Jurnal Sistem Informasi dan Sistem Komputer, Vol. 11, No. 2, Juli 2026
ISSN: 2715-906X (Online)
10.51717/simkom.v11i2.1414

PREDIKSI KEGAGALAN *BUILD REAL-TIME* PADA *PIPELINE* CI/CD MENGGUNAKAN PENDEKATAN *MACHINE LEARNING* BERBASIS DATA *STREAMING*

Hirmayanti[1*], Harianto[2], Rooy Marthen Thaniket[3]
[1]hirmayanti@hamzanwadi.ac.id, [2]harianto.27@hamzanwadi.ac.id[2], rooythaniket@gmail.com[3]
[1,2]Informatika, Universitas Hamzanwadi,[3]Informatika, Universitas Satya Wiyata Mandala

Abstrak

Kegagalan proses *build* pada lingkungan *Continuous Integration*/*Continuous Deployment* (CI/CD) dapat menghambat rilis perangkat lunak, meningkatkan biaya perbaikan, dan menurunkan efisiensi pengembangan. Hal ini menunjukkan perlunya sistem prediksi yang cepat, adaptif, serta mampu menangani data yang terus mengalir secara *real-time*. Penelitian ini bertujuan untuk menganalisis dan membandingkan kinerja beberapa algoritma *online machine learning* dalam memprediksi kegagalan *build* pada *pipeline* CI/CD. Dataset yang digunakan berasal dari TravisTorrent yang mencakup aktivitas repositori, perubahan kode, metrik pengujian, serta riwayat *build*. Tahapan penelitian meliputi *preprocessing* data, pembentukan skema data *streaming*, penerapan model *Hoeffding Tree*, *Adaptive Random Forest* (ARF), dan *Online Logistic Regression*, serta evaluasi menggunakan metrik *accuracy*. Hasil eksperimen menunjukkan model terbaik mencapai *accuracy* 97,35%, lebih tinggi dibandingkan penelitian sebelumnya sebesar 95,9%. Peningkatan absolut sebesar 1,45 poin persentase atau sekitar 1,51% ini membuktikan bahwa pendekatan yang digunakan lebih efektif dalam mengenali pola kegagalan *build* pada lingkungan CI/CD berbasis *streaming data*.

Kata kunci: *Adaptive Random Forest,* Prediksi Kegagalan *Build, CI/CD, Hoeffding Tree, Online Machine Learning*.

Abstract

Failure in the build process within Continuous Integration/Continuous Deployment (CI/CD) environments can delay software releases, increase maintenance costs, and reduce development efficiency. This highlights the need for a predictive system that is fast, adaptive, and capable of handling continuously evolving real-time data streams. This study aims to analyze and compare the performance of several online machine learning algorithms in predicting build failures in CI/CD pipelines. The dataset used is sourced from TravisTorrent, which contains repository activity, code changes, testing metrics, and build history information. The research process includes data preprocessing, constructing a streaming data framework, implementing Hoeffding Tree, Adaptive Random Forest (ARF), and Online Logistic Regression models, and evaluating their performance using accuracy as the main metric. The experimental results show that the proposed model achieves an accuracy of 97.35%, outperforming previous research with 95.9%. This represents an absolute improvement of 1.45 percentage points or about 1.51%, demonstrating better failure pattern detection in streaming-based CI/CD environments.

Kata kunci: *Adaptive Random Forest, Build Failure Prediction, CI/CD, Hoeffding Tree, Online Machine Learning.*

1. Pendahuluan

Perkembangan praktik DevOps telah mendorong implementasi *Continuous Integration* dan *Continuous Deployment* (CI/CD) sebagai pendekatan utama dalam pengembangan perangkat lunak modern. CI/CD memungkinkan proses integrasi kode, pengujian, dan *deployment* dilakukan secara otomatis sehingga mampu meningkatkan efisiensi pengembangan perangkat lunak. Namun, tingginya frekuensi commit, perubahan *source code* yang cepat, serta kompleksitas pengujian menyebabkan proses *build* pada CI/CD *pipeline* rentan mengalami kegagalan. Kegagalan *build* dapat berdampak pada keterlambatan *deployment*, penurunan produktivitas tim pengembang, serta meningkatnya biaya pemeliharaan perangkat lunak.

