*Contents lists available at MID Publisher International*

**Technology Sciences Insights Journal**

Journal homepage: https://journal.midpublisher.com/index.php/tsij

# Penerapan Support Vector Machine untuk Deteksi Gangguan pada Sensor Berbasis Data Suhu (Sensor Fault Detection)

**Meisy Dwi Putri**
Politeknik Negeri Batam

**INFORMASI ARTIKEL**

**Sejarah Artikel:**
Diterima: Desember 2025
Revisi: Juli 2026
Dipublikasi: Juli 2026

**Kata Kunci:** Deteksi gangguan, sensor, SVM, kecerdasan buatan, klasifikasi

**Keywords:** *Fault detection, sensor, SVM, artificial intelligence, classification*

*Penulis Korespondensi:
meisydwiputri17@gmail.com

**ABSTRAK**

Deteksi gangguan pada sensor menjadi elemen krusial dalam sistem monitoring industri untuk mencegah kerusakan dan memastikan keandalan sistem. Penelitian ini bertujuan mengembangkan model klasifikasi gangguan sensor menggunakan algoritma *Support Vector Machine* (SVM) berbasis data pembacaan suhu. Data diklasifikasikan sebagai gangguan jika nilainya di luar ambang batas 10–30. Proses meliputi *preprocessing*, pelatihan, dan evaluasi model menggunakan *metrik confusion matrix, ROC-AUC, MSE, dan validasi silang*. Hasil menunjukkan akurasi sekitar 90% dan nilai AUC >0.90, meskipun *recall* untuk kelas gangguan masih rendah akibat ketidakseimbangan data. Model ini menunjukkan potensi tinggi untuk diimplementasikan dalam sistem deteksi gangguan sensor berbasis kecerdasan buatan.

***ABSTRACT***

*Sensor fault detection is critical in industrial monitoring systems to prevent failures and ensure system reliability. This study aims to develop a sensor fault classification model using the Support Vector Machine (SVM) algorithm based on temperature reading data. Data are labeled as faulty if the value falls outside the 10–30 range. The process includes preprocessing, training, and model evaluation using confusion matrix, ROC-AUC, MSE, and cross-validation metrics. The results show an accuracy of about 90% and an AUC >0.90, although recall for the fault class is still low due to data imbalance. The model demonstrates strong potential for implementation in AI-based sensor fault detection systems.*



## PENDAHULUAN

Perkembangan sistem monitoring sensor pada perangkat industri dan lingkungan cerdas menjadi semakin krusial seiring dengan meningkatnya kebutuhan akan sistem yang andal dan akurat dalam mendeteksi kesalahan atau gangguan (*fault detection*) [1]. Salah satu tantangan utama dalam sistem sensor adalah kemampuan untuk mendeteksi nilai-nilai anomali yang berpotensi menandakan kerusakan atau kegagalan fungsi sistem. Kesalahan yang tidak terdeteksi dapat menyebabkan kerugian besar, baik dari segi operasional maupun keselamatan.

Pendekatan tradisional dalam fault detection, seperti metode berbasis ambang batas atau statistik konvensional, umumnya memiliki keterbatasan dalam mengidentifikasi pola kompleks atau anomali yang tidak linear. Oleh karena itu, pendekatan berbasis kecerdasan

buatan (AI) menjadi alternatif yang lebih unggul karena kemampuannya dalam melakukan klasifikasi berbasis pola dan menangani data dalam jumlah besar. Beberapa penelitian menunjukkan bahwa algoritma seperti *Support Vector Machine* (SVM) berhasil meningkatkan akurasi dalam mengidentifikasi kondisi normal dan anomali sensor [2]. SVM dikenal efektif dalam menangani klasifikasi *biner* pada data berdimensi tinggi dengan *margin* maksimal, sedangkan KNN mampu melakukan prediksi berbasis kemiripan lokal antar data tanpa perlu pelatihan model yang kompleks.

