JUSIFOR: Jurnal Sistem Informasi dan Informatika
Vol. 5, No. 1, Juni 2026, hlmn. 175-186
E-ISSN: 2830-3393 | P-ISSN: 2830-2443
DOI: https://doi.org/10.70609/jusifor.v5i1.9630

Sistem Prediktif Pemeliharaan Hidraulik dengan Pendekatan Algoritma Histogram Gradient Boosting

Faizah Via Fadhillah[1*], Muhammad Fatchan[2], Wahyu Hadikristanto[3]

[1,2,3] Prodi Teknik Informatika, Fakultas Teknik, Universitas Pelita Bangsa, Indonesia

Info Artikel

Riwayat Artikel:
Diterima : **20 April 2026**
Direvisi : **19 Mei 2026**
Diterbitkan : **01 Juni 2026**

Kata Kunci:

Predictive maintenance,
Sistem hidrolik,
Histogram Gradient Boosting,
Streamlit

ABSTRAK

Sistem hidrolik banyak digunakan dalam industri karena kepadatan daya tinggi dan kontrol presisi, namun kegagalan pada komponen seperti *cooler*, *valve*, *pump leakage*, dan *hydraulic accumulator* dapat menyebabkan *downtime* serta biaya perawatan tinggi. Pendekatan pemeliharaan korektif maupun preventif masih kurang optimal karena bergantung pada inspeksi manual atau jadwal tetap. Oleh karena itu, penelitian ini menerapkan *predictive maintenance* berbasis *machine learning* untuk memprediksi kondisi komponen hidrolik secara lebih dini. Penelitian ini bertujuan memprediksi kondisi empat komponen utama (*Cooler, Valve, Internal Pump Leakage,* dan *Hydraulic Accumulator*), mengevaluasi model Histogram *Gradient Boosting* (HGB) menggunakan *Accuracy* dan F1-*score* macro, serta menerapkan model ke dalam aplikasi Streamlit. *Dataset* yang digunakan adalah *Condition Monitoring of Hydraulic Systems* dengan 2205 siklus operasi. Data sensor dari file *.txt* diolah menjadi fitur tabular per siklus, distandarisasi menggunakan *StandardScaler*, dan dibagi menjadi data latih dan data uji dengan rasio 70:30. Model dibangun sebagai empat model HGB terpisah dan dievaluasi menggunakan *confusion matrix, Accuracy*, dan F1-*score* macro. Hasil menunjukkan performa sangat baik, yaitu *Cooler_condition* (*Accuracy* 0,998; F1-*score* 0,998), *Valve_condition* (0,834; 0,786), *Internal_pump_leakage* (0,968; 0,963), dan *Hydraulic_accumulator/bar* (0,989; 0,987). Model yang telah dilatih disimpan dalam format *.pkl* dan diintegrasikan ke aplikasi Streamlit untuk menampilkan hasil prediksi secara interaktif serta menyediakan output tabel dan unduhan CSV.

***Keywords*:**

Predictive maintenance,
Hydraulic system,
Histogram Gradient Boosting,
Streamlit

ABSTRACT

Hydraulic systems are widely used in industry due to high power density and precision control; however, failures in components such as coolers, valves, pump leakage, and hydraulic accumulators can cause downtime and high maintenance costs. Conventional corrective and preventive maintenance is limited due to reliance on fixed schedules and manual inspection. Therefore, this study applies machine learning-based predictive maintenance to predict hydraulic component conditions. This study aims to predict the condition of four components (Cooler, Valve, Internal Pump Leakage, and Hydraulic Accumulator), evaluate Histogram Gradient Boosting (HGB) using Accuracy and macro F1-score, and implement the model in a Streamlit application. The Condition Monitoring of Hydraulic Systems dataset (2,205 cycles) is used. Sensor data from .txt files is transformed into tabular features per cycle, standardized using StandardScaler, and split into 70:30 training and testing sets. Four independent HGB models are trained and evaluated using confusion matrix, Accuracy, and macro F1-score. Results show strong performance: Cooler_condition (Accuracy 0.998, F1 0.998), Valve_condition (0.834, 0.786), Internal_pump_leakage (0.968, 0.963), and Hydraulic_accumulator/bar (0.989, 0.987). The trained models are saved as .pkl files and integrated into a Streamlit application for interactive prediction and CSV export.

Penulis Korespondensi:
Faizah Via Fadhillah,
Prodi Teknik Informatika,
Universitas Pelita Bangsa
Email: faizahviafadhillah.312210460@mhs.pelitabangsa.ac.id

This is an open access article under the CC BY-SA license

1. PENDAHULUAN

Sistem hidrolik banyak digunakan dalam industri manufaktur, konstruksi, dan transportasi karena kepadatan daya tinggi dan kontrol presisi. Namun, kegagalan mendadak pada komponen seperti cooler, valve, pump leakage, dan hydraulic accumulator dapat menyebabkan downtime serta biaya perawatan yang besar. Studi terkini menekankan pentingnya diagnosis dan monitoring sistem hidrolik, mulai dari deteksi berbasis sensor hingga arsitektur diagnosis cerdas untuk meningkatkan keandalan dan efisiensi biaya [1]. Kegagalan tersebut umumnya disebabkan oleh leakage, wear, kavitasi, dan panas berlebih [2].

Pendekatan pemeliharaan tradisional seperti korektif dan preventif masih memiliki keterbatasan karena belum mempertimbangkan kondisi aktual sistem, sehingga berpotensi tidak tepat dalam penjadwalan perawatan [3]. Seiring perkembangan teknologi, machine learning memungkinkan penerapan predictive maintenance (PdM), yaitu pendekatan yang memprediksi kegagalan sebelum terjadi berdasarkan data historis dan sensor [4]. Machine Learning (ML) merupakan bagian dari kecerdasan buatan yang mampu mengelola dan memprediksi data dalam jumlah besar melalui algoritma pembelajaran [5]. Secara umum, ML terbagi menjadi supervised learning, unsupervised learning, dan reinforcement learning [6], serta terbukti meningkatkan akurasi prediksi pada sistem kompleks berbasis data [7]. Dalam konteks PdM, ML mampu menganalisis data sensor untuk memprediksi kegagalan, sehingga meningkatkan efisiensi operasional melalui deteksi dini [8].

Berbagai algoritma ML seperti *Support Vector Machine* (SVM), *Random Forest* (RF), dan Ar*tificial Neural Network* (ANN) telah digunakan untuk memodelkan degradasi sistem mekanis. Selain itu, metode *ensemble boosting* juga menunjukkan performa tinggi dalam menangani data besar dan hubungan non-linear, sehingga banyak digunakan dalam studi *predictive maintenance* [9].