Jurnal Sistem Informasi dan Sistem Komputer, Vol. 11, No. 2, Juli 2026
ISSN: 2715-906X (Online)
10.51717/simkom.v11i2.1414

Penelitian sebelumnya menunjukkan bahwa *build failure* dipengaruhi oleh perubahan kode, aktivitas pengembang, kualitas pengujian, dan durasi *build* yang tidak stabil [1]. Selain itu, karakteristik *repository, source code churn*, serta tingginya aktivitas kolaborasi pengembang juga menjadi faktor penting yang memengaruhi stabilitas *build* pada lingkungan *continuous integration* [2]. Studi lain menyebutkan bahwa meningkatnya kompleksitas perangkat lunak modern menyebabkan prediksi *build failure* menjadi tantangan penting dalam *software engineering* berbasis *DevOps* [3]. Implementasi CI/CD yang semakin luas pada proyek *open-source* maupun enterprise juga meningkatkan kebutuhan terhadap sistem prediksi kegagalan *build* yang adaptif dan otomatis [4].

Pendekatan tersebut kurang sesuai dengan karakteristik lingkungan DevOps modern yang bersifat dinamis dan menghasilkan data secara terus-menerus. Lingkungan CI/CD memiliki karakteristik *streaming data* yang menyebabkan terjadinya *concept drift*, yaitu perubahan distribusi data akibat perubahan *source code*, *dependency*, konfigurasi *build*, maupun perilaku pengembang [5]. Kondisi tersebut menyebabkan performa model *machine learning* konvensional dapat menurun ketika diterapkan pada data *real-time*. Oleh karena itu, pendekatan *online machine learning* menjadi solusi potensial karena mampu melakukan pembelajaran secara incremental dan adaptif terhadap perubahan data *streaming* [6]. Selain itu, metode *adaptive windowing* seperti ADWIN juga telah banyak digunakan untuk mendeteksi *concept drift* pada data *stream* secara otomatis [7].

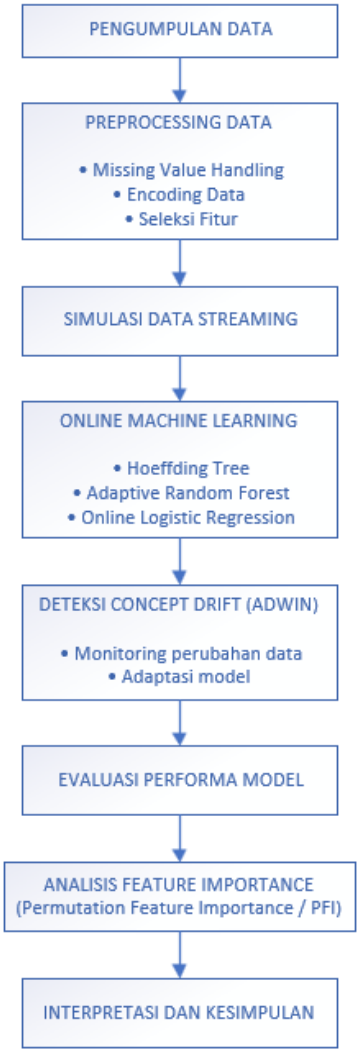
Berdasarkan permasalahan tersebut, penelitian ini mengusulkan pendekatan *online machine learning* untuk memprediksi kegagalan *build* secara *real-time* pada *Continuous Integration/Continuous Deployment* (CI/CD) menggunakan dataset TravisTorrent. Model yang dikembangkan memanfaatkan algoritma *Hoeffding Tree* dan *Adaptive Random Forest* yang dipadukan dengan mekanisme ADWIN sebagai pendeteksi *concept drift*, sehingga mampu menyesuaikan diri terhadap perubahan pola data yang terus berkembang pada lingkungan *streaming*. Berbeda dengan sebagian besar penelitian sebelumnya yang masih menerapkan pendekatan *offline learning* pada data statis dan memerlukan proses pelatihan ulang ketika data baru tersedia, penelitian ini mengadopsi pembelajaran inkremental yang memungkinkan model diperbarui secara berkelanjutan tanpa membangun ulang keseluruhan model.

Selain itu, penelitian terdahulu umumnya berfokus pada peningkatan akurasi prediksi, sementara aspek adaptasi terhadap perubahan distribusi data dan interpretasi faktor penyebab kegagalan *build* masih jarang dikaji. Oleh karena itu, penelitian ini mengintegrasikan *online learning*, *concept drift detection*, dan *feature importance analysis*. Pendekatan tersebut diharapkan menghasilkan model yang lebih adaptif, akurat, dan interpretatif sehingga dapat mendukung praktisi *DevOps* dalam meningkatkan stabilitas, keandalan, dan efisiensi pengelolaan *CI/CD pipeline* pada lingkungan pengembangan perangkat lunak modern.