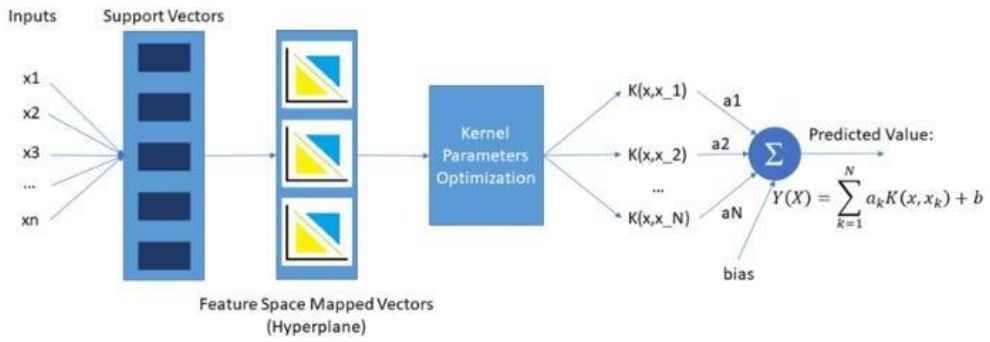
Penelitian 1 (satu) menunjukkan bahwa integrasi preprocessing data dan pemilihan fitur yang tepat sangat memengaruhi performa deteksi gangguan sensor, terutama dalam konteks *real-time monitoring* [3]. Di sisi lain, model-machine learning tetap menghadapi tantangan dalam menangani data yang tidak seimbang serta kebutuhan akan interpretasi hasil yang mudah dipahami oleh operator sistem [4]. Penelitian ini bertujuan untuk mengembangkan model klasifikasi fault sensor menggunakan algoritma SVM berdasarkan data aktual pembacaan sensor. Data dinyatakan sebagai gangguan jika nilai sensor berada di luar rentang ambang batas yang telah ditentukan. Selain membangun model klasifikasi, penelitian ini juga menganalisis performa model melalui evaluasi *metrik* seperti *confusion matrix, ROC-AUC, mean squared error (MSE), dan cross-validation*. Dengan pendekatan ini, diharapkan tercipta model yang akurat, efisien, dan siap diterapkan dalam sistem deteksi gangguan sensor berbasis AI.

## METODE PENELITIAN

Penelitian ini merupakan jenis penelitian terapan berbasis eksperimen kuantitatif yang bertujuan untuk mengembangkan dan mengevaluasi model klasifikasi gangguan sensor (sensor *fault*) menggunakan algoritma pembelajaran mesin, yaitu *Support Vector Machine* (SVM).

### *Support Vector Machine* (SVM)

*Support Vector Machine* (SVM) adalah salah satu algoritma pembelajaran mesin yang digunakan untuk menyelesaikan permasalahan klasifikasi maupun regresi. SVM bekerja dengan mencari *hyperplane* optimal yang memisahkan data dari dua kelas dengan *margin* maksimum, yaitu jarak terjauh antara titik data (*support vectors*) dan garis pemisah [6]. Model ini sangat efektif untuk data berdimensi tinggi dan dapat digunakan baik pada data yang terpisah secara linier maupun tidak linier dengan bantuan fungsi *kernel,* seperti *linear, RBF (Radial Basis Function), dan polinomial*. Untuk data yang tidak dapat dipisahkan secara linier, SVM menggunakan teknik kernel trick untuk memproyeksikan data ke dalam ruang berdimensi lebih tinggi, sehingga memungkinkan pemisahan yang lebih baik [7]. SVM juga dikenal memiliki performa yang baik dalam kasus dengan jumlah fitur yang besar, meskipun membutuhkan waktu komputasi yang lebih tinggi dan sensitif terhadap pemilihan parameter kernel serta regularisasi.




Gambar 1. Arsitektur SVM

Data yang digunakan merupakan data *time-series* pembacaan sensor suhu yang diperoleh dari sistem monitoring berbasis perangkat IoT. Dimana pengambilan dari https://www.kaggle.com/datasets yaitu https://www.kaggle.com/datasets/arashnic/sensor-fault-detection-data Data diolah dan dianalisis secara sistematis melalui tahapan-tahapan yang dijelaskan berikut ini.