Pendekatan *ensemble boosting* terbukti efektif dalam meningkatkan akurasi pada berbagai bidang, seperti *smart grid*, utilitas air [10] [11], serta sistem hidrolik dalam mendeteksi kondisi normal dan abnormal berbasis data sensor [12]. Studi lain juga menunjukkan peningkatan stabilitas prediksi melalui perbandingan RF, CatBoost, LightGBM, dan XGBoost, serta penerapan *deep learning* seperti *multiscale* 1D-CNN *dengan multi-attention* [1] [13].

Namun, penggunaan *Histogram Gradient Boosting* (HGB), yang unggul dalam efisiensi pelatihan, mampu menangani *missing values*, dan cocok untuk data skala besar [14] [15], masih terbatas. Selain itu, integrasi HGB dalam aplikasi interaktif berbasis Streamlit untuk prediksi kondisi *cooler*, *valve*, *pump leakage*, dan *hydraulic accumulator* juga masih jarang ditemukan, sehingga menjadi celah penelitian.

Oleh karena itu, studi ini berfokus pada pengembangan model *predictive maintenance* sistem hidrolik menggunakan algoritma HGB yang diintegrasikan dengan Streamlit untuk menghasilkan prediksi dan evaluasi kinerja secara interaktif.

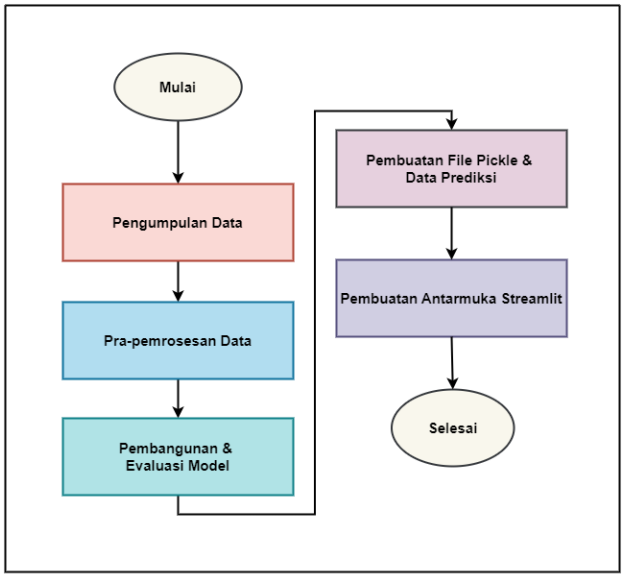
2. METODE PENELITIAN

Jenis penelitian yang diterapkan adalah penelitian kuantitatif eksperimen non-eksperimental dengan pendekatan prediktif. Karena data yang digunakan berupa data sekunder hasil pengukuran sensor pada sistem hidrolik yang diperoleh dari *dataset Condition Monitoring of Hydraulic Systems* di *platform* Kaggle. Penelitian tidak memberikan perlakuan langsung terhadap objek fisik, melainkan melakukan eksperimen komputasional dengan membangun dan mengevaluasi model klasifikasi berbasis algoritma *Histogram Gradient Boosting* (HGB) untuk memprediksi kondisi komponen hidrolik berdasarkan data sensor.

Berdasarkan sumbernya, data yang digunakan dalam penelitian ini berupa data sekunder. Data skunder adalah data yang telah dikumpulkan dan disusun oleh pihak lain, dalam hal ini dataset yang diperoleh dari situs kaggle "https://www.kaggle.com/code/mayank1897/predictive-maintenance-of-hydraulic-system". *Dataset* tersebut tersedia secara publik dan dalam penelitian ini hanya dimanfaatkan, diolah, dan dimodelkan untuk kepentingan pengembangan sistem prediktif pemeliharaan, tanpa dilakukan proses pengukuran sensor secara langsung. Tahapan penelitian merupakan langkah-langkah sistematis yang dilakukan untuk memastikan bahwa penelitian dilakukan secara terstruktur seperti yang dilihat pada Gambar 1.

Pengumpulan data merupakan langkah awal yang penting dalam penelitian ini. Tahap ini bertujuan memastikan bahwa data yang digunakan benar-benar merepresentasikan kondisi kerja sistem hidrolik pada

dunia industri sehingga model yang dibangun dapat memprediksi kebutuhan pemeliharaan secara andal. Setelah data dikumpulkan, langkah selanjutnya adalah melakukan pra-pemrosesan data. Tahap ini bertujuan untuk membersihkan data, memilih fitur yang relevan, menyeragamkan skala fitur, serta membagi data menjadi data latih dan data uji. Pada tahap ini dibangun model klasifikasi berbasis *algoritma Histogram Gradient Boosting* (HGB) untuk memprediksi kondisi empat komponen utama sistem hidrolik, yaitu *Cooler, Valve, Internal Pump Leakage,* dan *Hydraulic Accumulator*.



Gambar 1. Proses Metodologi

Setelah model *Histogram Gradient Boosting* untuk masing-masing komponen terbentuk, langkah berikutnya adalah mengevaluasi kinerjanya pada data uji yang tidak digunakan saat pelatihan. Setelah model *Histogram Gradient Boosting* untuk masing-masing komponen selesai dilatih dan dievaluasi, langkah berikutnya adalah menyimpan model dalam bentuk berkas sehingga dapat dimuat kembali pada tahap implementasi aplikasi tanpa perlu melatih ulang. Pada tahap ini, model *Histogram Gradient Boosting* yang telah disimpan dalam bentuk berkas .pkl diintegrasikan ke dalam sebuah antarmuka web menggunakan *framework* Streamlit.

3. HASIL DAN ANALISIS

3.1 Persiapan Data

Penelitian ini menggunakan *dataset Condition Monitoring of Hydraulic Systems* dari *platform* Kaggle yang terdiri dari 2205 sampel dengan 18 fitur sensor, meliputi PS1–PS6 (tekanan), EPS1, FS1–FS2 (debit), TS1–TS4 (suhu), VS1 (getaran), serta parameter CE, CP, SE, dan *Stable_flag*. Penelitian ini juga memiliki empat label prediksi, yaitu *Cooler_condition*, *Valve_condition*, *Internal_pump_leakage*, dan *Hydraulic_accumulator/bar*.

