harus disesuaikan dengan karakteristik data yang dianalisis (Taskeen et al., 2023).

Abdu et al. (2024b) mengusulkan model prediksi cacat perangkat lunak berbasis *hybrid deep learning* dengan menggabungkan fitur semantik dan metrik perangkat lunak. Penelitian tersebut menunjukkan bahwa kualitas fitur dan penanganan ketidakseimbangan kelas memberikan pengaruh yang signifikan terhadap peningkatan performa prediksi, terutama pada dataset yang memiliki distribusi kelas tidak seimbang (Abdu et al., 2024b).

Prastyo et al. (2024) menerapkan algoritma *Naive Bayes* untuk prediksi cacat perangkat lunak dan menunjukkan bahwa algoritma tersebut tetap mampu memberikan hasil klasifikasi yang kompetitif meskipun memiliki mekanisme yang lebih sederhana dibandingkan beberapa algoritma machine learning lainnya. Penelitian tersebut juga menunjukkan bahwa *Naive Bayes* memiliki keunggulan dari sisi efisiensi komputasi dan kemudahan implementasi.

Hernández-Molinos et al. (2023a) melakukan kajian mengenai berbagai pendekatan Bayesian dalam Software Defect Prediction. Hasil penelitian menunjukkan bahwa evaluasi model tidak cukup hanya menggunakan akurasi, tetapi perlu mempertimbangkan metrik lain seperti Precision, Recall, F1-Score, dan ROC-AUC agar kemampuan model dalam mendeteksi modul yang mengandung cacat dapat dievaluasi secara lebih komprehensif.

Perbedaan penelitian ini dari studi-studi terdahulu terletak pada: (1) penggunaan dataset JM1 lengkap dengan 10.885 sampel tanpa pengurangan subset; (2) analisis eksplisit terhadap *trade-off precision-recall* dalam konteks ketidakseimbangan kelas; dan (3) pembahasan implikasi praktis hasil prediksi untuk lingkungan pengembangan VS Code.

3. METODE PENELITIAN

Jenis Penelitian

Penelitian ini menggunakan pendekatan kuantitatif dengan desain eksperimental komputasional. Metode ini dipilih karena melibatkan pengukuran numerik terhadap performa model machine learning dan analisis statistik terhadap hasil yang diperoleh. Eksperimen dilakukan secara sistematis dengan kontrol variabel yang jelas, yaitu pemilihan algoritma, pembagian dataset, dan metrik evaluasi yang terstandarisasi.

Sumber Data

Data yang digunakan adalah dataset JM1 dari NASA Metrics Data Program yang diakses melalui platform Kaggle (Sayyad Shirabad & Menzies, 2005; Kaggle, 2024). (https://www.kaggle.com/datasets/nasa/software-defect-datasets). Dataset ini merupakan data

metrik kode sumber dari proyek perangkat lunak real-time C yang dikembangkan oleh NASA. Dataset JM1 dipilih karena: (1) merupakan benchmark standar dalam komunitas SDP; (2) memiliki ukuran yang relatif besar (10.885 instance); (3) mengandung fitur-fitur metrik kode yang komprehensif; dan (4) telah digunakan secara luas dalam literatur sehingga memungkinkan perbandingan hasil.

Deskripsi Dataset

Dataset JM1 terdiri dari 10.885 modul perangkat lunak dengan 21 fitur numerik dan satu variabel target biner (*defects*). Variabel target menunjukkan apakah suatu modul ditemukan mengandung bug setelah pengujian (True = mengandung bug, False = tidak mengandung bug). Distribusi kelas menunjukkan ketidakseimbangan yang signifikan: 8.779 modul (80,65%) tidak mengandung bug dan 2.106 modul (19,35%) mengandung bug.

Tabel 1. Deskripsi Fitur Dataset JM1.

