JURNAL FASILKOM
Volume 16 No. 2 | Agustus 2026: 257-263

P-ISSN : 2089-3353
E-ISSN : 2808-9162

Evaluasi Kritis *Random Forest* dalam Prediksi Kegagalan CI/CD *Pipeline* pada *Dataset Log* Sintetis

Rizqia Fauziah Rachma[1], Windu Gata[2], Farizal Ginanjar[3], Muhammad Arief Nadhofa[4]
[1,2,3,4]Jurusan Ilmu Komputer, Fakultas Teknik Informatika, Universitas Nusa Mandiri
[1]14250002@nusamandiri.ac.id*, [2]windu@nusamandiri.ac.id, [3]14250014@nusamandiri.ac.id, [4]14250027@nusamandiri.ac.id

Abstract

Continuous Integration and Continuous Deployment (CI/CD) pipelines are crucial in modern DevOps environments; however, pipeline failures often hinder software delivery. This study aims to implement Machine Learning specifically the Random Forest algorithm to predict the specific stage of failure (Build, Test, or Deploy phases) using execution logs. The research methodology encompasses data preprocessing, the extraction of three key dynamic features (execution duration, CPU usage, and memory consumption), model training using an 80:20 train-test split, and performance evaluation via a confusion matrix. Evaluation based on 9,000 test samples representing 20% of the total 45,000 synthetic failure logs yielded an overall accuracy of 33.22%, with F1-scores of 0.33 for the Build class, 0.35 for Deploy, and 0.32 for Test. These results indicate that the Random Forest model possesses very limited generalization capability because the synthetic dataset lacked naturally robust failure trace patterns, as evidenced by an accuracy level approaching that of random guessing. It is concluded that the use of a synthetic log dataset constitutes the primary limitation and the root cause of the model's generalization failure in this study. Therefore, future research is recommended to utilize actual operational logs from real-world CI/CD platforms, such as GitHub Actions or Jenkins, to enable the model to demonstrate more valid causal relationships.

Keywords: CI/CD pipeline, DevOps, AIOps, failure prediction, random forest

Abstrak

Jalur integrasi dan penerapan berkelanjutan (CI/CD) sangat penting dalam lingkungan *DevOps modern*, namun kegagalan *pipeline* sering kali menghambat pengiriman perangkat lunak. Penelitian ini bertujuan mengimplementasikan *Machine Learning*, khususnya algoritma *Random Forest*, untuk memprediksi tahapan spesifik dari kegagalan (fase *Build*, *Test*, atau *Deploy*) menggunakan log eksekusi. Metodologi penelitian mencakup pra-pemrosesan data, ekstraksi 3 fitur dinamis utama (durasi eksekusi, penggunaan CPU, dan konsumsi memori), pelatihan model dengan rasio 80:20 untuk data latih dan data uji, serta evaluasi performa melalui *confusion matrix*. Evaluasi terhadap 9.000 data uji, yaitu 20% dari keseluruhan 45.000 log kegagalan sintetis, menghasilkan akurasi total sebesar 33,22%, dengan *F1-score* 0,33 untuk kelas *Build*, 0,35 untuk *Deploy*, dan 0,32 untuk *Test*. Hasil ini mengindikasikan bahwa model *Random Forest* memiliki kemampuan generalisasi yang sangat terbatas karena dataset sintetis yang digunakan tidak memiliki pola jejak kegagalan yang kuat secara natural, terlihat dari tingkat akurasi yang mendekati probabilitas tebakan acak. Disimpulkan bahwa penggunaan dataset log sintetis merupakan limitasi utama dan menjadi akar masalah kegagalan generalisasi model dalam penelitian ini. Oleh karena itu, penelitian selanjutnya direkomendasikan untuk menggunakan log operasional asli dari *platform* CI/CD nyata seperti *GitHub Actions* atau *Jenkins* agar model dapat membuktikan sebab akibat yang lebih valid.

Kata kunci*: pipeline* CI/CD, *DevOps*, AIOps, prediksi kegagalan, *random forest*

1. Pendahuluan

Pengembangan perangkat lunak modern membutuhkan kecepatan, keandalan, dan efektivitas untuk memenuhi permintaan pasar, yang mendorong transisi dari metode tradisional menuju praktik DevOps [1]. Pendekatan ini mengintegrasikan fase pengembangan dan operasi untuk meningkatkan kolaborasi tim dan konsistensi kualitas perangkat lunak [2], [3]. Jantung dari operasi DevOps yang sukses adalah penerapan *Continuous Integration* dan *Continuous Deployment* (CI/CD) *pipeline* yang mengotomatiskan kompilasi, pengujian, dan penyebaran aplikasi [4], [5]. Dengan *pipeline* ini, pengembang dapat mendeteksi *bug* lebih awal pada fase pengembangan [6], [7], yang secara drastis menekan *time-to-market* dan menjamin integritas kode utama [8] Meskipun memberikan otomatisasi yang kuat, operasional CI/CD di dunia nyata sering kali dihadapkan pada kendala teknis di berbagai fasenya [9], [10]. Secara historis, kegagalan kompilasi (*build failure*) sering disorot sebagai isu krusial karena sifatnya yang memblokir langkah awal *pipeline* [11], [12]. Namun, dalam ekosistem DevOps yang komprehensif, kegagalan pada tahap pengujian (*test*) dan peluncuran (*deploy*) justru sering kali menimbulkan pemborosan sumber daya komputasi dan risiko *downtime* peladen yang jauh lebih merugikan secara operasional [13]. Oleh karena itu, sistem AIOps modern tidak cukup jika hanya dirancang menggunakan pendekatan klasifikasi biner (mendeteksi kegagalan *build* atau tidak). Untuk memberikan peringatan dini yang holistik, memfasilitasi *debugging* yang spesifik, dan mengoptimalkan alur kerja secara menyeluruh, model prediktif harus dirancang menggunakan pendekatan klasifikasi multi-kelas guna mengidentifikasi secara