	PS1	PS2	PS3	PS4	PS5	PS6	EPS1	FS1	FS2	TS1	TS2	TS3	TS4	VS1	CE	CP	SE	Stable_flag
0	9.73	9.73	9.73	9.73	9.73	9.73	9.73	9.73	9.73	9.73	9.73	9.73	9.73	9.73	9.73	9.73	9.73	0
1	9.53	9.53	9.53	9.53	9.53	9.53	9.53	9.53	9.53	9.53	9.53	9.53	9.53	9.53	9.53	9.53	9.53	0
2	9.43	9.43	9.43	9.43	9.43	9.43	9.43	9.43	9.43	9.43	9.43	9.43	9.43	9.43	9.43	9.43	9.43	0
3	9.34	9.34	9.34	9.34	9.34	9.34	9.34	9.34	9.34	9.34	9.34	9.34	9.34	9.34	9.34	9.34	9.34	0
4	9.26	9.26	9.26	9.26	9.26	9.26	9.26	9.26	9.26	9.26	9.26	9.26	9.26	9.26	9.26	9.26	9.26	0
5	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	0
6	9.14	9.14	9.14	9.14	9.14	9.14	9.14	9.14	9.14	9.14	9.14	9.14	9.14	9.14	9.14	9.14	9.14	0
7	9.11	9.11	9.11	9.11	9.11	9.11	9.11	9.11	9.11	9.11	9.11	9.11	9.11	9.11	9.11	9.11	9.11	0
8	9.06	9.06	9.06	9.06	9.06	9.06	9.06	9.06	9.06	9.06	9.06	9.06	9.06	9.06	9.06	9.06	9.06	0
9	9.02	9.02	9.02	9.02	9.02	9.02	9.02	9.02	9.02	9.02	9.02	9.02	9.02	9.02	9.02	9.02	9.02	0

Gambar 2. Dataset Sensor Sistem Hidrolik

Tahap persiapan data mencakup penggabungan data sensor dari beberapa file, pemeriksaan struktur data, pembersihan data, serta standarisasi fitur agar berada pada skala yang seragam. Data yang telah siap kemudian digunakan dalam proses pemodelan menggunakan algoritma *Histogram Gradient Boosting* (HGB).

3.2 Memuat Data

Tahap ini memuat *dataset Condition Monitoring of Hydraulic Systems* dari Kaggle ke lingkungan Google Colab. *Dataset* terdiri dari beberapa file *.txt* berisi data sensor (PS1–PS6, FS1–FS2, TS1–TS4, VS1, CE, CP, SE, EPS1) serta *profile.txt* sebagai label kondisi komponen. Data sensor yang berbentuk matriks kemudian dibaca dan disusun ulang menjadi format tabular.

```
BASE_DIR = "/content/data"   # folder yang berisi PS1.txt, PS2.txt,

files = ["PS1","PS2","PS3","PS4","PS5","PS6"]
temp = []

for nm in files:
    fp = os.path.join(BASE_DIR, f"{nm}.txt")
    arr = np.genfromtxt(fp)
    temp.append(arr)

len(temp), temp[0].shape
```

```
(6, (2205, 6000))
```

Gambar 3. Kode Memuat Berkas Sensor

Selanjutnya, setiap data sensor diringkas menjadi satu nilai representatif per siklus menggunakan nilai rata-rata (*mean*). Lalu seluruh fitur sensor yang telah diperoleh digabungkan menjadi satu *dataset* menggunakan fungsi *concat* dengan parameter *axis* =1 sehingga terbentuk *dataset* fitur sensor (*sensor_df*). Sementara itu, *dataset* label (*factors_df*) dibentuk dari empat target utama yaitu *Cooler_condition*, *Valve_condition*, *Internal_pump_leakage*, dan *Hydraulic_accumulator/bar*, serta ditambahkan variabel *Stable_flag* sebagai penanda kondisi sistem stabil.

```
sensor_df = pd.concat([df1, df2, df3, df4, df5, df6, df7, df8], axis=1)
sensor_df.head(10)
```

Gambar 4. Kode Membentuk *Dataset* Fitur Sensor

Kedua *dataset* kemudian digabungkan menggunakan fungsi *concat()* dari *library* pandas sehingga menghasilkan *dataset* akhir (*df*). *Dataset* ini memiliki 2205 baris dan 22 kolom tanpa *missing value*, sehingga siap digunakan pada tahap pra-pemrosesan, pelatihan model, dan evaluasi.

	PS1	PS2	PS3	PS4	PS5	PS6	EPS1	FS1	FS2	TS1	TS2	TS3	TS4	VS1	CE	CP	SE
0	9.73	9.73	9.73	9.73	9.73	9.73	9.73	9.73	9.73	9.73	9.73	9.73	9.73	9.73	9.73	9.73	9.73
1	9.53	9.53	9.53	9.53	9.53	9.53	9.53	9.53	9.53	9.53	9.53	9.53	9.53	9.53	9.53	9.53	9.53
2	9.43	9.43	9.43	9.43	9.43	9.43	9.43	9.43	9.43	9.43	9.43	9.43	9.43	9.43	9.43	9.43	9.43
3	9.34	9.34	9.34	9.34	9.34	9.34	9.34	9.34	9.34	9.34	9.34	9.34	9.34	9.34	9.34	9.34	9.34
4	9.26	9.26	9.26	9.26	9.26	9.26	9.26	9.26	9.26	9.26	9.26	9.26	9.26	9.26	9.26	9.26	9.26
5	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21	9.21
6	9.14	9.14	9.14	9.14	9.14	9.14	9.14	9.14	9.14	9.14	9.14	9.14	9.14	9.14	9.14	9.14	9.14
7	9.11	9.11	9.11	9.11	9.11	9.11	9.11	9.11	9.11	9.11	9.11	9.11	9.11	9.11	9.11	9.11	9.11
8	9.06	9.06	9.06	9.06	9.06	9.06	9.06	9.06	9.06	9.06	9.06	9.06	9.06	9.06	9.06	9.06	9.06
9	9.02	9.02	9.02	9.02	9.02	9.02	9.02	9.02	9.02	9.02	9.02	9.02	9.02	9.02	9.02	9.02	9.02

Gambar 5. Hasil *Dataset* Fitur Sensor

3.3 Pra-pemrosesan Data

Pra-pemrosesan data dilakukan setelah pembentukan dataset akhir (*df*) untuk menyiapkan data sebelum pelatihan model. Tahap ini mencakup analisis sebaran fitur sensor, standarisasi fitur, penyesuaian tipe data label, serta pembagian data latih dan uji. Proses ini bertujuan memastikan struktur data konsisten, mudah diproses, dan mendukung kestabilan pelatihan model.