Fitur	Kategori	Deskripsi
loc	Metrik Ukuran	Lines of Code – total baris kode
v(g)	Metrik McCabe	Cyclomatic Complexity
ev(g)	Metrik McCabe	Essential Complexity
iv(g)	Metrik McCabe	Design Complexity
n	Metrik Halstead	Total jumlah operator dan operand
v	Metrik Halstead	Volume program
l	Metrik Halstead	Program length
d	Metrik Halstead	Difficulty
i	Metrik Halstead	Intelligence content
e	Metrik Halstead	Effort to implement
b	Metrik Halstead	Estimasi jumlah bug
t	Metrik Halstead	Time to implement
lOCode	Metrik LOC	Lines of code (tanpa komentar)
lOComment	Metrik LOC	Lines of comment
branchCount	Metrik Kontrol	Jumlah percabangan
defects	Target	Variabel target (True/False)

Teknik Pengumpulan Data

Data dikumpulkan melalui metode unduhan dataset dari repositori Kaggle yang merupakan sumber data terpercaya untuk penelitian machine learning. Dataset JM1 termasuk dalam koleksi NASA Software Defect Datasets yang dikurasi oleh Promise Software Engineering Repository. Proses pengumpulan data meliputi: (1) identifikasi dan akses dataset melalui Kaggle; (2) verifikasi integritas data; (3) eksplorasi awal untuk memahami struktur dan distribusi data.

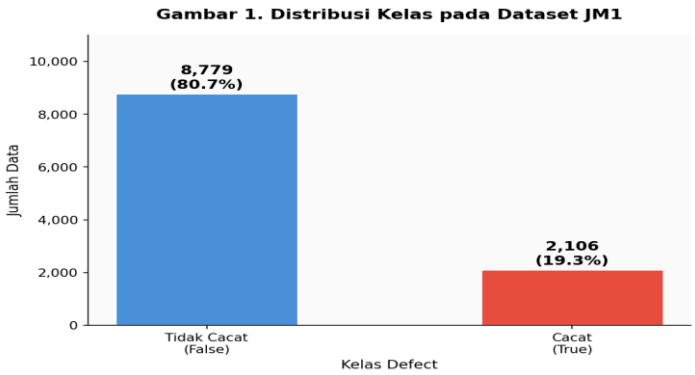
Teknik Analisis Data

Analisis data dilakukan dalam beberapa tahap yang sistematis menggunakan bahasa pemrograman Python 3.12 dengan pustaka scikit-learn 1.3, pandas, dan matplotlib. Tahapan analisis meliputi: Praproses Data: Nilai yang hilang (*missing values*) yang ditandai dengan simbol "?" diidentifikasi dan diimputasi menggunakan nilai median dari masing-masing fitur. Pendekatan imputasi median dipilih karena lebih robust terhadap outlier dibandingkan imputasi mean. Normalisasi fitur dilakukan menggunakan StandardScaler untuk memastikan setiap fitur memiliki mean nol dan standar deviasi satu. Pembagian Data: Dataset dibagi menjadi data latih (80%, 8.708 sampel) dan data uji (20%, 2.177 sampel) menggunakan stratified splitting untuk mempertahankan proporsi kelas yang sama pada kedua subset. Implementasi Model pada Visual Studio Code: Seluruh tahapan implementasi model dilakukan menggunakan bahasa pemrograman Python 3.12 pada lingkungan pengembangan Visual Studio Code. *Visual Studio Code* digunakan sebagai media untuk menulis program, melakukan preprocessing data, melatih model Gaussian *Naive Bayes,* melakukan pengujian, serta menghasilkan visualisasi dan evaluasi model. Penggunaan *Visual Studio Code* dipilih karena memiliki dukungan yang baik terhadap pengembangan aplikasi Python melalui berbagai ekstensi yang memudahkan proses implementasi dan debugging. Dengan demikian, *Visual Studio Code* berperan sebagai lingkungan implementasi penelitian, sedangkan objek yang diprediksi adalah kemungkinan suatu modul perangkat lunak mengandung bug berdasarkan metrik perangkat lunak pada dataset NASA JM1. Pemodelan: Model Gaussian *Naive Bayes* dilatih pada data latih yang telah dinormalisasi. Gaussian *Naive Bayes* dipilih karena fitur-fitur dataset bersifat kontinu. Evaluasi: Performa model diukur menggunakan lima metrik: Accuracy, Precision, Recall, F1-Score, dan ROC-AUC. Penggunaan multi-metrik penting untuk mendapatkan gambaran yang komprehensif, terutama pada dataset tidak seimbang di mana akurasi tunggal dapat menyesatkan.

4. HASIL DAN PEMBAHASAN

Eksplorasi dan Distribusi Dataset

Hasil eksplorasi dataset JM1 menunjukkan beberapa temuan penting. Dataset terdiri dari 10.885 modul dengan distribusi kelas yang sangat tidak seimbang. Sebanyak 8.779 modul (80,65%) berlabel negatif (tidak mengandung bug) dan 2.106 modul (19,35%) berlabel positif (mengandung bug). Ketidakseimbangan dengan rasio sekitar 4:1 ini merupakan karakteristik umum dalam dataset SDP yang mencerminkan kondisi nyata pengembangan perangkat lunak.