P-ISSN : 2089-3353
E-ISSN : 2808-9162

akurat di tahap mana sebuah *pipeline* berisiko mengalami kegagalan [14], [15].

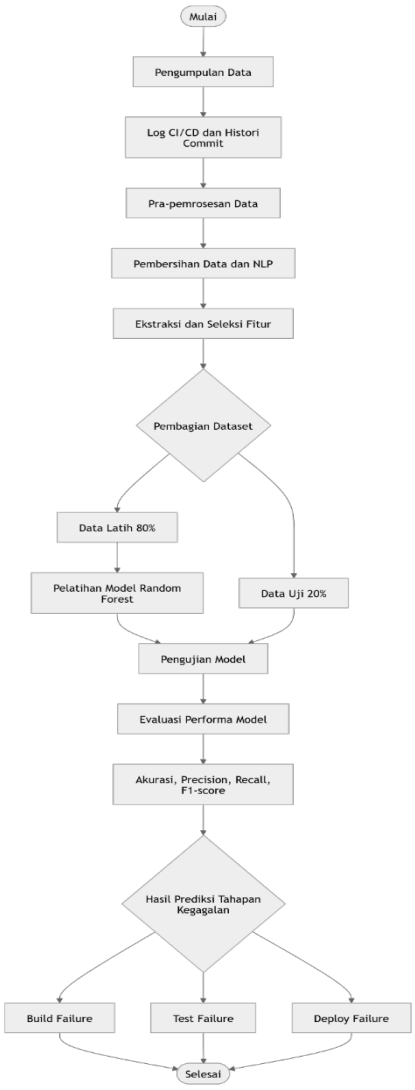
Sebagai landasan teori, ekosistem CI/CD modern memiliki tiga tahapan krusial yang menjadi target prediksi dalam penelitian ini. Pertama adalah *Fase Build* (kompilasi), di mana kode dikompilasi menjadi artefak; kegagalan umumnya disebabkan oleh kesalahan sintaksis atau dependensi yang hilang. Kedua adalah *Fase Test* (pengujian), di mana sistem menjalankan pengujian otomatis; kegagalan di sini memakan waktu komputasi besar dan dipicu oleh beban puncak CPU. Ketiga adalah *Fase Deploy* (peluncuran), bertugas mengirimkan artefak ke server produksi; kegagalan tahap ini paling kritis karena berisiko menyebabkan *downtime* akibat memori penuh atau kesalahan konfigurasi.

Untuk mengatasi batasan reaktif dari sistem pakar konvensional, industri kini mengadopsi konsep AIOps (*Artificial Intelligence for IT Operations*). Meskipun kemampuan AI sangat besar, tinjauan literatur menunjukkan adanya *research gap* pada studi terdahulu yang menggunakan algoritma prediktif. Keunggulan model saat ini terlalu bergantung pada ekstraksi fitur statis dari kode sumber, dan sering mengabaikan fitur dinamis dari log eksekusi [16], [17]. Mengingat data log operasional yang bersifat *real-time* umumnya tertutup rapat sebagai rahasia infrastruktur perusahaan, penelitian ini hadir untuk menguji sebuah pertanyaan kritis: apakah dataset log sintetis publik mampu menyimulasikan dinamika fitur tersebut secara memadai? Dengan demikian, penelitian ini bertujuan mengevaluasi sejauh mana model *Random Forest* yang dilatih pada *dataset* sintetis dapat menutup *gap* tersebut, sekaligus menganalisis keterbatasan kausalitas buatan yang ada di dalamnya.

Meskipun model-model canggih tersebut mampu mencapai tingkat akurasi teoretis yang tinggi, implementasi praktisnya di lingkungan *DevOp*s sering kali terhambat oleh masalah skalabilitas dan besarnya beban komputasi saat mengekstrak data log berdimensi tinggi secara *real-time* [18]. Tantangan kompleksitas komputasi pada algoritma *machine learning* ketika memproses data berukuran besar ini merupakan isu krusial yang menuntut efisiensi sistem [19]. Menghadapi tantangan tersebut, penelitian ini secara spesifik mengevaluasi penggunaan *Random Forest*. Algoritma *ensemble* ini dipilih karena rekam jejaknya yang tangguh dalam menangani ruang fitur yang heterogen serta kemampuannya yang telah terbukti andal dalam berbagai tugas klasifikasi berskala besar [20]. Melalui evaluasi kinerja *Random Forest* terhadap dataset log sintetis, penelitian ini diharapkan dapat memberikan pembuktian empiris mengenai sejauh mana kualitas dan kausalitas data memengaruhi keandalan model sebuah aspek penting yang menentukan apakah sistem prediksi dapat diaplikasikan di lingkungan produksi *DevOps* yang sesungguhnya.