3.3.1 Analisis awal sebaran fitur sensor

Analisis awal sebaran fitur sensor dilakukan menggunakan visualisasi *histogram* (*sns.histplot*) dan *boxplot* (*sns.boxplot*) untuk mengidentifikasi distribusi data, rentang nilai, dan indikasi *outlier*. Fungsi *subplots()* digunakan untuk mengatur tampilan beberapa grafik dalam satu *figure*. Hasil analisis ini menjadi dasar penentuan kebutuhan standarisasi fitur sebelum tahap pemodelan.

3.3.2 Standarisasi fitur sensor (*StandardScaler*)

Standarisasi fitur sensor (StandardScaler) dilakukan untuk menyamakan skala antar fitur yang memiliki rentang nilai berbeda. Proses ini menggunakan metode *Z-score* melalui *StandardScaler* dari *scikit-learn*, sehingga setiap fitur memiliki rata-rata 0 dan standar deviasi 1 sebelum masuk ke tahap pemodelan. Secara matematis, proses standarisasi dinyatakan sebagai berikut:

$$z = \frac{x - \mu}{\sigma}$$

dengan *x* sebagai nilai asli, μ sebagai rata-rata, dan σ sebagai standar deviasi. Transformasi ini menghasilkan data yang telah dinormalisasi terhadap distribusinya. Implementasi dilakukan menggunakan fungsi *fit_transform()*, di mana *fit* menghitung parameter (μ dan σ), sedangkan *transform* menerapkan perubahan pada data. Pada penelitian ini, standarisasi diterapkan pada fitur PS1–PS6, EPS1, FS1–FS2, TS1–TS4, VS1, CE, CP, dan SE. Parameter scaler kemudian disimpan dalam berkas *scaler.pkl* untuk memastikan konsistensi transformasi pada data baru.

3.3.3 Penyesuaian tipe data label dan *Stable_flag*

Penyesuaian tipe data label dan Stable_flag dilakukan setelah standarisasi untuk memastikan kolom label diperlakukan sebagai kategori dalam proses klasifikasi serta menjaga konsistensi format data. Kolom yang dikonversi meliputi *Cooler_condition*, *Valve_condition*, *Internal_pump_leakage*, *Hydraulic_accumulator/bar*, dan *Stable_flag*.

3.3.4 Pembagian *Dataset*

Pembagian dataset dilakukan menjadi data latih (70%) dan data uji (30%) dengan *random_state* 42 untuk menjaga konsistensi. Pemodelan bersifat *multi-target* dengan model terpisah untuk setiap target, sehingga pembagian data dilakukan per target menggunakan *stratified split* agar distribusi kelas tetap seimbang. Untuk mengatasi keterbatasan data pada kelas tertentu, digunakan mekanisme *collapse_rare* saat stratifikasi tanpa mengubah label asli pada pelatihan. Khusus target *Hydraulic_accumulator/bar*, digunakan *safe_split*, yaitu stratifikasi hanya diterapkan jika jumlah sampel tiap kelas mencukupi; jika tidak, pembagian dilakukan tanpa stratifikasi.

3.4 Pelatihan Model dan Evaluasi

Pelatihan model dan evaluasi dilakukan menggunakan algoritma *Histogram Gradient Boosting (HGB)* untuk membangun sistem *predictive maintenance*. Pendekatan yang digunakan adalah *multi-target* dengan empat model klasifikasi terpisah untuk masing-masing target: *Cooler_condition*, *Valve_condition*, *Internal_pump_leakage*, dan *Hydraulic_accumulator/bar*, sehingga pelatihan dan evaluasi dapat dilakukan lebih spesifik serta memudahkan implementasi ke aplikasi *streamlit*. Sebelum pelatihan, label pada setiap target di-*encoding* ke format numerik (0..K−1) menggunakan fungsi *to_zero_based*. Pemetaan label (*mapping*) kemudian disimpan bersama model agar hasil prediksi dapat dikonversi kembali ke label asli saat ditampilkan pada aplikasi.

3.4.1 Pelatihan Model *Histogram Gradient Boosting*

Pelatihan Model Histogram *Gradient Boosting* dilakukan menggunakan data latih dengan pendekatan *gradient boosting* yang membangun model secara iteratif. Secara matematis, pembaruan model dinyatakan sebagai berikut:

$$F_m(x) = F_{m-1}(x) + v.h_m(x)$$

di mana *F_m(x)* adalah model pada iterasi ke-*m*, *F_{m-1}(x)* model sebelumnya, *v* (*learning rate*) mengontrol kontribusi pembaruan, dan *h_m(x)* merupakan *weak learner*. Parameter *max_iter* menentukan jumlah iterasi pembentukan model. Pada penelitian ini, digunakan dua skenario konfigurasi:

- *Cooler_condition*, *Valve_condition*, dan *Internal_pump_leakage*: *max_iter* = 400, *learning_rate* = 0.08, *random_state* = 42.
- *Hydraulic_accumulator/bar*: *max_iter* = 650, *learning_rate* = 0.06, *max_depth* = None, serta penambahan *sample_weight* berdasarkan frekuensi kelas untuk menangani ketidakseimbangan data.

3.4.2 Evaluasi

Evaluasi dilakukan menggunakan data uji untuk menilai performa model HGB pada setiap target. Metrik yang digunakan meliputi *Accuracy* untuk mengukur proporsi prediksi benar secara keseluruhan dan *F1-score macro* untuk memberikan penilaian yang seimbang antar kelas. Selain itu, *confusion matrix* digunakan untuk menganalisis pola kesalahan prediksi pada tiap kelas.

3.4.3 Hasil Evaluasi

Hasil evaluasi menunjukkan ringkasan performa model pada setiap target berdasarkan data uji, sebagaimana disajikan pada Tabel 1. Penilaian dilakukan menggunakan metrik *Accuracy* dan *F1-score macro* yang dihitung melalui fungsi *classification_report* dari *scikit-learn*.