2. Metode

Penelitian ini menggunakan pendekatan kuantitatif eksperimental untuk membangun sistem prediksi kegagalan *build* secara *real-time* pada lingkungan *Continuous Integration/Continuous Deployment* (CI/CD) menggunakan metode *online machine learning*.

Jurnal Sistem Informasi dan Sistem Komputer, Vol. 11, No. 2, Juli 2026
ISSN: 2715-906X (Online)
10.51717/simkom.v11i2.1414



Gambar 1. Metodologi Penelitian

Penelitian ini menerapkan pendekatan *streaming* data untuk mensimulasikan kondisi *real-time* pada CI/CD *pipeline*. Data diproses secara *incremental* menggunakan *library River* sehingga setiap data *build* dipelajari secara bertahap sesuai urutan kedatangan data. Pendekatan ini digunakan untuk merepresentasikan kondisi lingkungan *DevOps* modern yang menghasilkan data secara *continue* dan dinamis.

Sebelum dataset digunakan pada proses pelatihan model, terlebih dahulu dilakukan tahap *preprocessing* untuk meningkatkan kualitas. Proses diawali dengan penanganan *missing value* melalui identifikasi atribut yang memiliki nilai kosong. Nilai yang hilang pada atribut numerik digantikan menggunakan nilai median karena lebih tahan terhadap pengaruh *outlier*, sedangkan atribut kategorikal di isi menggunakan nilai modus. Selanjutnya, atribut kategorikal dikonversi ke dalam bentuk numerik menggunakan *Encoding* [8]. Dalam dataset ini terdapat data yang memiliki nilai atribut kategorikal, sehingga perlu dilakukan *Encoding*, karena *machine* hanya memproses data numerik. Tahap berikutnya adalah seleksi fitur dengan cara memilih fitur yang paling relevan menggunakan *Permutation Feature Importance (PFI)* sehingga hanya atribut yang memiliki pengaruh besar terhadap kegagalan *build* yang digunakan sebagai masukan model [9].

Model prediksi dibangun menggunakan pendekatan *online machine learning* yang mampu melakukan pembelajaran secara incremental tanpa harus melakukan retraining penuh ketika data baru masuk. Penelitian ini menggunakan algoritma *Hoeffding Tree* dan *Adaptive Random Forest* karena kedua

Jurnal Sistem Informasi dan Sistem Komputer, Vol. 11, No. 2, Juli 2026
ISSN: 2715-906X (Online)
10.51717/simkom.v11i2.1414

algoritma tersebut memiliki kemampuan adaptif terhadap data *stream* dan perubahan pola data. Pada setiap iterasi *streaming*, model melakukan proses prediksi, pembaruan metrik evaluasi, deteksi *concept drift*, serta pembelajaran terhadap data baru secara otomatis.

Untuk mengatasi perubahan distribusi data pada lingkungan CI/CD, penelitian ini menggunakan metode *Adaptive Windowing* (ADWIN) sebagai mekanisme *concept drift detection*. ADWIN bekerja dengan memonitor perubahan distribusi *error* prediksi pada data streaming. Ketika terjadi perubahan pola data yang signifikan, model akan menyesuaikan proses pembelajaran agar tetap mampu mempertahankan performa prediksi secara stabil pada lingkungan *real-time*.

Evaluasi performa model dilakukan menggunakan beberapa *confusion matrix*. Selain itu, *confusion matrix* digunakan untuk menganalisis kemampuan model dalam membedakan *build* berhasil dan *build* gagal. Seluruh proses implementasi dilakukan menggunakan bahasa pemrograman *Python* dengan dukungan *library Pandas, River, Scikit-learn*, dan *Matplotlib* pada lingkungan *Google Colaboratory*.