2. Metode Penelitian

Penelitian ini menggunakan pendekatan klasifikasi multi-kelas untuk memprediksi tahapan kegagalan pipeline CI/CD yang terdiri atas *Build, Test*, dan *Deploy*. Pendekatan ini dipilih karena memungkinkan adanya pengujian variabel secara terkontrol di dalam lingkungan komputasi, sehingga efektivitas algoritma dapat diukur secara objektif melalui angka-angka performa yang dihasilkan. Metodologi penelitian meliputi pengumpulan data secara sistematis, pemrosesan awal data tersebut, rekayasa fitur terkait, pelatihan model prediktif, dan penilaian kinerjanya.



Gambar 1. *Flowchart* Tahapan Penelitian

2.1. Pengumpulan Data (*Dataset*)

Data yang digunakan dalam penelitian ini merupakan *dataset* sekunder yang diperoleh dari *platform Kaggle* "CI/CD *Pipeline Failures Dataset*" oleh Mirza Yasir Abdullah Baig (URL: https://www.kaggle.com/datasets/mirzayasirabdullah07/cicd-pipeline-failure-logs-dataset-for-aiops). Dataset berisi 45.000 log kegagalan *pipeline* CI/CD yang terdistribusi secara seimbang (masing-masing 15.000

P-ISSN : 2089-3353
E-ISSN : 2808-9162

sampel untuk kelas *Build, Test,* dan *Deploy*). Dataset ini bersifat sintetis (*synthetic dataset*), sehingga tidak berasal langsung dari lingkungan produksi nyata seperti *GitHub Actions* atau *Jenkins*. Eksplorasi data awal (EDA) yang dilakukan pada dataset ini menunjukkan tidak adanya nilai yang hilang (*missing values*), namun memperlihatkan indikasi kuat adanya homogenitas rentang nilai numerik antar kelas kegagalan.

Dataset yang digunakan dalam penelitian ini merupakan representasi dari log aktivitas CI/CD yang mencatat metrik sistem selama siklus hidup perangkat lunak. Karakteristik utama dari log eksekusi ini adalah ketergantungan waktu (*temporal dependency*) dan variabilitas beban kerja yang tinggi. Dalam lingkungan *DevOps* yang dinamis, durasi *build* dan penggunaan sumber daya komputasi sering kali dipengaruhi oleh ukuran *repository*, kompleksitas *dependency graph*, serta efisiensi konfigurasi *runner*. Dataset sintetis berusaha meniru distribusi statistik ini, namun sering kali gagal merepresentasikan anomali teknis (seperti *deadlock* pada tahap *test* atau kegagalan koneksi *database* saat *deploy*) yang justru menjadi pemicu utama kegagalan di dunia nyata. Pemahaman mengenai perbedaan karakteristik ini sangat krusial bagi peneliti AIOps dalam mengevaluasi apakah model prediktif yang dibangun benar-benar mempelajari perilaku sistem atau sekadar melakukan korelasi statistik pada data yang terdistribusi secara seragam.

2.2. Pra-Pemrosesan Data (*Data Preprocessing*)

Mengingat data operasional *DevOps* sangat rentan terhadap noise dan ketidaklengkapan, tahapan pra-pemrosesan dilakukan secara ketat. Langkah-langkah yang dilakukan difokuskan murni pada pembersihan data (*data cleaning*), yaitu menghapus baris log pipeline yang tidak valid, duplikat, atau memiliki nilai yang hilang (*missing values*). Tahapan ini secara khusus menyiapkan ekstraksi atribut numerik yang mencerminkan metrik kinerja sistem saat eksekusi berlangsung, tanpa melibatkan pemrosesan bahasa alami (NLP) mengingat tidak ada fitur teks yang digunakan sebagai prediktor.

2.3. Ekstraksi dan Pemilihan Fitur (*Feature Engineering*)

Fitur yang diekstraksi sebagai variabel prediktor (X) difokuskan pada tiga fitur dinamis berbentuk numerik, yaitu: durasi eksekusi *pipeline* (*build_duration_sec*), beban kerja CPU (*cpu_usage_pct*), dan konsumsi RAM (*memory_usage_mb*). Atribut-atribut ini diidentifikasi dan dievaluasi tingkat relevansinya melalui penerapan metodologi pemilihan fitur (*feature selection*) menggunakan metrik *Gini Importance*. Pendekatan ini dilakukan untuk mengurangi dimensi data, memastikan bahwa hanya fitur dengan bobot informasi tertinggi yang digunakan dalam pemodelan, serta sebagai upaya untuk memitigasi risiko *overfitting*.

2.4. Pengembangan Model Artificial Intelligence

Pada penelitian ini, algoritma yang diuji adalah metode *ensemble Random Forest* karena keandalannya dalam menangani hubungan non-linear pada data tabular berdimensi tinggi. Model ini dikonfigurasi secara optimal melalui *hyperparameter tuning* (menggunakan *GridSearchCV*) untuk menetapkan parameter dasar seperti n_estimators=100. Sebagai *baseline* (referensi batas bawah) untuk mengevaluasi signifikansi performa, kinerja *Random Forest* juga dikomparasikan dengan algoritma *Decision Tree* standar. Model dilatih menggunakan pembagian dataset dengan rasio 80% data latih (*training data*) dan 20% data uji (*testing data*).