1. Objek Penelitian

   Objek dari penelitian ini adalah nilai pembacaan sensor suhu yang diperoleh secara periodik. Setiap entri data memiliki tiga atribut: *Timestamp, SensorId, dan Value*. Data dengan nilai sensor yang berada di luar ambang batas normal (kurang dari 10 atau lebih dari 30) dianggap sebagai anomali (*fault*) dan diberi label 1, sedangkan nilai lainnya dianggap normal (label 0). Pendekatan *labeling* ini mengacu pada metode *threshold-based fault tagging* yang banyak digunakan dalam domain sensor analysis [5].

2. Teknik dan Instrumen Pengumpulan Data

   Data diperoleh dari log pembacaan sensor yang telah disimpan dalam format CSV (sensor-fault-detection.csv). Dataset mencerminkan kondisi operasional normal dan gangguan, yang menjadi dasar klasifikasi. Tidak dilakukan proses pengambilan data lapangan secara langsung karena data telah tersedia dalam bentuk historis digital.

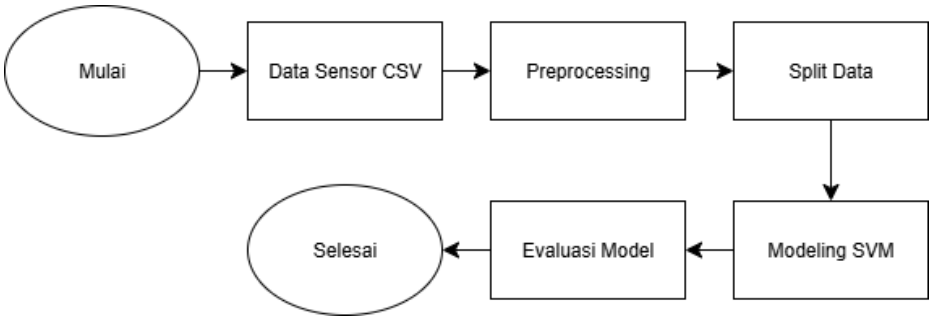
3. Desain dan Langkah Penelitian

Desain penelitian ini mengikuti alur eksperimen pembelajaran mesin yang umum digunakan, dimulai dari preprocessing data hingga evaluasi model klasifikasi. Prosesnya dijelaskan sebagai berikut:

Langkah-langkah penelitian:

1. *Preprocessing* Data: Meliputi pemisahan fitur *(SensorId, Value*) dan target (Label), pembagian dataset menjadi data latih dan uji (rasio 80:20), serta normalisasi fitur menggunakan *Standard Scaler* [5].
2. Pelatihan Model: Dua model pembelajaran mesin yaitu SVM dilatih menggunakan data latih. SVM digunakan dengan *kernel* [5].

3. Evaluasi Model: Model diuji menggunakan data uji, kemudian dievaluasi berdasarkan metrik *confusion matrix, ROC-AUC, mean squared error (MSE*), dan *5-fold cross-validation* untuk mengukur akurasi dan kemampuan generalisasi model [5].
4. Diagram Alur Penelitian




Gambar 2. Diagram alur penelitian

Penjelasan tiap langkah:

1. Data Sensor (CSV)
   Data historis pembacaan sensor suhu dikumpulkan dalam format .csv, berisi *Timestamp, SensorId, dan Value*. Data ini menjadi basis untuk klasifikasi kondisi normal atau fault.
2. *Preprocessing* Data
   - *Labeling*: Data diberi label otomatis berdasarkan ambang batas (misalnya, nilai <10 atau >30 dianggap *fault*).
   - Normalisasi: Fitur numerik seperti *SensorId dan Value* dinormalisasi menggunakan *Standard Scaler* agar semua fitur memiliki skala yang seragam. Ini penting untuk algoritma SVM.
3. Pembagian Dataset
   - Data dibagi menjadi dua bagian: 80% untuk pelatihan model (*training*) dan 20% untuk pengujian model (*testing*), guna mengukur generalisasi model.
4. *Modeling*
   - Algoritma AI digunakan:
   - SVM: Cocok untuk klasifikasi *biner*, bekerja optimal dengan data berdimensi tinggi.
5. Evaluasi Model
   - *Confusion Matrix*: Menunjukkan jumlah prediksi benar/salah per kelas.