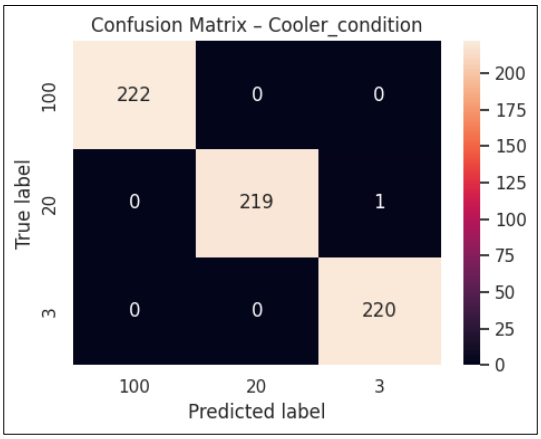
Tabel 1. Ringkasan Hasil Evaluasi Model HGB

Target	*Accuracy*	*F1-Score* macro
Cooler_condition	0.998	0.998
Valve_condition	0.834	0.786
Internal_pump_leakage	0.968	0.963
Hydraulic_accumulator/bar	0.989	0.987

Secara umum, model *HistGradientBoostingClassifier* menunjukkan performa sangat baik pada sebagian besar target. *Cooler_condition* dan *Hydraulic_accumulator/bar* memiliki nilai *Accuracy* dan *F1-score macro* yang sangat tinggi, diikuti oleh *Internal_pump_leakage* yang juga menunjukkan performa baik. Namun, *Valve_condition* memiliki performa relatif lebih rendah, khususnya pada *F1-score macro*, yang mengindikasikan adanya ketidakseimbangan performa antar kelas, kemungkinan akibat kemiripan karakteristik atau kompleksitas distribusi data. Analisis lebih lanjut dilakukan menggunakan *confusion matrix* untuk mengidentifikasi pola kesalahan prediksi pada tiap target.

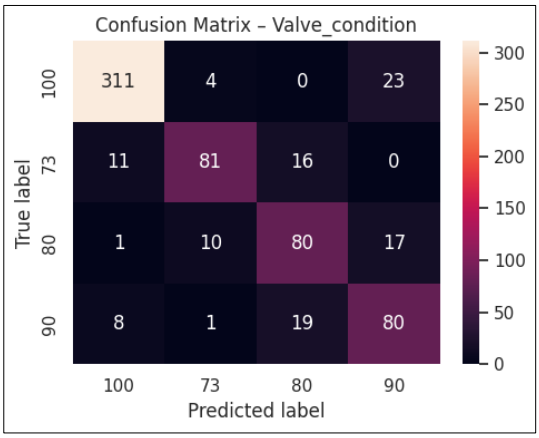
1. Hasil Evaluasi *Cooler_condition*

   Model menunjukkan performa sangat tinggi dengan nilai *Accuracy* dan *F1-score macro* sebesar 0.998. Berdasarkan *confusion matrix*, mayoritas prediksi berada pada diagonal yang menandakan klasifikasi sangat akurat. Kesalahan prediksi sangat minimal dan tidak signifikan, menunjukkan bahwa pola antar kelas pada target ini jelas sehingga mudah dipelajari oleh model.



Gambar 6. *Confusion Matrix Cooler_condition*

2. Hasil Evaluasi *Valve_condition*

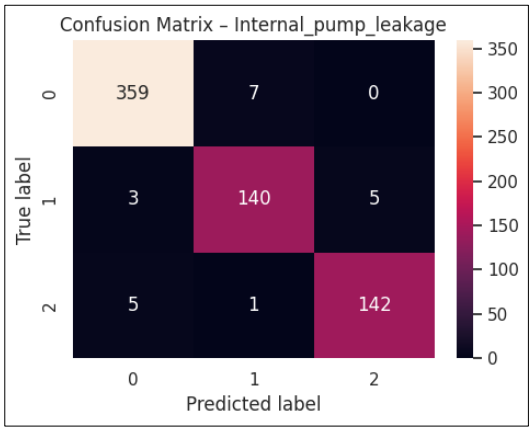


Gambar 7. *Confusion Matrix Valve_condition*

Target *Valve_condition* memiliki performa terendah dengan *Accuracy* sebesar 0.834 dan *F1-score macro* sebesar 0.786. Berdasarkan *confusion matrix*, terlihat adanya pertukaran prediksi antar beberapa kelas, yang menunjukkan model kesulitan membedakan kondisi valve dengan karakteristik sensor yang mirip. Nilai *F1-score macro* yang lebih rendah dari *Accuracy* mengindikasikan performa yang tidak merata pada tiap kelas. Hal ini menunjukkan bahwa kompleksitas dan kemiripan distribusi data pada target ini lebih tinggi, sehingga mempengaruhi kemampuan model dalam melakukan klasifikasi secara optimal.

3. Hasil Evaluasi *Internal_pump_leakage*

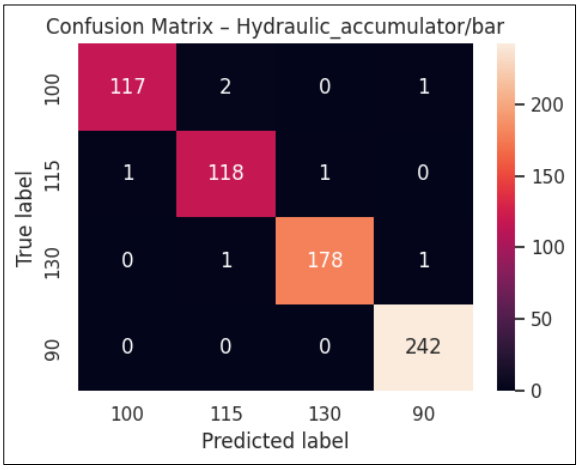
Model menunjukkan performa baik dengan *Accuracy* sebesar 0.968 dan *F1-score macro* sebesar 0.963. Berdasarkan *confusion matrix*, sebagian besar prediksi berada pada diagonal, menandakan klasifikasi yang akurat. Kesalahan prediksi relatif kecil dan hanya terjadi pada sebagian data, menunjukkan kemampuan model yang stabil dalam membedakan kondisi kebocoran. Meskipun masih terdapat sedikit kemiripan antar kelas, hal tersebut tidak berdampak signifikan terhadap kinerja model secara keseluruhan.



Gambar 8. *Confusion Matrix Internal_pump_leakage*

4. Hasil Evaluasi *Hydraulic_accumulator/bar*

Model menunjukkan performa sangat baik dengan *Accuracy* sebesar 0.989 dan *F1-score macro* sebesar 0.987. Berdasarkan *confusion matrix*, mayoritas prediksi berada pada diagonal, yang menandakan tingkat klasifikasi yang tinggi dengan kesalahan yang sangat minimal. Hasil ini menunjukkan bahwa model mampu membedakan setiap kelas dengan sangat baik. Selain itu, penggunaan konfigurasi khusus, termasuk penyesuaian parameter dan *sample_weight*, terbukti efektif dalam meningkatkan performa pada data dengan distribusi yang lebih kompleks atau tidak seimbang.