2.5. Evaluasi dan Validasi Model

Akurasi digunakan untuk mengukur persentase keseluruhan prediksi tahapan kegagalan yang benar. *Precision* menunjukkan ketepatan model ketika memprediksi suatu kelas tertentu, misalnya dari seluruh data yang diprediksi sebagai *Build*, berapa yang benar-benar termasuk kelas *Build*. *Recall* menunjukkan kemampuan model dalam menangkap seluruh data aktual pada suatu kelas tertentu, misalnya dari seluruh kegagalan *Build* yang sebenarnya, berapa yang berhasil dikenali oleh model. F1-*score* digunakan untuk mengukur keseimbangan antara *precision* dan *recall* pada masing-masing kelas.

2.6. Implementasi pada Data Uji

Tahapan krusial setelah model *Random Forest* selesai dilatih adalah menguji kemampuannya menggunakan sekumpulan data yang belum pernah "dilihat" sebelumnya oleh algoritma selama fase pelatihan. Dalam penelitian ini, sesuai dengan rasio pembagian data (*data splitting*) 80:20 dari total dataset log kegagalan, model dievaluasi secara ketat untuk mengukur sejauh mana kemampuan generalisasi (*generalization capability*) model terhadap variasi data baru, sekaligus menjadi instrumen untuk mendeteksi potensi terjadinya *overfitting* atau *underfitting* pada arsitektur yang dibangun.

Proses implementasi pada tahap ini dimulai dengan mengekstraksi seluruh variabel prediktor numerik dari data uji (direpresentasikan sebagai *matriks independent* X_{test}). Pada saat yang bersamaan, kolom target yang berisi jawaban asli mengenai tahapan kegagalan (direpresentasikan sebagai vektor dependen Y_{test}) disembunyikan atau diisolasi dari model. Matriks X_{test} yang berisi parameter metrik seperti durasi kompilasi, status memori, dan beban CPU kemudian diumpankan ke dalam model *Random Forest* yang telah matang.

Berdasarkan struktur percabangan dan bobot informasi (*Gini Impurity*) yang telah dipelajari sebelumnya, ke-100 pohon keputusan (*estimators*) di dalam model akan mengevaluasi setiap baris log tersebut. Masing-masing pohon akan mengeluarkan satu suara tebakan, dan melalui mekanisme *majority voting*, model akan menetapkan satu label prediksi akhir (direpresentasikan sebagai $\hat{y}$) untuk mengklasifikasikan apakah log

P-ISSN : 2089-3353
E-ISSN : 2808-9162

tersebut termasuk ke dalam kegagalan tahap *Build, Test,* atau *Deploy*.

Sebagai langkah akhir dari implementasi ini, array prediksi (y) yang dihasilkan oleh mesin disandingkan secara berhadap-hadapan dengan vektor nilai aktual (Y_{test}). Proses komparasi inilah yang kemudian diubah ke dalam bentuk tabulasi matriks kebingungan (*confusion matrix*). Evaluasi objektif antara nilai tebakan dan nilai aktual pada data uji ini menjadi tolak ukur tunggal untuk menyimpulkan kelayakan sistem *AIOps* sebelum direkomendasikan untuk diintegrasikan ke dalam ekosistem *DevOps* di dunia nyata.

3. Hasil dan Pembahasan

Penelitian ini menerapkan algoritma *Machine Learning*, khususnya model *Random Forest*, yang dilatih untuk menangani data berdimensi tinggi berupa log eksekusi CI/CD *pipeline*. Pada tahap ini, model dilatih untuk memprediksi secara spesifik di tahap mana (*failure stage*) sebuah *pipeline* kemungkinan besar akan mengalami kegagalan, yaitu pada tahap *Build, Deploy,* atau *Test.*

Tabel 1. Komposisi Pembagian Dataset Penelitian

Keterangan	Jumlah Sampel
Total Dataset	45.000
Data Latih (80%)	36.000
Data Uji (20%)	9.000

Sesuai dengan alur metodologi yang digambarkan pada Gambar 1, penelitian ini memulai fase eksperimen dengan melakukan pembagian dataset (data splitting) menggunakan rasio 80:20. Sebesar 80% dari dataset digunakan untuk proses pelatihan model *Random Forest*, sementara 20% sisanya dialokasikan sebagai data uji (testing data) yang berfungsi untuk mengukur kemampuan generalisasi model. Evaluasi performa kemudian dilakukan dengan menghitung metrik akurasi, *precision*, *recall*, dan *F1-score* untuk menghasilkan prediksi akhir tahapan kegagalan *pipeline (Build, Test,* atau *Deploy).*

