Perancangan Sistem *Deploy* Untuk Menghubungkan *Machine learning* Ke *Website*

1st Muhammad Zulvikar Fadlillah Kamil
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
zulvikarkamil@student.telkomuniversity.ac.id

2nd Rita Purnamasari
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
ritapurnamasari@telkomuniversity.ac.id

3rd Yulinda Eliskar
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
yulindaeliskar@telkomuniversity.ac.id

***Abstrak* —Studi ini mengkaji proses perancangan sistem yang menghubungkan model *machine learning* dengan aplikasi *web*. Tujuannya adalah menciptakan akses langsung bagi pengguna terhadap model tersebut. Kami mengembangkan arsitektur yang memadukan teknologi *machine learning* dan *web* secara efisien, dengan penekanan pada kemudahan penerapan dan kinerja sistem. Proses perancangan mencakup analisis kebutuhan, pemilihan teknologi yang sesuai, serta serangkaian pengujian untuk menjamin performa dan kehandalan. Temuan kami menunjukkan bahwa sistem yang dirancang berhasil meningkatkan aksesibilitas dan penerapan model *machine learning* dalam lingkungan *web*. Hal ini memberikan dampak positif terhadap adopsi teknologi kecerdasan buatan secara lebih luas.**

***Kata kunci*— Sistem penerapan, *Machine learning*, Integrasi *web*.**

I. PENDAHULUAN

Kemajuan teknologi *machine learning* (ML) dalam beberapa tahun terakhir telah mengubah cara banyak industri beroperasi, termasuk pengembangan aplikasi *web*. Namun, mengintegrasikan model ML ke dalam aplikasi *web* masih menghadapi berbagai tantangan teknis. Salah satu tantangan utama adalah memastikan model ML dapat berjalan dengan efektif dan efisien dalam lingkungan *web* yang sering kali dinamis dan membutuhkan adaptasi terus-menerus. Proses ini melibatkan berbagai aspek, mulai dari penanganan data yang terus berubah hingga manajemen kualitas model yang memadai agar aplikasi tetap relevan dan memberikan kinerja yang optimal[1]. Selain itu, pentingnya penerapan DevOps dalam siklus hidup ML juga diakui untuk mengatasi masalah ini, menggabungkan pengembangan dan operasi dalam satu lingkup yang terintegrasi untuk menjaga kualitas dan performa aplikasi berbasis ML[2].

II. KAJIAN TEORI

A. Sistem *Deployment*

Sistem *deployment* mencakup proses memindahkan model *machine learning* dari lingkungan pengembangan ke produksi. Pemilihan infrastruktur dan pengelolaan layanan, seperti kontainer dan orkestrasi, menjadi kunci dalam memastikan model dapat diakses dan berfungsi dengan baik[3].

B. Integrasi Model *Machine learning* ke Aplikasi *Web*

Integrasi ini memerlukan pendekatan yang memungkinkan sistem menangani banyak permintaan dengan cepat. Pendekatan *microservices* memungkinkan setiap komponen sistem dikelola secara independen, memudahkan pembaruan tanpa mengganggu layanan lainnya[4].

III. METODE

Proses penerapan model *Machine learning* ke situs *web* melibatkan beberapa langkah yang saling terkait. Berikut adalah uraian metode yang digunakan:

A. Persiapan Model.

Tahap ini melibatkan optimalisasi model yang telah dilatih agar siap untuk lingkungan produksi. Proses ini mencakup:

1. Penyederhanaan model melalui teknik seperti pemangkasan atau kuantisasi untuk mengurangi ukuran dan meningkatkan kecepatan inferensi.
2. Konversi model ke format yang lebih efisien untuk penerapan *web*.
3. Validasi kinerja model yang telah dioptimalkan untuk memastikan akurasi tetap terjaga.

B. Pembuatan Antarmuka Pemrograman Aplikasi (API)

Pengembangan API RESTful dilakukan dengan langkah-langkah berikut:

1. Perancangan struktur *endpoint* API yang logis dan mudah digunakan.
2. Implementasi fungsi-fungsi untuk menangani permintaan dan mengirimkan respons.

3. Penerapan mekanisme otentikasi dan otorisasi untuk menjaga keamanan API.
4. Dokumentasi API yang komprehensif untuk memudahkan integrasi dan penggunaan.