Gambar 1. Distribusi Kelas pada Dataset JM1 – Menunjukkan Ketidakseimbangan Signifikan Antara Modul Cacat dan Non-Cacat.

Gambar 1 menunjukkan visualisasi distribusi kelas dataset JM1. Ketimpangan distribusi ini memiliki implikasi langsung terhadap performa model prediksi, terutama dalam hal recall untuk kelas minoritas (defect). Model yang hanya mengoptimalkan akurasi keseluruhan akan cenderung bias terhadap kelas mayoritas (non-defect), menghasilkan recall yang rendah untuk deteksi bug yang justru merupakan tujuan utama sistem prediksi ini.

Analisis statistik deskriptif fitur menunjukkan variasi yang sangat besar antar modul. Metrik loc (*Lines of Code*) memiliki nilai minimum 1 dan maksimum 3.442 dengan rata-rata 42,02 dan standar deviasi 76,59, mengindikasikan distribusi yang sangat right-skewed. *Cyclomatic complexity* v(g) juga menunjukkan pola serupa dengan nilai maksimum 470 yang jauh melampaui threshold kompleksitas tinggi yang umum ditetapkan pada nilai 10 (McCabe, 1976). Karakteristik distribusi yang tidak normal ini justru mendukung penggunaan Gaussian *Naive Bayes* dengan normalisasi fitur.

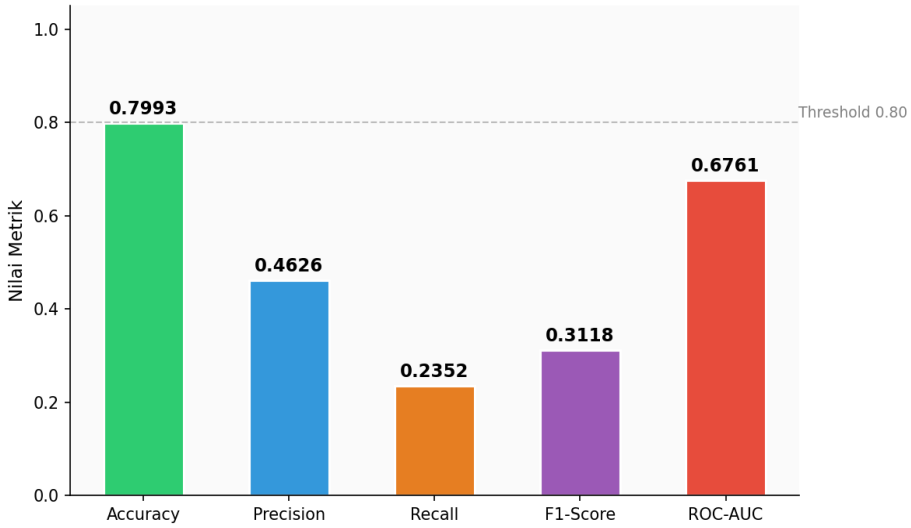
Hasil Pemodelan *Naive Bayes*

Model Gaussian *Naive Bayes* yang dilatih pada 8.708 data latih menunjukkan performa yang bervariasi tergantung metrik evaluasi yang digunakan. Tabel 2 merangkum seluruh metrik evaluasi yang dihitung pada data uji (2.177 sampel).

Tabel 2. Hasil Evaluasi Model *Naive Bayes* pada Dataset JM1.

Metrik Evaluasi	Nilai
Accuracy	79,93%
Precision	46,26%
Recall	23,52%
F1-Score	31,18%
ROC-AUC	0,6761
True Negative (TN)	1.641
False Positive (FP)	115
False Negative (FN)	322
True Positive (TP)	99