3.1. Evaluasi *Training Data* dan *Testing Data*

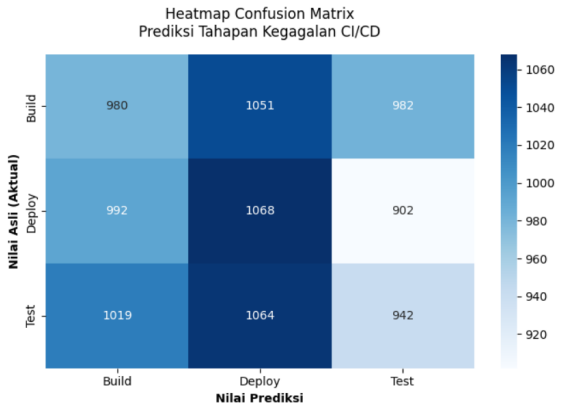
Pada proses pemodelan, evaluasi dilakukan secara komprehensif dengan membandingkan performa model saat memproses *training data* versus *testing data*. Pada fase pelatihan menggunakan 36.000 sampel data, algoritma *Random Forest* (dengan n_*estimators*=100) berhasil memetakan fitur secara absolut, mencapai tingkat akurasi latih (*training accuracy*) sebesar 100,00%.

Namun, hasil evaluasi pada 9.000 data uji (20% dari total 45.000 dataset) menunjukkan penurunan drastis, di mana model hanya mencapai tingkat akurasi keseluruhan sebesar 33,22%. Kesenjangan (*gap*) yang sangat besar antara performa latih absolut (100%) dan performa uji yang setara dengan tebakan acak (33%) ini membuktikan secara empiris terjadinya fenomena *overfitting* yang ekstrem. Model secara teknis berhasil "menghafal" *training data*, tetapi gagal total melakukan generalisasi saat dihadapkan pada *testing data*. Kegagalan generalisasi ini merupakan indikator kuat bahwa atribut fitur di dalam dataset sintetis tersebut tidak memiliki pola sebab-akibat yang nyata, melainkan hanya distribusi nilai acak.

Berdasarkan hasil eksperimen, model menunjukkan akurasi latih sebesar 100,00% dan akurasi uji sebesar 33,22%, yang merupakan indikator empiris terjadinya *overfitting* ekstrem.

3.2. Analisis Visual *Confusion Matrix*

Untuk membedah distribusi kesalahan prediksi, penelitian ini memvisualisasikan matriks kebingungan dalam bentuk *heatmap* seperti yang ditunjukkan pada Gambar 2. Prediksi Tahapan Kegagalan (*Confusion Matrix*)



Gambar 2. Visualisasi *Heatmap Confusion Matrix* 3-Kelas

Distribusi hasil klasifikasi secara lebih rinci dapat dilihat pada *Confusion Matrix* (Gambar 2). Visualisasi tersebut semakin mempertegas kegagalan generalisasi model. Pada skenario prediksi yang ideal, mayoritas data uji seharusnya terkonsentrasi pada diagonal utama (representasi *True Positive*). Namun, matriks menunjukkan bahwa model melakukan kesalahan klasifikasi yang terdistribusi secara hampir merata di seluruh kelas prediksi. Sebagai contoh, dari total data yang aslinya gagal di tahap *Build*, model secara keliru memprediksinya sebagai kegagalan *Deploy* (1.051 kasus) dan *Test* (982 kasus) dengan proporsi yang mengalahkan prediksi benarnya (980 kasus). Pola sebaran kesalahan yang merata ini secara empiris membuktikan bahwa model tidak menemukan sekuens logis apa pun, sehingga keputusan klasifikasi yang diambil setara dengan probabilitas tebakan acak.

3.3. Evaluasi Metrik Klasifikasi

Rincian performa model untuk setiap kelas target dijabarkan pada Tabel 1. Metrik yang digunakan meliputi *Precision*, *Recall*, dan *F1-Score*. Penggunaan ketiga metrik ini bertujuan untuk memberikan gambaran performa model secara komprehensif, mengingat akurasi saja tidak cukup untuk menggambarkan kualitas klasifikasi pada dataset yang memiliki karakteristik kompleks.

Tabel 2. *Classification Report* Prediksi Tahapan Kegagalan CI/CD

P-ISSN : 2089-3353
E-ISSN : 2808-9162

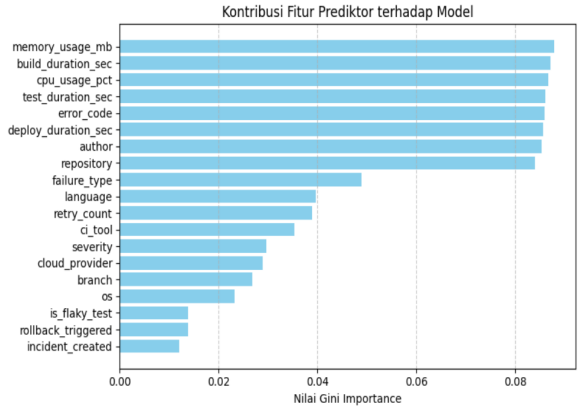
Tahap Kegagalan	*Precision*	*Recall*	*F1-score*	*Support*
Build (0)	0.33	0.33	0.33	3013
Deploy (1)	0.34	0.36	0.35	2962
Test (2)	0.33	0.31	0.32	3025
Accuracy			0.33	
Macro Avg	0.33	0.33	0.33	9000
Rata-rata (*weighted avg*)	0.33	0.33	0.33	9000