2.1. Dataset

Dataset yang digunakan dalam penelitian ini adalah TravisTorrent Dataset yang merupakan dataset publik pada bidang *DevOps* dan rekayasa perangkat lunak [10]. Dataset tersebut terdiri dari 2.640.825 baris data dengan 56 atribut yang berisi histori *build*, aktivitas commit, pengujian perangkat lunak, serta berbagai metrik *source code* yang berkaitan dengan proses CI/CD. Penelitian ini memanfaatkan fitur-fitur yang berkaitan langsung dengan aktivitas *build* dan *source code* seperti *gh_src_churn, gh_test_churn, tr_duration, tr_tests_fail, gh_files_modified,* dan *git_num_commits.*

2.2. Models

Continuous Integration (CI) dan *Continuous Deployment* (CD) merupakan praktik dalam rekayasa perangkat lunak yang mengotomatisasi proses integrasi kode, pengujian, dan distribusi aplikasi secara berkelanjutan sehingga setiap perubahan dapat divalidasi lebih cepat [11]. Penerapan CI/CD memungkinkan proses pengembangan menjadi lebih efisien karena setiap modifikasi kode langsung melewati rangkaian pengujian otomatis sebelum diterapkan pada lingkungan berikutnya [4]. Dalam proses implementasinya, setiap perubahan kode akan memicu *pipeline* yang menjalankan proses *build*, pengujian, hingga *deployment* sehingga status keberhasilan atau kegagalan dapat diketahui secara otomatis [3].

Build Failure Prediction merupakan pendekatan yang digunakan untuk memperkirakan kemungkinan terjadinya kegagalan pada proses *build* sebelum seluruh tahapan *pipeline* selesai di eksekusi. Prediksi dilakukan dengan memanfaatkan data historis dan karakteristik aktivitas pengembangan seperti perubahan kode, histori *commit*, durasi *build*, dan hasil pengujian sebelumnya. Informasi tersebut digunakan untuk mengenali pola yang sering muncul pada kondisi *build* gagal maupun *build* berhasil sehingga model dapat menghasilkan estimasi terhadap proses *build* berikutnya [5].

Online Machine Learning merupakan pendekatan pembelajaran mesin yang memperbarui model secara bertahap setiap kali menerima data baru tanpa perlu melakukan pelatihan ulang menggunakan seluruh dataset. Metode ini berbeda dari pendekatan tradisional karena proses pembelajaran dan prediksi berlangsung secara simultan selama sistem menerima aliran data. Setiap data yang masuk akan digunakan untuk menghasilkan prediksi terlebih dahulu, kemudian hasil aktual digunakan untuk memperbarui parameter model agar mampu menyesuaikan pola terbaru. Karakteristik tersebut membuat *online machine learning* lebih sesuai digunakan pada lingkungan yang menghasilkan data secara terus-menerus dan dinamis seperti sistem CI/CD [12].

Hoeffding Tree merupakan algoritma *decision tree* yang dikembangkan untuk menangani data streaming dalam skala besar dengan kebutuhan penyimpanan dan komputasi yang lebih efisien. Algoritma ini bekerja dengan membangun struktur pohon keputusan secara bertahap berdasarkan data yang diterima secara berurutan. *Node* pada pohon akan diperbarui secara incremental hingga informasi yang terkumpul cukup untuk menentukan pemisahan data berikutnya [5]. Pendekatan ini memungkinkan model tetap melakukan pembelajaran tanpa harus menyimpan seluruh data historis sehingga cocok diterapkan pada skenario klasifikasi *real-time.*

Adaptive Random Forest merupakan pengembangan dari metode *Random Forest* yang dirancang untuk menangani data streaming dan kondisi perubahan distribusi data. Algoritma ini bekerja dengan membangun beberapa *decision tree* secara bersamaan dan menggabungkan hasil prediksi dari seluruh pohon untuk menghasilkan keputusan akhir yang lebih stabil. Ketika pola data berubah, model dapat

Jurnal Sistem Informasi dan Sistem Komputer, Vol. 11, No. 2, Juli 2026
ISSN: 2715-906X (Online)
10.51717/simkom.v11i2.1414

menyesuaikan proses pembelajaran melalui mekanisme adaptasi sehingga performa prediksi tetap terjaga. Kemampuan tersebut menjadikan *Adaptive Random Forest* banyak digunakan pada penelitian klasifikasi data dinamis dan prediksi berbasis *streaming* [13].

Concept Drift merupakan kondisi ketika hubungan antara data masukan dan target prediksi mengalami perubahan seiring waktu. Perubahan ini dapat menyebabkan penurunan performa model apabila model tidak melakukan adaptasi terhadap kondisi terbaru. Deteksi *concept drift* dilakukan dengan memantau perubahan distribusi data dan performa prediksi selama proses pembelajaran berlangsung. Setelah perubahan teridentifikasi, model dapat menyesuaikan pembelajaran menggunakan data terbaru agar tetap relevan terhadap kondisi lingkungan yang berubah [14].