   Rumus:

   **Akurasi:**

$$Akurasi = \frac{TP+TN}{TP+FP+FN+TN} \quad (1)$$

**Presisi (*Postive Predictive Value*):**

$$Presisi = \frac{TP}{TP+FP} \quad (2)$$

**Recall (*Sensitivity*/*True Positive Rate*):**

$$Recall = \frac{TP}{TP+FP} \quad (3)$$

**F1-Score:**

$$F-1\ score = \frac{2\ x\ (Presisi\ x\ Recall)}{TP+FP+FN+TN} (4)$$

Dimana:

TP: *True Positive*, FP: *False Positive*

- *ROC Curve & AUC*: Mengukur kemampuan model membedakan antara *fault* dan normal.

  Rumus:

**TP: *True Positive*:**

$$TPR = \frac{TP}{TP+FN} \quad (5)$$

**FP: *False Positive*:**

$$FPR = \frac{FP}{FP+TN} \quad (6)$$

Dimana:
True Positive Rate (TPR / Recall), False Positive Rate (FPR) TP: *True Positive*, FP: *False Positive*

**AUC (*Area Under Curve*):**
Luas di bawah kurva ROC, menunjukkan kemampuan model membedakan kelas:

- Nilai AUC = 1 → sempurna
- Nilai AUC = 0.5 → seperti tebak acak
-
-

**MSE**: Menghitung rata-rata kesalahan prediksi model. Rumus:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \quad (7)$$

Dimana:

$n$: jumlah total data uji

$y_i$: nilai aktual (label sebenarnya)

$\hat{y}_i$: nilai prediksi dari model

$(y_i - \hat{y}_i)^2$: kuadrat dari selisih antara nilai aktual dan prediksi

***Cross Validation***: Model diuji dengan validasi silang 5-fold untuk memastikan hasil stabil dan tidak *overfitting*. Rumus:

$$CV\ Accuracy = \frac{1}{k}\sum_{i=1}^{k} Accuracy_i \quad (8)$$

## IMPLEMENTASI PROGRAM

```
# import Library
import pandas as pd
import numpy as np
from sklearn.pipeline import make_pipeline
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import cross_val_score
```

**import pandas as pd**

Mengimpor pandas**.** Digunakan untuk memproses data tabular (seperti file .csv) dalam bentuk DataFrame.

**import numpy as np**

Mengimpor NumPy, pustaka untuk komputasi numerik.

**from sklearn.pipeline import make_pipeline**

Mengimpor fungsi make_pipeline dari Scikit-learn. Digunakan untuk membuat pipeline**,** yaitu alur proses berurutan seperti:

*normalisasi data → training model*

**from sklearn.svm import SVC**

Mengimpor model *Support Vector Classifier (SVC)* dari *Scikit-learn.*

Ini adalah algoritma klasifikasi berbasis margin maksimum yang sangat kuat, terutama untuk data dua kelas atau lebih.

**from sklearn.preprocessing import StandardScaler**

Mengimpor *StandardScaler*, digunakan untuk normalisasi fitur (rata-rata 0, standar deviasi 1).