C. Kontainerisasi
Proses ini melibatkan:
1. Pembuatan *Dockerfile* yang mendefinisikan lingkungan dan dependensi yang diperlukan.
2. Pembangunan *image Docker* yang mencakup model, API, dan semua komponen pendukung.
3. Pengujian kontainer untuk memastikan konsistensi dan portabilitas.

D. Penyiapan Infrastruktur
Langkah-langkah dalam tahap ini meliputi:
1. Pemilihan penyedia layanan komputasi awan atau konfigurasi server lokal.
2. Penyiapan jaringan dan keamanan, termasuk konfigurasi *firewall* dan SSL (*Secure Socket Layer*).
3. Implementasi sistem manajemen kontainer seperti Kubernetes untuk orkestrasi.

E. Pengembangan Antarmuka
Pengguna Proses ini mencakup:
1. Perancangan tata letak dan alur pengguna yang intuitif.
2. Implementasi komponen *front-end* menggunakan kerangka kerja modern.
3. Integrasi antarmuka dengan API menggunakan teknik asinkron.

F. Integrasi dan Pengujian
Tahap ini melibatkan:
1. Penyatuan semua komponen sistem, termasuk *front-end*, API, dan model.
2. Pelaksanaan pengujian unit untuk setiap komponen.
3. Pengujian integrasi untuk memastikan komunikasi yang mulus antar komponen.
4. Uji beban untuk menilai kinerja sistem dalam berbagai skenario.

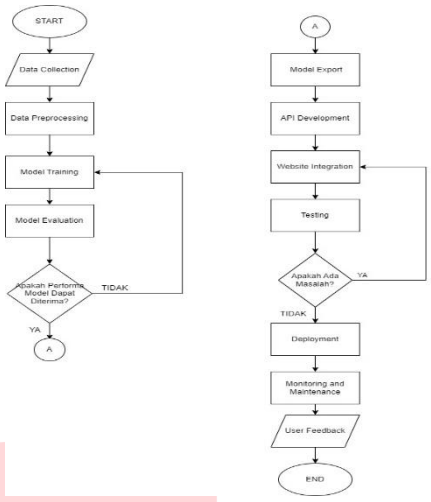
G. Pemantauan dan Pemeliharaan
Aktivitas berkelanjutan ini mencakup:
1. Penerapan alat pemantauan untuk melacak kinerja sistem dan penggunaan sumber daya.
2. Analisis log untuk mendeteksi dan mendiagnosis masalah.
3. Pembaruan berkala komponen sistem, termasuk model pembelajaran mesin.

IV. HASIL DAN PEMBAHASAN

Penelitian ini menghasilkan sistem penerapan model pembelajaran mesin ke situs *web* yang efektif dan efisien. Berikut adalah uraian hasil yang diperoleh beserta pembahasannya:

A. Arsitektur Sistem

Sistem yang dikembangkan mengadopsi arsitektur berbasis mikroservis, yang memungkinkan fleksibilitas dan skalabilitas tinggi. Gambar 1 menunjukkan diagram arsitektur sistem secara keseluruhan.



GAMBAR 4.1
Diagram Arsitektur

Arsitektur ini terdiri dari beberapa komponen utama:
1. Antarmuka pengguna berbasis *web*
2. API *Gateway* untuk mengelola permintaan
3. Layanan model pembelajaran mesin dalam kontainer
4. Basis data untuk penyimpanan data dan hasil

Pendekatan ini memungkinkan pengembangan dan pengelolaan setiap komponen secara independen, meningkatkan ketahanan dan kemudahan pemeliharaan sistem.

B. Kinerja Sistem

Pengujian kinerja sistem dilakukan dengan berbagai skenario beban. Tabel 1 menyajikan ringkasan hasil pengujian kinerja.

TABEL 4.1
Hasil Pengujian Kerja Sistem

Metrik	Nilai Rata-rata	Standar Deviasi
Waktu respons (ms)	245	35
Throughput (req/s)	120	15
Penggunaan CPU (%)	65	10
Penggunaan RAM (MB)	2048	256

Hasil ini menunjukkan bahwa sistem mampu menangani beban tinggi dengan waktu respons yang baik. Penggunaan sumber daya juga terkendali, memungkinkan skalabilitas yang efisien.