Adaptive Windowing (ADWIN) merupakan metode deteksi *concept drift* yang bekerja dengan membandingkan distribusi data pada beberapa periode waktu berbeda dalam aliran data. Ketika sistem menemukan perbedaan signifikan antara data lama dan data terbaru, maka perubahan tersebut dianggap sebagai *drift*. Setelah *drift* terdeteksi, proses pembelajaran akan difokuskan kembali pada data terbaru sehingga model dapat mempertahankan performanya. Integrasi ADWIN dengan *online machine learning* terbukti membantu meningkatkan kemampuan adaptasi model pada lingkungan *streaming* yang dinamis [7].

3. Hasil dan Pembahasan

Penelitian ini bertujuan untuk mengukur dan membandingkan kinerja beberapa algoritma *online machine learning* dalam memprediksi kegagalan *build* pada lingkungan CI/CD *Pipeline real-time*. Evaluasi dilakukan melalui perbandingan tiga model klasifikasi yang menggunakan pendekatan pembelajaran adaptif, yaitu *Hoeffding Tree*, *Adaptive Random Forest*, dan *Online Logistic Regression*. Pengukuran performa menggunakan metrik *accuracy* sebagai indikator utama dalam menilai kemampuan setiap model dalam membedakan kondisi *build* yang berhasil dan mengalami kegagalan.

Berdasarkan hasil eksperimen yang ditampilkan pada Gambar 2, diperoleh bahwa *Adaptive Random Forest* memberikan performa paling optimal dibandingkan dua model lainnya. Model ini memperoleh nilai akurasi sebesar 97.35%, yang menunjukkan kemampuan klasifikasi yang tinggi terhadap data *streaming* yang digunakan dalam proses prediksi kegagalan *build*. Sementara itu, *Hoeffding Tree* mencatat akurasi sebesar 88.74%, sedangkan *Online Logistic Regression* menghasilkan akurasi sebesar 87.40%. Perbedaan capaian performa tersebut menunjukkan bahwa pendekatan berbasis *ensemble learning* pada bahwa *Adaptive Random Forest* lebih efektif dalam menangani variasi dan kompleksitas pola data dibandingkan model tunggal. Hal ini mengindikasikan bahwa kombinasi beberapa model keputusan memberikan kemampuan generalisasi yang lebih baik pada lingkungan data yang bersifat dinamis seperti sistem CI/CD.

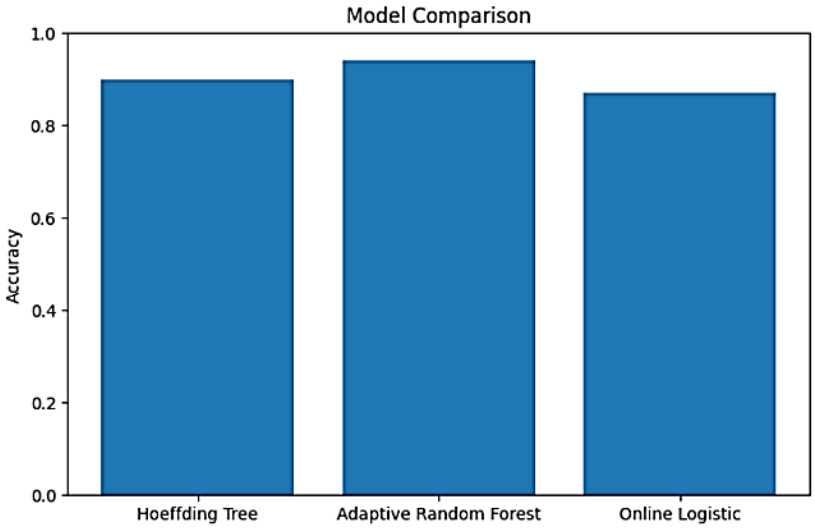
Keunggulan *Adaptive Random Forest* tidak terlepas dari penerapan metode *Ensemble Learning*, yang menggabungkan beberapa pohon keputusan sehingga mampu menekan varians prediksi sekaligus meningkatkan kemampuan generalisasi model. Selain itu, setiap pohon diperbarui secara inkremental dan didukung oleh mekanisme *Adaptive Windowin* untuk mendeteksi terjadinya *concept drift*. Mekanisme ini memungkinkan model menyesuaikan diri secara otomatis ketika terjadi perubahan distribusi data yang dipengaruhi oleh modifikasi kode, perubahan konfigurasi proyek, maupun aktivitas pengembang. Dengan kemampuan tersebut, *Adaptive Random Forest* mampu mempertahankan performa prediksi secara lebih konsisten pada data *streaming* [13], sedangkan *Hoeffding Tree* yang hanya mengandalkan satu pohon keputusan cenderung lebih sensitif terhadap perubahan pola data.