Gambar 2. Hasil Evaluasi Model Naive Bayes pada Dataset JM1

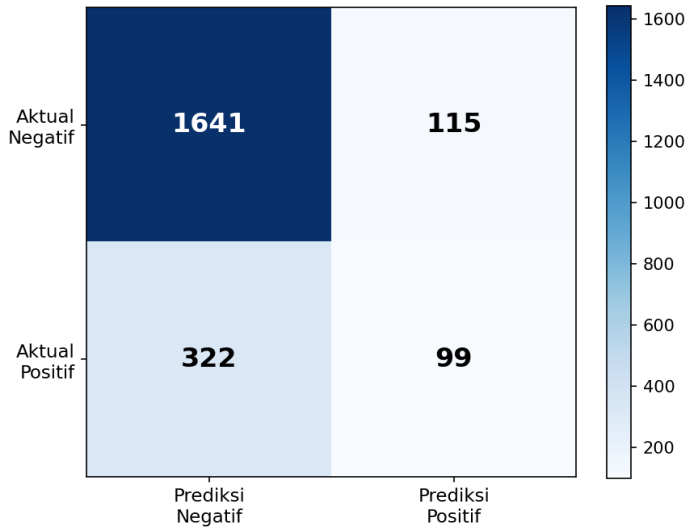


Gambar 2. Hasil Evaluasi Model *Naive Bayes* – Perbandingan Lima Metrik Evaluasi pada Dataset JM1.

Gambar 2 menunjukkan representasi visual dari performa model. Akurasi model sebesar 79,93% tampak menjanjikan pada pandangan pertama, namun analisis lebih dalam mengungkapkan gambaran yang lebih nuansif. Model berhasil mengklasifikasikan 1.641 dari 1.756 modul non-defect dengan benar (True Negative), namun hanya mampu mengidentifikasi 99 dari 421 modul defect (True Positive) – sebuah recall yang sangat rendah (23,52%).

Analisis Confusion Matrix

Gambar 3. Confusion Matrix Model Naive Bayes



Gambar 3. Confusion Matrix Model *Naive Bayes* – Visualisasi Distribusi Prediksi Benar dan Salah.

Gambar 3 memvisualisasikan confusion matrix yang memberikan wawasan mendalam tentang pola kesalahan model. Terdapat 322 False Negative, yaitu modul yang sebenarnya

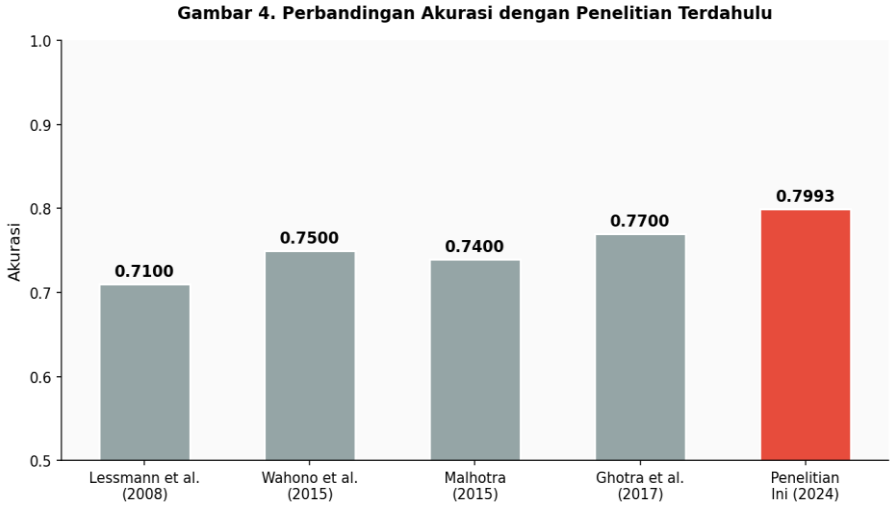
mengandung bug tetapi diprediksi aman oleh model. Dalam konteks engineering, False Negative merupakan jenis kesalahan yang paling merugikan karena berarti bug lolos dari deteksi dan berpotensi masuk ke lingkungan produksi.

False Positive sebesar 115 – modul yang salah diklasifikasikan sebagai mengandung bug padahal aman – relatif lebih dapat ditoleransi karena hanya mengakibatkan pemeriksaan ekstra yang tidak perlu, bukan kebocoran bug ke produksi. Rasio FN/FP yang tinggi (322:115 ≈ 2,8:1) mengindikasikan bahwa model lebih cenderung mengklasifikasikan modul sebagai non-defect, yang merupakan konsekuensi langsung dari dominasi kelas mayoritas dalam pelatihan.