Gambar 9. *Confusion Matrix Hydraulic_accumulator/bar*

3.4.4 Pembuatan dan penyimpanan file *.pkl*

Pembuatan dan penyimpanan file *.pkl* dilakukan untuk menyimpan model HGB yang telah dilatih agar dapat digunakan kembali tanpa pelatihan ulang saat aplikasi *Streamlit* dijalankan. Pada penelitian ini dihasilkan empat model, yaitu *cooler_model.pkl*, *valve_model.pkl*, *pump_model.pkl*, dan *accumulator_model.pkl*. Selain itu, digunakan file input *data_prediksi.csv* yang berisi fitur sensor sebagai masukan prediksi. File *.pkl* menjadi artefak utama yang dimuat oleh aplikasi dalam proses prediksi.

3.5 Implementasi Streamlit

Pada tahap ini dilakukan implementasi model *machine learning* yang telah dilatih ke dalam aplikasi berbasis web menggunakan *framework* Streamlit. Streamlit dipilih karena mampu membangun antarmuka pengguna yang sederhana dan interaktif, sehingga proses prediksi dapat dilakukan dengan mudah tanpa harus menjalankan *notebook* secara manual. Aplikasi yang dibangun pada penelitian ini memungkinkan pengguna untuk melakukan prediksi kondisi komponen sistem hidrolik dengan cara mengunggah file model dan data fitur, kemudian sistem akan menampilkan hasil prediksi beserta interpretasi yang mudah dibaca.

Aplikasi Streamlit terdiri dari dua bagian utama, yaitu *sidebar* untuk mengunggah file model (*.pkl*) dan data fitur (CSV), serta halaman utama untuk menampilkan pratinjau data, menjalankan prediksi, menampilkan hasil dan interpretasi, serta menyediakan unduhan hasil dalam format CSV.

1. Unggah model .pkl dan data fitur (CSV)

Pengguna mengunggah empat file model (*.pkl*) melalui sidebar yang kemudian dimuat menggunakan *joblib.load()*. Selain itu, pengguna juga mengunggah file CSV berisi data fitur sensor. Setelah data berhasil dibaca, aplikasi menampilkan pratinjau lima baris pertama untuk memastikan kesesuaian struktur data. Proses unggah file diimplementasikan menggunakan komponen *file_uploader*, sebagaimana ditunjukkan pada Gambar 10.

```
# ====================== Sidebar: upload artefak ======================
st.sidebar.header("1) Upload Artefak Model (.pkl)")
st.sidebar.write("Gunakan file .pkl yang disimpan dengan **joblib.dump(...)** dari notebook.")

up_X1 = st.sidebar.file_uploader("Upload cooler_model.pkl",      type=["pkl"])
up_X2 = st.sidebar.file_uploader("Upload valve_model.pkl",       type=["pkl"])
up_X3 = st.sidebar.file_uploader("Upload pump_model.pkl",        type=["pkl"])
up_X4 = st.sidebar.file_uploader("Upload accumulator_model.pkl", type=["pkl"])

st.sidebar.divider()
st.sidebar.header("2) Upload Data Fitur (CSV)")
up_csv = st.sidebar.file_uploader(
    "Pilih file CSV (kolom & nama sama seperti saat training / prediksi.csv)",
    type=["csv"],
```

Gambar 10. Kode Unggah Model dan Data Fitur

2. Penyelarasan kolom fitur sebelum prediksi (*align columns*)

Agar prediksi konsisten, kolom fitur pada data CSV harus sesuai dengan fitur saat pelatihan model. Aplikasi menerapkan fungsi *align columns* dengan memanfaatkan metadata *feature_names_in_* untuk menyesuaikan struktur data, yaitu dengan menambahkan kolom yang hilang (bernilai default), mengabaikan kolom yang tidak digunakan, serta mengurutkan kolom sesuai urutan pelatihan. Implementasi fungsi ini ditunjukkan pada Gambar 11.

```
# ====================== Utilitas ======================
def align_columns(df: pd.DataFrame, model) -> Tuple[pd.DataFrame, Optional[str]]:
    """
    Samakan urutan kolom dengan model.
    Jika ada fitur hilang, otomatis dibuat dan diisi nilai default (fill_missing_default).
    """
    try:
        # Cek metadata kustom dulu, lalu fallback ke sklearn
        feat = getattr(model, "_feature_names", None)
        if feat is None:
            feat = getattr(model, "feature_names_in_", None)
        if feat is None:
            return df, None

        feat = list(feat)
        missing = [c for c in feat if c not in df.columns]
        extra = [c for c in df.columns if c not in feat]

        warn_parts = []
        if missing:
            for c in missing:
                df[c] = fill_missing_default
            warn_parts.append(
                f"kolom hilang: {missing} -> dibuat & diisi {fill_missing_default}"
            )
        if extra:
            warn_parts.append(f"kolom ekstra: {extra} (diabaikan)")
```

```
        warn = (
            "Tidak semua kolom cocok dengan model (" + "; ".join(warn_parts) + ")."
            if warn_parts
            else None
        )
        df2 = df[[c for c in feat]]  # urut sesuai training
        return df2, warn
    except Exception as e:
        return df, f"Gagal menyamakan kolom: {e}"
```

Gambar 11. Kode Penyelarasan Kolom Fitur

3. Prediksi dan pengembalian label asli (*decode label*)

Setelah data siap, aplikasi melakukan prediksi menggunakan *model.predict()*. Hasil prediksi berupa indeks kelas kemudian dikonversi kembali ke label asli (misalnya 100/20/3 pada *cooler*) menggunakan *label_mapping_* yang disimpan saat pelatihan. Implementasi proses prediksi dan *decode label* ditunjukkan pada Gambar 12.

```
# Prediksi index kelas
yhat_idx = model.predict(X_use.values)

# Decode ke label asli (100, 20, 3, 90, 80, 73, 130, dst.)
mapping = getattr(model, "label_mapping_", None)
if mapping is not None:
    inv_map = {v: k for k, v in mapping.items()}  # index -> label asli
    yhat = np.array([inv_map.get(int(v), v) for v in yhat_idx])
else:
    yhat = yhat_idx

y_ser = pd.Series(yhat, index=features.index, name=f"{name}_pred")
```