Di sisi lain, *Online Logistic Regression* menghasilkan performa yang lebih rendah karena merupakan model linier yang mengasumsikan hubungan linier antara variabel masukan dan keluaran [15]. Asumsi tersebut kurang sesuai dengan karakteristik data CI/CD yang umumnya memiliki hubungan nonlinier akibat interaksi yang kompleks antara perubahan kode, hasil pengujian, dan riwayat *build*. Sebaliknya, *Adaptive Random Forest* mampu memodelkan hubungan yang lebih kompleks melalui kombinasi beberapa pohon keputusan, sehingga menghasilkan prediksi yang lebih akurat.

Hasil penelitian ini menunjukkan bahwa pendekatan *ensemble* berbasis *online learning* lebih efektif diterapkan pada lingkungan CI/CD berbasis *streaming* dibandingkan model linier maupun model berbasis satu pohon keputusan. Temuan ini sejalan dengan penelitian [13], yang menyatakan bahwa *Adaptive Random Forest* memiliki kemampuan adaptasi yang baik terhadap *concept drift* serta mampu mempertahankan kinerja klasifikasi pada distribusi data yang terus berubah. Oleh karena itu, *Adaptive Random Forest* berpotensi menjadi solusi yang andal untuk mendukung deteksi dini kegagalan *build*,

Jurnal Sistem Informasi dan Sistem Komputer, Vol. 11, No. 2, Juli 2026
ISSN: 2715-906X (Online)
10.51717/simkom.v11i2.1414

sehingga stabilitas, efisiensi, dan keandalan proses pengembangan perangkat lunak berbasis DevOps dapat ditingkatkan.



Gambar 2. Perbandingan Kinerja Model *Online Machine Learning*

Selain melakukan evaluasi terhadap kemampuan prediksi masing-masing model, penelitian ini juga melakukan analisis lebih lanjut terhadap kontribusi setiap fitur yang digunakan dalam proses klasifikasi. Analisis tersebut dilakukan menggunakan pendekatan *Permutation Feature Importance* (PFI) untuk mengidentifikasi fitur-fitur yang memiliki pengaruh paling besar terhadap hasil prediksi kegagalan *build* pada lingkungan CI/CD. Pendekatan ini dipilih karena mampu memberikan interpretasi terhadap kontribusi variabel tanpa hanya berfokus pada nilai performa akhir model [9]. *Feature importance* melalui mekanisme permutasi bekerja dengan mengamati perubahan performa model ketika nilai suatu fitur diacak, sehingga fitur yang menyebabkan penurunan performa paling besar dianggap memiliki kontribusi yang lebih signifikan [16].

Dengan mengetahui fitur yang paling berpengaruh, penelitian ini tidak hanya menghasilkan model dengan kemampuan prediksi yang baik, tetapi juga memberikan pemahaman yang lebih mendalam mengenai karakteristik data yang berperan dalam proses deteksi kegagalan *build* secara *real-time*. Hasil analisis PFI menunjukkan bahwa fitur tr_duration menjadi faktor paling dominan dengan nilai importance sebesar 0,143759, diikuti oleh *gh_sloc*, *tr_tests_fail*, *gh_asserts_cases_per_kloc*, dan *tr_ci_latency*.

Hasil analisis *feature importance* tidak hanya dimanfaatkan sebagai informasi tambahan, tetapi juga menjadi dasar untuk memahami pola yang menyebabkan terjadinya kegagalan *build* pada CI/CD *pipeline*. Dengan demikian, penelitian ini memberikan kontribusi tidak hanya pada peningkatan akurasi prediksi, tetapi juga pada aspek interpretabilitas model dan identifikasi faktor-faktor yang memengaruhi performa sistem secara lebih transparan [13].

Dibandingkan dengan penelitian Al-Baltah dkk., pendekatan yang digunakan dalam penelitian ini menghasilkan performa prediksi yang lebih tinggi dengan nilai akurasi sebesar 97,35%, sehingga mengindikasikan efektivitas pendekatan yang diusulkan pada skenario prediksi kegagalan *build* [10].