Secara rata-rata (*weighted avg*), model menghasilkan nilai F1-*score* sebesar 0,33. Rendahnya nilai *recall* pada tahap *Test* (0,31) menunjukkan bahwa model paling sering melewatkan kegagalan yang terjadi pada fase pengujian. Hal ini mengindikasikan bahwa model memiliki sensitivitas yang sangat lemah dalam mengidentifikasi pola kegagalan spesifik di lingkungan pengujian otomatis. Dalam praktiknya, kegagalan di tahap *Test* biasanya melibatkan interaksi yang kompleks antara kode aplikasi dan dependensi eksternal, namun karena dataset sintetis ini mendistribusikan metrik secara acak, model gagal menangkap sinyal-sinyal unik yang seharusnya membedakan fase ini dari fase lainnya. Akibatnya, banyak kegagalan kritis pada tahap pengujian yang tidak terdeteksi oleh sistem AI, yang secara operasional akan sangat merugikan integritas siklus pengiriman perangkat lunak di lingkungan produksi nyata.

3.4. Analisis *Overfitting* dan *Validitas* Data

Salah satu temuan krusial dalam penelitian ini adalah adanya indikasi *overfitting*. Selama fase pelatihan, model mampu mengenali pola pada data latih dengan sangat baik, namun gagal melakukan generalisasi pada data uji. Ketidakmampuan model untuk meningkatkan akurasi di atas 33,22% pada data uji mengindikasikan secara empiris bahwa dataset publik yang bersifat sintetis ini mengandung *noise* yang tinggi. Dalam dunia nyata, kegagalan *Build* biasanya diawali dengan lonjakan penggunaan CPU, sedangkan kegagalan *Deploy* sering dikaitkan dengan masalah jaringan; namun pada dataset ini, korelasi logis tersebut tidak ditemukan.

3.5. Analisis Kontribusi Fitur Prediktor



Gambar 3. Kontribusi Fitur Prediktor terhadap Model

Grafik *Gini Importance* pada Gambar 3 menunjukkan distribusi bobot setiap fitur dalam proses pengambilan keputusan model. Hasil visualisasi memperlihatkan bahwa tidak ada fitur tunggal yang mendominasi secara signifikan. Delapan fitur teratas (mulai dari *memory_usage_mb* hingga *repository)* memiliki nilai *Gini Importance* yang sangat dekat satu sama lain, yakni berada di kisaran 0.08 hingga 0.09.

Secara metodologis, distribusi bobot yang hampir seragam ini menunjukkan bahwa dataset sintetis memiliki tingkat homogenitas fitur yang tinggi. Model tidak mampu mengidentifikasi "sinyal" atau atribut pembeda yang kuat untuk setiap tahap kegagalan (*Build*, *Test*, atau *Deploy*). Hal ini secara empiris menjelaskan mengapa performa model sangat rendah: karena fitur-fitur yang digunakan (seperti durasi dan penggunaan sumber daya) tidak memiliki korelasi logis yang spesifik dengan kelas target di dalam dataset sintetis ini. Ketiadaan fitur yang dominan (*dominant predictor*) ini membuktikan bahwa model tidak mempelajari pola teknis yang otentik, melainkan hanya melakukan pengenalan pola acak yang tidak valid untuk sistem AIOps.

Meskipun akurasi rendah, algoritma *Random Forest* tetap memberikan gambaran mengenai bobot kontribusi fitur melalui *feature importance*. Variabel-variabel yang paling dominan dalam proses percabangan pohon keputusan dirinci pada Tabel 2. Analisis ini sangat penting karena menunjukkan parameter mana yang dianggap paling informatif oleh model saat berusaha meminimalkan nilai *Gini Impurity* pada setiap simpul keputusan. Dalam konteks ini, fitur-fitur seperti durasi eksekusi dan beban kerja sumber daya diposisikan sebagai prediktor utama karena secara logis merupakan indikator pertama dari adanya gangguan pada *pipeline*.

Tabel 3. Rincian dan Nilai Kontribusi Fitur Prediktor (*Feature Importance*) Teratas

Nama Fitur	Tipe Data	Deskripsi Operasional	Nilai Kepentingan (Gini)
memory_usage_mb	Integer	Konsumsi RAM oleh *runner* CI/CD	0.0878
build_duration_sec	Integer	Durasi waktu eksekusi fase *build*	0.0871
cpu_usage_pct	Float	Beban kerja CPU saat eksekusi	0.0867

3.6. Implikasi pada Implementasi *DevOps*

Secara praktis, model dengan akurasi 33,22% belum layak diimplementasikan dalam lingkungan produksi *DevOps* yang sebenarnya. Penggunaan model ini akan mengakibatkan dua masalah utama: *alert fatigue* akibat banyaknya peringatan palsu (*false positives*) dan risiko lolosnya kegagalan kritis (*false negatives*). Penelitian ini menegaskan bahwa untuk membangun sistem

P-ISSN : 2089-3353
E-ISSN : 2808-9162

AIOps yang andal, diperlukan dataset yang diekstraksi langsung dari lingkungan produksi nyata seperti *GitHub Actions* atau *Jenkins log* yang memiliki kausalitas otentik.