Gambar 12. Kode Prediksi dan Decode Label

4. Interpretasi hasil prediksi

Aplikasi tidak hanya menampilkan hasil prediksi dalam bentuk angka, tetapi juga menyediakan interpretasi berbasis aturan untuk memudahkan pemahaman. Sebagai contoh pada komponen *cooler*: 100 menunjukkan

full efficiency, 20 *reduced efficiency*, dan 3 *close to total failure*. Implementasi interpretasi hasil ditunjukkan pada Gambar 13.

```
def msg_cooler(v: float) -> str:
    if v == 100:
        return "✅ Cooler: Full efficiency. Tidak diperlukan tindakan pemeliharaan."
    if v == 20:
        return "🟠 Cooler: Reduced efficiency. Efisiensi pendinginan menurun, perlu dijadwalkan pemeliharaan."
    if v == 3:
        return "⚠️ Cooler: Close to total failure. Pendingin mendekati kegagalan total, perlu tindakan segera."
    return "⚠️ Cooler: Label kondisi tidak dikenali."
```

Gambar 13. Kode Interpretasi Hasil Prediksi

5. Unduh hasil prediksi

Aplikasi menyediakan fitur unduh hasil dalam format CSV yang mencakup data *input*, hasil prediksi, dan interpretasi. Fitur ini diimplementasikan menggunakan komponen *download_button*, sebagaimana ditunjukkan pada Gambar 14.

```
# Unduh gabungan input + prediksi + interpretasi
buf = io.StringIO()
pd.concat([X, out, out_text], axis=1).to_csv(buf, index=False)
st.download_button(
    "⬇️ Unduh Hasil (CSV)",
    buf.getvalue(),
    "hasil_prediksi.csv",
    "text/csv",
)
```

Gambar 14. Kode Unduh Hasil Prediksi

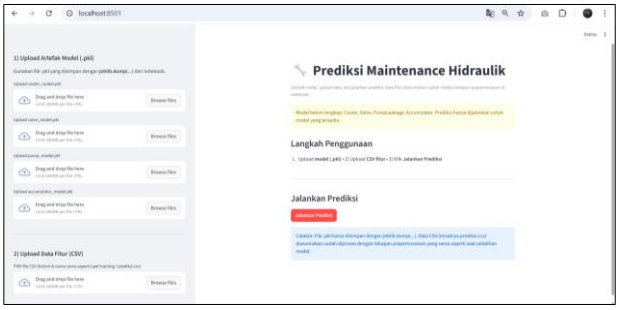
Setelah file *App.py* dibuat, aplikasi dijalankan melalui terminal atau *command prompt* menggunakan perintah berikut:

```
PS D:\SEMPRO PERAWATAN HYDROLIK> Streamlit run App.py

  You can now view your Streamlit app in your browser.
```

Gambar 15. Kode Perintah Menjalankan Streamlit

Jika aplikasi berhasil dijalankan, sistem akan menampilkan notifikasi bahwa aplikasi dapat diakses melalui *browser* dan antarmuka Streamlit akan terbuka. Pengguna kemudian dapat mengunggah model *.pkl* dan file CSV fitur, lalu menekan tombol Jalankan Prediksi untuk memperoleh hasil prediksi kondisi komponen, sebagaimana ditunjukkan pada Gambar 16.



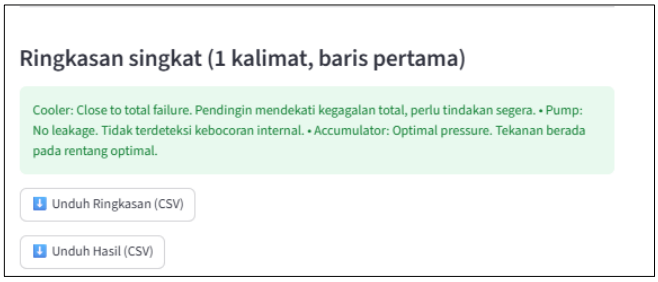
Gambar 16. Antarmuka Streamlit

3.5.1 Hasil Prediksi pada Aplikasi Streamlit

Setelah pengguna mengunggah file model *.pkl* dan data fitur (CSV) serta menekan tombol Jal*ankan Prediksi*, aplikasi memproses data dengan menyelaraskan kolom fitur sesuai model, kemudian melakukan prediksi untuk komponen *Cooler*, *Valve*, *Pump Leakage*, dan *Accumulator*. Hasil prediksi disajikan dalam dua bentuk, yaitu:

1. *Output* numerik berupa label asli masing-masing komponen, seperti *Cooler* (100/20/3), *Valve* (100/90/80/73), *Pump* (0/1/2), dan *Accumulator* (130/115/100/90). Label ini diperoleh melalui proses *decode* menggunakan *label_mapping_* yang disimpan saat pelatihan.
2. Interpretasi hasil dalam bentuk teks, seperti *Full efficiency*, *Reduced efficiency*, dan *Close to total failure* untuk memudahkan pemahaman kondisi komponen.

Selain ditampilkan pada antarmuka, hasil prediksi juga dapat diunduh dalam format CSV melalui tombol *Unduh Hasil*, sehingga dapat digunakan sebagai dokumentasi atau analisis lanjutan.



Gambar 17. Tombol Unduh Hasil Prediksi

Sebagai contoh, ketika model memprediksi *Cooler_condition* = 20, sistem memberikan interpretasi *Reduced efficiency* yang menunjukkan penurunan kinerja dan kebutuhan penjadwalan pemeliharaan. Sedangkan jika diperoleh *Cooler_condition* = 3, interpretasi yang ditampilkan adalah *Close to total failure* yang mengindikasikan kondisi kritis dan memerlukan tindakan segera. Pola interpretasi yang sama diterapkan pada komponen *Valve, Pump Leakage*, dan *Accumulator* sesuai label kondisi pada *dataset*.

Oleh karena itu, fitur hasil prediksi pada aplikasi Streamlit tidak hanya menampilkan *output* model, tetapi juga menyajikan interpretasi kondisi komponen serta menyediakan opsi penyimpanan hasil dalam format CSV.

3.6 Pembahasan Umum

Berdasarkan hasil evaluasi, model *Histogram Gradient Boosting* (HGB) menunjukkan performa sangat baik pada target *Cooler_condition* dan *Hydraulic_accumulator/bar* dengan nilai *accuracy* dan F1-*score macro* mendekati 1. Target *Internal_pump_leakage* juga menunjukkan performa tinggi dengan keseimbangan antar kelas yang baik. Namun, performa pada *Valve_condition* lebih rendah dibandingkan target lainnya, sehingga menunjukkan bahwa klasifikasi kondisi *valve* relatif lebih kompleks.