Jurnal Sistem Informasi dan Sistem Komputer, Vol. 11, No. 2, Juli 2026
ISSN: 2715-906X (Online)
10.51717/simkom.v11i2.1414

Tabel 1. Hasil Analisis *Feature Importance*

No	Fitur	Skor
1	tr_duration	0.143759
2	gh_sloc	0.060131
3	tr_tests_fail	0.044710
4	gh_asserts_cases_per_kloc	0.033785
5	tr_ci_latency	0.033025
6	gh_test_cases_per_kloc	0.028512
7	gh_test_lines_per_kloc	0.027192
8	gh_team_size	0.022900
9	gh_description_complexity	0.011792
10	tr_tests_ok	0.009130
11	gh_commits_on_files_touched	0.006867
12	tr_tests_run	0.006699
13	tr_setup_time	0.006518
14	git_num_commits	0.004132
15	tr_tests_skipped	0.003912
16	gh_src_churn	0.003832
17	gh_test_churn	0.001975
18	gh_src_files	0.000995
19	gh_files_modified	0.000711
20	gh_other_files	0.000557
21	gh_num_issue_comments	0.000297
22	gh_doc_files	0.000282
23	gh_num_commit_comments	0.000239
24	gh_files_added	0.000180
25	gh_num_pr_comments	0.000122
26	gh_files_deleted	0.000098
27	gh_tests_deleted	0.000066
28	gh_tests_added	0.000026

4. Kesimpulan dan Saran

4.1 Kesimpulan

Penelitian ini menunjukkan bahwa penerapan *online machine learning* pada prediksi kegagalan *build* di lingkungan CI/CD *Pipeline* mampu menghasilkan sistem prediksi yang adaptif terhadap perubahan data secara *real-time* tanpa memerlukan pelatihan ulang secara menyeluruh. Hal ini menunjukkan bahwa pendekatan pembelajaran bertahap dapat mempertahankan kemampuan klasifikasi pada kondisi data yang terus berkembang. Dari ketiga algoritma yang dievaluasi, *Adaptive Random Forest* memberikan kinerja terbaik dengan *accuracy* sebesar 97,35%, mengungguli *Hoeffding Tree* yang memperoleh 88,74% dan *Online Logistic Regression* sebesar 87,40%. Pencapaian tersebut mengindikasikan bahwa pendekatan *ensemble* berbasis *online learning* lebih mampu menangani perubahan karakteristik data *streaming* dibandingkan model linier maupun model berbasis satu pohon keputusan.

Selain evaluasi performa model, penelitian ini juga melakukan analisis *feature importance* menggunakan *Permutation Feature Importance* untuk mengidentifikasi faktor yang paling berpengaruh terhadap prediksi kegagalan *build*. Hasil analisis tersebut menunjukkan bahwa karakteristik *proses build*, aktivitas pengujian, dan kompleksitas perubahan perangkat lunak menjadi aspek yang lebih dominan dalam menentukan terjadinya kegagalan. Adapun hasil analisis *Permutation Feature Importance* mengidentifikasi tr_*duration* sebagai fitur yang memberikan kontribusi terbesar dalam memprediksi kegagalan *build*, diikuti oleh beberapa fitur yang berkaitan dengan proses pengujian dan perubahan kode seperti *gh_sloc*, *tr_tests_fail*, *gh_asserts_cases_per_kloc*, dan *tr_ci_latency*. Temuan ini menunjukkan bahwa model yang dikembangkan tidak hanya memiliki tingkat akurasi yang tinggi, tetapi juga mampu memberikan interpretasi terhadap faktor-faktor utama yang memengaruhi kegagalan *build*. Oleh karena

itu, pendekatan yang diusulkan berpotensi mendukung peningkatan stabilitas, keandalan, dan efisiensi proses pengembangan perangkat lunak pada lingkungan *DevOps* berbasis CI/CD.

4.2 Saran

Kontribusi penelitian ini terletak pada penggabungan *online machine learning, concept drift detection* menggunakan ADWIN, dan analisis interpretabilitas berbasis *Permutation Feature Importance* untuk membangun sistem prediksi yang adaptif sekaligus dapat dijelaskan. Namun, penelitian ini masih memiliki keterbatasan karena evaluasi dilakukan pada dataset dan karakteristik lingkungan CI/CD tertentu sehingga hasilnya perlu divalidasi lebih lanjut pada platform atau proyek yang berbeda. Selain itu, analisis interpretabilitas masih dilakukan melalui pendekatan *offline* sehingga integrasi interpretabilitas secara langsung ke mekanisme *online machine learning* menjadi peluang pengembangan pada penelitian selanjutnya.