**from sklearn.model_selection import cross_val_score**

Mengimpor fungsi untuk melakukan *cross-validation.*

```
# Load data dengan separator
df = pd.read_csv("sensor-fault-detection.csv", sep=';')

# Cek data
df.head()
```

Membaca data dari file CSV bernama sensor-fault-detection.csv

Dengan separator (pemisah kolom) berupa titik koma (;), bukan koma (,)

```
# Preprocessing Data
# Label: jika Value > 30 atau < 10 dianggap Fault (1), lainnya Normal (0)
df['Label'] = df['Value'].apply(lambda x: 1 if x > 30 or x < 10 else 0)

# Drop kolom Timestamp (tidak dibutuhkan), gunakan fitur numerik
X = df[['SensorId', 'Value']]
y = df['Label']

# Membagi Dataset menjadi Data Latih dan Uji
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

**df['Label'] = df['Value'].apply(lambda x: 1 if x > 30 or x < 10 else 0)**

Menentukan Label kelas

**X = df[['SensorId', 'Value']]**

**# Fitur y = df['Label'] #**

**Label (Target)**
Mengambil fitur dan label

**X_train, X_test, y_train, y_test =**

**train_test_split( X, y, test_size=0.2,**

**random_state=42**

)
Membagi data

```
# Normalisasi (Standarisasi) Data
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

**`X_train_scaled`** → data latih yang sudah dinormalisasi
**`X_test_scaled`** → data uji yang dinormalisasi (menggunakan *fit* dari data latih)

```
# Training SVM
from sklearn.svm import SVC

svm_model = SVC(kernel='linear', probability=True, random_state=42)
svm_model.fit(X_train_scaled, y_train)
```

**SVC(kernel='linear')**: Menggunakan kernel linear → cocok untuk data yang bisa dipisahkan garis lurus.
**probability=True**: Mengaktifkan prediksi probabilitas kelas. **random_state=42**: Agar hasilnya konsisten saat dijalankan ulang. **fit()**: Melatih model dengan data latih (X_train_scaled, y_train).

```
# Evaluasi Model
from sklearn.metrics import confusion_matrix, classification_report,
roc_auc_score, roc_curve, mean_squared_error
import matplotlib.pyplot as plt
import seaborn as sns

# Prediksi
y_pred = svm_model.predict(X_test_scaled)
y_prob = svm_model.predict_proba(X_test_scaled)[:, 1]

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

# ROC Curve
fpr, tpr, _ = roc_curve(y_test, y_prob)
plt.plot(fpr, tpr, label=f"AUC = {roc_auc_score(y_test, y_prob):.2f}")
```

```
plt.plot([0, 1], [0, 1], 'k--')
plt.title('ROC Curve')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.show()

# Other Metrics
print(classification_report(y_test, y_pred))
print("MSE:", mean_squared_error(y_test, y_pred))
```

**from sklearn.metrics import confusion_matrix, classification_report, roc_auc_score, roc_curve, mean_squared_error**
**import matplotlib.pyplot as plt**
**import seaborn as sns**
Untuk menghitung metrik evaluasi seperti: akurasi, precision, recall, AUC, MSE, dan membuat visualisasi.

**y_pred = svm_model.predict(X_test_scaled) # Prediksi label**
**y_prob = svm_model.predict_proba(X_test_scaled)[:, 1] # Probabilitas kelas 1 (fault)**
Untuk Prediksi

**cm = confusion_matrix(y_test, y_pred)**
**sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')**
Menampilkan jumlah benar/salah prediksi: *True Positive, True Negative, False Positive, False Negative.* Visual dengan heatmap untuk memudahkan analisis.

**fpr, tpr, _ = roc_curve(y_test, y_prob)**
**plt.plot(fpr, tpr, label=f"AUC = {roc_auc_score(y_test, y_prob):.2f}")**
ROC Curve menunjukkan trade-off antara TPR dan FPR. AUC (*Area Under Curve*): Nilai antara 0 dan 1, semakin dekat ke 1 berarti model makin bagus.