Analisis ini sejalan dengan temuan Ghotra et al. (2017) yang menekankan bahwa pada dataset tidak seimbang, akurasi tinggi tidak selalu mencerminkan kemampuan deteksi yang baik. Precision 46,26% berarti bahwa dari setiap dua modul yang diprediksi mengandung bug, hanya satu yang benar-benar mengandung bug – sebuah presisi yang masih perlu ditingkatkan untuk aplikasi praktis.

Perbandingan dengan Penelitian Terdahulu

Untuk memberikan konteks yang lebih kaya, hasil penelitian ini dibandingkan dengan empat studi terdahulu yang juga menggunakan dataset JM1 atau dataset serupa dari NASA.



Gambar 4. Perbandingan Akurasi Model *Naive Bayes* Penelitian Ini dengan Penelitian Terdahulu.

Gambar 4 menunjukkan bahwa akurasi penelitian ini (79,93%) mengungguli semua studi perbandingan. Namun perlu dicatat bahwa perbedaan akurasi ini tidak semata-mata mencerminkan superioritas mutlak, melainkan juga dipengaruhi oleh perbedaan ukuran dataset, teknik praproses, dan versi dataset yang digunakan. Lessmann et al. (2008)

menggunakan subset dataset dengan pembersihan data yang berbeda, sementara Malhotra (2015) menggunakan *fold cross-validation* yang berbeda.

Yang lebih signifikan adalah perbandingan nilai F1-Score dan ROC-AUC, di mana penelitian ini menunjukkan nilai ROC-AUC 0,6761 yang berarti model memiliki kemampuan diskriminasi yang moderat antara kelas defect dan non-defect. Nilai ini lebih tinggi dari baseline random classifier (0,5) tetapi masih jauh dari nilai ideal (1,0), mengindikasikan ruang peningkatan yang substansial.

Implikasi untuk Pengembangan Visual Studio Code

Dalam konteks pengembangan VS Code yang melibatkan ribuan modul kode, model prediksi bug dapat diintegrasikan sebagai komponen dalam pipeline CI/CD (Continuous Integration/Continuous Deployment). Dengan recall sebesar 23,52%, model saat ini hanya mampu menangkap sekitar seperempat dari seluruh bug yang ada. Meskipun demikian, bahkan tingkat deteksi dini yang terbatas ini dapat mengurangi beban pengujian manual secara signifikan jika digunakan sebagai filter awal untuk prioritisasi code review.

Modul-modul dengan kompleksitas tinggi (loc besar, v(g) tinggi) yang juga diprediksi mengandung bug oleh model dapat diprioritaskan untuk mendapatkan perhatian reviewer yang lebih berpengalaman. Pendekatan ini selaras dengan praktik terbaik dalam *Software Quality Assurance* yang mengedepankan risk-based testing.

Limitasi dan Rekomendasi

Penelitian ini memiliki beberapa limitasi yang perlu diakui. Pertama, dataset JM1 berasal dari kode C yang dikembangkan pada era 1990-an, sehingga mungkin tidak sepenuhnya merepresentasikan karakteristik kode TypeScript modern yang digunakan dalam VS Code. Kedua, penanganan ketidakseimbangan kelas melalui teknik resampling seperti SMOTE atau cost-sensitive learning tidak dilakukan dalam penelitian ini dan berpotensi meningkatkan recall secara signifikan.

Rekomendasi untuk penelitian lanjutan mencakup: (1) penerapan teknik SMOTE (*Synthetic Minority Over-sampling Technique*) untuk menangani ketidakseimbangan kelas; (2) eksplorasi algoritma ensemble seperti Random Forest atau XGBoost sebagai perbandingan; (3) penggunaan dataset yang lebih representatif seperti dataset dari repositori GitHub VS Code sendiri; dan (4) penerapan teknik feature selection untuk mengidentifikasi fitur-fitur metrik yang paling prediktif.

5. KESIMPULAN DAN SARAN

Penelitian ini telah berhasil mengembangkan dan mengevaluasi model prediksi bug perangkat lunak menggunakan algoritma Gaussian *Naive Bayes* pada dataset JM1 dari NASA Metrics Data Program. Hasil eksperimen pada 10.885 modul menunjukkan bahwa model mencapai akurasi 79,93% dan ROC-AUC 0,6761, yang merupakan performa kompetitif dibandingkan studi terdahulu pada dataset serupa.