Daftar Pustaka

[1] I. Saidani, A. Ouni, M. Chouchen, and M. W. Mkaouer, "*Predicting Continuous Integration Build Failures Using Evolutionary Search*," *Inf. Softw. Technol.*, vol. 128, no. March, p. 106392, 2020, doi: 10.1016/j.infsof.2020.106392.

[2] A. Barrak, E. E. Eghan, B. Adams, and F. Khomh, "*Why Do Builds Fail?—A Conceptual Replication Study*," *J. Syst. Softw.*, 2021, doi: 10.1016/j.jss.2021.110939.

[3] R. Sharma, E. Petrova, and J. O. Connolly, "*Machine Learning – Based Failure Prediction in Continuous Integration and Deployment Workflows*," 2025.

[4] M. Hilton, T. Tunnell, K. Huang, D. Marinov, and D. Dig, "*Usage, Costs, And Benefits Of Continuous Integration In Open-Source Projects*" *ACM Int. Conf. Autom. Softw. Eng.*, pp. 426–437, 2016, doi: 10.1145/2970276.2970358.

[5] P. Domingos and G. Hulten, "*Mining High-Speed Data Streams*" *KDD Conf.*, pp. 71–80, 2000.

[6] A. Bifet, "*Adaptive Learning And Mining For Data Streams And Frequent Patterns*," *ACM SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 55–56, 2009, doi: 10.1145/1656274.1656287.

[7] A. Bifet and R. Gavaldà, "*Learning From Time-Changing Data With Adaptive Windowing*," *Proc. 7th SIAM Int. Conf. Data Min.*, pp. 443–448, 2007, doi: 10.1137/1.9781611972771.42.

[8] Hirmayanti and E. Utami, "*Enhanced Heart Disease Diagnosis Using Machine Learning Algorithms : A Comparison of Feature Selection*," *Rekayasa Sist. dan Teknol. Inf.*, vol. 5, no. 158, pp. 385–392, 2025, doi: https://doi.org/10.29207/resti.v9i2.6175.

[9] A. Fisher, C. Rudin, and F. Dominici, "*All Models are Wrong , but Many are Useful : Learning a Variable ' s Importance by Studying an Entire Class of Prediction Models Simultaneously*," *J. Mach. Learn. Res.*, vol. 20, pp. 1–81, 2019.

[10] I. A. Al-Baltah, N. Al-Shaibany, M. Abdellatief, M. M. Al-Gawda, and S. Y. Al-Sultan, "*An Intelligent Multi-Class XGBoost-Based Model for Optimizing DevOps Continuous Integration and Continuous Deployment Failure Prediction*," *Inf.*, vol. 17, no. 2, pp. 1–14, 2026, doi: 10.3390/info17020178.

[11] J. Humble and D. Farley, *Continuous Delivery: Reliable Software Releases Through Build, Test, And Deployment Automation*. 2011. doi: 10.1201/9781003221579-1.

[12] A. A. Benczúr, L. Kocsis, and R. Pálovics, "*Online Machine Learning in Big Data Streams: Overview*," *Encycl. Big Data Technol.*, pp. 1207–1218, 2019, doi: 10.1007/978-3-319-77525-8_326.

[13] H. M. Gomes *et al.*, "*Adaptive Random Forests For Evolving Data Stream Classification*," *Mach. Learn.*, vol. 106, no. 9–10, pp. 1469–1495, 2017, doi: 10.1007/s10994-017-5642-8.

[14] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "*A Survey on Concept Drift Adaptation," ACM Comput. Surv.*, vol. 46, no. 4, 2014, doi: 10.1145/2523813.

[15] J. L. Crowley, *Generative Networks: EigenSpace Coding, Auto-Encoders, Variational Autoencoders and Generative Adversarial Networks*. 2020. doi: 10.1017/9781009218276.020.

[16] A. Altmann, L. Tolo, O. Sander, and T. Lengauer, "*Permutation Importance : a Corrected Feature Importance Measure*," *Bioinformatics*, vol. 26, no. 10, pp. 1340–1347, 2010, doi: 10.1093/bioinformatics/btq134.