**print(classification_report(y_test, y_pred))**
**print("MSE:", mean_squared_error(y_test, y_pred))**
Menampilkan metrik: *Precision, Recall, F1-score, Support (jumlah sample per kelas), MSE (Mean Squared Error*): Mengukur rata-rata kesalahan kuadrat antara prediksi dan nilai asli.

```
# Pipeline: normalisasi + model SVM
pipeline = make_pipeline(StandardScaler(), SVC(kernel='rbf'))
```

**pipeline = make_pipeline(StandardScaler(), SVC(kernel='rbf'))**
make_pipeline(...): Membuat alur otomatis dari preprocessing hingga model.
StandardScaler(): Menormalisasi data (mean = 0, std = 1).
SVC(kernel='rbf'): Model klasifikasi SVM dengan RBF kernel (cocok untuk data yang tidak linear).

```
# Evaluasi dengan 5-Fold Cross Validation
cv_scores = cross_val_score(pipeline, X, y, cv=5, scoring='accuracy')
print("Cross-validation scores:", cv_scores)
print("Rata-rata akurasi:", cv_scores.mean())
```

**cv_scores = cross_val_score(pipeline, X, y, cv=5, scoring='accuracy') print("Cross-validation scores:", cv_scores)**

**print("Rata-rata akurasi:", cv_scores.mean())**

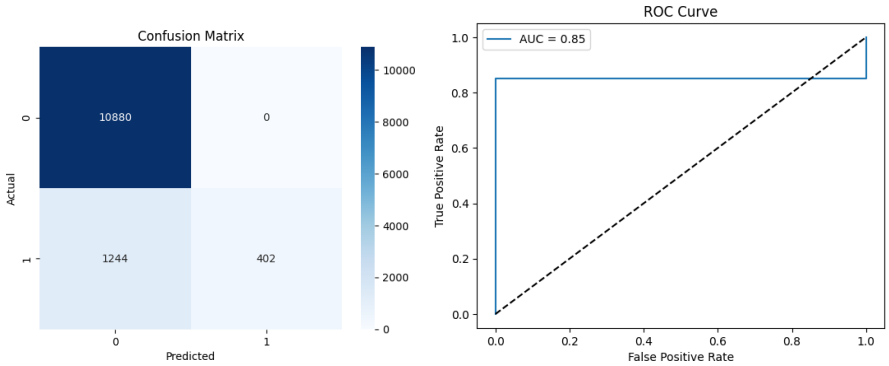
cross_val_score(...): Menilai performa model (*pipeline*) dengan *5-fold cross-validation*.

cv=5: Data dibagi menjadi 5 bagian, model dilatih dan diuji sebanyak 5 kali secara bergantian. scoring='accuracy': Metrik evaluasi yang digunakan adalah akurasi.

## HASIL DAN PEMBAHASAN

Penelitian ini berhasil mengembangkan model klasifikasi gangguan sensor menggunakan metode pembelajaran mesin yaitu *Support Vector Machine* (SVM). Model diuji terhadap dataset pembacaan sensor suhu, di mana data diberi label fault secara otomatis jika nilainya berada di luar rentang normal (10–30). Evaluasi dilakukan berdasarkan metrik: *confusion matrix, ROC-AUC, MSE, dan cross-validation*.

Berikut hasil Evaluasi:




| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.90 | 1.00 | 0.95 | 10880 |
| 1 | 1.00 | 0.24 | 0.39 | 1646 |
| accuracy | | | 0.90 | 12526 |
| macro avg | 0.95 | 0.62 | 0.67 | 12526 |
| weighted avg | 0.91 | 0.90 | 0.87 | 12526 |

MSE: 0.0993134280696152

Gambar 3. Hasil *confusion matrix, ROC-AUC, MSE*

Gambar 3. menujukkan bahwa Model SVM memberikan akurasi sekitar 90%, dengan *Mean Squared Error* (MSE) sebesar 0.0993. Model ini cenderung kuat dalam mendeteksi kelas mayoritas (normal), namun menunjukkan kelemahan dalam *recall* kelas minoritas (*fault*), yaitu hanya sekitar 24%, yang berarti sebagian fault tidak terdeteksi. Hal ini disebabkan oleh ketidakseimbangan data, di mana jumlah data normal jauh lebih besar dari data fault.

```
Cross-validation scores: [0.99984033 0.99984033 0.99992017 0.99992017 0.99992016]
Rata-rata akurasi: 0.9998882312016555
```

Walaupun nilai AUC model SVM cukup tinggi, yaitu >0.90, namun sensitivitas terhadap kelas minoritas masih perlu ditingkatkan.