3.7. Analisis *Deep Error* dan Karakteristik Distribusi Data

Rendahnya akurasi sebesar 33,22% menuntut analisis lebih mendalam mengenai distribusi fitur di dalam *dataset* yang digunakan. Dalam klasifikasi tiga kelas (*Build, Deploy, Test*), model *Random Forest* gagal menemukan *decision boundary* atau batas keputusan yang jelas. Hal ini disebabkan oleh fenomena homogenitas fitur, di mana nilai metrik seperti *cpu_usage_pct* dan *memory_usage_mb* memiliki rentang nilai yang hampir identik di seluruh kategori kegagalan.

Secara teoretis, kegagalan pada tahap *Build* seharusnya memiliki karakteristik penggunaan memori yang lebih rendah dibandingkan tahap *Test* yang menjalankan skenario pengujian beban. Namun, pada dataset sintetis ini, distribusi angka tersebut bersifat acak. Akibatnya, ketika model mencoba melakukan pemisahan simpul (*node splitting*) pada pohon keputusan, tidak ada pengurangan nilai *Gini Impurity* yang signifikan. Hal ini menjelaskan mengapa nilai *F1-score* di ketiga kelas hanya berkisar antara 0,32 hingga 0,35.

Lebih lanjut, analisis pada matriks kebingungan (Gambar 2) menunjukkan bahwa kesalahan prediksi tidak terpusat pada satu kelas tertentu, melainkan tersebar merata. Misalnya, kegagalan *Build* yang salah diprediksi sebagai *Deploy* sebanyak 1.051 kali mengindikasikan bahwa model melakukan klasifikasi tanpa dasar pola yang kuat.

Fenomena ini mempertegas bahwa tanpa adanya data log operasional yang menyimpan urutan penyebab kegagalan (*causal sequence*) yang otentik , model AI hanya akan berperan sebagai fungsi pemetaan acak yang tidak memiliki nilai aplikatif di lingkungan produksi DevOps yang sebenarnya.

4. Kesimpulan

4.1. Ringkasan Temuan

Berdasarkan rangkaian eksperimen, penelitian ini berhasil membuktikan secara empiris bahwa dataset log sintetis tidak memiliki pola kausalitas yang valid untuk mendukung pemodelan AI pada tahapan kegagalan CI/CD.

Meskipun arsitektur model *Random Forest* dengan konfigurasi 100 trees telah berhasil dibangun dan mencapai akurasi absolut sebesar 100,00% pada fase pelatihan, hasil pengujian pada 9.000 data uji baru hanya menghasilkan akurasi keseluruhan sebesar 33,22% (setara probabilitas tebakan acak untuk klasifikasi tiga kelas) dengan nilai *F1-score* konstan di kisaran 0,32 hingga 0,35. Analisis *feature importance* juga mengonfirmasi ketiadaan fitur dominan, di mana setiap metrik sistem hanya memiliki bobot yang tersebar merata di kisaran angka 0,08.

Rendahnya tingkat akurasi uji di tengah sempurnanya akurasi latih ini membuktikan secara nyata terjadinya fenomena overfitting yang ekstrem. Hal ini memberikan kontribusi temuan yang krusial bagi pengembangan bidang AIOps: bahwa algoritma secanggih apa pun akan gagal jika dilatih menggunakan dataset buatan yang tidak memiliki sekuens sebab-akibat *(causal sequence*) yang natural. Oleh karena itu, penggunaan data log operasional asli sangat diwajibkan untuk menjamin keandalan prediksi di lingkungan *DevOps*.

4.2. Saran dan Penelitian Mendatang

Berdasarkan keterbatasan yang ditemukan dalam penelitian ini, terdapat beberapa saran strategis untuk pengembangan sistem prediksi kegagalan di masa depan.

Pertama, penggunaan dataset operasional sekunder yang diekstraksi langsung dari repositori industri nyata (seperti log dari *GitHub Actions, Jenkins*, atau *Travis CI*) sangat direkomendasikan. Data nyata mengandung anomali dan sekuens *error* yang memiliki logika teknis, sehingga memungkinkan model untuk mempelajari batas keputusan (*decision boundary*) yang lebih valid.

Kedua, untuk penelitian selanjutnya, eksplorasi menggunakan algoritma *Deep Learning* yang lebih kompleks seperti *Long Short-Term Memory* (LSTM) atau *Transformer* dapat menjadi alternatif yang menjanjikan. Algoritma tersebut memiliki kemampuan untuk menangani data log yang bersifat sekuensial dan *time-series*, sehingga diharapkan dapat menangkap dependensi waktu antar kegagalan guna meningkatkan akurasi prediksi.

Terakhir, integrasi fitur-fitur tambahan seperti analisis teks pada pesan kesalahan (*error message*) menggunakan *Natural Language Processing* (NLP) dapat dipertimbangkan untuk memperkaya informasi prediktor dan membangun sistem deteksi dini yang lebih andal dalam ekosistem *DevOps modern*.

Selain pengembangan teknis, dari sisi operasional, peneliti menyarankan agar tim *DevOps* mulai menerapkan praktik pencatatan log yang terstandarisasi dengan menyertakan metadata tambahan, seperti ID *runner* dan spesifikasi lingkungan eksekusi secara eksplisit. Ketersediaan log yang terstruktur dengan baik tidak hanya memudahkan proses *debugging* secara manual, tetapi juga menjadi prasyarat krusial agar model *AIOps* di masa depan dapat mengekstrak pola-pola kausalitas yang lebih reliabel. Dengan demikian, kualitas data log operasional harus diprioritaskan setara dengan kualitas kode sumber itu sendiri demi tercapainya ekosistem *DevOps* yang lebih cerdas dan proaktif.