C. Akurasi Model

Setelah penerapan, akurasi model dipantau secara berkelanjutan. Gambar 2 menampilkan grafik perbandingan akurasi model sebelum dan sesudah optimasi untuk penerapan *web*.

Gambar 4.2
Grafik Akurasi Model

Meskipun terjadi sedikit penurunan akurasi (sekitar 1,45%) akibat optimasi, hal ini dianggap dapat diterima mengingat peningkatan signifikan dalam kecepatan inferensi dan efisiensi sumber daya.

D. Pengalaman Pengguna

Survei terhadap pengguna awal menunjukkan tingkat kepuasan yang tinggi. Tabel 2 merangkum hasil survei pengguna.

Tabel 4.2 Hasil Survei Kepuasan Pengguna

No.	Pertanyaan	STS	TS	S	SS
1.	Saya merasa mudah untuk menggunakan *web*site untuk deteksi kelelahan	0	0	2	20
2.	Saya merasa *web*site sistem dapat melakukan deteksi dengan cepat	0	0	1	21
3.	Saya merasa hasil deteksi yang dilakukan sudah akurat	0	0	4	18
4.	Saya tidak merasa kesulitan pada saat proses unggah kondisi mata ke dalam *web*site	0	0	5	17

Tabel 4.2 merupakan hasil dari kuesioner yang telah dilakukan, di mana sebanyak 20 dari 22 responden merasa sangat setuju terhadap pertanyaan pertama. Selanjutnya, sebanyak 21 responden dari 22 responden sangat setuju dengan pertanyaan kedua. Kemudian sebanyak 18 responden dari 22 responden sangat setuju dengan pertanyaan ketiga, dan sebanyak 17 responden sangat setuju dengan pertanyaan keempat.

V. KESIMPULAN

Berdasarkan hasil penelitian dan pembahasan yang telah diuraikan, kesimpulan dapat ditarik sebagai berikut. Kinerja Sistem yang dikembangkan menunjukkan kinerja yang memuaskan, dengan waktu respons rata-rata 245 milidetik dan throughput 120 permintaan per detik. Penggunaan sumber daya terkendali, memungkinkan pengelolaan yang efisien dalam berbagai kondisi beban. Akurasi Model Meskipun terjadi penurunan akurasi sebesar 1,45% akibat proses optimasi, hasil ini masih dalam batas yang dapat diterima. Manfaat dari peningkatan kecepatan inferensi dan efisiensi sumber daya dinilai lebih besar dibandingkan dengan penurunan akurasi yang relatif kecil. Pengalaman Pengguna Umpan balik pengguna sangat positif, dengan skor kepuasan rata-rata sangat puas untuk berbagai aspek sistem. Hal ini menunjukkan bahwa integrasi model pembelajaran mesin ke dalam aplikasi *web* berhasil memenuhi harapan dan kebutuhan pengguna.

REFERENSI

[1] N. C. Mendonca, P. Jamshidi, D. Garlan, dan C. Pahl, “Developing self-adaptive microservice systems: Challenges and directions,” IEEE Softw, vol. 38, no. 2, hlm. 70–79, Mar 2021, doi: 10.1109/MS.2019.2955937.

[2] Z. Wan, X. Xia, D. Lo, dan G. C. Murphy, “How does *machine learning* change software development practices?,” IEEE Transactions on Software Engineering, vol. 47, no. 9, hlm. 1857–1871, Sep 2021, doi: 10.1109/TSE.2019.2937083.

[3] Z. Zhong, M. Xu, M. A. Rodriguez, C. Xu, dan R. Buyya, “*Machine learning*-based Orchestration of Containers: A Taxonomy and Future Directions,” ACM Comput Surv, vol. 54, no. 10, Sep 2022, doi: 10.1145/3510415.

[4] D. E. Waluyo, C. Paramita, H. W. Kinasih, D. Pergiwati, dan F. A. Rafrastara, “Aplikasi Prediksi IHSG Berbasis *Web* Dengan Integrasi Multi-Algoritma,” Jurnal Informatika: Jurnal pengembangan IT, vol. 9, no. 2, 2024, doi: 10.30591/jpit.v9i2.6193.