Gambar 4. Hasil *cross-validation*

Gambar 4. menujukkan hasil output dari *cross-validation* pada sebuah model *machine*, yang menunjukkan akurasi model pada setiap lipatan (*fold*) dan rata-ratanya. Meskipun hasil ini tampak ideal, performa sempurna tersebut menimbulkan indikasi *overfitting* atau kemungkinan dataset terlalu mudah dipelajari karena label dibuat dengan *threshold* yang tegas (*rule-based*). Dalam konteks aplikasi nyata, hasil seperti ini perlu divalidasi lebih lanjut dengan data *real-world* yang lebih kompleks dan *noisy*. Dari sisi tujuan penelitian, hasil ini membuktikan bahwa metode AI seperti SVM dapat diterapkan untuk *fault detection* berbasis

data sensor, tetapi dengan catatan:

1. Distribusi label dan *noise* dalam data sangat memengaruhi performa model.
2. Pemilihan metrik evaluasi yang tepat penting untuk menghindari bias akurasi terhadap kelas dominan.
3. Model perlu diuji lebih lanjut pada data *streaming* atau kondisi sensor riil untuk mengukur kestabilan dalam lingkungan operasional nyata.

## SIMPULAN

Penelitian ini membuktikan bahwa algoritma SVM efektif digunakan untuk mengklasifikasi kondisi normal dan gangguan pada sensor berbasis data suhu, dengan akurasi tinggi dan kemampuan generalisasi yang baik. Meskipun performa terhadap kelas minoritas (*fault*) masih perlu ditingkatkan, pendekatan ini menunjukkan potensi untuk diterapkan dalam sistem monitoring berbasis AI. Penelitian mendatang disarankan untuk mengatasi ketidakseimbangan data dan menguji model pada lingkungan nyata dengan data *streaming* agar sistem dapat beradaptasi terhadap kondisi dinamis dan kompleks di lapangan.

## DAFTAR PUSTAKA

[1] M. A. B. Siddique et al., "Anomaly detection in sensor data: A survey," *Sensors*, vol. 25, no. 1, p. 60, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/25/1/60

[2] F. Li, C. Xu, and J. Chen, "Intelligent fault detection in sensor networks using KNN and PCA," *Sensors*, vol. 22, no. 5, p. 1876, 2022. [Online]. Available: https://doi.org/10.3390/s22051876

[3] L. Zhao, Y. Sun, and W. Huang, "Real-time anomaly detection in IoT sensor data using machine learning," *J. Intell. Fuzzy Syst.*, vol. 45, no. 2, pp. 1231–1240, 2023. [Online]. Available: https://doi.org/10.3233/JIFS-221102

[4] T. Chen, K. Zhang, and Y. Gao, "On interpretability and performance of AI-based fault detection models," *Appl. Soft Comput.*, vol. 94, p. 106435, 2020. [Online]. Available: https://doi.org/10.1016/j.asoc.2020.106435

[5] Y. Wang, X. Liu, and H. Zhang, "A hybrid SVM model for fault diagnosis of industrial sensors," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–10, 2021. [Online]. Available: https://doi.org/10.1109/TIM.2021.3056274

[6] A. Sharma and D. Kumar, "A survey on support vector machine and its variants for classification problems in bioinformatics," *Appl. Soft Comput.*, vol. 97, p. 106748, 2020. [Online]. Available: https://doi.org/10.1016/j.asoc.2020.106748

[7] S. Basak, S. Mondal, and N. Dey, "Support Vector Machines for Classification," in *Machine Learning Techniques and Analytics for Cloud Security*, N. Dey and A. S. Ashour, Eds. Singapore: Springer, 2021, pp. 35–52. [Online]. Available: https://doi.org/10.1007/978-981-16-1427-4_3