Penurunan performa pada *Valve_condition* diduga disebabkan oleh jumlah kelas yang lebih banyak serta kemiripan pola sinyal sensor antar tingkat degradasi, yang menyebabkan sebagian kelas saling tertukar pada *confusion matrix*. Kondisi ini berdampak pada penurunan nilai F1-*score macro* karena metrik tersebut memperlakukan setiap kelas secara setara.

Secara keseluruhan, hasil penelitian menunjukkan bahwa model HGB efektif digunakan untuk *predictive maintenance* pada sistem hidrolik. Implementasi melalui aplikasi Streamlit memungkinkan proses prediksi dilakukan secara praktis melalui unggahan data dan model terlatih, sehingga dapat mendukung pengambilan keputusan pemeliharaan. Pengembangan lebih lanjut dapat dilakukan melalui *tuning* parameter, penambahan fitur statistik, atau metode optimasi khusus untuk meningkatkan performa pada target *Valve_condition*.

4. KESIMPULAN

Penelitian ini menunjukkan bahwa model *Histogram Gradient Boosting* (HGB) mampu memprediksi kondisi empat komponen sistem hidrolik, yaitu *Cooler*, *Valve*, *Internal Pump Leakage*, dan *Hydraulic Accumulator*. Evaluasi menunjukkan performa sangat baik pada *Cooler_condition* (*Accuracy* 0,998; F1-*score* 0,998), *Hydraulic_accumulator/bar* (0,989; 0,987), dan *Internal_pump_leakage* (0,968; 0,963), sedangkan *Valve_condition* memiliki performa lebih rendah (0,834; 0,786).

Model HGB juga berhasil diimplementasikan ke dalam aplikasi Streamlit untuk menampilkan hasil prediksi dan interpretasi kondisi komponen secara interaktif, sehingga dapat digunakan sebagai prototipe sistem *predictive maintenance* berbasis data sensor.

REFERENSI

[1] J. Garcia, L. Rios-Colque, P. Alvaro, and L. Rojas, "Condition Monitoring and Predictive Maintenance in Industrial Equipment: An NLP-Assisted Review of Signal Processing, Hybrid Models, and Implementation Challenges," *Multidiscip. Digit. Publ. Inst.*, vol. 15, no. 10, 2025, doi: https://doi.org/10.3390/app15105465.

[2] J. Sun, H. Ding, N. Li, X. Sun, and X. Dong, "Intelligent Fault Diagnosis of Hydraulic System Based on Multiscale One-Dimensional Convolutional Neural Networks with Multiattention Mechanism," *Sensors*, vol. 24, no. 22, pp. 1–19, 2024, [Online]. Available: https://doi.org/10.3390/s24227267

[3] J. Azeta *et al.*, "Artificial Intelligence and Robotics in Predictive Maintenance: A Comprehensive Review," *Front. Mech. Eng.*, vol. 11, 2026, doi: https://doi.org/10.3389/fmech.2025.1722114.

[4] M. Achouch *et al.*, "On Predictive Maintenance in Industry 4.0: Overview, Models, and Challenges," *Appl. Sci.*, vol. 12, no. 16, 2022, doi: https://doi.org/10.3390/app12168081.

[5] E. Hermawan, S. D. Panjaitan, and E. F. Ripanti, "Sistem Prediksi Banjir Rob Kota Pontianak Berbasis Machine Learning Menggunakan Framework Streamlit," *J. Edukasi Penelit. Inform.*, vol. 10, no. 3, 2024, doi: https://doi.org/10.26418/jp.v10i3.79955.

[6] C. Tsallis, P. Papageorgas, D. Piromalis, and R. A. Munteanu, "Application-Wise Review of Machine Learning-Based Predictive Maintenance: Trends, Challenges, and Future Directions," *Appl. Sci.*, vol. 15, no. 9, 2025, doi: https://doi.org/10.3390/app15094898.

[7] M. Mahetaji and J. Brahma, "Enhancing Wellbore Stability Through Machine Learning for Sustainable Hydrocarbon Exploitation," *Sci. Rep.*, vol. 15, no. 1, 2025.

[8] S. Sethi, "Predictive Maintenance in IoT Environments Using Machine Learning: Opportunities, Challenges, and Innovations," *Int. J. Innov. Res. Eng. Multidisplinary Phys. Sci.*, vol. 11, no. 4, 2023, doi: https://doi.org/10.5281/zenodo.14945146.

[9] P. Mallioris, E. Aivazidou, and D. Bechtsis, "Predictive Maintenance in Industry 4.0: A Systematic Multi-Sector Mapping," *CIRP J. Manuf. Sci. Technol.*, vol. 50, pp. 80–103, 2024, doi: https://doi.org/10.1016/j.cirpj.2024.02.003.

[10] J. Wu *et al.*, "Fault Diagnosis of the HVDC System Based on the CatBoost Algorithm Using Knowledge Graphs," *Front. Energy Res.*, vol. 11, 2023, doi: https://doi.org/10.3389/fenrg.2023.1144785.

[11] S. M. Almazrouei, F. Dweiri, R. Aydin, and A. Alnaqbi, "An Ensemble Machine Learning Model for Predictive Maintenance on Water Injection Pumps in the Oil and Gas Industry," *Int. J. Adv. Comput. Sci. Appl.*, vol. 15, no. 11, 2024, doi: 10.14569/IJACSA.2024.0151141.

[12] D. Kim and T. Heo, "Anomaly Detection with Feature Extraction Based on Machine Learning Using Hydraulic System IoT Sensor Data," *Sensors*, vol. 22, no. 7, 2022, doi: https://doi.org/10.3390/s22072479.

[13] H. N. Noura, T. Chu, Z. Allal, O. Salman, and K. Chahine, "A Comparative Study of Ensemble Methods and Multi-Output Classifiers for Predictive Maintenance of Hydraulic Systems," *Results Eng.*, vol. 24, 2024, doi: https://doi.org/10.1016/j.rineng.2024.102900.

[14] N. L. Fitriyani *et al.*, "A Novel Approach Utilizing Bagging, Histogram Gradient Boosting, and Advanced Feature Selection for Predicting the Onset of Cardiovascular Diseases," *Mathematics*, vol. 13, no. 13, 2025, doi: https://doi.org/10.3390/math13132194.

[15] F. A. Ramirez-Rivera and F. Guerrero-Rodriguez, "Ensemble Learning Algorithms for Solar Radiation Prediction in Santo Domingo: Measurements and Evaluation," *Sustainability*, vol. 16, no. 18, 2024, doi: https://doi.org/10.3390/su16188015.