P-ISSN : 2089-3353
E-ISSN : 2808-9162

Daftar Rujukan

[1] J. Smith dan A. Kumar, "The Evolution of DevOps Practices: A Decade in Review," *IEEE Access*, vol. 10, hlm. 45120–45135, 2022, doi: 10.1109/ACCESS.2022.3168234.

[2] L. Chen, "Continuous Integration and Deployment in Modern Software Engineering," *Empir. Softw. Eng.*, vol. 28, no. 3, hlm. 55, 2023, doi: 10.1007/s10664-023-10311-2.

[3] R. Ali dan others, "Analyzing CI/CD Pipelines in Open Source Projects," *Inf. Softw. Technol.*, vol. 144, hlm. 106798, 2022, doi: 10.1016/j.infsof.2021.106798.

[4] M. Waseem, P. Liang, dan M. Shahin, "DevOps Transformation: A Systematic Literature Review," *IEEE Transactions on Software Engineering*, vol. 50, no. 1, hlm. 200–221, 2024, doi: 10.1109/TSE.2023.3325678.

[5] K. B. Olesen dan J. Y. Lee, "Automated Software Engineering in the Cloud: CI/CD Fundamentals," *Applied Sciences*, vol. 13, no. 4, hlm. 2450, 2023, doi: 10.3390/app13042450.

[6] T. Dingsøyr dan N. B. Moe, "Accelerating Release Cycles with Advanced CI/CD Pipelines," *IEEE Softw.*, vol. 39, no. 2, hlm. 34–41, 2022, doi: 10.1109/MS.2021.3130112.

[7] P. Anderson dan others, "Performance Evaluation of CI Tools in Agile Environments," *Empir. Softw. Eng.*, vol. 28, no. 5, hlm. 112, 2023, doi: 10.1007/s10664-023-10355-w.

[8] S. Gupta dan D. Sharma, "Scaling Continuous Integration in Enterprise Microservices Architecture," *IEEE Access*, vol. 12, hlm. 11020–11035, 2024, doi: 10.1109/ACCESS.2024.3351221.

[9] A. Zaidman, "Characterizing Build Failures in Continuous Integration Systems," *Journal of Systems and Software*, vol. 185, hlm. 111162, 2022, doi: 10.1016/j.jss.2021.111162.

[10] W. Lam dan others, "Understanding and Handling Flaky Tests in CI/CD Environments," *IEEE Transactions on Software Engineering*, vol. 49, no. 4, hlm. 1502–1520, 2023, doi: 10.1109/TSE.2022.3181122.

[11] M. Beller dan others, "Bottlenecks in Continuous Deployment Pipelines: An Empirical Study," *Software Quality Journal*, vol. 30, no. 1, hlm. 91–118, 2022, doi: 10.1007/s11219-021-09555-5.

[12] H. Jin dan K. Liu, "An Empirical Study of CI Build Failures and Resolution Strategies," *Electronics (Basel).*, vol. 12, no. 6, hlm. 1422, 2023, doi: 10.3390/electronics12061422.

[13] R. Torkar dan others, "The Cost of Broken Builds in Agile Teams," *IEEE Trans. Eng. Manag.*, vol. 69, no. 5, hlm. 1820–1834, 2022, doi: 10.1109/TEM.2020.3015560.

[14] E. Murphy-Hill dan others, "Impact of Build Breakages on Developer Productivity," *Inf. Softw. Technol.*, vol. 165, hlm. 107338, 2024, doi: 10.1016/j.infsof.2023.107338.

[15] C. Bezemer, "Measuring the Efficiency of CI/CD Operations in Large-Scale Repositories," *IEEE Access*, vol. 11, hlm. 55600–55615, 2023, doi: 10.1109/ACCESS.2023.3281145.

[16] F. Fagerholm dan others, "Delays and Disruptions in Automated Delivery Pipelines," *Empir. Softw. Eng.*, vol. 27, no. 4, hlm. 95, 2022, doi: 10.1007/s10664-022-10156-1.

[17] J. Penix, "Limitations of Static Rule-Based CI Monitoring," *IEEE Softw.*, vol. 40, no. 3, hlm. 22–28, 2023, doi: 10.1109/MS.2023.3245110.

[18] Y. Wang dan Z. Zhang, "Manual vs Automated Monitoring in DevOps," *Information*, vol. 13, no. 8, hlm. 385, 2022, doi: 10.3390/info13080385.

[19] S. H. Setiyani dan Y. Rahma, "Penerapan Dekomposisi Matriks untuk Reduksi Kompleksitas Komputasi pada Algoritma Machine Learning," Apr 2026, doi: https://doi.org/10.37859/jf.v16i1.11166.

[20] A. Navidkya dan M. Yusuf, "Analisis Sentimen Isu Artificial Intelligence di Twitter dengan SVM dan Random Forest," Apr 2026, doi: https://doi.org/10.37859/jf.v16i1.10037.