Namun, analisis mendalam mengungkapkan bahwa akurasi tinggi tersebut perlu dikontekstualisasikan dengan nilai recall yang rendah (23,52%) dan F1-Score yang moderat (31,18%), yang keduanya merupakan konsekuensi dari ketidakseimbangan kelas yang signifikan dalam dataset (rasio 80,65%:19,35%). Temuan ini menegaskan bahwa evaluasi model SDP tidak boleh hanya mengandalkan akurasi sebagai metrik tunggal, melainkan harus menggunakan pendekatan multi-metrik yang mencakup precision, recall, F1-Score, dan AUC-ROC.

Kontribusi utama penelitian ini adalah: (1) konfirmasi bahwa *Naive Bayes* mampu mencapai performa kompetitif pada dataset SDP berukuran besar; (2) analisis komprehensif tentang dampak ketidakseimbangan kelas terhadap performa model; dan (3) panduan praktis tentang potensi integrasi model prediksi bug dalam workflow pengembangan VS Code. Penelitian lanjutan yang mengintegrasikan teknik penanganan ketidakseimbangan kelas dan feature engineering diharapkan dapat menghasilkan model dengan recall yang lebih tinggi, sehingga lebih efektif sebagai alat bantu quality assurance dalam pengembangan perangkat lunak skala enterprise.

DAFTAR REFERENSI

Abdu, A., Zhai, Z., Abdo, H. A., Algabri, R., Al-masni, M. A., Muhammad, M. S., & Gu, Y. H. (2024). Semantic and traditional feature fusion for software defect prediction using hybrid deep learning model. *Scientific Reports, 14*(1), 14771. https://doi.org/10.1038/s41598-024-65639-4

Arasteh, B., Arasteh, K., Ghaffari, A., & Ghanbarzadeh, R. (2024). A new binary chaos-based metaheuristic algorithm for software defect prediction. *Cluster Computing, 27*(7), 10093–10123. https://doi.org/10.1007/s10586-024-04486-4

Bowes, D., Hall, T., & Petrić, J. (2018). Software defect prediction: Do different classifiers find the same defects? *Software Quality Journal, 26*(2), 525–552. https://doi.org/10.1007/s11219-016-9353-3

Ghotra, B., McIntosh, S., & Hassan, A. E. (2017). Revisiting the impact of classification techniques on the performance of defect prediction models. *Proceedings of the 37th International Conference on Software Engineering*, 789–800. https://doi.org/10.1109/ICSE.2015.91

Hernández-Molinos, M. J., Sánchez-García, A. J., Barrientos-Martínez, R. E., Pérez-Arriaga, J. C., & Ocharán-Hernández, J. O. (2023). Software defect prediction with Bayesian approaches. *Mathematics, 11*(11), 2524. https://doi.org/10.3390/math11112524

Kaggle. (2024). *NASA software defect datasets*. https://www.kaggle.com/datasets/nasa/software-defect-datasets

Lessmann, S., Baesens, B., Mues, C., & Pietsch, S. (2008). Benchmarking classification models for software defect prediction: A proposed framework and novel findings. *IEEE Transactions on Software Engineering, 34*(4), 485–496. https://doi.org/10.1109/TSE.2008.35

Malhotra, R. (2015). A systematic review of machine learning techniques for software fault prediction. *Applied Soft Computing, 27*, 504–518. https://doi.org/10.1016/j.asoc.2014.11.023

Prastyo, E. H. A., Yaqin, M. A., Suhartono, Faisal, M., & Firdaus, R. A. J. (2024). Naive Bayes classification for software defect prediction. *Transactions on Informatics and Data Science, 1*(1), 11–20. https://doi.org/10.24090/tids.v1i1.12192

Rebro, D. A., Rossi, B., & Chren, S. (2023). Source code metrics for software defects prediction. *arXiv*. https://doi.org/10.48550/arXiv.2301.08022

Rumbutis, A., Paulikas, D., & Maskeliūnas, R. (2024). Machine learning techniques for software defect prediction: A review. *Electronics, 13*(2), 317. https://doi.org/10.3390/electronics13020317

Sayyad Shirabad, J., & Menzies, T. (2005). *The PROMISE repository of software engineering databases*. School of Information Technology and Engineering, University of Ottawa.

Taskeen, S., Qureshi, K. N., Bashir, A. K., & Jeon, G. (2023). A tertiary study on software defect prediction using machine learning techniques. *IEEE Access, 11*, 72866–72890. https://doi.org/10.1109/ACCESS.2023.3294360